Diplomarbeit

Konzeption und Implementierung eines Prototypen für positions- und situationsabhängige Informationsdienste am Beispiel eines pervasiven Adventurespiels

vorgelegt von
Eileen Kühn

Betreuer:

Prof. Dr. Jürgen Sieck
Prof. Dr. Elke Naumann

Studiengang Angewandte Informatik
Fachbereich Wirtschaftswissenschaften II
Fachhochschule für Technik und Wirtschaft Berlin

Berlin, im November 2008

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation

Der Boom in der Spieleindustrie ist ungebrochen. Allein das Online-Rollenspiel *World of Warcraft* erreichte Ende Oktober 2008 eine Spielerzahl von 11 Millionen [15]. Der Spieler bewegt seinen Charakter meist durch eine dreidimensionale Fantasiewelt – durch verschiedene Städte und Dörfer, durch Wälder und Wüsten bis hin zu Dschungeln und speziellen Dungeons. Er sammelt dort Erfahrungspunkte und Belohnungen in Form von virtuellem Geld, Ausrüstungsgegenständen oder anderen Items. Hierbei ist das Medium Computer, als auch der Stil der Interaktion durch die Steuerung der Figur vom Programm vorgegeben. Alles in Allem ist die Trennung der Welten innerhalb des Spiels und der realen Welt zu jeder Zeit gegeben.

Pervasive Spiele dahingegen zielen auf eine neue Art des Spielens ab. Sie integrieren Informations- und Kommunikationstechnik, um die Grenzen herkömmlicher Spiele aufzubrechen und erweiterte Spielwelten zu schaffen. Mit Hilfe von mobilen Endgeräten und deren spezifischen Eigenschaften soll somit eine neue Art von Spielerlebnis geschaffen werden. Die Nutzung von ortsabhängigen Diensten und Umgebungsinformationen der Benutzer wird zu einem entscheidenden Merkmal des Spiels. Die Schlüsseltechnologien dafür finden sich in den Bereichen der drahtlosen Kommunikation und der Positionsbestimmung von Benutzern bzw. mobilen Geräten. Im Gegensatz zu klassischen Computerspielen, die in der virtuellen Welt stattfinden, sind bei pervasiven Spielen beispielsweise physikalische Bewegungen und soziale Interaktionen mit anderen Benutzern erforderlich. Der Spieler nutzt die physische Welt als Spielfeld und kann dennoch die Vorteile und Möglichkeiten der technischen Geräte und der virtuellen Welt nutzen.

Auf diese Weise können gänzlich neue Spielkonzepte realisiert werden. Aber auch eine Er-

weiterung tradtionioneller Spiele um Komponenten des Pervasive Computing ist denkbar.

Solche Spiele eröffnen ein breites Feld von Anwendungen. Es können neue Lernmethoden im Schulunterricht oder die unterhaltsame Vermittlung kultureller Aspekte angestrebt werden. Die Konzeption dessen gestaltet sich jedoch problematisch und wirft Fragen auf, wie die Schlüsseltechnologien in das Spiel zu integrieren sind.

1.2 Zielsetzung

Im Rahmen dieser Arbeit soll eine verteilte Anwendung entworfen und prototypisch implementiert werden, die es ermöglicht anhand unterschiedlicher Sensordaten eines mobilen Endgerätes die Situation und Position des Benutzers zu bestimmen. In Abhängigkeit dieser Daten sollen von einer zentralen Instanz die Daten zur Anreicherung des Spiels abgerufen werden. Auf diese Weise sollen dem Spieler Elemente der Story, neue Aufgaben oder Informationen über andere Spieler näher gebracht werden.

Ziel dieser Arbeit wird es sein, die Vorgehensweise bei der Positions- und Situationsbestimmung im Detail auszuarbeiten und die unterschiedlichen Technologien und Algorithmen zu untersuchen und den Anforderungen entsprechend auszuwählen. Diese werden schließlich in einem Software-Entwurf zu einem Gesamtsystem kombiniert, das alle Anforderungen an ein unterhaltsames Spiel für Pervasive Gaming erfüllt.

Anhand einer prototypischen Umsetzung soll die Machbarkeit sowie Genauigkeit der Positionsbestimmung bei Nutzung des Spiels überprüft und die Grenzen des zugrunde liegenden Verfahrens aufgezeigt werden.

1.3 Aufbau der Arbeit

In dieser Arbeit werden theoretische Grundlagen und gegebene Randbedingungen beleuchtet. Zunächst werden in Abschnitt 2.1 die Konzepte der Positionsbestimmung und einsetzbare Technologien beschrieben. Anschließend werden in Abschnitt 2.2 die Grundlagen des Ubiquitous und Pervasive Computing und die damit verbundenen Konflikte in den Bereichen Sicherheit und Datenschutz beleuchtet. In Abschnitt 2.3 wird auf die Notwendigkeit der Kontextsensitivität eingegangen. Und schließlich wird in Abschnitt 2.4 auf die aktuellen Trends und mögliche Endgeräte für die Umsetzung des Systems eingegangen.

Gegenstand der Diskussion in Kapitel 3 ist die Beleuchtung bereits vorhandener pervasiver Spiele und die Identifizierung vorhandener und fehlender Funktionalitäten. Die Analyse wird außerdem mögliche Anwendungsfälle aufzeigen, um die Komponenten auf

konzeptioneller Ebene zu erarbeiten. Aufbauend auf der Analyse wird das umzusetzende Spielkonzept erarbeitet und für den Prototypen definiert.

Der Systementwurf in Kapitel 4 wird die Ergebnisse der vorangegangenen Untersuchungen in einem Software-System zusammenführen. Zudem wird die Datenhaltung modelliert, die Architektur in Komponenten zerlegt und deren Funktionsumfang festgelegt.

Die Implementierung der Anwendung behandelt die Schritte, Probleme und Lösungen der Umsetzung. Außerdem wird auf die eingesetzten Implementierungs-Technologien eingegangen.

Eine Demonstration zeigt die zuvor in der Anforderungsdefinition erstellte Funktionalität der Anwendung. Außerdem testet eine Evaluation unter verschiedenen Gesichtspunkten die Flexibilität und Korrektheit der Anwendung.

Abschließend wird eine Zusammenfassung über den Verlauf der Arbeit und der erzielten Ergebnisse sowie ein Ausblick auf mögliche Erweiterungen des Systems aufgezeigt.

Kapitel 2

Grundlagen und Voraussetzungen

Mit diesem Kapitel werden die Grundlagen zur Umsetzung der wesentlichen Aspekte für positions- und situationsabhängige Informationsdienste gelegt.

Um dem gerecht zu werden, werden im ersten Schritt die technischen Grundlagen und Technologien zur Positionsbestimmung diskutieren.

Anschließend werden die Grundlagen des Ubiquitous und Pervasive Computing besprochen, bevor auf die Bedeutung des Kontexts und der Kontextsensitivität im Sinne der Aufgabenstellung eingegangen wird.

Abschließend werden Trends im Bereich der mobilen Endgeräte aufgezeigt und ausgewählte Geräte vorgestellt, um die zur Umsetzung des Adventurespiels benötigte Zielplattform zu identifzieren.

2.1 Positionsbestimmung

Sei es, klassischerweise, in der Seefahrt, in der Luft- und Raumfahrt, im Verkehrs- und Vermessungswesen oder aber bei der Orientierung im Outdoor-Bereich, der Positionsbestimmung und Ortung wird von jeher eine wichtige Rolle zuteil. In Abhängigkeit des Anwendungsbereichs müssen dabei unterschiedliche Anforderungen hinsichtlich Verfügbarkeit, Gebietsabdeckung, Genauigkeit und Echtzeitfähigkeit erfüllt werden. Die dabei verwendeten Methoden und Konzepte sind so vielfältig, wie deren Einsatzgebiete, deshalb sollen zunächst grundlegende Begriffe und Funktionsweisen zum besseren Verständnis erläutert werden, bevor auf einige ausgewählte Technologien und Anwendungen näher eingegangen wird.

2.1.1 Theoretische Grundlagen

Bevor die Technologien und Anwendungen zur Positionsbestimmung konkretisiert werden können, sollen zuvor die technischen Grundlagen vorgestellt werden.

2.1.1.1 Ortung vs. Positionsbestimmung

Trotzdem die beiden Begriffe *Ortung* und *Positionsbestimmung* gern synonym füreinander genutzt werden, verfolgen beide einen unterschiedlichen Ansatz. Bei der Ortung geht es um die Bestimmung der Position eines entfernten Objektes oder einer Person. Bei einer reinen Positionsbestimmung dahingegen, handelt es sich noch nicht um eine Ortung. Erst wenn die ermittelte Position beispielsweise über Mobilfunk übertragen wird, ist auch eine Ortung möglich.

In Abhängigkeit des benutzten Verfahrens kann die Standortbestimmung *kooperativ* oder *autonom* erfolgen [Man04]. Wird der Vorgang selbständig von dem ortenden Objekt durchgeführt, dann handelt es sich um eine autonome Vorgehensweise. Ist jedoch die aktive Mitwirkung technischer Einrichtungen erforderlich, handelt es sich um ein kooperatives Verfahren.

Das Ergebnis dieser Verfahren ist die Angabe der Position in einem meist dreidimensionalen Kontext bezüglich eines für den jeweiligen Zweck geeigneten Koordinatensystems - geographische Länge, geographische Breite und Höhe (vgl. [Man04, Bau02]).

2.1.1.2 Referenzsysteme und Koordinaten

Diese geographischen Koordinaten können beispielsweise für einen Nautiker auf See, der mit Hilfe von Peilungen eines Funkfeuers seinen Standort bestimmen will, von enormer Bedeutung sein. Die Kenntnis der Position dieses Bezugspunkts bildet die Grundlage der zur Ortung erforderlichen Berechnungen – der Bestimmung der Koordinaten eines Objektes oder eines Punktes.

Das gilt grundsätzlich für jedes Ortungsverfahren. Deshalb ist eine Übereinstimmung der Bezugssysteme auf konzeptioneller Seite in geometrischer als auch thematischer Modellbildung von zentraler Bedeutung [Vos95]. Dass es sich bei der Entwicklung eines solchen einheitlichen Standards um einen langwierigen, iterativen Vorgang handelte, lässt sich in Anbetracht der Tatsache, dass es sich bei der Erdoberfläche um eine komplexe, mathematisch nicht eindeutig beschreibbare Fläche handelt, bereits erahnen.

Ellipsoide, Geoide und topographische Oberflächen Die Frage nach der Figur des Erdkörpers ist die Frage nach einem mathematischen Modell, mit dem diese Figur beschrieben werden kann. Bei sehr kleinen Entfernungen (weniger als $10km$) kann ein flaches Modell verwendet werden [18]. Regionale Messungen dahingegen sollten zumindest auf dem Modell einer Kugel oder eines Ellipsoids basieren, um gravierende Fehler bei den Berechnungen zu vermeiden.

Zur weiteren Verfeinerung des Modells können zusätzlich noch die Unterschiede zwischen der tatsächlichen Meeresoberfläche und dem Ellipsoid herangezogen werden. Man erhält auf diese Weise einen *Geoid* zur Beschreibung der Erde, einen von der tatsächlichen Erdgestalt abweichenden theoretischen Körper, dessen Oberfläche die Feldlinien der Schwerkraft überall im rechten Winkel schneidet. Mit der Hilfe von mehreren Verfeinerungsstufen gelangt man auf diese Weise von der topografischen Oberfläche der Erde – unter Berücksichtigung aller Hügel, Berge und Täler – zu einem beschreibenden Geoid [Bau02]. In Abbildung 2.1 wird das so entstandene Geoid illustriert.

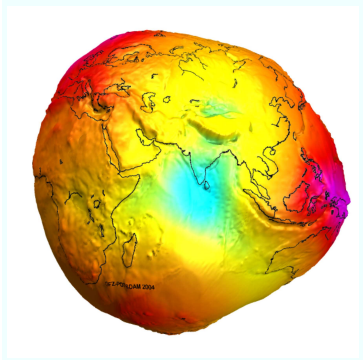


Abbildung 2.1: Das Geoid als Annäherung an die Erdoberfläche (Quelle: [25])

Das Ellipsoid als Vereinfachungsstufe des Geoiden versucht diesen nun möglichst gut zu beschreiben, da Berechnungen auf Grundlage des Geoiden schlecht handhabbar sind [Zog06].

In der nebenstehenden zweidimensionalen Abbildung 2.2 wird mit Ellipsen anstelle von Ellipsoiden visualisiert, dass das nicht immer einfach ist. Somit hat sich im Laufe der Zeit jedes Land sein eigenes bestangepasstes nicht-geozentrisches Rotationsellipsoid als Bezugsfläche für Vermessungsaufgaben entwickelt.

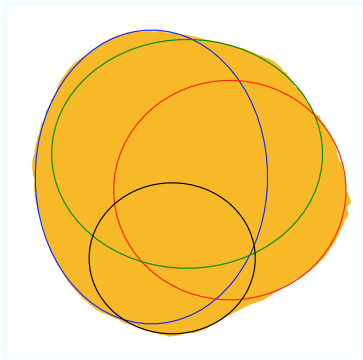


Abbildung 2.2: Unterschiedliche Ellipsoide passen in unterschiedliche Gebiete der Erde

Die Tatsache, dass Ellipsoide immer nur für kleine Gebiete sehr gut passen, ist der Grund dafür, dass es heute so viele unterschiedliche Referenzellipsoide und damit auch Kartenbezugssysteme gibt. Darin besteht ein wesentliches Problem bei der Verwendung von Ortungsverfahren: es existieren weltweit unterschiedliche Koordinatensysteme, weshalb die gemessene und berechnete Position oftmals nicht mit der vermeintlichen Position übereinstimmen.

Um ein global einsetzbares Ortungsverfahren zu realisieren, wird deshalb üblicherweise auf

das so genannte *Geodätische Weltsystem 1984 (WGS 84)* [IN00] zurückgegriffen. Dieses wird mit einem Satz von neun Parametern festgelegt, anhand derer sich die Erdoberfläche im Mittel als Rotationsellipsoid approximieren lässt – dem so genannten *WGS 84 Referenz-Ellipsoiden.*

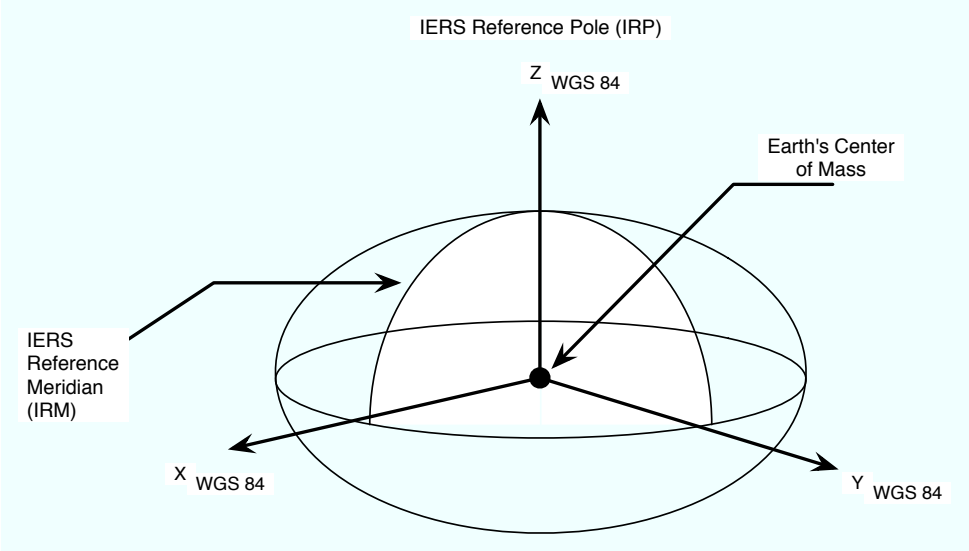


Abbildung 2.3: Definition des WGS 84 Referenz-Ellipsoiden (Quelle: [IN00])

2.1.1.3 Prinzip der Positionsbestimmung

Prinzipiell kann bei der Bestimmung der Position auf zwei unterschiedliche Verfahren zurückgegriffen werden: der *Trilateration* und der *Triangulation.*

Bei der Triangulation werden mindestens zwei Punkte mit bekannter Position benötigt. Ist diese Voraussetzung erfüllt, kann eine Positionsbestimmung anhand der Messung der Winkel erfolgen. Um bei dieser Methode auf dreidimensionale Daten zugreifen zu können, werden mindestens drei Winkel benötigt.

Die zweite Möglichkeit ist die Trilateration. Diese basiert, im Gegensatz zur Triangulation, auf der Distanzmessung zu mindestens drei Referenzpunkten. Der entscheidende Vorteil dieser Methode besteht darin, dass der technische Aufwand geringer ist. Beim Vorhandensein von mehr als drei Referenzpunkten, kann auf Multilateration zurückgegriffen werden, um die Genauigkeit der Messung entscheidend zu erhöhen. Um auch hier auf dreidimensionale Daten zugreifen zu können, werden mindestens vier Entfernungen benötigt.

Grundlage der Umsetzung dieser Methoden zur Positionsbestimmung bildet die Auswertung physikalischer Eigenschaften emittierter Signale am Empfänger des Ortungssystems, beispielsweise der Impulslaufzeit, Feldstärke oder dem Phasenwinkel von Schwingungen

(vgl. [Man04]). Dieser Auswertung wird die Annahme zugrunde gelegt, dass sich die gemessene Größe proportional zur Entfernung von Sender und Empfänger verhält, sich das Signal demnach geradlinig und ungestört ausbreitet.

Alle Punkte, an denen eine solche Größe den gleichen Betrag hat, liegen in der Ebene auf einem Kreis. Im Raum spannen sie eine Fläche auf – diese ist durch eine geometrische Größe, die der gemessenen proportional ist, definiert, beispielsweise durch die Oberfläche einer Ebene, einer Kugel oder eines Rotationshyperboloids. Auf dieser *Standlinie* oder *Standfläche* befindet sich der Empfänger. Abbildung 2.4 illustriert dies für den zweidimensionalen Fall.

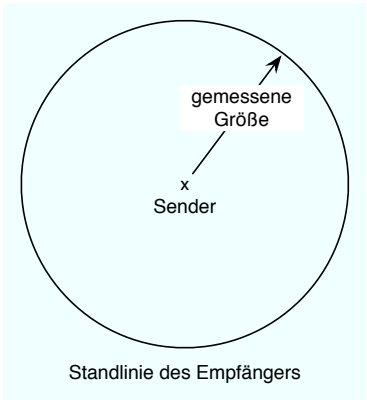


Abbildung 2.4: Standlinie des Empfängers in der Ebene

Um die unbekannte Position P eines Empfängers mit Hilfe eines kooperativen Ortungssystems zu bestimmen, müssen die Signale unterschiedlicher Sender S_1, S_2, S_3 ausgewertet werden. Vorausgesetzt wird, dass die Positionen der Sender im Raum bekannt sind.
Jede Einzelmessung liefert dann eine Standfläche in Abhängigkeit der Art des gemessenen Signals. In den folgenden Überlegungen wird daher exemplarisch die Messung der Laufzeit zu Grunde gelegt, was in der spezifischen Standfläche einer Kugel resultiert.

Die Kombination zweier Einzelmessungen ermöglicht dem Empfänger bereits eine Fixierung der Position auf einer Standlinie: die kreisförmige Schnittfläche der beiden Standflächen. Wenn bei diesen Betrachtungen zusätzlich die Messdaten eines dritten Signals S_3 berücksichtigt werden, so erhält man zwei weitere Standlinien. Der Schnittpunkt der drei Standlinien fixiert den gesuchten Standpunkt des Empfängers.

Die aus drei Messungen ermittelte Position P des Empfängers ist nicht zwangsläufig eindeutig. Es kann ein zusätzlicher *Scheinstandpunkt* P^* auftreten. Dies illustriert Abbildung 2.5. Erst eine vierte Messung liefert den eindeutigen Standort. Befindet sich der zu ortende Empfänger jedoch auf der Erdoberfläche oder auf einer Horizontalebene mit konstantem Abstand zur Erdoberfläche, so ist damit bereits eine vierte Standfläche gegeben und der Standpunkt kann eindeutig identifiziert werden.

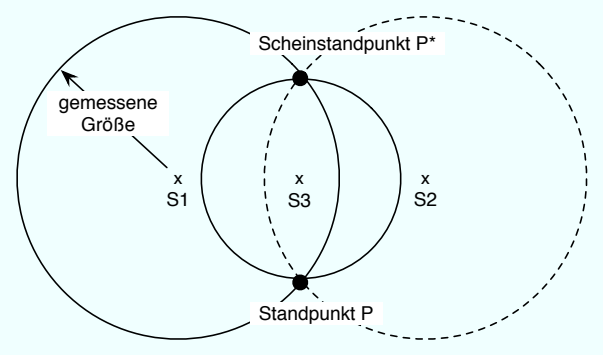


Abbildung 2.5: Visualisierung des Scheinstandpunktes P*

Bei den bisherigen Ausführungen wurde zudem vorausgesetzt, dass die Laufzeiten korrekt ermittelt wurden. Dies ist aber nur der Fall, wenn hochpräzise und synchronisierte Uhren bei der Messung verwendet würden. Fehlende Uhrensynchronisierung führt zu fehlerhaften Entfernungen (*Pseudostrecken*) und infolgedessen zu fehlerhaften Positionsbestimmungen. Bereits eine Verfälschung der Laufzeitmessung von nur $1\mu s$ verursacht einen Positionsfehler von $300m$.

Um diesen Fehler zu reduzieren, wird daher das Signal eines weiteren Senders S_4 benötigt.

2.1.1.4 Techniken

Bei den hier aufgelisteten Möglichkeiten zur Positionsbestimmung (siehe Abbildung 2.6) handelt es sich nicht um eine vollständige Übersicht dessen, was technisch realisierbar ist. Es existieren weitere Verfahren, wie beispielsweise die visuelle Positionsbestimmung [Rot05]. Deren Funktionsweise basiert auf der Untersuchung der Umgebung zur Identifizierung charakteristischer Merkmale. Diese werden mit im Vorfeld gespeicherten Referenzdaten verglichen, um eine Positionsbestimmung zu ermöglichen.

Messung der Laufzeit - Time of Arrival *Time of Arrival* (ToA) bezeichnet ein Verfahren mit dessen Hilfe die Entfernung anhand der Laufzeit eines Signals bestimmt wird. Dabei macht man sich die physikalische Eigenschaft elektromagnetischer Wellen zu nutze – die endliche Ausbreitungsgeschwindigkeit eines Signals. Aus der Zeitdifferenz des Sende- und Empfangszeitpunktes zu mindestens zwei Sendestationen – der Laufzeit – kann deshalb auf die Entfernung zwischen Sender und Empfänger geschlossen werden.

In der Regel handelt es sich bei diesem Verfahren um eine Einwegmessung, d.h. das Signal wird nicht reflektiert. Voraussetzung für eine entsprechende Genauigkeit sind jedoch synchronisierte Uhren. Eine Alternative ist der Einsatz eines weiteren Senders, um die Zeitdifferenz zwischen Sender- und Empfängeruhr herauszufinden (vgl. Abschnitt 2.1.1.3,

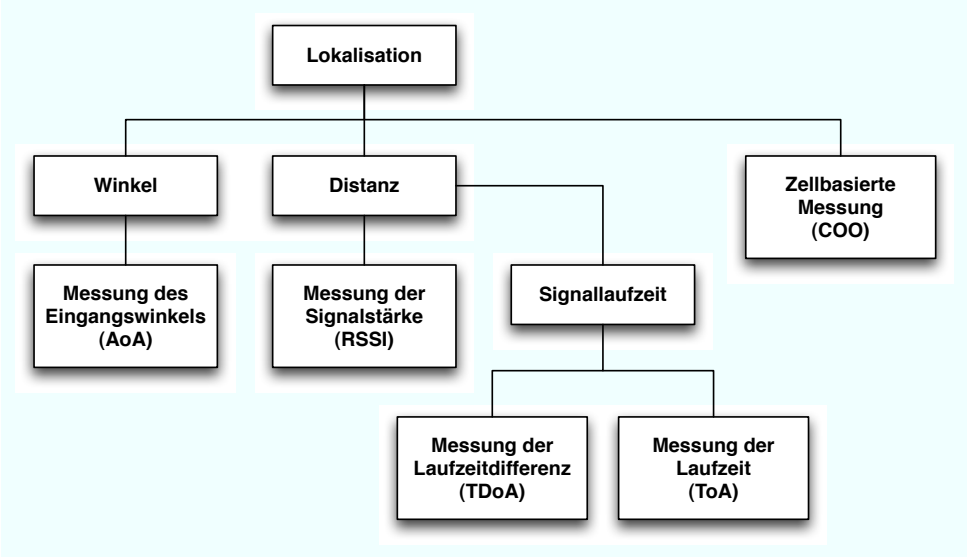


Abbildung 2.6: Übersicht der Techniken zur Bestimmung der Position

S. 9).

Die Synchronisation der Sender untereinander ist jedoch aufwändig und vor allem teuer. Deshalb werden beispielsweise beim GPS hochpräzise Atomuhren eingesetzt. Alternativ können auch pro Sender gleichzeitig zwei Signale mit unterschiedlichen Ausbreitungsgeschwindigkeiten übermittelt werden[1].

Auch die Durchführung einer Zweiweg-Messung ist möglich. In diesem Fall findet eine Reflexion des Signals beim Empfänger (Echo) statt. Deshalb ist keine Synchronisation von Nöten und die Distanzmessung ist trivial.

Laufzeitdifferenz - Time Difference of Arrival Im Gegensatz zum Verfahren ToA benötigt die mobile Station, die geortet werden soll, bei dem Verfahren zur Bestimmung der Laufzeitdifferenz (englisch *Time Difference of Arrival / TDoA*), keine eigene Uhr, da hier die Synchronität der Referenzstationen den Berechnungen zu Grunde gelegt wird. Zwei Referenzstationen senden gleichzeitig Signale aus. Um die Genauigkeit dessen zu gewährleisten, müssen die Referenzstationen untereinander synchronisiert sein. Der Empfänger misst die Zeitdifferenz des Empfangs der Signale beider Sender. Aus den Positionsangaben der Sender sowie der berechneten Zeitdifferenzen lässt sich nun eine hyperbolische Kurve berechnen, die die Standlinie des Empfängers kennzeichnet. Durch weitere Messungen zu anderen Referenzpaaren kann eine eindeutige Bestimmung des Standpunktes sichergestellt werden.

[1]Die Entfernung zu einem Blitzschlag kann beispielsweise durch die Messung zweier unterschiedlicher Signale gemessen werden: diese basieren auf der Differenz der Ankunftszeit von Licht- und Schallsignalen.

Die Berechnung der Position kann auch seitens der Referenzstationen ermöglicht werden (*remote positioning*). Mindestens drei Basisstationen fügen dem vom mobilen Gerät ausgesendetem Signal einen Zeitstempel hinzu und leiten diese Information zur Berechnung der Position weiter. Aus den Differenzen der jeweiligen Zeitstempel kann die Position des mobilen Gerätes nun mit Hilfe des Prinzips der Trilateration (vgl. Abschnitt 2.1.1.3, S. 7) berechnet werden.

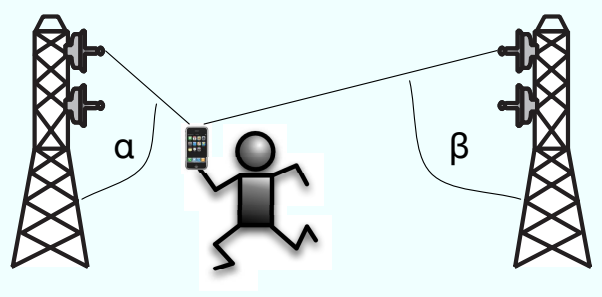


Abbildung 2.7: Prinzip des Angle of Arrival

Angle of Arrival Das Prinzip *Angle of Arrival* (AoA) beschäftigt sich mit der Bestimmung des Winkels des empfangenen Signals durch mindestens zwei Stationen. Skizziert wird dies in Abbildung 2.7.

Durch die Nutzung von Antennen mit Richtungscharakteristik kann dieses Verfahren ermöglicht werden. Dabei werden Antennen in einem bestimmten Winkelabstand in alle Richtungen aufgebaut (*Antennen-Arrays*). Dieser Aufbau ermöglicht die Messung des Eingangswinkels des Signals. Diese Winkelinformationen können mit Hilfe der Triangulation für die Positionsbestimmung genutzt werden. Auch bei diesem Verfahren können eventuelle Messabweichungen durch die Messung von mehreren Basisstationen reduziert werden.

Messung der Signalintensität - Received Signal Strength Indicator Das Verfahren der Messung der Signalintensität (englisch *Received Signal Strength Indicator / RSSI*) beruht auf dem Effekt der Freiraumdämpfung: wird von einem Sender – zur Vereinfachung der Betrachtungen von einem isotropen Kugelstrahler[2] – hochfrequente Energie P gleichmäßig in alle Richtungen abgestrahlt, bilden sich kugelförmige Flächen gleicher Leistungsdichte um den Sender. Je größer der Radius r der Kugel, desto größer ist auch die Kugeloberfläche $A_{Kugel} = 4\pi r^2$ auf die sich die Energie des Senders verteilt. Bezogen auf

[2]Ein isotroper Kugelstrahler ist eine theoretische Idealisierung eines Punktstrahlers und dient dem Vergleich mit realen Antennen. Er strahlt gleichmäßig in alle Richtungen und resultiert somit in einer kugelförmigen Leistungsverteilung.

eine angenommene gleichgroße Fläche verhält sich die Leistungsdichte somit quadratisch proportional zur Entfernung des Senders und wird mit steigendem Abstand infolgedessen geringer.

Dieses Verfahren ist allerdings besonders anfällig gegenüber Hindernissen und Abschattungseffekten und gilt deshalb nur im freien Raum: außer der Freiraumdämpfung treten in der Erdtmosphäre auch weitere Dämpfungen auf. Das heißt, die reale Dämpfung nimmt sehr viel größere Werte an, als durch die Freiraumdämpfung berechnet wird. Die gleichzeitige Messung der Signalstärken verschiedener Sender kann jedoch zur Verbesserung der Qualität der Ergebnisse ausgenutzt werden: bei günstiger Wahl der Anzahl der Sender und des Standorts, kann erreicht werden, dass die Signalstärkemessung an jedem Ort im relevanten Gebiet zu einem typischen „Fingerabdruck“ führt, der dank Vergleich mit einer Referenzkarte eine Positionsbestimmung erlaubt. Solche eine Referenzkarte muss jedoch im Vorfeld der Nutzung durch Stichproben-Messungen kartographiert werden.

Cell of Origin - Cell-ID Bei dem Verfahren *Cell of Origin* (COO) wird vorausgesetzt, dass die Infrastruktur der Sendestationen eine Zellstruktur aufweist. Da ausgesendete Signale wie Radiowellen, Infrarot oder Ultraschall eine begrenzte Reichweite haben, kann ein für die Positionierung relevantes Gebiet in Zellen aufgeteilt werden. Durch Identifizierung der aktuellen Zelle, in deren Empfangsbereich sich das Empfangsgerät befindet, lässt sich die aktuelle Position bestimmen. Diese Positionsbestimmung ist zweifelsohne recht ungenau: die Positionsgenauigkeit entspricht im ungünstigsten Fall dem Radius des Sendebereichs. Dieser kann im Bereich des Mobilfunk $200m$ bis zu $35km$ betragen. Ein großer Vorteil des Verfahren besteht darin, dass es sich um eine unkomplizierte Technik handelt. Die Grundvoraussetzung besteht allerdings darin, dass eine flächendeckende Zellstruktur vorhanden sein muss.

Heutzutage findet das COO-Verfahren Anwendung in einigen Location Based Services von Betreibern der Handynetzwerke. Da Sendemasten in ländlichen Gebieten allerdings eine sehr viel größere Fläche abdecken müssen als es in Städten der Fall ist, gibt es Ortungsgenauigkeiten von mehreren Kilometern.

2.1.2 Technologien und Anwendungen

Die Positionsbestimmung ist eine der Schlüsseltechnologien für positionsabhängige Dienste. Satellitengestützte Systeme und terrestrische Ortung in GSM-Mobilfunknetzen haben bislang diesen Bereich geprägt. Unter spezifischen Anforderungen lassen diese Techniken allerdings einige Wünsche offen. Erwünscht werden im Rahmen dieser Arbeit vor allem Systeme, die gleichzeitig Outdoor- und Indoor-Ortung bei geringen Kosten, hoher Genauigkeit, universeller Einsatzfähigkeit und problemloser Interoperabilität ermöglichen, ohne Änderungen an der vorhandenen Infrastruktur vornehmen zu müssen.

Anhand dieser Kriterien sollen mögliche Verfahren im Bereich der satelliten- und netzwerkbasierten Lokalisierung, der Lokalisierung innerhalb von Gebäuden als auch hybride Techniken nachfolgend vorgestellt und bewertet werden.

2.1.2.1 Satellitenbasierte Lokalisierung

Die Idee satellitenbasierte Systeme einzusetzen geht bis in die 80-er Jahre zurück. Diese Verfahren haben gegenüber anderen einen entscheidenden Vorteil: die Positionsbestimmung kann fast überall auf der Erde mit einer hohen Genauigkeit erfolgen. Umwelteinflüsse haben darauf kaum einen Einfluss. Allerdings gibt es in Städten in der Nähe von Häusern Abschattungseffekte, die das Ergebnis verfälschen können. Zudem muss eine Sichtverbindung zu mindestens zwei bis drei Satelliten bestehen. Zur Installation und Überwachung eines solchen Systems sind erhebliche Kosten von Nöten. Um eines der Systeme zu nutzen, benötigen mobile Komponenten einen Empfänger, um die Signale entsprechend auswerten zu können.

Neben dem vom amerikanischen Militär betriebenen Global Positioning System (GPS) existiert das ebenfalls militärisch betriebene russische GLONASS-System. Auch mit dem europäischen GALILEO-Projekt soll frühestens 2010 das erste zivil betriebene System zur Verfügung stehen.

Da das grundlegende Funktionsprinzip der drei Systeme identisch ist, wird nachfolgend auf eines der Systeme genauer eingegangen: das GPS.

GPS Bei dem *GPS* – die vollständige Bezeichnung lautet Navigation System with Timing And Ranging Global Positioning System, NAVSTAR-GPS – handelt es sich um ein vom amerikanischen Verteidigungsministerium entwickeltes satellitengestütztes Verfahren zur weltweiten Positionsbestimmung. Es besteht aus drei Segmenten: dem *Weltraum-*, *Kontroll-* und *Benutzersegment*.

Das Weltraumsegment umfasst derzeit 29 aktive Satelliten, die die Erde auf 6 Bahnen, welche um 55° gegen die Äquatorebene inkliniert sind, umkreisen. Seit der Inbetriebnahme des Systems im Jahre 1978 senden diese in 20.180km Höhe fliegenden GPS-Satelliten auf zwei Frequenzen (L1 = 1575,42 MHz und L2 = 1227,60 MHz) drei durch das Code Division Multiple Access (CDMA)[3] Verfahren kodierte Signale mit einer Taktrate von 50 Bit/s aus. Übertragen werden unter anderem präzise Bahndaten der Satelliten (Ephemeriden), ungenauere Bahndaten aller Satelliten (Almanach) sowie der C/A-Code für die zivile Nutzung oder das verschlüsselte P(Y)-Signal, dass der Nutzung für militärische Anwendungen vorbehalten ist.

Bei dem Kontrollsegment handelt es sich um alle zur Überwachung des Systems dienenden Bodenstationen: ein Hauptquartier, fünf mit Atomuhren ausgerüstete Monitorstationen, welche weltweit in der Nähe des Äquators verteilt sind und drei zusätzliche Bodenstationen, dessen Aufgabe die Übermittlung von Informationen an die Satelliten ist. Die Aufgaben des Kontrollsegments umfassen die Verfolgung der Satelliten, die Aktualisierung ihrer Umlaufpositionen sowie die Kalibrierung und Synchronisierung ihrer Uhren. Außerdem wird ebenfalls die künstliche Verfälschung der Signale (Selective Availability) gesteuert, um die Positionsgenauigkeit für zivile Anwender herabzusetzen.

Alle zivilen und militärischen Anwender die mit einem GPS-Empfänger ausgestattet sind, bilden das Benutzersegment dieses Systems.

Bei der Positionsbestimmung spielt der durch die Satelliten ausgesandte C/A-Code eine wichtige Rolle: er enthält die Identifikation und die Informationen eines jeden Satelliten. Anhand der einmaligen Struktur dieses Codes kann der Empfänger bestimmen, von welchem Satelliten er ausgesendet wurde. Aus der Laufzeit der Signale wird nun der Abstand bestimmt: dieser ergibt sich aus dem Abstand und der bekannten Position von vier Satelliten. Anhand dessen errechnet der Empfänger des Anwenders seine Länge, Breite, Höhe und Zeit.

Verschiedene Ursachen können bei dieser Berechnung allerdings zum Gesamtfehler beitragen:

- Obwohl jeder Satellit vier Atomuhren mit sich führt, bewirkt ein Zeitfehler von nur 10 ns bereits einen Fehler in der Größenordnung von 3 m.
- Die Signale vom Satelliten zum Anwender breiten sich mit Lichtgeschwindigkeit aus.

[3]Beim Code Division Multiple Access handelt es sich um ein Verfahren, dass mehreren Nutzern den Zugriff auf einen Übertragungskanal ermöglicht. Die Nutzsignale werden durch eine Spreizung des Nutzdatenkanals unterschiedlich codiert. Der Übertragungskanal kann deshalb gleichzeitig für mehrere Nutzkanäle genutzt werden.

Diese verlangsamt sich beim Durchqueren von Ionosphäre und Troposphäre und darf somit nicht mehr als konstant angenommen werden.

- Auch die Messung der Laufzeit kann fehlerhaft sein: diese kann durch den Empfänger nur mit einer beschränkten Genauigkeit bestimmt werden.
- Durch terrestrische Reflexionen (Multipath) wird der Fehleranteil noch erhöht.

DGPS Durch den Vergleich mit einer oder mehrerer Referenzstationen können viele Fehlerquellen eliminiert werden. Die Auswertung der auf beiden Stationen vorliegenden Beobachtungen kann entweder im Postprocessing erfolgen oder in Echtzeit, was durch den Zusatz RT (Real-Time) gekennzeichnet wird. Echtzeitlösungen (RT-DGPS) setzen eine Datenkommunikation von der Referenzstation zum mobilen Empfänger voraus.

Das Prinzip des DGPS basiert auf der Laufzeitmessung und ist sehr einfach. Es erfolgt in drei Phasen:

- Bestimmung der Korrekturgrößen bei der Referenzstation
- Übermittlung der Korrekturgrößen von der Referenzstation zum GPS-Anwender
- Korrektur der gemessenen Pseudostrecke beim GPS-Anwender

Eine Referenzstation mit genau vermessenen Koordinaten misst die Laufzeit zu allen sichtbaren GPS-Satelliten, mindestens vier, und bestimmt aus dieser Größe die fehlerbehafteten Pseudostrecken. Da die GPS-Referenzstation ihre genaue Position kennt, kann sie die Abweichung von der gemessenen Position bzw. der gemessenen Pseudoranges der einzelnen Satelliten berechnen. Die Differenz zwischen wahrer Distanz und fehlerbehafteter Pseudostrecke lässt sich durch einfache Subtraktion ermitteln und entspricht einer Korrekturgröße. Diese Korrekturgrößen gelten ebenfalls für alle im Bereich bis zu $200km$ vorhandenen GPS-Empfänger um die Referenzstation.

Da die Korrekturgrößen in einem weiten Umkreis zur Korrektur der gemessenen Pseudostrecken verwendet werden können, werden sie über ein geeignetes Medium (Funk, Telefon, Radio, ...) weiteren GPS-Anwendern ohne Zeitverzug übermittelt.

Der GPS-Anwender kann nach Empfang der Korrekturwerte die wahre Distanz aus seinen gemessenen fehlerbehafteten Pseudostrecken ermitteln. Aus der wahren Distanz lässt sich die genaue Anwenderposition berechnen. Alle Fehlerursachen, abgesehen von jenen, die vom Empfängerrauschen und vom Mehrwegempfang stammen, können so eliminiert werden.

Die bei den Messungen der Pseudodistanzen erreichbare Genauigkeit von 1 Meter kann bis in den Millimeterberech erfolgen, indem die Trägerphase des Satellitensignals ausgewertet wird. Durch Beobachtung mehrerer Satelliten zu verschiedenen Zeiten und durch ununterbrochenen Vergleich zwischen Anwender- und Referenzempfänger kann nach dem Lösen von umfangreichen Gleichungssystemen die Position im Bereich von einigen Millimetern bestimmt werden.

2.1.2.2 Netzwerkbasierte Lokalisierung

Der Aufbau eines Positionssystems verursacht hohe Kosten und einen hohen Aufwand. Deshalb sollte die bereits existierende Infrastruktur genutzt werden.

Mobilfunk-Ortung Bei GSM und UMTS handelt es sich um zellbasierte Netze. Deren Empfangsbereich ist in wabenförmige Zellen aufgeteilt. In jeder dieser Zellen befindet sich eine Sende-/Empfangsstation, die diese Zelle mit Funksignalen versorgt. In Ballungszentren sind viele dieser Stationen zu verzeichnen. Der Abdeckungsbereich beträgt dort wenige $100m$. In ländlichen Gegenden hingegen kann dieser bis zu $35km$ groß sein. Die Zellgröße ist demzufolge abhängig von der in der Region zu erwartenden Nutzerdichte.

Da sich der Empfangsbereich einer Zelle an den Zellgrenzen mit Nachbarzellen überschneidet, muss darauf geachtet werden, dass benachbarte Stationen nicht auf gleicher Frequenz senden, da das sonst zu Störungen führen kann.
Der Nutzer meldet sich immer an der Zelle mit dem besten Signal an.

Die Lokalisierung mit dem zuvor beschriebenen Verfahren der COO war von Anfang an möglich. Später wurde dieses ergänzt durch zusätzliche Verfahren wie TOA, TDOA, AOA und Nutzung der Signalstärke (vgl. Abschnitt 2.1.1.4, S. 9). Die Reichweite dessen beschränkte sich allerdings auf die Reichweite des GSM/UMTS Netzes. In Abhängigkeit des genutzten Verfahrens lassen sich unterschiedliche Genauigkeiten erzielen. Bei der Bestimmung der Cell-ID beschränkt sich diese auf $100m$ bis $35km$, da es erheblich von der Zellengröße abhängt. Bei einer hohen Dichte an Basisstationen lässt sich mit TOA eine Genauigkeit von $150m$ erreichen, wohingegen das Verfahren der Ermittlung der Signalstärke die Genauigkeit auf $300m$ beschränkt. Diese Genauigkeiten lassen sich jedoch durch weitere Verfeinerungen der Verfahren, als auch durch Aufrüstung der Infrastruktur durch Erhöhung der Dichte der Basisstationen erzielen. Da diese Genauigkeiten jedoch für viele Anwendungen bereits ausreichen, handelt es sich dabei um eine kostengünstige Positionierung, die auch innerhalb von Gebäuden nutzbar ist.

WLAN Mittlerweile existieren dutzende von verschiedenen Projekten und Produkten für die *WLAN*-Positionierung im Innenbereich. Durch ausgeklügelte Algorithmen können bereits Genauigkeiten von bis zu einem Meter erzielt werden (vgl. [IHS04]). Die Grundidee basiert auf der Annahme, dass sich an jedem Ort Signale mehrerer WLAN-Basisstationen mit unterschiedlicher Signalstärke messen lassen. Empfängt man mindestens die Signale von vier oder fünf Access Points (AP), so kennzeichnen die Messwerte den Ort eindeutig. Solch ein elektronischer Fingerabdruck besteht aus einer Liste von für jedes WLAN-Gerät eindeutigen MAC-Adressen[4] und zugehörigen Signalstärken. Dieser wird, zusammen mit dessen Standort, in einer Datenbank gespeichert. Eine Lokalisierung besteht demzufolge aus der Suche nach passenden Fingerabdrücken in der Datenbank sowie der Wichtung und Kombination der gefundenen Kandidatenpositionen zur Positionsschätzung. Komplexere Methoden dahingegen bedienen sich statistischer Verfahren um Fehler in der Positionsschätzung durch Betrachtung des bereits zurückgelegten Weges auszugleichen. Unplausible Bewegungen und Sprünge können so beispielsweise durch Einsatz des Kalman-Filters erkannt und korrigiert werden.

Aber auch für den Außeneinsatz ist dieses Prinzip nicht uninteressant: die dafür notwendige Infrastruktur sei in städtischen Bereichen bereits vorhanden (vgl. [19]). Innerhalb eines 25 Quadratkilometer großen Testgebietes in Nürnberg kartierten Forscher des Fraunhofer Institut für Integrierte Schaltungen (Fraunhofer IIS) beispielsweise durchschnittlich 2000 WLAN-Sender pro Quadratkilometer. Aufgabe des im Januar gestarteten Pilotprojekts ist die Erprobung einer WLAN-Lokalisierungstechnologie[5]. In den USA dahingegen hat sich ein solches System bereits etabliert: Apple hat als erster Hersteller diese neue Technik bereits im iPhone verbaut. Dieses basiert auf dem von der US-Firma Skyhook Wireless Inc. entwickelten System, mit dem man per WLAN seine Position bestimmen kann. Dessen Datenbasis umfasst bereits die katalogisierten Daten von über 23 Millionen Access Points in zehntausenden Städten weltweit. Die Genauigkeit, die auf diese Weise erreichbar ist, ist allerdings sehr unterschiedlich. Skyhook gibt an, die Position im Bereich von 10 bis 20 Metern genau feststellen zu können. Das Fraunhofer IIS dahingegen will im Außenbereich bis auf zehn, in Innenräumen sogar bis auf drei Meter genau sein.

Anders als bei Mobilfunknetzen gibt es jedoch keinen zentralen Anbieter, der deutschlandweit – geschweige denn weltweit – ein flächendeckendes WLAN betreibt. Zudem unterliegt die WLAN-Infrastruktur im öffentlichen Bereich häufigen Änderungen – sei es durch den Umzug von Anwohnern, der Zeitsteuerung des WLAN-Betriebs oder baulichen Verände-

[4] Die *Media Access Control (MAC)* Adresse bezeichnet die weltweit eindeutige Hardware-Kennung jedes Netzwerkadapters und wird zu dessen Adressierung im Netzwerk auf Schicht 2 des *Open Systems Interconnection (OSI)* Referenz-Modells verwendet.

[5] Nähere Informationen können unter [3] eingesehen werden

rungen, die die Signalausbreitung beeinflussen. Sowohl der Aufbau als auch die Wartung und Pflege der benötigten Referenzdaten sind somit mit erheblichem Aufwand verbunden, um eine Positionierung zu ermöglichen und die angestrebte Genauigkeit gewährleisten zu können. Eine erweiterte Abgleichsphase im Algorithmus der Lokalisierung des Fraunhofer IIS ermöglicht daher die Erkennung hinzugekommener oder fehlender Stationen gegenüber der Datenbasis. Diese Aktualisierungsvorschläge werden zentral bewertet und nach Kontrolle der Rückmeldungen weiterer Nutzer zur Pflege der Referenzdatenbank verwendet. Dies ermöglicht eine dynamische Pflege der Referenzdatenbasis der dezentralen innerstädtischen WLAN-Infrastruktur.

Da eine ausreichende Dichte an WLAN-Hotspots außerhalb von Städten heutzutage nicht gewährleistet werden kann, ist das *WiFi-Positioning System (WPS)* nicht geeignet um als alleiniges Navigationssystem zu dienen. Bereits in Randgebieten dürfte es kompliziert werden genügend Hotspots für eine zufriedenstellende Positionsbestimmung zu finden. Um solche WLAN-losen Phasen überbrücken zu können ist der Einsatz hybrider Verfahren, sofern die Hardware die entsprechenden Möglichkeiten bietet, erforderlich. Auf entsprechende Verfahren und Techniken wird daher in Kapitel 2.1.2.4 genauer eingegangen.

2.1.2.3 Lokalisierung innerhalb von Gebäuden

RFID Mittels *Radio Frequency Identification* (RFID) können Positionierungssysteme entwickelt werden, die mit dem Prinzip der Funk-Baken realisiert werden können. In den letzten Jahren ist die RFID-Technik entscheidend weiterentwickelt worden. Heute existieren RFID-Tags in der Grösse eines Reiskorns oder in Form hauchdünner Etiketten. Moderne Lesegeräte können hunderte von Tags in Sekundenschnelle auslesen.

Ein RFID-System besteht prinzipiell aus zwei Komponenten: dem *Transponder*[6], der an dem zu identifizierenden Objekt angebracht wird und dem *Lesegerät*, das je nach eingesetzter Technologie als Lese- oder Schreib-Einheit dient.

Beim Transponder handelt es sich um den eigentlichen Datenträger des Systems. Er wird deshalb als *Tag* bezeichnet. Neben der Technik für die Funkkommunikation enthalten die Tags einen Speicher zusammen mit einem Zustandsautomaten[7] oder einem Mikroprozessor. Um die Informationen eines RFID-Tags auszulesen, müssen Transponder und Lesegerät gekoppelt werden. Diese Kopplung erfolgt automatisch, wenn sich der Tag im Ansprechbereich des Lesers befindet, und basiert auf elektrischen, magnetischen oder elek-

[6]Ein Transponder ist ein aus den Wörtern Transmitter und Responder zusammengesetzter Begriff. Er kann Signale empfangen und auf diese antworten.

[7]Dieser wird durch eine elektrische Schaltung realisiert und erlaubt die Durchführung logischer Verknüpfungen sowie das Speichern von Variablenzuständen.

tromagnetischen Feldern. Durch die induktive Kopplung können Daten zwischen dem Lesegerät und dem Transponder ausgetauscht werden. Dies wird in Abbildung 2.8 skizziert.

