Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Jan-Peter Tutzschke

Reduktion der Komplexität in pervasiven Spielen -
Konzeption und Realisierung eines Rahmenwerks

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Jan-Peter Tutzschke

Reduktion der Komplexität in pervasiven Spielen - Konzeption und Realisierung eines Rahmenwerks

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

Betreuender Prüfer: Prof. Dr. Olaf Zukunft
Zweitgutachter: Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 5. Dezember 2008

Jan-Peter Tutzschke

Thema der Masterarbeit

Reduktion der Komplexität in pervasiven Spielen - Konzeption und Realisierung eines Rahmenwerks

Stichworte

Pervasives Rechnen, kontextbewusstes Rechnen, pervasives Spielen, Middleware, Rahmenwerk, ortsabhängige Spiele, mobile verteilte Systeme

Kurzzusammenfassung

Pervasive Spiele ermöglichen einem Spieler neue Spielerfahrungen durch die Anreicherung von traditionellen Spielen in der Realität mit Informations- und Kommunikationstechnologien. Dadurch wird eine Brücke zwischen der virtuellen und der physikalischen Welt geschlagen und das Spiel mit der Realität verwoben. Als mobiles verteiltes System mit kontextbewussten Eigenschaften und einer jederzeit verfügbaren Spielwelt ergeben sich im Gegensatz zu traditionellen mobilen Spielen neue Herausforderungen, die die Komplexität bei der Entwicklung erhöhen. Um die Komplexität im Rahmen der Entwicklung zu reduzieren, wird ein Rahmenwerk für einen wiederverwendbaren Entwurf auf der Grundlage eines konkreten Szenarios und einer anschließenden Analyse entwickelt und prototypisch realisiert.

Title of the Masterthesis

Reduction of complexity in pervasive games - design and implementation of a framework

Keywords

Pervasive computing, context-aware computing, pervasive gaming, middleware, framework, location-based games, mobile distributed systems

Abstract

Pervasive games allow new game experiences for the player through the enrichment of traditional games in the reality by integration of information and communication technologies. This bridges the virtual and the physical world and interweaves the game with the reality. As a mobile distributed system which is context-aware and which game world is available at anytime, there arise in contrast to traditional mobile games new challenges which increase the complexity of the development. To reduce this complexity of development a framework for reusable design will be developed based on a specified scenario and a subsequent analysis and afterwards prototypically realized.

Inhaltsverzeichnis

1. Einleitung

Pervasive Spiele ermöglichen neue Spielerfahrungen durch die Anreicherung von traditionellen Spielen in der Realität mit Informations- und Kommunikationstechnologien. Im Gegensatz zu klassischen Computerspielen, die in einer rein virtuellen Welt stattfinden, sind bei pervasiven Spielen physikalische Bewegungen und soziale Interaktionen mit anderen Benutzern in der physikalischen Welt notwendig (Magerkurth u. a., 2005). Dadurch wird eine Brücke zwischen der virtuellen und der physikalischen Welt geschlagen (Schneider und Kortuem, 2001) und das Spiel mit der Realität verwoben (Walther, 2005b).

1.1. Motivation

Eine Möglichkeit, traditionelle Spiele in der Realität mit Informations- und Kommunikationstechnologien anzureichern, kann durch den Einsatz von vernetzten modernen mobilen Endgeräten, wie z.B. modernen Mobiltelefonen, erreicht werden. Sensoren sammeln Informationen über den Umgebungskontext, wie z.B. die aktuelle Position des Spielers, und lassen diese Informationen dann in das Spiel einfließen. Die Spielerfahrung ist also demnach abhängig von dem aktuellen Kontext und den Aktionen der einzelnen Spieler (Walther, 2005b).

Neben den neuen Spielerfahrungen, die sich aus der Vermischung der realen mit der virtuellen Welt ergeben, erhöht sich auch die Komplexität bei der Entwicklung von pervasiven Spielen im Gegensatz zu klassischen Computerspielen durch neue technische, soziale und kulturelle Herausforderungen und Problemstellungen. In einem pervasiven Spiel bilden moderne mobile Endgeräte ein mobiles verteiltes System, das sich durch einen hohen Grad an Mobilität und Heterogenität auszeichnet. Die Spieler sind nach dem Paradigma des pervasiven Rechnens jederzeit miteinander vernetzt und haben Zugriff auf das Spiel. Gleichzeitig werden die Umgebungsinformationen der Spieler gesammelt, in Abhängigkeit des Spielkontexts verarbeitet, und an die Spieler bei Bedarf wieder kontextbewusst verteilt. Dabei müssen die Spielregeln eingehalten und Betrugsversuche identifiziert werden.

Im Rahmen des Entwicklungsprozesses von pervasiven Spielen werden aufgrund der aufgeführten Herausforderungen und der aufgezeigten Problemfelder Konzepte und Hilfsmittel benötigt, die eine zügige Realisierung und Validierung von neuen Spielideen und -konzepten ermöglichen und gleichzeitig die Komplexität bei der Entwicklung reduzieren.

1.2. Ziele

Aus der Motivation heraus, Spiele in der Realität mit virtuellen und pervasiven Elementen anzureichern und der Grundlage der technischen Möglichkeiten von modernen leistungsstarken mobilen Endgeräten, soll im Rahmen dieser Arbeit ein Rahmenwerk für die Entwicklung von pervasiven Spielen entwickelt werden, um die Validierung von neuen Spielideen und -konzepten zu erleichtern und die Komplexität bei der Entwicklung zu verringern.

Ausgehend von einer grundlegenden und differenzierten Betrachtung der Herausforderungen und Problemstellungen sollen auf der Grundlage eines konkreten Szenarios die elementaren funktionalen, technischen und nicht-funktionalen Anforderungen an ein Rahmenwerk für pervasive Spiele durch eine Analyse identifiziert und für einen konkreten Entwurf spezifiziert werden. Die Anforderungen sollen dann in einen Entwurf für ein Rahmenwerk für pervasive Spiele umgesetzt werden. Dabei soll der Fokus auf einen ganzheitlichen Entwurf gelegt werden, der die Anforderungen umsetzt und damit eine erweiterbare und stabile Plattform für die Entwicklung von konkreten pervasiven Spielen bereitstellt. Um die Tragfähigkeit des Entwurfs sicherzustellen, soll der Entwurf anschließend prototypisch realisiert und abschließend durch eine Testanwendung validiert werden.

1.3. Abgrenzung

Aufgrund der identifizierten Problemstellungen und der angeführten Komplexität bei der Entwicklung von pervasiven Spielen und Rahmenwerken für pervasive Spiele wird im Rahmen dieser Arbeit ein konkretes Szenario, in Form eines pervasiven Spiels, klassifiziert und dadurch die Anforderungen an ein Rahmenwerk für pervasive Spiele eingegrenzt. Durch die Fokussierung auf den funktionalen Nutzen auf der Grundlage einer soliden technischen Plattform werden nicht-funktionale Anforderungen, wie z.B. Sicherheit, Datenschutz und Anforderungen an die Leistung, nur teilweise im Rahmen eines Entwurfs berücksichtigt.

In dem beschriebenen Problemfeld existieren bereits einige Ansätze von Rahmenwerken für pervasive Spiele, die einen ähnlichen Ansatz verfolgen. Abgrenzend zu Fetter u. a. (2007) wird in dieser Arbeit ein erweitertes fachliches Domänenmodell entwickelt und ein anderer konzeptioneller und technischer Ansatz verfolgt werden. Ebenso soll kein anwendungsspezifisches Rahmenwerk, wie es z.B. in Suomela u. a. (2004) präsentiert wird, entwickelt werden, sondern es soll der Fokus auf die funktionale Unterstützung für eine konkrete Klasse von pervasiven Spielen gelegt werden.

1.4. Gliederung

In Kapitel 2 werden die grundlegenden Begriffe aus dem Titel dieser Arbeit, wie z.B. Rahmenwerk und pervasives Spielen, aufgegriffen und differenziert betrachtet. Gleichzeitig werden Konzepte aus überschneidenden und angrenzenden Forschungsgebieten, wie z.B. allgegenwärtiges und kontextbewusstes Rechnen, angeführt und Problemstellungen identifiziert, die im weiteren Verlauf dieser Arbeit für ein Rahmenwerk für pervasive Spiele berücksichtigt werden müssen. Damit ein Rahmenwerk für pervasive Spiele entwickelt werden kann, wird in Kapitel 3 die Idee für das konkrete pervasive Spiel *King of Location* skizziert und der Spielablauf und die Spielregeln für eine spätere Analyse spezifiziert. Abschließend wird die Eignung des Szenarios untersucht und anhand der Eigenschaften eindeutig klassifiziert. Anschließend wird in Kapitel 4 das Szenario analysiert und Anwendungsfälle und Anforderungen an das Szenario identifiziert. Auf der Grundlage der Analyse des Szenarios werden die Anforderungen an ein Rahmenwerk für pervasive Spiele mit Hilfe einer Domänenanalyse abgeleitet und spezifiziert. Zum Ende des Kapitels werden vergleichbare Projekte und mögliche Lösungsansätze für Teillösungen vorgestellt und mit den identifizierten Anforderungen abgeglichen. In Kapitel 5 wird anschließend ein Entwurf für ein Rahmenwerk für pervasive Spiele vorgestellt. Im Rahmen dieser Vorstellung wird auf Problemstellungen eingegangen und Entwurfsentscheidungen getroffen. Durch die vorgestellte prototypische Realisierung wird in Kapitel 6 die Tragfähigkeit des vorgestellten Entwurfs bewiesen, ein Überblick über den Realisierungsumfang gegeben und Details der Realisierung angesprochen. In Kapitel 7 werden dann die Anforderungen mit dem entwickelten Entwurf und der prototypischen Realisierung abgeglichen und es wird eine kritische Betrachtung und eine methodische Abstraktion durchgeführt. Abschließend fasst Kapitel 8 die wesentlichen Aspekte dieser Arbeit zusammen und gibt einen Ausblick auf mögliche Ansatzpunkte für weitere Arbeiten.

2. Grundlagen

In diesem Kapitel soll ein grundlegendes Verständnis für die konkrete Problemstellung und das Problemfeld des pervasiven Spielens geschaffen werden. Da ein Rahmenwerk für pervasives Spielen entwickelt werden soll, wird in Abschnitt 2.1 zunächst ein Überblick über die Funktionen und die Eigenschaften von Rahmenwerken gegeben und ein Bezug zu anderen Wiederverwendungsmöglichkeiten hergestellt. Da es sich sich bei pervasiven Spielen um verteilte Systeme handelt, werden in Abschnitt 2.2 Middlewarekonzepte für die Kommunikation in verteilten Systemen vorgestellt. Anschließend werden kontextbewusste Systeme betrachtet, die ihr Verhalten an die aktuelle Umgebung anpassen und/oder aber auf Basis von Umgebungsinformationen Informationen für den Benutzer bereitstellen können (Abschnitt 2.3). In Abschnitt 2.3.1 werden dann Herausforderungen bei der Entwicklung von kontextbewussten Systemen präsentiert. Damit ein Rahmenwerk für pervasives Spielen entwickelt werden kann, wird anschließend geklärt, was unter allgegenwärtigem und pervasivem Rechnen verstanden wird (Abschnitt 2.4) und welche Herausforderungen in diesem Umfeld existieren, die bei der Entwicklung beachtet werden müssen (Abschnitt 2.4.1). Pervasives Spielen wird dann als Teildisziplin des pervasiven Rechnens untersucht (Abschnitt 2.5). Anschließend werden die Anforderungen an Spiele im Allgemeinen (Abschnitt 2.5.1) betrachtet, ortsabhängige Spiele als eine Klasse von pervasiven Spielen (Abschnitt 2.5.2) vorgestellt und klassifiziert und anschließend die Einsatzmöglichkeiten von pervasiven Elementen in Spielen betrachtet (Abschnitt 2.5.3). Zum Abschluss werden dann einige pervasive Spiele beispielhaft vorgestellt (Abschnitt 2.5.4) und die wichtigsten Aussagen zusammengefasst (Abschnitt 2.6).

2.1. Rahmenwerk

Ein Rahmenwerk (engl. framework) ist eine Menge von Klassen, die einen wiederverwendbaren Entwurf für eine Klasse von Anwendungen darstellen (Johnson und Foote, 1988). Dabei wird die Architektur einer Anwendung durch das Rahmenwerk vorgegeben und dadurch die grundlegende Struktur, durch die Unterteilung in Klassen und Objekte, sowie deren Kontrollfluss definiert (Gamma u. a., 1994). Nach Johnson (1997) wird durch diese Definition der Aufbau von Rahmenwerken fokussiert. Im Gegensatz dazu existiert auch eine Definition aus einer funktionalen Perspektive. Ein Rahmenwerk kann somit also auch

als ein Skelett einer Anwendung verstanden werden, das von einem Anwendungsentwickler spezifisch angepasst werden kann (Johnson und Foote, 1988). Eine sehr abstrakte und allgemeine Definition wird von Szyperski (2002) vorgeschlagen, die ein Rahmenwerk als einen Teil der Architektur eines Systems definieren. In dieser Arbeit wird Rahmenwerk nach der Definition von Johnson und Foote (1988) verstanden und wie folgt definiert (Definition 1):

> **Definition 1 [Rahmenwerk]**
> *Ein Rahmenwerk ist eine Menge von Klassen, die einen wiederverwendbaren Entwurf für eine Klasse von Anwendungen darstellen.*

Die Entwicklung von Rahmenwerken erfolgt meist iterativ durch die Integration von neuen Anforderungen und durch die Konsolidierung im Einsatz (Fach, 2001).

Durch den Einsatz von Rahmenwerken können sich Anwendungsentwickler auf die Problemdomäne konzentrieren. Zusätzlich geben Rahmenwerke die Entwurfsentscheidung für den Anwendungsbereich vor. Der Einsatz von Rahmenwerken hat demnach in der Entwicklung folgende Vorteile (Fayad und Schmidt, 1997):

Modularität - Rahmenwerke erhöhen durch die Kapselung von Implementierungsdetails die Modularität hinter stabilen Schnittstellen. Durch die Modularität wird die Qualität der Software gesteigert, da die Auswirkungen von Änderungen am Design und der Implementierung besser erkannt werden können.

Wiederverwendbarkeit - Die von Rahmenwerken bereitgestellten Schnittstellen erhöhen die Wiederverwendbarkeit durch die Definition von generischen Komponenten, die in unterschiedlichen Anwendungen eingesetzt werden können. Die Wiederverwendbarkeit in Rahmenwerken steigert gleichzeitig die domänenspezifischen Kenntnisse, da wiederkehrende Problemstellungen und Anwendungsanforderungen generisch durch das Rahmenwerk gelöst werden. Die Wiederverwendung von Komponenten aus einem Rahmenwerk kann somit die Produktivität der Entwickler erhöhen und gleichzeitig die Qualität, die Geschwindigkeit, die Verlässlichkeit und die Interoperabilität der Software steigern.

Erweiterbarkeit - Ein Rahmenwerk verbessert die Erweiterbarkeit durch die Bereitstellung von expliziten Einstiegspunkten (Pree, 1994), die es Anwendungen erlauben, die stabilen Schnittstellen des Rahmenwerks zu erweitern. Die Einstiegspunkte entkoppeln systematisch die stabilen Schnittstellen und das Verhalten einer Anwendungsdomäne von der konkreten Implementierung des Rahmenwerks. Die Erweiterbarkeit von Rahmenwerken ermöglicht die zeitnahe Integration von neuen Anforderungen in eine bestehende Anwendung.

Umkehrung der Steuerung - Die Architektur zur Laufzeit wird durch die Umkehrung der Steuerung charakterisiert. Das Rahmenwerk ist für die Ablaufsteuerung verantwortlich und führt bei auftretenden Ereignissen die über die Einstiegspunkte integrierten anwendungsspezifischen Komponenten aus. Die Ablaufsteuerung wird somit nicht von der Anwendung vorgegeben, sondern durch das Rahmenwerk festgelegt.

Neben den aufgeführten Vorteilen wurden in der Praxis einige Nachteile identifiziert, die sich in vier Kategorien einteilen lassen (Bosch u. a., 2000):

Entwicklung - Im Gegensatz zur Entwicklung von Anwendungen, die die gestellten Anforderungen erfüllen müssen, müssen bei der Entwicklung von Rahmenwerken alle relevanten Konzepte innerhalb der Anwendungsdomäne abgedeckt werden. Die Korrektheit des abstrakten Verhaltens von Rahmenwerken muss sichergestellt werden, auch wenn vollständige Tests erst bei der Instanziierung möglich sind. Bei der Veröffentlichung von Rahmenwerken müssen Kriterien wie Stabilität, Wiederverwendbarkeit, Flexibilität und Dokumentation sichergestellt werden, obwohl z.B. für die Dokumentation keine allgemein akzeptierte Methode existiert, die alle Aspekte in einem Rahmenwerk umfasst.

Verwendung - Der Anwendungsentwickler muss bei dem Einsatz von Rahmenwerken entscheiden, ob das Rahmenwerk für die Anforderungen und die Problemdomäne geeignet ist, um den Entwicklungsaufwand, die Menge des anwendungsspezifischen Quellcodes und den Aufwand für die Fehlersuche und -beseitigung abschätzen zu können.

Komposition - Wenn Anwendungen mehrere Rahmenwerke gleichzeitig einsetzen, können bei der Entwicklung Probleme auftreten, da z.B. die Architekturen der Rahmenwerke nicht kompatibel sind, die Entitäten der Rahmenwerke sich überschneiden, die Ablaufsteuerung der Rahmenwerke nicht vereinbar sind, die Integration von bestehenden Komponenten nicht möglich ist oder aber durch die Komposition der Rahmenwerke nicht alle Anforderungen der Anwendungen abgedeckt werden.

Wartung - Durch die Weiterentwicklung von Anwendungsdomänen müssen Rahmenwerke stetig an neue Anforderungen angepasst und gewartet werden, die Änderungen in der Anwendungsdomäne berücksichtigen und die richtige Strategie für die Wartung wählen.

Bei der Wiederverwendung von Rahmenwerken können noch weitere Probleme auftreten, die sich durch die Anwendung ergeben und die sich in vier Kategorien einteilen lassen (Kirk u. a. (2005) und Kirk u. a. (2007)):

Funktionalität - Das Verständnis für die Funktionalität des Rahmenwerks fehlt in einzelnen Bereichen, so dass nicht eindeutig klar ist, wie das Rahmenwerk genau funktioniert und welche Funktionalität abgedeckt wird. Das kann dazu führen, dass Anwendungsentwickler eine angebotene Funktionalität des Rahmenwerks übersehen oder aber falsch verwenden.

Interaktion - Ein weiteres Problem ist das Verständnis für die Kommunikation zwischen den Klassen innerhalb eines Rahmenwerks. Das Problem tritt durch versteckte Abhängigkeiten innerhalb des Rahmenwerks auf, die durch die charakteristischen Eigenschaften von Rahmenwerken, wie z.B. Vererbung, Bindung zur Laufzeit oder die Umkehrung der Ausführung entstehen. Anwendungsentwickler können z.B. Änderungen am Programmcode nicht korrekt in die Architektur einordnen und umgehen unwissentlich wichtige Funktionalität des Rahmenwerks. Daraus resultierende Fehler werden erst in einer späteren Entwicklungsphase entdeckt, wenn der modifizierte Programmcode ausgeführt wird. Das kann unabsichtlich zu Änderungen im Rahmenwerk führen oder im Extremfall das Rahmenwerk zum Absturz bringen.

Mapping - Das Problem ist das Überführen einer abstrakten und konzeptionellen Lösung in eine konkrete Implementierung, die das Rahmenwerk mit seinen Möglichkeiten und Limitierungen nutzt. Das Problem tritt auf, wenn Anwendungsentwickler Schwierigkeiten haben, eine Lösung mit Hilfe der verfügbaren Möglichkeiten des Rahmenwerks zu realisieren. In diesen Fällen besteht ein Unterschied zwischen der Funktionalität des Rahmenwerks und der Funktionalität, die der Anwendungsentwickler erwartet.

Architektur - Modifikationen am Rahmenwerk können über einen längeren Zeitraum dazu führen, dass die Flexibilität des Rahmenwerks durch mangelndes Verständnis der Architektur verloren geht.

Ähnliche Probleme werden auch von Mattsson u. a. (1999) identifiziert und beschrieben. Als Lösung für die aufgeführten Probleme wird von Mattsson u. a. (1999), Kirk u. a. (2005) und Kirk u. a. (2007) der Einsatz geeigneter Dokumentationstechniken vorgeschlagen, für die aber bisher keine allgemeingültigen Ansätze existieren.

Neben den Eigenschaften und Vor- und Nachteilen können Rahmenwerke nach ihrem Anwendungsbereich in anwendungsspezifische und domänenspezifische Rahmenwerke und Rahmenwerke zur Unterstützung klassifiziert werden (Fayad und Schmidt (1997) und Fayad u. a. (1999)):

Anwendungsspezifische Rahmenwerke erleichtern die Entwicklung von portierbaren und effizienten Systeminfrastrukturen wie Betriebssystem- und Kommunikations-Rahmenwerke und Rahmenwerke für Benutzerschnittstellen. Anwendungsspezifische Rahmenwerke werden hauptsächlich innerhalb einer Organisation in der Entwicklung eingesetzt.

Rahmenwerke zur Unterstützung werden für die Integration von verteilten Anwendungen eingesetzt. Middleware-Integrations Rahmenwerke ermöglichen die einfache Integration in einer verteilten Umgebung und helfen dadurch dem Anwendungsentwickler, die Anwendungsinfrastruktur zu modularisieren, wiederzuverwenden und zu erweitern (z.B. nachrichtenorientierte Middleware (siehe Abschnitt 2.2)).

Domänenspezifische Rahmenwerke behandeln breite Anwendungsbereiche (wie z.B. Telekommunikation, Luftfahrt oder Versicherungen) und bilden das Fundament der Unternehmensaktivitäten. Im Gegensatz zu anwendungsspezifischen Rahmenwerken und Rahmenwerken zur Unterstützung ermöglichen domänenspezifische Rahmenwerke eine direkte Unterstützung bei der Entwicklung von Anwendungen für den Endbenutzer, sind jedoch häufig teurer in der Entwicklung oder in der Beschaffung.

Innerhalb von Rahmenwerken können zusätzlich unterschiedliche Bereiche identifiziert werden, die entweder statisch oder veränderbar (engl. frozen and hot spots) ausgeprägt sein können (Pree (1994), Szyperski (2002)):

Statische Bereiche - Die statischen Bereiche geben die Architektur einer Anwendung vor und bilden Aspekte ab, die in allen Anwendungen in einer Domäne vorhanden sind.

Veränderbare Bereiche - Die dynamischen Bereiche eines Rahmenwerks werden durch anwendungsspezifisches Verhalten ergänzt und sind nur für einen Teil der Anwendungen innerhalb der Domäne spezifisch.

Nach Matos und Fernandes (2006) besteht bei der Entwicklung von domänenspezifischen Rahmenwerke das Problem beim Klassifizieren der statischen und veränderbaren Bereiche.

Neben der Klassifikation nach Anwendungsbereichen können Rahmenwerke auch nach der Art der Nutzung in „White-Box“- und „Black-Box“- Rahmenwerke unterschieden werden (Johnson und Foote, 1988):

White-Box Rahmenwerke - Bei White-Box Rahmenwerken wird das anwendungsspezifische Verhalten durch die Implementierung von Methoden in den Unterklassen des Rahmenwerks definiert. Daher werden vorgegebene abstrakte Klassen aus dem Rahmenwerk anwendungsspezifisch spezialisiert. Das erfordert das Verständnis für die internen Strukturen des Rahmenwerkes.

Black-Box Rahmenwerke - Bei Black-Box-Rahmenwerken ist das anwendungsspezifische Verhalten in den Komponenten des Rahmenwerkes bereits enthalten. Die Komponenten von Black-Box Rahmenwerken können daher direkt eingesetzt werden und benötigen nur das Verständnis für das Protokoll zwischen den öffentlichen Schnittstellen der Komponenten des Rahmenwerks. Black-Box Rahmenwerke benötigen daher eine kürzere Einarbeitungszeit, sind aber im Gegensatz zu White-Box-Rahmenwerken nicht so flexibel und anpassbar.

Da Rahmenwerke ein Konzept der Wiederverwendung darstellen, sind Rahmenwerke mit anderen Konzepten der Wiederverwendung wie z.B. Klassenbibliotheken, Entwurfsmustern und Komponenten verwandt (Fayad und Schmidt, 1997):

Entwurfsmuster repräsentieren wiederkehrende Lösungen von Problemen in der Anwendungsentwicklung. Im Gegensatz zu Entwurfsmustern, die eine Wiederverwendung von abstrakten Entwürfen und Mikro-Architekturen fokussieren, fokussieren Rahmenwerke die Wiederverwendung von konkreten Entwürfen, Algorithmen und Implementierungen in einer spezifischen Programmiersprache. Rahmenwerke können als eine konkrete Instanz von Familien von Entwurfsmustern für eine bestimmte Problemstellung gesehen werden, wohingegen Entwurfsmuster die Architekturelemente bereitstellen und damit die Struktur eines Rahmenwerks beeinflussen können.

Klassenbibliotheken sind im Gegensatz zu Rahmenwerken weniger anwendungsspezifisch und nur für bestimmte Anwendungsbereiche einsetzbar. Gleichzeitig verhalten sich Klassenbibliotheken passiv, da sie im Gegensatz zu Rahmenwerken aktiv aufgerufen werden müssen und nicht den Kontrollfluss der Anwendung vorgeben und aktiv gestalten. Rahmenwerke können jedoch Klassebibliotheken benutzen, um dadurch die Entwicklung des Rahmenwerks zu vereinfachen.

Komponenten sind abgeschlossene Instanzen von abstrakten Datentypen, die zu vollständigen Anwendungen zusammengefügt werden können. Bei Komponenten handelt es sich um ein geschlossenes System unter Vernachlässigung des inneren Aufbaus (engl. blackbox), die nur durch die Kenntnis der Syntax und Semantik der öffentlichen Schnittstelle wiederverwendet werden können. Im Vergleich zu Rahmenwerken sind Komponenten weniger eng gekoppelt. Komponenten können z.B. verwendet werden, ohne von Klassen ableiten zu müssen. Rahmenwerke können für die Entwicklung von Komponenten eingesetzt werden, wobei die Schnittstellen der Komponenten eine Façade für die interne Klassenstruktur des Rahmenwerks darstellen. Genauso können Komponenten als austauschbare Strategien in Black-Box Rahmenwerken verwendet werden. Allgemein werden Rahmenwerke häufig eingesetzt, um die Entwicklung von Infrastruktur- und Middleware-Software zu vereinfachen, wogegen Komponenten häufiger bei der Entwicklung von Anwendungen für den Endbenutzer genutzt werden.

Gamma u. a. (1994) grenzen ebenso wie Fayad und Schmidt (1997) Rahmenwerke explizit von Klassenbibliotheken und Entwurfsmustern ab. Für Gamma u. a. (1994) werden die wieder zu verwendenen Teile einer Klassenbibliothek z.B. aktiv von der zu entwickelnden Anwendung aufgerufen, wohingegen der Programmcode beim Einsatz von Rahmenwerken durch das Rahmenwerk aufgerufen wird. Balzert (1996) differenziert zwischen Klassenbibliotheken und Rahmenwerken, indem Rahmenwerke den Entwickler dazu zwingen „die Architektur der eigenen Anwendung in die Bibliothek einzupassen. . . “.

2.2. Middleware

Middleware ist eine Kategorie von Software zwischen dem Betriebssystem und den Anwendungen und deckt Querschnittsfunktionalität ab, wie z.B. Kommunikation, Synchronisation und Koordination, die von vielen Anwendungen benötigt werden (Szyperski, 2002). Die Abstraktion, die von Middleware Systemen bereitgestellt wird, verbirgt die Heterogenität in einer verteilten Umgebung, unterstützt die Koordination von verteilten Entitäten und ermöglicht eine transparente verteilte Berechnung (Issarny u. a., 2007).

Es existieren viele Definitionen, die den Begriff Middleware aus unterschiedlichen Perspektiven beschreiben. Nach Bernstein (1996) ist eine Middleware ein allgemeiner Dienst, der zwischen der Plattform und den Anwendungen angesiedelt ist. Etwas spezifischer definiert Szyperski (2002) Middleware als eine Menge von Software, die zwischen verschiedenen Betriebssystemen und einer höheren Programmierplattform angesiedelt ist. Eine ähnliche, jedoch eher technische Definition wird von Hong und Landay (2001) gegeben, die Middleware als eine Infrastruktur beschreiben, die aus bewährten, überall vorhandenen, verlässlichen und öffentlichen Technologien besteht und als Grundlage für andere Systeme eingesetzt wird. Nach Tanenbaum und van Steen (2003) ist eine Middleware eine Anwendung, die sich logisch auf der Anwendungsschicht befindet, jedoch viele allgemeine Protokolle enthält, die ihre eigenen Schichten rechtfertigen, unabhängig von anderen, spezifischeren Anwendungen. Linthicum (2004) dagegen definiert Middleware über die Funktion. Demnach ist eine Middleware eine Art von Software, die eine Kommunikation zwischen zwei oder mehr Software-Systemen ermöglicht (Linthicum, 2004). Dabei handelt es sich um eine sehr allgemein gehaltene Definition, die aber nach Linthicum (2004) nötig ist, da eine Middleware bereits Interprozesskommunikation abdeckt, aber auch z.B. JAVA RMIRemote Method Invocation (RMI)[1] oder Transaktionsmonitore umfasst. Die wichtigste Rolle dabei ist das Teilen von Informationen zwischen unterschiedlichen Anwendungen. In dieser Arbeit wird Middleware wie von Bernstein (1996) definiert verstanden (Definition 2).

> **Definition 2 [Middleware]**
> *Middleware ist ein allgemeiner Dienst, der zwischen der Plattform und den Anwendungen angesiedelt ist*

Eine Klassifikation von Middleware-Systemen kann anhand des Koordinationsmodells, das die Systeme implementieren, erfolgen (Erweiterte Klassifikation von Issarny u. a. (2007) basierend auf Emmerich (2000):

Transaktionsorientierte Middleware - Transaktionen sind Verträge, die einen konsistenten Systemzustand garantieren. Transaktionale Middleware wird in unterschiedlichen

[1] http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp

verteilten Anwendungsdomänen wie z.B. in datenbankzentrierten Anwendungen, in Telekommunikations- und in sicherheitkritischen Systemen eingesetzt. Technisch ist eine Transaktion eine Folge von festen Operationen, die eine Einheit bilden und den ACID-Eigenschaften (Atomarität, Konsistenz, Isolation, Dauerhaftigkeit) unterliegen. Transaktionale Middleware stellt eine Schnittstelle für Transaktionen zwischen verteilten Komponenten bereit, wobei die ACID-Eigenschaften dabei entsprechend den Anforderungen der Anwendungen aufgeweicht werden können.