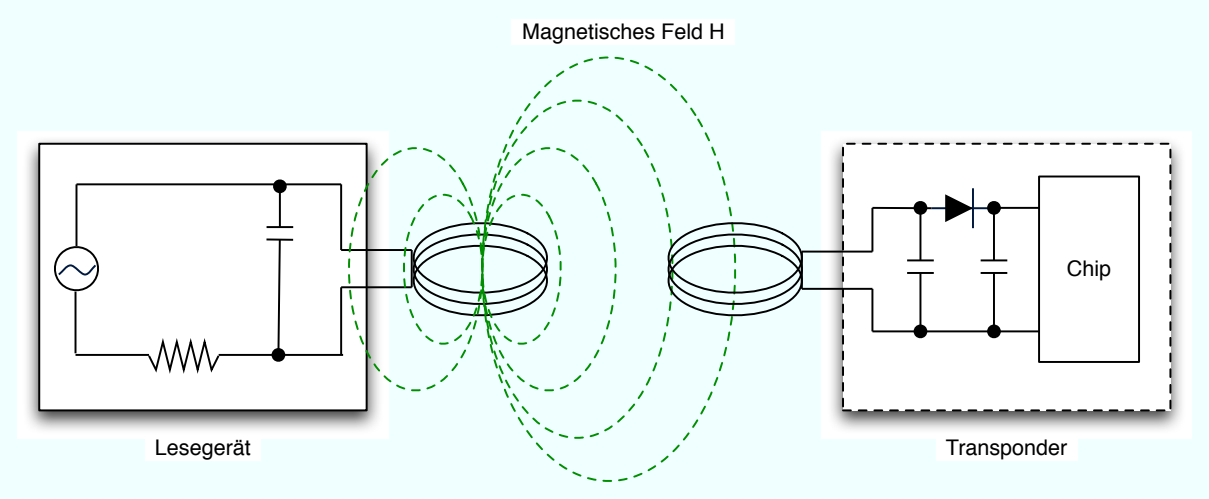


Abbildung 2.8: Prinzip der induktiven Kopplung (Darstellung nach [Fin06])

Damit die Elektronik des Tags arbeiten kann, muss sie mit Energie versorgt werden. Diese wird dem Transponder ebenfalls drahtlos durch die Kopplung mit dem Lesegerät zur Verfügung gestellt. Deshalb sind die meisten Tags passive Elemente, die ohne Batterie auskommen. Für gewisse Systeme sind aktive Tags mit Batterie nötig. Mit aktiven Tags sind solche gemeint, die eine Batterie besitzen und mit deren Energie selbständig in periodischen Intervallen Radiosignale aussenden können. Diese stützt jedoch lediglich den Betrieb der Elektronik. Die Übermittlung der Daten vom Transponder zum Lesegerät erfolgt ausschließlich dadurch, dass das vom Lesegerät ausgesendete Feld auf geeignete Weise verändert wird. Diese Veränderung kann dann vom Lesegerät erkannt und ausgewertet werden.

Die Berliner Firma Bitmanufaktur[8] stellt zum Beispiel eine offene und lizenzfreie Hard- und Softwareplattform *OpenBeacon* zur Verfügung, die die Positionierung mit aktiven Tags realisiert. Solch ein Szenario bietet sich besonders im Bereich des Ubiquitous Computing an: das Verknüpfen von physikalischen Objekten aus der realen Welt mit deren virtuellen Repräsentation in der virtuellen Welt.

[8] `http://www.openbeacon.org`

2.1.2.4 Hybride Techniken

Jedes der zuvor beschriebenen Verfahren hat spezifische Vor- und Nachteile und ist deshalb nur bedingt dafür geeignet als alleiniges Navigationssystem zu dienen.

GPS-basierte Systeme versagen bezüglich der Positionsbestimmung in Gebäuden oder in dicht bebauten Gegenden. Ein System, dass sich auf die Nutzung von WPS spezialisiert, kann aufgrund der fehlenden Abdeckung in ländlichen Gebieten nicht akzeptiert werden. Terrestrische Ortung mittels GSM/UMTS kann die Anforderungen an die zu erreichende Abdeckung und der daraus resultierenden Ungenauigkeit nicht erfüllen.

Die Lösung des Problems liegt in der Kombination sich ergänzender Technologien. Einige dieser hybriden Techniken sollen deshalb nachfolgend vorgestellt werden.

A-GPS Auf eine kontinuierliche Nutzung von GPS-Modulen mobiler Geräte zur Bereitstellung von Positionsdaten für standortbezogene Dienste wird, zu Gunsten des Stromverbrauchs, normalerweise verzichtet. Stattdessen wird es sporadisch genutzt. Deshalb kann es vorkommen, dass keine Daten über die aktuelle Position der Satelliten vorhanden sind. Die benötigten Bahndaten müssen deshalb erneut empfangen werden. Normalerweise werden dann mindestens 40 Sekunden benötigt [4], um die erforderlichen Daten zu beziehen und die erste Position berechnen zu können. Unter schwierigen Empfangsbedingungen, beispielsweise der Abschattung der Signale durch Hochhäuser, kann die Wartezeit bis zur ersten Positionsberechnung mehrere Minuten betragen, falls diese überhaupt ermöglicht werden kann. Diese Limitierung der GPS-Technologie lässt sich auch durch eine verbesserte Empfängertechnologie nicht beheben.

Beim *Assisted-GPS* (A-GPS) wird deshalb ein als *Aiding* bezeichneter Ansatz umgesetzt – das zur Verfügung stellen von Satelliten-Daten und weiteren Informationen über andere Kommunikationskanäle, beispielsweise über GMS, GPRS oder UMTS. Durch die Bereitstellung dieser Hilfsdaten, die unter anderem Informationen über den Almanach oder die Ephemeriden enthalten, kann der Empfänger die Position innerhalb kurzer Zeit berechnen, auch unter schlechten Empfangsbedingungen. Um A-GPS einsetzen zu können, muss der GPS-Empfänger jedoch über eine Schnittstelle verfügen, die es ermöglicht die Aiding-Daten zu empfangen.

XPS Bei dieser hybriden Technik, dessen Bezeichnung *XPS* durch die Firma Skyhook geprägt wurde, handelt es sich um die softwaretechnische Kombination der Techniken WPS, GPS und der Mobilfunkortung. Dieses hybride Verfahren in der Version 2.0 wurde am 30. Juni bekanntgegeben und vorgestellt [30]. Mit Hilfe dieser Software kann auf

jedem mobilen Gerät – sofern es die technischen Möglichkeiten zur Verfügung stellt – eine Positionsbestimmung mit einer Genauigkeit im Bereich von 10 bis 20 Metern ermöglicht werden.

Durch geschickte Kombination der Vorteile der zu Grunde liegenden Technologien lässt sich so eine sekundenschnelle, exakte und zuverlässige Positionierung realisieren. Die Verfügbarkeit und Genauigkeit kann demzufolge sowohl in ländlichen Gegenden als auch innerhalb von Gebäuden oder städtischen Umgebungen gewährleistet werden.

Die Genauigkeit der durch WPS initial ermittelten Position kann mit Hilfe von XPS und den zusätzlichen Informationen von 2 Satelliten um rund 35% verbessert werden. Verglichen zu A-GPS, was zur Bestimmung der Position ca. 40 Sekunden benötigt wenn noch keine GPS-Daten vorlagen, benötigt dieses Verfahren lediglich 4 Sekunden. Die Nutzung der Mobilfunkortung erhöht zudem die Verfügbarkeit, so dass dem Nutzer überall eine Position zugewiesen werden kann.

2.1.2.5 Zusammenfassung

Als Ergebnis der vorangegangenen Diskussion scheint das hybride Verfahren XPS im Rahmen dieser Arbeit am besten für die gestellte Aufgabe geeignet zu sein. Es erfüllt die zuvor definierten Anforderungen hinsichtlich der Abdeckung, Verfügbarkeit, Genauigkeit und Interoperabilität vollständig. Lediglich die Genauigkeit der Positionsbestimmung über WPS in dicht bebauten städtischen Umgebungen, kann vor praktischen Tests nicht eingeschätzt werden. Auch wenn dies nicht die erwünschten Daten liefern würde, stünden immer noch die sich ergänzenden Verfahren der Mobilfunkortung und des GPS zur Verfügung.

2.2 Ubiquitous und Pervasive Computing

Ubiquitous Computing, kurz UbiComp, bedeutet, dass Computer in allen Gegenständen und Geräten eingebettet werden und sich dabei nahtlos, für den Menschen völlig unbemerkt, in seine Umgebung und in seinen Alltag integrieren. Alle Geräte enthalten Sensoren und sind miteinander vernetzt, um Daten und Informationen aufzunehmen, zu analysieren und auszutauschen, um dem Menschen Arbeit abzunehmen und dessen Alltag zu erleichtern.

Diese technologische Vision des Ubiquitous Computing als eine allgegenwärtige Infrastruktur der Informations- und Kommunikationstechnik wurde 1991 von Mark Weiser in seinem visionären Artikel „The Computer for the 21st Century“ [29] geprägt.

> „In the twenty-first century the technology revolution will move into the everyday, the small and the invisible. The impact of technology will increase ten-fold as it is imbedded in the fabric of everyday life. As technology becomes more imbedded and invisible, it calms our lives by removing annoyances while keeping us connected with what is truly important. This imbedding, this invisibility, this radical ease-of-use requires radical innovations in our connectivity infrastructure.“ – M.D. Weiser [Wan00]

Seine Definition gilt noch heute als Fundament für alle daraus resultierenden technologischen und gesellschaftlichen Betrachtungen. Er beschreibt vier prägnante Entwicklungen in der Benutzung und Verbreitung computerisierter Systeme im Umfeld des Ubiquitous Computing:

- Unsichtbarkeit: Neue Systeme zeichnen sich dadurch aus, dass sie sehr klein und für den Anwender nahezu unsichtbar sind. Sie verschmelzen mit ihrer Umgebung und werden damit zu einem integralen Bestandteil des alltäglichen Lebens.
- Ubiquitäre Verfügbarkeit: Sinkende Größe, steigende Leistungsfähigkeit und fallende Preis von Computern führen zu einer rapide steigenden Verfügbarkeit.
- Die Beziehung zwischen Mensch und Computer erfährt einen grundlegenden Wandel: ein Benutzer besitzt und interagiert nicht mehr nur mit einem Gerät, sondern mit einer Vielzahl von Computern.
- Ubiquitäre Kommunikation: Computer können ubiquitär kommunizieren. Die Entwicklung auf dem Sektor der funkbasierten Kommunikation ermöglicht eine weitgehende Vernetzung der Computer.

Gerade dieser letzte Punkt unterscheidet damit das Ubiquitous Computing von den heute bekannten mobilen Netzen. Es zeichnet sich durch die allgegenwärtige und mobile Verfügbarkeit der eigentlichen Anwendungsdienste aus, unabhängig von der eigentlichen Zielplattform. Die Dienste werden plattformunabhängig je nach den gegebenen physikalischen Möglichkeiten des jeweiligen Geräts angeboten, sei es auf Mobilfunkgeräten, dem PDA oder anderen Geräten, die einen Mehrwert zur Kommunikation anbieten.

Heute bezeichnet Ubiquitous Computing einen Forschungsbereich, der sich in mehrere selbständige Teilgebiete untergliedern lässt: zentrale Bereiche sind hier mobile drahtlose Ad-hoc-Netzwerke, Mensch-Maschine Interaktion und Benutzerschnittstellen, mobile verteilte Sensornetzwerke, Systemsoftware, Betriebssysteme und Kontextverarbeitung (vgl. [Zim07]).

Parallel zum Begriff des Ubiquitous Computing hat sich auch der Begriff *Pervasive Computing*, der vor allem auf kurz- und mittelfristig machbare Lösungen abzielt, durchgesetzt. Während Weiser UbiComp eher als unaufdringliche, auf den Menschen zentrierte Technik beschrieben hat, mit der alles durchsetzt ist und die nur als Mittel zum Zweck verwendet wird, setzt die Industrie bei Pervasive Computing den Schwerpunkt auf mögliche e-Commerce-Szenarien und web-basierte Geschäftsprozesse. Er wurde vor allem von IBM geprägt (vgl. [HNS01]). Pervasive Computing wird somit weniger als eigenständiges Technologiefeld, sondern als eine neue Anwendungsform der Informations- und Kommunikationstechnik betrachtet, die sehr viel stärker als bisher in die Alltagswelt integriert wird. Ziel ist es, durch die Allgegenwart den Anspruch auf ständige Verfügbarkeit im Hinblick auf Datenverarbeitung und -übertragung zu realiseren. Das Pervasive Computing verfolgt somit einen komplementären Ansatz zur virtuellen Realität: statt die gesamte Welt im Computer abzubilden und zu simulieren, werden alle Gegenstände der realen Welt Teil eines Informations- und Kommunikationssystems – reale und virtuelle Welt überlagern sich und verschmelzen miteinander.

Auf Grund der Integration in Alltagsgegenstände werden künftige Computer meist gar nicht mehr als solche wahrgenommen werden. Sie bieten dem Benutzer Dienste an, die ihn bei seinen alltäglichen Aufgaben unterstützen, ohne dass dieser explizite Vorgaben macht bzw. Entscheidungen trifft: im Pervasive Computing denkt die Umgebung mit und wird – so die Vision – zu einem kooperativen Partner des Menschen. Die Aufmerksamkeit des Benutzers soll infolgedessen nicht mehr als unbedingt notwendig belastet werden, was durch Reduktion expliziter Interaktion realisiert werden kann. An ihre Stelle tritt die implizite Interaktion mit dem Benutzer und der Umwelt. Dies wird ermöglicht durch die Verfügbarkeit integrierter Sensorik und entsprechender Funkschnittstellen: sie können Informationen über die Umwelt sammeln und diese mit anderen Geräten kommunizieren. Die auf diese Weise zusammengetragenen Informationen werden genutzt, um Rückschlüsse

auf die aktuelle Situation – den Kontext (vgl. Kapitel 2.3, S. 28) – zu ziehen, in der sich das System befindet. Wird eine Situation oder ein Kontext erkannt, der für das System relevant ist, passt die pervasive Anwendung ihr Verhalten automatisch an. So wird ein explizites Eingreifen des Benutzers unnötig.

Die Vorstellungen über das Pervasive Computing werden gegenwärtig noch stark von zeitnahen Visionen geprägt. Es zeichnet sich ab, dass die Entwicklung in mindestens zwei Stufen verlaufen wird [Inf06]. In einem Pervasive Computing der ersten Stufe werden momentan zahlreiche Produkte und Anwendungen etabliert, die noch stark von den Entwicklungszielen Mobilität und Ad-hoc-Vernetzung gekennzeichnet sind. Im Wesentlichen handelt es sich dabei um die heutigen Trends der Miniaturisierung und der Integration verschiedener Funktionen in ein elektronisches Gerät und den so entstehenden intelligenten Gegenständen. Die Kontextsensitivität wird dabei bereits in vereinfachter Form, etwa in Form von Nutzerprofilen realisiert. Trotz der permanenten Verbindung mit Kommunikations- und Datennetzen handelt es sich bei den intelligenten Gegenständen weitestgehend um spezifische Lösungen, die eine Vielzahl von Fähigkeiten, insbesondere in Hinblick auf Kommunikation und Datenverarbeitung, in sich vereinen.

Auch technische Alltagsgegenstände werden vermehrt mit Mikrocontrollern und Sensoren ausgestattet und somit zu intelligenten Gegenständen aufgewertet. Auch deren Funktionalität wird sehr aufgabenspezifisch sein, und einfache Formen der Vernetzung bieten. Infolgedessen werden sich proprietäre Lösungen herausbilden, die vornehmlich anwendungs- oder herstellerspezifisch sind.

Im Pervaisve Computing der zweiten Stufe wird, nach der Überwindung bestehender Medienbrüche, eine wirklich offene Vernetzungsstruktur etablieren – Experten vermuten, dass dies innerhalb von 10 Jahren verfügbar sein wird (vgl. [Inf06]).

Das Konzept des Pervasive Computing zielt darauf ab, alle Lebensbereiche zu durchdringen, sie miteinander zu verbinden und auf diese Weise einen allgegenwärtigen Fluss von Daten, Informationen und – durch die zukünftige Integration kognitiver Leistungen – auch von Wissen zu ermöglichen. Ein derartiger fortwährender und allgegenwärtiger Austausch über Anwendungs-, Medien- und Ländergrenzen hinweg, beschreibt jene Vision, die Mark Weiser mit „Alles, immer, überall“ umrissen hat. Das wiederum wirft Fragen über die Systemsicherheit im Pervasive Computing auf.

2.2.1 Paradigmen des Pervasive Computing

Es gibt viele neue Herausforderungen, die durch die Realisierung des Pervasive Computing entstehen. Als zentrale und fundamentale Anforderungen an Geräte, Software, Protokolle und alle anderen beteiligten Technologien gelten folgende vier Paradigmen [HNS01]:

- Dezentralisierung
- Diversifikation
- Konnektivität
- Einfachheit

2.2.1.1 Dezentralisierung

Zu Beginn der Computer-Ära gab es nur einzelne Mainframes – Rechner, die ganze Räume ausfüllten und gleichzeitig von vielen Anwendern genutzt wurden. Dieses zentralisierte Konzept wurde durch die Verbreitung der Personal Computer und der Client-Server-Architektur bereits in Richtung Dezentralisierung verändert.

Prinzipiell gibt es nicht mehr nur einen Server, mit dem die Clients kommunizieren und Daten synchronisieren und der die Verwaltung der Geschäftslogik übernimmt. Stattdessen wird die Verantwortung auf eine Vielzahl kleinerer Geräte verteilt. Auf diese Weise findet eine Spezialisierung auf wenige Aktivitäten und Funktionen statt. Die Geräte bilden demnach ein dynamisches Netzwerk, welches die Kommunikation und Synchronisation untereinander, als auch mit Desktop Applikationen ermöglicht.

Trotz der Dezentralisierung muss der Zugriff auf benötigte Daten zu jedem Zeitpunkt gewährleistet sein. Da Informationen allerdings über unterschiedliche Instanzen verteilt sein können, müssen sich die einzelnen Komponenten des Systems spontan miteinander vernetzen um sich zu synchronisieren. Trotzdem muss die Datenintegrität gewährleistet werden.

2.2.1.2 Diversifikation

Im Bereich des Pervasive Computing beschränkt sich die Nutzung, wie bereits erwähnt, nicht auf einen universell einsetzbaren Computer, sondern auf die Nutzung der Funktionalitäten vieler spezialisierter Geräte. Dabei ist jedes Gerät auf genau eine Situation und Umgebung zugeschnitten. Jedoch können sich deren Einsatzbereiche überschneiden. So hat der Nutzer beispielsweise die Möglichkeit zuhause über den Fernseher aktuelle

Informationen abzurufen und dabei Visualisierungen in bestmöglicher Qualität zu genießen. Unterwegs steht ihm für diesen Anwendungsfall in der Regel nur ein Mobilgerät mit reduzierten grafischen Möglichkeiten zur Verfügung.

Um die technischen Anforderungen des Gerätes zu erfüllen, ist es notwendig, die eingesetzten Technologien an die Anforderungen der Applikationen anzupassen. Es ist eine große Herausforderung diese vielfältigen Funktionalitäten und Charakteristika der unterschiedlichen Geräte zu koordinieren und möglichst allgemeingültige Anwendungen für sie bereitzustellen. Hierbei müssen vor allem die großen Unterschiede im Bereich der Benutzerschnittstelle beachtet werden: die Darstellungsmöglichkeiten des Bildschirms sowie die Eingabemechanismen des Gerätes.

2.2.1.3 Konnektivität

> „Everybody‘s software, running on everybody‘s hardware, over everybody‘s network.“ – Lou Gerstner

Diese Aussage scheint unter den aktuellen Bedingungen nicht realisierbar zu sein. Abhängig vom verwendeten Prozessor, dem Betriebssystem, dem zur Verfügung stehendem Speicher oder Anschlüssen gibt es eine Vielzahl von Beschränkungen und Unterschieden. Um diese Hürden zu überwinden, um Interoperabilität und Konnektivität zu erreichen, müssen allgemeingültige Standards für Applikationen entwickelt und etabliert werden. Und das nicht nur für die unterschiedlichen Geräte, sondern auch für die Kommunikation, Markup-Sprachen und plattformübergreifende Software.

2.2.1.4 Einfachheit

Die Vielzahl der Fähigkeiten, die unsere heutigen Computer besitzen, machen sie für die Mehrheit der Endbenutzer sehr kompliziert. Bereits die Installation neuer Software stellt eine Herausforderung dar und viele Anwendungen sind ohne Vorkenntnisse oder Anleitungen nicht leicht zu bedienen.

Um im alltäglichen Leben akzeptiert zu werden, müssen die neuen Geräte und Applikationen vor allem leicht und intuitiv bedienbar sein. Jeder muss sie schnell und komfortabel nutzen können. Voraussetzung dafür ist beispielsweise, dass die komplexe Technologie unter einem nutzerfreundlichen Interface versteckt wird, was allerdings hohe Ansprüche an die Entwicklung und das Design stellt.

2.2.2 Sicherheit und Datenschutz im Pervasive Computing

Ein zentrales Merkmal des Pervasive Computing ist, dass nahezu alle intelligenten Gegenstände Informationen austauschen können. Die spontane Vernetzung der Geräte macht es dem Nutzer unmöglich zu verfolgen, wo welche persönlichen Daten über ihn gespeichert sind, wie sie verwendet und gegebenenfalls miteinander kombiniert werden. Diese Vernetzung und die verteilt erbrachten Dienste erschweren es, die Zusammenhänge zwischen einer Aktion und ihren Folgen hinsichtlich der Weitergabe und Verarbeitung der eigenen Daten zu erkennen. Nur eine Systemarchitektur, die die Wahrung der informationellen Selbstbestimmung von Beginn an einbezieht, kann das Entstehen ernsthafter Datenschutzkonflikte verhindern. Die große Anzahl an intelligenten Gegenständen und ihre spontane Vernetzung erschwert allerdings die Beherrschbarkeit des Gesamtsystems.

Diese Komplexität und die Unsichtbarkeit des Pervasive Computing können dazu führen, dass Systemausfälle und mutwillige Störungen nicht oder nur sehr spät bemerkt werden. Die Funktionssicherheit[9] ist damit bei sicherheitskritischen Anwendungen eine zwingende Voraussetzung. Zugleich soll das System auch bei Fehlbedienungen weitestgehend konform seiner Spezifikation reagieren. Das kann beispielsweise durch eine redundante Systemauslegung oder durch Ausweichsysteme garantiert werden.

Neben der Wahrung des Datenschutzes und der Sicherung der Funktionssicherheit, ist die Informationssicherheit, die dafür sorgt, dass nur befugte Personen bzw. Objekte an die für sie bestimmten Daten gelangen können, von zentraler Bedeutung. Angesichts der Tatsache, dass die Kommunikation meist drahtlos geschieht, verstärkt sich die Bedeutung der Sicherheitstechnologien umso mehr, da die Funkkommunikation prinzipiell leichter manipuliert werden kann, als dies bei leitungsgebundenen Netzen möglich ist.

[9] Unter Funktionssicherheit wird die Eigenschaft des Systems verstanden, trotz aufgetretener Systemfehler nicht in unkontrollierbare Systemzustände zu geraten, in denen das System sich selbst oder seine Umwelt in Gefahr bringt.

2.3 Kontext und kontextsensitive Dienste

Der Erkennung, Verarbeitung und Speicherung von Kontext kommt in pervasiven Systemen eine besondere Bedeutung zu: erst durch die Berücksichtigung von Kontext, also durch *Context-Awareness*, erreichen positions- und situationsabhängige Anwendungen (oder -Services) die von ihnen erwünschte Funktionalität, nämlich die automatische Berücksichtigung der Situation des Nutzers sowie die automatische Erkennung seiner Intention, die er mit Benutzung einer Applikation oder eines Service verfolgt. Kontext lässt sich allerdings nur effektiv und effizient benutzen, wenn klar ist, was damit gemeint ist, und wie mit ihm umzugehen ist.

Im Folgenden werden daher zunächst die Begriffe Kontext und Context-Awareness geklärt um anschließend einen Überblick über verschiedene Verfahren zur Kontexterkennung zu geben. Abschließend werden Herausforderungen im Umgang mit Kontext und kontextsensitiven Diensten erörtert.

2.3.1 Begriffsklärung

Eine Schwierigkeit auf dem Gebiet des Pervasive Computing ist das Fehlen einer allgemein gültigen Definition des Kontextbegriffes. Dieser Abschnitt gibt deshalb einen Überblick über dessen Entwicklung innerhalb des Pervasive Computing.

2.3.1.1 Kontext

Im Umfeld des Pervasive Computing erscheint der Begriff des *Kontext* zum ersten Mal 1994 in dem Artikel [ST94] von Bill Schilit und Marvin Theimer. Seit dieser Zeit finden sich eine Vielzahl zum Teil recht unterschiedlicher Ansätze, die den Begriff des Kontext bearbeiten und jeweils eigenständig interpretieren. Die ersten Definitionen charakterisieren Kontext durch Enumeration seiner Komponenten: Schilit und Theimer verstehen darunter die Lokation und Identität von Objekten und Personen und die Änderungen, die diese über die Zeit erfahren ([ST94]); die Identität von Personen in der Umgebung des Benutzers, die Tageszeit und die Jahreszeit werden dahingegen von Brown, Bovey und Chen ([BBC97]) als Kontext gesehen.

Hitz et al. [HKRS02] betrachten in ihrer Arbeit explizit die Modellierung von ubiquitären Web-Anwendungen: sie nehmen eine Unterscheidung zwischen natürlichem (Ort, Zeit), technischem (Endgerät, Browser, Netzwerk, Status) und sozialem Kontext (Benutzerprofil und -verhalten) vor. Die Definition durch Beispiellisten und feste Klassifikationen haben allerdings einen sehr statischen Charakter – sie erlauben keine dynamische Adaptierung

des Begriffs Kontext und sind somit potentiell ungeeignet für eine Umschreibung des Kontext in pervasiven Systemen.

Andere Definitionen lösen sich daher von der reinen Beschreibung durch physikalische Größen wie beispielsweise Ort, Zeit, räumliche Nähe, Temperatur und bedienen sich einer abstrakteren Sicht von Kontext: diese beinhalten zumeist die *Situation* als Komponente des Kontextes (vgl. [WJH97, DSA01]). Die Begriffsbestimmung von Dey et al. definiert folgendes:

> „Context is any information that can be used to characterize the situation of an entity, where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves.“ [DSA01, S. 3]

Sie differenziert damit die umgebende Situation, die beteiligten *Entitäten* (Personen, Objekte, Orte) und Interaktionen. Im Bereich des Pervasive Computing handelt es sich dabei wahrscheinlich um die weit verbreitetste Definition (vgl. [CCRR02, GPZ04, GWPZ04]), was auf ihre Allgemeinheit und Flexibilität zurückzuführen ist. Sie umfasst zudem automatisch gewonnene Informationen als auch manuelle Eingaben des Benutzers, insofern sie für die stattfindende Interaktion von Interesse sind. Auch im Rahmen dieser Arbeit werde ich mich auf diese generalisierte Definition des Begriffs Kontext beziehen.

Generell lässt sich feststellen, dass sich der Kontext eines Geräts oder Nutzers nicht nur auf die Bestimmung des Ortes beschränken lässt (vgl. [SBG99]). Es gibt sehr viel mehr Aspekte die beachtet werden müssen, um den kompletten Kontext zu beschreiben – einige von ihnen sind unter Umständen nur für spezifische Applikationen von Interesse, andere, wie beispielsweise der Ort, für einen Großteil von ihnen. In Tabelle 2.1 werden daher einige der Aspekte zusammengefasst.

Selbstverständlich benötigen unterschiedliche Applikationen differenzierte Kombinationen dieser Aspekte um Kontext zu beschreiben. Beispielsweise hängt eine Anwendung zur Kommunikation von dem technologischen Aspekt ab, wohingegen ein Touristenführer geographische Informationen benötigt. Ein einzelner Sensor scheint jedoch nicht praktikabel zu sein, um unterschiedliche Aspekte zu untersuchen. Stattdessen wird eine Vielzahl von Sensoren benötigt, um eine Teilmenge der zuvor genannten Punkte zur Auswertung der aktuellen Situation des Nutzers oder Geräts zu nutzen.

Kategorie	Beispiele
geographische	Land, Straße, Gebäude, Etage, Büro
physische	Helligkeit, Geräuschpegel, Temperatur, Beschleunigung, Neigung
organisatorische	Institution, Abteilung, Gruppe, Projekt
soziale	Familie, Freunde, Mitarbeiter, Single
emotionale	Herzfrequenz, Hautimpedanz
Nutzer	Profil, Ort, Rolle, Zugriffsrechte
Aufgabe	Dokumentieren, Programmieren, Aktivität
technologische	Konnektivität, Netzwerkbandbreite, Netzwerklatenz
zeitliche	Uhrzeit, Wochentag, Woche, Monat, Jahreszeit

Tabelle 2.1: Übersicht möglicher Aspekte des Kontexts

2.3.1.2 Kontextsensitivität

Die Fähigkeit (einer Anwendung oder eines Service), Kontext aus einer entsprechend instrumentierten Umwelt wahrzunehmen und zu benutzen (vgl. [TW05, NTGCP05]) um ihr Verhalten dementsprechend automatisch anzupassen, wird im Allgemeinen als *Kontextsensitivität* bezeichnet.

Für kontextsensitives Verhalten finden sich in der Literatur eine Vielzahl sich ähnelnder Definitionen, bei denen zum Teil Unterscheidungen zwischen Systemen, die Kontextinformationen benutzen und solchen, die durch Adaption auf Kontexte reagieren ([Hei98, CGS+02]), gemacht werden. Beispielhaft soll deshalb auch hier die Definition von [DSA01] angeführt werden:

> „A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.“ [DSA01, S. 6]

Diese Definition kontextsensitiver Systeme fasst frühere Ansätze zusammen. Kontextsensitivität ermöglicht demnach einer Anwendung bzw. einem Service, die Erwartungen des Nutzers in größerem Maße zu erfüllen, als es ohne die Berücksichtigung von Kontextinformationen möglich wäre, indem besser auf die Anforderungen des Nutzers eingegangen wird. Somit lassen sich insbesondere die Eigenschaften Anpassungsfähigkeit und Intentionalität – die von einem pervasiven System gefordert werden – erreichen.

Die Nutzung geeigneter Informationen zur Charakterisierung der aktuellen Situation eines Objektes, eines Ortes oder einer Person steht im Vordergrund. Er sollte infolgedessen

eine oder mehrere Kontextinformationen abfragen können. Die Fähigkeit diese Information – den Kontext einer Entität – anzufordern und zu benutzen, um die bereitgestellte Funktionalität idealerweise an die Intention des Nutzers anzupassen, ist der Kerngedanke dieser Dienste.
Das Verb „anfordern“ bedeutet in diesem Sinne allerdings nicht, dass der Dienst auf ein Pull-Verfahren beschränkt ist. Auch eine Anmeldung bei einem Dienst, der dafür sorgt, dass Ereignis-abhängig Kontextinformationen an den Dienst geschickt werden (Push-Verfahren), ist in diesem Sinne eine „Anforderung“.

2.3.1.3 Situation

Im Allgemeinen wird mit einer *Situation* die Gesamtheit der aktuellen Umstände oder Verhältnisse bezeichnet. Der Situationsbegriff geht anscheinend mit der allgemeinen Definition von Kontext einher – und umgekehrt.

Wie in Abschnitt 2.3.1.1 bereits angedeutet wurde, verwenden einige Auten die Begriffe Situation und *Situation-Awareness* synonym für Kontext bzw. Kontext-Awareness [ST94, WJH97, HNBr97].

Ausgehend von der Kontextdefinition auf Seite 28 beschreibt Dey et al. eine Situation als eine Menge von Zuständen von Entitäten und eine Situationsabstraktion als die Beschreibung der Zustände relevanter Entitäten [DSA01]. Nach Schmidt et al. sind kontextsensitive Anwendungen dadurch charakterisiert, dass sie situationsabhängige Informationen während der Verarbeitung berücksichtigen. Eine konkrete Situation ließe sich aber „im Allgemeinen nicht vollständig und objektiv beschreiben“[Sch02]. Für die Nutzung des Kontexts in Anwendungen genügt es, Situationen durch ihre charakteristischen Merkmale hinreichend genau zu beschreiben.

2.3.2 Der Prozess der Kontextgewinnung

Nachdem bereits die Grundlagen des Pervasive Computing (Abschintt 2.2, S. 22) und die Bedeutung von Kontext (Abschnitt 2.3.1.1) erörtert wurden, steht nun die Gewinnung von Kontext für die Verwendung in einer pervasiven Applikation im Vordergrund.

Kontext kann im einfachsten Fall durch eine Eingabe des Nutzers gewonnen werden. Allerdings ist dies im Bereich des Pervasive Computing nicht akzeptabel, da (wie in Abschnitt 2.2 bereits erläutert) die Applikation und die Interaktion mit dieser in den Hintergrund treten soll.
Eine automatisierte Erkennung des Kontexts ist daher unabdingbar. Im einfachsten Fall

kann dies durch eine einfache Abbildung der Sensor-Daten durch eine einzelne Funktion f erfolgen. In Abbildung 2.9 ist dieses generelle Vorgehen visualisiert.

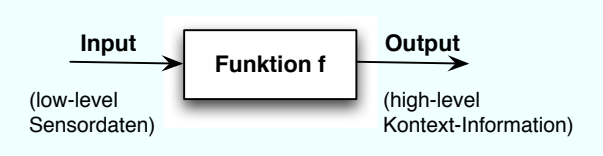


Abbildung 2.9: Idealisierter Überblick der Kontextgewinnung (adaptiert von [May04])

Bei dieser Abbildung handelt es sich um ein abgeschlossenes System und kann somit auch auf sehr viel komplexere Konzepte bezogen werden: es definiert das Ein- und Ausgabe-Verhalten des Prozesses der Kontextvorhersage.
In einer Messphase werden die Eingabedaten von verschiedenen Quellen erfragt, etwa von physikalischen Sensoren, über Lokalisierungsmethoden wie das GPS (vgl. Abschnitt 2.1.2.1) oder aus Datenbanken mit spezifischen Profilen. Die erhaltenen Rohdaten werden als *low-level context information* bezeichnet, welche in einem oder mehreren Verfeinerungsschritten zu *high-level context* werden. Diese zeichnen sich durch einen höheren Abstraktionsgrad aus.

Bei den Eingabedaten solcher Systeme handelt es sich im Allgemeinen um heterogene und komplexe Datenströme. Es wird hierbei zwischen digitalen und analogen Signalen unterschieden. Das Hauptaugenmerk liegt jedoch auf einer diskreten Folge von Elementarsignalen im Zeit- und Wertebereich. Zudem ist es möglich, dass es sich dabei um multidimensionale Eingabeströme handelt, bei denen zu einer konkreten Zeit t_n ein Vektor $\overrightarrow{x}$ zugeordnet ist.

Die Ausgabedaten des Systems charakterisieren den aktuellen Kontext oder die Vorhersage zukünftiger Kontext-Informationen. Dabei muss es sich nicht um einen konkreten Wert handeln: es ist auch möglich, eine Liste von Kontext-Daten als Ergebnis der Funktion zu erzeugen.
Bei der Erzeugung dieser Kontext-Information handelt es sich um einen sehr komplexen Prozess, der durch mehrere Schritte definiert wird. In Abbildung 2.10 wird dies skizziert.

Da eine vollständige Betrachtung der Thematik den Rahmen dieser Arbeit übersteigen würde, möchte ich mich auf die ersten drei Schritte beschränken:

- Sammeln der relevanten Daten
- Merkmalsextraktion
- Klassifikation

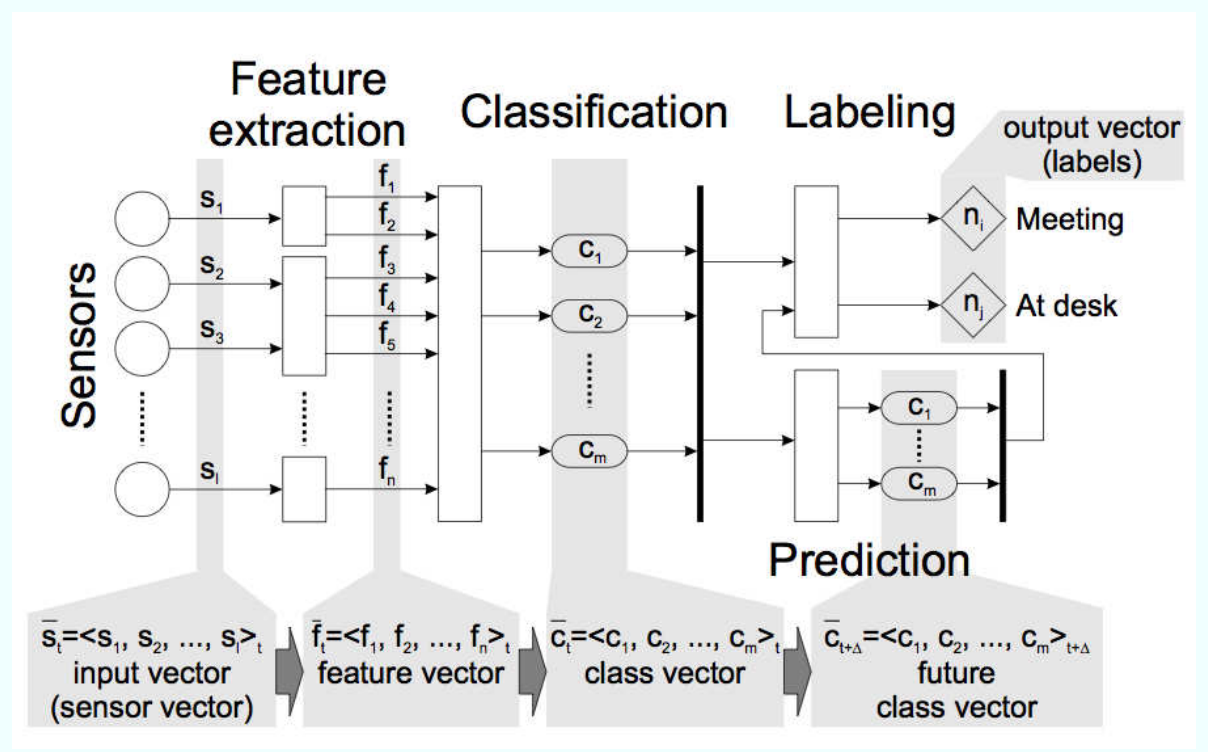


Abbildung 2.10: Architektur der Kontext-Vorhersage [May04]

Bei der Bestimmung des Kontexts und der aktuellen Situation handelt es sich um einen Bestandteil dieser Arbeit und somit einen wichtigen Teil des Entwurfs, deshalb sollen nachfolgend die unterschiedlichen Schritte der Kontextvorhersage besprochen werden.

2.3.2.1 Sammeln der Sensordaten

Um die Qualität der vorhergesagten Kontext-Informationen zu maximieren, ist die Auswahl der zu benutzenden Sensoren von höchster Wichtigkeit. Sie sollten eine möglichst vielfältige und komplementäre Sicht der aktuellen Umgebung bieten. Um das zu erreichen, ist es grundsätzlich möglich mehrere unterschiedliche Sensoren zu nutzen – jeder mit einer anderen Sicht auf die Nutzer und dessen Umwelt. Viele der heute verfügbaren Sensoren produzieren allerdings keinen numerischen Output, der direkt für die Vorhersage genutzt werden kann.[10]

In Tabelle 2.2 wird deshalb ein kurzer Überblick über mögliche Sensoren und deren resultierende Eingabedatenströme gegeben. Da sowohl gemessene Beschleunigungsdaten als auch durch den Nutzer ausgeführte Programme im Prinzip als Sensoren genutzt werden können, werden sie in [Sch02] in physische und logische Sensoren unterteilt. Diese Unter-

[10] Beispiele für solche Sensornetzwerke sind: Bluetooth oder WLAN.

Sensoren	Rohdaten
Kamera	Pixel
Mikrophon	Geräuschpegel
GPS	Positionskoordinaten
Gyroskop	Richtung
Beschleunigungssensor	Beschleunigung
WLAN	Netzwerkstatus

Tabelle 2.2: Übersicht einiger Sensoren und ihrer Eingabedaten in das System

teilung soll bei der Umsetzung des Prototyps allerdings nicht vorgenommen werden. In dieser Arbeit sind Sensoren Entitäten, die Messungen der Umwelt widerspiegeln, in denen sich der Nutzer gerade befindet.

2.3.2.2 Kontextverarbeitung

Eine zentrale Komponente kontextsensitiver Systeme ist die *Kontextverarbeitung*. Sie untersucht Algorithmen und Verfahren, welche die Erkennung von Kontexten auf der Basis sensorisch erfasster Daten oder anderer Kontexte ermöglicht und geht einher mit den Begriffen *Kontexterkennung* und *Kontextaggregation und -fusion*.

In der Kontexterkennung werden Algorithmen untersucht, mit deren Hilfe auf Basis von Sensordaten und optionalen Informationen über die Umgebung, Rückschlüsse auf die aktuelle Situation gezogen werden können. Die Verfahren der Aggregation und Fusion von Kontexten dienen der Verknüpfung abstrakterer Informationen: die Fusion von Daten bezeichnet im Allgemeinen die Zusammenfassung gleichartiger Daten zum Zweck der Genauigkeitsverbesserung oder Fehlerreduktion wohingegen die Aggregation sich mit der Verknüpfung verschiedenartiger Daten auseinandersetzt, um daraus neue Erkenntnisse zu gewinnen. Auf diese Weise können verschiedene Kontexte zusammengefasst werden, um umfassende Beschreibungen der aktuellen Situation abzuleiten.

Merkmalsextraktion Der Prozess der Merkmalsextraktion nimmt in zahlreichen Bereichen wie der Stimmen- oder Bilderkennung einen sehr großen Stellenwert ein. Aufgabe ist es, die Komplexität der Sensordaten zu minimieren indem Eigenschaften – auch bezeichnet als *Features* – bestimmt werden, die das Signal beschreiben.

Während der Extraktion können die Daten vereinfacht, transformiert oder zur besseren Interpretation sogar erweitert werden. Normalerweise werden für numerische Daten ein-

fache statistische Parameter wie die Standardabweichung, das Minimum oder Maximum, oder der Mittelwert genutzt. Bei nominalen als auch ordinalen Daten[11], müssen jedoch alternative Methoden entwickelt werden.
Dieser Vorgang ähnelt sehr stark dem der Stimmen- oder Bilderkennung, aber die anfallenden Daten sehen in diesem Fall anders aus: die gesammelten Informationen können jeweils anderen Anforderungen genügen. Dies setzt eine sehr differenzierte Behandlung voraus, was die Komplexität des Systems steigern kann, da unter Umständen viele unterschiedliche Methoden zur Extraktion der Features zur Verfügung gestellt werden müssen.

In der Literatur wird auf vielfältige Weise dargelegt, dass die Merkmalserkennung eine Schlüsselrolle bei der Erkennung der Situation einnimmt und somit einen sehr starken Einfluss auf die Güte der Ergebnisse ausübt:

> „The quality of feature selection/extraction limits the performance of the overall pattern recognition system. [...] Thus feature extraction is the most crucial step.“ [GB00, S. 411]

Deshalb sollte das Wissen über die zur Verfügung stehenden Sensoren genutzt werden, um die zur Umsetzung einer entsprechenden Applikation benötigten Features auszuwählen und in Folge dessen die Ergebnisse der Kontexterkennung zu verbessern. Dies ermöglicht, neben der Reduzierung der Komplexität der vorhandenen Rohdaten, zusätzlich eine Optimierung der genutzten Ressourcen.

Klassifikation und Aggregation Die Kontextgewinnung wird fast immer mit einem Klassifikationsprozess gleichgesetzt durch den ein Satz Eingangsdaten einem Kontext zugeordnet wird. Aufgabe der Klassifikation ist es, bekannte Muster im Feature-Raum zu identifizieren und einer bestimmten Klasse von Objekten zuzuordnen. Ein solcher Feature Vektor kann zu einer Vielzahl von Klassen gehören – mit einer jeweiligen Wahrscheinlichkeit.

Prinzipiell wird bei der Klassifikation zwischen kontrollierter und unkontrollierter Klassifikation unterschieden. Der Unterschied besteht in der Vorgabe der Klassen: bei der kontrollierten Klassifikation wird diese anhand bekannter Klassen durchgeführt, wohingegen bei der unkontrollierten Klassifikation anhand der Eingabedaten Klassen selbständig gefunden werden.

[11] Der Informationsgehalt von Daten hängt wesentlich von ihrem Messniveau ab. Daten auf einer Nominalskala bieten lediglich die Möglichkeit, sie auf ihre Gleichheit hin zu unterscheiden (z.B. Namen der gestarteten Programme des Nutzers). Daten auf Ordinalskalenniveau bieten zusätzlich die Möglichkeit einer Rangordnung.

Dabei kommen unterschiedliche Klassifikationsalgorithmen zum Einsatz, deren Auswahl meist von den Anforderungen an die Kontexterkennung, oder der Art der zu verarbeitenden Sensordaten beeinflusst wird.

Bei diesen Algorithmen zur Kontextgewinnung handelt es sich beispielsweise um C4.5 für die Erkennung von Aktivitäten [BI04], Bayes'sche Netze [KKP+03], Hidden Markov Models [CP98], neuronale Netze oder unscharfe Logik.

Auch die Gewinnung neuer Erkenntnisse aus bereits bestehenden Kontexten wird durch die Verwendung von Verfahren ermöglicht, die die Verknüpfung klassifizierter Daten unterstützen (vgl. [CXC+05, CGK05]).

Fusion Bei der *Kontextfusion* werden im Gegensatz zur Aggregation hauptsächlich gleichartige Daten verwendet, um die Genauigkeit und Zuverlässigkeit der Ergebnisse zu erhöhen. Bei diesen Daten kann es sich sowohl um Kontexte als auch um Sensordaten handeln, aus denen sie abgeleitet werden.

Für den Prozess der Fusion unkorrelierter Daten werden zum Beispiel das logische UND, ODER, K aus N, Mehrheitsentscheid oder Neyman-Person [TS80] genutzt. Auch speziell entwickelte Algorithmen sind in der Literatur zu finden (vgl. [CA98]). Für einen vollständigeren Überblick sei an dieser Stelle allerdings auf [CA98] verwiesen.

2.3.3 Herausforderungen im Umgang mit Kontext

Auch nach einwandfreier Erzeugung von Kontext ist zu erwarten, dass sich im praktischen Umgang mit den Ergebnissen einige Herausforderungen ergeben:

- *Inkonsistenz*: Wenn Kontext-Informationen von unterschiedlichen Quellen oder Sensoren erzeugt werden, so kann es zu erheblichen Inkonsistenzen kommen.[12]
- *(Un)genauigkeit*: Vor allem Daten, die von physikalischen Sensoren gewonnen werden, können von unterschiedlicher (Un)genauigkeit geprägt sein. Dies betrifft vor allem deren Präzision als auch Akkuratesse.
- *Verlässlichkeit*: Die Quellen zur Gewinnung der Kontext-Informationen – egal ob es sich dabei um externe Quellen oder Sensoren handelt – können prinzipiell jederzeit ausfallen und stellen somit ein Risiko dar.

[12] Inkonsistenzen können einen unterschiedlichen Charakter aufweisen: sie könnten sich beispielsweise auf unterschiedliche geographische Regionen beziehen oder bereits veraltet sein.

- *Privacy*: Auch die Frage nach der Privacy ist von großer Bedeutung. Welcher Service darf welche Kontex-Informationen kennen und/oder verarbeiten.

Bei der Implementierung von Services, der Nutzung von Kontext-Informationen und dem Betrieb pervasiver Anwendungen müssen obenstehende Aspekte beachtet werden. In Kapitel 4 dieser Arbeit wird deshalb – sowohl implizit als auch explizit – auf deren Bedeutung beim Design des Prototyps eingegangen.

2.3.4 Architektur kontextsensitiver Systeme

Kontextsensitive Systeme bilden die Grundlage für Pervasive Computing – sie verarbeiten Sensordaten und leiten daraus high-level Kontext-Informationen in Bezug auf die aktuelle Situation ab. Diese Kontext-Informationen werden für pervasive Systeme benötigt, um eine situationsabhängige Anpassung der Software an die speziellen Bedürfnisse der Anwender und ihrer Tätigkeiten vorzunehmen.

Ein Blick auf die Architektur der meisten bekannten kontextsensitiven Anwendungen verdeutlicht die unterschiedlichen Auffassungen und Definitionen von Kontext und Kontextsensitivität. Wie in Abbildung 2.11 zu sehen ist, bestehen kontextsensitive Anwendungen, wie wir sie heute kennen, zumeist aus lediglich drei eindeutig differenzierbaren Schichten. Diese sind der *Sensordaten Layer*, der *Kontext Layer*, der heute auch oft Kontext-Middleware genannt wird, und der *Anwendungs Layer*.

Der *Sensordaten Layer* als unterste Schicht stellt eine Ansammlung heterogener Dienste und Komponenten dar, aus denen Basiskontextinformationen in einfacher und unstrukturierter Form bezogen werden können. Dazu zählen vor allem geräte- und betriebssystemspezifische APIs, die die Schnittstelle zu den physikalischen Sensoren des mobilen Gerätes bilden. Auch Web und Bluetooth Services können als Dienst genutzt werden, um Informationen über die Umgebung des Gerätes oder des Benutzers zu erhalten. Ein einfaches Beispiel für die sinnvolle Verwendung von Services wäre die Anbindung einer externen GPS-Maus, die über einen Bluetooth Service Ortsinformationen liefert.