Tupelraumbasierte Middleware - Tupelraumbasierte Middleware (engl. tuplespace-based middleware) wurde erstmals mit Linda in Gelernter (1985) eingeführt. Daten werden als Tupel in einen Raum geschrieben und können anschließend von anderen Nutzern über Suchkriterien in dem Raum gefunden und verarbeitet werden (ebenfalls bekannt als Blackboard-Architektur (Buschmann u. a., 1998)). Charakteristisch bei dem Tupelraum-Modell ist die Entkopplung von Komponenten in Raum und Zeit. Kommunizierende Komponenten müssen weder gleichzeitig aktiv sein, noch eine Referenz aufeinander haben.

Nachrichtenorientierte Middleware - Nachrichtenorientierte Middleware (engl. message-oriented middleware (MOM)) kann als eine besondere Form der tupelraumbasierten Middleware gesehen werden, bei der Tupel als Nachrichten und der Raum als verteilte Nachrichtenwarteschlangen (engl. message queue) implementiert sind. Verteilte Komponenten kommunizieren miteinander durch das Veröffentlichen, Auswählen und Lesen von Nachrichten aus der jeweiligen Nachrichtenwarteschlange. Nachrichtenorientierte Middleware kann weiterhin in wartenschlagenbasierte Middleware (Nachrichten werden anhand von Zugehörigkeit zu einer Warteschlange selektiert) und Veröffentlichen/Abbonieren Middleware (Nachrichten werden anhand von Prädikaten selektiert) kategorisiert werden.

Entfernter Prozeduraufruf - Entfernter Prozeduraufruf (engl. remote procedure call (RPC)) ist ein Protokoll, das die Ausführung von Prozeduren auf entfernten Systemen erlaubt, ohne dabei explizit die Einzelheiten der Interaktionen bei der Entwicklung zu berücksichtigen. Eine Middleware für entfernte Prozeduraufrufe stellt in diesem Kontext Möglichkeiten für die Generierung von Client/Server-Stubs, für das Marshalling und Unmarshalling der Daten und für die synchrone Kommunikation bereit und stellt zudem nicht-funktionale Eigenschaften sicher.

Objekt- und komponentenorientierte Middleware - Objekt- und komponentenorientierte Middleware ist die Weiterentwicklung von der Middleware für entfernte Prozeduraufrufe, stellt jedoch eine bessere Abstraktion für die Ausnutzung der jeweiligen Programmierparadigmen in einer verteilten Umgebung bereit. Objekt- und komponentenorientierte Middleware stellt dabei Werkzeuge bereit, die aus einer definierten Schnittstelle die dazugehörenden Stubs generieren können, die Mechanismen für die Interaktion mit entfernten Objekten oder Komponenten ermöglichen, die synchrone Kommunikation aufbauen und die Methoden oder Operationen aufrufen kann und das

Marshalling und Unmarshalling der Daten sicherstellt. Objekt- und komponentenorientierte Middleware garantiert nicht-funktionale Eigenschaften wie Verlässlichkeit, Skalierbarkeit und Sicherheit.

Dienstorientierte Middleware - Der nächste Schritt in der Evolution in der komponentenbasierten Entwicklung ist das Paradigma der dienstorientierten Architekturen (engl. service oriented architecture (SOA)), das die Entwicklung von lose gekoppelten Netzwerkdiensten in verteilten Umgebungen unterstützt (Papazoglou und Georgakopoulos, 2003). In einer dienstorientierten Architektur werden Netzwerkressourcen als autonome Dienste bereitgestellt, die ohne Kenntnisse der darunterliegenden Technologie durch definierte und standardisierte Schnittstellen verwendet werden können. Dienstorientierte Architekturen bestehen aus Dienstanbietern, die Dienste zur Verfügung stellen, Dienstnutzern, die die angebotenen Dienste verwenden, und aus Dienstverzeichnissen, die die Verwaltung von Diensten ermöglichen. Dienstorientierte Middleware (engl. service oriented middleware (SOM)) ermöglicht das Bereitstellen von Diensten und die Koordination zwischen den Elementen einer dienstorientierten Architektur. Dienstorientierte Middleware bietet Möglichkeiten für die Bereitstellung von Diensten und die Bekanntmachung bei Dienstverzeichnissen und versteckt dabei die Heterogenität in der verteilten Umgebung durch allgemeine Dienstbeschreibungssprachen und Protokolle für die Bekanntmachung und Nutzung.

Ähnlich wie Issarny u. a. (2007) werden auch Middleware Modelle nach Linthicum (2004) klassifiziert. Für Linthicum (2004) können Middleware Modelle in ein physikalisches und ein logisches Modell unterschieden werden. Das logische Modell zeigt dabei, wie Informationen konzeptionell verteilt werden. Dagegen zeigt das physikalische Modell, wie die Informationen bewegt werden und welche konkrete Technik verwendet wird.

2.3. Kontext und kontextbewusstes Rechnen

Umgebungs- oder Kontextinformationen fließen bei der Kommunikation zwischen Menschen implizit ein und erhöhen so die Gesprächsbandbreite (Abowd u. a., 1999). Damit bei der Mensch-Maschine Interaktion nach der Vision des allgegenwärtigen und pervasiven Rechnens (siehe Abschnitt 2.4) eine unsichtbare rechnergestützte Informationsverarbeitung stattfinden kann, müssen rechnergestützte Systeme Zugriff auf die aktuellen Umgebungsinformationen haben und sich ihrer Umwelt bewusst sein, um sich angemessen verhalten zu können. Der Kontextbegriff wird auch in anderen Bereichen verwendet, z.B. in der künstlichen Intelligenz (engl. artificial intelligence (AI)). Der Bereich liegt aber nicht im Fokus dieser Arbeit und wird deshalb nicht weiter vertieft. Eine Zusammenfassung von Kontext im Bereich der künstlichen Intelligenz kann z.B. in Akman und Surav (1996) gefunden werden. In dieser Arbeit bezieht sich Kontext und Kontextbewusstsein auf das adaptive Verhalten von Anwendungen im Bereich von mobilen und pervasiven Anwendungen.

Bei der Definition von Kontext gibt es bei den unterschiedlichen Forschern und Forschergruppen unterschiedliche Ansätze. Baldauf u. a. (2007) haben dabei folgende Ansätze identifiziert:

Aufzählung von Kontextbeispielen - Kontextbewusstsein als Begriff wird nach Abowd u. a. (1999) erstmals in Schilit und Theimer (1994) eingeführt und bezeichnet Kontext als die Position, die Identitäten von in der Nähe befindlichen Menschen und Objekten und die Änderungen von diesen Objekten (vgl. Schilit u. a. (1994)). In der Anfangsphase der Erforschung kontextbewusster Systeme sind Aufzählungen von Kontextbeispielen häufig zu finden (vgl. Baldauf u. a. (2007)). Dey (1998) beschreibt Kontext als den emotionalen Zustand des Benutzers, den Fokus der Aufmerksamkeit, die Position und die Ausrichtung, das Datum und die Zeit sowie die Objekte und die Menschen in der Umgebung des Benutzers.

Verwendung von Synonymen - Hull u. a. (1997) beschreiben Kontext als die Aspekte der aktuellen Situation. Für Schmidt u. a. (1999b) ist Kontext *„that which surrounds, and gives meaning to something else“*, also alles Umgebende, dass etwas anderem Sinn verleiht.

Formale Definition - Abowd u. a. (1999) definieren Kontext als jede Information, die für die Charakterisierung der Situation einer Entität verwendet werden kann, wobei eine Entität eine Person, ein Ort oder ein Objekt darstellen kann, die für die Interaktion zwischen einem Benutzer und einer Anwendung als relevant eingestuft wird. Für Schmidt u. a. (1999a) ist Kontext der Zustand des Benutzers und der informationsverarbeitenden Geräte einschließlich der Umgebung, der Situation und der Position als ein Teil davon.

Im Gegensatz zum aufzählenden Charakter der ersten Definitionsversuche sind die Definitionen auf Basis von Synonymen meistens zu allgemein gehalten und dadurch zu unkonkret (Hull u. a., 1997). Für Hull u. a. (1997) ist die formale Definition nach Abowd u. a. (1999) eine der exaktesten. Für Winograd (2001) dagegen ist eine allgemein formulierte Definition wie die von Abowd u. a. (1999) zwar legitim, hilft aber nur bedingt den Kontextbegriff allgemeingültig zu definieren. Für Winograd (2001) ist Kontext spezifischer und eher ein operativer Begriff (Zeidler, 2007): „Wenn etwas oder oder jemand abhängig von eingehenden Informationen handelt, dann sind diese Informationen Kontextinformationen“. Als Beispiel kann die Spannung einer Hochspannungsleitung angeführt werden, die Kontext sein kann, wenn eine Aktion eines Benutzers oder eines Computers davon abhängig ist, sonst ist die Hochspannungsleitung nur ein Teil der Umgebung (Winograd, 2001). Ähnlich wie Winograd (2001) definieren Lieberman und Selker (2000) Kontext als alles, was die Berechnung beeinflusst, außer die Ein- und Ausgaben selbst (siehe Abbildung 2.1).

Was nach dieser Definition als Kontext angesehen wird, wird somit durch die Grenzen des informationsverarbeitenden Systems bestimmt und hängt von den jeweiligen Anforderungen ab (Lieberman und Selker, 2000).

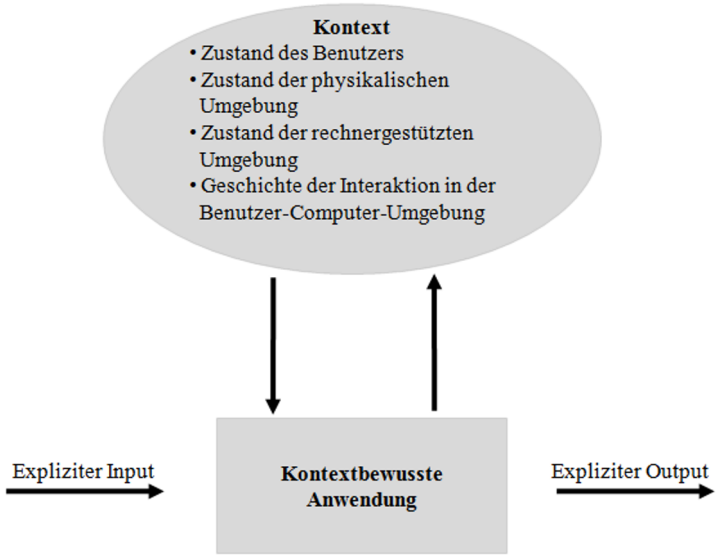

Abbildung 2.1.: Kontextdefinition nach Lieberman und Selker (2000)

Auf Basis der vorgestellten Definitionen und Erkenntnisse wird für diese Arbeit Kontext nach Lieberman und Selker (2000) und Winograd (2001) wie folgt definiert (Definition 3).

Definition 3 [Kontext]
Kontext sind alle Informationen, die die Ausführung eines informationsverarbeitenden Systems implizit oder explizit beeinflussen, außer den Ein- und Ausgaben selbst.

Die gewählte Definition macht keine Einschränkungen in der Art der Informationen, präzisiert aber gleichzeitig die Perspektive der Betrachtung.

Grundsätzlich wird noch zwischen unterschiedlichen Kontextdimensionen unterschieden, um Kontextinstanzen zu klassifizieren (vgl. Gwizdka (2000) und Prekop und Burnett (2003)):

Externer Kontext - Der externe Kontext beschreibt den Zustand der Umgebung und setzt sich aus der Position, der Entfernung zu anderen Objekten (Menschen oder Geräten) und dem zeitlichen Zusammenhang zusammen.

Interner Kontext - Der interne Kontext beschreibt den Zustand des Benutzers und setzt sich aus dem Arbeitskontext (z.B. dem Zustand der aktuellen Projekte, dem Status von zu erledigenden Tätigkeiten oder dem Projektteam), persönlichen Ereignissen,

dem Kommunikationskontext (z.B. dem Zustand der internen eMail-Kommunikation) und dem emotionalen Zustand des Benutzers zusammen.

Nach Gwizdka (2000) ist der interne Kontext dabei schwieriger zu ermitteln als der externe Kontext und kann in manchen Fällen nur aus den externen Kontextinformationen abgeleitet werden.

Einen ähnlichen Ansatz verfolgen Chen und Kotz (2000) und Nurmi und Floreen (2004), die bei Kontextinformationen zwischen Low-Level- und High-Level-Kontextinformationen unterscheiden:

Low-Level-Kontextinformationen werden von Sensoren des Systems als Rohdaten erfasst.

High-Level-Kontextinformationen werden aus der Kombination von Low-Level Kontextinformationen abgeleitet.

Durch die teilweise allgemein gehaltenen Definitionen für den Kontextbegriff werden in der Forschung Versuche unternommen, Kontextinformationen zu kategorisieren. Schilit u. a. (1994) kategorisieren Kontextinformationen durch die drei „W“-Fragen: „Wo Du bist? Wer bei Dir ist? Was für Ressourcen in der Nähe sind?“. Nach Abowd u. a. (1999) wird Kontext in primäre und sekundäre oder nach Dey u. a. (2001) in grundlegende und abgeleitete Kontextinformationen kategorisiert, wobei die primären oder grundlegenden Kontextinformationen folgende Kontexttypen für die Charakterisierung der Situation einer Entität beinhalten:

Position - Die Position ist mehr als die Ortsangabe in einem zweidimensionalen Raum. Sie umfasst gleichzeitig auch die Ausrichtung und die Höhenangabe, sowie alle Informationen, die die Ableitung von räumlichen Beziehungen zwischen Entitäten, wie z.B. der gleiche Aufenthaltsort oder die Nähe, ermöglichen.

Identität - Die Identität ergibt sich aus der Möglichkeit, einer Entität ein eindeutiges Identifizierungsmerkmal zuordnen zu können, wobei das Identifizierungsmerkmal nur innerhalb des benutzten Namensraums der jeweiligen Anwendung eindeutig sein muss.

Zeit - Die Zeit ist eine Kontextinformation, die für die Charakterisierung von Situationen hilfreich ist und die Verwendung von historischen Informationen ermöglicht. Die Zeit wird häufig zusammen mit anderen Kontextinformationen entweder als Zeitstempel oder als Zeitspanne für die Auszeichnung eines Momentes oder der Dauer, für die Kontextinformationen bekannt oder relevant sind, verwendet. Die Zeit kann auch nur für die relative Ordnung von Ereignissen oder dem Kausalzusammenhang genügen.

Zustand (Aktivität) - Der Zustand identifiziert intrinsische Charaktereigenschaften einer Entität, die wahrgenommen werden kann. Für einen Ort kann das z.B. die aktuelle Temperatur sein, das Umgebungslicht oder der Geräuschpegel. Für eine Person kann

sich der Zustand auf physiologische Faktoren, wie die Vitalzeichen oder die Müdigkeit oder die Aktivität, die die Person ausführt (wie z.B. Lesen oder Sprechen), beziehen. Bei Gruppen von Menschen kann sich der Zustand auf den Enthusiasmus oder den globalen Gefühlszustand oder auf die Beschreibung der Aktivität (wie z.B. eine Vorlesung oder eine Veranstaltung) beziehen. Bei Anwendungen wird der Zustand durch die Attribute (wie z.B. die Laufzeit, die Prozessorbelastung oder die Verfügbarkeit von Dateien im Dateisystem) beschrieben.

Die sekundären oder abgeleiteten Kontextinformationen können als Attribute der primären Kontextinformationen angesehen werden (z.B. ist die Telefonnummer eines Benutzers eine sekundäre Kontextinformation seiner Identität) (Abowd u. a., 1999).

Einen anderen Ansatz verfolgen Schmidt u. a. (1999b). Sie haben folgendes Kontextmodell für die Strukturierung von Kontextinformationen entwickelt:

- Ein Kontext beschreibt die Situation und die Umgebung eines Benutzers oder eines Gerätes.
- Ein Kontext wird über einen eindeutigen Namen identifiziert.
- Für jeden Kontext ist eine Menge von Eigenschaften relevant.
- Für jede relevante Eigenschaft ist eine Menge von Werten für den Kontext festgelegt (implizit oder explizit).

Auf der Basis dieses Kontextmodells haben Schmidt u. a. (1999b) dann die Kontextinformationen hierarchisch organisiert (siehe Abbildung 2.2).

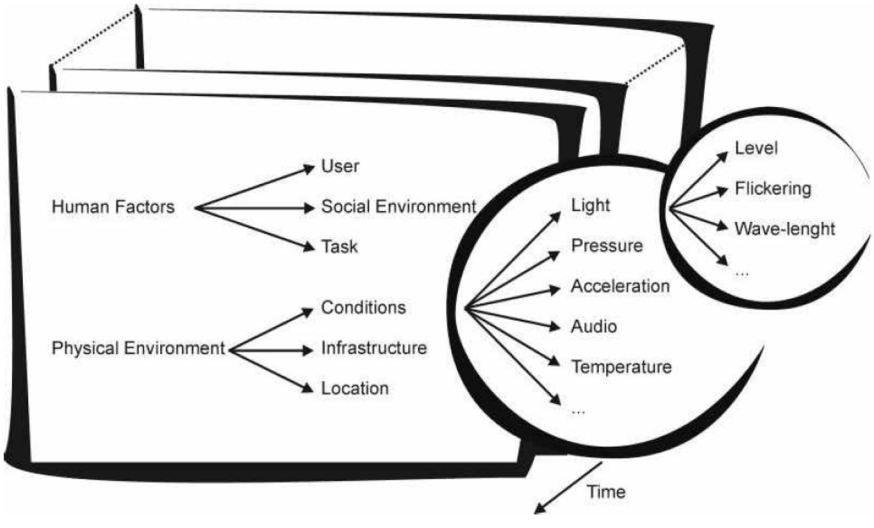

Abbildung 2.2.: Kategorisierung von Kontextinformationen (vgl. Schmidt u. a. (1999b))

Auf der obersten Ebene wird zwischen zwei Faktoren differenziert: den menschlichen Faktoren und der physikalischen Umgebung. Die zwei Faktoren werden durch jeweils drei Unterkategorien verfeinert. Nach Schmidt u. a. (1999b) handelt es sich dabei um die grundlegende Struktur der Kontextinformationen.

Die Kategorisierung der Kontextinformationen von Abowd u. a. (1999) und Schmidt u. a. (1999b) sind sich sehr ähnlich und ergänzen sich im Detail gegenseitig (z.B. wird in Schmidt u. a. (1999b) die Zeit als Kontextinformation explizit aufgeführt, dafür erfolgt in Schmidt u. a. (1999b) eine differenziertere Betrachtung der physikalischen Umgebung, die in Abowd u. a. (1999) in der Position zusammengefasst wird)(Bisgaard u. a., 2005).

Anwendungen können die identifizierten Kontextinformationen nutzen, um das eigene Verhalten an die Umgebung, in der sie sich befinden, anzupassen oder dem Benutzer angemessen Informationen zu präsentieren. Kontextbewusstsein ist die Fähigkeit einer mobilen Anwendung, Änderungen in der Umgebung, in der sie sich befindet, wahrzunehmen und auf diese Änderungen zu reagieren (Schilit und Theimer, 1994). Ähnlich wird Kontextbewusstsein auch durch Schilit u. a. (1994) definiert, die kontextbewusste Anwendungen durch das Anpassen an die Umstände der aktuellen Position und die Informationen von in der Nähe befindlichen Menschen, Zentralcomputern und erreichbaren Endgeräten sowie deren Änderungen in der Zeit beschreiben.

Kontextbewusst wird auch häufig mit anderen Begriffen synonym verwendet (Abowd u. a., 1999):

- Adaptiv (Brown, 1996)
- Reaktiv (Cooperstock u. a., 1995)
- Reagierend (Elrod u. a., 1993)
- Sich befindend (Hull u. a., 1997)
- Kontextsensitiv (Rekimoto u. a., 1998)
- Umgebungsgerichtet (Fickas u. a., 1997)

Kontextbewusstes Rechnen wird ebenfalls häufig als Synonym für Kontextbewusstsein verwendet (Schilit u. a. (1994), Abowd u. a. (1999), Chen und Kotz (2000), Bisgaard u. a. (2005)).

Basierend auf ihrer Definition von Kontextbewusstsein haben Schilit u. a. (1994) vier Kategorien von kontextbewussten Systemen identifiziert, die in vier orthogonale Dimensionen (manuell, automatisch, datenbezogen oder aktionsbezogen) eingeteilt werden (siehe Abbildung 2.3).

Die einzelnen Kategorien werden von Schilit u. a. (1994) wie folgt beschrieben:

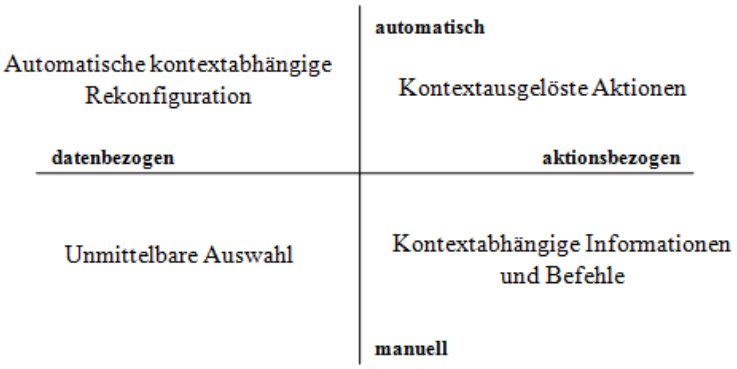

Abbildung 2.3.: Einordnung der Kategorien in die entsprechenden Dimensionen nach Schilit u. a. (1994)

Unmittelbare Auswahl - Kontextbewusste Systeme aus dieser Kategorie stellen dem Benutzer kontextbezogene Informationen zur Verfügung (z.B. wird die Position des Benutzers auf einer digitalen Landkarte zunächst zentriert dargestellt).

Automatische kontextabhängige Rekonfiguration - Bei der automatischen kontextabhängigen Rekonfiguration wird unter Berücksichtigung aktueller Kontextinformationen die Auswahl der für eine Aufgabe zu verwendenen Ressourcen ausgewählt und das System automatisch rekonfiguriert.

Kontextabhängige Informationen und Befehle - Dabei handelt es sich um kontextbewusste Systeme, die dem Benutzer kontextabhängig Informationen oder Eingabemöglichkeiten anbieten (z.B. kann beim Drucken aus einer Anwendung heraus immer der nächstgelegene Drucker ausgewählt werden).

Kontextausgelöste Aktionen - Auf Basis von festgelegten Regeln wird das Verhalten des kontextbewussten Systems festgelegt. Die Anpassungen des Systems an den aktuellen Kontext erfolgt anhand der spezifizierten Regeln und losgelöst von Benutzereingaben automatisch.

Im Gegensatz zu Schilit u. a. (1994) beschreibt Pascoe (1998) Kontextbewusstsein anhand von konkreten Einsatzmöglichkeiten losgelöst von Systemkategorien und identifiziert dabei folgende Eigenschaften für kontextbewusste Systeme:

Kontextabhängiges Abtasten - Die Informationen aus dem aktuellen Kontext werden von dem System wahrgenommen und dem Benutzer angemessen präsentiert.

Kontextabhängige Anpassung - Das System passt sich automatisch dem aktuellen Kontext an.

Kontextabhängiges Entdecken von Ressourcen - Ressourcen in der Umgebung werden durch das System wahrgenommen und können bei Bedarf genutzt werden.

Kontextabhängige Anreicherung - Der aktuelle Kontext wird durch das System digital angereichert und die Informationen dem Benutzer angemessen bereitgestellt.

Die identifizierten Eigenschaften von Pascoe (1998) decken die vorgestellten Kategorien von kontextbewussten Systemen von Schilit u. a. (1994) ab. Nach Pascoe u. a. (2000) kann jedoch ein kontextbewusstes System alle Kategorien von Schilit u. a. (1994) umfassen. Daher weicht Pascoe (1998) von der Kategorisierung von kontextbewussten Systemen ab (Abowd u. a., 1999).

Abowd u. a. (1999) haben aus diesen Eigenschaften und Definitionen eine abstraktere Definition von kontextbewussten Systemen aufgestellt (vgl. Bisgaard u. a. (2005)). Danach ist ein System kontextbewusst, wenn es Kontext verwendet, um relevante Informationen und/oder Dienste dem Benutzer anzubieten, wobei die Relevanz von der Tätigkeit des Benutzers abhängt. Auf Grundlage der Definition und der Zusammenführung der Kategorien nach Schilit u. a. (1994) und den Eigenschaften nach Pascoe (1998), kategorisieren Abowd u. a. (1999) kontextbewusste Systeme nach den Arten der möglichen Verhaltensänderung (Bisgaard u. a., 2005):

Präsentation - Dem Benutzer werden vom System Informationen und Dienste kontextabhängig zu gegebener Zeit zur Verfügung gestellt und präsentiert.

Ausführung - Das System führt Aktionen und Aktualisierungen automatisch und kontextabhängig auf Basis von auftretenden Ereignissen aus.

Kennzeichnung - Kontextinformationen werden durch das System gekennzeichnet und für den Benutzer für einen späteren Zeitpunkt, bis zur Rückkehr in den gleichen Kontext, persistent gehalten.

Auf der Grundlage von Abowd u. a. (1999) und Schilit u. a. (1994) haben Rothermel u. a. (2003) eine angepasste Einordnung bei der Unterscheidung von kontextbezogenen Systemen vorgenommen:

Kontextbezogene Selektion - Bei der Auswahl von Diensten und Informationen kann der Kontext einbezogen werden. Basierend auf den durch den Kontext ermittelten Objekten sind dann zwei weitere Arten der Verhaltensänderung möglich:

Kontextbezogene Präsentation - In Abhängigkeit des Kontexts verändert sich die Darstellung einer Anwendung (z.B. könnte über den Ort unter unterschiedlichen Darstellungsmedien in der Umgebung des Benutzers eines selektiert und von einer Anwendung für die Darstellung verwendet werden).

Kontextbezogene Aktionen - In Abhängigkeit von Kontextinformationen können Aktionen durch das System initiiert werden (z.B. könnte die Umgebung des Benutzer nach seinen Präferenzen automatisch konfiguriert werden).

Die Einordnung unterscheidet sich von Abowd u. a. (1999) dadurch, dass die Kennzeichnung von Kontextinformationen für eine spätere Verwendung als eine Verwaltungsaufgabe betrachtet wird und gegenüber der in Schilit u. a. (1994) vorgeschlagenen kontextbezogenen Aktion zurücktritt (Rothermel u. a., 2003).

Chen und Kotz (2000) identifizieren zwei unterschiedliche Möglichkeiten, Kontext zu verwenden und schlagen daher zwei Definitionen für Kontextbewusstsein vor:

Aktives Kontextbewusstsein - Beim aktiven Kontextbewusstsein handelt es sich um Systeme, die das eigene Verhalten an den aktuellen Kontext anpassen.

Passives Kontextbewusstsein - Beim passiven Kontextbewusstsein präsentiert das System neue oder aktualisierte Kontextinformationen dem Benutzer oder speichert die Informationen für den späteren Gebrauch.

Durch das aktive Kontextbewusstsein kann nach Chen und Kotz (2000) unnötige Mensch-Maschine Interkation vermieden und ein Verschwinden der Technologie aus dem Bewusstsein des Benutzers erreicht werden.

Auf Basis der vorgestellten Definitionen wird für diese Arbeit Kontextbewusstsein angelehnt an Abowd u. a. (1999) und Chen und Kotz (2000) wie folgt definiert (Definition 4):

> **Definition 4 [Kontextbewusstsein]**
> *Ein System handelt kontextbewusst, wenn es Kontext verwendet, um relevante Informationen und/oder Dienste dem Benutzer anzubieten, wobei die Relevanz von der Tätigkeit des Benutzers abhängt, und/oder wenn es Kontext verwendet, um das eigene Verhalten automatisch an den aktuellen Kontext anzupassen.*

Die gewählte Definition konkretisiert die allgemein gehaltene Definition von Abowd u. a. (1999) durch die explizite Aufführung des aktiven Kontextbewusstseins nach Chen und Kotz (2000) und schließt keine Systeme wie von Abowd u. a. (1999) gefordert aus (z.B. werden durch die Definition von Schilit und Theimer (1994) Systeme ausgeschlossen, die Informationen auf Basis des aktuellen Kontexts dem Benutzer präsentieren, da nach deren Definition Systeme kontextbewusst handeln, wenn sie ihr Verhalten an den aktuellen Kontext automatisch anpassen).

2.3.1. Herausforderungen

Damit sich Anwendungen wie in Abschnitt 2.3 kontextbewusst verhalten können, ergeben sich technologische und kulturelle Herausforderungen für die Realisierung eines pervasiven kontextbewussten Spiels, die in diesem Abschnitt allgemein identifiziert und für den weiteren Verlauf dieser Arbeit betrachtet werden sollen.

Schmidt (2002) hat bei der Untersuchung von kontextbewussten Anwendungen folgende zentrale Herausforderungen identifiziert:

Verständnis für das Kontextkonzept - Das Verständnis für das Kontextkonzept und die Verbindung mit Situationen in der realen Welt muss gesteigert und vereinheitlicht werden, damit ein einheitliches Verständnis von Kontext und der Repräsentation von Kontextinformationen entsteht, wie auch das Verständnis zu möglichen Einsatzfeldern.

Einsatz von Kontextinformationen - Der sinnvolle Einsatz von Kontextinformationen in kontextbewussten Systemen ist eine weitere Herausforderung. Die Kontextinformationen müssen bestimmten Anforderungen genügen und z.B. ausfallsicher und eindeutig sein, damit ein kontextbewusstes System sich nach den Erwartungen des Benutzers verhält. Ein kontextbewusstes System muss daher auch die Beziehung der Kontextinformationen untereinander, die zusätzlichen Eingaben in das System durch den Benutzer und den Einfluss dieser unterschiedlichen Informationen aufeinander berücksichtigen können.