Wie die Komponenten des Sensordaten Layers zeigen, kann der Kontext mit Hilfe von Daten aus verschiedenen Informationsquellen angegeben werden. Der *Kontextverarbeitungs Layer* muss zunächst von den unterschiedlichen Dienstarten abstrahieren, damit die vorhandene Heterogenität im Sensordaten Layer überwunden wird. Aufgabe der *Sensordaten-Abstraktion* ist es vor allem, eine abstrakte Dienstbeschreibung zu spezifizieren. Die Beschreibung und Verwaltung homogener Dienste in einer verteilten mobilen Umgebung ermöglicht die Anwendungen auf dem *Anwendungs Layer* eine transparente Nutzung von Diensten.

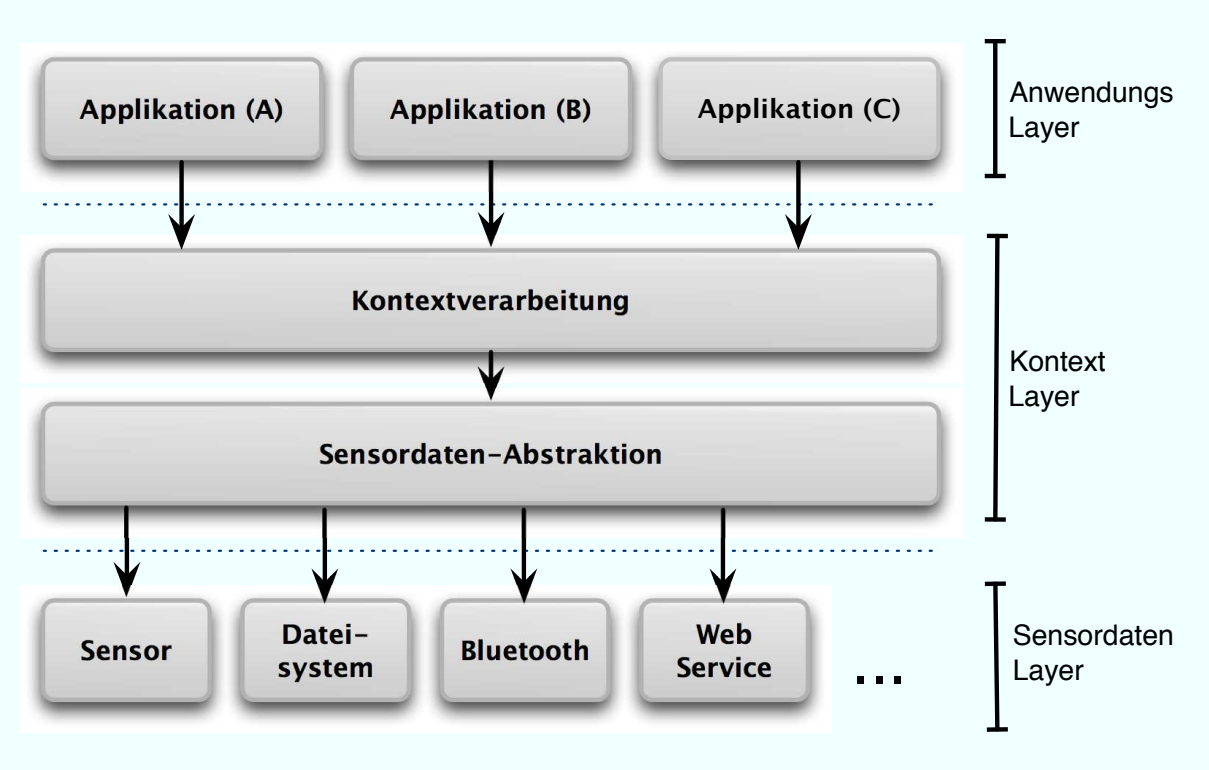


Abbildung 2.11: Einfache Architektur heutiger kontextsensitiver Anwendungen

Die Architektur dieser Anwendungen und damit auch der Systeme, die aus diesen Anwendungen bestehen, besteht aus Schichten, die unterschiedliche Funktionen übernehmen, sich allerdings oft nicht eindeutig voneinander trennen lassen. Meist fällt auf, dass das (semantische) Kontextmodell, welches Teil des Kontextverarbeitungsblockes ist, den zentralen Kern des Systems darstellt. Es beeinflusst sowohl die Ausgestaltung der Kommunikation zwischen diesen Anwendungen, als auch die Funktionalität und Benutzerschnittstellen der Anwendungen, die auf ihm beruhen.

2.3.4.1 Serviceorientierte Architekturen

Eine mögliche Umsetzung einer kontextsensitiven Anwendung im Sinne des Pervasive Computing besteht in dem Aufbau einer *serviceorientierten Architektur* (SOA). Viele der Arbeiten, die sich mit der infrastrukturellen Frage des Pervasive oder Ubiquitous Computing auseinandersetzen, benutzen Aspekte des Architekturmusters der SOA. Auch die Nutzung von Web Services würde zu einer solchen Architektur führen. Deshalb sollen in diesem Abschnitt auf einige grundlegende Aspekte serviceorientierter Architekturen eingegangen werden, um zu klären, warum diese zum Aufbau pervasiver Systeme besonders geeignet erscheinen.

Grundlagen Eine serviceorientierte Architektur stellt eine flexible und adaptierbare Architektur aus dem Bereich der verteilten Systeme dar. Sie baut auf einigen grundlegenden Eigenschaften auf, die auch als Elemente einer serviceorientierten Architektur bezeichnet werden.

Dazu gehören zunächst die Begriffe *Einfachheit*, *Sicherheit* und *Akzeptanz*, die als fundamental für die Umsetzung einer SOA erachtet werden. Darauf bauen die Elemente *Verteilung*, *lose Kopplung*, *Standardisierung* und *Prozessorientierung* auf. Das bedeutet, dass Komponenten einer solchen Struktur – Programme, Daten, Hardware – im Sinne eines verteilten Systems verstanden werden. Zudem ist eine lose Kopplung zwischen diesen Komponenten sinnvoll, da das System sich ständig ändernden Anforderungen und Gegebenheiten unterliegen kann. Dabei sollte Wert gelegt werden, auf die Verwendung von Standards: diese unterstützen die flexible Interaktion innerhalb einer SOA und gewährleisten die Funktionalität auf unterschiedlichen Plattformen. Außerdem soll jeder einzelne Service idealerweise als kleinstmöglicher Schritt eines größeren Prozesses verstanden werden, um deren Vorteile bei der Orchestrierung (der Komposition) von Services zu einem größeren Ganzen voll ausschöpfen zu können.

Diese Charakteristika einer SOA entsprechen in vielen Punkten denen des Pervasive Computing: auch hier wird die Einfachheit, ein hohes Maß an Sicherheit und die Standardisierung gefordert. Ohne die Verwendung von Standards könnte ein Zusammenspiel unterschiedlichster Geräte, Softwareplattformen, Sensoren, Rechnersysteme und Services nicht funktionieren. Auch die Forderung nach einer losen Kopplung ist nützlich, um flexibel auf sich ändernde Anforderungen reagieren zu können.

Eine SOA ist demnach sehr gut geeignet zum Aufbau einer pervasiven Infrastruktur. Daher werden Teile der Architektur bei der Umsetzung des Prototypen zum Tragen kommen.

Web Services Web Services sind das derzeit populärste Konzept einer SOA, daneben existieren jedoch auch andere geeignete Konzepte wie beispielsweise CORBA oder RMI. Bei der Nutzung von Web Services können allerdings in Sachen Sicherheit und Prozessorientierung, in Abhängigkeit der konkreten Implementierung, Herausforderungen auftreten, die allerdings im Rahmen dieser Arbeit nicht erörtert werden sollen.

Prinzipiell handelt es sich bei einem Web Service um eine Anwendung, die mit einer eindeutigen URI identifziert werden kann und deren Schnittstellen als XML-Artefakte definiert sind. Solch ein Web Service dient nicht zur direkten Kommunikation mit dem Menschen, sondern der Interaktion mit anderen Software-Agenten unter der Verwendung XML-basierter Nachrichten, die über internetbasierte Protokolle ausgetauscht werden.

Vorteile Gegenüber anderen Ansätzen bietet eine mittels SOA entworfene Infrastruktur einige Vorteile bei der Umsetzung situationsabhängiger Systeme:

- Die Standardisierung der genutzten Komponenten, Datenformate und Netzwerkprotokolle resultiert im besten Fall in der Unabhängigkeit von Hardware, Betriebssystem und Programmiersprache.
- Durch die Nutzung von Komponenten kann die Wartung und Weiterentwicklung erheblich erleichtert werden.
- Ressourcen können zwischen unterschiedlichen Diensten geteilt werden.
- Bei den Services handelt es sich um spezialisierte Einheiten, die nur einen minimalen Funktionsumfang besitzen, dies steigert die Wiederverwendbarkeit.
- Durch Komposition von Services können neue Services entstehen.
- Falls ein Service während der Laufzeit ausfällt, können andere mit gleicher oder ausreichend ähnlicher Funktionalität genutzt werden.

2.4 Mobile Endgeräte

Die mobilen Endgeräte werden immer kleiner und vielseitiger. Die Integration von Zusatzgeräten wie MP3-Player, Internet und Kamera ist bereits in fast allen Neugeräten zu finden. Zunehmend werden aber auch Sensoren integriert: GPS, um die aktuelle Position des Geräts zu erfahren, Beschleunigungssensoren, um die Lage des Geräts zu erkennen, Helligkeitssensoren, um bei Bedarf Strom zu sparen, und neuerdings gibt es bereits Geräte, die einen Kompass integrieren. Der Trend ist klar zu erkennen: die Geräte stellen in zunehmendem Maße mehr Schnittstellen zu Informations- und Kommunikationskanälen sowie zur Kontexterkennung bereit.
Noch werden die Geräte bevorzugt für ihre ursprüngliche Verwendung wie Telefonate und den Versand von Kurznachrichten genutzt, aber durch die Kombination mit standortrelevanten Informationen und Diensten oder der Erfassung des Kontextes, in welchem sich ein Nutzer befindet, können diese Geräte auf unterschiedliche Weise erweitert und genutzt werden, beispielsweise für neuartige Spielerfahrungen.

Die einsetzbaren Geräte bieten unterschiedliche Vor- und Nachteile, wobei die größte nachteilige Gemeinsamkeit vermutlich die geringe Rechenleistung und Bildschirmgröße ist. Die zunehmende technische Verbesserung lässt jedoch vermuten, dass diese in nicht allzu ferner Zeit in ausreichender Größe vorhanden sein werden und die Nachteile immer

stärker zurücktreten, ähnlich wie es bei der Entwicklung des Computers zu beobachten ist.

Für die Realisierung der Zielsetzung dieser Arbeit, ein positions- und situationsabhängiges Adventurespiel zu entwickeln, wird ein mobiles Gerät benötigt, dass den Zugriff auf integrierte Sensoren und eine entsprechende Programmierumgebung bietet sowie unterschiedliche Kommunikationskanäle zur Verfügung stellt. Auf dem Markt ist bereits eine größere Anzahl an Geräten zu finden, die diese Anforderungen erfüllen. Daher werden die nachfolgenden Betrachtungen beispielhaft am *Nokia N95*, dem *Apple iPhone 3G* und dem *HTC G1* durchgeführt.

Ein entscheidender Aspekt unterscheidet diese Geräte: das auf ihnen installierte Betriebssystem. Auf dem N95 ist das Symbian S60 Betriebssystem zu finden. Dort steht es dem Entwickler frei mit welcher Programmiersprache er entwickeln möchte: Python, Java ME oder Carbride C++. Den besten Zugriff auf integrierte Sensoren erhält man mit C++, doch ist der Aufwand, der für die Entwicklung einer solchen Applikation betrieben werden muss, für einen Prototypen nicht zu rechtfertigen. Der Zugriff auf die integrierten Sensoren, ein GPS-Sensor und ein Beschleunigungssensor, kann durch jede dieser Programmiersprachen realisiert werden. Bei ersten prototypischen Implementierungen stellte sich allerdings heraus, dass das GPS nicht die gewünschte Verfügbarkeit und Zuverlässigkeit bietet und bei den meisten Versuchen bis zu 10 Minuten benötigte, um eine Position zu bestimmen.

Eine Alternative ist das iPhone. Auf diesem läuft das iPhone OS. Dieses Betriebssystem hat vier Abstraktionslayer: Core OS, Core Services Media Layer und Cocoa Touch. Der Zugriff auf alle verfügbaren Sensoren kann auf diesem System leider nicht realisiert werden, da die Programme in einer Sandbox laufen. Das iPhone kommt von Hause aus mit einem Beschleunigungssensor, einem GPS-Sensor, einem Annäherungssektor und einem Umgebungslichtsensor. Der Zugriff auf die wichtigsten der beiden, dem Beschleunigungs- und GPS-Sensor, kann jedoch recht bequem über das iPhone SDK realisiert werden. Das iPhone arbeitet zudem mit A-GPS: es verwendet Daten der Mobilfunknetze, um die nächstgelegenen Satelliten schnell zu finden, sodass sich der Standort sehr viel schneller als mit normalem GPS ermitteln lässt. Außerdem bietet es die Möglichkeit der Lokalisierung über WLAN. Die Kombination der Positionsbestimmung anhand von GPS-Signalen, Wi-Fi-Hot-Spots und Mobilfunk-Sendemasten kann es ermöglichen, das umzusetzende Spiel sowohl für Indoor- als auch Outdoor-Szenarien umzusetzen.

Bei dem G1 handelt es sich um das erste Handy mit einem quelloffenen Betriebssystem. Android ist eine Plattform die von der Open Handset Alliance entwickelt wurde. Es baut auf einem Linux-Kernel auf, der die Speicher- und Prozessverwaltung und die Netz-

werkkommunikation sicherstellt. Außerdem bildet er die Hardwareabstraktionsschicht für den Rest der Software. Zum Programmieren wird eine Vielzahl von Javaklassen und -schnittstellen zur Verfügung gestellt. Anwendungen für die Android-Plattform werden also ausnahmslos in Java geschrieben, bei geschwindigkeitskritischen Berechnungen wird jedoch intern auf native C/C++ Bibliotheken zugegriffen. Ein großer Vorteil dieses Geräts ist, dass die Hardware frei programmierbar ist. Unter anderem ergeben sich dahingehend sehr viele Möglichkeiten, da in diesem Smartphone GPS, Beschleunigungssensoren und ein digitaler Kompass integriert wurden. Hier in Deutschland ist es allerdings noch nicht erschienen.

2.4.1 Zusammenfassung

Um eine kontextsensitive Adaption der Anwendung zu ermöglichen, werden zur Berechnung der Position und Situation neben GPS-Daten auch weitere Sensordaten benötigt. Da bei dem N95 die gewünschte Genauigkeit der Positionsbestimmung nicht erreicht werden kann, muss dieses Gerät ausgeschlossen werden. Das G1 würde zwar die notwendigen Voraussetzungen erfüllen und zusätzlich sogar weitere Sensoren integrieren, die man nutzen könnte, doch in Deutschland wird es erst Anfang 2009 erscheinen.

Das iPhone bietet dagegen alles, was benötigt wird, um das Szenario umzusetzen. Die von Apple zur Verfügung gestellten Frameworks erlauben allerdings keinen uneingeschränkten Zugriff auf alle verfügbaren Sensoren. Daher werden sich die folgenden Betrachtungen und Implementierungen auf die Nutzung von GPS- und Beschleunigungssensoren im Allgemeinen und das iPhone im Speziellen beschränken.

Kapitel 3

Pervasive Spiele

Zu spielen ist eine soziale Aktivität, die es ermöglicht, sich mit Personen in seiner Umwelt zu messen und Freundschaften aufzubauen. Spielen ist ein Weg kognitive und motorischen Eigenschaften zu verbessern. Spielen ist eine Möglichkeit der realen Welt zu entkommen und der Fantasie freien Lauf zu lassen – eine Art der Entspannung. Spielen ist eine Form des Lernens, sowohl bei der Entwicklung des strategischen und taktischen Verständnisses [Cra84] als auch der Bildung. Man lernt dabei nicht nur zu gewinnen sondern auch zu verlieren.

Eine echte Spielerevolution war der Einzug des Computerspiels. Anfangs handelte es sich dabei noch um sehr einfache Spiele wie Pac Man oder Space Invaders, doch auch diese Spiele brachten neue Spielmöglichkeiten und Spielerfahrung. Über die Jahre wurden die Grafiken besser und die Spiele komplexer [Cra84]. Mittlerweile sind viele Spiele über das Internet in Massive-Multiplayer-Online-Role-Playing-Games (MMORPG) spielbar. In vielen Spielen tauchen die Spieler dabei in eine abenteuerliche, phantastische Welt ein, in denen sie am Computer virtuelle Welten erforschen oder Rätsel lösen, um auf abenteuerlichen Wegen das Ziel des Spiels zu erreichen.

Viele dieser Spiele, bilden dabei die Wirklichkeit ab, aber nur wenige beziehen sie mit ein. Seit einigen Jahren gibt es allerdings das Genre der *Pervasiven Spiele*, die es sich zur Aufgabe gemacht haben, eine Brücke zwischen realer und virtueller Welt zu schlagen. Sie versuchen, mit den spezifischen Eigenschaften mobiler Geräte – deren hohe Kommunikationsfähigkeit und den Schnittstellen zur Nutzung weiterer Technologien, wie z.B. GPS – eine neue Form des Spielerlebnis zu schaffen.

Im Gegensatz zu klassischen Computerspielen, die in der virtuellen Welt stattfinden, sind beim Pervasiven Spielen beispielsweise physikalische Bewegungen und soziale Interaktionen mit anderen Benutzern in der physikalischen Welt nötig [MCMN05]. Der Ablauf des

Spiels ist eine Mischung aus Physikalität, Mobilität und Virtualität: der Spieler nutzt die physische Welt als Spielfeld und kann dennoch die Vorteile und Möglichkeiten der technischen Geräte und der virtuellen Welt nutzen. Auf diese Weise können gänzlich neue Spielkonzepte realisiert werden. Aber auch eine Erweiterung traditioneller Spiele um Komponenten des Pervasive Computing ist denkbar, um Teilaspekte aus der Sicht des Nutzers zu vereinfachen.

3.1 Ausgewählte Beispiele Pervasiver Spiele

Um den aktuellen Stand der Entwicklungen im Bereich der pervasiven Spiele besser beurteilen zu können, sollen hier einige der Vertreter dieses relativ neuen Genres vorgestellt und bewertet werden.

3.1.1 Hitchers

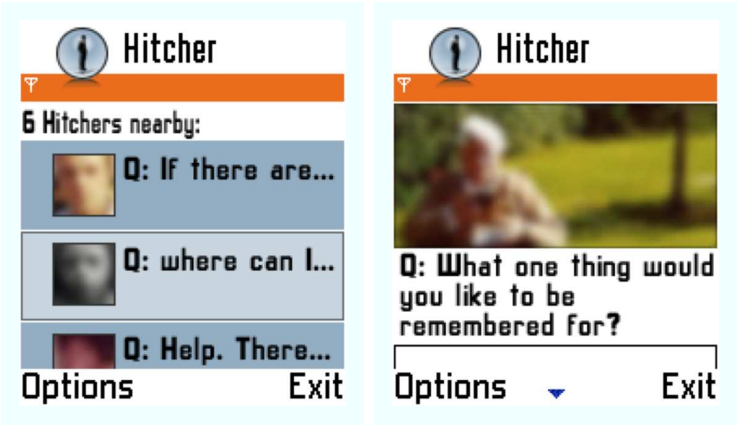
Bei *Hitchers*[1] handelt es sich um eine prototypische Implementierung eines positionsabhängigen Spiels. Es wurde im Rahmen eines von der EU geförderten Projekts IPerG[2] umgesetzt um technologische Aspekte der Positionsbestimmung mittels Mobilfunk-Ortung zu demonstrieren.

In diesem Spiel erstellen Spieler digitale Anhalter, die mit einem Namen, einem Ziel und einer Aufgabe versehen werden. Diese können dann in der aktuellen Mobilfunkzelle des Spielers platziert und von anderen Spielern gefunden werden, wenn diese sich in der entsprechenden Zelle befinden. In Abbildung 3.1(a) wird visualisiert wie dies dem Spieler präsentiert wird. Alle Anhalter, die sich in seiner Nähe befinden, werden in der Applikation aufgelistet.

Er hat nun die Möglichkeit mit einem dieser Hitcher zu agieren – dessen Fragen zu beantworten oder Missionen zu erfüllen (vgl. Abbildung 3.1(b), S. 45). Anschließend kann er den Hitcher in einer beliebigen Mobilfunkzelle wieder aussetzen. Damit dieser von einem anderen Spieler gefunden werden kann, wird diese Zelle mit einem aussagekräftigen Namen beschrieben. Auf der Webseite des Spiels können interessierte Spieler dann erken-

[1] Weite Informationen zum Konzept des Prototypen können der Webseite `http://www.amutualfriend.co.uk/html/hitchers_about.html` entnommen werden

[2] Bei IPerG (Integrated Project on Pervasive Gaming) handelt es sich um ein von der EU gefördertes Projekt das am 01. September 2004 startete und bis Ende Februar 2008 verfolgt wurde. Ziel war die Erforschung gestalterischer, technischer und ökonomischer Aspekte des Pervasive Gaming. Weitere Informationen und Ergebnisse des Projekts können auf der Webseite `http://www.pervasive-gaming.org/` gefunden werden.



(a) Hitchers in der Nähe (b) Hitchers stellt eine Frage

Abbildung 3.1: Screenshots des Programm *Hitchers* (Quelle: [27])

nen, wo sich der Hitcher momentan befindet damit dieser zum nächsten Ziel transportiert werden kann.

Der digitale Anhalter reist somit von Person zu Person, von Handy zu Handy und von Zelle zu Zelle. Dabei sammelt er Informationen, die er durch die Beantwortung der Fragen und durch das Erfüllen der Missionen erhält. Durch zellbasierte Ortung mobiler Geräte wird demnach ein ortsabhängiges Spielen ermöglicht (vgl. [DBT+06]).

3.1.2 MobiMissions

Das Spielprinzip von *MobiMissions* basiert auf dem des zuvor vorgestellten Prototyps Hitchers. Die Spieler bewegen sich auch hier zwischen unterschiedlichen Mobilfunkzellen. Dort können sie Missionen finden, die von anderen Spielern erstellt wurden. Diese können aufgehoben und beantwortet werden. Die aktuelle Zelle in der diese Missionen beantwortet werden, wird zur Zelle, in der diese durch den nächsten Spieler gefunden werden können. Die zentrale Funktionalität des Spiels basiert auf der Erstellung von Missionen. Diese werden durch Kombination von Fotos und Text auf einem Mobiltelefon erstellt und in der aktuellen Mobilfunkzelle des Spielers platziert. In Abbildung 3.1.2 wird eine Detailansicht einer Mission aufgezeigt. Für die Erstellung und die Beantwortung als auch der Qualität ihrer Missionen und Antworten bekommen die jeweiligen Spieler Punkte, anhand dessen sich die Leistungen der Spieler

Abbildung 3.2: Detailansicht einer Aufgabe bei MobiMissions (Quelle: [10])

untereinander vergleichen und somit eine Rangliste generieren lässt.

Dieses Prinzip der Bewertung wurde jedoch so umgesetzt, dass die Nutzer sich im Nachhinein im Internet anmelden müssen, um zu erfahren, dass Missionen beantwortet wurden. Es erfolgt keine explizite Information auf ihr Mobilgerät. Eine Bewertung der Antworten war somit nicht zwingend erforderlich. Zudem war bei der Auswertung des Projekts der Trend zu erkennen, dass die Spieler ihre Missionen hauptsächlich zuhause erstellten (vgl. [10]) – die Verfügbarkeit der Missionen sich also auf bestimmte Orte zentrierte.

3.1.3 REXplorer

Bei dem *REXplorer*[3] handelt es sich um einen Touristenführer in der Stadt Regensburg, der es jungen Erwachsenen ermöglichen soll, an einer interessanten, interaktiven Führung teilzunehmen. Durch physische, einstudierte Bewegungen des dort verliehenen mobilen Gerätes werden Zaubersprüche ausgeführt, die Geister in bekannten Gebäuden der Stadt erwecken. Diese haben einiges zu erzählen.

Abbildung 3.3: Interaktion mit dem Mobilgerät (Quelle: [28])

Die Tipps der Geister führen den Spieler durch die Stadt und somit dem endgültigen Ziel entgegen. Um die für den Spielverlauf benötigten Gegenstände zu erhalten, müssen beispielsweise Fotos geschossen werden. Diese werden katalogisiert und in aufbereiteter Form in Gestalt eines Weblogs im Internet zugänglich gemacht. Auch während des Spiels kann darauf zugegriffen werden – zum Einsehen des aktuellen Spielstatus.

Im Gegensatz der zuvor vorgestellten Prototypen, wird hierbei die Interaktion mit anderen Spielern erforderlich. Es kann passieren, dass ein Geist den Spieler fortschickt, falls er nicht genug Power hat – also nur alleine unterwegs ist. Das Prinzip der Interaktion basiert auf der Analyse aufeinanderfolgender Bilder der Kamera, welche durch die Daten von Beschleunigungssensorgen ergänzt werden.

[3] Die Webseite des Projekts ist unter `http://wiki.caad.arch.ethz.ch/Research/REXplorer` zu finden.

3.1.4 Your Way Your Missions

Der Einfluss des Prototyps Hitchers lässt sich auch an dem Projekt *Your Way Your Missions* erkennen, welches wiederum auf dessen Framework basiert und zudem das Spielerlebnis von MobiMissions unter Einbeziehung von Routen nutzt [CB07]. Mit diesem Prinzip soll auch Spielern das Spielen ermöglicht werden, die sich mit dem Bus oder mit der Bahn vorwärts bewegen. Entscheidende Änderungen wurden auch im Bereich der Positionsbestimmungen vorgenommen. Die Bestimmung erfolgt hierbei nach dem Prinzip des GPS. Jede Mission bekommt somit eine exakte Position zugeordnet.

Bei Spielantritt gibt der Spieler dem Server seine voraussichtliche Route bekannt, um eine individuelle Adaption der Missionen serverseitig vornehmen zu lassen. Hierbei werden Missionen gefunden, die sich auf oder in der Nähe der Route befinden. Diese werden dem Spieler im Laufe seiner Fahrt zugänglich gemacht.

3.1.5 Zusammenfassung

Aus der Analyse vorhandener pervasiver Spiele lassen sich viele Gemeinsamkeiten erkennen. Der wichtigste ist eindeutig die Verknüpfung von physikalischer und virtueller Umgebung.

Dies wird durch alle zuvor vorgestellten Systeme durch eine ortsabhängige Implementierung umgesetzt. Sie nutzen die Position des Anwenders als zentrales Konzept zur Gestaltung des Spiels: bei Erreichen bestimmter Orte werden Aktionen innerhalb des Spiels ausgelöst indem dem Nutzer spezifische Aufgaben gestellt werden. Zu Beginn der Entwicklungen wurde die Positionsbestimmung anhand der Mobilfunkortung durchgeführt. Im Laufe der Zeit – mit Etablierung von GPS-Sensoren in mobilen Geräten – wurde dieses von satellitengestützten Verfahren abgelöst. Allerdings beschränkt sich die Nutzung des Ortes bei aktuellen Implementierungen auf den Outdoor-Bereich: das Vorhandensein eines GPS-Signals wird vorausgesetzt um am Geschehen des Spiels teil zu haben. Dies widerspricht allerdings dem Prinzip des Pervasive Computing, dass mit „Alles, immer, überall“ umrissen wurde. Die Umsetzung des Prototypen soll daher explizit auch die Fortführung des Spiels unterstützen, wenn kein GPS verfügbar ist.

Durch die Nutzung der aktuellen Position des Nutzers erfolgt demzufolge eine Adaption der Applikation an den Kontext des Ortes. Weitere Adaptionen in Bezug auf die aktuelle Situation des Nutzers oder dessen Umgebung durch sensorische Wahrnehmung sind allerdings kaum zu finden. Lediglich bei dem Spiel REXplorer werden Gesten des Nutzers interpretiert um eine Steuerung des Spiels zu ermöglichen und bestimmte Aktionen auszulösen. Eine umfassendere Implementierung der automatisierten Kontexterkennung

und -adaptierung stellt daher einen wünschenswerten Aspekt bei der Umsetzung eines positions- und situationsabhängigen Prototypen für pervasive Spiele dar.

Alle Systeme eint zudem die Mobilität des Nutzers. Infolgedessen handelt es sich auch bei der drahtlosen Kommunikationsfähigkeit der Clients um einen weiteren zentralen Aspekt der Umsetzung. Diese kommunizieren in einer verteilten Architektur mit dem Spieleserver über Web Services oder andere entsprechende Schnittstellen. Der Server übernimmt dabei in der Regel die Verwaltung der Nutzer als auch der im Spiel zu erfüllenden Aufgaben und der Spielelogik.

Für den Entwurf eines pervasiven Spiels lassen sich infolgedessen einige Schlüsselmerkmale identifizieren:

- **Mobilität**: Die Spieler eines pervasiven Spiels können sich in der realen Welt frei bewegen. In Abhängigkeit der Implementierung kann die Größe des Spielfelds variieren: dieses kann beispielsweise auf ein bestimmtes Areal oder eine Stadt beschränkt sein. Der Gebrauch mobiler Geräte und der Einsatz verschiedener Positionierungstechniken erlaubt die Lokalisierung des Spielers und seiner Aktionen auf diesem Spielfeld. Aufgrund dieser Mobilität müssen beim Entwurf des Spiels einige Risiken berücksichtigt werden: plötzliche Verbindungsabbrüche, ungenaue Positionierungen, Design der Benutzerschnittstellen u.s.w.

- **Konnektivität**: Die Konnektivität der Geräte betrifft sowohl die Kommunikation mit einer zentralen Instanz um aktuelle Daten zu synchronisieren als auch die spontane Vernetzung mit Geräten anderer Spieler. Um Interoperabilität und Konnektivität zu gewährleisten, müssen allgemeingültige Standards verwendet werden. Zudem muss mit Verbindungsabbrüchen gerechnet werden. Um diese technischen Details vor dem Spieler zu verbergen, sollten benötigte Daten des Spiels bereits lokal auf dem Gerät vorgehalten werden.

- **Ortsspezifität**: Erfahrungen während des Spiels hängen von der Position und dem spezifizierten Areal ab, in welchem das pervasive Spiel durchgeführt wird. Entsprechende Aktionen und Aufgaben innerhalb des Spiels können somit in Abhängigkeit der Position des Nutzers ausgeführt werden.

- **Öffentliche Interaktion**: Das Spiel kann an öffentlichen Plätzen stattfinden und dabei Spieler als auch Nicht-Spieler miteinander vermischen. Ein pervasives Spiel sollte sich demnach auch mit den Fragestellungen auseinandersetzen wie Nicht-Spieler eventuell in die Handlung des Spiels integriert oder explizit ausgeschlossen werden können, um diese in ihrem Handlungsfreiraum nicht einzuschränken.

3.2 Spielkonzept

Das Konzept des Spiels soll prinzipiell auf der Metapher einer *Schnitzeljagd*, basieren: an fest definierten Orten im Raum Berlin werden unterschiedliche Aufgaben sowie Teile der Story versteckt – diese können durch den Nutzer allerdings nicht von einem Ort zu einem anderen transportiert werden. Zudem können weitere Elemente des Spiels durch die Aktivitäten des Nutzers aufgedeckt oder sogar neue erstellt werden.

Im Laufe des Spiels werden durch das System automatisch individuelle Herausforderungen generiert: jeweils zwei Spieler treten in einen direkten Vergleich und müssen eine gestellte Aufgabe bestmöglich erfüllen, um diesen Wettstreit zu gewinnen. Für die Erfüllung der gefundenen Aufgaben als auch der soeben beschriebenen Wettkämpfe, kann der Spieler so genannte *Aktivitätspunkte* erhalten. Diese ermöglichen einen direkten Vergleich der Leistungen der einzelnen Spieler untereinander und kennzeichnen den Fortschritt im Spiel.

Sie honorieren damit implizit die sportliche Aktivität des Spielers. Um dies zu realisieren werden während des Spiels unterschiedliche Sensordaten des mobilen Geräts ausgewertet und kategorisiert, um herauszufinden, was der Spieler macht – sitzt er zu Hause, läuft er durch den Park, fährt er mit dem Rad oder steigt er Treppen.

Alle Aktivitäten die erkannt werden und einen sportlichen Charakter aufweisen, ermöglichen es dem Spieler, Aktivitätspunkte zu sammeln. Auch wenn er in diesem Moment beispielsweise keine spezielle Aufgabe findet oder bearbeitet. Da diese Aufgaben aber auch von der Ausführung spezieller Tätigkeiten – bspw. dem Joggen – ausgelöst werden können, wird der Spieler animiert, aktiv durch das Leben zu gehen und damit im Spiel voranzukommen.

Neben diesen Punkten steht auch die persönliche sportliche Entwicklung des Spielers im Vordergrund. Aus seinen persönlichen Bestzeiten und seinen Lieblingsbeschäftigungen wird ein Profil erstellt. Anhand dieses Profils kann der Server analysieren, welche Spieler gegeneinander in einen Wettstreit treten werden. Auf diese Weise kann gewährleistet werden, dass nur Spieler gegeneinander konkurrieren, die sich auf einem ähnlichen sportlichen Niveau befinden. Ein besonderer Anreiz wird zudem geliefert, wenn man sich auf die Spieler in seiner Nähe konzentriert, um gegen diese anzutreten und somit neue soziale Kontakte zu knüpfen.

Bei der Umsetzung der Konzepte zur Erstellung der ortsbezogenen Informationen soll auf Konzepte und die Funktionalität eines bereits zuvor entwickelten multimodalen Informationssystems – „Boje - Berliner Orte jüdischer Erinnerung“ – zurückgegriffen werden. Dieses System ermöglicht die individualisierte Zusammenstellung von Touren durch einen Nutzer. Diese können durch Kombination unterschiedlicher *Tourbausteine*, die die genaue

Position der Objekte sowie weiterführende Informationen enthalten, erstellt und für unterschiedliche Endgeräte optimiert dargestellt werden.
Diese bereits untersuchten Konzepte sollen durch das Spiel aufgegriffen und erweitert werden, um Autoren zu ermöglichen, die Storyline des Spiels flexibel zu gestalten und sowohl in Abhängigkeit des Ortes als auch bestimmter Situationen zu realisieren.

3.2.1 Die Story

Bei dem Spiel *Sportix* geht es um eine Gruppe von kugelrunden Wesen, die von ihren Artgenossen verstoßen wurden. Ihre Untätigkeit hat sie erst in diese missliche Lage gebracht.

Früher lebten sie in ihrer eigenen kleinen Welt. Sie kannten keine Probleme und hatten den ganzen Tag Spaß. Eigentlich hatten sie immer etwas zu tun. Irgendwann machte ihnen das allerdings keinen Spaß mehr und sie saßen nur zu hause. Sie taten nichts. Doch das ließen sich die anderen nicht bieten. Sie wurden auf die Reise geschickt: wenn sie nicht wieder etwas aktiver würden, dürften sie nicht zurückkommen.

Um also wieder in ihre Welt zurückzukehren, müssen sie lernen, aktiv am Leben teilzunehmen – sie müssen lernen Spaß zu haben und Sport zu treiben. Allein können sie jedoch nicht die Motivation aufbringen, sich aufzuraffen und durch sportliche Aktivitäten die Energie zu sammeln, die sie benötigen, um in ihre Welt zurückzukehren. Sie sind darauf angewiesen, dass jemand sie findet und dazu antreibt sich zu bewegen und endlich wieder nach hause zu kommen.

Momentan sitzen sie faul in der Gegend rum und versuchen sich so gut es geht, vor allen nur erdenklichen Aktivitäten zu drücken. Um das zu gewährleisten, haben sie sich ein gutes Versteck gesucht, wo sie niemand zu finden droht. Zuletzt konnten sie allerdings auf der Webseite `http://boje.f4.fhtw-berlin.de` beobachtet werden – und mit viel Glück sind sie auch heute noch dort anzutreffen.

3.2.2 Spielmodi

Bei diesem Spiel wird zwischen unterschiedlichen Zuständen des Spielens unterschieden. Der Spieler kann wählen, ob er in regelmäßigem Kontakt mit dem Server stehen möchte, oder ob er seine Daten erst synchronisieren will, wenn er sich beispielsweise in einem freien WLAN befindet und dadurch für ihn keine zusätzlichen Kosten entstehen. Um dies zu ermöglichen, ist es erforderlich auf dem Gerät des Nutzers Kopien der Daten des Servers zu hinterlegen, um somit zumindest die grundlegenden Funktionen – das Lösen von Aufgaben und das Erleben der Story – zu ermöglichen. Der Nachteil dessen

besteht allerdings darin, dass keine Interaktion mit weiteren Spielern ermöglicht werden kann: der Spieler weiß nicht, wer sich gerade in seiner Nähe befindet und der Server kann, in Ermangelung der aktuellen Daten des Spielers, keine Wettkämpfe für den Spieler generieren.

Bei einer regelmäßigen Kommunikation mit dem Server stehen dem Spieler darüber hinaus weitere Möglichkeiten zur Verfügung. Er kann stets über die Spieler in seiner Nähe unterrichtet werden und diese beispielsweise in einer bestimmten Disziplin herausfordern. Außerdem stehen aktuelle Daten zur Verfügung: man wird über aktuelle Ereignisse und Orte informiert, an denen andere Spieler vor kurzem neue Bestwerte aufgestellt haben. Die Kenntnis darüber ermöglicht es dem Spieler dementsprechend darauf zu reagieren: er kann versuchen diesen Wert zu übertreffen und so einen neuen Rekord aufzustellen.

Der größte Vorteil betrifft allerdings die automatische Generierung der Wettkämpfe durch den Server: dieser sucht in regelmäßigen Abständen nach aktiven Spielern und ermittelt diejenigen, die auf Grund ihrer Leistungen gegeneinander antreten können. Diese Eigenschaft des Spiels macht es sehr dynamisch und abwechslungsreich und garantiert ein Weiterkommen in sowohl sportlicher als auch spielerischer Hinsicht.

Aus der Sicht des Spielers lassen sich darüber hinaus drei Zustände des Spiels identifizieren: ein Spieler kann sich entweder aktiv am Spiel beteiligen, er kann sich in einem Ruhezustand befinden (Pause) oder er hat das Spiel gar nicht gestartet. Bei dem Wechsel zwischen den einzelnen Zuständen (Starten, Pausieren, Beenden) handelt es sich um ein wichtiges Gestaltungskriterium. Für den Spieler sollte dies möglichst reibungslos vonstatten gehen – er sollte nicht das Gefühl haben, das Spiel überhaupt zu beenden oder zu starten. Des Weiteren darf er durch diesen Prozess in seinen Aktivitäten innerhalb und außerhalb des Spiels nicht eingeschränkt werden. Folgende Punkte sind deshalb von Relevanz:

- Flexible Unterbrechung und Wiederaufnahme des Spiels. Der Spieler soll sich innerhalb des Programms an der selben Stelle wiederfinden, an der es beendet wurde.
- Visualisierung des Status eines Spielers bei seinen Mitspielern.
- Synchrone und asynchrone Spielzustände: die Spieler können sowohl simultan gegeneinander antreten oder innerhalb einer bestimmten Zeitspanne die Aufgabe bewältigen.
- Eine Beendigung der Kommunikation oder des Spiels sollte bei laufenden Aufgaben entsprechend berücksichtigt werden.

3.3 Anwendungsszenarien

Auf Grundlage des in Abschnitt 2.3.2 beschriebenen generalisierten Konzepts der Kontextgewinnung lassen sich eine Vielzahl mobiler Informationssysteme entwickeln, die in Abhängigkeit der aktuellen Position und Situation eines Nutzers spezifische Informationen zur Verfügung stellen. Aus der Sicht des Nutzers lassen sich prinzipiell zwei Anwendungsszenarien unterscheiden: die Umsetzung des klassischen Szenario des Ubiqutious und Pervasive Computing in dem die aktuelle Situation des Nutzers automatisch aus der mit Sensoren instrumentierten Umgebung des Nutzers gewonnen wird und der Nutzung eines mobilen Gerätes, welches die Schnittstelle zwischen den darin integrierten Sensoren und dem Benutzer darstellt, um auf dessen Situation entsprechend zu reagieren.

3.3.1 Pervasive Infrastruktur

Bei einer entsprechend verfügbaren Infrastruktur benötigt der Nutzer kein zusätzliches Endgerät: anhand seiner Position werden die Daten der in der Umgebung verfügbaren Sensoren auf einer zentralen Instanz gesammelt und ausgewertet. Entsprechend der aktuellen Klassifikation der Situation können auf Aktoren (beispielsweise Displays) in der Reichweite des Anwenders entsprechende Informationen ausgegeben oder Services bereitgestellt werden. Die Dynamik der Infrastruktur setzt allerdings voraus, dass die Auswertungen unabhängig von der Verfügbarkeit spezieller Sensordaten vonstatten gehen kann.

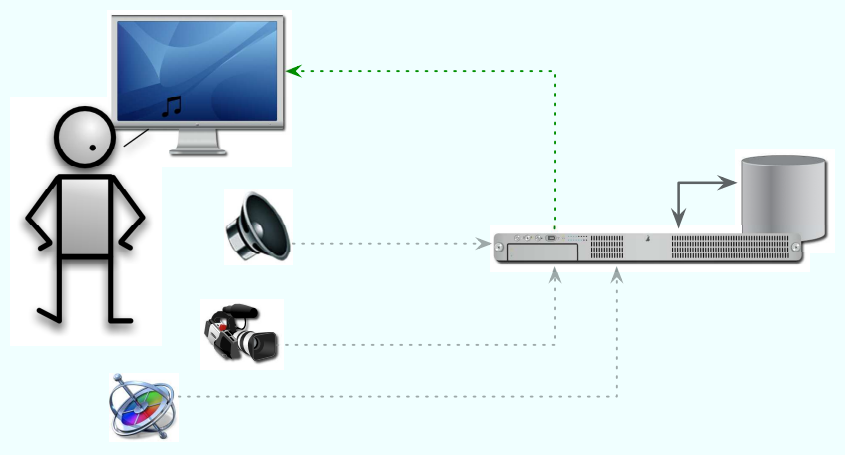


Abbildung 3.4: Anwendungsszenario: Einsatz des Systems in einer pervasiven Infrastruktur

3.3.2 Mobile Endgeräte als Schnittstelle

Im Unterschied zur Umsetzung einer pervasiven Infrastruktur, bei der entsprechend Sensoren in der Umgebung des Nutzers vorhanden sein müssen, bezieht sich die Umsetzung mit Hilfe eines mobilen Gerätes auf die damit verfügbaren Sensordaten. Diese müssen sich jedoch nicht nur auf physikalische Sensoren beschränken – auch die Nutzung logischer oder virtueller Sensoren (beispielsweise durch den Zugriff auf Web Services oder Bluetooth Services) ist möglich. Die Mobilität der Nutzer setzt allerdings voraus, dass verschiedene Online- und Offline-Umgebungen gestaltet und unterschiedliche Techniken kombiniert werden. Beispielsweise kann es passieren, dass durch den Ausfall virtueller Sensoren nicht alle Daten die benötigt werden, zur Verfügung stehen.

Die Aufgabe des mobilen Gerätes beschränkt sich allerdings nicht nur auf das Sammeln der Sensordaten. Auch eine Auswertung dieser Daten kann bereits vorgenommen werden. Nach der Kontextverarbeitung werden die Ergebnisse an eine zentrale Instanz weitergeleitet um die für den Nutzer relevanten Daten abzurufen und zu visualisieren. Es dient damit zudem als Aktuator: es ermöglicht die direkte Beeinflussung der Umwelt und dient vornehmlich als Benutzerschnittstelle.

Abbildung 3.5: Anwendungsszenario: Einsatz des Systems auf einem mobilem Endgerät

3.3.3 Resultierende Anforderungen an den System-Entwurf

Der grundsätzliche Ablauf des Verfahrens bleibt bei den zuvor beschriebenen Szenarien gleich. Der einzige Unterschied besteht in der Verteilung der Systemkomponenten und der Schnittstelle zum Nutzer: während bei einer etablierten Pervasive Computing Infrastruktur die Verarbeitung der Sensordaten komplett auf dem Server durchgeführt werden muss, sollte bei einem mobilen Gerät, in dem mehrere Sensoren integriert sind, eine Vorverarbeitung und Extraktion der Eigenschaften bereits auf diesem durchgeführt werden.

Dies ermöglicht es unter anderem die Kosten für die Ausführung zu minimieren, da keine kontinuierliche Kommunikation mit dem Server aufrecht erhalten werden muss.

Die Dynamik der Infrastruktur und die Mobilität des Nutzers setzen voraus, dass das Spiel so gestaltet wird, dass technische Hilfsmittel und Informationen nicht jederzeit zwingend verfügbar sein müssen. Die Herausforderung des Verlusts der Kommunikationsmöglichkeiten als auch der Sensordaten erfordert deshalb ein robustes Spiele-Design, das mit unterschiedlichen Verbindungszuständen umgehen kann. Zudem muss davon ausgegangen werden, dass die benötigte technische Infrastruktur, die die Umgebung des Nutzers sensorisch widerspiegelt, in der heutigen Zeit in dem erwünschten Umfang nicht vorhanden ist. Stattdessen lässt sich jedoch ein Trend erkennen (vgl. Abschnitt 2.4, S. 40), dass zunehmend mehr Sensoren in Mobilgeräte integriert werden. Dies ermöglicht bei der Implementierung das Auslesen zum Teil sehr unterschiedlicher Informationen in Bezug auf die Situation des Nutzers.

Um eine Akzeptanz des entwickelten System zu erreichen, müssen zudem einige Eigenschaften erfüllt werden, die sich auf die praktische Umsetzung beziehen und mitunter erhebliche Herausforderungen darstellen können: es muss beispielsweise eine Verwendung standardisierter Protokolle und Schnittstellen vorgesehen werden. Zudem sollte die Lösung möglichst modular und flexibel gestaltet sein, sodass sie sich mit wenig Aufwand an neue Einsatzgebiete anpassen lässt. Auch die Skalierbarkeit und Performanz der Anwendung sind von hoher Bedeutung: für den Nutzer sollte es im Prinzpip keinen Unterschied machen, ob er auf Inhalte auf seinem Computer oder auf ein entferntes System zugreift.

Gerade in sicherheitskritischen Bereichen muss zudem eine enorme Zuverlässigkeit des Systems erreicht werden. Nur wenn ein Nutzer dem System vertraut und sich bei dessen Benutzung sicher fühlt, wird er es auch benutzen. Dies beschränkt sich nicht nur auf die Sicherheit im Sinne von Privacy, sondern vor allem auch im Sinne von Security: der Widerstandsfähigkeit des Systems gegen Sabotageakte und bewusste Störaktionen (siehe Abschnitt 2.2.2, S. 27).

Aus den vorangegangen Beschreibungen lässt sich bereits entnehmen, dass es sich bei der Umsetzung der Anwendung prinzipiell um eine *verteilte Architektur* handelt. Die Sensordaten, unabhängig ob diese bereits vorverarbeitet oder im Rohzustand vorliegen, müssen zur weiteren Verarbeitung an ein entferntes System kommuniziert werden.

Die prototypische Umsetzung des Systems kann jedoch in Ermangelung der fehlenden Infrastruktur nicht beide Szenarien abdecken und wird infolgedessen die Umsetzung für ein Smartphone in den Vordergrund der Betrachtungen stellen. Um jedoch prinzipiell auch die Realisierung weiterer Szenarios zu unterstützen, wird eine logische Trennung der Komponenten der Benutzeroberfläche, der Anwendungslogik und der Datenhaltung von

zentraler Bedeutung sein um diese flexibel und unabhängig voneinander anpassen und warten zu können.

3.4 Anwendungsfälle und Rollen

Die Konzeption und der Entwurf der Anwendung soll sich in Anbetracht der praktisch nicht vorhandenen Infrastruktur auf das zuvor vorgestellte Szenario beschränken, bei dem das Mobiltelefon sowohl die Schnittstelle zum Nutzer als auch zu den Sensoren darstellt.

Im Funktionsumfang sollen einige der Konzepte der zuvor analysierten bestehenden Systeme aufgegriffen und um eigene Erweiterungen ergänzt werden. Die Anforderungen unterscheiden sich vor allem in dem Fakt der Kontextsensitivität und der Positionsbestimmung: die Architektur der Applikation soll es ermöglichen situationsabhängig zu reagieren und die Positionierung sowohl Indoor als auch Outdoor vornehmen zu können.

Um einen Einblick in den gewünschten Funktionsumfang geben zu können, soll daher nachfolgend eine Auswahl von Anwendungsfällen dargestellt und erörtert werden. Abbildung 3.6 gibt bereits einen Überblick über einige der Anforderungen.

Prinzipiell lassen sich bei der Interaktion mit dem System drei unterschiedliche Nutzerrollen identifizieren. Bei der Verwaltung des Systems und des Spiels sind hier die *Autoren* und *Administratoren* zu nennen. Die eigentliche Nutzung wird allerdings durch die *Spieler* initiiert.

Aufgabe der Autoren ist die *Verwaltung der Story* als auch die *Verwaltung der Aufgaben* innerhalb des Spiels. Vor allem bei der Story muss ein qualitativ hochwertiges Rahmenwerk geschaffen werden, um das Interesse der Spieler zu wecken. Um den Spielspaß aufrecht zu erhalten, müssen zudem interessante und fordernde Aufgaben zur Verfügung gestellt werden. Die geforderte Qualität dieser Spielelemente kann jedoch nur gewährleistet werden, wenn diese Aktionen durch entsprechend geschulte Anwender vorgenommen werden. Diese zu bestimmen ist Aufgabe des Administrators: nur ihm steht es frei die *Rechte der Nutzer im System zu verwalten*. Nach Fertigstellung der Aufgaben und der Story des Spiels ist es dem Spieler vorbehalten, das *Spiel zu spielen*.

3.4.1 Benutzerverwaltung

Der Administrator möchte im System einen neuen Spieleautor bekannt machen. Um Zugang zum Benutzermanagement zu erhalten, muss er sich zuvor erfolgreich mit einer mit Administrator-Priviligien ausgestatteten Nutzerkennung am System anmelden. Dort steht

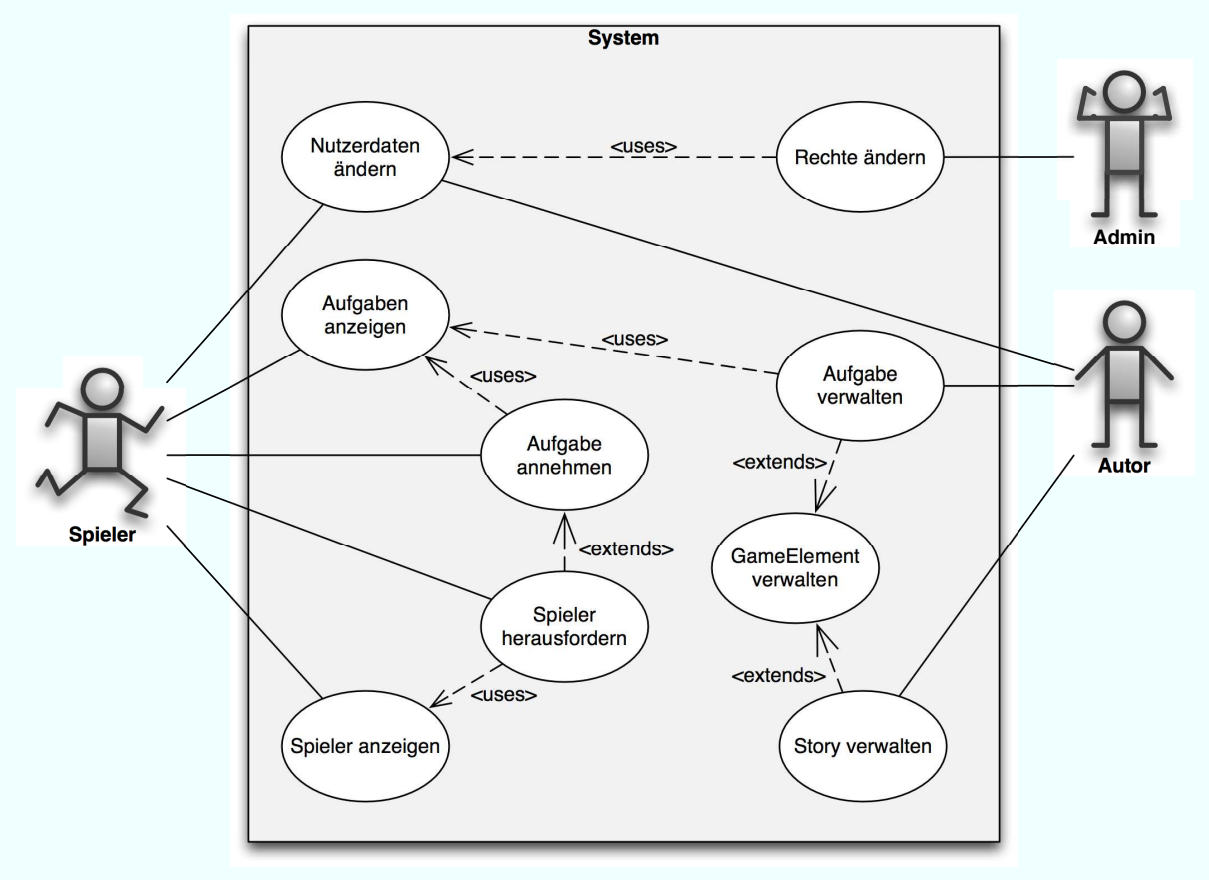


Abbildung 3.6: Anwendungsfall-Diagramm: Ausgewählte Funktionen des Systems

es ihm frei einem bereits im System bekannten Nutzer mit den entsprechenden Rechten auszustatten oder einen neuen Nutzer zu erstellen.
Dieser Prozess soll im Folgenden durch ein Aktivitätsdiagramm genauer skizziert werden.

Wie in Bild 3.4.1 zu erkennen ist, hat der Administrator nach der erfolgreichen Anmeldung am System prinzipiell zwei Optionen vorzugehen:

- **Auswahl eines bestehenden Nutzers**: Bei der Auswahl eines bereits im System bekannten Nutzers, steht es dem Administrator frei, dessen Daten seinen Vorstellungen entsprechend anzupassen oder aber den Nutzer aus dem System dauerhaft zu löschen.

- **Erstellen eines neuen Nutzer**s: Ist der Nutzer, der als Spieleautor bekannt gemacht werden soll, im System noch nicht bekannt, so hat der Administrator die Möglichkeit, diesen anzulegen. Um diese Aktion abzuschließen, müssen zumindest der Name, ein Initialisierungspasswort und die Rechte des Nutzers angegeben werden. Diese können selbstverständlich im Nachhinein durch den Nutzer geändert werden.

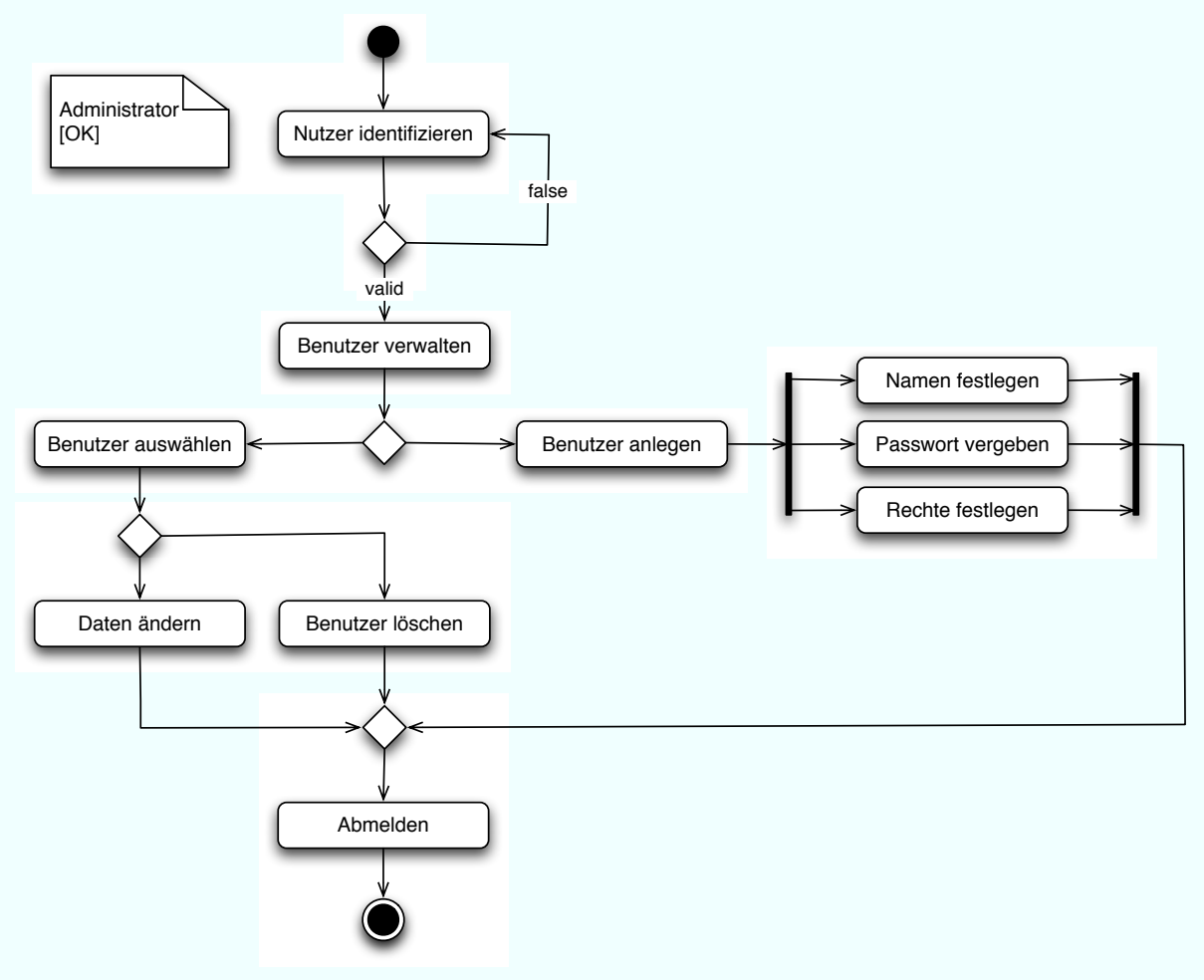


Abbildung 3.7: Aktivitätsdiagramm der Benutzerverwaltung

Nach dem erfolgreichen Abschluss der Anpassungen kann der Administrator entweder weitere Änderungen am System vornehmen oder sich vom System abmelden.

3.4.2 Administration der Storyline

Die Storyline des Spiels soll durch einen Spieleautor den aktuellen Anforderungen entsprechend angepasst oder erweitert werden. Um Zugang zur Verwaltung der Storyline zu bekommen, ist es auch hier nötig sich mit den entsprechenden Rechten am System anzumelden. Der konkrete Ablauf des Vorgangs bis zur fertig editierten Storyline ist in Abbildung 3.4.2 aufgeführt.

Nach erfolgreicher Anmeldung am System, kann der Spieleautor Details in der Storyline-Verwaltung einsehen. Prinzipiell hat er auch hier, analog zu den Funktionalitäten der Benutzerverwaltung, die Möglichkeit, ein neues Event der Storyline hinzuzufügen, oder ein bereits bestehendes zu editieren. Beim Anlegen eines neuen Events sind lediglich die

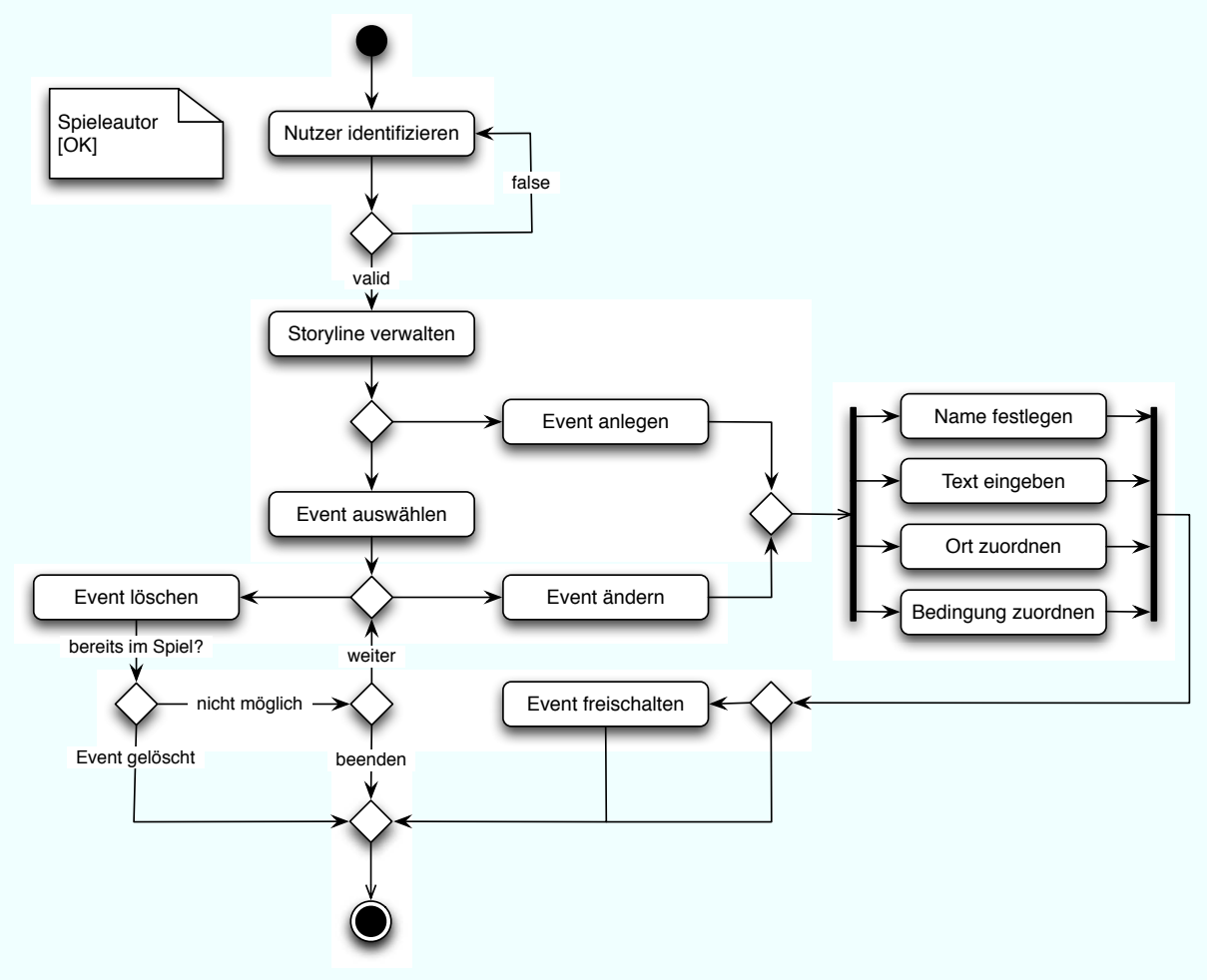


Abbildung 3.8: Aktivitätsdiagramm der Storyline-Verwaltung

dafür benötigten Daten einzugeben und am System bekannt zu machen. Ist dies geschehen, so steht es dem Autor frei, ob das neue Event bereits im aktuellen Spieleverlauf aktiviert wird oder nicht. Das kann auch im Nachhinein vorgenommen werden.
Soll jedoch ein bereits existierender Event bearbeitet werden, so muss dieser aus der Liste der bereits bestehenden ausgewählt werden. Das Vorhaben einen Event zu löschen, setzt voraus, dass kein Spieler dieses Event je gestartet hat. Diese Einschränkung muss getroffen werden, um Inkonsistenzen in der Datenbank zu vermeiden. In diesem Falle könnte der Autor dieses Event nur den neuen Anforderungen entsprechend editieren.
Das Ändern eines Events unterliegt dem selben Ablauf, wie die Erstellung desselbigen. Diesem Vorgang sind bisher noch keine Restriktionen zugewiesen, der Spieleautor kann demnach alle relevanten Daten auch im Nachhinein ändern.

3.4.3 Erstellen einer neuen Aufgabe

Eine Aufgabe ist eines der zentralen Elemente im System, da mit dessen Hilfe, die Handlungen des Spiels gesteuert und entworfen werden können. Die Erstellung einer Aufgabe ist allerdings nur seitens der Verwaltungs-Anwendung auf Spieleautoren oder Administratoren beschränkt – die Spieler können durch das Erzielen neuer persönlicher Bestzeiten implizit Aufgaben für andere Spieler erstellen. Die Handlung des Spiels unterliegt so einer ständigen dynamischen Entwicklung. Lediglich der Rahmen wird durch die Storyline und somit den Autoren vorgegeben.

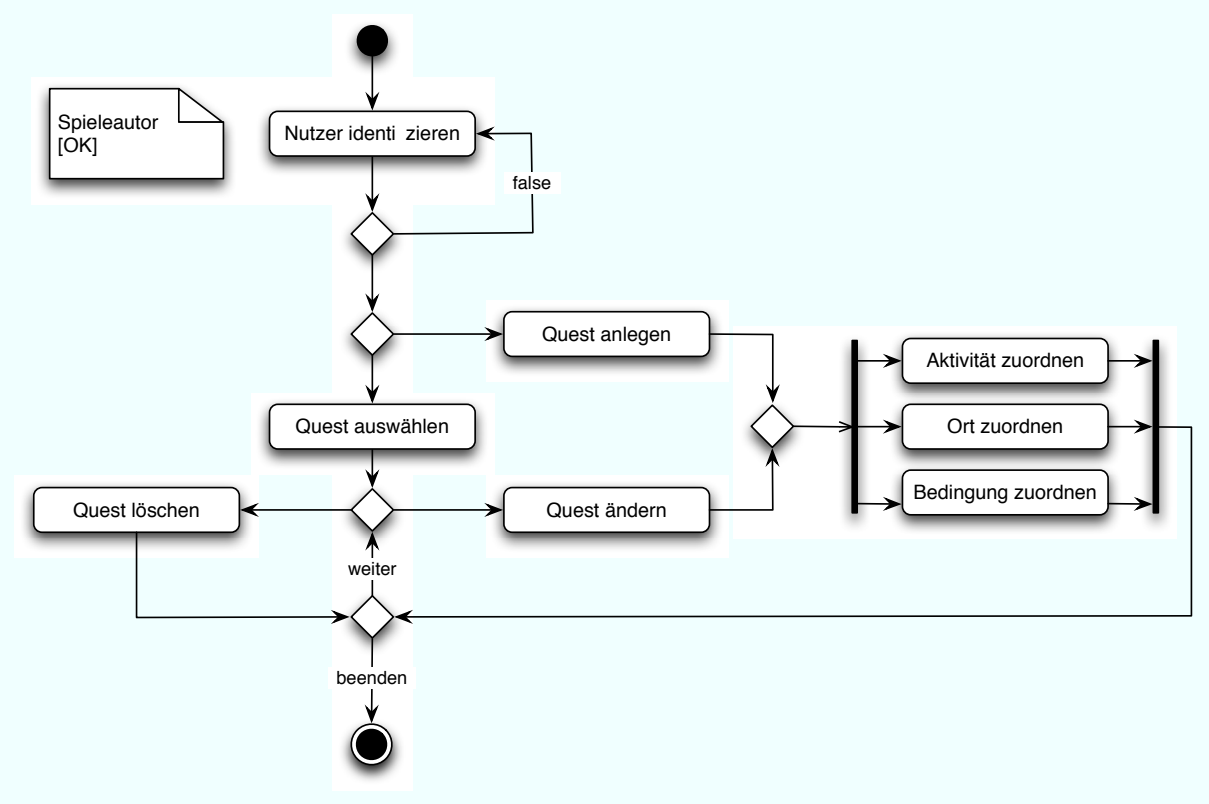


Abbildung 3.9: Aktivitätsdiagramm zur Erstellung eines neuen Quests

In Abbildung 3.4.3 ist der grobe Ablauf der Erstellung einer neuen Aufgabe dargestellt. Nachdem die entsprechenden Rechte zur Erstellung durch das System überprüft wurden, können die entsprechenden Eingaben durch Autoren vorgenommen werden. Eine Aufgabe kann durch unterschiedliche Parameter definiert werden: eine spezifische Dauer oder Strecke, eine Geschwindigkeit oder Sportart oder einem prozentualen Wert in Abhängigkeit der Leistungen des Spielers. Diese Parameter können in beliebiger Zusammenstellung kombiniert werden um interessante und fordernde Aufgaben zu erstellen. Optional können diese an einen bestimmten geographischen Ort im Raum Berlin gebunden werden.

Auch Spielern wird die Möglichkeit geboten, Aufgaben für andere zu hinterlassen: beim

Erreichen einer neuen Bestleistung, egal ob es sich dabei um eine neue Maximalgeschwindigkeit, Dauer oder Strecke handelt, kann automatisch eine neue Aufgabe an der aktuellen Position des Spielers hinterlassen werden. Andere Spieler, die diese Position finden, werden informiert über die Leistung des Erstellers und bekommen die Möglichkeit, diesen herauszufordern. Wird der Rekord gebrochen, so wird der Spieler entsprechend honoriert und eine neue Aufgabe wird hinterlassen.

3.4.4 Spieldurchführung

Um an der Handlung des Spiels teilzunehmen, muss sich der Spieler zuerst am System registrieren, damit seine Aktionen im Spiel eindeutig zugeordnet werden können. Diese Daten muss er dann einmalig in den Einstellungen seines iPhones hinterlegen.

Zu Beginn des Spiels wird dem Spieler kurz dessen Geschichte erläutert und was das Ziel des Spiels ist. Er erhält daraufhin seine erste Aufgabe. Diese könnte beispielsweise lauten: „Laufe $100m$ in maximal 60 Sekunden“. Akzeptiert der Spieler diese Aufgabe, so startet eine Stoppuhr und dessen Bewegung wird anhand der ermittelten GPS-Daten gemessen und ausgewertet. Für die Erfüllung dieser Aufgabe, wird er mit so genannten Aktivitätspunkten belohnt. Diese Punkte erhält er auch, wenn er Sport treibt, ohne eine spezielle Aufgabe bekommen zu haben. Nach der Erfüllung dieser Aufgabe, hat er seine ersten Bestwerte aufgestellt. Hier werden unterschiedliche Werte gespeichert, unter anderem:

- Längste Strecke, die zurückgelegt wurde
- Schnellste $5km$
- Schnellste $10km$
- Durchschnittsgeschwindigkeit
- Maximalgeschwindigkeit

Jedes Mal, wenn eine dieser Bestzeiten/-werte verbessert werden, wird an der aktuellen Position des Spielers eine Aufgabe für einen anderen Spieler hinterlassen. Bei diesen Aufgaben, die sowohl von Spielern als auch von den Autoren des Systems erstellt werden, findet zusätzlich eine Unterscheidung in unterschiedliche Sportarten statt.

Der Spieler bekommt während des Spiels Aufgaben angezeigt, die sich in seiner Nähe befinden. Er kann sich diese anschauen um die Herausforderung anzunehmen.

Neben diesen Aufgaben, die der Spieler selbständig finden muss, gibt es Aufgaben, die dem Spieler nur angezeigt werden, wenn er bestimmte Bedingungen erfüllt hat. Dabei

könnte es sich beispielsweise um eine Aufgabe handeln, die mit der Bedingung verknüpft ist, dass der Spieler seit 30 Minuten durchgehend Rad fährt. Nur wenn diese Bedingung erfüllt ist, kann diese Aufgabe von ihm angenommen werden.

Eine weitere Möglichkeit ist die, mit einem anderen Spieler in eine Art Wettstreit zu treten. Der Spieler kann beispielsweise einen in seiner Nähe befindlichen Spieler manuell herausfordern. Eine weitere Möglichkeit besteht in der automatischen Generierung von Wettkämpfen: der Server analysiert regelmäßig die Spieler, die momentan aktiv sind und sucht diejenigen, die sich ungefähr auf dem gleichen Leistungsniveau bewegen und übermittelt beiden eine Aufgabe, die sie zu bewältigen haben. Nach Beendigung werden beide miteinander verglichen und der Gewinner wird entsprechend honoriert.

3.5 Alternative Ansätze für den Systementwurf

Anhand der vorstehenden Analysen und Feststellungen lässt sich der Diskussionsbereich des Systementwurfs dieser Arbeit bereits im Vorfeld etwas eingrenzen. Zentraler Bestandteil der Betrachtung wird die Umsetzung des zweiten Szenarios unter Nutzung des iPhones als zentrale Schnittstelle sein (vgl. Abschnitt 3.3.2, S. 53). Alternativ könnte beispielsweise auch das Nokia N95 genutzt werden, da auch dieses die zur Umsetzung des Prototypen benötigten Sensoren mitbringt. Ein großer Nachteil besteht allerdings in der Usability: das N95 hat keinen Touchscreen und auch das Display ist kleiner als das des iPhones. Da es sich bei dem Spiel jedoch um ein Sport-Adventure handeln soll und die Usability demzufolge von großer Relevanz ist, muss von der Nutzung des N95 abgesehen werden.

Neben einer Applikation die die Schnittstelle zum Spieler darstellt – im Folgenden als *Spiele-Anwendung* bezeichnet – wird eine *Verwaltungs-Anwendung* für den Aufbau und die Pflege der durch das System angebotenen Spiele entworfen und prototypisch umgesetzt. Diese Anforderungen und das herausragende Merkmal der Mobilität in pervasiven Systemen (vgl. Abschnitt 3.1.5, S. 47) erfordert die Umsetzung des Systems in Form einer *Client-Server Architektur*. Die Spiele-Anwendung wird für diesen Zweck für die Ausführung auf einem mobilen Gerät konzipiert wohingegen die Verwaltungs-Anwendung die zentrale Instanz der Architektur darstellt und entsprechende Funktionalität für den Client zur Verfügung stellt.

Es ist die Aufgabe des Clients die benötigten Daten vom Server zu erfragen und die aktuellen Aktivitäten des Spielers zu kommunizieren. Die Kommunikation soll demnach auf ein Pull-Verfahren beschränkt werden: der Client stößt serverseitige Funktionen an und erhält in Folge dessen vom Server die benötigten Informationen. Diese unidirektionale Kommunikation trägt zur Flexibilität des Gesamtsystems bei: der Client kennt das In-

terface des Servers wohingegen dem Server das des Clients nicht bekannt sein muss. Dies bietet die Möglichkeit bei zukünftigen Erweiterungen ohne Änderung des bestehenden Systems zusätzliche Client-Komponenten für weitere mobile Geräte hinzuzufügen.

Um anfallende Kosten bei den Spielern zu minimieren, soll die Kommunikation zwischen Spiele- und Verwaltungs-Anwendung nicht dauerhaft aufrecht erhalten werden. Die Datenintegrität auf dem Server muss trotzdem gewährleisten werden können, deshalb müssen im Entwurf zusätzlich Möglichkeiten der Replikation der Daten betrachtet werden.

Die Forderungen des Pervasive Computing nach Interoperabilität und Konnektivität stellen weitere Randbedingungen dar, die es zu berücksichtigen gilt: die Umsetzung soll mit offenen und allgemeingültigen Standards (vgl. Abschnitt 2.2.1.3, S. 26), vor allem im Bereich der Kommunikation, erfolgen. Auf diese Weise soll eine offene und flexible Architektur umgesetzt werden, die unabhängig von verwendeten Plattformen, Programmiersprachen und Protokollen ist. Die Nutzung proprietärer Protokolle oder Übertragungsformen ist daher nicht von Interesse.

Bei der Konzeption kontextsensitiver Dienste hat sich zur Umsetzung solcher Architekturen bereits der Ansatz einer serviceorientierten Architektur durchgesetzt, weshalb auch bei der Umsetzung des Prototypen die Kommunikation über *Web Services* erfolgen soll. Dies erfolgt durch den Austausch XML-basierter Nachrichten über internetbasierte Protokolle, wie HTTP.

Unter Berücksichtigung der Diskussion in Abschnitt 3.3.3 wird für den Entwurf der Komponenten – sowohl der Client- als auch der Server-Komponente – zudem eine *dreischichtige Architektur* vorgesehen, welche *Datenhaltung*, *Anwendungslogik* und *Präsentation* logisch voneinander trennt. Diese auch als *3-Tier Architektur* bezeichnete Aufteilung ermöglicht die Realisierung verteilter und skalierbarer Anwendungen. Sie gestattet zudem eine flexible Anpassung einzelner Systemkomponenten an die Anforderungen unterschiedlicher Anwendungsszenarien, ohne Teile des übrigen Systems modifizieren zu müssen.

Die Funktionalitäten des Systems werden in Module gekapselt, welche jeweils einen abgegrenzten Aufgabenbereich abdecken. Dies ermöglicht deren Wiederverwendbarkeit und verbessert zugleich die Wartbarkeit des Gesamtsystems.

Kapitel 4

Systementwurf

Das zu entwickelnde System soll es ermöglichen in Abhängigkeit des aktuellen Ortes und der aktuellen Situation und Aktivität des Nutzers relevante Daten und Informationen von einem zentralen Dienst abzurufen. Dieses Prinzip soll im Rahmen dieser Arbeit in einer prototypischen Umsetzung eines pervasiven Adventurespiels aufgezeigt werden.

Einige Technologien und Verfahren zur Umsetzung des Systems wurden bereits in den vorangegangenen Kapiteln identifiziert. Es wurde identifiziert, dass die folgenden Schritte bei dem Entwurf der Kontexterkennung umgesetzt werden müssen:

- das Sammeln der benötigten Sensordaten,
- die Featureextraktion und
- anschließend die Klassifizierung der Aktivität

Die Erwartung ist, dass mit Hilfe von GPS- und Beschleunigungssensor-Daten ein Profil der aktuellen Betätigung des Spielers erstellt werden kann. Dabei soll vor allem die Nutzung der GPS-Daten im Zentrum der Betrachtungen stehen, da hauptsächlich lineare Bewegungen erkannt werden sollen. Die Beschleunigungssensoren sollen diese Daten ergänzen und so eine möglichst genaue Klassifizierung der Aktivität ermöglichen.

Der Aspekt der situationsabhängigen Adaption der Anwendung durch den aktuellen Kontext stellt allerdings nur einen Teil des zu entwickelnden Systems dar. Prinzipiell handelt es sich um eine *verteilte Architektur*, die als *Client-Server-System* realisiert wird. Der Client – die Spiele-Anwendung – stellt die Schnittstelle zum Nutzer dar: er wertet sowohl dessen Eingaben als auch dessen Kontext aus und visualisiert dementsprechend relevante Daten, die er durch die Kommunikation über *Web Services* von einem zentralen Server abruft.

Ausgehend von den Anforderungen an ein solches System, werden in diesem Kapitel das Konzept und die Funktionalität des prototypisch umzusetzenden Systems und seine Architektur definiert. Dabei werden sowohl die einzelnen Komponenten der Client/Server-Kommunikation, als auch der konkrete Aufbau des Clients und Servers mit der Datenhaltung, der Geschäftslogik und der Präsentation betrachtet.

4.1 Architektur des Systems

Die Anwendung soll als verteilte Architektur – genauer als Client-Server-Architektur – realisiert werden. Sie besteht aus der Spiele-Anwendung und der Verwaltungs-Anwendung, welche zusätzlich zur Kommunikation mit dem Client entsprechende Schnittstellen anbietet. Dabei wird der Ansatz der serviceorientierten Architektur verfolgt: es erfolgt eine lose Kopplung, indem der Server mit plattformunabhängigen Schnittstellen, in Form von Web Services, die Inhalte und Aufgaben des Spiels zur Verfügung stellt.

Dieser prinzipielle Aufbau wird in Abbildung 4.1 anhand eines Verteilungsdiagramms illustriert: es demonstriert die Verteilung der einzelnen Komponenten des Systems auf die Spiele- und Verwaltungs-Anwendung.

Neben der Realisierung des Systems als Client-Server-Architektur wird für beide Anwendungen, sowohl die Spiele- als auch die Verwaltungs-Anwendung, eine mehrschichtige Architektur im Sinne des *Model-View-Controller Architekturmusters* entwickelt.

Bei dem Model-View-Controller Architekturmuster bilden sich drei unabhängige Schichten heraus: die Präsentations-, die Anwendungs- und die Datenhaltungs-Schicht. Jede Schicht kommuniziert nur mit ihren unmittelbar benachbarten Schichten über fest definierte Schnittstellen.

Wenn bei dem traditionellen Modell des Model-View-Controller Musters (vgl. Abbildung 4.2(a)) durch den Nutzer Änderungen an den Präsentations-Komponenten (*View*) vorgenommen werden, wird als Ergebnis ein Event generiert. Dieses wird an die steuernde Komponente – den *Controller* – weitergeleitet. Dieser koordiniert und verarbeitet das Event in einer anwendungsspezifischen Weise: dies kann beispielsweise in der Änderung des zugrunde liegenden Datenmodells (*Model*) oder der View resultieren. Infolgedessen benachrichtigt das Datenmodell alle Objekte, die an diesem als Observer registriert sind, um die Änderung seines Status bekannt zu geben. Handelt es sich bei einem solcher Objekte beispielsweise um eine View, so kann sich diese entsprechend anpassen, indem sie die aktuellen Daten des Datenmodells abfragt, um ihre Darstellung zu aktualisieren.

Bei der Umsetzung soll allerdings eine etwas modifizierte Version des MVC-Musters ge-

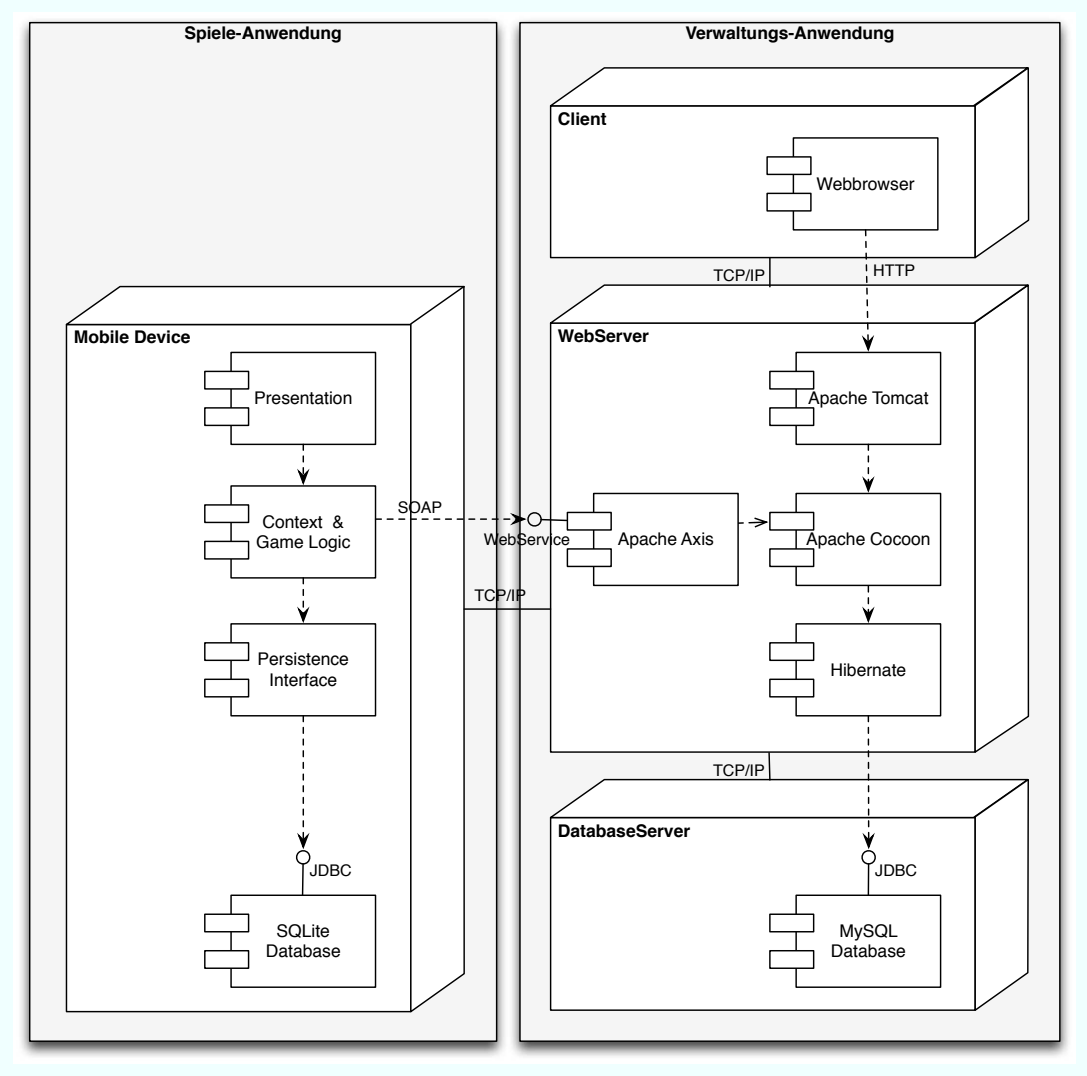
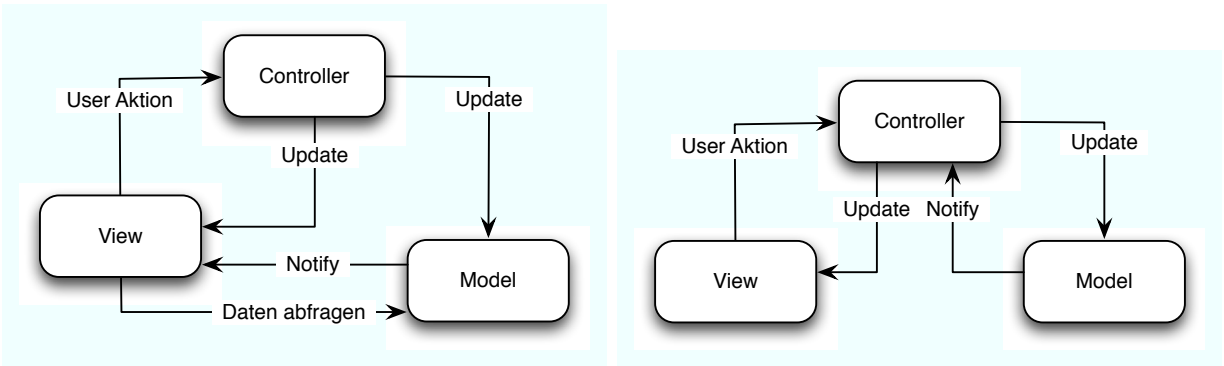


Abbildung 4.1: Verteilungsdiagramm: Übersicht der Architektur des Prototypen



(a) Traditionelle Version (b) Modifizierte Cocoa-Version

Abbildung 4.2: Unterschiedliche Versionen des MVC Architekturmusters

nutzt werden. Diese wird beispielsweise bei Implementierungen von Apple präferiert (vgl. [11]). Sie weisen darauf hin, dass das traditionelle Design ein theoretisches Problem aufwirft: sowohl die Objekte der View als auch die des Modells sollten einen sehr hohen Grad an Wiederverwendbarkeit aufweisen um die Konsistenz des Erscheinungsbilds und des Verhaltens ähnlicher Applikationen und Anwendungsszenarien zu garantieren. Das ist in der traditionellen Variante allerdings nicht der Fall.

Objekte der Datenhaltung kapseln per Definition bereits die mit der Problemdomäne assoziierten Daten und führen auf diesen Funktionen aus. Deshalb ist es am besten, die Objekte des Modells und der View voneinander zu separieren, um auf diese Weise die Wiederverwendbarkeit zu steigern. Infolgedessen werden bei den meisten Cocoa Applikationen Benachrichtigungen über Statusänderungen des Datenmodells durch den Controller an die View-Objekte kommuniziert. Abbildung 4.2(b) skizziert diese modifizierte Konfiguration.

Die Vorzüge der Umsetzung dieser Architektur sind vor allem bei der Reduzierung der Komplexität und Abhängigkeit innerhalb des Systems zu finden: ein einfacher Austausch sowie die Erweiterung von Komponenten wird beispielsweise durch die hohe Kapselung der Schichten ermöglicht. Ein Austausch der Datenhaltungs-Schicht wäre so ohne Beeinträchtigung des Clients möglich, da die Schnittstellen für den Zugriff zentral in der Anwendungsschicht definiert und verwaltet werden. Auch Änderungen in der Logik können vorgenommen werden, ohne dass dies Konsequenzen für den Zugriff beim Client nach sich ziehen würde.

Durch den erhöhten Kommunikationsaufwand zwischen den Schichten können allerdings Performance-Verluste identifiziert werden, was als nachteilig erachtet wird. Die Vorteile die diese Architektur mit sich bringt, überwiegen jedoch, weshalb trotzdem, sowohl beim Client als auch beim Server, dieses Architekturmuster eingesetzt wird.

In den folgenden Abschnitten wird deshalb die Funktionalität des Systems gemäß ihrer Zugehörigkeit zu den Schichten Präsentation, Anwendungslogik und Datenhaltung besprochen.

Zur Veranschaulichung der Zusammenhänge der eingeführten Klassen und Schnittstellen wird hierbei wiederum aktiver Gebrauch von den Diagrammen der *Unified Modelling Language* (UML) gemacht.

4.2 Datenhaltungs-Schicht

Die Komponenten der *Datenhaltungs-Schicht* realisieren die persistente Speicherung und Verwaltung der zur Realisierung des Systems benötigten Daten, indem sie diese mit Hilfe eines objektrelationalen Mappers auf eine relationale Datenbank abbilden und der Anwendungslogik Schnittstellen für den Zugriff bereitstellen. Abbildung 4.3 skizziert dies anhand der serverseitigen Umsetzung.

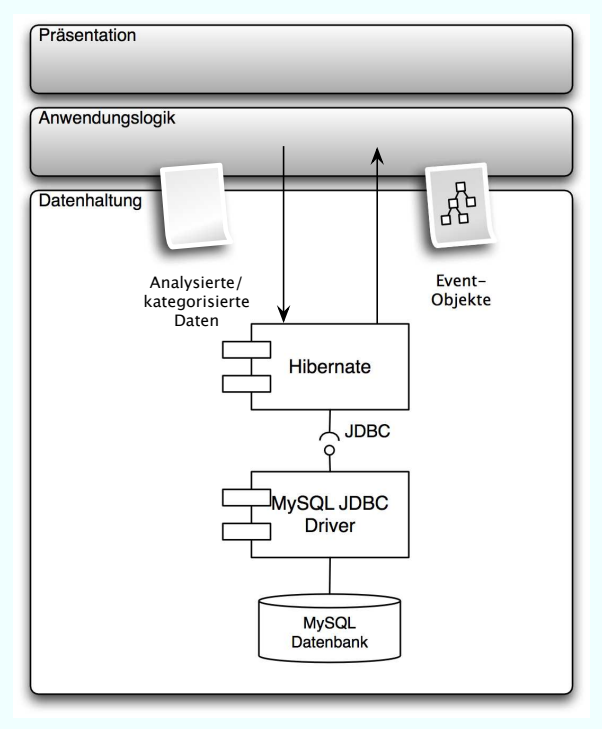


Abbildung 4.3: Übersicht der Integration der Datenhaltungs-Schicht ins System

Zur persistenten Speicherung großer Datenmengen haben sich Datenbankmanagementsysteme etabliert. Die meisten von ihnen basieren auf dem relationalen Schema von Edgar F. Codd und werden folglich als *relationales Datenbankmanagementsysteme* (RDBMS) bezeichnet.

Prinzipiell werden die Daten in Form von Relationen – die mathematische Beschreibung einer Tabelle – festgehalten. Sie definieren die Menge und Art von Attributen, welche einer Tabellenspalte entspricht. Einzelne Datensätze werden als Tupel zeilenweise abgelegt, wodurch jedem Attribut ein atomarer Wert zugeordnet wird. Objektorientierte Programmiersprachen dahingegen kapseln Daten und Verhalten in Objekten. Diese beiden Paradigmen sind grundlegend verschieden. Um den Widerspruch aufzulösen kann ein *objektrelationaler Mapper* (kurz: O/R-Mapper) genutzt werden: er ermöglicht die Darstellung der Objekte und deren Beziehungen in einer relationalen Datenbank. Auch die Erzeugung

von Objekten aus den entsprechenden Datensätzen wird auf diese Weise ermöglicht. Im einfachsten Fall werden Klassen auf Tabellen abgebildet: jedes Objekt entspricht dann einer Tabellenzeile und jedes Attribut einer Tabellenspalte.

Diese Funktionalität befreit die Umsetzung der Datenhaltung zusätzlich vom Schreiben spezifischer Datenbank-Zugriffe und hält die Applikation somit unabhängig vom individuellen SQL-Dialekt der verwendeten relationalen Datenbank. Dies ermöglicht, sehr flexibel auf die jeweiligen Anforderungen des umzusetzenden Systems einzugehen.

Solche OR-Mapper werden für viele gängige Programmiersprachen angeboten. Auch für Java gibt es viele Optionen, die man nutzen kann, um die Datenhaltung auf diese Weise zu kapseln. Beispielsweise kann die Java Persistence API (JPA)[1] genutzt werden. Diese wurde als Teil der Spezifikation JSR 220[2] der Enterprise Java Beans 3.0 definiert. Daneben existiert als sehr bekanntes Beispiel auch Hibernate[3] oder auch Apache Cayenne[4].

Bei dem Entwurf und der Umsetzung des Prototypen wird davon ausgegangen, dass Hibernate als O/R-Mapper genutzt wird. Dieser könnte aber bei einer Adaptierung der Entwürfe durch einen beliebigen anderen ersetzt werden. Auf die Einflüsse eines solchen O/R-Mappers auf das Design des Datenmodells wird im Folgenden etwas genauer eingegangen.

4.2.1 Datenmodell des Systems

Abbildung 4.4 zeigt auszugsweise das Klassendiagramm der Entität `GameEvent`.Bei diesem Ausschnitt handelt es sich um einen zentralen Teil des konzeptuellen Datenmodells und gibt einen Überblick über dessen Attribute und Beziehungen.

Das zentrale Element des Spiels wird hierbei durch die Entität `GameEvent` repräsentiert. Es wird genutzt, um nahezu alle Elemente des Spiels zu repräsentieren. Sie enthält eine numerische Event-ID die als Primärschlüssel fungiert. Da ein solches Event im Laufe des Spiels meist mit einem konkreten Ort in Verbindung gebracht wird, ist zwischen dem `GameEvent` und der `Location` eine 1:1 Beziehung. Die Entität eines Ortes enthält Attribute wie den Längengrad, Breitengrad, Höhe sowie die Information ob diese Position Indoor oder Outdoor ermittelt wurde und kennzeichnet somit eine eindeutige geografische Position. Bei dem Objekt der `Location` handelt es sich um eine schwache Entität. Dessen

[1] Auf `http://java.sun.com/javaee/technologies/persistence.jsp` können genauere Informationen eingesehen werden.

[2] `http://jcp.org/en/jsr/detail?id=220`

[3] Die Dokumentation und weitere Informationen zu Hibernate können der Seite `http://www.hibernate.org` entnommen werden.

[4] Weitere Informationen sind unter `http://cayenne.apache.org` zu finden.

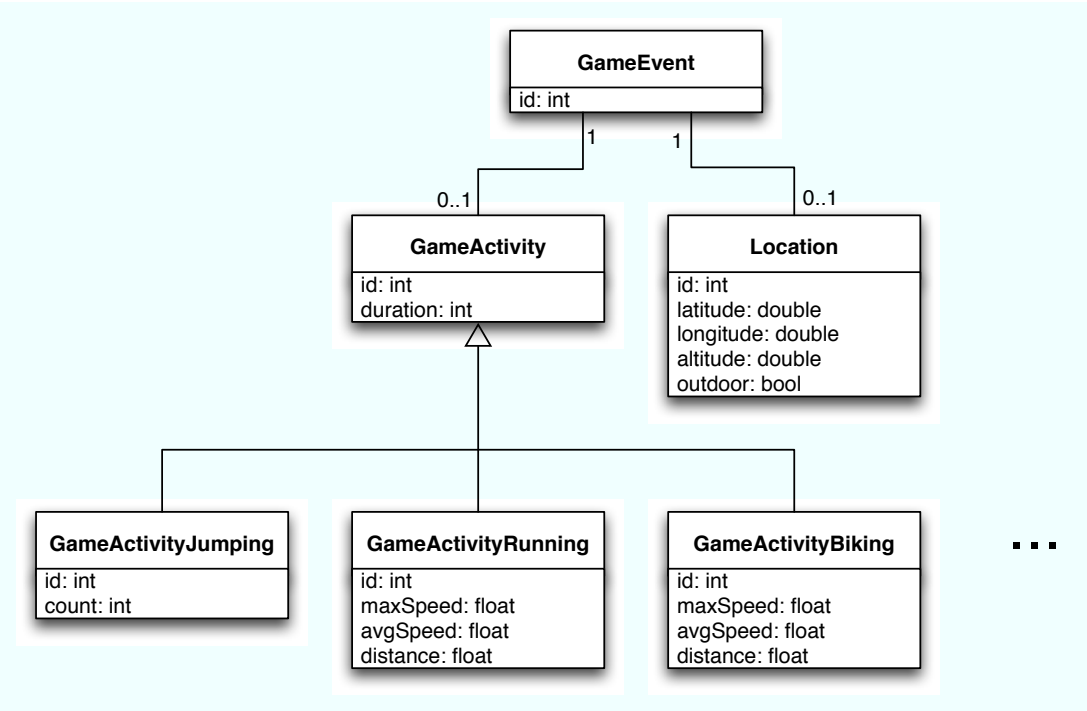


Abbildung 4.4: Klassendiagramm der Entität `GameEvent`

Existenz setzt das Vorhandensein des `GameEvents` voraus.

Neben dem Ort enthält das Event zusätzlich eine Referenz auf eine `GameActivity`. Diese Klasse dient als Superklasse für alle Aktivitäten, die durch das Spiel automatisch erkannt werden sollen. Exemplarisch werden in der Abbildung die Aktivitäten `GameActivityJumping`, `GameActivityRunning` und `GameActivityBiking` illustriert. Diese enthalten, in Abhängigkeit der Aktivität, unterschiedliche Merkmale. Eine Aktivität, die das Laufen repräsentiert, enthält deshalb Attribute, die die Durchschnittsgeschwindigkeit, die maximale Geschwindigkeit und die zurückgelegte Distanz beschreiben. Auch hierbei handelt es sich um eine schwache Entität: wird das `GameEvent` gelöscht, so werden auch die referenzierten Daten der `Location` und `GameActivity` gelöscht.

Für das Abbilden einer Vererbungsstruktur stehen im Hibernate-Mapping drei Strategien zur Verfügung:

1. **Eine Tabelle für die Klassenhierarchie**: Eine Tabelle in der Datenbank enthält alle Attribute der Unterklassen. Für die Unterscheidung der spezifischen Klasse wird eine zusätzliche Spalte – die Diskriminatorspalte – eingeführt.

2. **Eine Tabelle pro Klasse**: Für jede Klasse aus der Hierarchie wird eine eigene Tabelle erstellt. Die Beziehung zur Superklasse wird mit einer 1:1-Beziehung über den Primärschlüssel realisiert.

3. **Eine Tabelle pro konkreter Klasse**: Bei diesem Verfahren wird eine Tabelle für jede Klasse am Ende der Vererbungshierarchie erstellt. Die Tabelle enthält aus diesem Grund alle Attribute aus den Superklassen.