Beschaffung von Kontextinformationen - Die Beschaffung von Kontextinformationen ist eine Voraussetzung für kontextbewusste Systeme. Bei dem Prozess der Beschaffung von Kontextinformationen wird eine reale Situation in der physikalischen Welt festgehalten, die signifikanten Eigenschaften ausgewertet und eine abstrakte Repräsentation erstellt, die von dem kontextbewussten System z.B. für das eigene adaptive Verhalten eingesetzt werden kann. Dafür werden unterschiedliche Mechanismen wie Sensorsysteme und Modelle für die Ableitung von Kontextinformationen und deren Darstellung benötigt.

Verbindung von Kontextbeschaffung und Kontextnutzung - In verteilten Systemen kann die Beschaffung von Kontextinformationen verteilt durch andere kontextbewusste Systeme erfolgen und muss nicht lokal auf die Sensoren des eigenen Systems beschränkt bleiben. Um die Kontextbeschaffung in verteilten Systemen zu ermöglichen, wird eine einheitliche Repräsentationsform von Kontextinformationen benötigt, die von unterschiedlichen kontextbewussten Systemen verstanden wird.

Verständnis für den Einfluss auf die Mensch-Maschine-Interaktion - Das Verhalten von kontextbewussten Systemen ist abhängig von dem aktuellen Kontext oder der Situation, in der das System ausgeführt wird, und muss sich nach den Erwartungen

des Benutzers verhalten. Die Herausforderung dabei ist, das Verständnis des Benutzers für das System und das Verhalten zu gewinnen und dem Benutzer trotzdem die Kontrolle über das System zu ermöglichen.

Unterstützung für die Entwicklung von kontextbewussten allgegenwärtigen Systemen - Kontextbewusstsein ist für die Vision des allgegenwärtigen Rechnens eine zentrale Anforderung. Für die Entwicklung wird daher ein zentraler Ansatz benötigt, der wiederverwendet werden kann.

Evaluation von kontextbewussten Systemen - Durch das kontextabhängige Verhalten von kontextbewussten Anwendungen muss eine Evaluation ebenfalls im gleichen Kontext erfolgen. Dafür muss die gleiche Situation mit dem gleichen Kontext erzeugt werden, um eine Bewertung des kontextbewussten Systems zu ermöglichen.

Die Herausforderungen nach Schmidt (2002) geben einen allgemeinen Überblick über die Herausforderungen von kontextbewussten Systemen. Konkreter gehen Widjaja und Balbo (2005) auf Herausforderungen der Mensch-Maschine Interaktion ein, die sich aber auch in allgemeiner Form in den Herausforderungen von Schmidt (2002) wiederfinden:

Interpretation - Fehlinterpretationen können aus der Komplexität des aktuellen Kontexts resultieren, wenn die Bedeutung nicht einfach aus Kontextinformationen von niedrigerer Ebene (z.B. direkten Sensordaten) abgeleitet werden kann. Die Durchführung einer Aktivität ist jeweils an eine Situation gebunden und kann keinem Determinismus untergeordnet werden (z.B. ist die Sprechblase in der Task-Leiste von Windows XP mit der Nachricht, dass ungenutzte Schnellstartsymbole auf der Arbeitsfläche vorhanden sind, ein Beispiel für eine allgemeine Fehlinterpretation).

Darstellung - Eine falsche Darstellung kann aus Fehlinterpretationen oder ungenauer Darstellungslogik resultieren. Wenn z.B. Kontextinformationen nur für den Darstellungsfokus eingesetzt werden, hat ein Benutzer trotzdem Zugriff auf weitere Informationen und Interaktionsmöglichkeiten. Wird jedoch ein Benutzer auf eine Darstellung beschränkt, ist keine weitere Interaktion möglich.

Ausführung - Durch z.B. das automatische Anpassen von kontextbewussten Systemen an den aktuellen Kontext kann der Benutzer von der kognitiven Belastung befreit werden. Gleichzeitig wird jedoch durch den Einsatz von Kontextinformationen das Risiko durch initiale Fehlinterpretationen und fehlende oder veraltete Kontextinformationen für das adaptive Verhalten erhöht. Dadurch wird der Aufwand für den Benutzer erhöht, da er für eventuelles Fehlverhalten Lösungen finden muss.

Ebenso wie Widjaja und Balbo (2005) sehen Crowley u. a. (2002) die größte Herausforderung in der Mensch-Maschine Interaktion aus der Benutzerperspektive. Die Interaktion von Benutzern ist geprägt durch ständige Wechsel zwischen unterschiedlichen Zielen und das Hinzufügen von neuen Zielen. Kontextbewusste Systeme müssen dieses unberechenbare Verhalten erlauben und voraussehen können.

Für Nurmi und Floreen (2004) dagegen ist das Schlussfolgern auf Basis des Kontexts eine der Hauptherausforderungen. Die Herausforderungen beim Schlussfolgern aus dem aktuellen Kontext können nach Nurmi und Floreen (2004) aus vier Perspektiven betrachtet werden:

Benutzersicht - Damit aus dem aktuellen Kontext eine Benutzersicht entwickelt werden kann, muss aus den gegebenen Sensorinformationen jeweils eine aktuelle Momentaufnahme des Kontexts bereitgestellt werden. Dabei können Kontextinformationen aus mehreren und unterschiedlichen Kontextquellen bezogen werden oder aber die Kontextinformationen sind fehlerhaft oder fehlen komplett.

Anwendungssicht - Aus der Sicht der Anwendung besteht die Herausforderung in der Fähigkeit einer Anwendung, den Kontext intelligent einzusetzen.

Kontext Monitoring - Damit Anwendungen vorausschauend handeln können, müssen Kontextänderungen rechtzeitig erkannt werden und Reaktionen auf diese Änderungen möglich sein.

Modell Monitoring - Anwendungen müssen die erkannten Änderungen im Kontext wahrnehmen und das Kontextmodell stetig aktualisieren. Gleichzeitig müssen die Entscheidungen der Anwendungen beobachtet und ausgewertet werden und die Rückmeldung des Benutzers in das Verhalten der Anwendungen einfließen.

Daneben gibt es noch weitere Herausforderungen z.B. im Bereich der Sicherheit und der Privatssphäre (Bhaskar und Ahamed, 2007), die aber im Rahmen dieser Arbeit nicht weiter ausgeführt werden.

2.4. Allgegenwärtiges und pervasives Rechnen

Beim allgegenwärtigen Rechnen handelt es sich um eine von Mark Weiser (Weiser) geprägte Vision von verschwindenden Computern, die unsichtbar den Menschen bei seinen Tätigkeiten unterstützen und ihn von lästigen Routineaufgaben weitestgehend befreien.

> *„The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.“*
>
> — Mark Weiser (Weiser, 1991)

Aus dieser Vision lässt sich folgende Definition nach Weiser (1991) ableiten (Definition 5).

Definition 5 [Allgegenwärtiges Rechnen]
Allgegenwärtiges Rechnen bezeichnet die Allgegenwärtigkeit der rechnergestützten Informationsverarbeitung im Alltag von Menschen.

Dabei handelt es sich nach Weiser beim allgegenwärtigen Rechnen um die dritte Welle in der rechnergestützten Informationsverarbeitung, die zurzeit beginnt (siehe Abbildung 2.4).

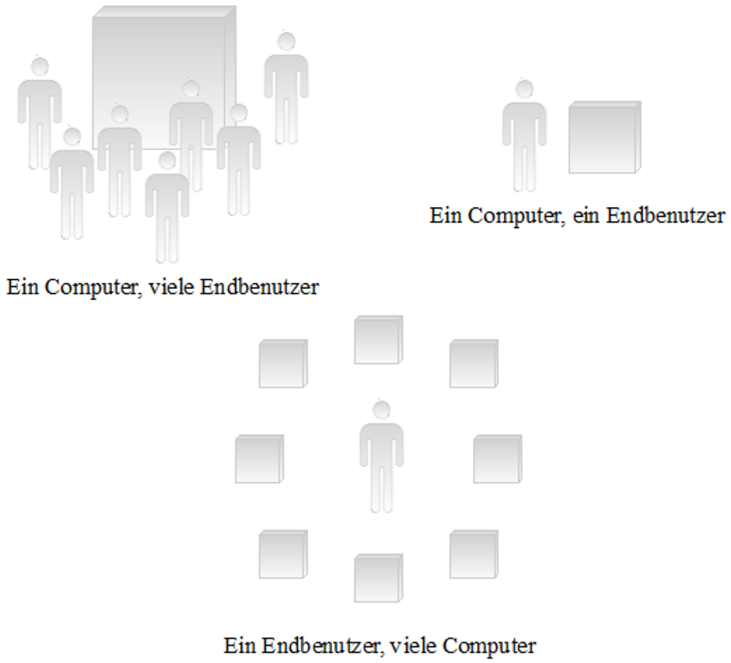

Abbildung 2.4.: Die drei Wellen der rechnergestützten Informationsverarbeitung (vgl. Zeidler (2007))

In der Beschreibung der drei Wellen der rechnergestützten Informationsverarbeitung von Zeidler (2007) bilden Großrechner (engl. mainframes) die erste Welle in der Datenverarbeitung, bei der die Rechenleistung zentral zur Verfügung gestellt wird und dann von vielen Endbenutzern geteilt werden muss. Die Nutzung des Großrechners muss durch die Endbenutzer vorbereitet und geplant werden, ist teuer und kann nur von Experten erfolgen. Die zweite Welle ist die Zeit der personalisierten Computer. Die Endbenutzer haben die Kontrolle über und den exklusiven Zugriff auf die Rechenleistung eines Computers und können Tätigkeiten bei Bedarf durchführen. Im Gegensatz zu Großrechnern, die zentral gewartet werden können, ist bei personalisierten Computern der Endbenutzer für die Wartung und Administration zuständig. Dafür ist die gesamte Aufmerksamkeit des Endbenutzers nötig. Seit Mitte der achtziger Jahre ist die Anzahl der Endbenutzer von personalisierten Computern größer als die von Großrechnern. In der dritten aktuell, aufsteigenden Welle interagiert

jeder Endbenutzer mit vielen Computern parallel und unterbewusst. Die Computer vernetzen sich spontan untereinander und erfüllen unsichtbar die Wünsche des Endbenutzers. Im Gegensatz zur ersten Welle ist in der zweiten Welle nicht die Rechenleistung der begrenzende Faktor, sondern die nötige Aufmerksamkeit des Endbenutzers. Die kontinuierlich nötige Aufmerksamkeit (aus der zweiten Welle) ist für dieses Szenario der dritten Welle nicht wünschenswert. Aus diesem Grund werden Technologien benötigt, die wenig Aufmerksamkeit durch den Endbenutzer benötigen und für den Endbenutzer nicht sichtbar sind, also verschwindende Computer (Weiser, 1993).

Bei diesem Zustand wird auch vom unsichtbaren Rechnen (engl. calm computing) gesprochen (Weiser u. a., 1999). Beim unsichtbaren Rechnen verschwinden die Computer in den Hintergrund und rücken nur bei Bedarf in das Zentrum der Aufmerksamkeit (Weiser und Seely Brown, 1995). Technologien für das unsichtbare Rechnen müssen dabei folgende Eigenschaften aufweisen (Tugui, 2004):

1. Durch unsichtbare Technologien wird der Fokus aus dem Zentrum unserer Aufmerksamkeit in den Hintergrund verschoben. Diese technologische Ausrichtung kann entweder durch das reibungslose und leichte Verschieben aus dem Zentrum der Aufmerksamkeit in den Hintergrund und zurück erreicht werden oder aber durch das Übertragen von Informationen in den Hintergrund. Ein Beispiel ist der Vergleich zwischen einer Videokonferenz und einer Telefonkonferenz. In einer Videokonferenz können Körpergesten und Gesichtsausdrücke wahrgenommen werden, die in einer Telefonkonferenz nicht zugänglich sind.
2. Eine Technologie ist unsichtbar, wenn die unbewusste Wahrnehmung gesteigert wird und eine direkte Auswirkung auf das Wissen hat. Dadurch kann ein angemessenes Verhalten in verschiedenen Situationen ermöglicht werden, ohne dabei mit Informationen überfrachtet zu werden. Unsichtbare Technologien schaffen damit eine angenehme Umgebung.
3. Die technologische Verbundenheit ermöglicht eine schnelle Bestimmung von Situationen unter bestimmten Umständen und vor dem Hintergrund sich schnell ändernder Hintergrundinformationen, die eine schnelle Wahrnehmung von der Vergangenheit, Gegenwart und Zukunft eines Gegenstandes bedingen.

In Bezug auf die heutige Computertechnik kann die Vision von Weiser von unsichtbaren Computern mit Hilfe folgender Aspekte interpretiert werden (Mattern, 2008):

Eingebettetes Rechnen - Kleine und preiswerte Prozessoren, Sensoren, Speicher und Kommunikationsmodule lassen sich in Alltagsgegenstände integrieren, was unter eingebettetem Rechnen (engl. embedded computing) verstanden wird.

Tragbares Rechnen - Wenn Computer am Körper oder in der Kleidung getragen werden, dann wird vom tragbaren Rechnen (engl. wearable computing) gesprochen.

Sensornetzwerke - Wird die Umwelt mit Informationstechnologien ausgestattet, etwa um die Umgebung zu beobachten, dann handelt es sich um Sensornetze.

Durch den rasanten technologischen Fortschritt und die Verdopplung der Leistungsfähigkeit von Prozessoren und Speicherbausteinen (bzw. der entsprechenden Verkleinerung und Verbilligung bei konstanter Leistungsfähigkeit (Mattern, 2007)) in etwa allen 18 Monaten nach dem nach Gordon E. Moore (Moore, 1965) aufgestellten und nach ihm benannte „Gesetz“, ist eine teilweise Realisierung der Vision von Mark Weiser (Weiser) mittel- bis langfristig denkbar (Mattern, 2005).

Der von IBM eingeführte Begriff des pervasive Rechnens resultiert aus einem Interview mit Marc Bregman (Bregman) und verfolgt im Vergleich zum allgegenwärtigen Rechnen einen eher pragmatischen Ansatz.

> *„Pervasive computing is about enabling people to gain immediate access to information and services anywhere, anytime, without having to scrounge for a phone jack. However, while mobility and wireless technology are a big part of it, it's really about making e-business personal. Thanks to the explosive growth of the Internet, people will soon expect to be able to engage in electronic business effortlessly.“*
>
> — Marc Bregman (Bregman)

Ziel ist die Durchdringung des Alltags mit verfügbaren Technologien, unter kommerziellen Gesichtspunkten, bei der Informationen und Dienste jederzeit für den Endbenutzer verfügbar sind. Um ein einheitliches Verständnis des Begriffs des pervasiven Rechnens für diese Arbeit zu erreichen, wird der Begriff für den weiteren Verlauf dieser Arbeit wie folgt definiert (siehe Definition 6):

Definition 6 [Pervasives Rechnen]
Pervasives Rechnen bezeichnet die alles durchdringende Vernetzung des Alltags des Menschen durch den Einsatz von verfügbaren intelligenten, rechnergestützten Informationstechnologien aus dem Bereich des mobilen Rechnens und von Kommunikationstechnologien und die daraus resultierende Verfügbarkeit von Informationen und Diensten zu jedem Zeitpunkt.

In der vorgestellten Definition wird bewusst auf die unternehmenskulturelle Perspektive verzichtet, da eine Betrachtung unter kommerziellen Gesichtspunkten in dieser Arbeit nicht erfolgt.

Allgegenwärtiges und pervasives Rechnen wurden ursprünglich aus unterschiedlichen Perspektiven interpretiert. Aus der technologischen Perspektive und der Nutzung beim allgegenwärtigen Rechnen und aus unternehmenskultureller Perspektive beim pervasiven Rechnen (vgl. Nieuwdorp (2007)). Nach Nieuwdorp (2007) unterliegen die Begriffe aber einer

andauernden Wandlung durch Neudefinitionen und Vergleiche mit anderen Konzepten und Begriffen und werden daher in der Forschung unterschiedlich verstanden:

Parallel, austauschbar - Pervasives und allgegenwärtiges Rechnen werden als ähnlich oder austauschbar verstanden (vgl. Gupta u. a. (2002), Want und Pering (2005), Nakajima (2003), Roth (2005) oder Satyanarayanan (2001)). In anderen Fällen werden diese Begriffe nicht direkt miteinander in Verbindung gebracht, aber das pervasive Rechnen wird durch den direkten Bezug zu Aspekten aus dem allgegenwärtigen Rechnen, als Beispiele für das pervasive Rechnen, gleichgesetzt (Huang u. a., 1999).

Korrelierend zu anderen Konzepten - Pervasives und allgegenwärtiges Rechnen werden als korrelierend zu anderen Konzepten aus der rechnergestützten Informationsverarbeitung verstanden, z.B. zum mobilen Rechnen (engl. mobile computing) (Schafer, 1999), Umgebungsrechnen (engl. ambient computing) (Payne und Macdonald, 2004) und physikalischen, umgebenden und eingebetteten Rechnen. Beim Umgebungsrechnen oder im Bereich der Umgebungsintelligenz (engl. ambient intelligence) wird von einigen Forschern auch auf Technologien verwiesen, die generell als pervasiv oder allgegenwärtig betrachtet werden, da sie eingebettet, intelligent und einfach zu bedienen sind (Markopoulos u. a., 2005).

Unterscheidbar - Pervasives und allgegenwärtiges Rechnen werden getrennt voneinander und zu anderen Formen der rechnergestützten Informationsverarbeitung betrachtet (Lyytinen und Yoo (2002), Singh u. a. (2006)).

Das unterschiedliche Verständnis der Begriffe wird durch unterschiedliche Perspektiven und Definitionen motiviert (Nieuwdorp, 2007). In dieser Arbeit werden diese Begriffe als austauschbar betrachtet, wobei das allgegenwärtige Rechnen eher die Vision von Mark Weise (Weiser) aus einer kulturellen Perspektive widerspiegelt und das pervasive Rechnen eine Annäherung an diese Vision mit verfügbaren Technologien aus einer technologischen Perspektive darstellt.

2.4.1. Herausforderungen

Um die in der Definition des pervasiven Rechnens (siehe Definition 6) beschriebene Vernetzung des Alltags des Menschen mit verfügbarer Informationstechnologie zu erreichen, ergeben sich neue technologische Herausforderungen z.B. gegenüber dem mobilen Rechnen, die in diesem Abschnitt dargestellt werden. Gleichzeitig ergeben sich auch neue soziale und kulturelle Herausforderungen, die in dieser Arbeit jedoch nicht im Detail verfolgt werden, da der Fokus auf der Entwicklung eines Rahmenwerks für pervasive Spiele liegt und daher eher technisch ausgeprägt ist.

Nach Satyanarayanan (2001) können die Herausforderungen des pervasiven Rechnens anhand von drei Evolutionsstufen charakterisiert werden (siehe Abbildung 2.5).

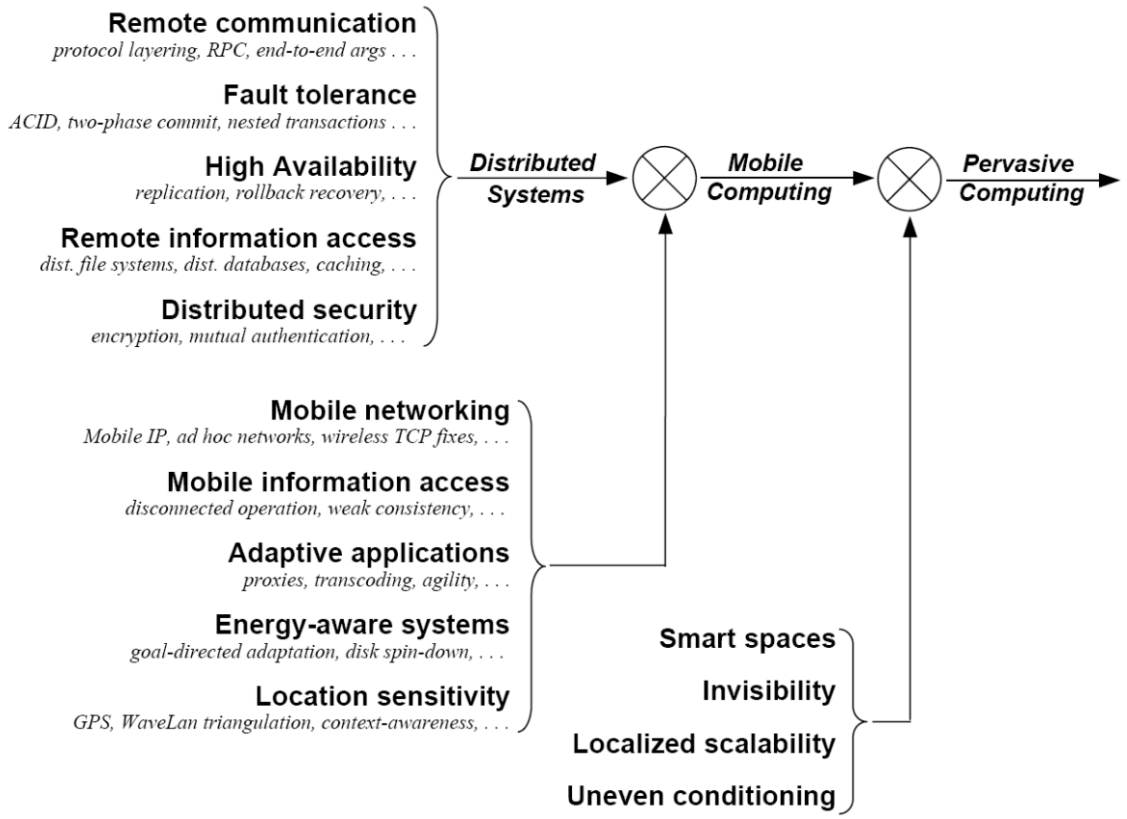

Abbildung 2.5.: Herausforderungen des pervasiven Rechnens anhand von drei Evolutionsstufen (vgl. Satyanarayanan (2001))

Ausgehend von den Herausforderungen von verteilten Systemen, wie z.B. entfernter Kommunikation durch die Vernetzung von Computern über kabelgebundene lokale Netzwerke (engl. local area network (LANLocal Area Network (LAN))), werden durch das mobile Rechnen neue Problembereiche hinzugefügt, wie z.B. der mobile Zugriff auf Informationen und die Mobilität der Endgeräte. Die prinzipiellen Herausforderungen von verteilten Systemen bleiben dabei auch weiterhin bestehen oder werden sogar noch verstärkt, z.B. der Zugriff auf entfernte Informationen mit wandernden (engl. roaming) Endgeräten (Zeidler, 2007). Beim pervasiven Rechnen ergeben sich wiederum neue Herausforderungen und bestehende werden schwieriger lösbar (z.B. die Unsichtbarkeit von Informationstechnologie, eine minimal nötige Interaktion mit der Informationstechnologie oder die automatische Anpassung von Anwendungen und Endgeräten an die Umgebung und die Umgebungsinformationen (siehe Kontext und Kontextbewusstsein in Abschnitt 2.3).

In der Forschung werden die allgemeinen Herausforderungen ausgehend von der jeweiligen Perspektive von verschiedenen Forschern und Forschergruppen unterschiedlich betrachtet.

Henricksen u. a. (2001) identifizieren die Herausforderungen an eine Infrastruktur für das pervasive Rechnen in Form eines konzeptionellen Modells bestehend aus Endgeräten, Softwarekomponenten, Benutzern und der Benutzerschnittstelle aus der Software-Engineering Perspektive.

Endgeräte - Pervasive Umgebungen zeichnen sich durch eine Heterogenität von Endegräten aus, die trotz unterschiedlicher Ausstattung und Anwendungen miteinander interagieren können müssen. Dafür wird eine Infrastrukur benötigt, die Kenntnisse über die Eigenschaften der Endgeräte verwaltet und die Integration der Endgeräte in ein kohärentes System ermöglicht, z.B. die spontane Interaktion zwischen einem mobilen Endgerät und einer Desktop-Arbeitsstation. Durch die Mobilität der Endgeräte, z.B. zwischen unterschiedlichen Netzwerken, sollte die Infrastruktur mit den Anwendungen kooperieren können, um z.B. die Verwaltung der Datenreplikation bei Verbindungsabbrüchen zu übernehmen.

Softwarekomponenten - Die Infrastruktur für pervasives Rechnen ist für die Unterstützung der Anwendungen in Bezug auf die kontextbewusste Verarbeitung von Umgebungsinformationen, die Anpassung an die Umgebung, die Migration der Anwendungen zwischen unterschiedlichen Endgeräten, die Verteilung der Softwarekomponenten und die Interoperabilität zuständig. Außerdem sollte eine pervasive Infrastruktur eine zügige Entwicklung und Inbetriebnahme erleichtern und das Auffinden von Komponenten in der pervasiven Infrastruktur ermöglichen. Dazu muss die Skalierbarkeit sichergestellt werden.

Benutzer - Die Benutzer in pervasiven Umgebungen können mobil sein und dadurch Sitzungen verteilt über unterschiedliche Endgeräte unterhalten oder zwischen unterschiedlichen Endgeräten für eine Tätigkeit wechseln. Die Infrastruktur für pervasives Rechnen sollte den Benutzer unterstützen und das Wissen über den Kontext des Benutzers, z.B. die persönlichen Präferenzen, und die Aufgaben des Benutzers verwalten.

Benutzerschnittstelle - Die Benutzer in pervasiven Umgebungen werden den allgegenwärtigen Zugriff auf Anwendungen fordern. Dadurch sind neue universell verfügbare Benutzerschnittstellen nötig, die neue Anforderungen an die Benutzerschnittstelle stellen, z.B. unterschiedliche Ein- und Ausgabemöglichkeiten für die Interaktion und die dynamische Anpassung der Benutzerschnittstelle an den aktuellen Kontext, z.B. des Benutzers- oder des Endgeräts.

Banavar und Bernstein (2002) haben anhand von beispielhaft angeführten Szenarien folgende charakteristische Eigenschaften des pervasiven Rechnens identifiziert und daraus die Herausforderungen abgeleitet.

Aufgabendynamisierung - Anwendungen für das pervasive Rechnen sollten jederzeit und überall verfügbar sein und sich der Dynamik der Umgebung des Benutzers anpassen.

In diesen Umgebungen können Benutzer ihre Ziele ändern oder ihre Aktionen der sich ändernden Umgebung anpassen. Dafür werden Anwendungen benötigt, die sich dynamisch Änderungen anpassen und bei Bedarf die Ziele oder den Plan zur Zielerfüllung modifizieren. Dabei kann der Benutzer für die neue Aufgabe entweder aktiv das System rekonfigurieren oder aber das System rekonfiguriert sich anhand von Sensordaten für die jeweilige Aufgabe automatisch. Anwendungen müssen dann auch in der Lage sein, die getroffenen Entscheidungen für die Anpassung der Aufgaben zu begründen und aus der Entscheidung lernen.

Endgeräteheterogenität und Einschränkungen in den Ressourcen - Durch die Mobilität der Benutzer oder der Endgeräte müssen Anwendungen für das pervasive Rechnen dem Benutzer folgen und sich den technologischen Ressourcen in der Umgebung anpassen können. Besonders die Ressourcen von mobilen Endgeräten sind dabei begrenzt (z.B. Energiezufuhr, Displaygröße, Netzwerkbandbreite). Wenn eine Anwendung einem Benutzer in einer pervasiven Umbegung folgen soll, dann muss sich die Anwendung der Verfügbarkeit der unterschiedlichen Ressourcen in der Umgebung (z.B. unterschiedliche Ein- und Ausgabemöglichkeiten oder unterschiedliche Netzwerktypen) und/oder Software-Diensten anpassen können.

Informationsverarbeitung in einer sozialen Umgebung - Charakteristisch für pervasive Umgebungen ist der Einfluss auf die soziale Umgebung. Durch die Durchdringung der Umgebung mit Informationstechnologie muss die Nutzung und der Zugriff auf die Informationen eingeschränkt und eindeutig geklärt werden.

Kindberg und Fox (2002) unterteilen die Herausforderungen des pervasiven Rechnens in die physikalische Integration und die spontane Interaktion:

Physikalische Integration - Pervasive Systeme umfassen die teilweise Integration zwischen informationsverarbeitenden Knoten und der physikalischen Welt (z.B. ein intelligenter Kaffeebecher, der zusätzlich noch Sensoren besitzt und den eigenen Zustand in der physikalischen Welt kommunizieren kann).

Spontane Interaktion - In einer pervasiven Umgebung kollaborieren unterschiedliche Anwendungskomponenten, z.B. Dienste, Ressourcen oder Anwendungen, spontan und stellen dadurch die benötigte Funktionalität zur Verfügung. Spontan interagierende Komponente können die Partner bei der Bearbeitung von Aufgaben ständig wechseln, während die Komponente selbst mobil ist oder andere Komponenten in Reichweite kommen. Der Wechsel der Partner erfolgt dabei ohne die Notwendigkeit neuer Anwendungen oder von Parametern.

Franklin (2001) betrachtet die Herausforderungen an das pervasive Rechnen aus der Perspektive des *„ubiquitous data management“*, also der allgegenwärtigen Datenverwaltung.

Unterstützung der Mobilität - Durch die Kompaktheit der Endgeräte und den Möglichkeiten zur drahtlosen Kommunikation wird die Nutzung in mobilen Umgebungen ermöglicht. Existierenden Anwendungen muss es ermöglicht werden, in heterogenen und dynamischen Umgebungen zu arbeiten, auch wenn wenn z.B. das Endgerät von einem Netz- oder Dienstanbieter zu einem anderen wechselt oder den Standort verändert.

Kontextbewusstsein - Kontextbewusste Endgeräte müssen sich der Umgebung und den aktuellen oder zukünftigen Aufgaben des Endbenutzers anpassen können. Kontextbewusste Anwendungen reichen von intelligenten Benachrichtigungssystemen, die den Endbenutzer über wichtige Informationen informieren, bis hin zu Räumen oder Umgebungen, die sich anhand der Anwesenheit von Personen oder Endgeräten der jeweiligen Aktivität anpassen.

Unterstützung der Kollaboration - Für die Kollaboration von Gruppen von Menschen oder Endgeräten wird Unterstützung benötigt, z.B. für Konferenzen oder die gleichzeitige Bearbeitung von gemeinsamen Daten durch unterschiedliche Endbenutzer.

Daneben gibt es noch weitere Herausforderungen für das pervasive Rechnen, die an konkreten Technologien identifiziert werden (Schmidt u. a., 2006), kulturelle Herausforderungen in Bezug auf die Interaktion beim pervasiven Rechnen zeigen (Thackara, 2001) oder Herausforderungen im Bereich der Sicherheit und Privatsphäre betrachten (Bhaskar und Ahamed, 2007).

Die Herausforderungen des pervasiven Rechnens werden von unterschiedlichen Forschern und Forschergruppen teilweise aus unterschiedlichen Perspektiven betrachtet und ergänzen sich dadurch gegenseitig. Die identifizierten Herausforderungen können in dieser Arbeit als Kriterien für die Analyse herangezogen werden und können dadurch bei der Entwicklung eines Rahmenwerks für pervasives Spielen helfen, relevante Problembereiche rechtzeitig zu erkennen.

2.5. Pervasives Spielen

Beim pervasiven Spielen (engl. pervasive gaming) handelt es sich um eine Teildisziplin des pervasiven Rechnens (siehe Abschnitt 2.4), bei der traditionelle Spiele in der realen Welt mit Informations- und Kommunikationstechnologien angereichert werden und somit eine Brücke zwischen der virtuellen und der physikalischen Welt geschlagen wird (Schneider und Kortuem, 2001). Dadurch verschwimmen die Grenzen zwischen der Spielwelt und der realen Welt und es findet eine Vermischung des Spiels mit der Realität statt (Benford u. a. (2005), De Souza e Silva (2004), Walther (2005a), Walther (2005b), Montola (2005)). Daher ist die Idee von pervasiven Spielen die Integration von Technologien aus dem Bereich des pervasiven und mobilen Rechnens zur (Hinske u. a., 2007):

Unterstützung z.B. von traditionellen Spielen, bei denen Teilaspekte durch Technologien aus dem Bereich des pervasiven Rechnens ersetzt werden, um diese Teilaspekte aus Benutzersicht zu vereinfachen,

Anreicherung z.B. durch das Einsetzen von Technologien aus dem Bereich des pervasiven Rechnens, um Komponenten hinzuzufügen, die bisher nicht existiert haben oder vorher nicht umsetzbar waren und/oder

Realisierung des Spiels selbst z.B. durch das Umsetzen von gänzlich neuen Spielen, die vorher nicht existiert haben.

Im Gegensatz zu klassischen Computerspielen, die in einer rein virtuellen Welt stattfinden, sind beim pervasiven Spielen physikalische Bewegungen und soziale Interaktion mit anderen Benutzern in der physikalischen Welt nötig (Magerkurth u. a., 2005). In der Forschung existiert jedoch nach Nieuwdorp (2007) keine allgemein anerkannte und allgemeingültige Definition für den Begriff des pervasiven Spielens oder von pervasiven Spielen. In Schneider und Kortuem (2001) wird nach Nieuwdorp (2007) ersmalig der Begriff des pervasiven Spiels als Begriff verwendet und anschließend durch weitere Forscher und Forschergruppen definiert, wobei der Begriff des pervasiven Spielens dabei grundlegend aus zwei unterschiedlichen Perspektiven betrachtet wird (Nieuwdorp, 2007):

Technologische Perspektive - Bei der technologischen Perspektive wird unter dem Begriff des pervasiven Spielens ein Spiel verstanden, das in erster Linie von pervasiven Technologien und nichtstandardisierten Eingabegeräten abhängig ist (Schneider und Kortuem (2001), Harris u. a. (2004), Chalmers (2005), Holleis u. a. (2006)). Oder aber es handelt sich um ein existierendes Spiel, das durch Computer angereichert wird und dadurch eine Mischung zwischen der realen und virtuellen Welt entsteht (Magerkurth u. a., 2005).

Kulturelle Perspektive - Bei der kulturellen Perspektive wird das Spiel selbst fokussiert und dabei die Beziehung der virtuellen Welt zur physikalischen Welt betrachtet (De Souza e Silva (2004), Benford u. a. (2005), Walther (2005a), Walther (2005b), Bartelme (2005), Magerkurth u. a. (2005)).

Teilweise erfolgt die Betrachtung des Begriffs des pervasiven Spielens von den Forschern und Forschergruppen durch beide Perspektiven gleichzeitig (Nieuwdorp, 2007). In Magerkurth u. a. (2005) wird z.B. die gesamte Welt als Spielfeld gesehen („*. . . to regard the entire world . . . as a game board. . . .* “), was der kulturellen Perspektive entspricht, gleichzeitig erfolgt aber durch die Aussage, dass traditionelle Spiele in der Realität durch Computerfunktionalität angereichert werden („*. . . in which traditional, real-world games are augmented with computing functionality. . .* “), eine Betrachtung aus der technologischen Perspektive.

Aufgrund der Vielzahl von unterschiedlichen Definitionsversuchen und den zwei vorgestellten Perspektiven aus dem Bereich des pervasiven Spielens durch Nieuwdorp (2007) wird

für den weiteren Verlauf dieser Arbeit der Begriff des pervasiven Spielens nach Hinske u. a. (2007) wie folgt definiert (Definition 7):

> **Definition 7 [Pervasives Spielen]**
> *Pervasives Spielen ist eine spielerische Form von vermischter Unterhaltung mit Zielen, Regeln, Wettbewerb und Angriffsmöglichkeiten, die auf der Verwendung des mobilen und/oder pervasiven Rechnens basieren.*

Hinske u. a. (2007) definieren den Begriff des pervasiven Spielens in ihrem Definitionsversuch aus der technologischen Perspektive durch den explizit aufgeführten Einsatz von Technologien aus dem Bereich des mobilen und/oder pervasiven Rechnens. Gleichzeitig wird aber durch den Bezug auf die vermischte Unterhaltung implizit die kulturelle Perspektive betrachtet und zusätzlich die in Abschnitt 2.5.1 identifizierten Anforderungen an ein Spiel berücksichtigt. Auf Basis dieser Definition kann eine Eignung eines Szenarios für die Validierung für ein Rahmenwerk für pervasive Spiele in dieser Arbeit überprüft und gemessen werden.

Im weiteren Verlauf dieses Abschnitts werden die grundlegenden Anforderungen an ein Spiel und Aspekte der Spielgestaltung betrachtet (Abschnitt 2.5.1) und anschließend die konkreten Möglichkeiten der Unterstützung durch Technologien des pervasiven Rechnens für pervasive Spiele vorgestellt (Abschnitt 2.5.3). Im letzten Abschnitt werden einige pervasive Spiele vorgestellt und beispielhaft angeführt (Abschnitt 2.5.4).

2.5.1. Spiele und Spielgestaltung

Damit im Rahmen dieser Arbeit ein Rahmenwerk für pervasive Spiele entwickelt werden kann, wird eine Spielidee für ein Spiel benötigt, die allgemeine Anforderungen an ein Spiel widerspiegelt und somit auch für andere Spielideen anwendbar ist. Aus diesem Grund werden in diesem Abschnitt die Eigenschaften eines Spiels identifiziert und das Spielerlebnis für den Spieler in einem Spiel untersucht.

Zum Begriff des Spiels existieren eine Vielzahl von unterschiedlichen Definitionen. Eine allgemein anerkannte und viel zitierte Definition stammt von dem niederländischen Kulturanthropologen Johan Huizinga.

> *„Spiel ist eine freiwillige Handlung oder Beschäftigung, die innerhalb gewisser festgesetzter Grenzen von Zeit und Raum nach freiwillig angenommenen, aber unbedingt bindenden Regeln verrichtet wird, ihr Ziel in sich selber hat und begleitet wird von einem Gefühl der Spannung und Freude und einem Bewusstsein des „Andersseins“ als das „gewöhnliche Leben“.“*
>
> — Johan Huizinga (Huizinga, 1938)

Hinske u. a. (2007) haben existierende Definitionen untersucht und dabei folgende sechs zentrale Schlüsselelemente identifiziert, die sich zum Teil auch in der Definition von (Huizinga, 1938) wiederfinden lassen:

- Regeln
- Wettbewerb
- Ziele
- Ergebnis
- Entscheidungen
- Emotionale Bindung

Auf Basis der sechs identifizierten Schlüsselelemente haben Hinske u. a. (2007) eine erweiterte Klassifikation der Unterhaltungsformen von Crawford (2003) eingeführt, nach der unterschiedliche Unterhaltungsformen klassifiziert und die Anforderungen an Spiele, als eine mögliche Form der Unterhaltung, abgelesen werden können (siehe Abbildung 2.6).

Die Klassifikation nach Hinske u. a. (2007) beginnt mit der Betrachtung des Unterhaltungsbegriffs. Wird eine spielerische (und interaktive) Komponente hinzugefügt, resultiert das in Spielzeugen. Spielzeuge mit Regeln und eindeutigen Zielen wiederum sind Herausforderungen, wobei Spielzeuge ohne Ziele und Regeln Spielwaren sind. Herausforderungen lassen sich dann in Puzzle (wenn es keinen Gegenspieler gibt) und Konflikte (wenn Gegenspieler existieren) aufteilen. Dabei handelt es sich um Wettkämpfe, wenn keine direkten Angriffe erlaubt sind, und um Spiele, wenn Angriffe auf den Gegenspieler möglich sind. Wenn dem Unterhaltungsbegriff eine erzählende Komponente hinzugefügt wird, ergibt sich eine Geschichten-Erzählung. Filme, Bücher usw. sind in diesem Fall nicht interaktiv, wogegen Rollenspiele interaktiv sind, aber dafür kein einheitliches Ziel und nicht zwingend Regeln besitzen. Aus diesem Grund sehen Hinske u. a. (2007) Rollenspiele auch irrtümlich als Spiele nach den Eigenschaften der vorgestellten Klassifikation bezeichnet.

Spiele selber können auch wieder unterschiedlich klassifiziert werden. Eine Klassifikation von Spielen wird zum Beispiel von Lindley (2003) vorgestellt und vertieft. Da in dieser Arbeit aber nicht die Entwicklung und Analyse des Spiels und der Spielidee an sich im Vordergrund steht, sondern das Spiel nur ein mögliches Szenario darstellt, ist eine Betrachtung der allgemeinen Anforderungen an ein Spiel im Rahmen dieser Arbeit ausreichend.

In der Betrachtung eines Spiels aufgrund der Klassifikation der Unterhaltungsformen nach Hinske u. a. (2007) ist bisher nicht auf die Spieler eingegangen worden. Da nach Klabbers (2003) Spiele, unabhängig vom Spielformat und vom Inhalt, soziale Systeme repräsentieren, lassen sich grundlegend drei in Beziehung stehende Bereiche in Spielen identifizieren (Klabbers, 2003):

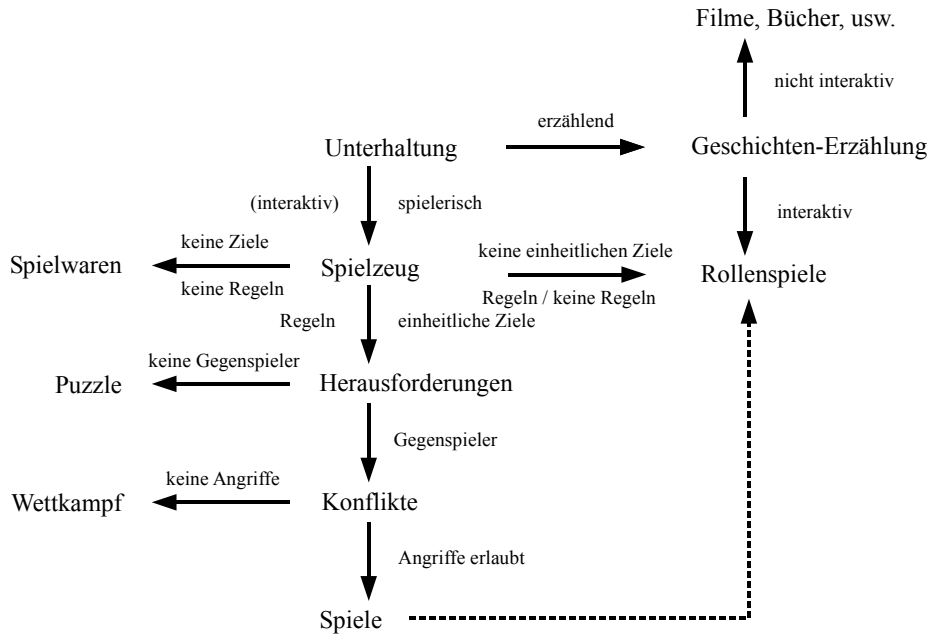

Abbildung 2.6.: Spiele als Form der Unterhaltung: Klassifikation nach Crawford (2003) erweitert durch Hinske u. a. (2007)

Akteure - Die teilnehmenden Spieler können unterschiedliche Rollen innerhalb eines Spiels einnehmen und Aktivitäten ausführen, wobei die Spieler individuell oder in Gruppen organisiert sein können.

Regeln - Die Regeln umfassen die Manipulationsmenge des Spiels, die mögliche Bewegungen mit den Gegenständen oder der Positionen im Spiel ermöglichen.

Ressourcen - Die Ressourcen sind die Menge der Gegenstände zum Spielen. Die Verknüpfung der Gegenstände wird durch die Regeln definiert. Die Gegenstände sind in der Spielwelt angesiedelt, initial angeordnet und können sich während des Spiels ändern. Die Menge aller Positionen in einem Spiel ist die Spielwelt.

In Spielen interagieren die Spieler (Akteure) miteinander unter der Anwendung von Regeln und dem Einsatz von Ressourcen. Hinske u. a. (2007) haben das Modell von Spielen nach Klabbers (2003) durch die sechs Schlüsselelemente eines Spiels erweitert und dadurch den Bereich der Regeln ergänzt (siehe Abbildung 2.7).

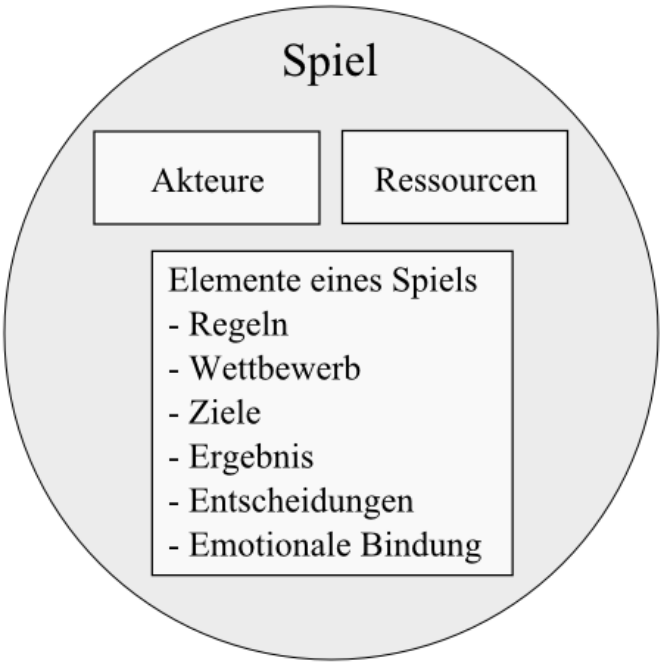

Abbildung 2.7.: Die drei Bereiche eines Spiels nach Hinske u. a. (2007): Akteure, Ressourcen und die sechs Elemente eines Spiels

Mit Hilfe des Erweiterung des Modells von Klabbers (2003) durch Hinske u. a. (2007) und den dadurch identifizierten Anforderungen an ein Spiel kann im weiteren Verlauf dieser Arbeit die Eignung des Szenarios als Spiel untersucht und bewertet werden.

Neben dem Spielbegriff ist das Spielerlebnis und die Eintauchmöglichkeiten in das Spiel durch den Spieler ein zentraler Bestandteil bei der Gestaltung von Spielen. Der Spieler muss in das Spiel einfach eintauchen und so in den *magischen Kreis* (engl. the magic circle) eintreten können (Salen und Zimmerman, 2003). Der *magische Kreis* ist ein von Huizinga (1938) geprägter Begriff und beschreibt die Spielwelt als eine temporäre Welt innerhalb der gewöhnlichen Welt, wobei die Grenze des Kreises die Trennung zwischen den beiden Welten symbolisiert. Das Flow-Modell nach Csikszentmihalyi (1990) ist ein erster Ansatz, damit der Spieler in den *magischen Kreis* eintreten und ein tiefes Spielerlebnis erfahren kann.

Bei Flow handelt es sich nach Csikszentmihalyi (1990) um befriedigende Erfahrungen, die durch das vollständige Aufgehen in einer Tätigkeit selbst erreicht werden und bei der das Ergebnis dieser Tätigkeit nicht ausschließlich im Vordergrund steht. Flow setzt sich dabei aus acht Elementen zusammen (Csikszentmihalyi, 1990):

- Eine Aktivität, die abgeschlossen werden kann.

- Die Fähigkeit, sich auf die Aktivität zu konzentrieren.
- Die Aktivität hat klare Ziele.
- Die Aktivität gibt unmittelbare Rückmeldung.
- Das Gefühl der Kontrolle über Aktionen.
- Die Sorgen des alltäglichen Lebens verschwinden.
- Die Zielsetzung liegt in der Handlung selbst.
- Das Zeitgefühl verändert sich.

Durch die Kombination dieser acht Elemente kann nach Csikszentmihalyi (1990) ein Gefühl der totalen Befriedigung eintreten, wobei die Anforderungen an die Fähigkeiten der Person und die mit der Aktivität verbundenen Herausforderungen ein gewisses Niveau erreichen müssen.

Aus dem Flow-Modell nach Csikszentmihalyi (1990) und dem Nachweis von Flow-Erlebnissen in Spielen haben Sweetser und Wyeth (2005) das GameFlow-Modell für Spiele entwickelt, das sich wiederum aus acht Kernelementen zusammensetzt:

Konzentration - Spiele sollten Konzentration benötigen, und der Spieler sollte sich auf das Spiel konzentrieren können.

Herausforderungen - Spiele sollten den Spieler ausreichend herausfordern und den Fähigkeiten des Spielers entsprechen.

Spielerfertigkeiten - Spiele müssen die Entwicklung und die Beherrschung der Fähigkeiten des Spielers unterstützen.

Kontrolle - Spieler sollten ein Gefühl der Kontrolle über ihre Tätigkeiten im Spiel haben.

Eindeutige Ziele - Spiele sollten dem Spieler jederzeit eindeutige Ziele bereitstellen.

Feedback - Spieler müssen durchgängig Rückmeldungen über ihren Fortschritt erhalten.

Immersion - Spieler sollten das Spiel durch tiefe, aber mühelose Einbindung erleben.

Solziale Interaktion - Spiele sollten Gelegenheiten für die soziale Interaktion ermöglichen und unterstützen.

Durch das GameFlow-Modell können nach Sweetser und Wyeth (2005) Spiele bewertet und validiert werden. Zusammen mit dem erweiterten Modell der drei Bereiche eines Spiels von Hinske u. a. (2007) (Abbildung 2.7) kann im Rahmen dieser Arbeit die Eignung einer Spielidee als Spiel als mögliches Szenario untersucht und sichergestellt werden.

2.5.2. Ortsabhängige Spiele

Ortsabhängige Spiele sind eine spezielle Klasse von pervasiven Spielen und kontextbewussten Systemen, bei denen der Spielverlauf von der Position des Spielers oder aber von bestimmten Orten auf dem Spielfeld abhängig ist. Spiele aus dieser Klasse von pervasiven Spielen können wiederum durch eine systematische Methode anhand der Technologien, des Spielkonzeptes oder von Raum und Zeit charakterisiert werden, wobei die einzelnen Dimensionen orthogonal zueinander sind und somit jede Kombination möglich ist (Kiefer u. a. (2006) und Kiefer u. a. (2007)):

Integration in die Spielwelt - Diese Dimension beschäftigt sich mit der Integration der Spielwelt in die physikalische Umgebung des Spielers. Dabei kann zwischen drei Möglichkeiten unterschieden werden, die sich in den Klassen 'reine ortsabhängige Spiele' (engl. pure location based games (LBG)), 'ortsabhängige Spiele in einer vermischten Realität' (engl. mixed reality location based games (MR)) und ortsabhängige Spiele in einer angereicherten Realität (engl. augmented reality location based games (AR)) widerspiegeln. Für die Begriffe existiert ein einheitliches Verständnis, obwohl es keine präzise Definition für ortsabhängige Spiele gibt. Die Eigenschaften der einzelnen Klassen können wie folgt zusammengefasst werden:

Ortsabhängige Spiele sind Spiele, die durch Technologien für die Positionsbestimmung unterstützt werden und die Position der Spieler als ein Hauptbestandteil in den Spielregeln integriert wird.

Ortsabhängige Spiele in einer vermischten Realität fügen der physikalischen Welt eine virtuelle Schicht hinzu, die durch kognitives Denken und die eigene Vorstellungskraft eingebettet wird.

Ortsabhängige Spiele in einer angereicherten Realität sind eine Untermenge der ortsabhängigen Spiele in einer vermischten Realität, bei der die virtuelle Spielumgebung nicht durch die eigene Vorstellungskraft erzeugt und z.B. durch mobile Endgeräte unterstützt werden muss, sondern aus der Ich-Perspektive erlebt wird (z.B. durch ein am Kopf befestigtes Display (engl. head mounted display (HMD))).

Spielkonzept - Spielkonzepte sind für ortsabhängige Spiele spezifischer als die normalerweise verwendeten Genres für Spiele für den Computer oder die Spielkonsole, z.B. Abenteuerspiele und Simulationen, und allgemeiner als Entwurfsmuster für das Design von Spielen (Björk u. a. (2003) und O. Davidsson (2004)). Im Gegensatz dazu orientiert sich das Spielkonzept von Kiefer u. a. (2007) an den Fähigkeiten des Spielers, die für die Erfüllung der Aufgaben und zum Gewinnen des Spiels benötigt werden. Ortsabhängige Spiele können Instanzen von einem oder mehrerer der folgenden Spielkonzepte darstellen:

Jagdspiele - Bei dem Spielkonzept von Jagdspielen spielen die physischen Eigenschaften der Spieler eine Hauptrolle zum Gewinnen.

Jagd nach Gegenständen - Bei der Jagd nach Gegenständen werden Gegenstände in der natürlichen Umgebung auf dem Spielfeld versteckt (z.B. in einem Behälter unter einem Strauch) und müssen dann durch die Spieler gefunden werden.

Puzzlespiele - Instanzen von diesem Spielkonzept werden durch das Lösen von Puzzles gewonnen. Die Puzzle können eine Wissensfragen beinhalten, aber auch komplexe Handlungen, wie z.B. in Abenteuerspielen, abdecken.

Strategiespiele - Strategiespiele werden durch überlegene Planungsfähigkeiten gewonnen.

Raum und Zeit - Die Dimension von Raum und Zeit beschreibt ein Spiel nach dem Zeitpunkt und dem Ort, an dem die für das Spiel relevanten Aktionen stattfinden. Dabei sind folgende Kombinationen möglich:

Orts- und zeitdiskret - Für das Spiel relevante Aktionen werden nicht an beliebigen Orten auf dem Spielfeld ausgelöst, sondern nur an vordefinierten Orten. Gleichzeitig darf sich der Spieler innerhalb des Spiels nur bewegen, wenn dem Spieler dies durch Spiel erlaubt wurde.

Ortsdiskret, aber zeitkontinuierlich - Für das Spiel relevante Aktionen sind auf vordefinierte Orte auf dem Spielfeld begrenzt, aber der Spieler kann selbst den Zeitpunkt bestimmen, an dem diese Orte aufgesucht werden sollen. Zeitkontinuierliche Spiele sind also nicht rundenbasiert.

Orts- und zeitkontinuierlich - Für das Spiel relevante Aktionen können an jedem beliebigen Ort auf dem Spielfeld und zu jedem Zeitpunkt ausgelöst werden.

Ortskontinuierlich, aber zeitdiskret - Für das Spiel relevante Aktionen können an jedem Ort auf dem Spielfeld ausgelöst werden, aber nur zu bestimmten Zeitpunkten (rundenbasiert).

Ortsabhängige Spiele können auch eine Mischform der vorgestellten Dimensionen darstellen, wie z.B. *Human Pacman* (siehe Abschnitt 2.5.4), bei dem einige Aktionen auf bestimmte Orte beschränkt sind, wohingegen andere Aktionen an beliebigen Orten stattfinden können.

Ortsabhängige Spiele können auch einfach über die Art der Nutzung der Ortsinformationen klassifiziert werden (Nicklas u. a., 2001):

Mobile Spiele - Bei mobilen Spielen können Ereignisse nur auftreten, wenn sich zwei Spieler treffen. Die exakte Position der Spieler wird nicht benötigt, sondern nur die Möglichkeit, mit Sensoren die Umgebung untersuchen und lokal kommunizieren zu können.

Ortsbewusste Spiele - Bei ortsbewussten Spielen ist die geografische Position der Spieler von Bedeutung. Ereignisse im Spiel können nur durch den Besuch von bestimmten Orten ausgeslöst werden.

Räumlich bewusste Spiele - Bei räumlich bewussten Spielen wird die physikalische Welt in das Spiel integriert. Gebäude, Straßen und die Landschaft sind z.B. im Spiel verfügbar. Ereignisse im Spiel werden durch die Anwesenheit des Spielers in einem bestimmten räumlichen Kontext ausgeslöst (z.B. beim Betreten irgendeiner Kirche).

Auch die Klassen in dieser Klassifikation sind orthogonal zueinander. Ortsabhängige Spiele können somit in eine oder mehrere Klassen eingeordnet werden.

Durch diese Klassifikationsmöglichkeiten können ortsabhängige Spiele eindeutig klassifiziert werden. Bei der Entwicklung eines Rahmenwerks für pervasive Spiele kann somit die benötigte Funktionalität bei einem ortsabhängigen Spiel an den Anforderungen der entsprechenden Klasse ausgerichtet werden.

2.5.3. Unterstützung durch pervasives Rechnen

Nachdem der Begriff des pervasiven Spielens in Abschnitt 2.5 definiert und in Abschnitt 2.5.1 Spiele und das Spielerlebnis betrachtet wurden, werden in diesem Abschnitt die Möglichkeiten untersucht, wie die virtuelle Welt mit der physikalischen Welt durch den Einsatz von Informations- und Kommunikationstechnologien aus dem Bereich des pervasiven Rechnens angereichert werden kann, damit der Spieler ein tiefes Spielerlebnis erfährt.

Jegers (2007) hat das GameFlow-Modell von Sweetser und Wyeth (2005) zu einem pervasiven GameFlow-Modell erweitert und damit einen Ansatz für die Integration von pervasiven Elementen in Spielen vorgestellt (siehe Tabelle 2.1).

Aspekte	**Unterstützung durch pervasives Rechnen**
Konzentration	Pervasive Spiele sollten die Spieler beim Wechseln zwischen Aufgaben in Spielen und den umgebenden Faktoren unterstützen.
Herausforderung	Pervasive Spiele sollten die Spieler stimulieren und sie bei der Gestaltung der Spielszenarien und dem Tempo der Durchführung unterstützen. Pervasive Spiele sollten den Spielern helfen, die Balance bei den Wegen und der eigenen Entwicklung in der Spielwelt zu finden, dabei aber nicht zu viel Kontrolle oder Einschränkungen beim Tempo der Durchführung und der Herausforderung entwickeln zu lassen.
Spielerfertigkeiten	Pervasive Spiele sollten flexibel sein und die Geschwindigkeit der Entwicklung der Fertigkeiten des Spielers durch den Spieler selbst festlegen lassen.
Kontrolle	Pervasive Spiele sollten den Spielern einen einfachen Zugang zu einem andauernden Spiel ermöglichen und einen schnellen Überblick über den Zustand der Spielwelt erlauben, um den Zustand zu bewerten, wie sich das Spiel entwickelt hat, nachdem der Spieler zuletzt die Spielwelt besucht hat.
Eindeutige Ziele	Pervasive Spiele sollten die Spieler bei der Erstellung und Kommunikation von Teilzielen unterstützen.
Immersion	Pervasive Spiele sollten einen nahtlosen Übergang zwischen den alltäglichen Kontexten unterstützen und keine Aktionen des Spielers voraussetzen oder erwarten, die die Verletzung von sozialen Normen im alltäglichen Kontext bedeuten würden. Pervasive Spiele sollten dem Spieler das Ändern des Fokus' zwischen den virtuellen und den physikalischen Teilen des Spiels ermöglichen, ohne zu viel Gefühl der Vertiefung zu verlieren.
Solziale Interaktion	Pervasive Spiele sollten die Möglichkeiten für spielorientierte, bedeutsame und gezielte soziale Interaktion innerhalb des Spielsystems unterstützen und ermöglichen. Pervasive Spiele sollten Strukturen und Auslöser bereitstellen (z.B. Aufgaben und Ereignisse, Interessengruppen, Gilden oder Gruppen), die die Spieler zur Kommunikation und sozialen Interaktion motivieren.

Tabelle 2.1.: Unterstützung von Aspekten im GameFlow-Modell durch pervasives Rechnen nach Sweetser und Wyeth (2005)

Nach Hinske u. a. (2007) müssen für eine fesselnde Spielerfahrung alle diese aufgeführten Aspekte unterstützt werden. Zusätzlich haben Hinske u. a. (2007) das vorgestellte Modell von Spielen aus Abschnitt 2.5.1 auf die Möglichkeiten der Unterstützung durch das pervasive Rechnen untersucht (siehe Tabelle 2.2).

Elemente	**Unterstützung durch pervasives Rechnen**
Regeln	Pervasive Spiele sollten das Spiel unauffällig, aber fortwährend beobachten, die Regeln überwachen und den aktuellen Zustand des Spiels jederzeit kennen. Der Zustand des Spiels muss jederzeit für alle Spieler zugänglich sein, und die Verletzung von Regeln sollte sofort angemessen gemeldet werden.
Wettbewerb	Pervasive Spiele sollten die Spieler zu einem fairen Wettbewerb verpflichten.
Ziele	Siehe Tabelle 2.1
Ergebnis	Pervasive Spiele sollten zu jeder Zeit den Spielstand des Spiels registrieren. Die Spieler müssen zu jedem Zeitpunkt auf den aktuellen Spielstand zugreifen können.
Entscheidungen	Pervasive Spiele müssen das Treffen von Entscheidungen durch die Spieler jederzeit erlauben. Dafür ist eine unauffällige Überwachung der Entscheidungen oder der Eingaben des Spielers notwendig.
Emotionale Bindung	Pervasive Spiele sollten eine fesselnde Erfahrung für die Spieler ermöglichen, die nahtlos z.B. unterschiedliche Medien und multimodale Endgeräte kombiniert, um physikalische, intellektuelle und soziale Erfahrungen und Herausforderungen sowie eine gute Immersion zu realisieren.