Diese Strategien weisen sowohl Vor- als auch Nachteile auf: das erste der zuvor genannten Verfahren setzt voraus, dass die Felder der Kind-Klassen NULL werden dürfen, auch wenn dies beim Entwurf der Klassenhierarchie nicht vorgesehen oder explizit ausgeschlossen wurde. Auf diese Weise wird für das Speichern von Daten auch mehr Platz benötigt. Der Vorteil dieser Strategie besteht allerdings in der Performance: diese wird von keinem der beiden anderen Verfahren übertroffen. Sowohl das Speichern als auch das Lesen kann sehr viel schneller ermöglicht werden. Das ist vor allem von Vorteil, wenn die Objekte fortlaufend benötigt werden.

Der objektorientierte Entwurf wird am natürlichsten durch das zweite der Verfahren umgesetzt, da Hibernate auf diese Weise Polymorphie sehr gut unterstützt. Dies schlägt sich allerdings auf die Performance nieder, da bei jeder Abfrage mehrere Joins notwendig sind. Bei dem letzten der vorgestellten Verfahren wird dahingegen Speicherplatz eingespart doch die Zugriffszeit erhöht sich dementsprechend, da bei polymorphen Abfragen mehrere Beziehungen aufgelöst werden müssen. Deshalb sollte dieses Verfahren vorwiegend dann genutzt werden, wenn eine Klasse viele Daten enthält.

Um das Klassendiagramm auch in der Datenbank möglichst objektorientiert zu repräsentieren und zudem den benötigten Speicherplatz zu minimieren, wird für die Umsetzung in diesem Fall die Nutzung der zweiten Strategie vorgezogen. Es soll eine Tabelle pro Klasse erstellt werden.

Das Entity Relationship Diagramm des Ausschnitts des konzeptuellen Datenmodells in Abbildung 4.5 illustriert deshalb die resultierenden Attribute und Beziehungen der Entitäten bei Nutzung dieser Strategie:

Die für die Gestaltung des Benutzerbereichs benötigten Daten werden in den Entitäten der Abbildung 4.6 illustriert. Alle im System vorhandenen Rollen spezialisieren die Klasse `User`. Sie erben infolgedessen die spezifischen Attribute der Klasse: diese enthält Attribute wie eine numerische id als Primärschlüssel, Name, E-Mail und das als MD5-Hash gespeicherte Passwort. Dieses Vorgehen ermöglicht, dass die Nutzer des Systems durch eine spezifische Klasse identifiziert werden. Eine Erweiterung der Implementierung kann somit durch eine Anpassung der Sub-Klasse leicht und schnell vorgenommen werden und hat lediglich das Hinzufügen einer Tabelle in der Datenbank zur Folge.

Eine besondere Rolle wird hierbei dem `Player` zuteil. Während des gesamten Spiels soll dieser durch die Entität `GameCharacter` repräsentiert werden, damit durch diese Anony-

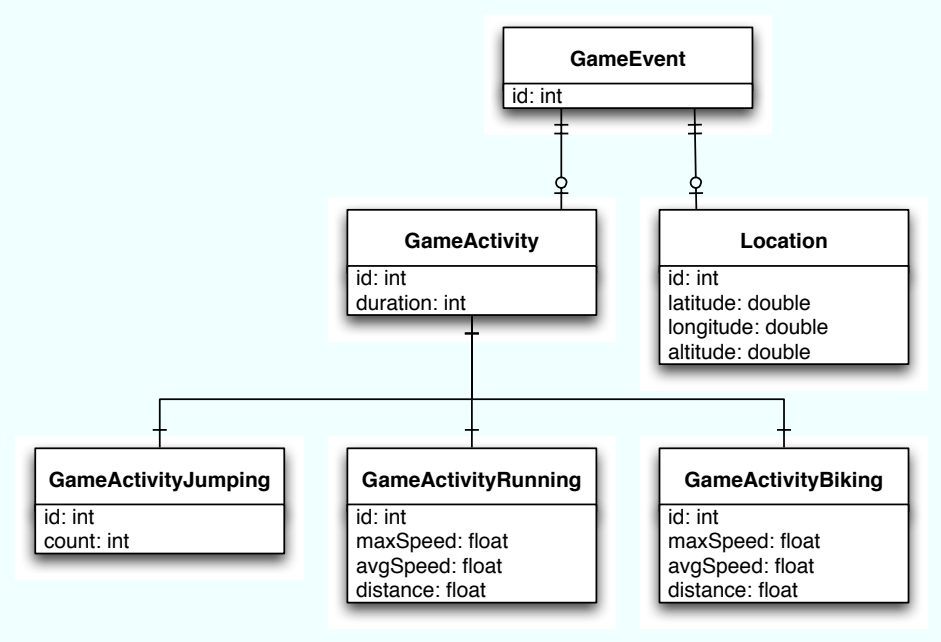


Abbildung 4.5: Entity Relationship Diagramm des `GameEvents`

misierung möglichst nicht auf die reale Person geschlossen werden kann. Deshalb wird dieser durch das eindeutige Attribut `deviceId` identifiziert. Die Bewertung des Spielers kann durch die Geschäftslogik anhand des Attributs `activityPoints` und dem Fortschritt in der Storyline verwirklicht werden.

Die Umsetzung des Klassendiagramms zu einem Entity Relationship Diagramm wird in Abbildung 4.7 visualisiert. Auch hier wird wiederum der objektorientierte Ansatz verfolgt.

Bei den zuvor vorgestellten Ausschnitten der Datenbank fällt auf, dass ich auf bidirektionale Beziehungen zum größten Teil verzichtet habe. Dies ist allerdings beabsichtigt, da Hibernate zum Verändern, also zum Einfügen und Löschen alle zu einer Liste gehörenden Objekte laden würde. Damit ist dieses Vorgehen für große Datenmengen ungeeignet, weil zum einen ein Performance-, und zum anderen ein Speicherproblem droht. Zudem kann davon ausgegangen werden, dass sich bei einem solchen Spiel sehr schnell viele Daten in der Datenbank ansammeln werden.

4.2.2 Datenintegrität

Die Aktionen zur Manipulation der Daten werden bei Nutzung der Verwaltungs-Anwendung webbasiert über Formulare durch den Autoren oder Administratoren initiiert. Um die Integrität der Daten zu gewährleisten können Plausibilitätsprüfungen explizit durch das Setzen von Validatoren für entsprechende Eingabefelder und implizit durch die Deklaration spezifischer Annotationen in den Entitäten der Datenhaltung durchgeführt

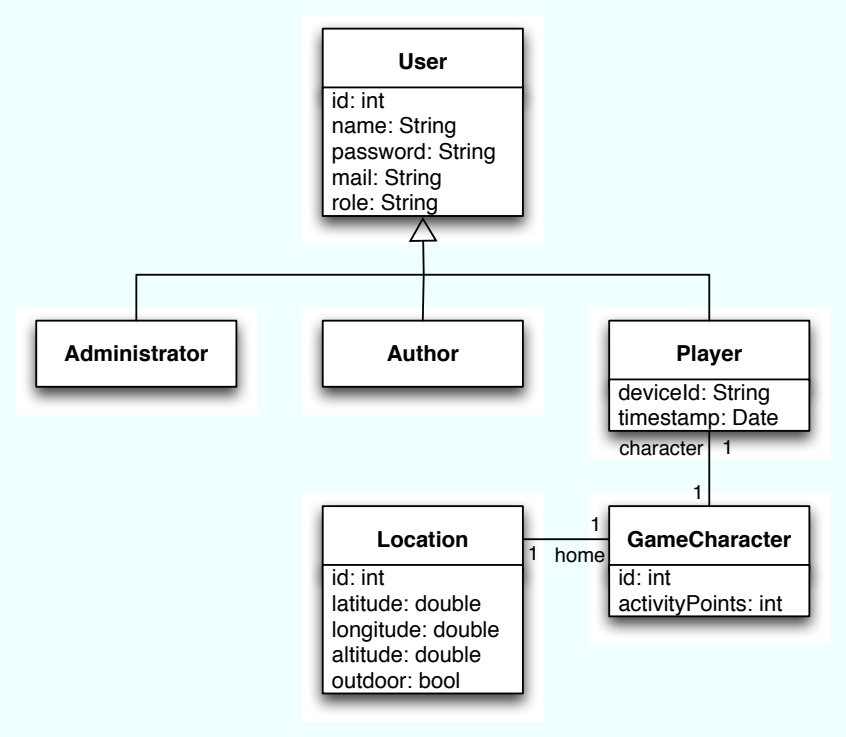


Abbildung 4.6: Hierarchie und Abhängigkeiten der Entität `User` im System

werden. Auf diese Weise kann sicher gestellt werden, dass nur valide Daten in der Datenbank gespeichert werden.

Bei der Spiele-Anwendung gestaltet sich die Integritätsprüfung etwas einfacher: da der Nutzer kaum Möglichkeiten hat, in Eingabefeldern Werte einzugeben, braucht keine explizite Überprüfung der Eingabedaten durchgeführt zu werden. Es muss somit lediglich sichergestellt werden, dass die durch die Sensoren ermittelten Werte nicht fehlerhaft sind. Dies kann durch entsprechende Validierungen bei Zuweisung der Werte erfolgen.

4.2.3 Abstraktion des Datenzugriffs

Um die eigentliche Programmlogik von den technischen Details der Datenhaltung zu befreien, wird das Entwurfsmuster des *Datenzugriffsobjekts* (engl. *Data Access Object / DAO*) genutzt. Zu jedem der im System vorhandenen Entitäts-Objekte wird eine sogenannte DAO-Schnittstelle zur Verfügung gestellt, die Methoden spezifiziert, um neue Datensätze zu erstellen, bestehende Datensätze zu ändern oder zu löschen oder anhand bestimmter Filterkriterien den Zugriff auf eine Untermenge an gespeicherten Daten zu erlauben. Die `GameEventDAO` Schnittstelle sieht beispielsweise eine Methode vor, mit der sich die Objekte aller `GameEvents` in einem bestimmten Radius einer definierten GPS-Position ermitteln lassen.

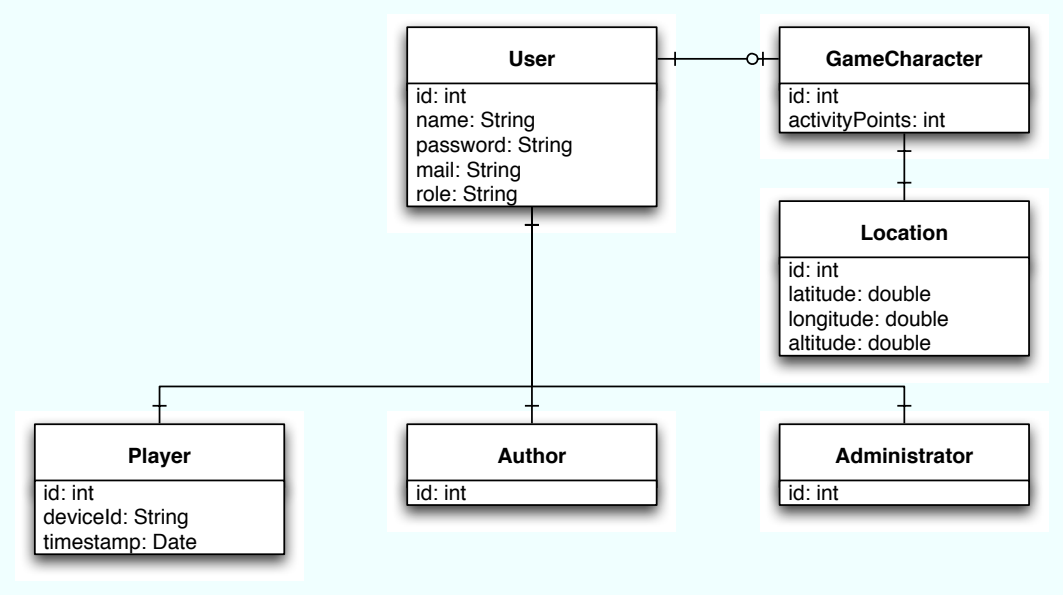


Abbildung 4.7: Entity Relationship Diagramm des `Users`

Die Verwendung von Schnittstellen bietet hier einen großen Vorteil: bei der Art des Datenzugriffs kann eine große Flexibilität erreicht werden, da die Anwendungslogik weder Kenntnis über die konkrete Implementierung der DAO-Objekte, noch über die zugrunde liegende Datenquelle hat.

Die Anwendungslogik des `MessagingService` hat lediglich Kenntnis über die Methoden der DAO-Schnittstelle `GameMessageDAO` (siehe Abbildung 4.8). Die konkrete Implementierung `HibernateGameMessageDAOImpl` dieser Schnittstelle realisiert die Abbildung der Datenstrukturen auf die physikalische Datenhaltung und beachtet hierbei die spezifischen Anforderungen der zugrunde liegenden Technik – in diesem Falle den Zugriff von Hibernate auf eine MySQL Datenbank. Dem `MessagingService` wird somit der Zugriff auf das Daten-Objekt `GameMessage` gestattet, ohne Kenntnis von der Datenhaltung sowie dem Zugriff auf die Datenquelle zu haben.

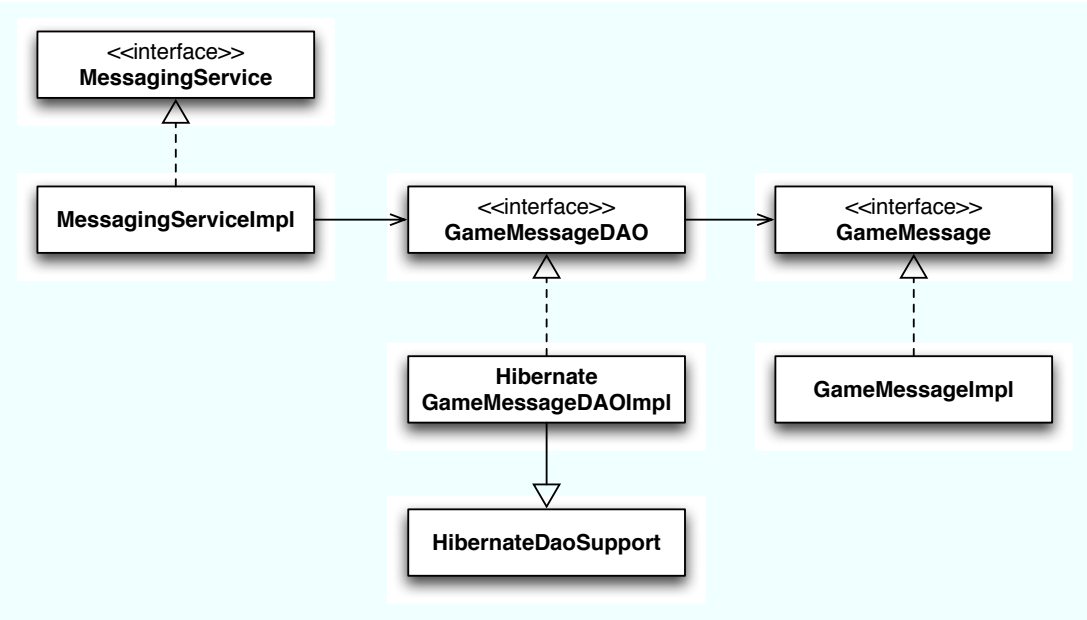


Abbildung 4.8: Spring Hibernate Datenzugriffsobjekt-Implementierung gegen Interfaces

4.3 Anwendungslogik-Schicht

Die *Anwendungsschicht* beinhaltet die Geschäftslogik der Anwendung. Sie verarbeitet die von der Präsentations-Schicht weitergeleiteten Eingaben des Nutzers, kommuniziert mit der Datenhaltungs-Schicht und produziert Daten für die Ausgabe. Es werden sowohl Schnittstellen für lokale Methodenaufrufe, als auch für entfernte Aufrufe im Sinne von Web Services zur Verfügung gestellt.

Das zu entwickelnde System stellt unterschiedliche Module bereit, welche bereits bei der Beschreibung des Spielekonzepts angedeutet wurden:

- das Sammeln der Sensordaten
- die Vorverarbeitung und Analyse der kontinuierlichen Sensordaten
- die Identifizierung der Aktivität des Spielers
- die Bestimmung passender Wettkampf-Partner

In Abhängigkeit der vorhandenen Infrastruktur werden diese Module in unterschiedlichen Zusammenstellungen sowohl in der Spiele- als auch der Verwaltungs-Anwendung kombiniert. Bevor die Module besprochen werden, wird daher ein Überblick über ihre Verwendung gegeben.

4.3.1 Anwendungslogik der Verwaltungs-Anwendung

Die Anwendungslogik der Verwaltungs-Anwendung beschränkt sich, neben der Bereitstellung der für die Spiele-Anwendung benötigten Web Services, darauf, die von der Präsentations-Schicht weitergeleiteten Nutzer-Interaktionen in Abläufe zur Manipulation der Datenbasis umzusetzen. Die Umsetzung dessen erfolgt im Sinne einer klassischen Verwaltungsanwendung. Solche Ansätze werden in der Literatur bereits sehr ausführlich diskutiert. Infolgedessen soll in dieser Arbeit nicht näher darauf eingegangen werden. Stattdessen konzentriert sich der Entwurf auf die Bereitstellung der Web Services und die Bestimmung passender Gegner.

In Abbildung 4.9 wird die Kommunikation und der Ablauf der Verarbeitung der durch die Verwaltungs-Anwendung eingesetzten Module schematisch dargestellt.

Die in der Präsentations-Schicht eingegebenen Informationen zur Erstellung, Verwaltung oder dem Löschen von Daten werden von der Anwendungslogik validiert. Wird dabei eine Inkonsistenz in den Daten festgestellt, bricht die Verarbeitung mit einer entsprechenden

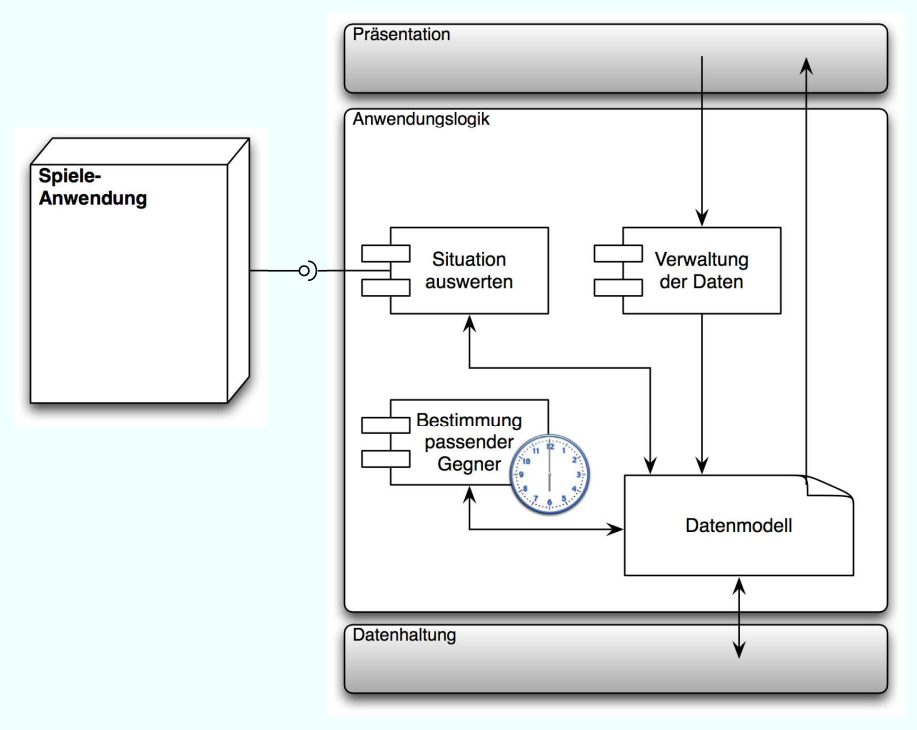


Abbildung 4.9: Übersicht der Integration der Anwendungslogik-Schicht ins System der Verwaltungs-Anwendung

Fehlermeldung ab und übergibt diese an die Präsentationsschicht, um sie zu visualisieren. Bei Erfolg werden die Informationen an das zentrale Datenmodell mit Hilfe der DAO-Klassen übergeben. Möchte ein Autor beispielsweise ein Element der Storyline ändern, so muss das entsprechende Objekt im ersten Schritt durch die Datenhaltung zur Verfügung gestellt werden. Über das `HibernateStoryElementDAO` kann das Referenz-Objekt über seine `id` geladen und als Instanz des `StoryElement` zurück gegeben werden. Die Änderungen an dieser Instanz werden nach Beendigung wiederum über eine entsprechende Schnittstelle der DAO-Klasse persistent gemacht.

Neben der Verarbeitung der durch Autoren und Administratoren vorgenommen Änderungen, stellt die Verwaltungs-Awendung außerdem Web Services für die Spiele-Anwendung bereit. Diese bieten die zur Durchführung des Spiels benötigten Dienste an: die Registrierung, die An- und Abmeldung der Spieler am Server, die situationsabhängige Abfrage neuer Aufgaben und StoryElemente und die positionsabhängige Ermittlung der Spieler, die sich in der Nähe befinden anhand der Übertragung des aktuellen Spielstatus. Bei aktivierter Online-Funktionalität (vgl. Abschnitt 3.2.2) überträgt der Client demzufolge kontinuierlich die aktuelle Situation des Spielers: fährt dieser gerade Rad, joggt er oder ist er gerade nicht aktiv. Diese Informationen werden zusätzlich ergänzt durch die aktuellen Position und weitere relevante Daten[5] des Spielers. Diese Daten werden durch den

[5]Die Relevanz der Daten ist abhängig von dem aktuellen Kontext des Nutzers: bei einer Sportart, bei

Server verarbeitet und ausgewertet: das aktuelle Profil des Nutzers wird entsprechend der aktuellen Daten auf dem Server angepasst und relevante Aufgaben und StoryElemente werden aus der Datenbank geladen und dem Client in Textform im Body der Response zur Verfügung gestellt. Um dies zu gewährleisten werden wiederum mit Hilfe der DAO-Klassen entsprechende Referenz-Objekte aus dem Datenmodell geladen, indem diese anhand der Kriterien der Situation und GPS-Position des Nutzers gefiltert und geladen werden.

Zusätzlich werden durch den Server Wettkämpfe für aktive Spieler generiert. Auf diese Funktionalität soll im Folgenden etwas genauer eingegangen werden.

4.3.1.1 Bestimmung passender Gegener

Die an den Client zurückgegebenen Daten können durch zufällig durch den Server erstellte Herausforderungen ergänzt werden. In Abhängigkeit der Anzahl der momentan aktiven Spieler wird am Server das Modul `CompetitionLogic` ausgeführt. Dieses wertet die Profile der momentan am Server angemeldeten Spieler aus und identifiziert zwei Spieler mit ähnlichen Profilen. Hierbei werden vor allem die Best- und Durchschnittswerte der Spieler sowie deren sportliche Aktivität betrachtet. Wurden zwei Spieler bestimmt, die sich leistungsmäßig auf einem ähnlichen Niveau befinden, so wird automatisch eine Aufgabe generiert, die beiden Spielern zugesendet wird. Diese Herausforderung muss von beiden allerdings nicht synchron bewerkstelligt werden. Sobald beide Spieler die Aufgabe abgeschlossen haben, wird der Gewinner dieser Herausforderung bestimmt: dieser erhält als Bonus Aktivitätspunkte, die durch die Schwierigkeitsstufe der Aufgabe bestimmt werden.

4.3.1.2 Entwurf benötigter Web Services

Die Verwaltungs-Anwendung bietet neben einer Benutzeroberfläche für die Administratoren und Autoren des Spiels auch Schnittstellen für mobile Geräte an. Diese ermöglichen es dem Mobilgerät die zur Umsetzung der Spielelogik benötigten Daten aus der zentralen Datenbank abzurufen und den Spielfortschritt des Spielers zu speichern. Die dafür notwendige Kommunikation zwischen den Anwendungen läuft in mehreren unabhängigen Schritten ab, die im Folgenden genauer vorgestellt werden sollen. Ein Überblick über die im System benötigten Web Services wird zudem in Tabelle 4.1 gegeben.

Dies geschieht durch Parametrisierung der Anfrage mit der aktuellen Position und Situation des Spielers. Die Antwort des Server hängt jedoch nicht ausschließlich von der

der man eine Strecke zurücklegt, wie beispielsweise dem Laufen oder Fahrrad fahren, betrifft diese die aktuelle Geschwindigkeit oder die Durchschnittsgeschwindigkeit, anderenfalls könnte beispielsweise die Anzahl von Sprüngen von Interesse sein.

Name	Funktion
`registerPlayer`	Ermöglicht die Registrierung eines neuen Spielers auf der Plattform.
`initPlaying`	Der Status des Spielers auf der Plattform wird auf *aktiv* gesetzt. Er kann nun das Spiel spielen und Aktivätspunkte sammeln indem er sich bewegt und Aufgaben erledigt.
`transmitState`	Die aktuelle Situation des Spielers – dessen Position und Aktivität – wird in der Datenbasis aktualisiert.
`transmitSettings`	Änderungen der spezifischen Einstellungen des Spielers können initiiert werden.
`transmitActivity`	Eine komplette Aktivität wird nach Beendigung übertragen.
`endPlaying`	Der Status des Spielers wird auf *inaktiv* gesetzt.

Tabelle 4.1: Übersicht über die Funktionalität der in der Verwaltungs-Anwendung zu implementierenden Web Services.

aktuellen Position ab, sondern auch vom Profil des Nutzers. Um eine eindeutige Zuordnung zu ermöglichen, wird zusätzlich die Angabe einer eindeutigen Spieler-ID benötigt. Daher wird in erster Instanz vom Server eine Funktion angeboten, die es ermöglicht, den Spieler zu anzumelden. Er erhält daraufhin eine eindeutige Spieler-ID, die er für weitere Anfragen an benutzen muss.

Neben Funktionen, die es ermöglichen, spezifische Aufgaben und StoryElemente abzurufen, werden Services angeboten, die es ermöglichen eine Aktivität und deren Kontext in der Datenbasis des Servers zu speichern. Wurde diese Aktivität durchgeführt, ohne dass dem Spieler eine entsprechende Aufgabe gestellt wurde, werden lediglich die beschreibenden Daten dieser Aktivität übermittelt. Handelt es sich dahingegen um den Versuch eine Aufgabe zu bewältigen, so werden zusätzliche Parameter benötigt, ob und wie diese bewältigt wurde.

4.3.2 Anwendungslogik der Spiele-Anwendung

Die Anwendungslogik der Spiele-Anwendung kann grundsätzlich in drei unterschiedliche Kategorien eingeteilt werden:

- die Verarbeitung und Analyse der Sensordaten zur Bestimmung der aktuellen Situation,

- die Kommunikation mit dem Verwaltungs-Server und
- die Umsetzung der Spielelogik.

Gesteuert und verwaltet werden diese Funktionen durch die Controller-Klasse der Anwendung, einer Instanz des `SportixAppController`. Alle Daten-Manipulationen werden hierbei auf einer Instanz der Klasse `SportixAppModel` ausgeführt, dieses zentrale Datenmodell der Anwendung bildet das Bindeglied zwischen Präsentations- und Datenhaltungs-Schicht. In Abbildung 4.10 wird deshalb schematisch der im Folgenden beschriebene Ablauf schematisch dargestellt.

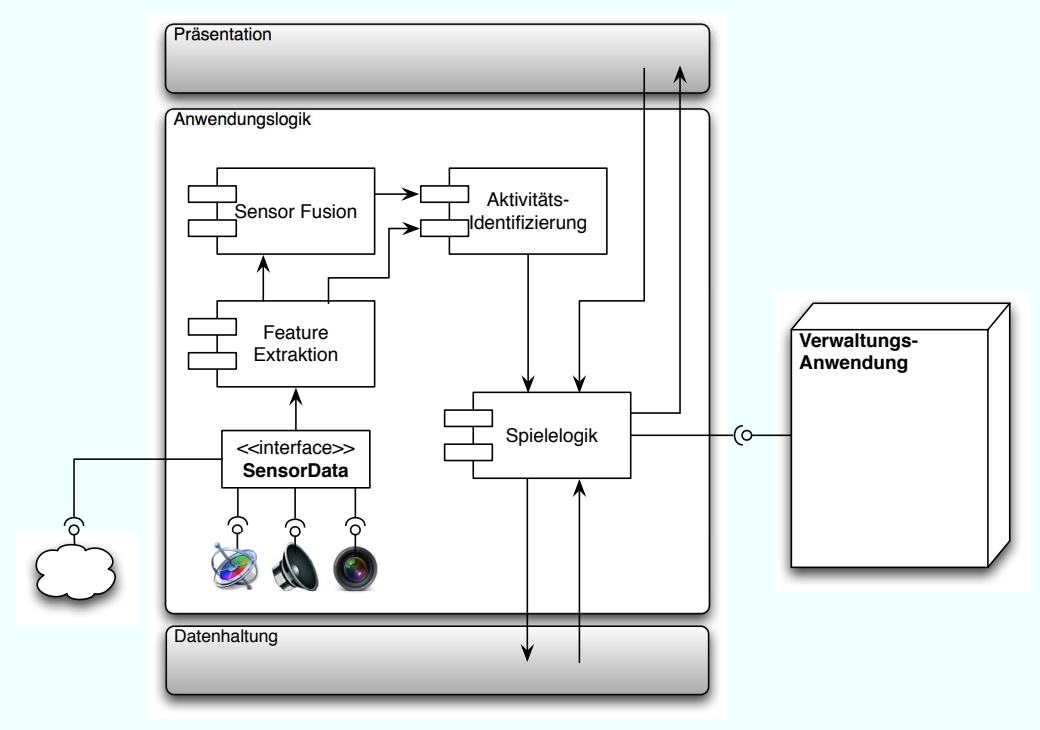


Abbildung 4.10: Übersicht der Integration der Anwendungslogik-Schicht ins System der Spiele-Anwendung

Bei der Instanziierung des `SportixAppController` wird zunächst eine weitere Controller-Klasse, der `SportixContextController` instanziiert, dieser kapselt die komplette Funktionalität zum Sammeln der Sensordaten, zum Extrahieren der Features sowie zum Klassifizieren der aktuellen Situation des Nutzers. Die konkrete Aktivität wird über eine entsprechende Schnittstelle an den `SportixController` übergeben. Dieser stellt die Schnittstelle zur Manipulation der Datenbasis dar. Die aktuelle Aktivität kann ohne weitere Verarbeitungsschritte an das Datenmodell weitergeleitet und dort verarbeitet werden.

Außerdem stellt die Kommunikation mit der Verwaltungs-Anwendung einen weiteren großen Bereich der Anwendungslogik dar. In zeitlich diskreten Abständen wird die aktuelle Aktivität und Position des Spielers an den Server übertragen. Diese Daten werden

durch den Server ausgewertet (vgl. Abschnitt 4.3.1). Zusätzlich werden Anfragen an den Server gestellt, um aktuell verfügbare Aufgaben und Elemente der Story für den Spieler zu erfragen. Diese werden im ersten Schritt der Implementierung durch die aktuelle Position des Nutzers bestimmt: Elemente, die in einem bestimmten Umkreis verfügbar sind, werden an den Client übertragen. Zusätzlich kann diese Überprüfung in einem weiteren Schritt in Abhängigkeit der Situation des Nutzers durchgeführt werden. Beispielsweise können diese an Bedingungen gekoppelt sein, dass der Nutzer bereits einen bestimmten Status im Spiel erreicht hat.

Eine wichtige Rolle bei der Durchführung des Spiels nehmen die Aktivitätspunkte ein: sie repräsentieren den Fortschritt im Spiel. Im Vergleich zu anderen Spielen, werden diese allerdings nicht konkret mit der Durchführung einer speziellen Aufgabe verknüpft. Spezielle Aufgaben ermöglichen es dem Spieler lediglich, zusätzliche Bonuspunkte zu verdienen. Auch wenn der Spieler demzufolge keine Aufgabe bearbeitet, kann er seinen Level im Spiel erhöhen: die Punktzahlen sollen sich dynamisch und flexibel nach der jeweiligen Situationen des Nutzers richten. Spieler werden deshalb entsprechend der Intensität der ausgeführten Aktivitäten mit einer entsprechenden Punktzahl honoriert.

Da von den Spielern keine spezifischen Details zum Alter und Gewicht angegeben werden müssen, soll diese Berechnung auf einem relativen Vergleich des Energieverbrauchs einer Person basieren. Es wird daher eine Schnittstelle entworfen, die aktivitätsspezifische Kennzahlen auswertet und anhand dessen einen relativen Vergleichswert zur Verfügung stellt – dieser repräsentiert gleichzeitig die Punktzahl, die dem Spieler infolgedessen gutgeschrieben wird.

4.3.2.1 Verarbeitung und Analyse der Sensordaten

Die Analyse und Auswertung von Sensordaten stellt einen zentralen Teil einer kontextsensitiven Anwendung dar. Dabei ist es egal, ob es sich um physikalische, logische oder virtuelle Sensoren handelt. Bei der Implementierung soll vorwiegend beispielhaft die Auswertung physikalischer Sensoren in Betracht gezogen werden: die Aggregation der Merkmale der Positions- und Beschleunigungs-Daten sollen Aufschluss über die aktuelle Aktivität des Spielers geben. Die Features der GPS-Daten sollen dabei den überwiegenden Teil der Analysen ausmachen.

Um diesen Bereich der Anwendungslogik möglichst flexibel und erweiterbar zu gestalten, wird der Prozess der Kontextgewinnung abstrahiert. Die unterschiedlichen Sensoren werden durch die Klasse `SensorData` repräsentiert. Diese ermöglicht die Kapselung sehr unterschiedlicher Sensoren und stellt die Schnittstelle für den Entwickler dar. Dementsprechend werden die Klassen `LocationSensorData` und `AccelerationSensorData`

entworfen. Die Implementierung eines entsprechenden Protokolls ermöglicht es, den `SensorController` über die aktuellsten Daten zu informieren. Dieser verarbeitet die Rohdaten und stellt spezifische Features bereit. Wird von dem SensorController eine Änderung der aktuellen Aktivität registriert, so wird ein Event am `SportixAppController` erzeugt. Dessen Aufgabe ist es nun, die Spielelogik entsprechend der aktuellen Aktivität anzupassen.

In der ersten Instanz der Entwicklung soll die Spiele-Anwendung lediglich lineare sportliche Aktivitäten unterstützen. Prinzipiell soll die Geschwindigkeit des Spielers bestimmt werden. Anhand dieser soll eine erste logische Einschätzung anhand eines Entscheidungsdiagramms vorgenommen werden. Um die Aktivität letztendlich konkret zu spezifizieren, sollen die Features der Beschleunigungssensoren in die Betrachtung einbezogen werden. Diese sollen die Intensität der aktuellen Bewegung kennzeichnen.

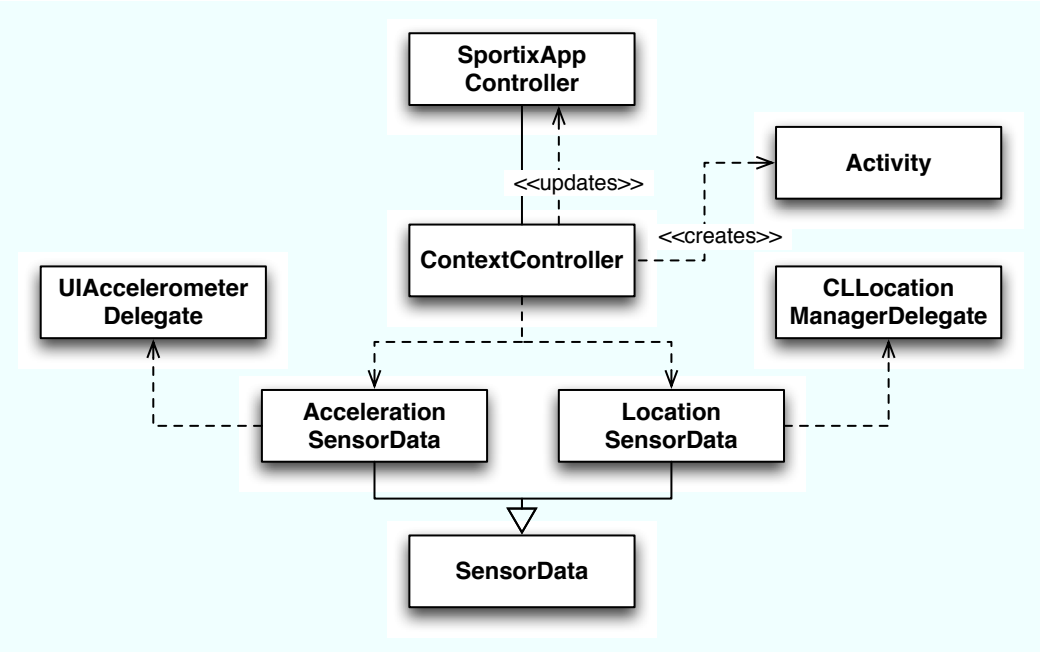


Abbildung 4.11: Beziehungen der Controller-Klasse `ContextController` zu den Modulen der Anwendungslogik

4.4 Präsentationsschicht

Die *Präsentationsschicht* stellt die Schnittstelle des Systems zum Nutzer dar und dient der Darstellung der Inhalte sowie der Entgegennahme von Eingaben.

Die Benutzeroberflächen der Spiele- als auch der Verwaltungs-Anwendung genügen unterschiedlichen Anforderungen und werden deshalb nachfolgend separat besprochen.

4.4.1 Benutzeroberfläche der Verwaltungsanwendung

Die Benutzeroberfläche der Verwaltungs-Anwendung bildet die in der Datenbasis gespeicherten Informationen zur Storyline und zu denen im Spiel verfügbaren Aufgaben für die *Autoren* des Systems grafisch ab und erlaubt eine bequeme Verwaltung, sowie das Einpflegen neuer Daten. Darüber hinaus wird es dem *Administrator* ermöglicht Nutzerdaten grafisch einzusehen, zu verwalten und Rechte im System zuzuweisen.

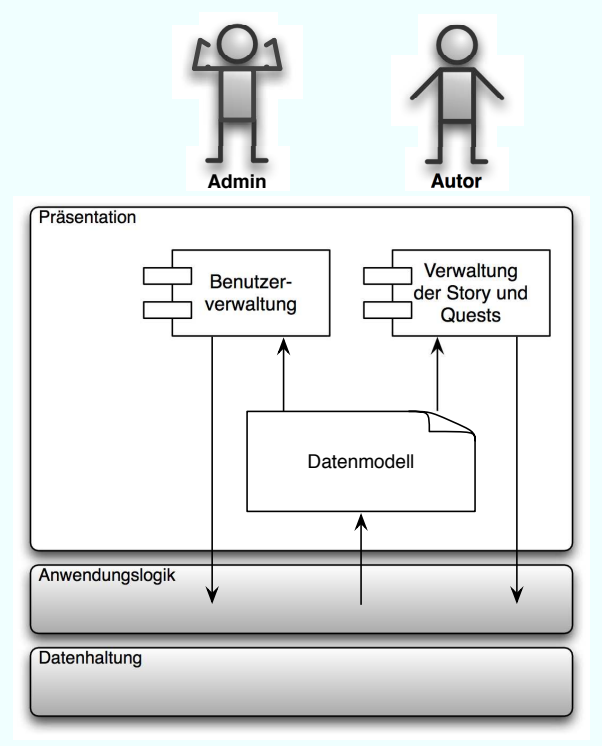


Abbildung 4.12: Präsentations-Schicht der Verwaltungs-Anwendung

Neben der Eingabe und Bearbeitung der textuellen und grafischen Informationen der Event-Stammdaten und -Informationen wird auch die Verwaltung der Bedingungen der durch den Spieler zu erfüllenden Kriterien unterstützt.

Eingabedaten werden zur weiteren Verarbeitung an die Anwendungslogik weitergeleitet,

welche diese validiert und an die Schnittstelle zur persistenten Speicherung weiterleitet. Zusammen mit bereits gespeicherten Datensätzen stellt die Anwendungslogik diese der Benutzeroberfläche in einem Modell gekapselt für die Visualisierung und interaktive Bearbeitung zur Verfügung. In Abbildung 4.12 werden diese zentralen Komponenten und deren Zusammenspiel skizziert.

Neben der Anzeige der zur Umsetzung des Spiels benötigten Daten wird zudem die Darstellung statistischer sowie aktueller Daten der Spieler im System[6] ermöglicht. Dies gestattet es Nutzern mit entsprechenden Rechten Schlussfolgerungen über die Akzeptanz und Nutzung des Systems zu ziehen.

Der Zugriff auf diese Benutzeroberfläche erfolgt über den Webbrowser. Üblicherweise werden bei Webanwendungen für die Umsetzung der Präsentationsschicht Frameworks, beispielsweise Struts, JavaServer Faces oder JavaServer Pages, verwendet.

4.4.1.1 Bedienkonzept

Mitentscheidend für den Erfolg einer Anwendung ist eine klar strukturierte Darstellung der Inhalte mit einem schlüssigen Navigationskonzept. Eine einfache Handhabung, die die selbstbeschreibende Interaktion durch den Nutzer ermöglicht, ist ebenso wichtig, wie die konstante qualifizierte Rückmeldung über den Erfolg oder Misserfolg einer getätigten Aktion.

Das ständige Vorhandensein der zur Verfügung gestellten Funktionalitäten ist daher ein wichtiges Kriterium, dass bei der Umsetzung des Systems beachtet wird. Jedwede Änderung soll schnellstmöglich realisiert werden können.

In jedem der Bereiche der Verwaltungs-Anwendung – Bearbeitung der Nutzerdaten, Verwaltung der StoryElemente und Verwaltung der Aufgaben – wird dem Nutzer außerdem eine Übersicht über die im System eingepflegten Daten visualisiert. Die Auswahl eines Objekts ermöglicht daraufhin die weitere Bearbeitung oder gar das Löschen der Daten. Entsprechend der zugewiesenen Rechte, haben die Nutzer allerdings unterschiedliche Bereiche zur Auswahl.

Dem Autor wird die Möglichkeit geboten, neue `StoryElemente` anzulegen, vorhandene Einträge zu löschen und an einem Story-Objekt vorgenommene Änderungen dauerhaft zu speichern. Für das aktuell ausgewählte Story-Objekt können über Text-Eingabefelder die Stammdaten des `StoryElements` (Name und Beschreibung) editiert werden. Außer-

[6] Unter Berücksichtigung der Einschränkungen zur Wahrung des Datenschutzes werden die Daten nur erhoben, wenn der Spieler darin eingewilligt hat. Außerdem wird keine eindeutige Zuordnung der Daten zu den Nutzern ermöglicht (*Anonymisierung*).

dem können diesem Objekt Kriterien zugeordnet werden, die durch den Spieler erfüllt werden müssen, um dieses Story-Objekt entsprechend frei zu schalten. Die Verwaltung der Aufgaben wird zugunsten der besseren Übersichtlichkeit in einem eigenen Bereich der Benutzeroberfläche realisiert.

Auch hier erhält der Autor zunächst eine Übersicht der bereits eingepflegten Informationen: sowohl die Aufgaben, die durch Autoren erstellt wurden als auch die der Spieler. Aus dieser Liste können Einträge für die weitere Bearbeitung ausgewählt werden, zudem können vorhandene Parameter aus der Beschreibung der Aufgabe gelöscht oder neue über einen Auswahldialog hinzugefügt werden.

Die Oberfläche bietet zusätzlich für Administratoren des Systems die Möglichkeit, die Daten der Nutzer anzuzeigen, zu verwalten oder zu löschen. Das Hauptaugenmerk liegt hierbei in der Zuordnung entsprechender Rechte zu einem Nutzer.

4.4.2 Benutzeroberfläche der Spieleanwendung

Im Gegensatz zu der Benutzeroberfläche der Verwaltungsanwendung bietet das Display eines mobilen Gerätes nur eine begrenzte physikalische Darstellungsfläche. Die visualisierte Funktionalität muss deshalb, situations- und positionsabhängig, auf die nötigsten Komponenten beschränkt werden.

Ferner soll der Ablauf des Spiels möglichst automatisiert ablaufen, daher wird die Benutzeroberfläche der Spiele-Anwendung bewusst wenig Funktionalität für die Nutzerinteraktion zur Verfügung stellen. Es ist die Aufgabe der Anwendung, die Aktivitäten und Intentionen des Nutzers zu identifizieren und die entsprechenden Aktionen durchzuführen. Sie muss dem Nutzer jedoch ausreichend Feedback geben, indem grundlegende Handlungen visualisiert werden, die Verarbeitung von Details jedoch im Hintergrund bleibt. Zudem muss das Ergebnis der Handlungen – der Fortschritt im Spiel – präsentiert werden. Abbildung 4.13 visualisiert dies.

Auch die Human Interface Guidelines von Apple (vgl. [13]) geben einige Aspekte des Designs vor, die die Entwickler beachten und umsetzen sollen.
Neben ästhetischen Aspekten der Gestaltung steht vor allem der Nutzer im Vordergrund der Entwicklung: kritische Aktionen müssen explizit vom Nutzer initiiert werden, nur er hat das Recht diese zu kontrollieren. Dabei ist zu beachten, dass Objekte nur so lange auf dem Bildschirm sichtbar sein dürfen, wie die entsprechenden Aktionen ausgeführt werden. Die Resultate dessen sollen daraufhin sofort eingeblendet werden. Hierbei ist das Feedback für den Nutzer von besonderer Bedeutung – auf jede Aktion soll möglichst eine Animation folgen.

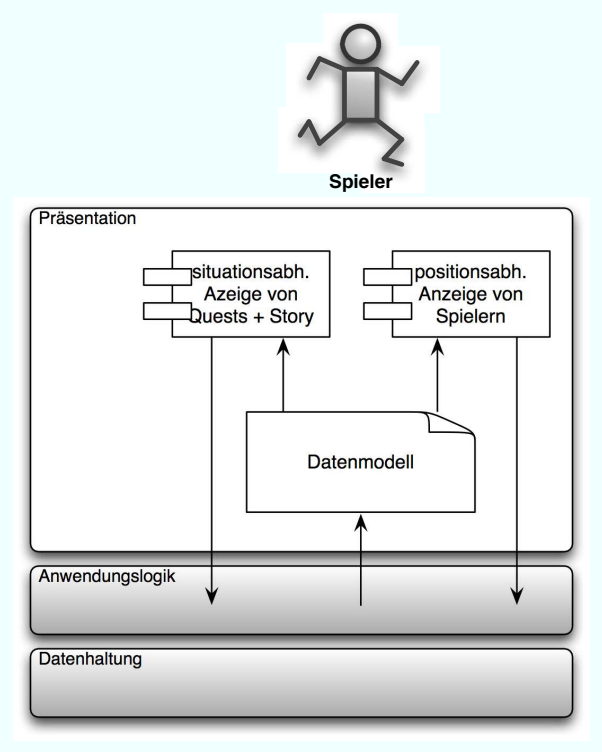


Abbildung 4.13: Präsentations-Schicht der Spiele-Anwendung

Prinzipiell nutzt das iPhone OS, ähnlich wie Mac OS X, *Windows* und *Views*, um graphische Inhalte auf dem Bildschirm darzustellen. Obwohl beispielsweise Windows in Mac OS X eine sehr große Rolle spielen, ist dies in iPhone Applikationen signifikant reduziert: es existiert nur eine Instanz, welche durch die Klasse `UIWindow` repräsentiert wird. Sie wird beim Start der Applikation instanziiert und ist für die Verwaltung einer oder mehrerer Views zuständig. Diese spielen eine sehr wichtige Rolle, da sie sowohl die darzustellenden Inhalte auf dem Bildschirm rendern, als auch auf `UITouchEvent`s reagieren. Sie stellen demnach die primäre Schnittstelle für die Interaktion mit dem Nutzer dar und sind deshalb für den Entwurf der Benutzeroberfläche der Anwendung von besonderer Bedeutung.

In iPhone Applikationen besteht überdies eine enge Verbindung zwischen Views und ViewControllern (`UIViewController`). ViewController kümmern sich um das Laden und Entfernen von Views, um die Rotation des Interfaces und die Interaktion mit Navigations-Komponenten, die genutzt werden, um komplexe Benutzeroberflächen zu erstellen.

Die Darstellung situations- und positionsabhängiger Informationen erfordert den Entwurf einer übersichtlich und intuitiv gestalteten Navigationsstruktur. Abbildung 4.14 zeigt einen möglichen Aufbau der Benutzeroberfläche einer iPhone-Applikation. Im oberen Bereich befindet sich die `UINavigationBar`. Diese ermöglicht die Navigation zwischen inhaltlich zusammenhängenden Bereichen der Applikation. `UIBarButtonItem`s ermöglichen

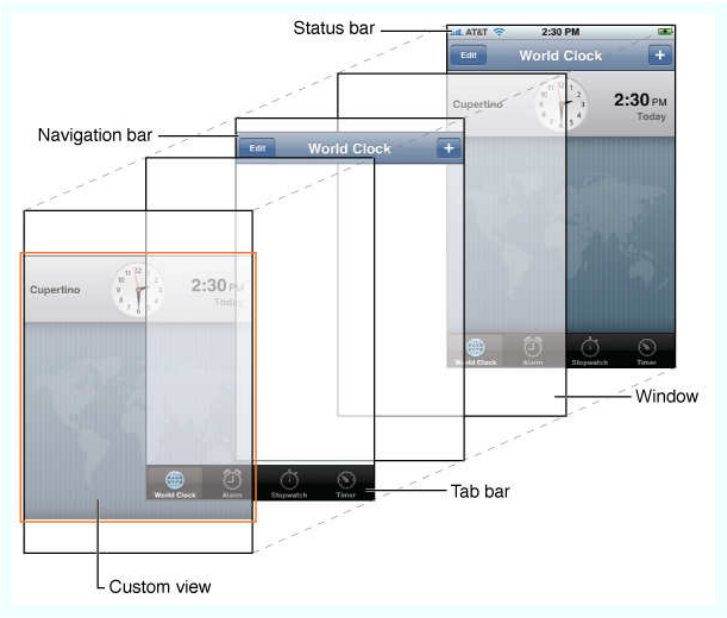


Abbildung 4.14: Komponenten einer iPhone-Applikation (Quelle: [14])

die Navigation zum vorhergehenden oder folgenden Screens. Auf diese Weise kann ein hierarchischer Aufbau der Applikation umgesetzt werden.