Tabelle 2.2.: Die sechs Schlüsselelemente eines Spiels mit Bezug zum pervasiven Rechnen nach Hinske u. a. (2007)

Auffällig ist, dass Hinske u. a. (2007) die Bereiche der Akteure und der Ressourcen nicht weiter aufgeführt haben. Nach Hinske u. a. (2007) ist es schwierig, Ratschläge zu geben, wie Spieler und Ressourcen unterstützt werden können, da diese Punkte vom jeweiligen Spiel abhängig sind. Als Beispiel werden Sportspiele angeführt, die z.B. durch Elemente des tragbaren Rechnens (engl. wearable computing) unterstützt werden könnten.

Durch das pervasive GameFlow-Modell (siehe Tabelle 2.1) und die Unterstützungsmöglichkeiten der sechs Schlüsselelemente von Spielen durch pervasives Rechnen (siehe Tabel-

le 2.2) können Spielideen konkret auf die Eignung als pervasives Spiel hin analysiert und bewertet werden.

2.5.4. Existierende pervasive Spiele

Im Bereich der Forschung existieren unterschiedliche Ansätze und Ausprägungen von pervasiven Spielen, von denen in diesem Abschnitt einige beispielhaft angeführt werden, um das breite Spektrum an Realisierungsmöglichkeiten aufzuzeigen.

Insectopia

Bei *Insectopia* (Peitz u. a., 2007) treten die Spieler gegeneinander an, um die wertvollste Sammlung an Insekten zusammenzustellen. Die Insekten für die Sammlung können die Spieler mit ihren Mobiltelefonen finden. Das Spiel benutzt den Bluetooth Entdeckungsmechanismus, um Bluetooth-Geräte zu finden, die als Brutstätte für die Insekten dienen. Durch das Fangen von Insekten und das Tauschen mit anderen Spielern können die Spieler ihre eigene Sammlung stetig erweitern und verbessern. Wenn ein Spieler nach Insekten sucht, werden durch das Spiel alle Insekten in der Umgebung des Spielers angezeigt, von denen eines gefangen werden kann. Die Insekten sind dabei unterschiedlich wertvoll und selten. Nachdem die Spieler eine Sammlung von Insekten besitzen, muss diese stetig aufgefrischt werden, da die Insekten nach acht Tagen eingehen, es seidenn die Spieler besuchen den Ort, wo das jeweilige Insekt erstmalig gefunden wurde. Dadurch können neue Spieler dem Spiel ohne Nachteile beitreten, da innerhalb von einer Woche die Vorteile einer bereits bestehenden Sammlung von Insekten durch das nötige Auffrischen verringert werden.

Human Pacman

Human Pacman (Cheok u. a., 2004) ist eine Realisierung des bekannten Arcarde Klassikers aus den 80er Jahren als pervasives Spiel und wurde vom Mixed Reality Labor an der Universität Signapore[2] entwickelt. In der traditionellen Version navigiert ein Spieler seine Spielfigur (Pacman) durch ein Labyrinth und versucht dabei, möglichst viele Punkte aufzusammeln, die überall im Labyrinth verteilt sind. Dabei wird der Spieler von Gespenstern verfolgt, denen der Spieler ausweichen muss. Durch das Aufsammeln von besonderen Punkten kann ein Spieler selbst für einen gewissen Zeitraum die Gespenster verfolgen. Eine Spielrunde ist gewonnen, wenn alle Punkte in einem Labyrinth aufgesammelt worden sind.

[2] http://www.mixedrealitylab.org/

Das Spielprinzip wurde von den Forschern auf die physikalische Welt übertragen. Die Spielfiguren (Pacman und Gespenster) werden von realen Spielern dargestellt, die sich mit Informationstechnologien angereichert in der physikalischen Welt frei bewegen können (siehe Abbildung 2.8).

Abbildung 2.8.: Ausrüstung der Spieler in Human Pacman (Cheok u. a., 2004)

Über ein am Kopf des Spielers befestigtes Display (engl. head-mounted display (HMD)) bekommt der Spieler die virtuellen Objekte der Spielwelt (z.B. die Punkte und die Geister) vermischt mit dem Bild der physikalischen Welt angezeigt (engl. mixed reality)(siehe Abbildung 2.9).

Abbildung 2.9.: Perspektive des Spielers (Cheok u. a., 2004)

Das Fangen wird durch eine Berührung an der Schulter symbolisiert und durch Sensoren am Spieler wahrgenommen. Die Positionen der Spieler werden von einem zentralen Server mittels Positionierungsmechanismen und drahtloser Funkübertragung stetig überwacht. Zusätzlich zu den Spielern in der realen Welt existieren noch so genannte Helfer, die einen

virtuellen Blick auf das Spiel haben und den Spielern helfen können. Dadurch wird die Interaktion und die Kommunikation zwischen den Spielern untereinander und den Spielern und den Helfern erhöht.

Physikalische Interaktionen zwischen Spielern ist traditionell aus klassischen Spielen wie z.B. Fangen oder Verstecken bekannt und wird im Bereich der Computerspiele wieder neu zum Leben erweckt (Cheok u. a., 2004).

GeoTicTacToe

In Schlieder u. a. (2006) wird das klassische Zwei-Personen-Strategiespiel *Tic Tac Toe* als ortsabhängiges Spiel auf die physikalische Welt übertragen. Ursprünglich machen zwei Spieler auf einem 3x3 Felder großen Spielfeld abwechselnd entweder Kreuze oder Kreise. Es gewinnt der Spieler, der als erstes drei seiner eigenen Zeichen horizontal, vertikal oder diagonal setzen kann. Wenn das keinem Spieler gelingt, kommt es zu einem Unentschieden. Bei der pervasiven Version *GeoTicTacToe* werden die einzelnen Felder auf Orte in der physikalischen Welt übertragen, die ein Spieler durch seine physische Fortbewegung erreichen muss. Erreicht ein Spieler ein Feld, wird das Feld, sofern bisher nicht durch den Gegenspieler bereits markiert, mit dem eigenen Zeichen versehen. Auf einem mobilen Endgerät bekommen die Spieler das Spielfeld mit den Feldern auf einer Karte angezeigt.

Daneben gibt es noch weitere pervasive Spiele wie z.B. *Pirates!* (Björk u. a., 2001), *Mobile Chase* (Fetter u. a., 2007) oder *Can you see me now?* (Benford u. a., 2006), die aber an dieser Stelle nicht weiter betrachtet werden.

2.6. Zusammenfassung

Pervasive Spiele ermöglichen ein neues Spielvergnügen und neue Möglichkeiten in der Gestaltung von Spielen durch das Verbinden der virtuellen mit der physikalischen Welt. Als Teildisziplin des pervasiven Rechnens müssen bei der Entwicklung von pervasiven Spielen eine Vielzahl von Problemen und Herausforderungen des pervasiven und kontextbewussten Rechnens beachtet werden. Da die Probleme und Herausforderungen des pervasiven und kontextbewussten Rechnens bei der Entwicklung von pervasiven Spielen nicht domänenspezifisch sind (wie z.B. die Wahrnehmung der Umgebung durch eine Anwendung oder die Kommunikation in einer verteilten pervasiven Umgebung), bieten sich Rahmenwerke an, die ein Konzept für die Wiederverwendung von Architekturentwürfen darstellen. Dadurch könnten Spielideen schneller umgesetzt und evaluiert werden, da der Fokus ausschließlich auf die Realisierung einer Spielidee gelegt werden kann.

Im folgenden Kapitel wird ein Szenario in Form einer konkreten Spielidee vorgestellt und der Spielablauf im Detail spezifiziert. Auf Basis der Erkenntnisse aus diesem Kapitel kann dann ein Rahmenwerk für pervasive Spiele entwickelt werden.

3. Szenario

Um die Entwicklung von pervasiven Spielen durch ein Rahmenwerk zu unterstützen, wird in diesem Kapitel unter Einbeziehung der Erkenntnisse aus Kapitel 2.5 das Spiel *King of Location* als Szenario für ein ortsabhängiges pervasives Spiel vorgestellt (Abschnitt 3.1) und der Spielablauf im Detail spezifiziert (Abschnitt 3.2). Damit ein Bezug auf die Spielregeln im weiteren Verlauf dieser Arbeit möglich ist, werden die Spielregeln innerhalb der Abschnitte explizit hervorgehoben und eindeutig gekennzeichnet. In Abschnitt 3.3 werden dann im Anschluss die pervasiven Elemente des Szenarios betrachtet und eine Eignung als Musterszenario im Rahmen dieser Arbeit herausgestellt.

3.1. King of Location

King of Location (KoLoc) ist ein Spiel für mehrere Spieler in der physikalischen Welt. Ziel ist es, mit einem Team, bestehend aus mehreren Spielern, möglichst viele Zielgebiete mit mehr eigenen Spielern physisch zu besetzen, als die einzelnen anderen Teams an Spielern in den Zielgebieten vorweisen können. Das Zielgebiet, in dem die Ziele lokalisiert sein können, wird durch das Spielfeld eingegrenzt und festgelegt. Als Spielfeld kann dabei jedes als geeignet erachtete Gebiet in Betracht gezogen werden, bei dem die Möglichkeit der Erreichbarkeit aller Zielgebiete in der vorgegebenen Zeit sichergestellt ist. Innerhalb eines Teams können sich die Spieler absprechen und so festlegen, welche der Zielgebiete erobert werden sollen und welches Teammitglied welches Zielgebiet besetzt. Das Spiel gewinnt das Team, welches die meisten Zielgebiete in einer vorgegebenen Zeit erobern kann.

Im Rahmen dieser Arbeit wird auf eine formale Betrachtung unter spieltheoretischen Gesichtspunkten und auf Grundlage der Spielgestalung (engl. Game Desing) bewusst verzichtet, da nicht die Spielidee den Kern dieser Arbeit darstellt, sondern die Spielidee nur als ein mögliches Szenario für die Entwicklung eines Rahmenwerks für pervasive Spiele dienen soll. Vertiefende Literatur zu diesen Themen ist z.B. in Holler und Illing (2005) und Fullerton (2008) zu finden.

Im Folgenden werden die verwendeten Begriffe im Kontext von *King of Location* spezifiziert und für ein besseres Verständnis in einen direkten Zusammenhang gebracht. Da es sich

bei *King of Location* um ein ortsabhängiges Spiel handelt, werden Ortsinformationen auf Basis von geometrischen Entitäten[3] für geografische Informationssysteme beschrieben:

Spieler - Jeder, der an *King of Location* teilnehmen möchte und sich einem Spiel angeschlossen hat, ist ein Spieler des ausgewählten Spiels und kann an einer Spielrunde teilnehmen.

Team - Die Spieler, die in einer Spielrunde zusammen in einer Gruppe spielen, sind in einem Team.

Ziele - Ein Ziel markiert eine Fläche auf dem Spielfeld, die durch einen Radius um einen definierten Punkt festgelegt wird und sich innerhalb des Spielfelds befindet. Jede Fläche innerhalb eines Spielfelds darf exklusiv nur einem Ziel zugeordnet werden, d.h. Ziele dürfen sich innerhalb eines Spielfelds nicht überlagern (Spielregel 1).

> **Spielregel 1**
> *Ein Ziel muss innerhalb des Spielfelds lokalisiert sein und darf sich nicht mit anderen Zielen überschneiden.*

Spielfeld - Ein Spielfeld beschreibt eine Fläche in der realen Welt mit einem Radius um einen definierten Punkt und enthält *x* mögliche Ziele.

Spielrunde - Eine Spielrunde umfasst den Zeitraum der Dauer eines Spiels, hat einen definierten Anfangs- und ein definierten Endzeitpunkt und kann erst gestartet werden, wenn mindestens die im Spiel spezifizierte Anzahl von Spielern *x* beigetreten und die Spieler gleichmäßig auf die Anzahl der spezifizierten Teams aufgeteilt sind (Spielregel 2). In jeder Spielrunde wird aus den möglichen Zielen eines Spielfelds zufällig eine im Spiel definierte Menge *x* als valide ausgewählt, die von den Spielern besetzt werden kann.

> **Spielregel 2**
> *Eine Spielrunde startet, sobald die für ein Spiel minimal spezifizierte Anzahl x an Spielern erreicht ist und die Spieler gleichmäßig auf die spezifizierte Anzahl x an Teams aufgeteilt sind.*

Spiel - Ein Spiel innerhalb von *King of Location* ist eine Schablone und gibt die konkreten Parameter für diese Instanz vor. Die Parameter eines Spiels legen das Spielfeld fest und bestimmen die minimale Anzahl benötigter Spieler, die Anzahl der Teams, die Dauer einer Spielrunde, die Zeit bis zum Start einer Spielrunde und die Anzahl der auszuwählenden Ziele. Zu jedem Spiel gibt es maximal eine aktive Spielrunde (Spielregel 3). Wobei *King of Location* aus *x* Spielen besteht.

[3] Eine Definition der geografischen Entitäten aus der Vektorgeometrie von geografischen Informationssystemen ist z.B. in Bartelme (2005) S. 73 ff. zu finden.

Spielregel 3
Für ein Spiel gibt es maximal eine aktive Spielrunde.

Die Erläuterungen zu den einzelnen Begriffen im Kontext von *King of Location* bilden die Grundlage für die Spezifikation des Spielablaufs (siehe Abschnitt 3.2) und ermöglichen ein einheitliches Verständnis für die Betrachtung im weiteren Verlauf dieser Arbeit.

3.2. Spielablauf

Damit eine Spielrunde in einem Spiel von *King of Location* gestartet werden kann, muss sich die in einem Spiel spezifizierte Anzahl an mindestens benötigten Spielern zusammenfinden und auf die in einem Spiel spezifizierte Anzahl an Teams aufteilen. Die Verteilung der Spieler auf die einzelnen Teams muss innerhalb einer Spielrunde gleichmäßig erfolgen, damit alle Teams die gleichen Gewinnchancen besitzen und erfolgt nach Absprache unter den Spielern einer Spielrunde. So können z.B. bestehende Teams gegeneinander antreten oder aber völlig fremde Spieler sich verabreden und dann eine Verteilung nach eigenen Kriterien (z.B. zufällig) vornehmen. Bei bestehenden Teams richtet sich die maximale Anzahl der erlaubten Spieler in einer Spielrunde nach dem Team mit den wenigsten Spielern (wobei Spielregel 2 beachtet werden muss). Daraus ergibt sich, dass unter Umständen einige Spieler eines bestehenden Teams eine Spielrunde pausieren müssen, wenn die Anzahl der Spieler des eigenen Teams die erlaubte maximale Anzahl von Spielern in einer Spielrunde erreicht oder überschritten hat.

Die Startposition der einzelnen Spieler spielt für die Ausgangssituation keine Rolle und kann von jedem Spieler beliebig gewählt werden. Damit unterschiedliche Strategien möglich sind, können sich die Spieler eines Teams absprechen und das Vorgehen für die aktuelle Spielrunde festlegen und absprechen (Spielregel 4).

Spielregel 4
Die Kommunikation zwischen Spielern ist nur innerhalb eines Teams erlaubt.

Die Spieler versuchen dann, ein valides Ziel einer Spielrunde zu erreichen und zu besetzen. Durch die Möglichkeit der Kommunikation innerhalb eines Teams hat jeder Spieler bei Bedarf vollständige Kenntnis über die Positionen der anderen Spieler seines Teams und erhält somit einen Überblick über den Spielverlauf aus der Sicht des eigenen Teams. Wenn die Zeit für eine Spielrunde abgelaufen ist, wird diese Spielrunde beendet (Spielregel 5).

Spielregel 5
Eine Spielrunde ist beendet, wenn die für das Spiel festgelegte Dauer abgelaufen ist.

Nachdem eine Spielrunde beendet ist, wird der aktuelle Status des Spiels als Momentaufnahme für die Auswertung festgehalten. Dafür werden die aktuellen Positionen der Spieler erfasst und ausgewertet. Bewegungen der Spieler auf dem Spielfeld fließen ab diesem Zeitpunkt nicht mehr in die Auswertung mit ein. Jeder Spieler einer Spielrunde wird gewertet, wenn seine letzte Position innerhalb eines Ziels angesiedelt ist (Spielregel 6).

Spielregel 6
Jeder Spieler, der ein für eine Spielrunde valides Ziel besetzt hat, wird für diese Spielrunde gewertet.

Für die Auswertung wird zunächst für jedes Ziel separat die Anzahl der Spieler aus den einzelnen Teams überprüft, und es wird festgelegt, ob ein Team dieses Ziel erobern konnte. Um ein Ziel zu erobern muss ein Team mehr eigene Spieler innerhalb dieses Ziels positioniert haben als die einzelnen anderen Teams (Spielregel 7).

Spielregel 7
Ein Ziel gilt als erobert, wenn es von mehr Spielern aus einem Team besetzt wird, als die einzelnen anderen Teams an Spielern vorweisen können.

Das Team mit den meisten eroberten Zielen ist der Gewinner der aktuellen Spielrunde und somit der *„King of Location"* (Spielregel 8). Das andere Team oder die anderen Teams haben somit verloren und erhalten keine Punkte in dieser Spielrunde.

Spielregel 8
Das Team, das die meisten Zielpunkte erobert hat, gewinnt diese Spielrunde.

Die Spieler des Teams, das gewonnen hat, bekommen für die akutelle Spielrunde x Punkte für das persönliche Punktekonto gutgeschrieben (Spielregel 9). Die Punkte für einen Sieg werden für alle Spiele in *King of Location* festgelegt und können nicht auf Spielebene definiert werden.

Spielregel 9
Die Spieler des Gewinnerteams erhalten x Punkte für das persönliche Punktekonto.

Zusätzlich erhält jeder Spieler des Gewinnerteams *x* definierte Punkte auf das persönliche Punktekonto gutgeschrieben, wenn er ein Ziel besetzt, das durch das eigene Team erobert worden ist (Spielregel 10).

Spielregel 10
Die Spieler des Gewinnerteams, die ein erobertes Ziel besetzen, erhalten zusätzlich x Punkte für das persönliche Punktekonto.

Die Punkte der aktuellen Spielrunde und die Gesamtpunktestände der einzelnen Spieler können nach der Auswertung der aktuellen Spielrunde von allen Spielern eingesehen werden. Damit ist die Spielrunde beendet und der Zyklus kann bei genügend Spielern erneut beginnen.

3.3. Pervasive Elemente

In diesem Abschnitt wird die Eignung des vorgestellten Spiels *King of Location* aus Abschnitt 3.1 als Musterszenario für die Entwicklung eines Rahmenwerks für pervasive Spiele herausgearbeitet und auf Basis der sechs identifizierten Elemente eines Spiels nach Hinske u. a. (2007) (vgl. Abschnitt 2.5.3) betrachtet. Dabei sollen die Einsatzmöglichkeiten von modernen mobilen Endgeräten im Rahmen von *King of Location* gezeigt und die Anreicherung des Spiels durch pervasive Aspekte zu einem pervasiven Spiel verdeutlicht werden. Abschließend wird *King of Location* anhand der Erkenntnisse aus dem Kapitel 2 klassifiziert.

Die identifizierten Elemente eines Spiels nach Hinske u. a. (2007) (vgl. Abschnitt 2.5.3) können wie folgt in *King of Location* mit pervasiven Elementen angereichert werden:

Regeln - Mobile Endgeräte und andere Technologien können eine ständige Überwachung des Spiels, des aktuellen Spielstands und die Einhaltung der in den Abschnitten 3.1 und 3.2 spezifizierten Regeln ermöglichen und sicherstellen. Die Kommunikation zwischen den Spielern ist z.B. nach Spielregel 4 nur innerhalb eines Teams erlaubt. Durch diese Spielregel könnten z.B. jedem Spieler die Informationen über die Positionen der anderen Spieler des eigenen Teams jederzeit zur Verfügung gestellt und

auf dem mobilen Endgerät in geeigneter Form visualisiert werden. Informationen bezüglich der aktuellen Positionen der Spieler der anderen Teams sind dann jedoch zeitgleich für diesen Spieler nicht zugänglich. Hingegen kann die Summe der Informationen z.B. von einer zentralen Instanz genutzte werden, um den aktuellen Spielstand einer Spielrunde zu überwachen und das Spielgeschehen zu beobachten.

Ziele - Durch die Unterstützung von mobilen Endgeräten könnte (z.B. der Aspekt der *Kontrolle* (siehe Jegers (2007))) der Spieler an einigen Spielrunden von *King of Location* teilnehmen und dann das Spiel verlassen. Die Position des Spielers oder des Teams in der Rangliste ist nach dem erneuten Betreten des Spiels verfügbar und könnte z.B. abrufbar sein, damit sich der Spieler über die Veränderungen während seiner Abwesenheit informieren kann. Ebenso könnte z.B. der Aspekt der *Sozialen Interaktion* durch die Möglichkeit der Kommunikation unter den Spielern eines Teams über beispielsweise einen Chat realisiert werden.

(Quantifizierbares) Ergebnis - Da nach den Spielregeln 9 und 10 die Punkte am Ende einer Spielrunde unter den Spielern des Gewinnerteams verteilt werden, könnte der aktuelle Punktestand eines Teams oder eines konkreten Spielers jederzeit in einer Spielrunde über das moderne mobile Endgerät durch einen Spieler abgerufen werden und wäre dadurch immer verfügbar. Dadurch könnten sich die Spieler oder Teams untereinander vergleichen.

Entscheidungen - Jeder Spieler hat innerhalb einer Spielrunde die Möglichkeit, ein valides Ziel auszuwählen, das er besetzen möchte. Die Auswahl könnte z.B. über ein mobiles Endgerät erfolgen und nach Spielregel 4 den anderen Spielern des eigenen Teams zur Verfügung gestellt werden. Auf dem mobilen Endgerät des Spielers, der die Auswahl getroffen hat, könnte dann als Bestätigung das Ziel in einer anderen Farbe markiert werden.

Emotionale Bindung - Durch den Einsatz von modernen mobilen Endgeräten wird die Erfahrung, die ein Spieler innerhalb von *King of Location* machen kann, gesteigert und das Spielerlebnis z.B. durch die visuelle Darstellung des Spielfelds und der aktuellen Positionen der Spieler eines Teams vertieft.

Eine Anreicherung der Bereiche der Akteure und der Ressourcen aus dem Modell (siehe Abschnitt 2.5.1 Abbildung 2.7) durch pervasive Technologien ist in *King of Location* als Szenario nicht vorgesehen. Durch Technologien aus dem Bereich des tragbaren Rechnens (engl. wearable computing) sind jedoch weitere Szenarien denkbar, die aber in dieser Arbeit nicht weiter betrachtet werden, da ausschließlich moderne mobile Endgeräte eingesetzt werden sollen (siehe Abschnitt 1.1).

King of Location kann anhand der Erkenntnisse aus Abschnitt 2.5.2 als ortsabhängiges Spiel in einer vermischten Realität als ortsdiskretes, aber zeitkontinuierliches Jagd- und Strategiespiel charakterisiert werden. Durch die Anreicherung des Spiels mit mobilen Endgeräten

wird eine virtuelle Komponente in das Spiel eingeführt. Durch die Begrenzung von Aktionen auf bestimmte Orte ist das Spiel ortsdiskret. Die Bewegungen der Spieler sind aber nicht auf bestimmte Zeitpunkte festgelegt und können in einer Spielrunde beliebig stattfinden. Demnach handelt es sich um ein zeitkontinuierliches Spiel. Zum Gewinnen sind die physischen Eigenschaften der Spieler und ein strategisches Geschick notwendig. Damit basiert das Spiel auf dem Spielkonzept eines Jagd- und Strategiespiels.

Die beispielhaft angeführten Möglichkeiten für die Integration von modernen mobilen Endgeräten in das vorgestellte Spiel *King of Location* durch die Betrachtung der grundlegenden sechs Elemente eines Spiels zeigen, dass sich durch eine Verallgemeinerung der Beispiele grundlegende und häufig benötigte pervasive Elemente identifizieren lassen (wie z.B. die Verarbeitung von Kontextinformationen (z.B. Erfassung und Verarbeitung von Ortsinformationen) und die Möglichkeit der Kommunikation unter den Spielern), die eine Eignung von *King of Location* als Musterszenario für ein Rahmenwerk für pervasive Spiele mit einem ähnlichen Spielprinzip bestätigen. Zusätzlich kann das Spiel *King of Location* eindeutig klassifiziert werden. Dadurch können die Anforderungen für die Entwicklung eines Rahmenwerks für pervasive Spiele zusätzlich von den Anforderungen der identifizierte Klasse abgeleitet werden.

3.4. Zusammenfassung

Auf Basis der grundlegenden Anforderungen und Eigenschaften von pervasiven Spielen (siehe Abschnitt 2.5) wurde in diesem Kapitel das Spiel *King of Location* als Musterszenario für die Entwicklung eines Rahmenwerks für pervasive ortsabhängige Spiele vorgestellt und die Spielregeln spezifiziert. Anschließend wurde in Abschnitt 3.3 die Eignung des Spiels als Musterszenario für diese Arbeit betrachtet und das Spiel klassifiziert.

Das Szenario ermöglicht in Kapitel 4 die Analyse im Hinblick auf die funktionalen, technischen und nicht funktionalen Anforderungen an eine Realisierung des konkreten Spiels *King of Location* und ein Rahmenwerk für pervasive ortsabhängige Spiele.

4. Analyse

Auf der Grundlage des vorgestellten pervasiven ortsabhängigen Spiels *King of Location* aus Kapitel 3 als ein mögliches Szenario werden in diesem Kapitel die konkreten Anforderungen an eine Realisierung identifiziert und spezifiziert (Abschnitt 4.1) und im Anschluss daran die Anforderungen an ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, herausgearbeit (Abschnitt 4.2). Zum Abschluss werden dann existierende Ansätze und Projekte präsentiert, die in diesem Problemfeld angesiedelt sind und mögliche Lösungen oder Teillösungen im Rahmen dieser Arbeit darstellen können (Abschnitt 4.3).

4.1. Anforderungsspezifikation des Spiels *King of Location*

Damit ein Rahmenwerk für pervasive ortsabhängige Spiele entwickelt werden kann, werden in diesem Abschnitt auf der Grundlage des Szenarios aus Kapitel 3 zu Beginn die Anwendungsfälle identifiziert und der Systemkontext festgelegt. Die Anwendungsfälle werden dann einzeln spezifiziert und anschließend die funktionalen, technischen und nichtfunktionalen Anforderungen an das Spiel *King of Location* abgeleitet. Aus diesen Anforderungen können dann in Abschnitt 4.2 die Anforderungen an ein Rahmenwerk für pervasive Spiele allgemein herausgearbeitet werden.

Die Spezifikation der Anwendungsfälle erfolgt in einer standardisierten Form mit Hilfe von Anwendungsfalldiagrammen (engl. use-case diagrams) und Aktivitätsdiagrammen (engl. activity diagrams) aus der „Unified Modeling Language“ (UML)[4] in der Version 2.1.

4.1.1. Systemkontext

Der Systemkontext (siehe Abbildung 4.1) zeigt das Spiel *King of Location* mit den identifizierten Anwendungsfällen und den beteiligten Akteuren in Form eines UML 2.1 Anwendungsfalldiagramms.

[4] http://www.uml.org/

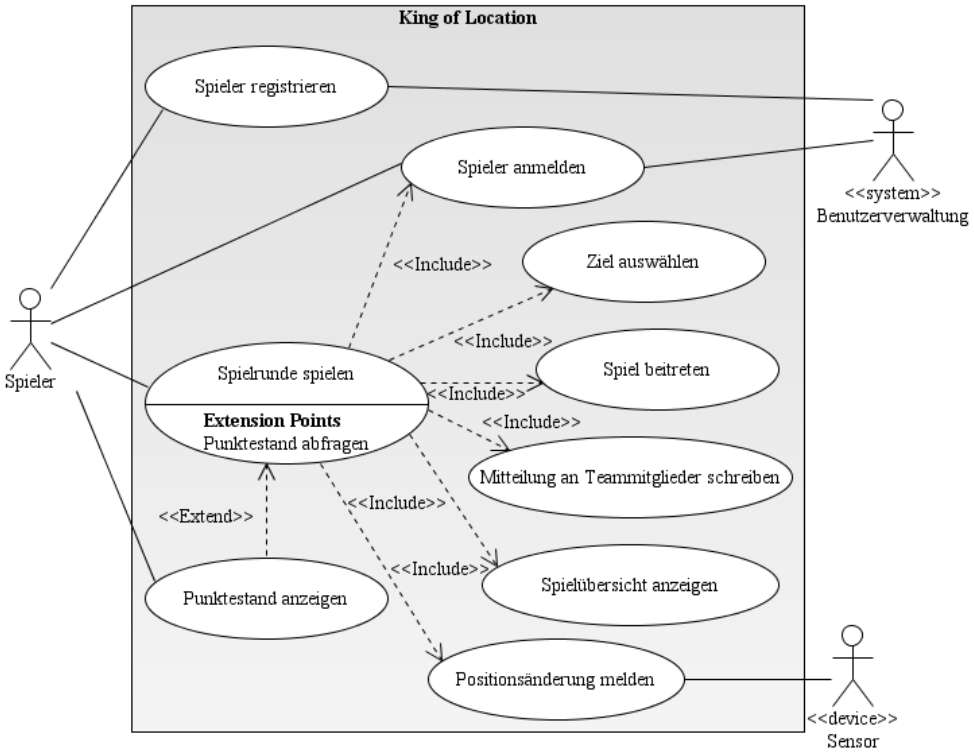

Abbildung 4.1.: Identifizierte Anwendungsfälle in KoLoc auf Basis des Szenarios

Die Anwendungsfälle und Akteure ergeben sich aus dem vorgestellten Szenario und den Möglichkeiten für die Integration von pervasiven Elementen aus Kapitel 3. Der Spieler ist der Hauptakteur und der Auslöser für die Anwendungsfälle. Der Sensor ist als eigenständiger Akteur modelliert, der die aktuelle Position des Spielers bereitstellt. Die Benutzerverwaltung ist nicht Teil des System und kann daher ebenso als Akteur modelliert werden, um eine Betrachtung als Schnittstelle zu erreichen.