Im unteren Bereich des Bildschirms kann sich eine `UITabBar` oder eine `UIToolBar` befinden. Prinzipiell wird mit einer `UIToolBar` eine Anzahl von Aktionen im aktuellen Kontext angeboten. Exemplarisch wird in der Abbildung allerdings eine `UITabBar` dargestellt: diese setzt die Funktionalität um, dem Nutzer verschiedene Perspektiven auf die gleichen Daten anzubieten. Bei Auswahl eines der Elemente der `UITTabBar`, wird eine neue Ansicht generiert und im Zentrum des Bildes visualisiert.

Im Bild handelt es sich um eine `UITableView`, die dem Nutzer verschiedene Möglichkeiten der Einstellung bietet. Speziell ist bei solchen Optionen darauf zu achten, den Nutzer möglichst wenig Texteingaben machen zu lassen. Es ist besser diesem Listen zur Auswahl zu bieten.

4.4.2.1 Bedienkonzept

Der Nutzer soll bei der Bedienung der Applikation weitestgehend auf explizite manuelle Eingaben verzichten können. Die manuelle Eingabe von Text auf mobilen Geräten kann unter Umständen sehr schwierig sein. Unter Berücksichtigung der Tatsache, dass ein Spiel entworfen wird, bei dem man körperlich aktiv sein muss, gewinnt dieser Aspekt zusehends an Bedeutung. Ziel muss es demnach sein, diese zu minimieren und den Kontrollfluss des Spiels zu automatisieren.

In Abhängigkeit der aktuellen Aktivität und Position des Nutzers sollen deshalb automatische Adaptionen der Visualisierung vorgenommen werden. Prinzipiell stehen dem Nutzer allerdings unterschiedliche Ansichten des Spiels zur Verfügung:

- Liveansicht
- Listenansicht der Aufgaben, die sich in der Nähe befinden
- Listenansicht der Spieler, die sich in der Nähe befinden
- Statistik
- Einstellungen

Da das iPhone über ein *Multi-Touch Display* verfügt, muss bei dem Entwurf der Oberfläche speziell Rücksicht darauf genommen werden, dass die Größe der Schaltflächen entsprechend umgesetzt wird: der Spieler soll nicht aus Versehen eine Funktionalität aktivieren, weil er den gewünschten Button nicht millimetergenau angetippt hat. Auch die systemweit etablierten Gesten sollen bei der Umsetzung unterstützt und implementiert werden, um beispielsweise die Liste von Aufgaben in der Nähe schnell durchsuchen zu können und somit die Integration der Applikation in das bestehende iPhone OS zu gewährleisten und die Usability der Applikation zu steigern. Zudem soll die Nutzung mehrerer Finger keine unerwünschten Nebeneffekte erzielen.

Die Liveansicht des Spiels soll als zentrale Schnittstelle zum Spieler darstellen. Auf dieser visualisiert das System die aktuelle Aktivität, indem ein entsprechendes Icon angezeigt wird. Dort können zudem weitere ermittelte Werte, die zur Erfüllung spielrelevanter Aufgaben von Interesse sind, eingesehen werden. Sobald neue Informationen oder Teile der Story verfügbar sind, wird der Nutzer entsprechend benachrichtigt. Dies geschieht, in Abhängigkeit der Situation, auf akustischem oder visuellem Weg.

Neben der Bereitstellung dieser Grundfunktionalitäten ist es sinnvoll, in der prototypischen Anwendung zusätzlich Informationen über den aktuellen Status der Identifizierung zu visualisieren. Dies vereinfacht die Analyse korrekter und fehlerhafter Identifizierungsergebnisse und ermöglicht eine Einschätzung der Leistungsfähigkeit der zugrunde liegenden Algorithmen.

Im Vordergrund steht demnach die situationsabhängige Darstellung von Informationen. Es werden keine komplexen Interaktionen des Spielers mit dem System vorausgesetzt.

Abbildung 4.15 skizziert den Aufbau ausgewählter Bereiche der Benutzeroberfläche.

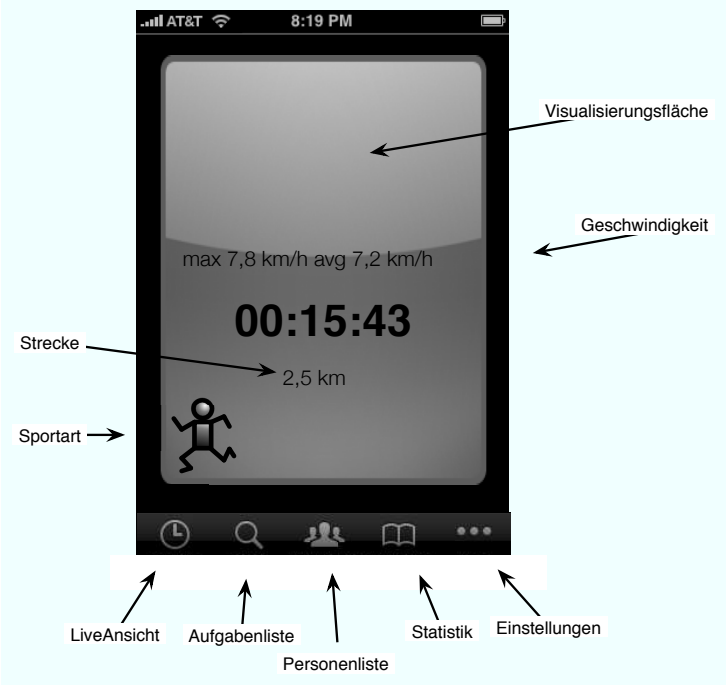


Abbildung 4.15: Bedienkonzept der Spiele-Anwendung

4.4.2.2 Klassen-Konzept

Der zentrale Teil der Benutzeroberfläche der Spiele-Anwendung wird durch die ViewController Klasse `SportixMainViewController` repräsentiert. Diese instanziiert und verwaltet sowohl die zentrale Navigationskomponente des Systems – den `SportixTabBarController` – als auch die unterschiedlichen Ansichten der Applikation. Zusätzlich wird ihr von dem Applikations-Delegate `SportixAppDelegate` die Instanz der Controller-Klasse `SportixAppController` übergeben. Dieser ermöglicht es, Änderungen am zentralen Datenmodell `SportixAppModel` zu initiieren. Nutzer-Aktionen werden demzufolge von den View-Komponenten an die Controller-Klasse übergeben und dort verarbeitet. Ändert sich hierbei das Datenmodell oder von ihm verwaltete Datenstrukturen, so werden entsprechende Notifications erzeugt und dem `NSNotificationCenter` der Applikation übergeben. ViewController, die die Änderung bestimmter Daten durch Views visualisieren, können sich für entsprechende Notifications registrieren. Dies hat den Vorteil, dass sich aktuelle Daten laufender Prozesse kontinuierlich aktualisieren und darstellen lassen.

Neben Standard-Darstellungskomponenten des iPhone SDK (`UITextField`, `UIImageView`, `UILabel`, etc.) werden auch systemspezifische Komponenten erstellt, da die vordefinierten Elemente nicht alle Anforderungen der Darstellung systemspezifischer Informationen und Zustände erfüllen können – beispielsweise die Darstellung der Übersicht und Detailan-

Implementierende Klasse	Funktionalität
`LiveViewController`	Visualisierung der LiveAnsicht des Spiels
`ListViewController`	Darstellung listenbasierter Informationen (`Player`, `GameQuest`)
`StatisticsViewController`	Visualisierung der Bestwerte des Spielers
`SettingsViewController`	Visualisierung grundlegender Einstellungen des Spiels

Tabelle 4.2: Übersicht einiger Implementierungen der Schnittstelle `UIViewController` und der umzusetzenden Funktionalität

sicht der Aufgaben des Spiels lassen sich durch die Standardkomponenten nicht auf die gewünschte Art und Weise visualisieren.

Daher werden unterschiedliche graphische Darstellungskomponenten bereitgestellt, um sowohl die gewünschten Animationen als auch Visualisierungen zu ermöglichen. Diese umfassen sowohl Spezialisierungen von `UITableViewCell`s zur Realisierung der Listen von Spielern und Aufgaben als auch spezielle `UIView`s die der Anzeige der Detailansichten dienen. Auch einzelne Komponenten der Liveansicht, wie beispielsweise der Fortschrittsbalken wird durch eine entsprechende spielspezifische Implementierung realisiert.

Im Prinzip wird jede Bildschirm-Ansicht auf dem iPhone durch einen ViewController repräsentiert. Um die gewünschten Funktionalitäten umzusetzen, gibt es daher im Spiel eine Reihe von ViewControllern, die sowohl das spezifische Verhalten (beispielsweise die Reaktion auf TouchEvents) als auch die Visualisierung steuern. In Tabelle 4.2 wird daher ein Auszug der im System vorgesehenen Implementierungen der Schnittstelle `UIViewController` gegeben.

Zusätzlich zu diesen spezifischen Klassen müssen weitere ViewController zur Verfügung gestellt werden, die beispielsweise die Fähigkeit umsetzen, auf TouchEvents des Nutzers zu reagieren oder bestimmte Komponenten der View zu animieren. Auf diese Komponenten soll hier allerdings nicht speziell eingegangen werden. Eine komplette Übersicht und Hierarchie dessen kann auf der beiliegenden CD eingesehen werden.

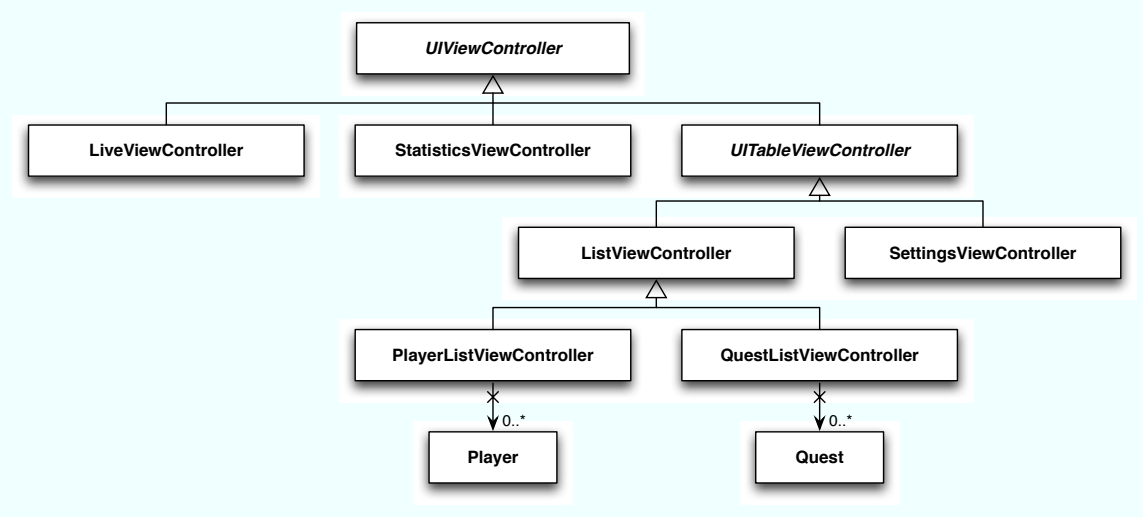


Abbildung 4.16: Klassendiagramm ausgewählter Implementierungen der Schnittstelle `UIViewController`

Kapitel 5

Implementierung

Im Rahmen einer prototypischen Implementierung wurde das im Kapitel „Systementwurf“ konzipierte System vollständig umgesetzt. Gegenstand dieses Kapitels wird es sein, ausgewählte Aspekte der Umsetzung darzulegen und die eingesetzten Technologien und verwendeten Werkzeuge vorzustellen.

5.1 Realisierung der Umsetzung

Bei der Umsetzung des Prototypen wurden für den Client als auch den Server grundlegend verschiedene Technologien und Programmiersprachen verwendet. Die Implementierung des Servers wurde vollständig in *Java* implementiert wohingegen der Client mit *Objective-C* realisiert wurde. Deshalb sollen nachfolgend die verwendeten Tools und Technologien separat aufgeführt und vorgestellt werden.

5.1.1 Realisierung der Server-Anwendung

Eclipse IDE Die *Eclipse IDE* unter Führung der Eclipse Foundation bietet unter anderem Unterstützung in Bereichen der Programmierung mit Java, C/C++ und Web Services. Zudem werden durch die Einbindung von Plug-Ins umfangreiche Erweiterungsmöglichkeiten zur Verfügung gestellt. Zudem ist die Eclipse IDE in Java geschrieben und kann somit plattformübergreifend eingesetzt werden.

Java SE 6 JDK Die Standard-Distribution des *Java Development Kit (JDK)* in Version 6 bietet dem Entwickler eine Laufzeitumgebung die zur Entwicklung und Ausführung von Java-Applikationen benötigt wird.

Apache Cocoon 2.2 Bei *Apache Cocoon* handelt es sich um ein Framework, dass seit Version 2.2 auf *Spring 2.0* [17] basiert und die Konzepte der *Separation of Concerns (SoC)* und komponentenbasierten Webentwicklung umsetzt. Diese Konzepte werden in Cocoon durch die Realisierung von Pipelines bewerkstelligt. Jede Komponente der Pipeline spezialisiert einen bestimmten Teil von Operationen. So wird es ermöglicht Weblösungen zu erstellen indem einzelne Komponenten miteinander zu einer Pipeline verbunden werden. Auf diese Weise werden einzelne Bereiche voneinander getrennt gehalten und somit eine parallele Entwicklung in allen Aspekten einer Webanwendung erlaubt, wobei zusätzlich die Geschwindigkeit dieser Entwicklung beschleunigt und Konflikte verringert werden können.

Die Integration von Spring in der aktuellen Version von Cocoon resultiert in einigen Vorteilen. Die Nutzung eines O/R-Mappers wie Hibernate wird dadurch beispielsweise erheblich erleichtert. Der Entwickler erhält neben einer Vereinheitlichung der Konfiguration vor allem ein durchgängiges und bequemes Programmiermodell. Auch das Management verschiedener Ressourcen, wie z.B. der Datenbankverbindung, wird durch Spring in einer einheitlichen Art und Weise übernommen.

Apache Axis 1.4 *Apache Axis* [7] ist eine Open Source Implementierung des Web Service Standards SOAP der Apache Software Foundation.

Apache Tomcat 5.5 Bei dem Apache Tomcat handelt es sich um einen Servlet-Container der die Laufzeitumgebung für die Verwaltungs-Anwendung darstellt. Er implementiert die offiziellen Spezifikationen der Java Server Pages und Java Servlets Technologien von Sun Microsystems. Damit wird es ermöglicht, Java-basierte Webanwendungen zu entwickeln und auszuführen. Zudem ist in Kombination mit Eclipse die Handhabung und Steuerung des Webservers sehr einfach umzusetzen.

MySQL Die Daten der Verwaltungs-Anwendung werden in einer relationalen Datenbank gespeichert. Im umgesetzten System kommt hierbei das relationale Datenbankmanagementsystem *MySQL 5.0 Community Server* [23] zum Einsatz. Es ist als freie Software unter der GPL erhältlich, kann aber auch unter kommerziellen Lizenzen mit abgestuften Garantien und Unterstützungen erworben werden. Der Zugriff der Anwendung auf die Datenbank wird über die *Java Database Connectivity (JDBC)* Schnittstelle im Zusammenspiel mit dem *MySQL Connector/J 5.0* [24] JDBC-Treiber realisiert.

Hibernate Bei Hibernate (vgl. Abschnitt 4.2.1) handelt es sich um einen Objekt-Relationalen Mapper, der es ermöglicht, JavaBeans in einer relationalen Datenbank abzubilden. Hibernate dient hierbei als Middleware und unterstützt somit die Umsetzung des MVC-Patterns.

5.1.2 Realisierung der Client-Applikation

iPhone SDK 2.1 Das *iPhone SDK* enthält die *XCode Development Tools*, eine IDE für Mac OS X, die aus XCode, dem Interface Builder und einer Sammlung von Build, Debugging und Performance Tools besteht. Mit Hilfe des Interface Builders lassen sich Benutzeroberflächen per Drag & Drop erstellen. Diese werden in XML Archiven, den so genannten Nib Dateien, gesichert. Außerdem ist im iPhone SDK noch der iPhone Simulator enthalten, der es erlaubt, Applikationen lokal auf dem Rechner zu testen und zu debuggen.

Bei dem SDK werden die folgenden Schichten unterschieden: der OS X Kernel, der den Zugriff auf hardwarenahe Funktionen ermöglicht. Diese Schicht wird abstrahiert vom Layer Core Services, welcher Threads, Netzwerkkommunikation und SQLite unterstützt. Darauf setzt schlussendlich der Media Layer und der Cocoa Touch Layer auf. Diese ermöglichen es beispielsweise Animationen umzusetzen, auf Multitouch-Events zu reagieren und die integrierten Sensoren zu nutzen.

5.2 Beschreibung der Implementierung

5.2.1 Realisierung der Datenhaltung

Die physische Datenhaltung wird zentral mit Hilfe des relationalen Datenbankmanagementsystems MySQL realisiert. Über einen Apache Webserver mit PHP-Unterstützung und der Applikation phpMyAdmin ist während der prototypischen Implementierung zudem eine webbasierte Kontrollmöglichkeit gegeben. Der Zugriff der Verwaltungs-Anwendung auf die Datenbank wird über die JDBC Schnittstellen mit dem entsprechenden JDBC-Treiber umgesetzt. Die Entwicklung der Relationen wurde durch einen Top-Down-Ansatz realisiert: die Relationen des RDBMS wurden anhand erstellter Entity-Klassen unter Nutzung von *Hibernate Annotations* automatisch durch Hibernate generiert (vgl. Abschnitt 5.2.3.1, S.97).

Neben der zentralen Datenhaltung spielrelevanter Daten werden auch auf dem mobilen Gerät benötigte Daten persistent gespeichert. Dies erfolgt unter Nutzung von SQLite und

so genannter `NSUserDefaults`. Mit Hilfe der UserDefaults werden Daten gespeichert, die das grundlegende Verhalten der Applikation beeinflussen: diese umfassen beispielsweise die Registrierungsdaten des Spielers oder seine Einstellungen bezüglich des Kommunikationsverhaltens der Anwendung. Daten die allerdings einer regelmäßigen Änderung unterliegen und beispielsweise für die Darstellung von Aufgaben oder StoryElementen oder der Kategorisierung der Aktivität benötigt werden, werden in der lokalen SQLite-Datenbank gespeichert.

5.2.2 Projektstruktur

5.2.2.1 Paketstruktur der Server-Anwendung

Die Verwaltungs-Anwendung besteht aus mehreren *Cocoon Blocks* – Cocoon Blocks stellen in Cocoon die Möglichkeit der Modularisierung zur Verfügung um die Erweiterbarkeit und Wartbarkeit der Anwendung zu garantieren. Aus diesen werden bei Distribution des Projekts Java Archive (jar) generiert, die wiederum in anderen Blocks integriert werden können. Der Workflow der Erstellung eines Blocks und die Einbindung dessen Abhängigkeiten wird in Cocoon durch die Nutzung von *Maven* realisiert.

Das Projekt setzt sich deshalb aus mehreren Teilprojekten zusammen. Diese wurden jeweils auf Unterpakete verteilt. Systemweit genutzte Klassen wurden analog zu dieser Vorgehensweise inhaltlich zusammengefasst und eigenen Unterpaketen zur Verfügung gestellt. Daraus ergibt sich folgende Paketstruktur:

- `de.sportix.auth`
- `de.sportix.exception` – Systemweit genutzte Klassen für Exceptions
- `de.sportix.logic`
- `de.sportix.persistence.entities` – Systemweit genutzte Entity-Klassen
- `de.sportix.persistence.entities.dao` – Abbildung der Entity-Klassen auf die Datenhaltung
- `de.sportix.security` – Komponenten zur Verschlüsselung sicherheitsrelevanter Daten
- `de.sportix.util` – Hilfsklassen
- `de.sportix.webservice`

Das Listing zeigt die Maven-Konfigurationsdatei, die die einzelnen Module der Applikation bündelt und ein einzelnes Projekt erstellt, welches nach der Erstellung der war-Datei im Applikationsserver veröffentlicht werden kann. Vorteil der Nutzung von Maven ist es, dass die durch eine Applikation benötigten Abhängigkeiten automatisch geladen und jeweils die neueste Version der Komponenten ins Projekt eingebunden werden. Dies klappt jedoch nicht bei der Erstellung der Konfigurationsdateien des Cocoon-Projekts. Die Lösung dieser Problematik basiert in diesem Fall auf dem Ausprobieren unterschiedlicher Kombinationen und Reihenfolgen der einzelnen Komponenten. Zusätzlich kann es vorkommen, dass spezielle Versionen von Komponenten händisch ausgeschlossen werden müssen, da es sonst zu nicht reparierbaren Fehlermeldungen beim Compilieren der Applikation kommen kann.

Listing 5.1: Maven-Konfigurationsdatei der Verwaltungs-Anwendung

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project>
3    <!-- more configuration -->
4    <dependencies>
5      <dependency>
6        <groupId>de.sportix</groupId>
7        <artifactId>auth</artifactId>
8        <version>1.0</version>
9      </dependency>
10     <dependency>
11       <groupId>de.sportix</groupId>
12       <artifactId>logic</groupId>
13       <version>1.0</version>
14     </dependency>
15     <!-- more special dependencies -->
16   </dependencies>
17 </project>
```

Auch die Pipelines, die in den einzelnen Modulen des Projekts erstellt wurden, können durch jedes andere Modul genutzt werden, wenn dies erwünscht ist. Um dies zu ermöglichen, müssen diese in der Datei `block-servlet-service.xml` miteinander gekoppelt werden. Um beispielsweise im Modul der Anwendungslogik direkt auf die im Block **de.sportix.security** definierten Sicherheitsmechanismen zugreifen zu können, muss eine Servlet-Verbindung zwischen diesen definiert werden.

5.2.2.2 Struktur der Client-Applikation

Auch bei der Entwicklung der Client-Anwendung wurde diese Struktur weitestgehend umgesetzt. Bedingt durch die Vorgaben eines XCode-Projekts ergibt sich hierbei folgende Projektstruktur:

- Classes

  - **Model** – Systemweit genutzte Entity-Klassen
  - **Context** – Kapselung der Klassen zur Situationsklassifizierung
  - **XML** – Modul zum parsen und schreiben von XML-Daten sowie der Implementierung der SOAP-Unterstützung
  - **Utilities** – Hilfsklassen
  - **UI** – Graphische Darstellungskomponenten und Controller
- Other Sources
- Resources – Konfigurationsdateien, Bilder, Audio- und Nib-Files
- Frameworks
- Products

Hierbei handelt es sich allerdings nur um eine logische Strukturierung des Projekts. Bei Betrachtung der Ordnerstruktur befinden sich die Ressourcen im Root-Verzeichnis und die unter Classes angegebenen Module im gleichnamigen Ordner Classes.

5.2.3 Realisierung der Anwendungslogik

5.2.3.1 Anwendungslogik in der Verwaltungs-Anwendung

Gemäß dem Prinzip *Inversion of Control (IoC)*, werden in Spring alle Domain-Objekte – sogenannte *Spring Beans* – in XML-Dateien konfiguriert indem abhängige Objekte zur Laufzeit zugewiesen werden. Dieses Prinzip ist auch bekannt unter dem Stichwort *Dependency Injection* und stellt eine Lösung zum Aufbau von Objektnetzen dar. Das Objekt nimmt dadurch eine passive Rolle ein: ihm werden die abhängigen Objekte von außen zugewiesen, statt sie z.B. bei einer Factory aktiv zu erzeugen. Besonders flexibel ist dieser Ansatz, wenn die Objekte nur noch von Interfaces abhängen. Dann können beliebige Objekte verwendet werden, um das Interface zu implementieren. Deshalb wird dieses Vorgehen bei der Implementierung der Anwendungslogik verfolgt, um von der Kombination der Vorteile von Spring und Cocoon zu profitieren.

Neben den Geschäftsmethoden hält das Projekt auch die Entity-Klassen, welche das O/R-Mapping auf die angebundene Datenbank abbilden sowie Konfigurationsdateien. Um Hibernate in das System zu integrieren muss zuvor die Konfiguration angepasst und eine entsprechende Datenquelle, in Form einer Spring Bean vom Typ `org.springframework.jdbc.datasource.DriverManagerDataSource`, definiert werden.

Zudem müssen Properties definiert werden, die für die Identifizierung an der MySQL-Datenbank benötigt werden. Zusätzlich muss eine zweite Spring Bean definiert werden: die `AnnotationSessionFactoryBean`. Diese ist verantwortlich für die Erstellung der Session und den Zugriff auf diese um die Realisierung von Transaktionen und den Zugriff auf die Datenbank zu gewährleisten. Außerdem müssen die Entitäts-Klassen im Persistenzkontext bekannt gemacht werden, dies erfolgt im Persistenz-Deployment-Deskriptor **persistence-application-context.xml** durch die Zuordnung der Entity-Klassen zur soeben definierten Persistenz-Einheit. Darüber hinaus wird die soeben beschriebene Bean `DriverManagerDataSource` injiziert, um diese als Datenquelle festzulegen. Weitere Properties können überdies genutzt werden, um Hibernate zu konfigurieren. Die entsprechenden Einstellungen sollen jedoch hier nicht aufgezählt werden, sondern können im Projekt in der Konfigurationsdatei eingesehen werden.

Abbildung der Entity-Klassen Seit Java 1.5 und Hibernate ab Version 3 kann man durch die Verwendung von Annotations in den Entity-Klassen vollständig auf die Nutzung von hbm.xml Dateien verzichten, in denen das Hibernate Mapping definiert wurde. Dieser Verzicht macht das Mapping insgesamt übersichtlicher und bündelt alle relevanten Mapping-Informationen in den jeweiligen Entity-Klassen. Es muss demzufolge nur noch eine Datei angepasst werden, wohingegen zuvor die Synchronität der Klassen und ihrer hbm-Dateien gewährleistet werden musste. Zudem basiert das Prinzip der Annotations auf dem *Java Persistence API (JPA)* Standard wie er auch von EJB3 benutzt wird.

Bei den Entity-Klassen, die im Paket `de.sportix.entities.pojo` implementiert sind, handelt es sich dabei allerdings nicht mehr um reine *Plain Old Java Objects (POJOs)*, da sie nun eine Abhängigkeit zu JPA besitzen. Außerdem sind im Paket de.sportix.entities.dao entsprechende DAO-Klassen implementiert, die den Zugriff auf die Entitäten kapseln. Damit die Attribute der Klassen durch Hibernate genutzt werden können, müssen entsprechende Getter- und Setter-Methoden verfügbar sein. Neben der Annotation zur Deklaration der Klasse als Entität können sich weitere Annotationen auf die Beziehungen und Attribute der Klasse beziehen, indem sie vor dem jeweiligen Getter-Methoden spezifiziert werden.

Um einen Einblick in die spezifische Abbildung der relationalen Datenstruktur auf die Entity-Klassen zu geben, wird folgend ein Beispiel für die Klasse der Entity `User` angeführt.

Implementierung der User-Entity-Klassen Um die Vererbungshierarchie der Relation `User` abzubilden, fällt die Wahl auf die Strategie `InheritanceType.JOINED`, welche

sowohl die Basis-Entität als auch die Unterklassen `Administrator`, `Author` und `Player` in je einer eigenen Relation abbildet und die Daten mit einer 1:1 Beziehung verbindet.

Listing 5.2 zeigt einen Abriss der Basis-Entität der Klasse `User`. Neben der Annotation zur Deklaration als Entity-Klasse und dem Überschreiben des Namens der Relation in Kleinbuchstaben wird in Zeile 3 die gewählte Mapping-Strategie der Vererbung bekannt gegeben.

Listing 5.2: Ausschnitt der Datei UserImpl.java

```
1 @Entity
2 @Table(name = "user")
3 @Inheritance(strategy = InheritanceType.JOINED)
4 public class UserImpl implements User {
5   @Id
6   @GeneratedValue(strategy = GenerationType.IDENTITY)
7   @Column(nullable = false, unique = true)
8   public int getId() {
9     return _id;
10  }
11  /* more attributes, constructors, getters and setters */
12 }
```

Jedes persistente Objekte muss die Deklaration mindestens eines Primärschlüssel-Attributes besitzen. Diese Property wird mit `Id` festgelegt. Um eine automatische Schlüsselgenerierung umzusetzen, wurde folgend die Strategie `GenerationType.IDENTITY` gewählt, welche auf der Auto-Increment-Funktion des RDBMS, in diesem Fall MySQL, aufbaut. Des Weiteren sind in der Spaltendefinition die Attribute `nullable=false` gesetzt, um bereits bei Übergabe des Wertes an die DAO-Klasse zur persistenten Speicherung keine NULL-Werte in der identifizierenden Primärschlüsselspalte zuzulassen.

Listing 5.3 zeigt einen Auszug aus der Kind-Klasse `Player`, welche die Basis-Entität `User` implementiert. Desweiteren enthält der `Player` unter anderem eine 1:1 Referenz zu der Entität, die seinen `GameCharacter` repräsentiert.

Listing 5.3: Ausschnitt der Sub-Klasse PlayerImpl.java

```
1 @Entity
2 @Table(name = "player")
3 public class PlayerImpl extends UserImpl implements Player {
4   @OneToOne(targetEntity = GameCharacterImpl.class)
5   public GameCharacter getGameCharacter() {
6     return _character;
7   }
8   /* more attributes, constructors, getters and setters */
9 }
```

Implementierung der DAO-Klassen Durch die Verwendung von DAOs kann die Architektur sehr viel flexibler gestaltet werden (vgl. Abschnitt 4.2.3, S. 72), weil die Per-

sistenzschicht ausgetauscht werden kann. Zudem können sowohl die Geschäfts- und Persistenzlogik unabhängig voneinander weiterentwickelt werden. Ein wichtiger Vorteil ist auch, dass man bei Nutzung einer DAO-Schicht ggf. die Technologie-Entscheidung revidieren kann. So kann man beispielsweise in performance-sensitiven Bereichen der Anwendung eine API wie JDBC nutzen, die direkten Zugriff auf die Features der Datenbank bietet oder einen O/R-Mapper wie Hibernate nutzen.

Sollte sich im Nachhinein herausstellen, dass die Nutzung von Hibernate für die Umsetzung eines Spiels nicht praktikabel ist, so ließen sich entsprechende DAO-Klasse gegen die entwickelten Interfaces implementieren. Deshalb wird für jedes der entwickelten Entity-Objekte ein DAO-Interface entwickelt. Konkret werden diese durch `HibernateDAO`-Implementierungen in der Anwendung realisiert. Auf das Beispiel der User bezogen wird dementsprechend ein Interface `UserDAO` und die dazugehörige Implementierung `HibernateUserDAOImpl` bereitgestellt. Diese enthält die Geschäftslogik zum Laden, Speichern und Ändern der User.

Bestimmung passender Gegner Die Bestimmung passender Gegener durch die Verwaltungsanwendung wird durch `org.apache.cocoon.components.cron.CocoonPipelineCronJob`[1] von Cocoon gesteuert. In regelmäßigen Abständen werden die momentan aktiven Spieler ermittelt. Dies wird ermöglicht, indem bei jeder Anfrage an den Server ein `timestamp` gesetzt wird. Sendet der Spieler innerhalb von 30 Minuten keine neue Anfrage, so gilt dieser `timestamp` als ungültig und der Spieler infolgedessen als inaktiv. Neben der Aktivität ist auch von Interesse, ob für diesen Spieler bereits ein Wettbewerb vorgemerkt wurde. Falls dies der Fall ist, kann auch dieser Spieler nicht ausgewählt werden.

Zufällig wird einer der als aktiv klassifizierten Spieler ausgewählt. Die statistischen Daten seines Profils werden ausgewertet und kategorisiert, um einen passenden Gegner aus den verbleibenden aktiven Spielern auszuwählen.

Da eine sequentielle Überprüfung der Spieler zur Bestimmung eines optimalen Gegners bei einem hinreichend genutzten Spiel nicht akzeptabel ist, habe ich mich für die Umsetzung des $\frac{1}{e}$-Gesetz der besten Wahl [Bru04] entschieden. Die Strategie des Algorithmus besteht darin, das erste Drittel der Profile der Spieler in einer Analysephase auszuwerten und den besten bereits vorzumerken. Daraufhin werden die verbliebenen Profile sequentiell betrachtet und mit dem des zuvor vorgemerkten Spielers verglichen. Findet sich eines, dass eine bessere Übereinstimmung liefert, so wird dieser Spieler als Gegner angenommen.

[1] `http://cocoon.apache.org/2.1/apidocs/org/apache/cocoon/components/cron/CocoonPipelineCronJob.html`

Implementierung der Web Services Die Implementierung der Web Services wurde mit Hilfe von *Apache Axis* vorgenommen. Auch die Integration von Axis war serverseitig mit erheblichen Schwierigkeiten verbunden. Es musste ein `AxisRPCReader` entwickelt werden, der es ermöglicht SOAP-Requests innerhalb von Cocoon zu verarbeiten und entsprechende Responses zurückzugeben. Dies wird realisiert, indem die Funktionalität von Axis gekapselt wird. Der Reader stellt somit die Schnittstelle zu Axis dar.

Um nun auf entsprechende SOAP-Request via HTTP-Post reagieren zu können, muss der AxisRPCReader in der Sitemap der Anwendung bekannt gemacht werden. Daraufhin kann mit Hilfe eines `WildcardURIMatcher`[2] ein Teil einer URI angegeben werden, die den Zugriff auf die durch das System angebotenen Web Services ermöglichen soll. Diese können, nachdem die Validität der Integration sichergestellt wurde, über so genannte WSDD-Dateien definiert und in Axis eingebunden werden.

Ruft nun ein unregistrierter Spieler die Funktion `registerPlayer` auf, wird der SOAP-Request in der Pipeline entsprechend erkannt und an den `AxisRPCReader` weitergeleitet. Dieser wertet die Anfrage entsprechend aus und sorgt dafür, dass die entsprechenden Methoden der Web Services aufgerufen werden. In diesem Fall wird serverseitig überprüft, ob es den durch den Spieler angegebenen Namen bereits gibt, wenn ja, gibt er eine entsprechende Fehlermeldung zuruck, die von der Spiele-Anwendung entsprechend ausgewertet wird.

Bei Erfolg gibt der Server eine entsprechende `userId` zurück. Diese ermöglicht dem Spieler, seinen Status durch `initPlaying` am Server auf aktiv zu setzen. Sobald der Spieler das erste mal seinen aktuellen Status überträgt – seine Position und Aktivität – gibt der Server die für den Spieler relevanten `GameQuest`s und `StoryElement`s zurück.

5.2.3.2 Anwendungslogik der Spiele-Anwendung

Bei der Implementierung der Anwendungslogik spielt, neben der Kommunikation mit der Verwaltungsanwendung, die Verarbeitung der Sensordaten zur Bestimmung der aktuellen Aktivität des Spielers und die Umsetzung der Spielelogik eine große Rolle.

Die Implementierung dieser Module gestaltete sich unerwartet schwierig. Aufgrund der durch Apple verhängten NDA, die nach Veröffentlichung des iPhone SDKs bis zum 20. Oktober 2008 aufrecht erhalten wurde, gab es für das System des iPhones keine Referenz-Implementierungen, Code-Beispiele oder Frameworks außerhalb der durch Apple zur Verfügung gestellten APIs. Deshalb mussten viele grundlegende Funktionen implementiert werden, um die Funktionalität des Prototypen umsetzen zu können. Nachfolgend

[2] `http://cocoon.apache.org/2.1/userdocs/default/wildcarduri-matcher.html`

soll nun auf einzelne Aspekte und ausgewählte Algorithmen dieser Implementierungen eingegangen werden.

Kommunikation mit der Verwaltungs-Anwendung Die Kommunikation mit der Verwaltungs-Anwendung soll laut dem Entwurf der Anwendung über Web Services unter Nutzung von SOAP realisiert werden. Das Parsen von XML-Daten gestaltet sich in Objective-C zunächst sehr einfach: als fester Bestandteil des iPhone SDKs stellt Apple den `NSXMLParser`[3] zur Verfügung. Dieser erlaubt das event-basierte Einlesen von XML-Strukturen. Der Zugriff auf die einzelnen Elemente und Attribute der Daten kann so sehr einfach realisiert werden. Allerdings wird auf diese Weise nur das Lesen von XML-Dateien erlaubt. Das SDK sieht keine Möglichkeit vor, XML-Strukturen zu schreiben.

Für die Implementierung bedeutet dies, dass ein eigener XML-Serializer entwickelt werden muss, der zusätzlich das Schreiben von XML-Strukturen ermöglicht. Außerdem wird ein spezifischer XMl-Parser zur Verfügung gestellt, der die Funktionalität des `NSXMLParser`s implementiert und erweitert. Überdies wurde im zweiten Schritt eine Erweiterung des entwickelten `XmlSerializer` und `XmlParser` vorgenommen, um direkt Soap-Requests generieren und deren Responses auswerten zu können. Auch hier gab es keine vorhandenen Lösungen für das iPhone.

Dieser Arbeitsaufwand bei der Implementierung der Kommunikationsschnittstelle seitens des Clients kam sehr unerwartet, jedoch konnte auf diese Weise eine Lösung implementiert werden, welche sich gut in das System integriert und flexibel erweitert werden kann.

Durch entsprechende Tests bei unterschiedlichen Bedingungen musste allerdings festgestellt werden, dass sich die Kommunikationsfähgikeit des mobilen Gerätes sehr oft ändert. Die Verfügbarkeit eines WLAN-Netzes, ohne die Möglichkeit auf das Internet zuzugreifen, eine Verbindung über EDGE oder UMTS oder auch das Fehlen einer entsprechenden Möglichkeit zur Kommunikation prägen die Ausführung der Anwendung. Deshalb konnte vor dem Senden eines Requests nicht sichergestellt werden, in welcher Zeit und mit welchem Ergebnis er abgearbeitet werden würde. Es sollte deshalb sichergestellt werden können, dass keine Daten übertragen werden, wenn der Server nicht erreichbar ist oder keine Verbindung hergestellt werden kann. Um dies zu realisieren wurde eine als *Singleton* implementierte Klasse `Reachability` realisiert, deren Aufgabe es den Status der Konnektivität zu überwachen und deren Änderungen über an das *NotificationCenter* der Applikation zu kommunizieren. Dem Singleton-Muster folgend wird diese Klasse nur einmalig instanziiert. Der Status der Verbindung kann daher auf unterschiedlichen Ebe-

[3] `http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSXMLParser_Class/Reference/Reference.html`

nen der Anwendungslogik gleichermaßen genutzt werden: sowohl bei der Implementierung virtueller Sensoren, die durch die Kommunikation mit Web Services umgesetzt werden als auch bei der Kommunikation mit der Verwaltungs-Anwendung.

Jeder iPhone Applikation ist eine Instanz des `NSNotificationCenters`[4] zugeordnet. Alle Objekte der Applikation können sich an diesem als *Observer* registrieren, um die für sie relevanten `NSNotifications` zu erhalten. Wenn sich nun der Status ändert und momentan nicht auf das Internet zugegriffen werden kann, kommuniziert die Klasse `Reachability` eine entsprechende Notification mit dem Namen `kNetworkReachabilityChangedNotification`:

```
// Post a notification to notify the client that the network reachability changed.
[[NSNotificationCenter defaultCenter]
    postNotificationName:@"kNetworkReachabilityChangedNotification"
                  object:nil];
```

Damit der `SportixAppController` nun über diese Notification informiert werden kann, registriert er sich für entsprechende Notifications am `NSNotificationCenter` der Applikation.

```
// Register for notification called "kNetworkReachabilityChangedNotification"
[[NSNotificationCenter defaultCenter]
    addObserver:self
       selector:@selector(reachabilityChanged:)
           name:@"kNetworkReachabilityChangedNotification"
         object:nil];
```

Bei jeder Änderung wird infolgedessen am `SportixAppController` die Methode `reachabilityChanged:` aufgerufen, die den aktuellen Status von der Klasse `Reachability` erfragt um die entsprechende Geschäftslogik auszuführen. Beispielsweise die Kommunikation mit dem Server.

Diese wird in der Spiele-Anwendung in einem eigenen Thread verwaltet. Dieser überprüft in regelmäßigen Abständen die Aktualität der Daten. Sind die von einer bestimmten Position abhängenden Daten bereits so alt, dass sich diese nicht mehr in einem Umkreis von $500m$ befinden, so muss dafür gesorgt werden, dass diese entsprechend aktualisiert werden. Wenn der Status der Klasse `Reachability` angibt, dass eine Internetverbindung vorhanden ist, kann dieser Schritt sofort ausgeführt werden. Andererseits wird diese Aufgabe in eine Queue geschrieben, um später ausgeführt zu werden.

Die Implementierung der Klasse `Reachability` kann außerdem als ein *virtueller Sensor* betrachtet werden: es ist möglich, die aktuelle Situation des Spielers anhand der Konnektivität zu kategorisieren. Sind WLANs in der Nähe, kann beispielsweise darauf geschlossen

[4] `http://developer.apple.com/documentation/Cocoa/Reference/Foundation/Classes/NSNotificationCenter_Class/Reference/Reference.html`

werden, dass er sich in der Nähe oder innerhalb eines Gebäuden befindet. Auswertungen dieser Art wurden bei der Implementierung des Prototypen allerdings nicht berücksichtigt. Der Fokus lag hier auf die Nutzung physikalischer Sensoren.

Verarbeitung der Sensordaten Mit Hilfe des iPhone OS ist es möglich auf unterschiedliche Hardware Features zuzugreifen. Zu diesen gehören die Beschleunigungssensoren, die Kamera und das GPS-Modul: durch die Nutzung der Beschleunigungssensoren kann die Geschwindigkeit über einen bestimmten Zeitraum entlang der drei Hauptachsen (x, y und z) des iPhones gemessen werden; die Kamera ermöglicht beispielsweise eine Auswertung der aktuellen Umgebung anhand aufgenommener Fotos und das GPS-Modul erlaubt die Bestimmung der Position des Nutzers.

Der Zugriff auf diese Funktionen wird durch das iPhone SDK durch entsprechende Frameworks und Klassen ermöglicht. Die Implementierung des Prototypen nutzt zur Bestimmung der Aktivität des Spielers die Beschleunigungs-Sensoren und das Location Framework. Der Zugriff auf die Sensoren wird entsprechend der Definition des Entwurfs gekapselt in den Klassen `AccelerationSensorData` und `LocationSensorData`. Die einheitliche Schnittstelle der Objekte ermöglicht so eine einfache Erweiterung der Applikation, falls weitere Sensoren hinzugefügt werden sollen.

Das Auslesen der Beschleunigungs-Sensoren gestaltet sich sehr einfach: jeder Applikation ist bereits ein `UIAcceleration` Objekt zugeordnet, dessen Instanz genutzt werden kann, um das gewünschte Intervall der Messungen zu definieren und das *Delegate* festzulegen, dem die gemessen Sensordaten übermittelt werden. Sobald das Delegate Objekt definiert wurde, wird die Übermittlung der Daten gestartet. In Listing 5.4 werden die grundlegenden Schritte zur Konfiguration der Beschleunigungssensoren aufgeführt.

Listing 5.4: Konfiguration der Beschleunigungssensoren