Durch den Systemkontext werden die Systemgrenzen festgelegt und die Bereiche spezifiziert, die realisiert werden sollen und für die Anforderungen ermittelt werden müssen (Rupp, 2007). Gleichzeitig werden die Schnittstellen zu anderen Systemen identifiziert und dadurch eine Abgrenzung verdeutlicht.

4.1.2. Anwendungsfälle

In diesem Abschnitt wird der Hauptanwendungsfall “Spielrunde spielen“ aus dem Systemkontext aus Abbildung 4.1 für das Spiel *King of Location*, stellvertretend für alle Anwendungsfälle, durch ein Aktivitätsdiagramm verfeinert und schriftlich spezifiziert (siehe Abbildung 4.2). Die anderen Anwendungsfälle werden im Anhang B nach dem gleichen Verfah-

ren spezifiziert, damit ein konkreter Bezug zu den Anwendungsfällen bei der Spezifikation der Anforderungen hergestellt werden kann. Um die Komplexität der Anwendungsfälle und Aktivitäten auf den wesentlichen Ablauf zu begrenzen, wurden z.B. mögliche Kommunikationsprobleme in verteilten Systemen, Sicherheitsaspekte und Objektflüsse an dieser Stelle nicht explizit modelliert.

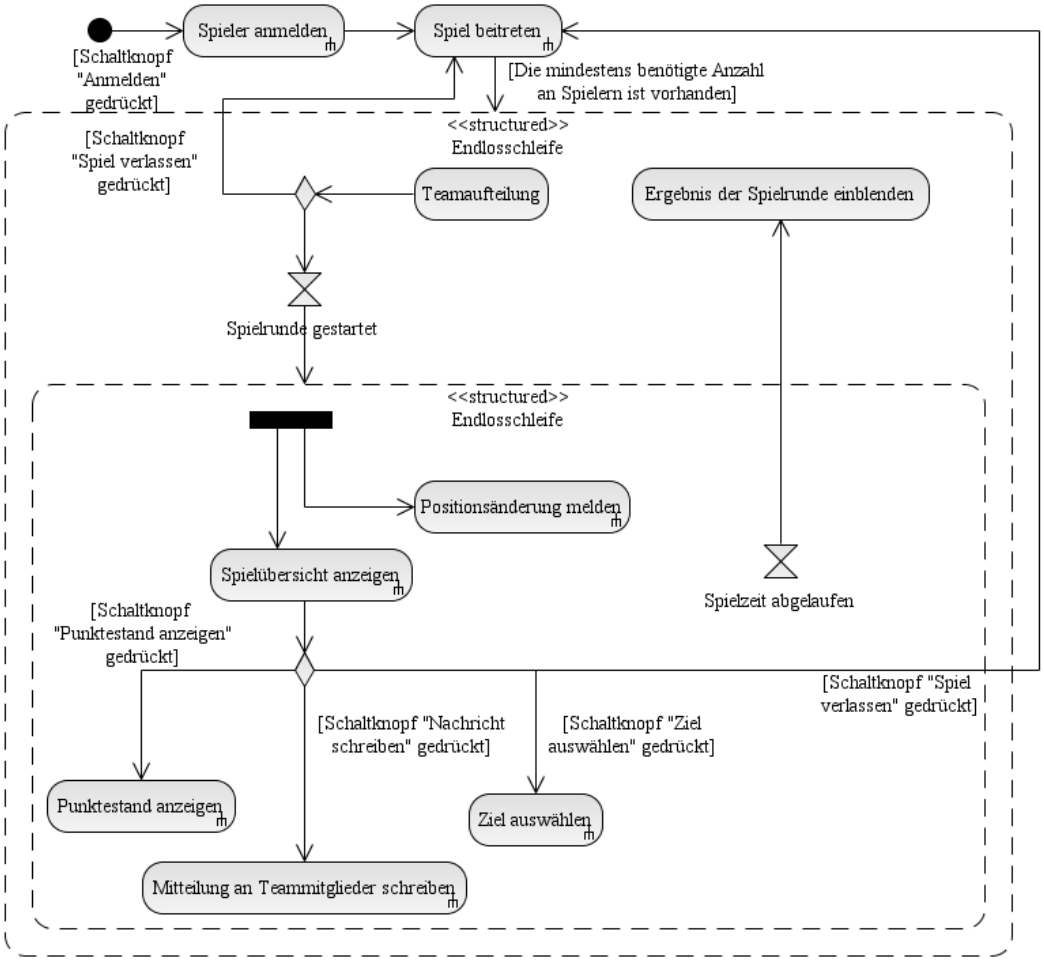

Abbildung 4.2.: Aktivitätsdiagramm zum Anwendungsfall „Spielrunde spielen“

Zusätzlich zu dem aufgeführten Aktivitätsdiagramm aus Abbildung 4.2 erfolgt eine Beschreibung des Anwendungsfalls „Spielrunde spielen“ in Form einer Tabelle (siehe Tabelle 4.1.2).

Name	**Spielrunde spielen**
Kurzbeschreibung	Der Spieler spielt eine/mehrere Spielrunde(n) des Spiels *King of Location*.

Verwendete Anwendungsfälle	Spieler anmelden, Spiel beitreten, Spielübersicht anzeigen, Mitteilung an Teammitglieder schreiben, Ziel auswählen, Positionsänderung melden.
Erweiterungspunkte	Punktestand anzeigen.
Akteure	Spieler, Sensor, Benutzerverwaltung
Vorbedingung	Der Spieler ist für das Spiel *King of Location* registriert und es stehen genügend Spieler für ein Spiel zur Verfügung.
Ergebnis	Die Spielrunde wurde beendet und das Ergebnis der Spieler persistent gespeichert.
Nachbedingung	Eine neue Spielrunde kann gestartet werden, wenn genügend Spieler für eine weitere Spielrunde zur Verfügung stehen.

Standard Ablaufschritte	1. Der Spieler meldet sich mit seinem Benutzernamen und Passwort an. 2. Der Spieler wählt ein Spiel aus einer Liste aus und tritt dem ausgewählten Spiel bei. 3. Die Spieler teilen sich in Teams ein. 4. Das Spiel startet nach einer definierten Zeit automatisch. 5. Die Spielübersicht wird angezeigt. 6. Der Spieler wählt ein valides Ziel der Spielrunde aus. 7. Der Spieler versucht, das ausgewählte Ziel physisch zu erreichen. 8. Positionsänderungen des Spielers und der Teammitglieder werden auf der Spielübersicht angezeigt. 9. Nach Ablauf der Spielzeit endet die Spielrunde. 10. Das Ergebnis der Spielrunde wird angezeigt.

Alternative Ablaufschritte	4.1. Der Spieler verlässt eine Spielrunde oder das Spiel durch das Betätigten des Schaltknopfes „Spiel verlassen“. 5. Der Spieler kann wieder ein Spiel aus der Liste mit Spielen auswählen und einem anderen Spiel beitreten. 6.1. Der Spieler verlässt die aktuelle Spielrunde durch das Betätigen des Schaltknopfes „Spiel verlassen“. 7. Der Spieler kann wieder ein Spiel aus der Liste mit Spielen auswählen und einem anderen Spiel beitreten. 6.2. Der Spieler betätigt den Schaltknopf „Punktestand anzeigen“. 7. Der Gesamtpunktestand wird dem Spieler angezeigt. 6.3. Der Spieler betätigt den Schaltknopf „Mitteilung schreiben“. 7. Der Spieler schreibt eine Mitteilung an die Spieler seines Teams.

Tabelle 4.1.: Anwendungsfallbeschreibung „Spielrunde spielen“

Die Verfeinerung der Anwendungsfälle durch Aktivitätsdiagramme und die zusätzliche textuelle Beschreibungen in Form einer Tabelle dienen in den folgenden Abschnitten als Grundlage für die Spezifikation der funktionalen, technischen und nicht-funktionalen Anforderungen.

4.1.3. Anforderungen

Auf der Grundlage der Anwendungsfälle aus Abschnitt 4.1.2 und Anhang B werden in diesem Abschnitt die funktionalen, technischen und nicht-funktionalen Anforderungen an das Spiel *King of Location* identifiziert. Die Terminologie bei der Spezifikation der Anforderungen leitet sich aus der Vorstellung des Spiels *King of Location* ab (siehe Abschnitt 3.1) und ist dadurch eindeutig definiert. Gleichzeitig ermöglicht die eindeutige Terminologie ein einheitliches Verständnis bei der Spezifikation der Anforderungen (Ebert, 2005). Die Anforderungen werden zusätzlich nach einer einheitlichen Schablone spezifiziert, die eine syntaktische Struktur festlegt (Rupp, 2007).

4.1.3.1. Funktionale Anforderungen

Die funktionalen Anforderungen werden mit einem eindeutigem Bezeichner versehen und können dadurch im weiteren Verlauf dieser Arbeit leichter referenziert werden. Die Anforderungen werden in der Reihenfolge der einzelnen Aktivitäten aus dem Aktivitätsdiagramm aus Abbildung 4.2 spezifiziert, wobei an dieser Stelle exemplarisch die funktionalen Anforderungen an die Aktivität „Spieler anmelden“ aufgeführt werden. Die funktionalen Anforderungen der anderen Aktivitäten werden im Anhang C nach dem gleichen Vorgehen spezifiziert.

Anforderungen an die Aktivität „Spieler anmelden“:

K-FA-1 [Nachdem der Spieler den Schaltknopf „Anmelden“ auf dem Startbildschirm auf dem mobilen Endgerärt betätigt hat], soll das Spiel *King of Location* dem Spieler die Möglichkeiten geben, seinen Benutzernamen und sein Passwort einzugeben.

K-FA-2 [Falls der Spieler den Schaltknopf „Abbrechen“ betätigt hat], soll das Spiel *King of Location* den Startbildschirm auf dem mobilen Endgerät anzeigen.

K-FA-3 [Nachdem der Spieler Benutzernamen und Passwort eingegeben hat und den Schaltknopf „Anmelden“ auf dem mobilen Endgerät betätigt hat], soll das Spiel *King of Location* den Benutzernamen und das Passwort des Spielers an die Benutzerverwaltung übertragen.

K-FA-4 Das Spiel *King of Location* soll fähig sein, Informationen an die Benutzerverwaltung zu übertragen und Meldungen zu empfangen.

K-FA-5 [Falls der Spieler den Schaltknopf „Anmelden“ betätigt hat und kein Benutzername und/oder kein Passwort eingegeben wurde], soll das Spiel *King of Location* eine Fehlermeldung auf dem mobilen Endgerät anzeigen.

K-FA-6 [Nachdem die Benutzerverwaltung den Benutzernamen und das Passwort von dem Spiel *King of Location* vollständig empfangen hat], soll die Benutzerverwaltung den Benutzernamen und das Passwort des Spielers validieren.

K-FA-7 [Wenn die Benutzerverwaltung den Benutzernamen und das Passwort validiert hat], soll die Benutzerverwaltung eine Erfolgs- oder Fehlermeldung an das Spiel *King of Location* übertragen.

K-FA-8 [Nachdem das Spiel *King of Location* die Erfolgs- oder Fehlermeldung vollständig von der Benutzerverwaltung empfangen hat], soll das Spiel *King of Location* diese Meldung anzeigen.

K-FA-9 [Falls das Spiel *King of Location* eine Fehlermeldung anzeigt], soll das Spiel *King of Location* dem Spieler die Möglichkeit geben, seinen Benutzernamen und sein Passwort erneut einzugeben.

4.1.3.2. Technische Anforderungen

Die technischen Anforderungen spiegeln sich in der Kategorie „Designbedingungen“ im Qualtitätsmodell der VOLERE-Methode nach Robertson und Robertson (2006) wider, stellen Anforderungen an die einzusetzende Technik im Spiel *King of Location* und machen Einschränkungen im Design.

Damit das Spiel *King of Location* von vielen Spielern gespielt werden kann, ist die Ausführung auf möglichst vielen heterogenen mobilen Endgeräten notwendig. Daraus lässt sich folgende technische Anforderung ableiten:

K-TA-1 Das Spiel *King of Location* soll auf heterogenen mobilen Endgeräten als Hardwareplattform ausgeführt werden.

Damit die funktionalen Anforderungen (wie z.B. Anforderung K-FA-4) realisiert werden können, ergeben sich dedizierte Anforderungen an die Ausstattung des mobilen Endgeräts:

K-TA-2 Das mobile Endgerät soll mit Hilfe von Sensoren fähig sein, die aktuelle Position zu bestimmen.

K-TA-3 Das mobile Endgerät soll fähig sein, mit einer zentralen Instanz im Internet zu kommunizieren.

K-TA-4 Das mobile Endgerät soll fähig sein, die Eingaben des Spielers anzunehmen und das Spiel auf einem Display darzustellen.

Für die Verteilung der Punkte nach einer abgeschlossenen Spielrunde und für die Überwachung der Spielregeln, wie z.B. in Anforderung K-FA-47 gefordert, wird eine zentrale Instanz benötigt, die die Steuerung des Spiels *King of Location* übernimmt, für die die mobilen Endgeräte jederzeit erreichbar sind und mit der die mobilen Endgeräte jederzeit kommunizieren können. Daraus ergeben sich folgende technische Anforderungen:

K-TA-5 Das Spiel *King of Location* soll durch eine zentrale Instanz im Internet überwacht und gesteuert werden.

K-TA-6 Die zentrale Instanz im Internet soll fähig sein, mit den mobilen Endgeräten der Spieler zu kommunizieren und Informationen auszutauschen.

Durch die Integration der externen Benutzerverwaltung (siehe Anwendungsfalldiagramm in Abbildung 4.1 in Abschnitt 4.1.1) für die Anmeldung und die Registrierung der Spieler ist eine Betrachtung der Anforderungen an die Schnittstelle zu diesem System notwendig. Da die Benutzerverwaltung zur Vereinfachung bewusst als externes System spezifiziert wurde, wird nur grundlegende technische Anforderung an die Schnittstelle und das System gestellt:

K-TA-7 Die zentrale Instanz im Internet soll fähig sein, mit der Benutzerverwaltung zu kommunizieren.

An dieser Stelle wird bewusst auf explizite Technologienanforderungen verzichtet, da das Spiel *King of Location* an dieser Stelle nicht auf konkrete Technologien beschränkt werden soll.

4.1.3.3. Nicht-funktionale Anforderungen

Die nicht-funktionalen Anforderungen werden wiederum nach der gleichnamigen Kategorie aus dem Qualitätsmodell der VOLERE-Methode nach Robertson und Robertson (2006) verstanden und spezifiziert. Die einzelnen Subkategorien, wie z.B. Performanz und Sicherheit, finden sich teilweise auch in ähnlicher Form z.B. in den Qualitätsmodellen des Standards 9126 der International Organization for Standardization (ISO 9126)[5] und in Wallmüller (2001) wieder, wobei Unterschiede in den Definitionen der einzelnen Eigenschaften zu finden sind (Rupp, 2007).

Operationelle Anforderungen

Bei den operationellen Anforderungen wird unter anderem die physikalische Umgebung, in der das Spiel *King of Location* ausgeführt wird, spezifiziert. Da das Spiel *King of Location*, wie bei den technischen Anforderungen in Abschnitt 4.1.3.2 gefordert, von der Kommunikation zwischen den mobilen Endgeräten und einer zentralen Instanz im Internet und der Erfassung der Position abhängig ist, kann folgende Anforderung an die physikalische Umgebung gestellt werden:

K-NFA-1 Das Spiel *King of Location* muss an Orten gespielt werden, die den mobilen Endgeräten die Möglichkeit zur Kommunikation mit der zentralen Instanz im Internet geben und die Bestimmung der Position ermöglichen.

Weitere Anforderungen im Bereich der operationellen Anforderungen, die die Distribution des Spiels *King of Location* betreffen oder den Zyklus für Veröffentlichungen festlegen, werden an dieser Stellt nicht weiter betrachtet und sind somit auch nicht Bestandteil dieser Arbeit.

Wartbarkeits- und Portierungsanforderungen

Bei den Portierungsanforderungen werden Plattformen und Umgebungen beschrieben, auf die das Spiel *King of Location* portiert werden soll. Da das Spiel *King of Location* von vielen Spielern gespielt werden soll und neben der Hardwareheterogenität bei den mobilen Endgeräten gleichzeitig unterschiedliche heterogene Programmierplattformen existieren, ergibt sich zusätzlich folgende nicht-funktionale Anforderung:

[5] http://www.iso.org

K-NFA-2 Das Spiel King of Location wird auf unterschiedlichen Programmierplattformen ausführbar sein.

In den Wartbarkeitsanforderungen dagegen wird quantitativ der Zeitraum für die Realisierung von neuen Anforderungen und der Leistungsumfang für die Unterstützung durch ein Service-Team festgelegt. Da in dieser Arbeit kein konkretes Produkt entwickelt und ausschließlich eine prototypische Realisierung vorgenommen wird, wird auf die Spezifikation von dedizierten Wartungsanforderungen verzichtet.

Leistungsanforderungen

Bei den Leistungsanforderungen werden unter anderem die Geschwindigkeits- und Latenzanforderungen für einzelne Aufgaben und Abläufe quantitativ spezifiziert. Aufgrund des prototypischen Charakters der Realisierung werden folgende weiche Anforderungen an die Geschwindigkeit und die Latenzen gestellt:

K-NFA-3 Die mobilen Endgeräte der Spieler, die zentrale Instanz im Internet und die Benutzerverwaltung sollen fähig sein, eine Reaktioneszeit < 1 Sekunde bei 95% der Anfragen zu gewährleisten.

K-NFA-4 Die Sensoren in den mobilen Endgeräten der Spieler sollen die Position in 99% der Anfragen mit einer maximalen Verzögerung von 1 Sekunde bereitstellen.

Ebenso ist eine Betrachtung der Anforderungen an die Präzision und die Exaktheit quantitativ notwendig. Besonders die Erfassung der Position des Spielers durch Sensoren auf dem mobilen Endgerät muss dafür explizit betrachtet werden:

K-NFA-5 Der Sensor im mobilen Endgerät des Spielers soll die Position des Spielers mit einer maximalen Ungenauigkeit von 25 Metern[6] bereitstellen.

Ebenso sind Anforderungen an die Zuverlässigkeit und Verfügbarkeit, an die Robustheit und die Fehlertoleranz, an die Kapazitäten und die Skalierbarkeit und Erweiterbarkeit notwen-

[6] Das entspricht der Genauigkeit des Global Positioning Systems (GPS) (Roth, 2005), das häufig in modernen mobilen Endgeräten integriert ist.

dig, damit das Spiel *King of Location*, z.B. bei Verbindungsabbrüchen bei der Kommunikation oder bei steigender Spieleranzahl (Skalierbarkeit), weiterhin für die Spieler spielbar und funktionsfähig bleibt. Die Anforderung an diese Eigenschaften wird zusammenfassend wie folgt spezifiziert:

K-NFA-6 Das Spiel *King of Location* und alle beteiligten Komponenten sollen angemessen die Spielbarkeit des Spiels durch den Spieler sicherstellen.

Die Anforderungen wurden teilweise bewusst weich formuliert, um den Aufwand bei der Verifikation der prototypischen Realisierung zu minimieren. Trotzdem muss sichergestellt werden, dass die oben aufgeführten Anforderungen z.B. an die Geschwindigkeit und die Latenzen bei der Konzeption und Realisierung berücksichtigt werden, damit die Spielbarkeit des Spiels *King of Location* gewährleistet werden kann.

Oberflächenanforderungen

Für eine mögliche prototypische Realisierung des Spiels *King of Location* werden im Rahmen dieser Arbeit keine dedizierten Anforderungen an die Erscheinung und den Stil der Oberfläche gestellt, um den Rahmen dieser Arbeit zu begrenzen.

Benutzbarkeitsanforderungen

Auch im Bereich der Benutzbarkeitsanforderungen werden keine Anforderungen im Rahmen dieser Arbeit formuliert. Sollte das Spiel *King of Location* als Produkt realisiert werden, müssten Anforderungen an die Benutzbarkeit, die Personalisierung und Internationalisierung, den Lernprozess, die Verständlichkeit und die Zugänglichkeit für Menschen mit einer Behinderung spezifizert werden.

Sicherheitsanforderungen

Bei den Sicherheitsanforderungen werden unter anderem Anforderungen an den Zugriff, die Integrität der Daten und die Privatsphäre spezifiziert. Im Rahmen dieser Arbeit wird der Aspekt der Sicherheit, wie bereits in der Themenabgrenzung in Abschnitt 1.3 geschildert, nicht explizit betrachtet.

4.1.4. Zusammenfassung

In diesem Abschnitt wurde das Spiel *King of Location* für die spätere Konzeption und Realisierung analysiert und Anwendungsfälle und Systemgrenzen identifiziert. Die Anwendungsfälle wurde dann für das bessere Verständnis als UML Anwendungsfalldiagramm dargestellt und die einzelnen Aktivitäten teilweise durch UML Aktivitätsdiagramme verfeinert. Anschließend wurden aus den Diagrammen die funktionalen, technischen und nichtfunktionalen Anforderungen entwickelt. Die Anwendungsfälle und Anforderungen werden im folgenden Abschnitt erneut analysiert, um allgemeine Anforderungen an pervasive Spiele aus der Klasse der ortsabhängigen Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, abzuleiten und für ein Rahmenwerk für pervasive Spiele zu spezifizieren.

4.2. Anforderungsspezifikation eines Rahmenwerks

Nach der Definition 1 in Abschnitt 2.1 stellt ein Rahmenwerk eine Menge von Klassen dar, die einen wiederverwendbaren Entwurf für eine Klasse von Anwendungen ermöglicht. Damit die Komplexität bei der Entwicklung von pervasiven Spielen reduziert und ein Rahmenwerk für ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, entwickelt werden kann, wird zunächst die Domäne der identifizierten Klasse und das Spiel *King of Location* erneut analysiert, eine Eingrenzung der Domäne vorgenommen, wiederverwendbare Entitäten herausgearbeitet und eine logische Modellierung vorgenommen. Die Ergebnisse aus der Domänenanalyse können im Anschluss in funktionale, technische und nicht-funktionale Anforderungen an ein Rahmenwerk überführt und spezifiziert werden, die die Grundlage für die Konzeption in Kapitel 5 darstellen.

4.2.1. Domänenanalyse

In der Domänenanalyse werden die Entitäten, die alle Applikationen in einer festgelegten Domäne gemeinsam haben, identifiziert (Karlsson, 1995). Das Ergebnis der Domänenanalyse ist eine definierte Abgrenzung der Domäne, die eine Charakterisierung der Domäne erlaubt und ein statisches Modell der elementaren Entitäten in der Domäne ermöglicht (Landin und Niklasson, 1995). Die Durchführung erfolgt durch eine Domäneneingrenzung und die anschließende Domänenmodellierung.

Domäneneingrenzung

Die Domäneneingrenzung wird durch die identifizierte Klasse des vorgestellten pervasiven Spiels *King of Location* vorgegeben. In Abschnitt 3.3 wurde das pervasive Spiel *King of Location* als ortsabhängiges Spiel in einer vermischten Realität, als ortsdiskretes, aber zeitkontinuierliches Jagd- und Strategiespiel, auf Grundlage der vorgestellten Klassifikation aus Abschnitt 2.5.2 klassifiziert. Durch diese Klassifikation wird eine konkrete Eingrenzung bei der Integration in die Spielwelt, beim Spielkonzept und bei der räumlichen und zeitlichen Dimension erreicht:

Ortsabhängige Spiele in einer vermischten Realität - Die konkrete Position des Spielers ist der elementare Hauptbestandteil bei der Integration der Spielwelt in die physikalische Umgebung des Spielers. Mobile Endgeräte unterstützen den Spieler und fügen eine virtuelle Schicht in das Spiel ein. Durch diese Eigenschaften kann eine klare Abgrenzung zu anderen pervasiven Spielen erreicht werden, die nicht ortsabhängig sind. Gleichzeitig wird zusätzlich die eingesetzte Hardware auf mobile Endgeräte beschränkt, da z.B. Technologien aus dem Bereich des tragbaren Rechnens einer anderen Klasse zugeordnet werden.

Jagd- und Strategiespiele - Durch die Beschränkung auf Jagd- und Strategiespiele besteht eine eindeutige Abgrenzung zu Spielen, bei denen Jagd auf Gegenstände in der Spielwelt gemacht wird oder bei denen Puzzles gelöst werden müssen.

Ortsdiskret, aber zeitkontinuierlich - Durch diese Einschränkung können Spiele ausgeschlossen werden, bei denen Aktionen an beliebigen Orten stattfinden können oder die rundenbasiert sind und somit als zeitdiskret eingestuft werden.

Durch diese Einschränkungen wird die zu erwartende Funktionalität und der Umfang der zu betrachtenden Eigenschaften und Entitäten, wie gefordert, eindeutig und überprüfbar eingegrenzt.

Domänenmodellierung

Auf der Grundlage der Einordnung als pervasives Spiel, der Eingrenzung der Domäne auf die identifizierte Klasse von ortsabhängigen Spielen in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, und durch das exemplarische Spiel *King of Location* als eine konkrete Instanz dieser Klasse können im Folgenden allgemeingültige Entitäten und deren Beziehungen in der Domäne herausgearbeitet werden:

Spieler - Der Spieler ist im Spiel *King of Location* der zentrale Akteur. Nach Klabbers (2003) ist der Spieler als Akteur fester Bestandteil eines Spiels und somit eine eigenständige zentrale Spielentität.

Entität - Im Spiel *King of Location* stellen Ziele Flächen mit einer eigenen Position dar, die elementare Bestandteile im Spielablauf abbilden. Diese Ziele sind somit Entitäten des Spiels, die dem Bereich der Ressourcen in einem Spiel nach Klabbers (2003) zugeordnet werden können. Dadurch markieren Spielentitäten z.B. Orte, an denen spielrelevante Ereignisse in ortsdiskreten, ortsabhängigen Spielen ausgelöst oder Aufgaben gelöst werden können.

Ereignis - Im Spiel *King of Location* sind z.B. der Start und das Ende einer Spielrunde Ereignisse im Spiel, die von der Spielverwaltung zu einem gewissen Zeitpunkt ausgelöst werden. Bei Jagdspielen können Spielereignisse durch Spielentitäten ausgelöst werden, wenn z.B. in einer pervasiven Version des Spiels "Mr. X" Mr. X durch einen Spieler oder eine Gruppe gefasst oder generell eine Aufgabe in einem pervasiven Spiel abgeschlossen worden ist. Ereignisse können auch von Aktionen, Entscheidungen und Aufgaben ausgelöst werden, wenn das Ereignis für das Spiel oder andere Entitäten im Spiel von Interesse ist.

Position - Der Spielverlauf ist im Spiel *King of Location* ausschließlich von den Positionen der Spieler auf dem Spielfeld abhängig. Diese Eigenschaft ist gleichzeitig das Hauptmerkmal der Klasse von ortsabhängigen Spielen und muss somit als zentrale Entität abgebildet werden.

Regel - Die Regeln im Spiel *King of Location* legen die Voraussetzungen und den Ablauf im Spiel fest. Da Regeln nach Hinske u. a. (2007) Schlüsselelemente in Spielen sind, sind Regeln Bestandteil auch von jedem pervasiven Spiel.

Aufgabe - Im Spiel *King of Location* haben die Spieler einer Gruppe die Aufgabe, möglichst viele Ziele mit mehr eigenen Spielern zu besetzen, als Spieler der anderen Gruppen jeweils an diesem Ziel vorhanden sind. Ebenso hat jeder Spieler die Aufgabe, ein persönliches Ziel auszuwählen und dieses dann physisch zu besetzen. In Hinske u. a. (2007) ist der Begriff der Ziele einer der sechs Schlüsselelemente eines Spiels (siehe Abschnitt 2.5.1), der als Synonym für den Begriff der Aufgabe betrachtet werden kann.

Spiel - Das Spiel *King of Location* ist für die Einhaltung der Regeln und für die Steuerung des Spielablaufs zuständig. Diese Eigenschaft wird durch die Entität „Spiel" beschrieben, da in jedem pervasiven Spiel eine Instanz benötigt wird, die diese Aufgaben übernimmt und als Container für die übrigen Bestandteile eines Spiels dient.

Gruppe - Das Spiel *King of Location* kann nur in Gruppen von mehreren Spielern gespielt werden. Die Möglichkeit, sich in Gruppen zu organisieren, ist jedoch nicht auf das Spiel *King of Location* begrenzt, sondern ist eine häufig auftretende Eigenschaft im Spielkonzept von Jagdspielen wie z.B. bei einer pervasiven Version des Spiels Mr. X.

Nachricht - Die Spieler im Spiel *King of Location* können Nachrichten in einem Team austauschen, um z.B. Strategien abzusprechen oder sonstige Absprachen zu treffen.

Diese Funktionalität ist auch bei anderen Spielen mit den gleichen Spielkonzepten denkbar, um eine Kommunikation unter den Spielern zu ermöglichen. Gleichzeitig stellt die soziale Interaktion im vorgestellten GameFlow-Modell in Tabelle 2.1 aus Abschnitt 2.5.3 einen Aspekt bei den Zielen von Elementen eines Spiels dar, die durch die Kommunikation zwischen den Spielern erreicht werden kann.

Protokoll - Im Spiel *King of Location* werden die in einer Spielrunde durch die Spieler erreichten Punkte persistent gespeichert und, z.B. zu einer Rangliste aufbereitet, dem Spieler zur Ansicht angeboten. Nach Hinske u. a. (2007) ist, wie in Abschnitt 2.5.1 beschrieben, ein quantitatives Ergebnis eines der Schlüsselelemente eines Spiels und muss daher explizit betrachtet werden. Gleichzeitig wird wiederum in Tabelle 2.1 in Abschnitt 2.5.3 bei dem Schlüsselelement des Ergebnisses der Zugriff auf den aktuellen Spielstand zu jeder Zeit als Möglichkeit der Unterstützung in Spielen durch pervasive Elemente vorgestellt.

Entscheidung - Im Spiel *King of Location* kann ein Spieler im Spiel eine Entscheidung treffen und zwischen Aktionen auswählen wie z.B. eine Nachricht an andere Spieler des gleichen Teams zu schreiben oder ein Ziel auszuwählen, das er erreichen möchte, oder aber den Punktestand anzuzeigen. Das Treffen von Entscheidungen ist nach Hinske u. a. (2007) eines der Schlüsselelemente eines Spiel und muss deshalb explizit betrachtet werden.

Aktion - Das Auswählen eines Ziels, das der Spieler im Spiel *King of Location* erreichen möchte, ist eine Aktion, für die sich der Spieler entscheiden kann. Da Entscheidungen wie bereits beschrieben eine wichtige Entität in Spielen und damit auch in pervasiven Spielen darstellen, werden die Entscheidungsmöglichkeiten als Entitäten repräsentiert, die Aktionen abbilden.

Das Ergebnis ist ein erstes logisches Modell der Domäne, das mit einer an UML Klassendiagramme angelehnten Notation beschrieben werden kann (siehe Abbildung 4.3).

Die Beziehungen zwischen den Entitäten wird durch die oben aufgeführten Beschreibungen widergespiegelt. Da es sich bei der Entität der Gruppe ebenfalls um eine Spielentität handelt, die Aufgaben erfüllen kann, aber keine Position besitzt, wird zwischen einer einfachen Spielentität und einer Spielentität mit einer Position unterschieden. Durch diese Modellierung kann eine Gruppe neben Spielern auch einfache Spielentitäten umfassen, die Aufgaben erfüllen müssen. Dadurch werden die Möglichkeiten bei der Entwicklung von pervasiven ortsabhängigen Spielen der eingegrenzten Klassen erweitert, ohne die Komplexität des Modells unnötig zu erhöhen.

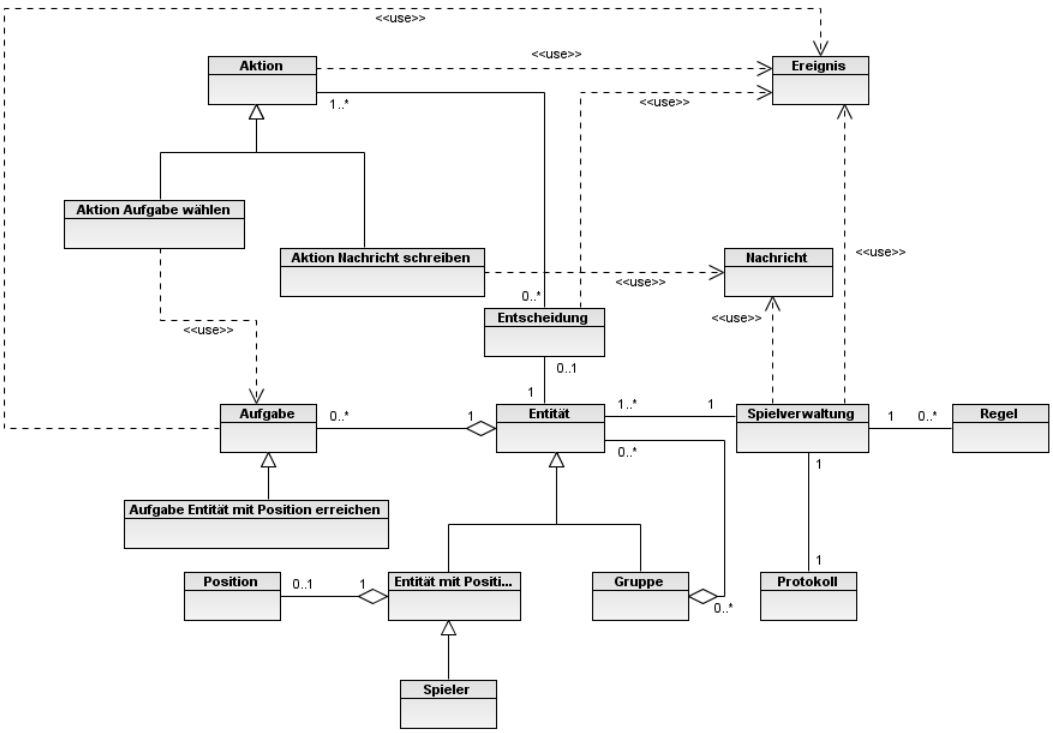

Abbildung 4.3.: Domänenmodell für ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen

4.2.2. Anforderungen

Das Domänenmodell aus der Domänenanalyse aus Abschnitt 4.2.1 und die Erkenntnisse aus der Analyse des vorgestellten Szenarios aus Abschnitt 4.1 stellen die Grundlage für die Spezifikation der Anforderungen an ein Rahmenwerk für ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Strategie- und Jagdspiele darstellen, dar. Durch die geschaffene Terminologie durch die Beschreibung der Entitäten wurde, wie bereits für die Anforderungen an das konkrete pervasive Spiel *King of Location*, ein einheitliches Verständnis für die weitere Spezifikation erreicht. Die Spezifikation der Anforderungen erfolgt nach der gleichen Schablone, die bereits in Abschnitt 4.1.3 eingesetzt wurde.

4.2.2.1. Funktionale Anforderungen

Die funktionalen Anforderungen ergeben sich aus den identifizierten Entitäten aus dem Domänenmodell und deren Beziehungen untereinander aus Abschnitt 4.2.1 oder leiten sich

aus abstrahierten Anwendungsfälle des vorgestellten konkreten Spiels King of Location aus Abschnitt 4.1 oder dessen funktionale Anforderungen ab und werden im Folgenden in abstrakter Form für einen wiederverwendbaren Entwurf spezifiziert. Die Spezifikation erfolgt in diesem Fall aus der Sicht des Rahmenwerks und der Klienten des Spiels, die das Rahmenwerk einsetzen.

R-FA-1 Das Rahmenwerk soll fähig sein, die Einhaltung der Regeln sicherzustellen, den Ablauf des Spiels durch das Auslösen von Ereignissen zu steuern und den Spielablauf und mögliche Ergebnisse zu protokollieren.

R-FA-2 Das Rahmenwerk soll fähig sein, einem Spiel Eigenschaften hinzufügen, Eingenschaften zu aktualisieren oder zu entfernen.

R-FA-3 [Wenn sich die Eigenschaften eines Spiels ändern], soll das Rahmenwerk fähig sein, Benachrichtigungen an für das jeweilige Ereignis registrierte Klienten zu übertragen.

R-FA-4 Das Rahmenwerk soll fähig sein, Nachrichten mit beliebigem Inhalt an beliebe Entitäten zu übertragen.

R-FA-5 Das Rahmenwerk soll fähig sein, Ereignisse auszulösen und Benachrichtigungen an für das jeweilige Ereignis registrierte Klienten zu übertragen.

R-FA-6 Das Rahmenwerk soll fähig sein, Aufgaben beliebigen Entitäten zuzuweisen oder Aufgaben zu entfernen.

R-FA-7 Das Rahmenwerk soll fähig sein, zum Spielverlauf passende Aufgaben für eine Entität bereitzustellen und anzubieten.

R-FA-8 [Wenn eine Entität eine/mehrere Aufgaben abschließt oder verwirft], soll das Rahmenwerk Benachrichtigungen mit dem Erfolg oder Misserfolg an für das Ereignis registrierte Klienten übertragen.

R-FA-9 Das Rahmenwerk soll fähig sein, Eigenschaften einer Entität hinzuzufügen, zu ändern oder zu entfernen.

R-FA-10 [Wenn sich der Zustand oder die Eigenschaften einer Entität ändert], soll das Rahmenwerk fähig sein, bei Bedarf Ereignisse auszulösen und Benachrichtigungen an für das Ereignis registrierte Klienten zu übertragen.

R-FA-11 Das Rahmenwerk soll fähig sein, Entitäten einer oder mehrerer Gruppen hinzuzufügen oder Entitäten aus einer Gruppe zu entfernen.

R-FA-12 [Wenn eine Entität einer oder mehrerer Gruppen hinzugefügt oder aus einer Gruppe entfernt wird], soll das Rahmenwerk fähig sein, Benachrichtigungen an für das Ereignis registrierte Klienten zu übertragen.

R-FA-13 Das Rahmenwerk soll fähig sein, die Spieler zu registrieren und zu authentifizieren.

R-FA-14 [Wenn es sich bei den Entitäten um Entitäten mit einer Position handelt], soll das Rahmenwerk fähig sein, Änderungen der Position einer Entität mit Position durch Benachrichtigungen des Ereignisses Bedarf an für das Ereignis registrierte Klienten zu übertragen.

R-FA-15 Das Rahmenwerk soll fähig sein, den aktuellen Spielstand abzufragen und bereitzustellen.

Bei der Spezifikation der fachlichen Anforderungen wurde z.B. bewusst auf die Spezifikation von konkreten Aufgaben verzichtet, um den Umfang einzugrenzen. Beispielhaft kann hier als konkrete Aufgabe das Erreichen einer konkreten Entität mit Position durch eine andere oder einen Spieler angeführt werden.

4.2.2.2. Technische Anforderungen

Die technischen Anforderungen an ein Rahmenwerk für ortsabhängige Spiele der betrachteten Klasse entsprechen den technischen Anforderungen an das Spiel *King of Location* aus Abschnitt 4.1.3.2 als eine konkrete Instanz dieser Klasse und werden deshalb nicht erneut aufgeführt.

4.2.2.3. Nicht-funktionale Anforderungen

Im Gegensatz zu den technischen Anforderungen unterscheiden sich die nicht-funktionalen Anforderungen von Rahmenwerken zu den nicht-funktionalen Anforderungen an das konkrete Spiel *King of Location*, da die nicht-funktionalen Anforderungen von Rahmenwerken mehr designorientiert sind als die nicht-funktionalen Anforderungen, die durch das konkrete Spiel *King of Location* vorgegeben werden (Landin und Niklasson, 1995). Gleichzeitig werden unterschiedliche Benutzergruppen angesprochen. Bei Rahmenwerken werden die Anforderungen durch die Anwendungsentwickler formuliert und nicht, wie beim Spiel *King of Location*, durch die Spieler des Spiels (Landin und Niklasson, 1995).

Die nicht-funktionalen Anforderungen werden erneut nach der gleichnamigen Kategorie aus dem Qualtitätsmodell der VOLERE-Methode nach Robertson und Robertson (2006) verstanden und spezifiziert.

Operationelle Anforderungen

Ebenso wie das Spiel *King of Location* können pervasive ortsabhängige Spiele der betrachteten Klasse, die auf dem Rahmenwerk basieren, nur in bestimmten physikalischen Umgebungen eingesetzt werden. Verallgemeinert ergibt sich daraus folgende operationelle Anforderung:

R-NFA-1 Pervasive ortsabhängige Spiele der betrachteten Klasse, die auf dem Rahmenwerk basieren, müssen an Orten gespielt werden, die den mobilen Endgeräten die Möglichkeit zur Kommunikation mit der Spielverwaltung und die Bestimmung der Position geben.

Wartbarkeits- und Portierungsanforderungen

Die Portierungsanforderungen an ein Rahmenwerk für pervasive ortsabhängige Spiele der betrachteten Klasse entsprechen grundlegend denen des konkreten Spiels *King of Location* und können wir folgt formuliert werden:

R-NFA-2 Das Rahmenwerk wird auf unterschiedlichen Programmierplattformen ausführbar sein.

Durch diese Portierungsanforderung können z.B. heterogene mobile Endgeräte mit heterogenen Programmierplattformen durch das Rahmenwerk unterstützt werden. Der Teil des Rahmenwerks, der auf dem mobilen Endgerät ausgeführt wird, könnte z.B. portiert werden, um eine Vielzahl von heterogenen mobilen Endgeräten zu integrieren und dadurch die Anzahl der Spieler zu erhöhen.

Da Rahmenwerke durch neue und wechselnde fachliche Anforderungen erweiterbar und modular aufgebaut sein sollten, ist die Betrachtung von Wartungsanforderungen notwendig. Besonders die Integration von neuen Aufgabentypen und Ereignissen sollte möglichst einfach realisierbar sein. Gleichzeitig sollte z.B. die Erweiterung um Funktionalitäten von pervasiven ortsabhängigen Spielen aus angrenzenden Klassen mit vertretbarem Aufwand möglich sein. Daraus ergeben sich folgenden Wartungsanforderungen:

R-NFA-3 Das Rahmenwerk soll Änderungen, Erweiterungen und die Behebung von Fehlern mit vertretbarem Aufwand ermöglichen.

Leistungsanforderungen

Die Leistungsanforderungen an das Rahmenwerk für die Entwicklung von pervasiven ortsabhängigen Spielen für die betrachtete Klasse spiegeln sich in den Leistungsanforderungen an das konkrete Spiel *King of Location* wider (vgl. K-NFA-3 bis K-NFA-6). Die Perspektive muss jedoch auf das Rahmenwerk eingegrenzt werden und betrifft in diesem Fall Spiele, die auf dem Rahmenwerk basieren.

Oberflächenanforderungen

Da es sich bei dem zu entwickelnden Rahmenwerk um ein domänenspezifisches Rahmenwerk handelt, das ein fachliches Rahmenwerk darstellt und Abläufe und Beziehungen zwischen Entitäten abbildet, werden keine expliziten Anforderungen an die Oberfläche spezifiziert.

Benutzbarkeitsanforderungen

Im Gegensatz zu den Benutzbarkeitsanforderungen an das konkrete Spiel *King of Location* müssen bei der Entwicklung zusätzliche Benutzbarkeitsanforderungen spezifiziert werden, die sich aus dem Unterschied bei der angesprochenen Benutzergruppe ergeben. Das Rahmenwerk wird von Anwendungsentwicklern eingesetzt, um konkrete ortsabhängige Spiele der betrachteten Klasse zu entwickeln und muss daher Anforderungen an die Benutzbarkeit stellen, die die Einarbeitungszeit und das Verständnis für den Einsatz bei der Entwicklung betreffen.

R-NFA-4 Das Rahmenwerk soll dem Entwickler ermöglichen, sich auf die fachliche Realisierung bei der Entwicklung von Spielen zu konzentrieren.

R-NFA-5 Das Rahmenwerk soll dem Entwickler Komponenten, Schnittstellen und abstrakte Klassen für die Entwicklung von konkreten pervasiven ortsabhängigen Spielen der betrachteten Klasse zur Verfügung stellen, die kein tiefes Verständnis für den inneren Aufbau erfordern und mit denen konkrete Spiele zügig realisiert werden können.

R-NFA-6 Das Rahmenwerk soll dem Entwickler durch den Einsatz von allgemein anerkannten Entwurfsmustern (z.B. in Gamma u. a. (1994)) und spezifizierten Komponenten ermöglichen, die Einarbeitungszeit zu minimieren.

Weitere Benutzbarkeitsanforderungen werden an dieser Stelle nicht betrachtet, da die Eigenschaften sich eher auf konkrete Produkte beziehen und nicht auf Rahmenwerke.

Sicherheitsanforderungen

Besonders die Integration von Sicherheitsaspekten sollte bei der Entwicklung durch ein Rahmenwerk erfolgen, um den Aufwand für den Entwickler zu verringern und diese Aspekte zentral und einheitlich abzubilden. Um den Umfang dieser Arbeit zu begrenzen, wird, wie in Abschnitt 1.3 beschrieben, der Bereich der Sicherheit nicht explizit betrachtet und der Fokus auf die fachlichen Aspekte gelegt.

4.2.3. Zusammenfassung

In diesem Abschnitt wurden Anforderungen an ein Rahmenwerk für die Realisierung von pervasiven ortsabhängigen Spielen in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, spezifiziert. Dafür wurde eine Domänenanalyse vorgenommen, die Domäne eingegrenzt und ein Domänenmodell durch die Analyse der elementaren Entitäten und Beziehungen aus der betrachteten Klasse von pervasiven ortsabhängigen Spielen entwickelt. Auf der Grundlage der Anforderungsanalyse für das konkrete Spiel *King of Location* aus Abschnitt 4.1 und der Domänenanalyse aus Abschnitt 4.2.1 wurden dann die funktionalen, technischen und nicht-funktionalen Anforderungen spezifiziert, die die Grundlage für die Konzeption und die Realisierung eines Rahmenwerks für die Entwicklung von pervasiven ortsabhängigen Spielen der betrachteten Klasse in den folgenden Kapiteln darstellen.

4.3. Existierende Ansätze

Auf der Grundlage der identifizierten Anforderungen an ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, werden in diesem Abschnitt existierende Ansätze betrachtet und untersucht. Neben existierenden Rahmenwerken und Infrastrukturen für pervasive Spiele werden auch benachbarte Forschungsgebiete für Teillösungen, z.B. für die Verarbeitung und Bereitstellung von Kontextinformationen, in die Untersuchung mit einbezogen, um gleichzeitig Lösungsansätze für die nicht-funktionalen Anforderungen zu betrachten.

4.3.1. Rahmenwerke und Infrastrukturen für pervasive Spiele

In diesem Abschnitt werden existierende Rahmenwerke und Infrastrukturen für die Entwicklung von pervasiven Spiele vorgestellt und auf einen möglichen Einsatz im Rahmen dieser Arbeit untersucht.

Mobile Chase

Mobile Chase (Fetter u. a., 2007) ist die Umsetzung einer klassischen Schnitzeljagd in Form eines pervasiven ortsabhängigen Spiels für mobile Endgeräte des Fraunhofer Instituts. Realisiert wurde Mobile Chase basierend auf einem domänenspezifischen Rahmenwerk für pervasive ortsabhängige Spiele, das die Entwicklung von unterschiedlichen Spielkonzepten von ortsabhängigen Spielen unterstützen und vereinfachen soll. Ein Überblick über die Architektur des Rahmenwerks ist in Abbildung 4.4 dargestellt.

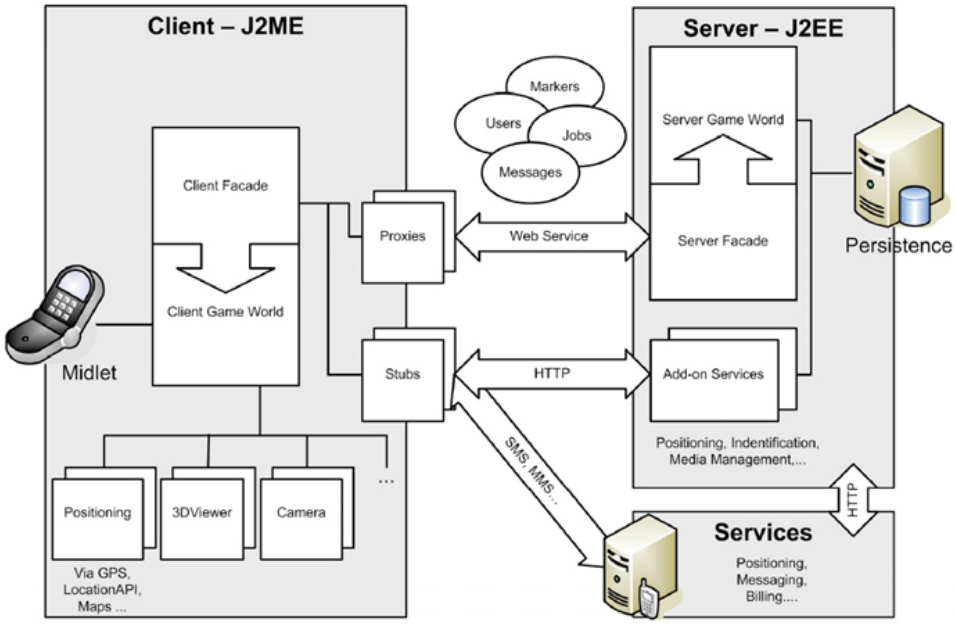

Abbildung 4.4.: Architektur des Rahmenwerks für pervasive ortsabhängige Spiele (Fetter u. a., 2007)

Grundlage dieses Ansatzes ist ein Domänenmodell, das sich aus vier grundlegenden Entitäten zusammensetzt, die im Folgenden beschrieben werden:

User - Der `User` repräsentiert den Spieler im Spiel mit einer Position, wobei durch das gewählte Konzept auch virtuelle Spieler erlaubt werden, die durch Spieler von statio-

nären Informationssystemen in der realen Welt oder durch Informationssysteme selbst gesteuert werden können.

Marker - Die `Marker` beschreiben Objekte oder Plätze mit einer für das Spiel relevanten Position in der physikalischen Welt.

Job - Ein `Job` stellt ein Ziel dar, das erreicht werden muss. Dabei kann ein `Job` einem `User` oder einem `Marker` zugeordnet werden.

Message - Durch eine `Message` können Nachrichten z.B. zwischen den Spielern eines Teams ausgetauscht werden. Gleichzeitig kann eine `Message` auch für die Kommunikation mit einem `Marker` verwendet werden, das z.B. ein Display in der physikalischen Welt darstellt, das die Nachricht anzeigen kann. Durch die gewählte Modellierung einer `Message` mit einem Textfeld können beliebige z.B. mit XML beschriebene Nachrichtenformate eingesetzt werden.

Das zentrale Element, das den Spielablauf vorgibt, ist die `GameWorld`, die als abstrakte Klasse angeboten wird und in die die Regeln des Spiels und die Spiellogik implementiert werden können. Die `GameWorld` stellt bereits grundlegende Funktionalität für den Spielablauf bereit und kann je nach Rechenaufwand auf der zentralen Instanz und/oder auf dem mobilen Endgerät erweitert werden.

Technisch ist das Rahmenwerk in Java implementiert, wobei auf den mobilen Endgeräten die Java Platform Micro Edition (Java ME)[7] und auf der zentralen Instanz die Java Platform Standard Edition (Java SE)[8] oder Java Platform Enterprise Edition (Java EE)[9] eingesetzt werden kann. Die Kommunikation zwischen den mobilen Endgeräten und der zentralen Instanz erfolgt über standardisierte Webdienste mit SOAP durch definierte Schnittstellen. Dadurch wird eine Entkopplung von der verwendeten Technologie erreicht und eine mögliche Integration von anderen Hardware- und/oder Programmierplattformen erreicht. Werden neben den bereitgestellten Schnittstellen noch andere Funktionalitäten und Dienste benötigt, können diese am Kommunikationsfluss des Rahmenwerks vorbei eingesetzt werden. Auf diese Weise können z.B. Bezahl- und/oder Geoinformationsdienste in das Spiel integriert werden.

Das vorgestellte domänenspezifische Rahmenwerk für ortsabhängige Spiele hat ein vergleichbares Ziel wie es in dieser Arbeit angestrebt wird. Durch die begrenzten Informationen ist nicht eindeutig feststellbar, ob durch das Rahmenwerk der Spielablauf unterstützt wird oder ob Konstrukte für die Modellierung von z.B. Spielregeln angeboten werden. Eine detaillierte Betrachtung dieses Ansatzes ist mit den verfügbaren Informationen nicht möglich. Dadurch kann eine mögliche Eignung im Rahmen der Problemstellung in dieser Arbeit nicht weiter untersucht werden. Das Domänenmodell im Rahmen der Domänenanalyse aus

[7] http://java.sun.com/javame/index.jsp
[8] http://java.sun.com/javase/
[9] http://java.sun.com/javaee/

Abschnitt 4.2.1 findet sich allerdings auch in einer ähnlichen Form in dem Rahmenwerk von Fetter u. a. (2007) wieder.

MUPE und MAR

Beim Mulit-User Publishing Environment (MUPE) (Suomela u. a., 2004) handelt es sich um ein Rahmenwerk und eine Plattform für mobile kontextbewusste Mehrbenutzeranwendungen, das vom Nokia Forschungszentrum in Finnland entwickelt wurde. Die MUPE Plattform basiert auf einer Client-Server Architektur, bei der die mobilen Endgeräte mit einer zentralen Instanz im Internet, dem Server, kommunizieren. Auf dem mobilen Endgerät muss dafür der MUPE Client als Anwendung installiert werden. Die konkret entwickelten Anwendungen und Spiele können anschließend über die MUPE Plattform dynamisch ohne eine explizite Installation auf dem mobilen Gerät bei Bedarf über die zentrale Instanz im Internet nachgeladen werden. Ein Überblick über die Plattformstruktur wird in Abbildung 4.5 gegeben.

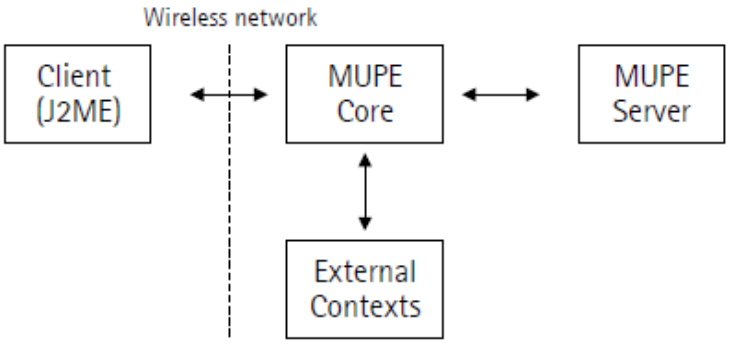

Abbildung 4.5.: Die MUPE Plattformstruktur (Suomela u. a., 2004)

Die MUPE Anwendungen und Spiele werden für den MUPE Server entwickelt, der die aktuelle Benutzeroberfläche für das mobile Endgerät in einer eigenen in XML beschriebenen Notation erstellt und an das mobile Endgerät überträgt. Externe Informationen, wie z.B. aus dem Internet, können über individuelle Kontextkomponenten integriert werden. Die MUPE Anwendungen und Spiele, die MUPE Kontextkomponenten und der MUPE Server werden über die MUPE Middleware, den MUPE Kern, miteinander verbunden.

Der MUPE Client verarbeitet die XML Beschreibungen der Benutzeroberflächen für alle Anwendungen und Spiele. In der XML Beschreibung können außerdem vorgegebene oder neu definierte Einstiegspunkte (engl. hooks) genutzt werden, damit Funktionalität auf dem mobilen Endgerät ohne explizite Kommunikation mit dem MUPE Server ausgeführt werden kann. Der MUPE Server enthält die gesamte Funktionalität und die Logik der Anwendungen und Spiele und kennt den aktuellen Zustand der Anwendung oder des Spiels und der

verbundenen Spieler. Der MUPE Client stellt eine Anfrage an den MUPE Server, der mit einer Beschreibung einer Benutzeroberfläche antwortet.

Der MUPE Server stellt ein `World` Objekt bereit, das die Spielwelt mit dem Inhalt repräsentiert. Der Inhalt setzt sich aus vier Typen zusammen, die im Folgenden beschrieben werden:

User - Der `User` repräsentiert einen einzelnen Endbenutzer im System. In Spielen ist der `User` ein Spieler.

Room - Ein `Room` ist ein Ort in der Spielwelt, wobei die Spielwelt durch einen einzigen `Room` oder durch mehrere abgebildet werden kann.

Item - Ein `Item` kann jede andere Information im Spiel darstellen, die benötigt wird.

Service - Durch einen `Service` wird zusätzliche Funktionalität wie z.B. eine Nachrichtenübermittlung zwischen Spielern in einem Spiel abgebildet.

Die `World` und die vier Typen bilden somit das Domänenmodell von MUPE, das in Abbildung 4.6 dargestellt wird.

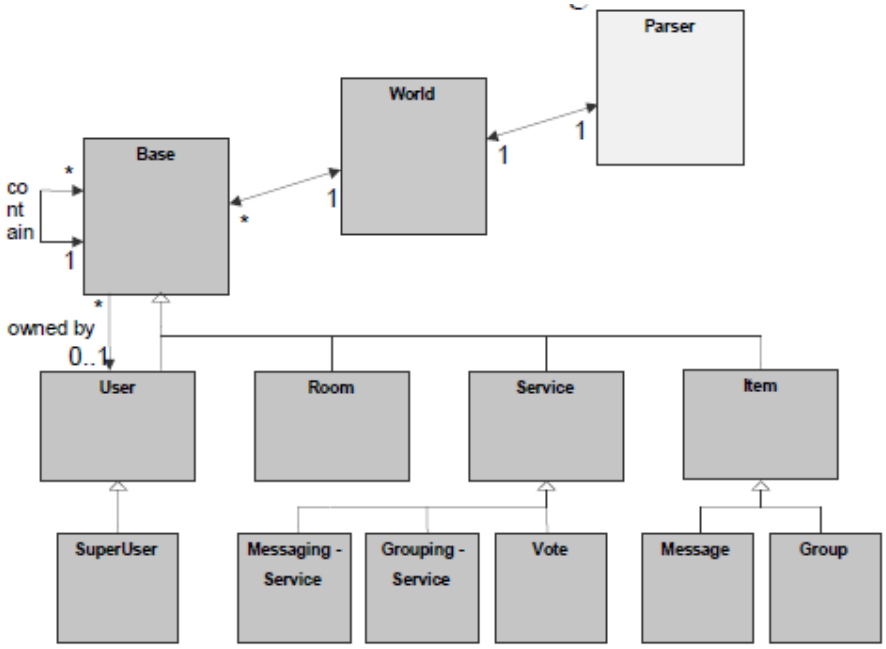

Abbildung 4.6.: Das MUPE Domänenmodell (Suomela u. a., 2004)

Neben einem grundlegenden Domänenmodell werden Ansätze zur Verarbeitung von Kontextinformationen durch das Rahmenwerk angeboten. Kontextinformationen wie z.B. die Position des Spielers können von dem MUPE Client an den MUPE Server übermittelt und

dort weiterverarbeitet werden. Gleichzeitig können auch durch den MUPE Server Kontextinformationen erzeugt und z.B. durch einen `Service` an die MUPE Clients verteilt werden.

Technisch wurde MUPE in Java implementiert, wobei auf dem mobilen Endgerät die Java Platform Micro Edition (Java ME) und für den Server die Java Platform Standard Edition (Java SE) eingesetzt wird. Der Quellcode der MUPE Plattform und des Rahmenwerks ist frei unter der Nokia Open Source License 1.0a[10] verfügbar und kann auf der Internetseite[11] heruntergeladen werden. Zur Unterstützung der Entwicklung wird eine Erweiterung für die offene Entwicklungsplattform Eclipse[12] der Eclipse Foundation bereitgestellt, mit der z.B. die XML basierte Erstellung der Benutzeroberflächen unterstützt wird.