```
1 #define kAccelerometerFrequency        50 //Hz
2 -(void)configureAccelerometer {
3     UIAccelerometer*  theAccelerometer = [UIAccelerometer sharedAccelerometer];
4     theAccelerometer.updateInterval = 1 / kAccelerometerFrequency;
5
6     theAccelerometer.delegate = self;
7     // Delegate events begin immediately.
8 }
```

In regelmäßigen Abständen wird am Delegate die Methode `accelerometer:didAccelerate:` aufgerufen. Diese Methode wird genutzt, um die gemessenen Daten auszuwerten. Je schneller sich ein Mensch fortbewegt, umso höher ist auch seine Schrittfrequenz. Und desto größer ist demzufolge die Intensität der Beschleunigungswerte. Zur Kategorisierung dieser Werte werden Untersuchungen zu

Grunde gelegt, dass kein signifikanter Beschleunigungswert über 15 Hz liegt. Dem Nyquist-Shannon-Abtasttheorem zufolge muss demnach eine Abtastung von mindestens 30 Hz erfolgen. Die Geschwindigkeit muss allerdings nicht über die Beschleunigungssensoren bestimmt werden, da dieses Feature aus den Positionsdaten errechnet wird. Für die Umsetzung des Prototypen gehe ich davon aus, dass der Spieler wenn er sich innerhalb eines Gebäudes befindet, nicht joggen wird. Daher kann von der Verfügbarkeit der Positionsdaten ausgegangen werden.

Die Beschleunigungssensoren werden stattdessen für die Berechnung der Intensität der Betätigung genutzt. Dies geschieht, indem über ein Fenster von 128 Werten[5] der Mittelwert, die Standardabweichung und die Varianz der Daten berechnet wird.

Ähnlich einfach gestaltet sich die Nutzung des Core Location Frameworks zur Bestimmung der Position des Spielers: um die Position zu erhalten, muss ein Objekt vom Typ `CLLocationManager` erzeugt und konfiguriert werden. Auch hier muss zuvor ein entsprechender Delegate angegeben werden. Da die Nutzung dieses Frameworks die Batterielaufzeit stark beeinflusst, muss explizit die Methode `startUpdateLocation` aufgerufen werden, um kontinuierliche Positionsdaten zu erhalten. Sobald neue Daten verfügbar sind, informiert der LocationManager das Delegate.

Die Nutzung von GPS-Daten kann nur entsprechende Ergebnisse erzielen, wenn die Berechnungen auf den Messdaten von mindestens vier Satellien beruhen. Als weiteres Auswahlkriterium kann der HDOP Wert betrachtet werden. Dieser wird vom Empfänger im NMEA Protokoll übermittelt und bedeutet *Horizontal Dilution of Precision* [2]. Er drückt in Zahlen die geometrische Verteilung der sichtbaren Satelliten aus. Je kleiner der HDOP, um so genauer ist die errechnete horizontale Position, da die Satelliten gut verteilt sind. Die API des iPhone SDK erlaubt es dem Entwickler jedoch nicht, auf die entsprechenden Daten zuzugreifen. Stattdessen hat man die Möglichkeit das Attribut `CLLocationAccuracy` auszuwerten. Es enthält sowohl die horizontale als auch vertikale Genauigkeit der ermittelten Position in Meter.

Bei der Nutzung des *Core Location Framework* des iPhone SDKs gilt es, einige Dinge zu beachten. Da die Positionsdaten durch unterschiedliche Berechnungen und Technologien zur Verfügung gestellt werden, können diese erheblich in der Genauigkeit und der Qualität variieren. Für entsprechende Berechnungen – beispielsweise der Geschwindigkeit des Spielers – ist es daher von Nöten, diese Daten zu analysieren und unbrauchbare Daten zu verwerfen. Die Entscheidung, ob es sich um brauchbare Daten handelt oder nicht, lässt sich allerdings nicht pauschalisieren. Befindet sich der Nutzer in Bereichen, in denen kein

[5] Bei einer Frequenz von 50 Hz umfassen die Berechnungen demnach ein Zeitintervall von ca. 2,5 Sekunden.

GPS verfügbar ist, ist es durchaus sinnvoll, Daten mit unzureichenden Genauigkeiten zu nutzen. Ist allerdings ein ausreichend gutes Signal vorhanden, können ungenaue Daten verworfen werden, um das Ergebnis nicht unnötig zu verfälschen.

Die ersten Daten nach Aktivierung des Core Location Frameworks werden in der Regel über das im iPhone integrierte WPS bestimmt. Diese Daten sind sehr hilfreich, wenn sich der Spieler im Inneren eines Gebäudes befindet. Allerdings können Genauigkeiten von über $1300m$ zustande kommen (vgl. GPS-Auswertungen im Anhang und in Abschnitt 6). Diese Daten lassen sich durch Analyse der `horizontalAccuracy` und `verticalAccuracy` der `CLLocation`-Objekte identifizieren. Ein Ansatzpunkt stellt hierbei beispielsweise die nicht vorhandene Höhe dar. Diese wird durch eine vertikale Accuracy von -1 gekennzeichnet. Auch eine Bestimmung der Position durch Mobilfunkortung wird unter Umständen durchgeführt. Auf die Verwendung dieses Verfahrens kann man allerdings nicht explizit durch Analyse der durch die API zur Verfügung gestellten Werte schließen. Der einzige Ansatzpunkt besteht hier wiederum nur in der Überprüfung der horizontalen Genauigkeit.

Neben der Attributierung der Genauigkeit spezifischer Sensordaten lässt sich auch das Alter der Daten in die Auswertung und Analyse mit einbeziehen. Jeder der durch den `CLLocationManager` bereitgestellten Position des Nutzers ist ein `timestamp` zugeordnet. Dieser gibt allerdings Aufschluss über die Zeit zu der die Positionsanfrage gestellt wurde, nicht über den Zeitpunkt, wann diese Ergebnisse liefert. Da, wie bereits erläutert, unterschiedliche Technologien zur Bestimmung der Position genutzt werden, kann es deshalb passieren, dass Daten in scheinbar zufälliger Reihenfolge geliefert werden. Um die Aktualität der Sensoraten gewährleisten zu können, ist daher die Überprüfung des Alters von hoher Relevanz.

Auf Grundlage der gefilterten Sensordaten durch Auswertung der Kontextattribute Alter und Genauigkeit lassen sich bereits einige der benötigten Features bestimmen. Allerdings ist festzustellen, dass die GPS-Daten trotz dieser ersten Filterung, einer recht hohen Ungenauigkeit unterliegen. Deshalb wird die Vorverarbeitung durch einen weiteren Aspekt ergänzt: der Glättung der Positionsdaten.

Glättung der GPS-Positionen Das GPS-Signal springt häufig hin und her, so dass es einer Glättung bedarf, bevor anhand dessen die Geschwindigkeit des Spielers berechnet wird. Da dieser Wert ein entscheidendes Kriterium zur Bestimmung der Aktivität ist, muss eine entsprechende Genauigkeit gewährleistet werden können. Quentin Ladetto benutzt in [LGM01] ein Schwerpunktverfahren, um die GPS-Positionen zu glätten. Dazu wird der Schwerpunkt von n Positionen berechnet. Der Schwerpunkt S einer Menge von Punkten P ist das Mittel ihrer Addition.

$$S = \frac{1}{n} \sum_{i=1}^{n} P_i \tag{5.1}$$

Nach Ladetto sind bereits drei Punkte ausreichend, um eine gute Glättung zu erreichen. Der Schwerpunkt dieser Punkte wird zur Berechnung der aktuellen Geschwindigkeit benutzt. Ladetto bezeichnet dieses Verfahren als eine Art Tiefpassfilter für GPS-Positionen.

Bei diesem Schwerpunktfilter wird jede Position als gleich schwer betrachtet. Je genauer jedoch der angegebene Werte der horizontalen Genauigkeit, desto genauer ist ebenfalls die Position. Nach meinen Überlegungen und Tests lässt sich die Glättung verbessern, wenn nicht alle Positionen mit den gleichen Wahrscheinlichkeiten in die Berechnung des Schwerpunkts eingehen, sondern das Gewicht w einer jeden Position P_i abgeleitet von der Güte des Wertes der `horizontalAccuracy` h_i eingeht.

$$\begin{aligned} w_i &= \frac{1}{h_i} \\ w &= \sum_{i=1}^{n} w_i \\ S &= \frac{1}{w} \sum_{i=1}^{n} w_i P_i \end{aligned} \tag{5.2}$$

Besonders bei Positionen, die im Vergleich mit den anderen betrachteten Positionen einen großen Unterschied in der horizontalen Genauigkeit aufweisen, ist die Glättung sehr effektiv. Das GPS-Signal wird sehr gut geglättet und kann zu weiteren Berechnungen verwendet werden.

Geodätische Berechnungen Wie in Abschnitt 2.1.1.2 bereits erörtert wurde, kann man die Gestalt der Erde als Kugel betrachten wenn man für Berechnungen keine hohen Genauigkeitsansprüche stellt. Auf diese Weise kann man den Abstand zweier Punkte mit den Verfahren der Sphärischen Trigonometrie bestimmen: hat der erste Ort A die Koordinaten (λ_A, ϕ_A) und der zweite Ort B die Koordinaten (λ_B, ϕ_B) und liegen beide auf der Kugeloberfläche, so liefert der sphärische Cosinussatz den Winkel ψ zwischen den beiden Orten.

$$cos\psi = sin\phi_A \cdot sin\phi_B + cos\phi_A \cdot cos\phi_B \cdot cos(\lambda_A - \lambda_B) \tag{5.3}$$

Mit dem Erdradius $R_E = 6371km$ erhält man daraus die Gleichung für die Berechnung der Distanz s zwischen A und B:

$$s = \frac{R_E \pi \psi}{180} \tag{5.4}$$

Klassifizierung der Aktivität Vor der Implementierung des Moduls zur Identifizierung der Aktivität des Spielers war es notwendig, Referenzdaten zu sammeln und diese auszuwerten und zu analysieren. Da sich die Erkennung allerdings im ersten Schritt der Implementierung hauptsächlich mit geradlinigen Bewegungen, wie Walking, Joggen, Fahrrad fahren oder Auto fahren befassen sollte, wurde das Hauptaugenmerk auf die Auswertung und Optimierung der Features der LocationSensoren gelegt. Wenn gute GPS-Daten zur Verfügung stehen, kann auch eine hinreichend gute Geschwindigkeit für den aktuellen Zeitpunkt ermittelt werden. Die Auswertungen zeigten allerdings, dass diese extremen Schwankungen unterlagen. Deshalb müssen zusätzliche Filter entwickelt werden, die die Plausibilität der Daten überprüfen. Zusätzlich wurde das Schwerpunktverfahren eingeführt, was zusätzliche Schwankungen der Rohdaten eliminieren konnte. Letztendlich habe ich mich aber dafür entschieden, die Werte stets über einen gewissen Zeitraum zu betrachten. Dieses Verfahren, wird sowohl bei den LocationSensoren als auch den Beschleunigungssensoren umgesetzt. Die daraus hervorgehenden Daten bilden eine gute Basis für die Bestimmung der Aktivität.

Umsetzung der Spielelogik Um die für den Spielfortschritt benötigten Aktivitätspunkte zu berechnen, wurde eine Berechnung implementiert, die in Anlehnung an das *Metabolische Äquivalent* entwickelt wurde. Das Metabolische Äquivalent wird verwendet, um den Energieverbrauch verschiedener Aktivitäten miteinander zu vergleichen (vgl. [AHW+00]). Es gibt die Leistung von Aktivitäten als ein Vielfaches des Ruheumsatzes an. Da sich der Energieumsatz in der Praxis allerdings von Spieler zu Spieler unterscheidet, sich die Bestimmung der zu vergebenen Punktzahl jedoch nur auf die Intensität der Aktivität beschränken soll, wird dieses Prinzip an den Anforderungen entsprechend angepasst: die Betrachtung individueller Merkmale einer Person darf demzufolge keine Rolle spielen.

Durch Analysen unterschiedlicher Tabellen die das Metabolische Äquivalent unterschiedlicher Aktivitäten auf Grundlage statistischer Messungen definieren (vgl. [1, 9]) habe ich für die Implementierung des Prototypen die in Tabelle 5.1 aufgeführten Werte für unterschiedliche Aktivitäten angenommen.

Wenn die Bedingung und die Sportart erfüllt ist, dann berechnen sich die Punkte nach der letzten Spalte, ansonsten wird der Basiswert benutzt.

Aktivität	Bedingung	Basiswert	Berechnung
Walking	true	1	speed
Joggen	speed > 7	7	speed
Fahrrad fahren	speed > 16	4	(speed - 4)/2

Tabelle 5.1: Übersicht der Berechnungsgrundlage der Aktivitätspunkte

5.2.4 Realisierung der Präsentationslogik

Die Benutzeroberfläche der Verwaltungs-Anwendung wurde mit Hilfe von *Cocoon Forms (CForms)* und dem *JXTemplate Generator* realisiert. Die der Spiele-Anwendung dahingegen vollständig durch die Nutzung des iPhone SDKs. Im Folgenden soll für beide Anwendungen beispielhaft erläutert werden, wie dies umgesetzt wurde.

5.2.4.1 Präsentationslogik der Verwaltungs-Anwendung

In Anwendungen, die nach dem Model-View-Controller Entwurfsmuster konzipiert sind, entfällt ein wesentlicher Anteil der Implementierungsarbeit üblicherweise auf die Verwaltung der Event-Listener, welche die Datenmodelle der Anwendungslogik mit den Darstellungs- und Bearbeitungskomponenten der Benutzeroberfläche synchronisieren.

Um dies zu vereinfachen, wurde bei der Umsetzung der Formulare zur Verwaltung der Entitäten auf das *Binding Framework* von Cocoon zurückgegriffen. Es ermöglicht Daten in JavaBeans und XML-Dateien durch XPath-Ausdrücke zu adressieren und Formularelementen zuzuweisen. Die Bindungen werden mit Hilfe von XML-Konfigurationsdateien erstellt. Der Name eines in der Benutzeroberfläche vorhandenen Textfeldes wird beispielsweise in dieser Datei durch einen JXPath-Ausdruck mit einem Attribut eines ausgewählten Referenz-Objekts, dass über das zentrale Datenmodell ermittelt wird, verknüpft. Das Framework überträgt den Inhalt des Objekts automatisch an das Textfeld. Umgekehrt werden durch diese Bindung auch Eingaben im Formularelement automatisiert in das Referenz-Objekt innerhalb des Datenmodells zurückgeschrieben.

```
<fb:context xmlns:fb="http://apache.org/cocoon/forms/1.0#binding" path="/">
  <!-- Attribute name of object is binded to form element called username -->
  <fb:value id="username" path="name" />
  <fb:value id="email" path="mail" />
  <fb:value id="password" path="password" />
</fb:context>
```

Im obenstehenden Beispiel ermöglicht das Binding Framework die Bindung des Attributs `name` der Klasse `User` an das in der Benutzeroberfläche mit id *username* bezeichnete

Formularelement. Eventuelle Änderungen am Inhalt des Formularelements wirken sich infolgedessen auch auf das aktuelle Referenz-Objekt aus.

5.2.4.2 Internationalisierung

Der Prozess der *Internationalisierung* schafft die Voraussetzungen zur *Lokalisierung* einer Anwendung – der Adaption an die kulturellen und sprachlichen Gegebenheiten des Nutzers. Hierbei handelt es sich um zwei komplementäre Ansätze: Die Internationalisierung kennzeichnet den Vorgang der Bereitstellung der notwendigen Infrastruktur, die benötigt wird, um lokalisierte Inhalte zu unterstützen. Die Lokalisierung dahingegen fügt der Applikation die individuellen Ressourcen der unterstützen Sprachen hinzu. Sie umfasst hierbei unter Anderem die Anpassung und Bereitstellung folgender Aspekte:

- Benutzeroberflächen müssen so gestaltet werden, dass sie übersetzten Text akzeptieren
- Icons und Grafiken müssen so gestaltet werden, dass sie kulturellen Anforderungen genügen
- Sounds müssen bei gesprochenen Inhalten in unterschiedlichen Versionen angeboten werden
- statische Texte müssen übersetzt werden
- dynamisch generierte Texte (auch Datum, Zeit und numerische Werte) müssen lokalisiert werden

Auf die Besonderheiten und die Umsetzung der Internationalisierung der Spiele- als auch der Verwaltungs-Anwendung soll deshalb nachfolgend eingegangen werden.

Internationalisierung der Verwaltungs-Anwendung Im Kontext von Cocoon erleichtert die Nutzung von XML und XSLT die Internationalisierung erheblich für den Entwickler, besonders in Anbetracht der Trennung von Inhalt, Logik und Präsentation. Dieser Ansatz der Internationalisierung von XML-Dokumenten basiert in Cocoon auf einem Transformator – `org.apache.cocoon.transformation.I18nTransformer`. Sprachrelevante Informationen, wie bspw. Beschriftungen von Aktionskomponenten, Listeneinträgen und Ausgabetexten, werden in XML Dictionarys, den so genannten Katalog-Dateien, für jede zu implementierende Sprache ausgelagert. Ähnlich dem Properties-Format, werden in einer XML-Struktur Schlüssel-/Wertpaare gehalten, die für die Übersetzungen im *I18nTransformer* genutzt werden können. Voraussetzung dessen ist, dass sie bei Definition

des Transformators in der Sitemap des Projekts bekannt gemacht werden. Listing 5.5 zeigt die Vorgehensweise bei der Bekanntmachung des I18nTransformers. Es können beliebig viele Katalog-Dateien angegeben werden, welche anschließend über deren `id` angesprochen werden können, um spezielle Übersetzungen vorzunehmen.

Listing 5.5: Bekanntmachen des `I18nTransformer`s

```
<map:transformer name="i18n" src="org.apache.cocoon.transformation.I18nTransformer"\>
  <catalogues default="mymessages">
    <catalogue id="mymessages" name="messages" location="i18n"/>
    <!-- more catalogue definitions -->
    <catalogue id="forms" name="messages"
      location="resource://org/apache/cocoon/forms/system/i18n"/>
  </catalogues>
  <cache-at-startup>true</cache-at-startup>
</map:transformer>
```

Die Datei `messages.xml` enthält die Daten in englischer Sprache (da dies in diesem Fall die definierte Standardsprache ist), die Datei `messages_de.xml` enthält die gleichen Informationen, jedoch in deutscher Sprache. Weitere Sprachdateien können analog durch Anfügen des Sprachkürzels gemäß ISO 639[6], ergänzt durch eine eventuelle Länder/Regionenkennung gemäß ISO 3166[7], erstellt werden.

Innerhalb einer Ausgabekomponente kann nun bspw. das `<i18n:text>` Tag genutzt werden, um die darin definierten Elemente durch den I18nTransformator zur Laufzeit ersetzen zu lassen. Die Transformation dessen muss allerdings in der Sitemap vor der Serialisierung der XML-Datei explizit aufgerufen werden: `<map:transform type=i18n`.

Um die Internationalisierung in der Spiele-Anwendung zu implementieren, werden String-Tables im XCode-Projekt angelegt. Diese folgen der Namensgebung `localizable.strings`. Dieser Datei können Lokalisierungen hinzugefügt werden. Analog zu dem oben beschriebenen Verfahren, werden auch hier Werte für Keys angegeben, um die Übersetzungen zu realisieren, z.B. `title=sportix;`. Um diesen angelegten Key zu nutzen, kann man folgende Funktion nutzen: `NSLocalizedString(@title,@x)`. Entsprechend der Sprache, die auf dem aktuellen System vorhanden ist, wird diese aus der Datei `localizable.strings` ausgewählt.

5.2.4.3 Validatoren

Um die Datenintegrität hinsichtlich der Eingaben des Nutzers gewährleisten zu können, werden übermittelte Werte der Eingabekomponenten entsprechend der zugeordneten *Validatoren* überprüft.

[6] `http://www.loc.gov/standards/iso639-2/`

[7] `http://www.iso.org/iso/country_codes.htm`

Cocoon-Standardvalidatoren Von Cocoon wird bereits eine große Anzahl an vordefinierten Standardvalidatoren zur Verfügung gestellt. Diese kann man in beliebiger Kombination und Anzahl den Widgets zuweisen. Falls die Validierung dessen fehlschlägt, wird eine Fehlermeldung an diesem gesetzt. Eines der wohl am häufigsten verwendeten Standardvalidatoren ist die Überprüfung auf das Vorhandensein einer Eingabe: Listing ... zeigt die Implementierung mit Setzen des required-Validators und den Inhalt einer individualisierten Fehlermeldung beim Fehlschlagen. Daneben gibt es beispielsweise Validatoren zur Überprüfung einer bestimmten Länge der Eingabe, der Überprüfung auf reguläre Ausdrücke oder auch speziellere Valdiatoren zur Überprüfung von E-Mail-Adressen oder Kreditkartennummern.

Eigene Validatoren Zur Validierung von Benutzereingaben, die eine weiterführende Überprüfung – auch unter Hinzunahme der Geschäftslogik – benötigen, ist die Implementierung eigener Validatoren möglich, die als Teil der Form-Definition (im `<fd:validation>` Element) oder der Instanz des Widgets zur Laufzeit zugeordnet werden. Beispielsweise wurde eine Validator-Klasse implementiert, die bei Änderung der E-Mail-Adresse eines Benutzerkontos das Vorhandensein in einem anderen überprüft, da dieser Eintrag in den Entity-Klassen als `unique` gesetzt ist und somit Verletzungen bereits in der Präsentationslogik abgefangen werden können. Um eigene Klassen nutzen zu können, muss die Schnittstelle `WidgetValidator` implementiert werden.

Um Validierung und damit auch die Datenintegrität auf dem Client gewährleisten zu können, ist die Nutzung des *Key-Value Codings* von enormer Relevanz. In Objective-C wird Key-Value Coding hauptsächlich durch das `NSKeyValueCoding` Protokoll, welches von der Klasse `NSObject` implementiert wird, umgesetzt. Ein großer Vorteil dessen besteht in der konsequenten Umsetzung einer API um die Werte von Attributen zu validieren. Die Infrastruktur der Validierung ermöglicht es einer Klasse spezifische Werte anzunehmen, alternative Werte zu erzeugen oder den Wert in Kombination mit einer entsprechenden Fehlermeldung abzuweisen. Um dies in den Modellklassen der Applikation umzusetzen, ist es vonnöten der Konvention `validate<Key>:error:` zu folgen, um für das Attribut `Key` eine entsprechende Methode zur Validierung zur Verfügung zu stellen. Ein Aufruf einer entsprechenden Methode wird in Listing 5.6 für das Attribut `name` der Klasse `Player` nachvollzogen. Diese Methode zum Überprüfen der Attributwerte kann entweder direkt, oder durch Aufruf der Methode `validateValue:forKey:error:` ausgeführt werden.

Listing 5.6: Aufruf der Validierung

```
1 NSString *newName;
2 NSError *outError;
3
4 newName = [[[NSString alloc] initWithString:@"freddy"] autorelease];
```

```
5
6 if ([person validateName:&newName error:&outError]) {
7   // set the value, which will retain the newName object
8   [person setValue:newName forKey:@"name"];
9 } else {
10  // inform the user that the value is invalid
11 }
```

Kapitel 6

Evaluation und Demonstration

In diesem Kapitel soll eine Auswertung des im Rahmen dieser Arbeit entwickelten positions- und situationsabhängigen Adventurespiels gegeben werden. Als Grundlage dessen soll im ersten Abschnitt die prinzipielle Herangehensweise und Machbarkeit bei der Bestimmung der Aktivität des Spielers ausgewertet werden. Um eine systematische Betrachtung der Ergebnisse zu gewährleisten, muss eine Vielzahl von Daten aus unterschiedlichen Bereichen untersucht werden – sowohl Aktivitäten, die durch den Prototypen explizit erkannt werden sollen als auch Daten, die andere Eigenschaften aufweisen. Auf diese Weise soll auf die Stärken und Schwächen der eingesetzten Verfahren eingegangen werden. Im Anschluss an diese vorwiegend statistischen und praktischen Untersuchungen wird die umgesetzte Funktionalität der Verwaltungs- als auch Spiele-Anwendung demonstriert.

6.1 Evaluation des Systems

Um die korrekte Funktionalität des Spiels zu gewährleisten muss zunächst die Verlässlichkeit der Aktivitätserkennung, im Folgenden als so genannte *Aktivitätserkennungsrate* bezeichnet, sicher gestellt werden.

Die Klassifizierung der Aktivität hängt jedoch von der Verfügbarkeit und Qualität unterschiedlicher Komponenten – den Features der Sensordaten – ab. Daher ist es sinnvoll, deren Validität bereits vor der Evaluierung des Gesamtergebnisses zu betrachten. Da sich die Daten der Beschleunigungssensoren in der Praxis allerdings schlecht bewerten lassen, sollen sich diese Vorbetrachtungen auf die Qualität des Features der Geschwindigkeit und der Positionsdaten beschränken.

Im Rahmen der Evaluation der Situationserkennung lässt sich zudem anhand der erzielten Ergebnisse die Leistungsfähigkeit des Prototypen bezüglich unterschiedlicher Katego-

rien bewerten. Hierbei sind insbesondere die folgende Fragestellungen von Interesse: wie verlässlich ist die Bestimmung der aktuellen Aktivität des Spielers? Dies impliziert sowohl eine korrekte Unterscheidung bekannter als auch unbekannter Aktivitäten.

Um Aussagen zu dieser Fragestellung treffen zu können, müssen für die Evaluation entsprechende Daten realer Aktivitäten zur Verfügung stehen. Daher wird nach den notwendigen Vorbetrachtungen zunächst auf die Zusammenstellung der Testdaten eingegangen, bevor die Ergebnisse der simulierten Aktivitätserkennung präsentiert und durch verschiedene Feldversuche validiert werden.

6.1.1 Qualität der Sensordaten

Da die Güte der Situationserkennung maßgeblich von der Qualität der Sensordaten abhängt, sollen diese im ersten Schritt anhand der zur Verfügung gestellten Positions-Daten validiert werden. Um eine Aussage über die Qualität und Genauigkeit treffen zu können, wurden zwei unterschiedliche Versuchsreihen durchgeführt.

Bei der ersten wurden die im Zeitraum von einer Stunde durch das Core Location Framework bereitgestellten GPS-Daten gespeichert ohne eine Änderung der Position während dieser Zeit vorzunehmen. Diese ermöglichen unterschiedliche Auswertungen: sowohl die Schwankung der Anzahl der zur Verfügung gestellten Positionen als auch die Schwankung der Position an sich. Infolgedessen lässt sich etwas über die Qualität der Berechnungen und der Zuverlässigkeit des Core Location Frameworks aussagen.

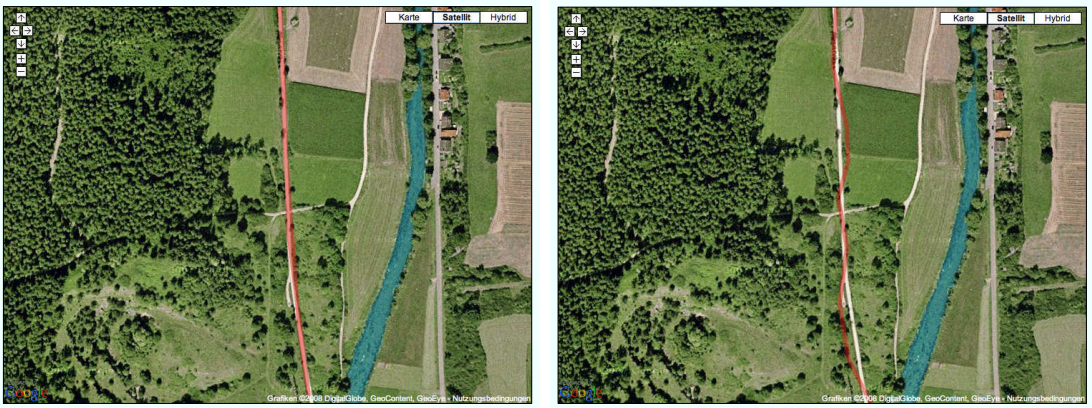
In Tabelle 6.1 ist zu erkennen, dass das Core Location Framework sehr zuverlässig über neue Positionen informiert. Im Schnitt werden in 5 Minuten 396,4 Positionen erzeugt. Das beträgt ungefähr eine Update-Frequenz von 1,32 Hz. Man kann einen vorsichtigen Schluss zulassen, dass diese Sensordaten sehr zuverlässig zur Verfügung stehen.

Die Auswertung der konkreten GPS-Daten hat zudem gezeigt, dass diese einer ständigen Änderung unterliegen. Auch wenn das Gerät nicht berührt oder bewegt wird. Infolgedessen wurde bei der Implementierung das Schwerpunktverfahren eingeführt. Dieses sorgt dafür, dass diese Effekte etwas geglättet werden.

Ziel der zweiten Versuchsreihe war es, die Daten eines in einem Smartphone integrierten GPS-Sensors mit denen eines zur Berechnung von GPS-Positionen ausgelegten Gerätes zu vergleichen. Die Messungen dieses Tests basieren auf dem Vergleich der gelieferten Daten des iPhones und eines *Forerunner 305* von Garmin [20]. Die Tests wurden auf unterschiedlichen Strecken und Streckenlängen innerhalb Deutschlands durchgeführt um einen Querschnitt möglicher Ergebnisse auszuwerten.

Zeitraum	Anzahl gelieferter Messwerte
14:29 - 14:34	402
14:34 - 14:39	397
14:39 - 14:44	401
14:44 - 14:49	402
14:49 - 14:54	403
14:54 - 14:59	403
14:59 - 15:04	399
15:04 - 15:09	390
15:09 - 15:14	387
15:14 - 15:19	392
15:19 - 15:24	389
15:24 - 15:29	393
15:29 - 15:34	395

Tabelle 6.1: Auflistung der Anzahl der erhaltenen Daten des CoreLocation Frameworks über einen Zeitraum von jeweils 5 Minuten.



(a) Mit dem Forerunner aufgenommene Strecke (b) Mit dem iPhone aufgenommene Strecke

Abbildung 6.1: Vergleich der Genauigkeit der Positionsdaten

Verfahren	Abweichung in Prozent
gemessene Strecke [km]	3,5
Maximale Geschwindigkeit [km/h]	75
Durchschnittsgeschwindigkeit [km/h]	3,5
Durchschnitts-Pace [min/km]	4,9

Tabelle 6.2: Übersicht statistischer Abweichungen gemessener Positionsdaten

Die Google Maps Visualisierung 6.1(a) zeigt die aufgenommenen Daten des Forerunner 305. Es ist zu erkennen, dass die Strecke im Gegensatz zu Abbildung 6.1(b) sehr viel glatter dargestellt wird. Insgesamt wirkt sich das allerdings kaum auf die Gesamtstrecke und Durchschnittswerte der Daten aus. Erstaunlich ist allerdings, dass die durch das iPhone berechnete Höhe sehr viel genauer als die des Garmin ist.

In Tabelle 6.2 werden statistische Werte der Abweichung angegeben. Es ist erkennbar, dass kaum Abweichungen bei den Durchschnittswerten festzustellen sind. Lediglich die Höchstgeschwindigkeit sollte bei den Untersuchungen und Klassifizierungen der Situation des Spielers nicht mit einbezogen werden, da dies die Daten verfälschen würde.

Weitere Versuche zeigten, dass sich bei höherer Geschwindigkeit, also beispielsweise beim Fahrrad fahren, die Genauigkeit des iPhones an die des Garmins annähert. Lediglich in Bereichen, wo WLANs verfügbar sind, kann es zu steigenden Ungenauigkeiten kommen. In der Regel liegen die durch WLAN bestimmten Positionen bei einer horizontalen Genauigkeit von mindestens $1300m$. Nach einigen Sekunden adaptiert sich dies und es kann eine Genauigkeit von ca. 120m erreicht werden. Diese Werte wirken jedoch störend, wenn die Berechnungen der Strecke bisher ausschließlich durch GPS-Daten erfolgte. Infolgedessen sollte bei solchen Betrachtungen ein zusätzlicher Filter genutzt werden, um die Herkunft der Daten zu klassifizieren.

6.1.2 Simulierte Tests zur Situationserkennung

6.1.2.1 Testdaten

Für die Testreihen wurden unterschiedliche Aktivitätsprofile durch eine eigens dafür entworfene Applikation aufgenommen. Diese Aktivitätsprofile setzen sich aus den in der Spiele-Anwendung benötigten Sensordaten zusammen: unterschiedliche Positions- und Beschleunigungs-Daten, die mit Hilfe des iPhones aufgenommen und entsprechend serialisiert wurden, um diese für die Situationserkennung nutzbar zu machen. Diese Profile

Profilname	Aktivität	Verfügbarkeit	
		Position	Beschleunigung
Stehen01	Stehen	-	x
Stehen02		x	-
Stehen03		x	x
Laufen01	Laufen	-	x
Laufen02		x	-
Laufen03		x	x
Joggen01	Joggen	-	x
Joggen02		x	-
Joggen03		x	x
Shaking01	Keine	-	x
Shaking02		x	-
Shaking03		x	x

Tabelle 6.3: Übersicht der Aktivitätsprofile für die Evaluierung

unterschieden sich in mehreren Aspekten:

- die konkrete Aktivität, die sie beschreiben: sowohl Aktivitäten, die explizit durch die Spiele-Anwendung erkannt werden sollen, als auch unerwartete Daten, wie das Schütteln des iPhones,
- die unterschiedliche Genauigkeit und Qualität der Daten sowie
- das Fehlen der Daten eines Sensors.

Insgesamt wurden für die Testreihen 12 unterschiedliche Aktivitätsprofile zusammengestellt. Diese bilden jeweils unterschiedliche Aspekte ab. Tabelle 6.3 zeigt eine Übersicht dieser Profile und gibt ein Zusammenfassung über deren spezifische Konfiguration. Es gibt demnach 3 Profile, denen keine Aktivität zugeordnet werden dürfte. Aber auch die fehlenden Sensordaten, dürften bei der Kategorisierung zu fehlerhaften Ergebnissen führen.

6.1.2.2 Testergebnisse

Entsprechend der im vorigen Abschnitt angegebenen Aktivitätsprofile wurden unterschiedliche Testdaten vorbereitet, anhand derer die Versuchsreihen durchgeführt wurden.

Die Ergebnisse dieser Versuchsreihen sind kategorisch unterteilt in die für das System bekannten Aktivitäten „Stehen“, „Laufen“ und „Rennen“.

Nachfolgend sind die Features der Bewegungssensoren dieser 3 unterschiedlichen Aktivitätsprofile dargestellt. Abbildung 6.2 visualisiert, dass der Spieler an einer Stelle steht und sich nicht bewegt. Alle Untersuchungen, die zu diesem Profil gemacht wurden, konnten richtig klassifiziert werden. Es ist klar zu erkennen, dass die einzelnen Bewegungssensoren kaum einer Änderung unterliegen. Sobald sich der Spieler jedoch etwas mehr bewegt, wird die Auswertung nicht mehr zu 100% erfüllt. Dann war es im Schnitt jeder 3. der nicht korrekt zugeordnet werden konnte. In der Grafik ist zu erkennen, dass sich die x- und y-Werte in der fast ausschließlich in der Nähe von 0 befinden, wohingegen z (aufgrund der Erdanziehungskraft) bei -1 liegt. Für dieses Profil wurde infolgedessen ein Muster erstellt. Wird dieses erkannt und zudem durch die Geschwindigkeit des Nutzers bestätigt, so kann definitiv bestätigt werden, dass der Spieler steht.

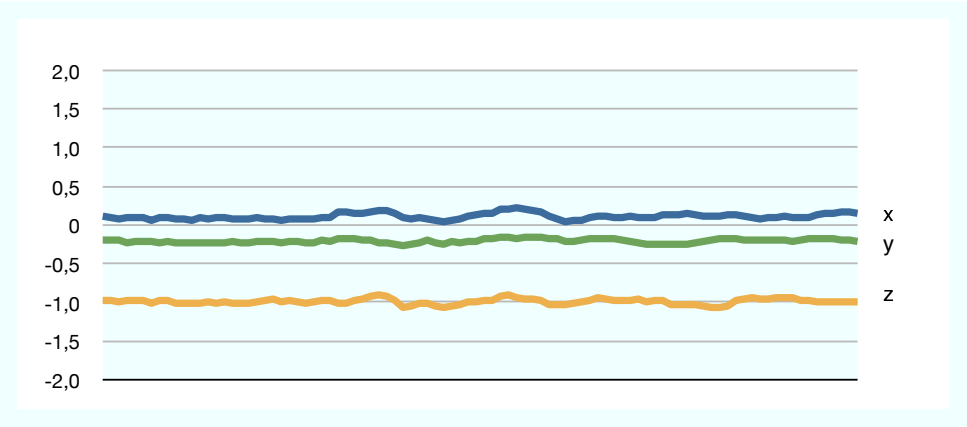


Abbildung 6.2: Beschleunigungsprofil bei der Aktivität *Stehen*

Der nächste Test wurde anhand eines normalen Laufes durchgeführt. Hier sind die Schwankungen etwas größer, aber die Mittelung der Werte führt auch hier wieder zum Erfolg. Es werden ca. 90% der Aktivitäten korrekt erkannt. Hier ist allerdings auch die Geschwindigkeit ein sehr wichtiger Indikator, da es auch andere Bewegungen und Aktivitäten gibt, die ein ähnliches Profil aufweisen. Da jedoch für den Prototypen erst einmal festgelegt wurde, dass sich dieser auf lineare Bewegungen beschränken soll, sind auch hier die Ergebnisse sehr zufriedenstellend ausgefallen.

Das dritte Profil hebt sich bereits deutlich von den beiden zuvor vorgestellten Profilen ab. Es sind deutliche Schwankungen zu erkennen. Die Intensität dieser Bewegung war auch dementsprechend hoch. Die Geschwindigkeit lag bei ca. 10 km/h. Wenn alle Sensordaten verfügbar sind, dann lässt sich auch dieses Profil sehr gut erkennen. Fallen allerdings Sensordaten aus, dann kann das zu Problemen führen. Diese Aussage kann allerdings für

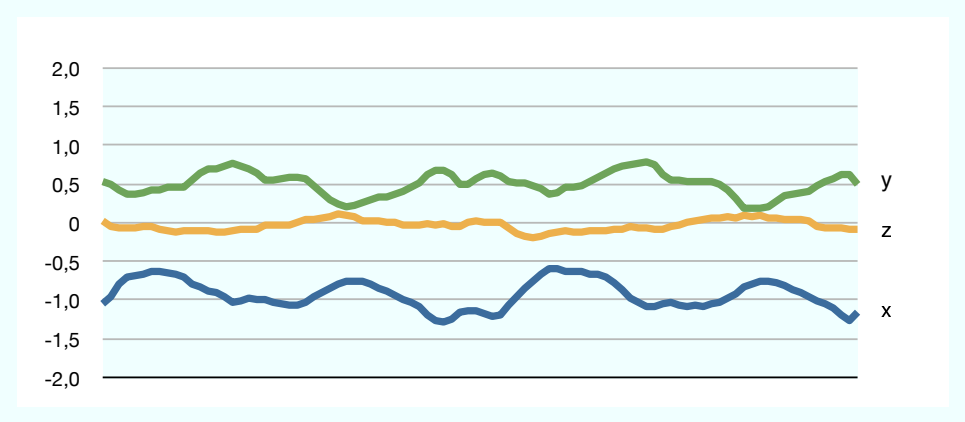


Abbildung 6.3: Beschleunigungsprofil bei der Aktivität *Laufen*

alle Profile getroffen werden. Sobald beispielsweise die Geschwindigkeit des Spielers nicht mehr bestimmt werden kann, wird es schwer, nur anhand der Beschleunigungsdaten das Aktivitätsprofil des Nutzers zu erkennen.

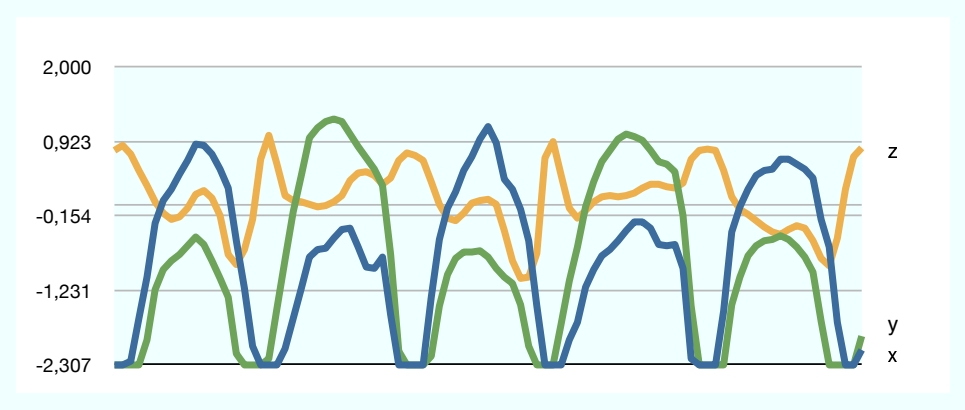


Abbildung 6.4: Beschleunigungsprofil bei der Aktivität *Joggen*

Zusammenfassend lässt sich aber feststellen, dass in ca. 89% der Fälle die Profile korrekt zugeordnet werden konnten. Unbekannte Profile sind allerdings noch schwer zu erkennen, wenn beispielsweise die Geschwindigkeit auf einen normalen Dauerlauf hindeutet. Wichtig ist es allerdings, dass die Aktivitäten mit einer hohen Intensität richtig eingestuft werden. Denn für diese bekommen die Spieler am meisten Punkte. Sollte man das Spiel in diesem Fall austricksen können, wäre das nicht von Vorteil. Bisher konnten jedoch fast alle Profile mit einer hohen Intensität korrekt zugeordnet werden.

6.1.3 Feldversuche

Der Prototyp wurde in unterschiedlichen Stadien der Entwicklung bei Feldversuchen eingesetzt, um es in einem kontinuierlichen Optimierungsprozess an die spezifischen Eigenschaften des iPhones anzupassen. Insbesondere stand dabei die Usability, die Auslastung des Speichers und die Akkulaufzeit bei unterschiedlich aktivierten Funktionalitäten im Vordergrund.

Die ersten dieser Feldversuche wurden in Hinsicht auf die Bedienbarkeit der Spiele-Anwendung durchgeführt. Es wurde untersucht, wie die Benutzerschnittstelle entworfen werden muss, um die Navigation und Interaktion innerhalb der Applikation auch bei sportlicher Betätigung gewährleisten zu können. Die Resultate ergaben, dass sich einige der durch Apple bereitgestellten Buttons und Navigationskomponenten aus der aktiven Bewegung nur schlecht oder gar nicht bedienen lassen. Dies zog eine Adaption der betreffenden Darstellungskomponenten nach sich. Weitere Tests ergaben, dass die Usability daraufhin den Anforderungen an ein bewegungsreiches Spiel im Sinne der Aufgabenstellung gerecht wird.

Daraufhin wurden spezielle Tests gemacht, um die Aktivitätserkennung zu bewerten. Dabei fiel auf, dass kurzzeitig ausgeführte Tätigkeiten bei weitem nicht so gut und genau erkannt werden können, wie langwierigere Betätigungen. Dass lässt sich leicht anhand der Berechnungen und Analysen erklären: die Algorithmen basieren momentan bei der Bestimmung Aktivität auf Durchschnittswerten. Für die GPS-Daten werden hierbei die Daten über die gesamte Aktivität gesammelt. Wenn man beispielsweise bereits eine viertel Stunde gejoggt ist, man aber nun gezwungenermaßen stehen bleiben muss, um über die Straße zu kommen, so verfälscht sich der ermittelte Wert und es wird unter Umständen eine andere Aktivität erkannt.

Bei weiteren Versuchen viel auf, dass die Umsetzung des Spiels unter Nutzung des iPhone SDKs zudem weitere Problemstellungen aufwirft: es ist beispielsweise nicht möglich die Kommunikation mit dem Server, das Sammeln der Sensordaten und die Auswertung spielrelevanter Ereignisse aufrecht zu erhalten, wenn sich das iPhone im Standby-Modus befindet. Die Richtlinien von Apple untersagen jegliches Eingreifen in die Hardware oder Hintergrundprozesse; auch das veröffentlichte API bietet keine Möglichkeit, diesen Effekt zu verhindern. Allerdings konnten zwei Möglichkeiten evaluiert werden, die jedoch die Interaktion des Spielers einbeziehen, um die korrekte Funktionsweise des Spiels zu garantieren. Einerseits kann dieser in den Einstellungen des Gerätes hinterlegen, dass die *automatische Sperre* nie genutzt werden soll und andererseits besteht die Möglichkeit durch das Hören von Musik über die *iPod-Funktion*, die Deaktivierung der Sensoren zu unterbinden.

Bei der Umsetzung und Nutzung eines Prototypen können diese Varianten akzeptiert werden, für einen realen Einsatz ist dies allerdings nicht praktikabel. Bei weiterführenden Implementierungen eines solchen Spiels, sollten allerdings, um die Qualität der Klassifikation zu steigern, weitere Sensoren genutzt werden. Auf der Plattform des iPhones ist dies allerdings nur bedingt möglich. Das Gerät bietet neben der Nutzung des GPS- und Beschleunigungs-Sensors zudem weitere Daten von der Kamera und dem Mikrophon. Auch virtuelle Sensordaten können durch den Ausbau der Kommunikation mit Web Services integriert werden. Die Nutzung externer Sensoren über die Bluetooth-Schnittstelle wird allerdings rigoros durch Apple untersagt. Auf diese Weise kann beispielsweise kein Herzfrequenzmessgerät zur genaueren Analyse der Aktivitäten hinzugezogen werden. Diese Unflexibilität stellt einen erheblichen Nachteil gegenüber anderen verfügbaren Plattformen dar: es handelt sich hierbei momentan um ein geschlossenes System.

In letzter Instanz stand bei den Feldversuchen speziell die Akkulaufzeit im Zentrum der Betrachtung. Der entwickelte Prototyp wurde von drei Testpersonen über eine Distanz von $42,195km$ beim *Dresdner Kleinwort Frankfurt Marathon* [21] am 26. Oktober eingesetzt. Da es sich bei allen dreien um Läufer handelte, spielte die Kategorisierung der Aktivität bei diesem Event eine untergeordnete Rolle. Die dabei eingesetzten iPhones wurden unterschiedlich konfiguriert: dies betraf einerseits die Frequenz der Kommunikation mit der Verwaltungsanwendung und andererseits die Einschränkung der Kommunikationskanäle durch Deaktivierung von 3G und WLAN. Nachdem die ersten Aussagen getroffen werden konnten, dass die Akkulaufzeit sehr stark von der Aktivierung des WLAN und 3G abhängt, wurden weitere Tests mit dem Fahrrad und entsprechenden Einstellungen gemacht. Auf diese Weise konnte die Akkulaufzeit soweit es ging, optimiert werden. Beim Marathon war es beispielsweise der Fall, dass die ersten iPhones nach 3 Stunden aus gingen. Das letzte der drei ging dann ca. eine dreiviertel Stunde später aus. Bei den letzten Testläufen konnte allerdings schon gezeigt werden, dass die Akkulaufzeit auf bis zu 5 Stunden maximiert werden konnte.

Neben diesen Feldversuchen wurden zudem regelmäßige Systemtests mit der in der XCode IDE integrierten Anwendung *Instruments* durchgeführt, um die Performance der Anwendung in Bezug auf die Speicherauslastung und -nutzung zu untersuchen. Hierbei können sowohl die Objekt-Allokationen als auch Speicherlecks in der Applikation aufgedeckt werden. Bei einem mobilen Gerät mit begrenztem Speicher handelt es sich hierbei um einen sehr relevanten Faktor zumal es die Aufgabe des Entwicklers ist, entsprechenden Speicher zu allokieren und wieder freizugeben.

In Abbildung 6.5 ist ein Screenshot eines solchen Testlaufs dargestellt. Es ist zu erkennen, dass im Bereich der mit *Leaks* betitelt ist, keine Daten verzeichnet werden. Das bedeutet für den Prototypen, dass er in dieser Hinsicht den Speicher nicht unnötig belastet. Zudem

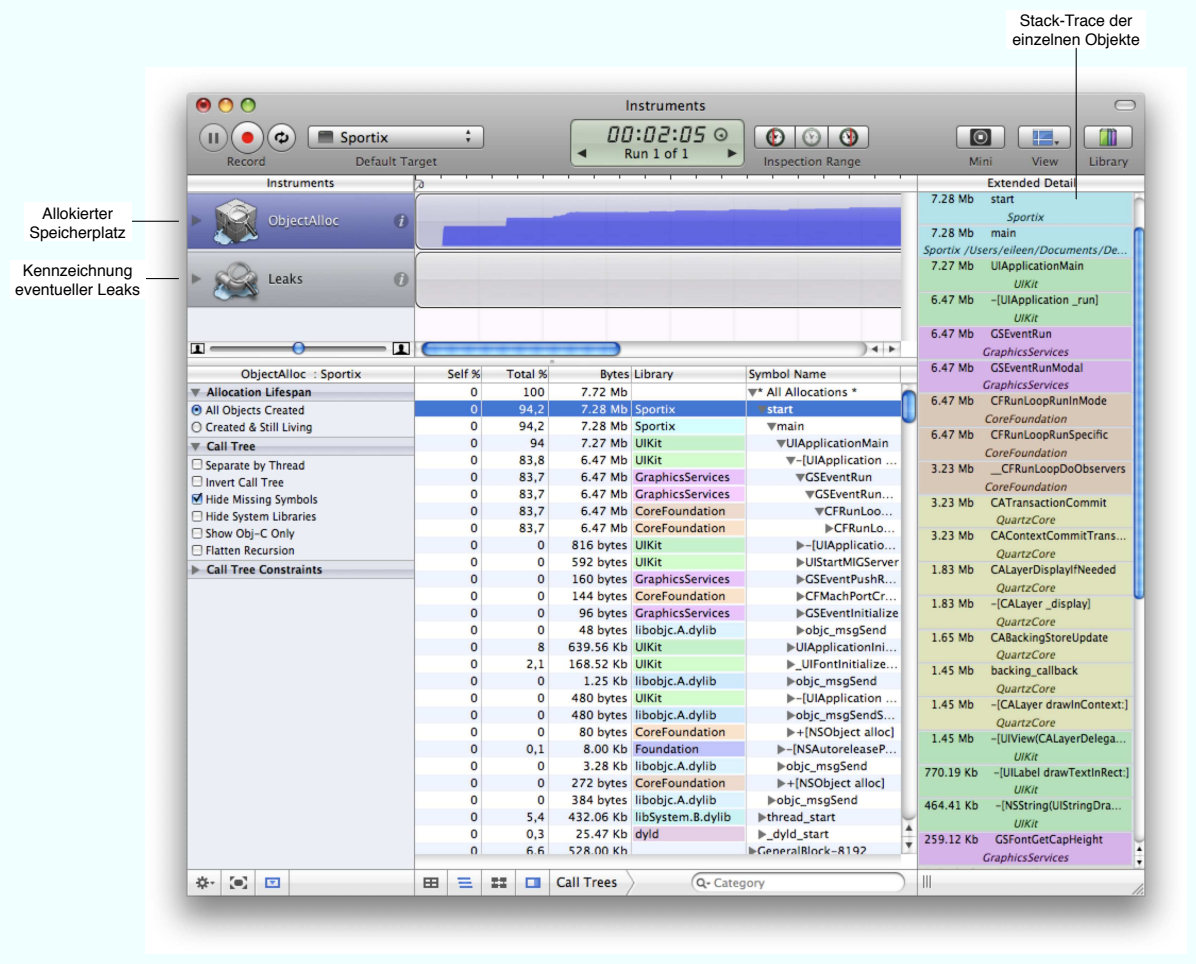


Abbildung 6.5: Auswertung der Applikation mit Instruments

ist im Bereich *ObjectAlloc* visualisiert, wann im Programm welcher Speicher allokiert wird. Beim Starten des Programms ist dies so umgesetzt, dass große Grafiken bereis vorgeladen werden, damit diese beim Ausführen der Applikation schnell geladen und genutzt werden können um auf diese Weise der Usability in einem weiteren Punkt gerecht zu werden und keine unnötigen Wartezeiten zu generieren.

6.2 Demonstration der Funktionalität

Zum Aufzeigen der Implementierung werden ausgewählte Funktionalitäten der Verwaltungs- als auch der Spiele-Anwendung unter Hinzunahme von Screen-Shots aufgezeigt. Dabei wird der Ansatz verfolgt, den kompletten Workflow von der Erstellung einer Aufgabe seitens der Autoren, über die Registrierung eines neuen Spielers, bis hin zur Durchführung des Spiels zu beschreiben.

6.2.1 Funktionen der Verwaltungs-Anwendung

Startseite Bei einem ersten Kontakt mit dem System ist zu beachten, dass es zwei Startseiten gibt: eine für die Autoren und Administratoren und eine, die dem Nutzer den Einstieg in das Spiel ermöglichen soll – das so genannte *Rabbit Hole*[1].

Abbildung 6.6 zeigt die Startseite des Spielers. Er hat auf dieser Seite den ersten Kontakt zu dem entwickelten Spiel. Im mittleren Bereich der Seite stehen ihm zwei Links zur Verfügung: „Vorgeschichte“ und „Download“. Bevor er das Spiel auf dem iPhone installiert, kann er sich über den Hintergrund und die Geschichte des Spiels informieren. Ist sein Interesse geweckt, könnte er sich theoretisch das Spiel herunterladen. In der Praxis verhält es sich allerdings anders: die Software muss mit einer entsprechenden Lizenz versehen werden, damit diese überhaupt auf dem iPhone installiert werden kann.

Die Startseite für die Autoren und Administratoren ist in Abbildung 6.7 zu sehen. Diese ist sehr viel einfacher gestaltet. Sie ermöglicht es, sich am System anzumelden. Nach erfolgreicher Anmeldung werden die verfügbaren Navigationsmöglichkeiten des Nutzers, entsprechend seiner Rolle im System, im linken Bereich der Seite dargestellt. Im Inhaltsbereich wird eine aktuelle Statistik des Spiels dargestellt: beispielsweise die Anzahl der registrierten Spieler oder wer gerade online ist.

Benutzerverwaltung Die Benutzerverwaltung stellt im jeweiligen Bereich die Anzeige und Änderungsmöglichkeit der Registrierungsdaten des angemeldeten Benutzers dar. Im Administratorbereich ist zusätzlich die Option vorhanden, ein neues Nutzerkonto zu erstellen. Auf diese Weise können Nutzer angelegt werden, die unterschiedliche Rechte im System haben: sowohl Spieler, Autoren als auch Administratoren. In Abbildung 6.8 wird der Prozess des Anlegens eines neuen Nutzers visualisiert.

[1] In Anlehnung an Lewis Carrol‘s Alice im Wunderland wird der Einstiegspunkt in Pervasive Spiele oftmals mit dem Begriff Rabbit Hole gleichgesetzt.

Abbildung 6.6: Startseite des Spiels Sportix

Verwaltung der Storyline Autoren haben im System die Möglichkeit, die Storyline des Spiels entsprechend anzupassen und zu erweitern. Er kann sich beispielsweise die Übersicht der im System vorhandenen StoryEvents anzeigen lassen. Der Autor kann sich durch Auswahl eines Events zudem weitere Details anzeigen lassen und diese entsprechend editieren. Neben den grundsätzlichen Daten eines solchen StoryEvents, der Beschreibung und dem Namen, werden zusätzlich die Bedingungen visualisiert, die erfüllt sein müssen, damit dieses Event beim Spieler angezeigt wird. Dabei kann es sich beispielsweise um die Spieldauer oder um das Erreichen einer bestimmten Punktzahl handeln.

Neben der Funktion diese Daten zu visualisieren und zu verändern, gibt es zusätzlich die Möglichkeit ein neues StoryEvent anzulegen, um es somit in der StoryLine zu integrieren. Dieser Prozess wird ähnelt in der Visualisierung dem des Anlegen eines neuen Nutzers, daher wird an dieser Stelle auf den entsprechenden Screenshot verzichtet.

Die Applikation kann zudem die zuvor vorgestellten Elemente in der englischen Sprachversion darstellen. In diesem Fall werden sowohl die Punkte der Navigation als auch die Formularbeschriftungen und Inhalte der Seite in englisch dargestellt.

6.2.2 Funktionen der Spiele-Anwendung

Einstellungen im System Bevor das Spiel gespielt werden kann, muss der Nutzer seine Registrierungsdaten im Bereich der Einstellungen der Applikation hinterlassen. Er hat

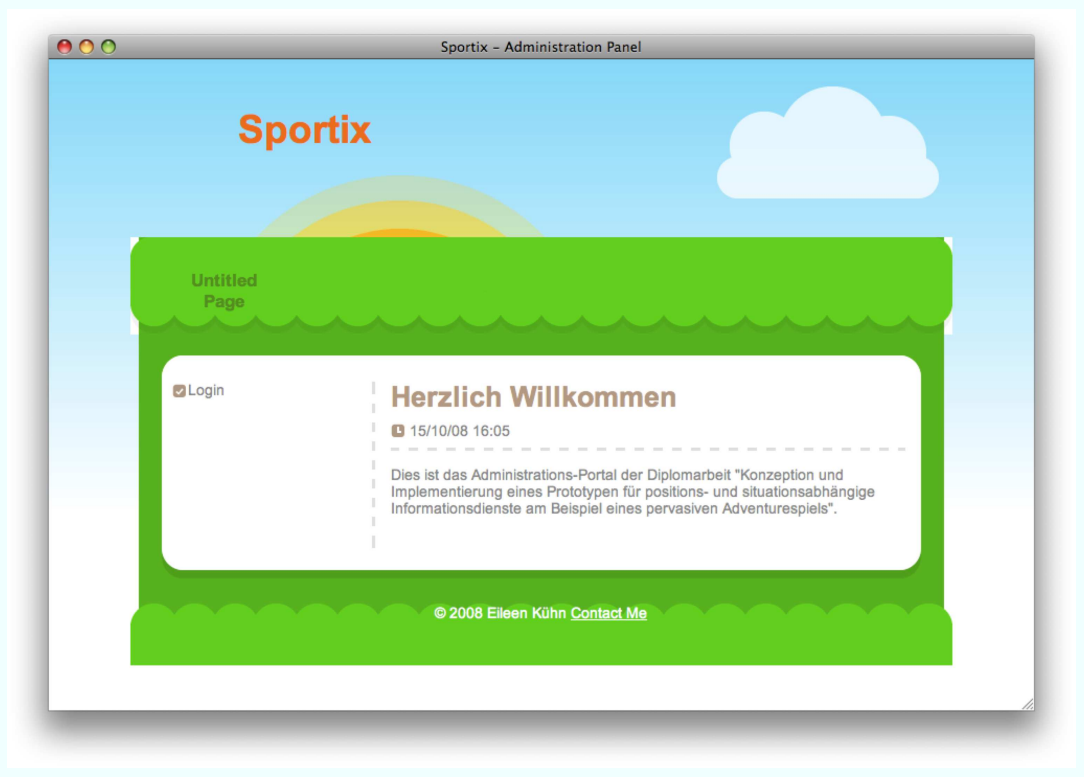


Abbildung 6.7: Startseite der Verwaltungs-Anwendung

dort die Möglichkeit, seinen Nutzernamen und Passwort zu hinterlegen. Dies skizziert Abbildung 6.9(a). Zusätzlich können erweiterte Einstellungen vorgenommen werden. Möchte der Spieler beispielsweise nicht, dass er von anderen Spielern als aktiv erkannt wird, so kann er dies deaktivieren. Außerdem kann eingestellt werden, in welchem Intervall der Kontakt zum Server aufrecht erhalten werden soll. In diesem Fall muss der Spieler das allerdings nicht händisch eintippen, er kann sich die möglichen Werte bequem auswählen (siehe Abbildung 6.9(b))

Live-Ansicht des Spiels Bei der Durchführung des Spiels befindet sich der Spieler die ganze Zeit über in der Live-Ansicht. Diese kann durch den linken Button in der TabBar aufgerufen werden. Auf dieser Ansicht werden alle Informationen gebündelt. Während der Bewegung wird er beispielsweise informiert, wenn eine neue Aufgabe verfügbar ist oder sich ein Spieler in seiner Nähe befindet.

Nachdem der Spieler eine Aufgabe angenommen hat, hat er eine bestimmte Bedingung zu erfüllen. Abbildung 6.10(a) zeigt beispielsweise, dass der Spieler gerade läuft. Dies ist erkennbar an dem Icon in der rechten unteren Ecke. Sobald er die Aufgabe geschafft hat, wird er darüber informiert und bekommt die Aktivitätspunkte, die ihm zustehen. Diesen Augenblick visualisiert Abbildung 6.10(b).

Abbildung 6.8: Anlegen eines neuen Nutzers in der Verwaltungs-Anwendung

Listenbasierte Ansichten des Spiels Neben der Live-Ansicht gibt es noch listenbasierte Ansichten. Diese zeigen in Abhängigkeit der Entfernung entweder andere Spieler, oder Aufgaben, die sich in der Nähe befinden. Diese Ansichten sollten aber manuell nicht unbedingt während des Spiels genutzt werden. Denn sobald sich eine passende Gelegenheit ergibt, werden diese automatisch in der Live-Ansicht visualisiert.

(a) Übersicht über mögliche Einstellungen der Applikation

(b) Ansicht der Einstellung des Request Intervalls

Abbildung 6.9: Einstellungen der Applikation Sportix

(a) Live-Ansicht der Applikation (b) Live-Ansicht nach erledigter Aufgabe

Abbildung 6.10: Einstellungen der Applikation Sportix

6.2.3 Zusammenfassung

Zusammenfassend möchte ich gern einige Daten visualisieren. Anbei sind jeweils zwei Spieler dargestellt, deren Daten im Nachhinein aufbereitet wurden, um die Funktionalität zusammenfassend zu präsentieren. Der erste Spieler in Abbildung 6.11 und 6.12 hat eine längere Strecke mit dem Rad zurückgelegt. In der Kartenansicht wurde speziell ein Teil der Strecke hervorgehoben, der zu Beginn der Implementierungen nur sehr unsauber visualisiert werden konnte. Die Verbesserungen in den Algorithmen, speziell die Umsetzung des Schwerpunkt-Verfahrens, haben es ermöglicht, diese Daten zu glätten. Auch die Kurve seiner Geschwindigkeiten sieht sehr solide aus. Es gibt eine Piks, die eine etwas unrealistische Geschwindigkeit angeben, doch auch das hat den Erkennungsalgorithmus nicht durcheinandergebracht.

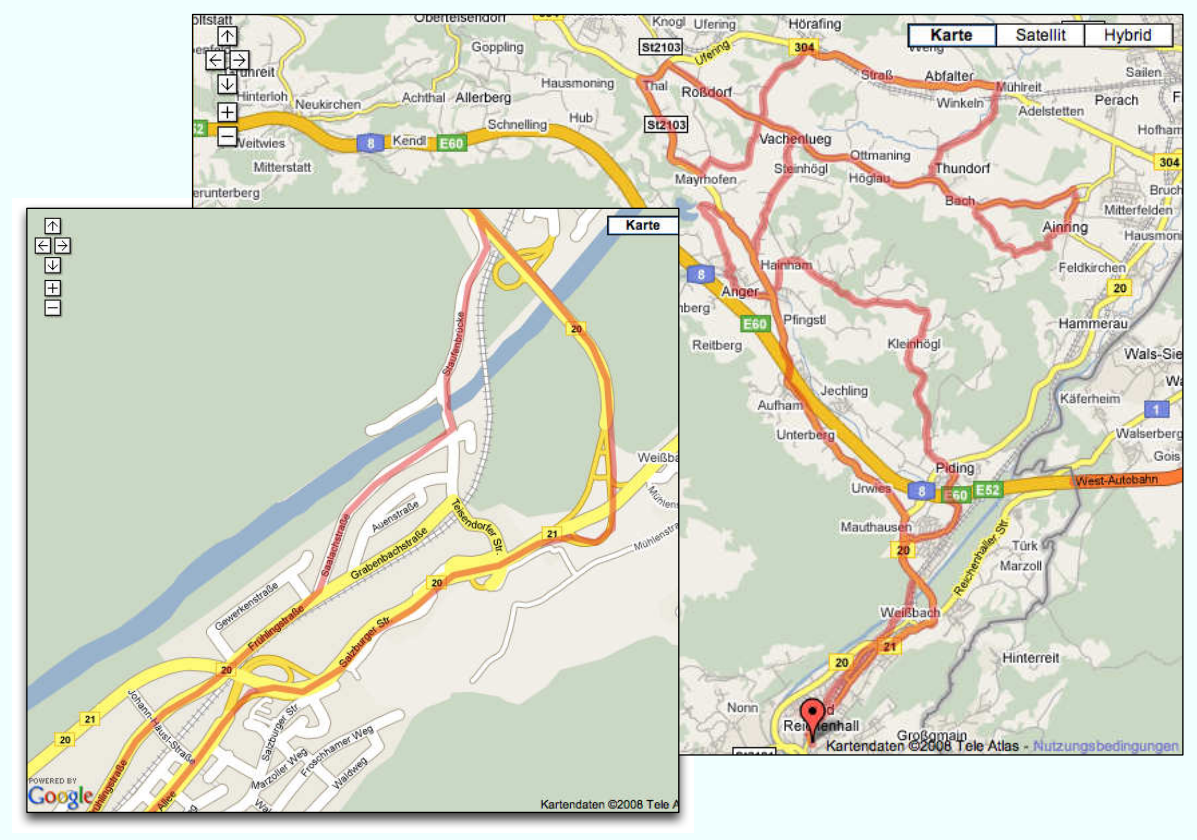


Abbildung 6.11: Gesamtübersicht und Detailausschnitt einer Radtour

Etwas schlechter sehen die Daten allerdings noch bei Läufern aus. Aufgrund der niedrigen Geschwindigkeit und demzufolge der sehr geringen Distanz, die zwischen den einzelnen Snapshots zurückgelegt wird, gibt es hier etwas unsaubere Daten. Trotzdem ist auch dieses Ergebnis bereits sehr zufrieden stellend und es wurde sehr gut erkannt, dass diese Strecke joggend zurückgelegt wurde.

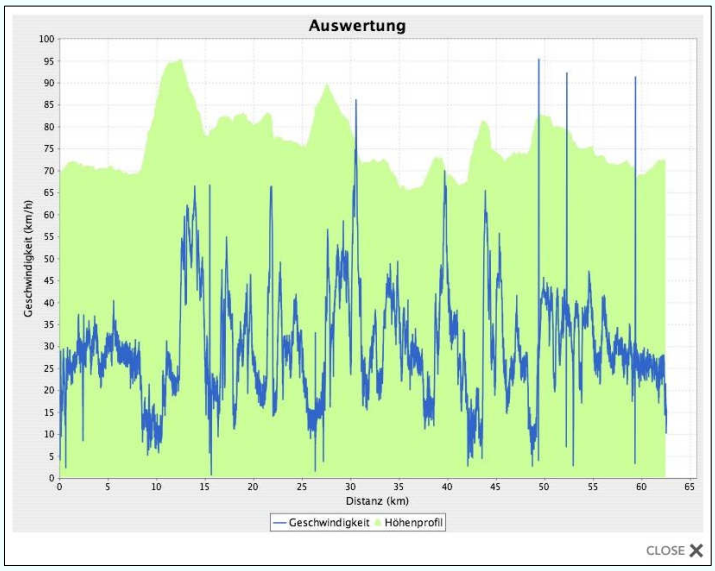


Abbildung 6.12: Geschwindigkeitsprofil der Radtour

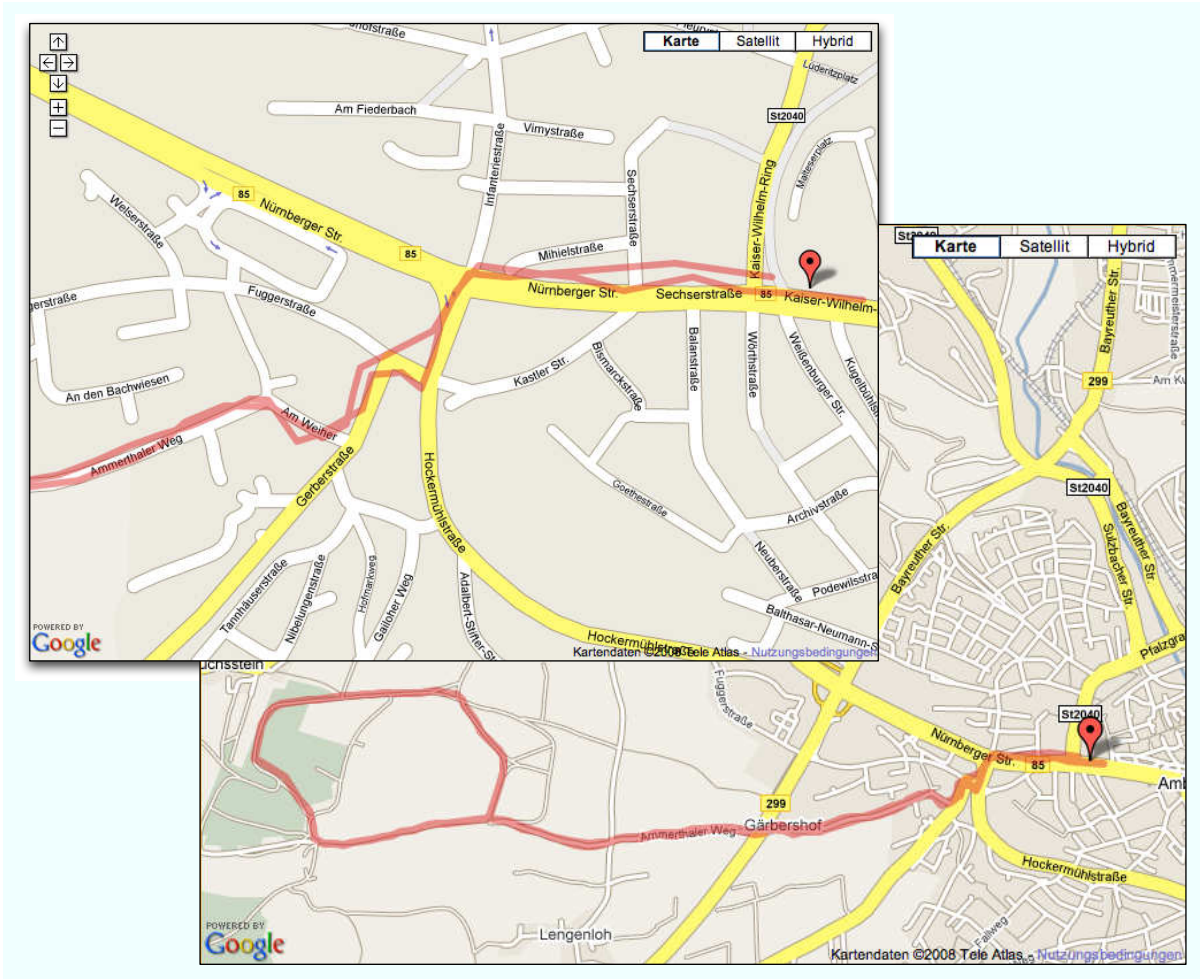


Abbildung 6.13: Gesamtübersicht und Detailausschnitt einer Jogging-Runde

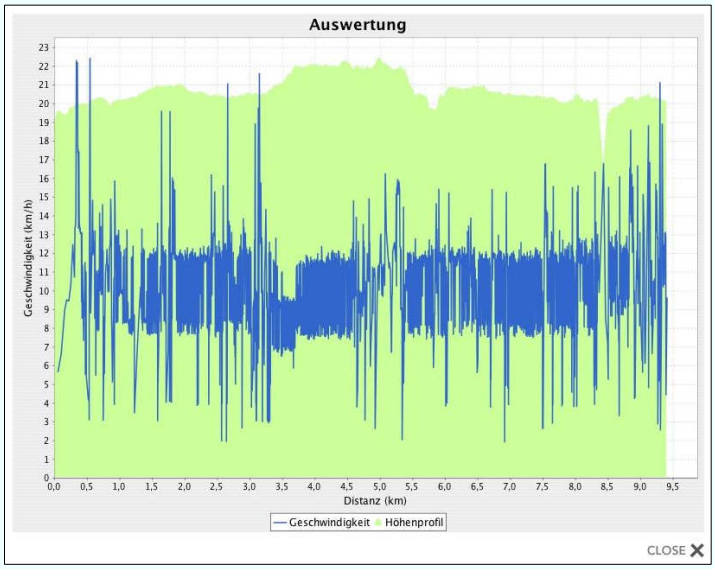


Abbildung 6.14: Geschwindigkeitsprofil der Jogging-Runde

Kapitel 7

Zusammenfassung und Ausblick

Das Ziel dieser Arbeit war es, ein pervasives Adventurespiel zu konzipieren und prototypisch zu implementieren, dass in Abhängigkeit der Aktivität und Position des Nutzers entsprechende Informationen und Aufgaben visualisiert, die von einem zentralen Server abgerufen werden.

Um die gestellte Aufgabe zu lösen wurden im theoretischen Teil dieser Arbeit zunächst die benötigten Technologien und Konzepte der Positions- und Kontextgewinnung untersucht und ein Rahmenwerk auf der Basis serviceorientierter Architekturen für die Nutzung im Pervasive Computing konzipiert. Zudem wurden aktuelle Entwicklungen und Trends im Bereich der Smartphones vorgestellt, um die für die Kontextgewinnung nutzbaren Sensordaten zu identifizieren. Das Ergebnis dessen war die Identifizierung der GPS- und Beschleunigungs-Sensoren.

Die Analyse ausgewählter, vorhandener Anwendungen zeigte Umsetzungsmöglichkeiten und Defizite im Bereich des Pervasive Gaming auf, woraufhin die Anforderungen des zu entwickelnden Spiels konkretisiert wurden. Es wurde festgelegt, dass es sich bei der Mobilität, der Konnektivität, der Ortsspezifität und der öffentliche Interaktion um die zentralen Konzepte des pervasiven Spiels handeln soll.

Unter Beachtung serviceorientierter Systemarchitekturen wurden die einzelnen Schichten der Spiele- und Verwaltungsanwendung entworfen. Dies umfasste die Datenhaltung, die einzelnen Komponenten der Anwendungslogik, die Präsentationsschicht sowie die Kommunikation innerhalb des Systems über Web Services. Diese Komponenten wurden im praktischen Teil der Arbeit zu einem Gesamtsystem kombiniert und in Software-Module überführt.

Die Zuverlässigkeit und Leistungsfähigkeit des prototypisch umgesetzten Systems wurde durch unterschiedliche Feldversuche im Bundesgebiet Deutschlands demonstriert und in

mehreren Iterationen evaluiert. Für ca. 89% aller durchgeführten Aktivitäten lieferte das System hierbei das richtige Ergebnis, d.h. die Punktzahl der Spieler, das Lösen der Aufgaben und die positions- und situationsabhängige Abfrage neuer Aufgaben konnte korrekt durchlaufen werden. Fehlerhafte Ergebnisse lieferte das System in ca. 10% der getesteten Fälle. Lediglich für Aktivitäten, die nur für Bruchteile von Sekunden durchgeführt wurden, konnte in den meisten Fällen keine korrekte Erkennung erzielt werden. Die Ergebnisse lassen den Schluss zu, dass bereits die Aggregation zwei unterschiedlicher Sensoren die Integration der Umwelt in pervasive Spiele zuverlässig verbessern kann.

Das Ziel dieser Diplomarbeit, ein pervasives Adventurespiel zu entwickeln, dass in Abhängigkeit der Position und Situation des Nutzers Aufgaben und Informationen zur Verfügung stellt, wurde demnach erreicht: das hier vorgestellte Informationssystem löst die Aufgabe erfolgreich und in bereits guter prototypischer Qualität.

7.1 Ausblick

Das pervasive Adventurespiel Sportix liefert für im System vordefinierte Aktivitäten bereits gute Ergebnisse. Vorausgesetzt wird hierbei allerdings, dass diese für den entsprechenden Spieler vorkonfiguriert vorliegen. Zukünftige Weiterentwicklungen des hier vorgestellten Systems sollten daher primär zum Ziel haben, das im Rahmen der Arbeit entworfene Rahmenwerk zur Kontextgewinnung um entsprechende Algorithmen zu erweitern.

Hohes Potential für die Optimierung der Anwendung bietet in erster Linie der eingesetzte Algorithmus für die Situationserkennung: dieser ließe sich beispielsweise sehr gut durch Strategien der Nächsten-Nachbar-Suche, den Einsatz von Partikelfiltern oder dem Naiven Bayes-Klassifikator erzielen.

Überdies sollte die soziale Interaktion und Kommunikation zwischen den Spielern durch die Erweiterung des Systems realisiert werden. Dies kann beispielsweise durch den Aufbau von Ad-hoc-Netzen zwischen den iPhones der Spieler erreicht werden: Informationen die beispielsweise nur einmal im Stadtgebiet gefunden werden können, könnten auf diese Weise ausgetauscht werden. Außerdem sollte eine Art „Verfolgungsrennen“ realisiert werden. Ein Spieler läuft voraus und ein zweiter muss versuchen, diesem so lange wie nur möglich zu folgen. Dies kann zudem für Gruppen von Spielern realisiert werden. Um dies zu honorieren, könnte zusätzlich das Modell der Berechnung der Aktivitätspunkte angepasst werden, um diese in Abhängigkeit der Anzahl der Spieler in der Nähe zu bestimmen.

Auch wenn der hier vorgestellte Prototyp bereits gute Ergebnisse liefert, ließe sich durch die Integration zusätzlicher Sensordaten die Individualität des Spielverlaufs und der Spiel-

gestaltung enorm verbessern. Unter Nutzung des iPhone SDKs besteht allerdings lediglich die Möglichkeit, neben den bereits eingebundenen Sensordaten, Audio oder Video zu nutzen. Schnittstellen zu externen Sensoren dürfen beispielsweise über die Bluethooth-Schnittstelle nicht integriert werden. Deshalb sollte die Nutzung virtueller Sensoren oder Sensoren, die über die WLAN-Schnittstelle angesprochen werden können, genutzt werden. Eine enorme Verbesserung in Bezug auf die Spielelogik würde die Integration eines Herzfrequenzmessgeräts bieten. Die Bestimmung der Aktivitätspunkte könnte so sehr viel genauer und individualisierter bestimmt werden – das Spiel könnte dann auch für Trainierte Sportler so gestaltet werden, dass diese gefordert werden. Die Herzfrequenz ist dafür ein sehr guter Indikator.

Vor der Nutzung des Spiels muss zudem die Verwaltungs-Anwendung entsprechend ausgebaut werden. Diese sollte durch Integration von AJAX-Funktionalitäten die Interaktionen der Administratoren und Autoren in Bezug auf die Usability verbessern, ohne zuvor die komplette Seite laden zu müssen. Hierbei sollen exemplarisch die Autovervollständigung in Text-Eingabefeldern genannt werden. Des Weiteren ist die Integration von Services, wie bspw. der Google Maps API angedacht, um eine visuelle Auswertung der aktiven Spieler zu ermöglichen.

Im Rahmen einer Weiterentwicklung ist auch eine Umverteilung der Zuständigkeiten innerhalb der entworfenen Client-Server Architektur denkbar, z.B. könnte die im Rahmen der Analyse vorgestellte pervasive Infrastruktur umgesetzt werden. Es gibt bereits heute eine große Anzahl an technischen Geräten, die physikalische Sensoren integrieren oder virtuelle Sensoren über ihre Kommunikationsschnittstelle auslesen können. Diese könnten auf diese Weise, unter Voraussetzung einer gewissen Standardisierung im Bereich des Pervasive und Ubiquitous Computing, bequem in das System integriert werden. In Zukunft wird es immer mehr darum gehen, die technischen Möglichkeiten sinnvoll miteinander zu verknüpfen um die Anwendung für den Nutzer intuitiv und adaptierbar zu gestalten und ihm die Informationen zur Verfügung zu stellen, die er benötigt.

Literaturverzeichnis

[AFW02] Amberg, Michael ; Figge, Stefan ; Wehrmann, Jens: Compass - Ein Kooperationsmodell für situationsabhängige mobile Dienste. In: *Mobile and Collaborative Business 2002, Proceedings zur Teilkonferenz der Multikonferenz Wirtschaftsinformatik 2002*, GI, 2002. – ISBN 3–88579–345–8, S. 31–50

[AHW+00] Ainworth, BE ; Haskell, WL ; Whitt, MC ; Irwin, ML ; Swartz, AM ; Strath, SJ ; O'Brien, WL ; Basset, DR J. ; Schmitz, KH ; Emplaincourt, PO ; Jacobs, DR J. ; Leon, AS: Compendium of physical activities: An update of activity codes and MET intensities. In: *Medicine and Science in Sports and Exercise* 32 (2000), S. 489–516

[AW02] Amberg, Michael ; Wehrmann, Jens: A Framework for the Classification of Situation Dependent Services. In: *Proceedings of the Eighth Americas Conference on Information Systems*. Dallas, USA, 2002, S. 1838–1843

[Bau02] Bauer, Manfred: *Vermessung und Ortung mit Satelliten. GPS und andere satellitengestützte Navigationssysteme*. 5. Auflage. Heidelberg : Herbert Wichmann Verlag, 2002. – ISBN 3–879–07360–0

[BBC97] Brown, Peter J. ; Bovey, John D. ; Chen, Xi A.: Context-aware Applications: from the Laboratory to the Marketplace. In: *IEEE Personal Communications* 4 (1997), Oktober, S. 58–64

[BHH+01] Burkhardt, Jochen ; Henn, Horst ; Hepper, Stefan ; Rindtorff, Klaus ; Schäck, Thomas: *Pervasive Computing. Technologie und Architektur mobiler Internetanwendungen*. München : Addison-Wesley, 2001. – ISBN 3–827–31729–0

[BI04] Bao, Ling ; Intille, Stephen S.: Activity Recognition from User-Annotated Acceleration Data. In: *Pervasive 2004* (2004), April, S. 1–17

[Bru04] Bruss, Franz T.: Strategien der besten Wahl, 2004, S. 102–104

[CA98] Chen, Jian guo ; Ansari, Nirwan: Adaptive fusion of correlated local decisions. In: *IEEE Trans. on Systems, Man, and Cybernetics* 28 (1998), Nr. 2, S. 276–281

[CB07] Chen, Ling ; Benford, Steve: Your way your missions: from location-based to route-based pervasive gaming. In: *Advances in Computer Entertainment Technology*, 2007, S. 232–233

[CCRR02] Crowley, James L. ; Coutaz, Joëlle ; Rey, Gaeten ; Reignier, Patrick: Perceptual Components for Context Aware Computing. In: *UBICOMP 2002, International Conference on Ubiquitous Computing, Goteborg*, Springer, 2002, S. 117–134

[CGK05] Christopoulou, Eleni ; Goumopoulos, Christos ; Kameas, Achilles: An ontology-based context management and reasoning process for UbiComp applications. In: *sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence*. New York, NY, USA : ACM, 2005. – ISBN 1–59593–304–2, S. 265–270

[CGS+02] Cheng, Shang-Wen ; Garlan, David ; Schmerl, Bradley ; Sousa, João ; Spitznagel, Bridget ; Steenkiste, Peter ; Hu, Ningning: Software Architecture-Based Adaptation for Pervasive Systems. Version: 2002. `http://dx.doi.org/10.1007/3-540-45997-9_7`. 2002, 217–233

[CP98] Clarkson, B. ; Pentland, A.: Extracting Context from Environmental Audio. In: *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1998. – ISBN 0–8186–9074–7, S. 154

[Cra84] Crawford, Chris: *Art of Computer Game Design: Reflections of a Master Game Designer*. Osborne/McGraw-Hill, 1984. – ISBN 0–078–81117–1

[CXC+05] Cao, Jiannong ; Xing, Na ; Chan, Alvin T. S. ; Feng, Yulin ; Jin, Beihong: Service Adaptation Using Fuzzy Theory in Context-Aware Mobile Computing Middleware. In: *RTCSA '05: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. Washington, DC, USA : IEEE Computer Society, 2005. – ISBN 0–7695–2346–3, S. 496–501

[DBT+06] Drozd, Adam ; Benford, Steve ; Tandavanitj, Nick ; Wright, Michael ; Chamberlain, Alan: Hitchers: Designing for Cellular Positioning. In: *UbiComp 2006: Ubiquitous Computing* (2006), S. 279–296

[DSA01] Dey, Anind ; Salber, Daniel ; Abowd, Gregory D.: A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. In: *Human-Computer Interaction (HCI) Journal* 16 (2001), S. 97–166

[EM05] Ermi, Laura ; Mäyrä, Frans: Player-Centred Game Design: Experiences in Using Scenario Study to Inform Mobile Game Design. In: *Game Studies* 5 (2005), Nr. 1

[FFSB04] Freeman, Eric ; Freeman, Elisabeth ; Sierra, Kathy ; Bates, Bert: *Head First Design Patterns*. Sebastopol : O'Reilly Media, 2004. – ISBN 0–596–00712–4

[Fin06] Finkenzeller, Klaus: *RFID-Handbuch : Grundlagen und praktische Anwendungen induktiver Funkanlagen, Transponder und kontaktloser Chipkarten*. 4. Hanser, 2006. – ISBN 3–446–40398–1

[GB00] Gibson, Jerry D. (Hrsg.) ; Bovik, Al (Hrsg.): *Handbook of Image and Video Processing*. Orlando, FL, USA : Academic Press, Inc., 2000. – ISBN 0–121–19790–5

[GJ01] Gessler, Stefan ; Jesse, Kai: Advanced location modeling to enable sophisticated lbs provisioning in 3g networks. In: *In Proceedings of Location Modeling Workshop at Ubicomp*, 2001

[GPZ04] Gu, Tao ; Pung, Hung K. ; Zhang, Da Q.: A Bayesian approach for dealing with uncertain contexts. In: *Austrian Computer Society* Bd. 176, 2004, S. 205–210

[GWPZ04] Gu, Tao ; Wang, Xiao H. ; Pung, Hung K. ; Zhang, Da Q.: An ontology-based context model in intelligent environments. In: *In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004, S. 270–275

[Hei98] Heineman, George T.: Adaptation and software architecture. In: *ISAW '98: Proceedings of the third international workshop on Software architecture*. New York, NY, USA : ACM, 1998. – ISBN 1–58113–081–3, S. 61–64

[HKRS02] Hitz, Martin ; Kappel, Gerti ; Retschitzegger, Werner ; Schwinger, Wieland: Ein UML-basiertes Framework zur Modellierung ubiquitärer Web-Anwendungen. In: *Wirtschaftsinformatik* 44 (2002), Nr. 3, S. 225 – 235

[HNBr97] Hull, Richard ; Neaves, Philip ; Bedford-roberts, James: Towards Situated Computing. In: *In Proceedings of The First International Symposium on Wearable Computers*, 1997, S. 146–153

[HNS01] Hansmann, Uwe ; Nicklous, Martin S. ; Stober, Thomas: *Pervasive computing handbook*. New York, NY, USA : Springer-Verlag New York, Inc., 2001. – ISBN 3–540–67122–6

[Hof04] Hofstadter, Douglas R.: *Gödel, Escher, Bach ein Endloses Geflochtenes Band*. 10. Auflage. München : Deutscher Taschenbuch Verlag, 2004

[IHS04] Ibach, Peter ; Hübner, Tobias ; Schweigert, Martin: MagicMap - Kooperative Positionsbestimmung über WLAN. Version: 2004. `http://www.ccc.de/congress/2004/fahrplan/files/48-ad-hoc-collaboration-paper.pdf`. In: *Chaos Communication Congress Online Proceedings*. 2004

[IN00] Imagery, National ; (NIMA), Mapping A.: Department of Defense World Geodetic System 1984 – Its Definition and Relationships With Local Geodetic Systems. 2000. – NIMA Technical Report

[Inf06] Informationstechnik, Bundesamt für Sicherheit in der: *Pervasive Computing: Entwicklungen und Auswirkungen*. Bonn : SecuMedia, 2006 `http://www.bsi.bund.de/literat/studien/percenta/Percenta_bfd.pdf`. – ISBN 3–922–74675–6

[JRJ05] Jin, Hai (Hrsg.) ; Reed, Daniel A. (Hrsg.) ; Jiang, Wenbin (Hrsg.): *Network and Parallel Computing, IFIP International Conference, NPC 2005, Beijing, China, November 30 - December 3, 2005, Proceedings*. Bd. *3779*. Springer, 2005 (Lecture Notes in Computer Science). – ISBN 3–540–29810–X

[Ken02] Kennedy, Michael: *The Global Positioning System and GIS: An Introduction*. 2. Auflage. London, New York : CRC Press, 2002

[KKP+03] Korpipää, Panu ; Koskinen, Miika ; Peltola, Johannes ; Mäkelä, Satu-Marja ; Seppänen, Tapio: Bayesian approach to sensor-based context awareness. In: *Personal Ubiquitous Comput.* 7 (2003), Nr. 2, S. 113–124. – ISSN 1617–4909

[Kna02] Knauffels, Franz-Joachim: *Wireless LANs*. Bonn : mitp-Verlag, 2002

[Kra06] Krause, Michael: *Kontextbereitstellung in offenen, ubiquitären Systemen*, Ludwig-Maximilians-Universität München, Diss., September 2006

[Ler04] Lerch, Reinhard: *Elektrische Messtechnik: Analoge, digitale und computergestützte Verfahren.* 2. Auflage. Berlin, Heidelberg, New York : Springer, 2004. – ISBN 3–540–21870–X

[LGM01] Ladetto, Q. ; Gabaglio, V. ; Merminod, B.: Combining Gyroscopes, Magnetic Compass and GPS for Pedestrian Navigation. In: *International Symposium on Kinematic Systems in Geodesy, Geomatics and Navigation (KIS), Banff, Canada*, 2001, S. 205–212

[Man04] Mansfeld, Werner: *Satellitenortung und Navigation. Grundlagen und Anwendung globaler Satellitennavigationssysteme.* 2. Auflage. Wiesbaden : Vieweg Verlag, 2004. – ISBN 3–528–16886–5

[May04] Mayrhofer, Rene: An Architecture for Context Prediction. In: *In Advances in Pervasive Computing, number 3-85403-176-9. Austrian Computer Society (OCG*, 2004

[MCMN05] Magerkurth, Carsten ; Cheok, Adrian D. ; Mandryk, Regan L. ; Nilsen, Trond: Pervasive games: bringing computer entertainment back to the real world. In: *Comput. Entertain.* 3 (2005), Nr. 3, S. 4–4. – ISSN 1544–3574

[NTGCP05] Neto, Renato B. ; Teixeira, César A. C. ; Graça Campos Pimentel, Maria da: A Semantic Web-Based Infrastructure Supporting Context-Aware Applications. In: [YAL+05], S. 900–909

[OLW+08] Oates, Richard ; Langer, Thomas ; Wille, Stefan ; Lueckow, Torsten ; Bachlmayr, Gerald: *Spring Hibernate. Eine praxisbezogene Einführung.* 2. Auflage. München, Wien : Hanser, 2008. – ISBN 3–446–41213–1

[PB02] Prekop, Paul ; Burnett, Mark: Activities, Context and Ubiquitous Computing. In: *CoRR* cs.IR/0209021 (2002)

[Pop08] Popp, Gunther: *Konfigurationsmanagement mit Subversion, Ant und Maven. Grundlagen für Softwarearchitekten und Entwickler.* 2. Auflage. Heidelberg : dpunkt.verlag, 2008. – ISBN 3–898–64487–1

[Rot05] Roth, Jörg: *Mobile Computing: Grundlagen, Technik, Konzepte.* 2. Auflage. Dpunkt Verlag, 2005. – ISBN 3–898–64366–2

[Sad04] Sadewasser, Thomas: *Geocaching: Suchen und Verstecken mit GPS - Unterstützung.* BoD - Books on Demand, 2004. – ISBN 3–833–41540–1

[SBG99] Schmidt, Albrecht ; Beigl, Michael ; Gellersen, Hans w.: There is more to context than location. In: *Computers and Graphics* 23 (1999), S. 893–901

[Sch02] Schmidt, Albrecht: *Ubiquitous computing – computing in context*, Lancaster University, Diss., November 2002

[Sch03] Schiller, Jochen: *Mobilkommunikation*. Pearson Studium, 2003. – ISBN 3–827–37060–4

[Sch07] Schuytema, Paul: *Game Design: A Practical Approach*. Boston : Charles River Media, 2007. – ISBN 3–898–64465–0

[SS05] Schoblick, Robert ; Schoblick, Gabriele: *RFID Radio Frequency Identification*. Poing : Franzis Verlag, 2005. – ISBN 3–772–35920–5

[ST94] Schilit, Bill N. ; Theimer, Marvin M.: Disseminating active map information to mobile hosts. In: *IEEE Network* 8 (1994), S. 22–32

[ST07] Scheible, Jürgen ; Tuulos, Ville: *Mobile Python. Rapid Prototyping of Applications on the Mobile Platform*. Chichester : Wiley, 2007. – ISBN 0–470–51505–8

[TRPC08] Tippenhauer, Nils O. ; Rasmussen, Kasper B. ; Pöpper, Christina ; Capkun, Srdjan: Attacks on Public WLAN-based Positioning Systems. SysSec technical Report, April 2008. – Forschungsbericht

[TS80] Tenney, Robert R. ; Sandell, Nils R.: Detection with distributed sensors. In: *Decision and Control including the Symposium on Adaptive Processes, 1980 19ht IEEE Conference* Bd. 19, 1980, S. 433–437

[TW05] Tang, Yunting ; Wu, Qing: A Semantic and Adaptive Context Model for Ubiquitous Computing. In: [JRJ05], S. 495–502

[Vos95] Voser, Stefan A.: Datenaustausch zwischen GIS - Abbildung zwischen zwei Datenmodellen auf konzeptioneller und logischer Ebene. In: *3. deutsche Arc/Info-Anwender-Konferenz*. Freising, März 1995

[Wal05] Walz, Steffen P.: Deligthful Identification Persuasion: Towards an Analytical and Applied Rhetoric of Digital Games. In: *Works and Days. Special Issue: Capitalizing on Play: The Politics of Computer Gaming* Bd. 22. Indiana : Indiana University of Pennsylvania, 2005, S. 185–200

[Wan00] Want, Roy: Remembering Mark Weiser: Chief Technologist, Xerox PARC. In: *IEEE Personal Communications* (2000), Februar, S. 8–10

[WJH97] Ward, A. ; Jones, A. ; Hopper, A.: A New Location Technique for the Active Office. In: *IEEE Personal Communications* 4 (1997), Oktober, Nr. 5, S. 42–47

[Wol07] Wolff, Eberhard: *Spring 2. Frameworks für die Java-Entwicklung.* 2. Auflage. Heidelberg : dpunkt.verlag, 2007

[Xia07] Xiao, Yang: *Security in Distributed, Grid, Mobile, and Pervasive Computing.* Auerbach Publications, 2007. – ISBN 0–849–37921–0

[YAL+05] Yang, Laurence T. (Hrsg.) ; Amamiya, Makoto (Hrsg.) ; Liu, Zhen (Hrsg.) ; Guo, Minyi (Hrsg.) ; Rammig, Franz J. (Hrsg.): *Embedded and Ubiquitous Computing - EUC 2005, International Conference EUC 2005, Nagasaki, Japan, December 6-9, 2005, Proceedings.* Bd. *3824*. Springer, 2005 (Lecture Notes in Computer Science). – ISBN 3–540–30807–5

[Zim07] Zimmer, Tobias H.: *Verbesserung der Kontexterkennung in Ubiquitären Informationsumgebungen*, Carl-Friedrich-Gauß Fakultät der Technischen Universität Carolo-Wilhelmina zu Braunschweig, Diss., April 2007

[Zog02] Zogg, Jean-Marie: *Telemetrie mit GSM /SMS und GPS-Einführung.* Franzis Verlag, 2002. – ISBN 3–772–35776–8

[Zog06] Zogg, Jean-Marie: *Grundlagen der Satellitennavigation*, 2006. `www.u-blox.de/customersupport/docs/GPS_Basics(GPS-X-01006).pdf`

Internet-Quellen

[1] Ainsworth, BE: *The Compendium of Physical Activities Tracking Guide*. http://prevention.sph.sc.edu/tools/docs/documents_compendium.pdf. Version: Januar 2002. – Zugriffsdatum: 28.09.2008

[2] Baddeley, Glenn: *GPS - NMEA sentence information*. http://aprs.gids.nl/nmea/. Version: Juli 2001. – Zugriffsdatum: 21.05.2008

[3] Briele, Marc: *Testumgebung WLAN-Lokalisierung Nürnberg: Vor-Ort-Information und ortsabhängige Dienste mit starken Partnern*. http://www.iis.fraunhofer.de/pr/Presse/Presseinformationen_2008/WLAN_Testumgebung.jsp. Version: Januar 2008. – Zugriffsdatum: 06.07.2008

[4] Eberl, Jürgen: *connect: Navigation: So funktioniert AGPS*. http://www.connect.de/themen_spezial/Standortvorteil_4502071.html. Version: September 2008. – Zugriffsdatum: 28.09.2008

[5] Foundation, The Apache S.: *Apache Tomcat*. http://tomcat.apache.org/. Version: 2008. – Zugriffsdatum: 20.04.2008

[6] Foundation, The Apache S.: *Cocoon 2.2 Site – Overview*. http://cocoon.apache.org/2.2/. Version: Mai 2008. – Zugriffsdatum: 16.09.2008

[7] Foundation, The Apache S.: *WebServices – Axis*. http://ws.apache.org/axis/. Version: April 2008. – Zugriffsdatum: 21.08.2008

[8] Foundation, The E.: *Eclipse.org home*. http://www.eclipse.org. Version: 2008. – Zugriffsdatum: 14.04.2008

[9] GmbH, Novafeel: *Berechnung des Leistungsumsatzes mit MET = metabolisches Äquivalent*. http://www.novafeel.de/ernaehrung/met-metabolische-aequivalent.htm. Version: 2008. – Zugriffsdatum: 28.09.2008

[10] Grant, Lyndsay ; Tim Rudd, Hans D.: *Mobile, collaborative and location-based learning: a case study of the MobiMissions prototype.* http://www.futurelab.org.uk/resources/documents/project_reports/MobiMissions_research_report.pdf. Version: 2007. – Zugriffsdatum: 10.07.2008

[11] Inc., Apple: *Cocoa Fundamentals Guide: The Model-View-Controller Design Pattern.* http://developer.apple.com/documentation/Cocoa/Conceptual/CocoaFundamentals/CocoaDesignPatterns/chapter_5_section_4.html. Version: Oktober 2007. – Zugriffsdatum: 26.09.2008

[12] Inc., Apple: *Apple Developer Connection – Developer Tools and Technologies – Download Xcode.* http://developer.apple.com/technology/xcode.html. Version: 2008. – Zugriffsdatum: 13.06.2008

[13] Inc., Apple: *Apple Human Interface Guidelines.* http://developer.apple.com/documentation/UserExperience/Conceptual/AppleHIGuidelines/index.html. Version: Juni 2008. – Zugriffsdatum: 20.09.2008

[14] Inc., Apple: *iPhone Dev Center.* http://developer.apple.com/iphone/. Version: 2008. – Zugriffsdatum: 08.10.2008

[15] Inc., Blizzard E.: *World Of Warcraft zählt jetzt mehr als 11 Millionen Abonnenten weltweit.* http://eu.blizzard.com/de/press/081028.html. Version: Oktober 2008. – Zugriffsdatum: 28.10.2008

[16] Inc., Red H.: *Hibernate – Relational Persistence for Java and .NET.* http://www.hibernate.org/. Version: 2008. – Zugriffsdatum: 15.08.2008

[17] Inc., SpringSource: *Springframework.org.* http://springframework.org/. Version: 2008. – Zugriffsdatum: 05.05.2008

[18] Köhne, Anja ; Wössner, Michael: *kowoma: Wie funktioniert GPS. Alles Wissenswerte.* http://www.kowoma.de/gps/. Version: 2008. – Zugriffsdatum: 21.05.2008

[19] Kremp, Matthias: *Positionsbestimmung per W-Lan: WPS statt GPS.* http://www.spiegel.de/netzwelt/tech/0,1518,528797-2,00.html. Version: Januar 2008. – Zugriffsdatum: 18.05.2008

[20] Ltd., Garmin: *Forerunner 305.* https://buy.garmin.com/shop/shop.do?cID=142&pID=349. Version: 2008. – Zugriffsdatum: 29.09.2008

[21] Marathon, Dresdner Kleinwort F.: *Dresdner Kleinwort Frankfurt Marathon.* http://www.frankfurt-marathon.com/. Version: Oktober 2008. – Zugriffsdatum: 26.10.2008

[22] Meyer, Steffen: *Bodenständig Positionsbestimmung per WLAN statt GPS*. http://www.heise.de/mobil/Positionsbestimmung-per-WLAN-statt-GPS--/artikel/105369/0. Version: März 2008. – Zugriffsdatum: 18.05.2008

[23] MySQL, AB: *MySQL Community Server*. http://dev.mysql.com/downloads/mysql/5.0.html. Version: 2008. – Zugriffsdatum: 22.04.2008

[24] MySQL, AB: *MySQL Connector/J 5.0*. http://dev.mysql.com/downloads/connector/j/5.0.html. Version: 2008. – Zugriffsdatum: 22.04.2008

[25] Rothacher, Markus: *The CHAMP Mission*. http://www.gfz-potsdam.de/champ/. Version: September 2007. – Zugriffsdatum: 21.05.2008

[26] Svahn, Matthias ; Benford, Steve ; Goetcherian, Vartkes: *IPerG - Integrated Project of Pervasive Games*. http://www.pervasive-gaming.org/. Version: 2008. – Zugriffsdatum: 20.05.2008

[27] Tandavanitj, Nicholas: *A mutual friend - Hitchers*. http://www.amutualfriend.co.uk/html/hitchers_about.html. Version: Januar 2005. – Zugriffsdatum: 10.07.2008

[28] Walz, Steffen P.: *Research / REXplorer*. http://wiki.caad.arch.ethz.ch/Research/REXplorer. Version: 2007. – Zugriffsdatum: 10.07.2008

[29] Weiser, Mark: *The Computer for the 21st Century*. http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html. Version: September 1991. – Zugriffsdatum: 20.05.2008

[30] Wireless, Skyhook: *Skyhook Wireless Announces XPS 2.0 - the Most Advanced Hybrid Positioning System*. http://www.skyhookwireless.com/press/skyhookxps.php. Version: Juni 2008. – Zugriffsdatum: 07.07.2008

[31] Wireless, Skyhook: *SKYHOOK Wireless: Application Developers - Overview*. http://www.skyhookwireless.com/developers/. Version: 2008. – Zugriffsdatum: 17.05.2008

[32] Wireless, Skyhook: *SKYHOOK Wireless Wi-Fi Positioning System (WPS)*. http://www.skyhookwireless.com/. Version: 2008. – Zugriffsdatum: 17.05.2008

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Abkürzungsverzeichnis

AOA	Angle of Arrival
API	Application Programming Interface
COO	Cell of Origin
DAO	Data Access Object
GPS	Global Positioning System
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HDOP	Horizontal Dilution of Precision
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSF	JavaServer Faces
JSP	JavaServer Pages
JSR	Java Specification Request
LBS	Location Based Service
MAC	Media Access Control
MVC	Model View Control
PDA	Personal Digital Assistant
PRN	Pseudo Random Noise
RDBMS	Relationales Datenbankmanagementsystem

RFID	Radio Frequency Identification
SDK	Software Development Kit
TDOA	Time Difference of Arrival
TOA	Time of Arrival
UML	Unified Modelling Language
UMTS	Universal Mobile Telecommunications System
WLAN	Wireless Local Area Network
WPS	Wi-Fi Positioning System
XML	eXtensible Markup Language

Glossar

Aktivitatserkennungsrate	Die Aktivitatserkennungsrate gibt das durchschnittliche Verhaltnis zwischen korrekt erkannten Aktivitaten und der Anzahl der im System vorhandenen, verschiedenen Aktivitaten an, 113
GPS	Bei dem sogenannten *Global Positioning System* handelt es sich um ein satellitengestütztes Verfahren zur weltweiten Positionsbestimmung. Es wird vom amerikanischen Militär betrieben und kann von zivilen und militärischen Anwendern genutzt werden., 13
Kontext	Kontext bezeichnet die Einheit aus Kontextdaten und Kontextattributen. Des Weiteren besitzt Kontext einen Typ, der durch die enthaltenen Kontextdaten festgelegt wird, 28
Kontextaggregation	Die Aggregation von Daten bezeichnet die algorithmische Verknupfung von verschiedenartigen Daten zum Zweck der Gewinnung neuer Erkenntnisse, 35
Kontextattribut	Ein Kontextattribut ist eine Eigenschaft eines Sensordatums, die von dessen semantischer Bedeutung unabhangig ist, 105
Kontextfusion	Die Fusion von Daten bezeichnet die algorithmische Verknupfung gleichartiger Daten zum Zweck der Genauigkeitsverbesserung oder Fehlerreduktion, 36

Kontextsensitivität	Ein System oder eine Anwendung ist kontextsensitiv (context-awareness), wenn es Kontextinformationen verarbeitet und sein/ihr inneres und/oder außeres Verhalten aufgrund der verarbeiteten Kontextinformation andern kann, 29
Sensordaten	Sensordaten sind Daten, die ein Artefakt oder ein Infrastrukturdienst durch physikalische Umweltsensoren erfassen kann, 32

Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbständig und unter ausschließlicher Verwendung der angegebenen Literatur und Hilfsmittel erstellt zu haben.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Berlin, den 6. November 2008

Eileen Kühn