Auf der Grundlage von MUPE wurden von Kuikkaniemi u. a. (2006) im Rahmen eines Forschungsprojektes des Instituts für Informationstechnologie in Helsinki ein Satz von Erweiterungen in Form eines Werkzeugsatzes für eine mobile angereicherte Realität (engl. mobile augmented reality (MAR)) präsentiert. MAR stellt für die Entwicklung zusätzliche Komponenten für die Darstellung und Verwendung einer Landkarte (MAP Komponente), die Markierung von physikalischen Gegenständen durch z.B. Barcodes (engl. physical object tagger (POT)), für die Darstellung des Spielstandes auf einem Display für die Spieler oder den Spielleiter (engl. public display (PUD)) und für die geräuscharme Kommunikation (engl. silent communicator (SIC)) bereit. Durch die Kombination von MUPE und MAR wird die Entwicklung von pervasiven Spielen durch eine stabile Plattform und die Bereitstellung von Komponenten unterstützt und vereinfacht.

MUPE stellt eine ausgereifte technische Plattform bereit, die durch MAR um zusätzliche Funktionalität erweitert wird. Grundlegende Entitäten können, wie im Domänenmodell in Abbildung 4.6 dargestellt, zu einer Anwendung oder einem Spiel verwoben werden. Dabei handelt es sich um einen generischen Ansatz, der aber z.B. keine Unterstützung bei der Einhaltung von Spielregeln oder beim Spielablauf bereitstellt. Die Plattform oder das Rahmenwerk lässt sich demnach als ein anwendungsspezifisches Rahmenwerk, wie in Abschnitt 2.1 beschrieben, klassifizieren und stellt aus diesem Grund keine Lösung für die Entwicklung eines domänenspezifischen Rahmenwerks für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, dar. Ein Einsatz als technische Plattform ist trotzdem denkbar und könnte im Rahmen der Konzeption in Betracht gezogen werden.

[10] http://www.opensource.org/licenses/nokia.html
[11] http://www.mupe.net/
[12] http://www.eclipse.org/

Weitere Ansätze

Neben den vorgestellten Rahmenwerken existieren noch weitere technische Infrastrukturen oder anwendungsspezifische Rahmenwerke, auf denen ein Spiel entwickelt werden kann. SNAP Mobile[13] z.B. ist eine kommerzielle Plattform von Nokia für die Integration von Gemeinschaftsfunktionalitäten in Spiele (wie z.B. Nachrichtenaustausch oder Ranglisten), die auf mobilen Endgeräten, auf denen Java Platform Micro Edition (Java ME) ausgeführt wird, genutzt werden kann. Ein ähnlicher Ansatz wird auch mit der Neutron Plattform von Exit Games[14] bereitgestellt. Ein anwendungsspezifisches Rahmenwerk für die technische Infrastruktur in Spielen wird z.B. von Greenhalgh u. a. (2007) mit der EQUIP2 Plattform vorgestellt.

4.3.2. Kontextbewusstsein

Nach Definition 3 von Kontext aus Abschnitt 2.3 sind alle Informationen Kontext, die die Ausführung eines informationsverarbeitenden Systems implizit oder explizit beeinflussen, außer den Ein- und Ausgaben selbst. Aus den funktionalen, technischen und nicht-funktionalen Anforderungen an ein Rahmenwerk für pervasive ortsabhängige Spiele der betrachteten Klasse wird ersichtlich, das auf Änderungen im Kontext reagiert werden muss, z.B. kann sich der Kontext ändern, wenn ein Spieler eine bestimmte Position erreicht hat (wie beim Szenario *King of Location*). Das Spiel reagiert in diesem Fall mit einer Aktualisierung der Benutzeroberfläche, handelt nach Definition 4 aus Abschnitt 2.3 also kontextbewusst. Aus dieser Motivation heraus wird in diesem Abschnitt ein konkreter Ansatz präsentiert, der einen generischen Ansatz darstellt und so den nicht-funktionalen Anforderungen an Rahmenwerke in Bezug auf Erweiterbarkeit und Wartbarkeit nachkommt (siehe Abschnitt 2.1).

WildCAT

WildCAT (David und Ledoux, 2005) ist ein erweiterbares Rahmenwerk für die Erstellung von kontextbewussten Anwendungen, das im Rahmen von SAFRAN (einem Komponentenmodell für die Erstellung von kontextbewussten Anwendungen) entwickelt wurde. Über einheitliche Schnittstellen können unterschiedliche Kontextquellen, wie z.B. Hardwaresensoren, integriert und die Informationen bereitgestellt werden. Die Bereitstellung der Kontextinformationen kann dabei synchron oder asynchron erfolgen. Ein Überblick über die Architektur von WildCAT ist in Abbildung 4.7 dargestellt.

[13] http://snapmobile.nokia.com/
[14] http://www.exitgames.com/

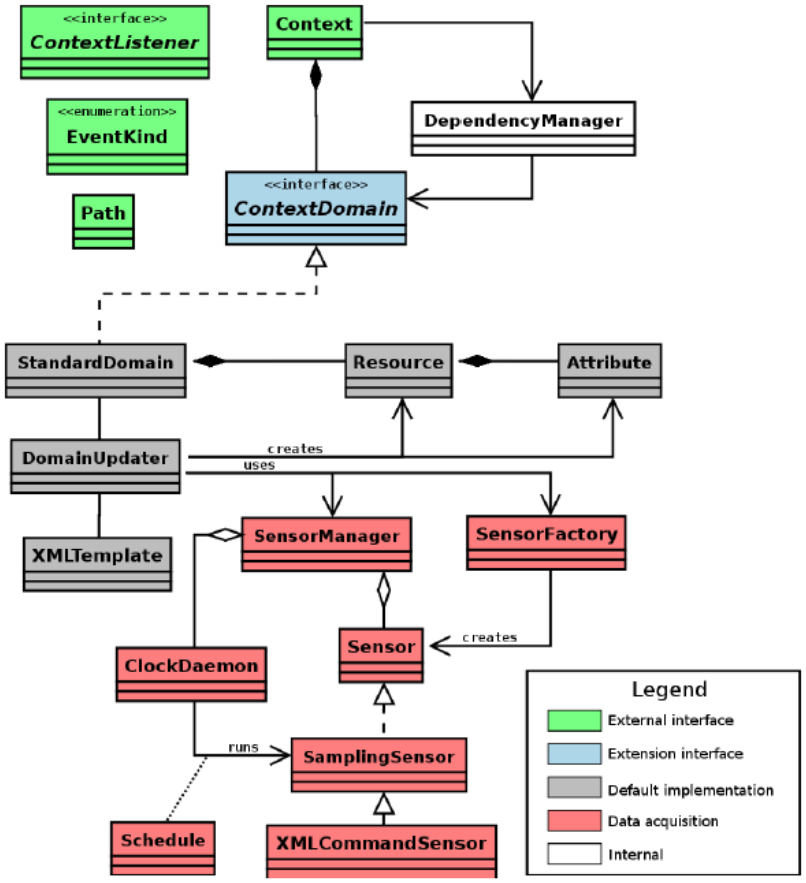

Abbildung 4.7.: Architektur von WildCAT (vgl. David und Ledoux (2005))

Der zentrale Einstiegspunkt für den Einsatz ist der `Context`, der durch eine zentrale Schnittstelle als Fassade[15] bereitgestellt wird. Der Kontext besteht aus unterschiedlichen Domänen, die in Form von unterschiedlichen `ContextDomains` integriert werden können. Die Domänen können z.B. den Benutzerkontext oder den Systemkontext abbilden. Jede `ContextDomain` beinhaltet unterschiedliche Ressourcen, die durch einen Baum abgebildet, werden und dadurch ein logisches Modell für die Anwendung darstellen. Um Kontextinformationen abzubilden werden in WildCAT einheitliche Bezeichner für Ressourcen (engl. uniform resource identifier (URI)) verwendet, die unabhängig von der jeweiligen Implementierung der `ContextDomains` sind.

Technisch ist WildCAT in Java realisiert. Der Quellcode[16] ist frei unter der GNU Library

[15] Eine Fassade ist ein Entwurfsmuster in der Entwicklung nach Gamma u. a. (1994)
[16] http://wildcat.objectweb.org

General Public License v2.1 (LGPL)[17] Lizenz verfügbar und kann heruntergeladen werden.

Bei WildCAT handelt es sich um ein anwendungsspezifisches Rahmenwerk mit der Aufgabe, die Verwaltung und Bereitstellung von Kontextinformationen zentral und generisch in Anwendungen zur Verfügung zu stellen. Im Rahmen dieser Arbeit könnte dieser oder ein ähnlicher Ansatz bei der Entwicklung eines Rahmenwerks für pervasive ortsabhängige Spiele der betrachteten Klasse helfen, die nicht-funktionalen Anforderungen, wie Erweiterbarkeit und Wartbarkeit, abzudecken und die Wiederverwendbarkeit des Entwurfs zu erhöhen.

Weitere Ansätze

Neben dem vorgestellten Ansatz gibt es in der Forschung noch weitere Rahmenwerke für kontextbewusste Anwendungen und Systeme. In Floréen u. a. (2005) wird z.B. ein Rahmenwerk für das Management von Kontextinformationen mit einem vergleichbaren Ansatz vorgestellt. Beim Hydrogen Rahmenwerk (Hofer u. a., 2003) werden Kontextinformationen über einen zentralen `ContextServer` erfasst und Anwendungen zur Verfügung gestellt. Ein Überblick über weitere existierende kontextbewusste Systeme und Rahmenwerke kann z.B. in Baldauf u. a. (2007) gefunden werden.

4.3.3. Zusammenfassung

Die vorgestellten Ansätze erfüllen die spezifizierten Anforderungen nur teilweise. In Tabelle 4.3.3 werden die vorgestellten Ansätze mit dem angestrebten eigenen Ansatz verglichen und die elementaren Eigenschaften gegenübergestellt.

Rahmenwerk Eigenschaften	**Mobile Chase**	**MUPE und MAR**	**WildCAT**	**Eigener Ansatz**
Kategorie[18]	√	- (Anwendungs-spezifisch)	- (Anwendungs-spezifisch)	√
Portierbarkeit	√	(√)	(√)	√

[17] http://www.gnu.org/licenses/lgpl.html

[18] Domänenspezifisches Rahmenwerk für pervasive Spiele, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen.

Wartbarkeit und Erweiterbarkeit	?	√	√	√
Kontextbewusstsein	(√)	√	√	√

Tabelle 4.17.: Übersicht und Vergleich der vorgestellten existierenden Ansätze mit dem eigenen angestrebten Ansatz

Beim Vergleich der Ansätze ist erkennbar, dass der Ansatz des Rahmenwerks, auf dem Mobile Chase basiert, die fachlichen Anforderungen weitestgehend erfüllt (obwohl das entwickelte Domänenmodell für den eigenen Ansatz weitere Entitäten wie z.B. Regeln umfasst) und durch den Einsatz von standardisierten Webdiensten auch portierbar scheint, kann keine Aussage über die Wartbarkeit und Erweiterbarkeit getroffen werden, da der Quellcode des Projekts für eine tiefere Analyse nicht öffentlich und frei zur Verfügung steht. Dagegen stellen MUPE und MAR einen möglichen pragmatischen Ansatz für pervasive Anwendungen und Spiele dar. Jedoch sind domänenspezifische Aspekte wie z.B. das Lösen und Überwachen von Aufgaben durch den anwendungsspezifischen Charakter des Rahmenwerks nicht berücksichtigt. Da bei der Kommunikation zwischen den mobilen Endgeräten und dem MUPE Server XML Dokumente ausgetauscht werden, ist eine Portierbarkeit grundsätzlich möglich. Die Wartbarkeit und Erweiterbarkeit ist durch den quelloffenen Quellcode und die Architektur gewährleistet. Kontextbewusste Aspekte sind explizit berücksichtigt und modelliert worden. WildCAT liefert im Gegensatz zu den beiden anderen Ansätzen keinen fachlichen Beitrag, kann aber als anwendungsspezifisches Rahmenwerk im Rahmen des eigenen Ansatzes technisch unterstützend eingesetzt werden und damit die Wartbarkeit und Erweiterbarkeit der Lösung steigern. Das ermöglicht einen allgemeineren Ansatz für die Integration von kontextbewussten Aspekten. Der eigene Ansatz soll einen ganzheitlichen Ansatz darstellen, der sowohl fachlich als auch technisch die identifizierten Anforderungen erfüllt und die gewünschten Eigenschaften der vorgestellten Projekte vereint.

4.4. Zusammenfassung

Auf der Basis des vorgestellten klassifizierten pervasiven ortsabhängigen Spiels *King of Location* als ein mögliches Szenario für ein pervasives Spiel wurden in diesem Kapitel die Anforderungen an das konkrete Spiel *King of Location* spezifiziert und für ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, erneut analysiert. In diesem Prozess

wurde im Rahmen einer Domänenanalyse die Domäne auf die identifizierte Klasse eingegrenzt und mit den allgemeinen Anforderungen an pervasive Spiele und der identifzierten Klasse ein Domänenmodell entwickelt. Aus den Erkenntnissen der Domänenanalyse wurden dann konkrete Anforderungen an ein Rahmenwerk für pervasive ortsabhängige Spiele der betrachteten Klasse spezifiziert. Abschließend wurden existierende Ansätze für Rahmenwerke und Infrastrukturen für pervasive Spiele und angrenzenden Problembereichen, wie z.B. die Verarbeitung und Bereitstellung von Kotextinformationen, betrachtet und auf einen möglichen Einsatz in dieser Arbeit untersucht und mit dem eigenen Ansatz verglichen.

5. Konzeption

Auf der Grundlage der Analyse aus Kapitel 4 wird in diesem Kapitel ein Konzept für ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, entwickelt und vorgestellt. Das Rahmenwerk soll, wie bereits in Abschnitt 1.1 beschrieben, die Komplexität bei der Entwicklung von pervasiven ortsabhängigen Spielen der betrachteten Klasse reduzieren und dadurch eine zügige Realisierung von neuen Spielideen und Spielkonzepten ermöglichen.

Dafür wird in Abschnitt 5.1 ein Überblick über das System gegeben und Problemstellungen und Eigenschaften von verteilten mobilen Systemen identifiziert, die sich aus den Anforderungen aus Abschnitt 4.2 ergeben. Aufbauend auf den Erkenntnissen und den Anforderungen aus Abschnitt 4.2 werden dann grundlegende Konzeptentscheidungen für einen eigenen Ansatz präsentiert und vorgestellt.

Anschließend wird die Systemarchitektur des Konzepts angelehnt an das QUASAR (Quality Software Architecture) Konzept (Siedersleben (Hrsg.), 2003) durch die Anwendungsarchitektur (A-Architektur), die Architektur für die technische Infrastruktur (TI-Architektur) und die Technikarchitektur (T-Architektur) beschrieben und spezifiziert.

5.1. Systemarchitektur

Damit die Komplexität in pervasiven Spielen der betrachteten Klasse durch ein Rahmenwerk reduziert werden kann, muss das Konzept die fachlichen Aspekte abbilden und gleichzeitig eine technische Infrastruktur bereitstellen, die eine Realisierung von neuen Spielideen und neuen Spielkonzepten ohne die explizite Betrachtung der Problemstellungen in mobilen verteilten Systemen ermöglicht und unterstützt. In Abbildung 5.1 wird ein grundlegender Überblick über die Architektur des gesamten Systems gegeben.

Ein auf dem Rahmenwerk basierendes pervasives Spiel der betrachteten Klasse bildet damit ein mobiles verteiltes System, bei dem die mobilen Endgeräte mit einem zentralen Server kommunizieren (Client-Server-Architektur). Das Rahmenwerk stellt für die Entwicklung des Spiels auf dem mobilen Endgerät und auf dem Server funktionale Schnittstellen zur

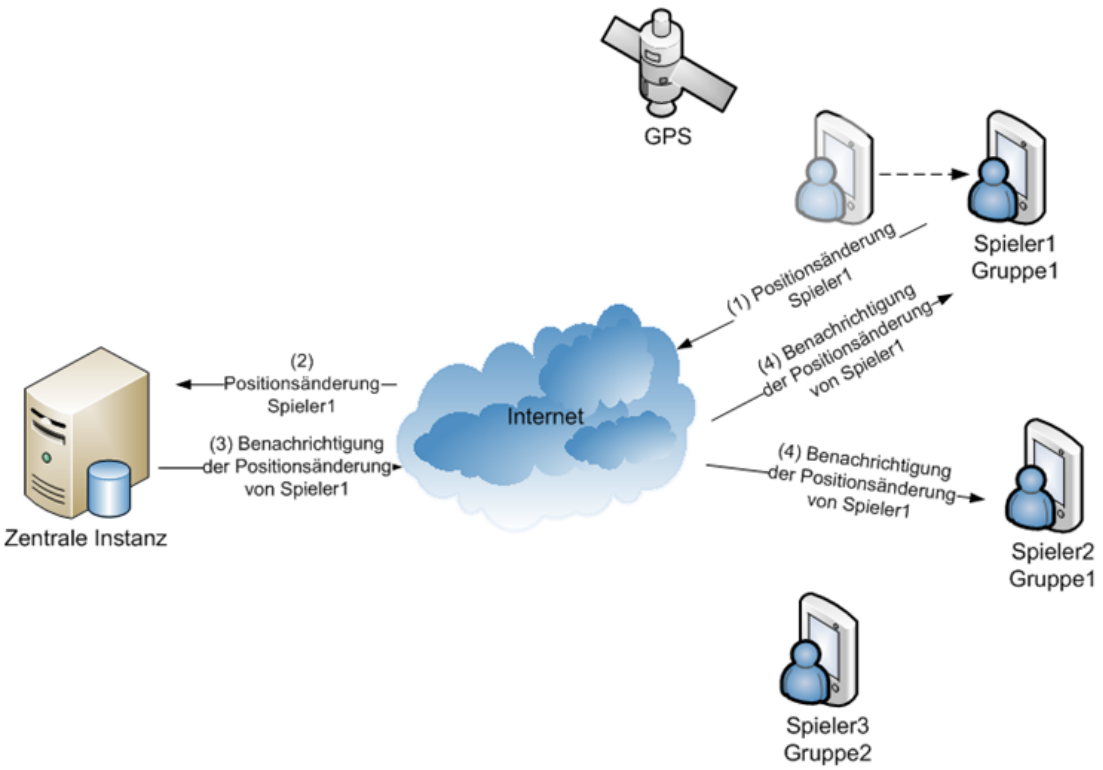

Abbildung 5.1.: Übersicht über die Systemarchitektur

Verfügung, die die Entwicklung eines konkreten pervasiven Spiels unterstützen und vereinfachen. Die Schnittstellen erlauben dabei die Manipulation der Entitäten und deren Beziehungen innerhalb eines Spiels und das Registrieren für spielrelevante Ereignisse, wie z.B. das Ändern der Position eines mobilen Endgeräts oder das Abschließen einer Aufgabe.

Der Server ist die zentrale Instanz und hat Zugriff auf den aktuellen Zustand des gesamten Spiels. Änderungen des Zustands können bei Bedarf an die mobilen Endgeräte verteilt werden, die sich für ein spezifisches Ereignis registriert haben. Gleichzeitig können die mobilen Endgeräte Informationen am Server anfragen und in die Darstellung des Spiels integrieren. Die mobilen Endgeräten haben je nach Spiel unter Umständen nur Zugriff auf einen Ausschnitts des Spiels. So sind z.B. Szenarien denkbar, die wie abgebildet auf den mobilen Endgeräten nur die Positionen von bestimmten anderen mobilen Endgeräten darstellen, aber von anderen wiederum keine Kenntnis besitzen[19].

[19] Vgl. auch das vorgestellte Szenario King of Location in Abschnitt 3

5.1.1. Interaktions- und Koordinationsmodell

Eine häufig wiederkehrende funktionale Anforderung aus Abschnitt 4.2.2.1 verlangt die Benachrichtigung bei spielrelevanten Ereignissen. Da es sich wie in Abschnitt 5.1 beschrieben um ein mobiles verteiltes System handelt und die mobilen Endgeräte ständig mit der zentralen Instanz im Internet kommunizieren müssen und jederzeit Benachrichtigungen erhalten können, wird technisch eine Infrastruktur benötigt, die eine Realisierung der funktionalen Anforderungen ermöglicht und gleichzeitig die technischen und nicht-funktionalen Anforderungen berücksichtigt. Alternativ zum Anfrage/Antwort-Schema (engl. request/reply schema), bei dem Informationen explizit synchron von einem Klienten am Server angefordert werden müssen (synchrones Interaktionsmuster), ist der Einsatz eines Push-Systems denkbar. Bei Push-Systemen werden Informationen von einem Informationsanbieter klassifiziert, der dann auch deren Verfügbarkeit ankündigt. Interessenten können diese Anforderungen abonnieren und sobald neue Informationen verfügbar sind, werden diese an die Subskribienten verteilt (Dustdar u. a., 2003). Push-Systeme ermöglichen damit einen asynchronen Informationsaustausch. Durch diesen Ansatz entfällt nach Dustdar u. a. (2003) das andauernde Anfragen nach neuen Informationen (engl. polling), und gleichzeitig wird unnötiger Ressourcenverbrauch durch „leere“ Anfragen vermieden. Die Kommunikation zwischen den Erzeugern und den Konsumenten erfolgt lose gekoppelt über einen Mediator, der für die Verteilung der Informationen zuständig ist.

Das vorgestellte Koordinationsmodell der nachrichtenorientierten Middleware aus Abschnitt 2.2 stellt eine Möglichkeit dar, Push-Systeme zu realisieren. Durch die unterschiedlichen Ausprägungen, z.B. warteschlangenorientiert, oder durch das Veröffentlichen/Abonnieren und den daraus folgenden unterschiedlichen Möglichkeiten der Informationsselektion, wie z.B. kanal- oder subjektbasiert, muss eine Konkretisierung dieses Ansatzes vorgenommen werden.

Durch die Nebenläufigkeit in verteilten Systemen muss das Rahmenwerk gleichzeitig technisch sicherstellen, dass konkrete, auf dem Rahmenwerk basierende Spiele auf allen mobilen Endgeräten Zugriff auf die relevanten Informationen und Ressourcen zur gleichen „Zeit“ und in der gleichen Reihenfolge erhalten. Damit eine logische Zeit gewährleistet werden kann, wird die Kommunikation über eine nachrichtenorientierte Middleware nach dem Veröffentlichen/Abonnieren Muster realisiert, die eine kanalbasierte Informationsselektion erlaubt.

5.1.2. Ereignisverarbeitung

Auf der Grundlage von spielrelevanten Informationen werden in einem pervasiven Spiel Ereignisse ausgelöst und Benachrichtigungen verteilt. Um einen wiederverwendbaren Ansatz zu ermöglichen, bieten sich Regelbasierte Systeme aus dem Bereich des kontextbewussten

Rechnens an. Das Schließen (engl. reasoning) von Low-Level Kontextinformationen, wie z.B. der aktuellen Position eines Spielers, auf High-Level Kontextinformationen, wie z.B. das Lösen einer Aufgabe, und das Auslösen von Ereignissen auf Grund von Kontextänderungen werden durch Regeln definiert und im System hinterlegt. Da jedoch der Bereich der kontextbewussten Systeme ein aktuelles Forschungsfeld ist (siehe Abschnitt 2.3), wird ein pragmatischer Ansatz, wie z.B. der vorgestellte Ansatz WildCAT (siehe Abschnitt 4.3.2), benötigt, der mit den aktuell verfügbaren Technologien und Ressourcen realisierbar ist.

5.1.3. Verteilung der Architekturschichten

Nach Book u. a. (2005) hat der Grad der Mobilität Einfluss auf die Architektur eines Systems und die Verteilung der Architekturschichten z.B. in einer klassischen Drei-Schichten-Architektur. Damit häufig wiederkehrende Entwurfsentscheidungen bei mobilen verteilten Systemen nicht von unstrukturierten Kriterien abhängen, haben Book u. a. (2005) eine Klassifikation anhand von drei Klassifikationskriterien (Benutzermobilität, Gerätemobilität und Dienstkonnektivität) aus der Sicht des Benutzers entwickelt, die die Verteilung der Architekturschichten vorgibt:

Benutzermobilität drückt den Grad der Orts- und Bewegungsfreiheit aus, der einem Benutzer gegeben wird, während dieser mit einem Dienst arbeitet. Dabei wird zwischen *lokal*, *verteilt*, *mobil* und *in Bewegung* differenziert. Wenn das Spiel, basierend auf dem zu entwickelnden Rahmenwerk, einen Dienst darstellt, muss das Kriterium für das Rahmenwerk durch die andauernde Bewegung der Spieler als *in Bewegung* klassifiziert werden.

Gerätemobilität ist die Fähigkeit, sich leicht zwischen den Abdeckungsgebieten der Zugangspunkte eines Netzwerks zu bewegen bzw. bewegt zu werden. Durch diese Definition kann wiederum zwischen den Graden *lokal*, *verteilt*, *mobil* und *in Bewegung* unterschieden werden. Da sich das mobile Endgerät bei ortsabhängigen Spielen häufig in Bewegung befindet, kann das Kriterium der Gerätemobilität als *in Bewegung* klassifiziert werden.

Dienstkonnektivität beschreibt die Abhängigkeit eines Dienstes von einer Netzverbindung. Da die Abdeckungsgebiete der Zugangspunkte eines Netzwerks nicht notwendigerweise überall überlappen müssen und dadurch Gebiete existieren, die keine Netzabdeckung besitzen, müssen Vorkehrungen, getroffen werden, um diese Situation handhaben zu können. Dabei kann der Grad der Dienstkonnektivität zwischen *Offline-Dienst*, *Hybrid-Offline-Dienst*, *Hybrid-Online-Dienst* und *Online-Dienst* unterschieden werden. Durch die nicht-funktionale operationelle Anforderung R-NFA-1 (Verfügbarkeit der Konnektivität zur zentralen Instanz zu jeder Zeit) kann der benötigte Grad der Dienstkonnektivität als Online-Dienst eingestuft werden.

In Abhängigkeit von der Dienstkonnektivität kann z.B. eine klassische Drei-Schichten-Architektur (bei der zwischen der Präsentationslogik, Geschäftslogik und der Datenhaltungsschicht unterschieden wird) in einem mobilen verteilten System nach unterschiedlichen Strategien verteilt werden. Durch die Klassifikation als Online-Dienst mit dem höchsten Bewegungsgrad in der Benutzer- und Gerätemobilität ist minimal eine Präsentationsschicht auf dem Klienten notwendig. Caches oder Synchronisationsmechanismen und eine Auslagerung von Teilen der Geschäftslogik auf den mobilen Klienten werden erst bei einer Realisierung als Hybrider-Offline- bzw. Hybrider-Online-Dienst benötigt. Da das zu entwickelnde Rahmenwerk jedoch auf dem mobilen Endgerät Schnittstellen für die Entwicklung von konkreten pervasiven Spielen bereitstellen soll, sind Teile der Geschäftslogik und der Daten auf den Klienten verfügbar. Deshalb wird im Rahmen dieser Arbeit eine Realisierung als Hybrider-Online-Dienst angestrebt.

5.1.4. Konzeptionelle Entwurfsentscheidungen

Für den Entwurf und die Architektur des Rahmenwerks werden folgende grundlegende konzeptionelle Entscheidungen getroffen, die sich aus den nicht-funktionalen Anforderungen ableiten lassen:

Entwurfsmusterorientierung - Entwurfmuster stellen wie in Abschnitt 2.1 angeführt Lösungen für wiederkehrende Problemstellungen auf Mikro-Architekturebene bereit, die durch z.B. Gamma u. a. (1994) weit verbreitet und allgemein bekannt sind und dadurch das Verständnis für den inneren Aufbau des Rahmenwerks erleichtern. Dadurch wird die Umsetzung der geforderten nicht-funktionalen Anforderungen wie z.B. Wartbarkeit und Erweiterbarkeit gefördert und unterstützt.

Komponentenorientierung - Komponenten stellen, wie in Abschnitt 2.1 beschrieben, ein weiteres Konzept der Wiederverwendung dar, das eine modulare Einheit mit definierten Schnittstellen bereitstellt und den inneren Aufbau vollständig vor dem Nutzer der Komponente versteckt (Siedersleben, 2004). Durch den Einsatz von Komponenten wird die Umsetzung der nicht-funktionalen Anforderungen weiter gefördert und eine Kapselung zusammengehörender fachlicher und technischer Funktionalität erreicht.

Durch die Entwurfsmuster- und Komponentenorientierung sollen die nicht-funktionalen Anforderungen an das zu entwickelnde Rahmenwerk, wie z.B. Wartbarkeit und Erweiterbarkeit, erfüllt werden. Ziel ist aber nicht die Entwicklung eines reinen Komponenten-Rahmenwerks, sondern einer Mischform, die sowohl objektorientierte (für die spielspezifische Konfiguration) als auch komponentenorientierte Eigenschaften in einem Rahmenwerk vereint (Reussner und Hasselbring, 2006).

5.1.5. Zusammenfassung

In diesem Abschnitt wurde die grundlegende Systemarchitektur vorgestellt und durch einen Überblick visualisiert. Da die Systemarchitektur ein mobiles verteiltes System darstellt, wurden anschließend einige Problemstellungen von mobilen verteilten Systemen aufgeführt und Lösungen im Rahmen des Entwurfs präsentiert, die die technischen und nichtfunktionalen Anforderungen aus den Abschnitten 4.2.2.2 und 4.2.2.3 umsetzen. Abschließend wurden grundlegende konzeptionelle Entwurfsentscheidungen getroffen, die einen wiederverwendbaren Entwurf sichern sollen.

5.2. Anwendungsarchitektur

Die Anwendungsarchitektur beschreibt die anwendungsspezifischen Komponenten des Rahmenwerks mit den dazugehörigen Schnittstellen und Beziehungen ohne Annahmen über die darunterliegende Technik (Siedersleben, 2004) und bildet damit die identifizierten fachlichen Anforderungen aus Abschnitt 4.2.2.1 ab.

5.2.1. Übersicht

Die Komponenten der Anwendungsarchitektur bilden die Anwendungsfälle eines Softwaresystems ab und realisieren das Prinzip der Trennung von Zuständigkeiten und der Kapselung von Informationen. Dabei ist die Schnittstelle einer Anwendungskomponente die Vereinigung der Schnittstellen ihrer Anwendungsfälle (Siedersleben (Hrsg.), 2003).

Die Anwendungsarchitektur des Rahmenwerks besteht grundlegend aus zwei anwendungsspezifischen Komponenten, der Spielwelt und der Benutzerverwaltung. Ein Überblick über die Architektur der Anwendung ist in Abbildung 5.2 dargestellt.

Die einzelnen Anwendungskomponenten kapseln dabei die Zuständigkeit für unterschiedliche anwendungsspezifische Informationen und Anwendungsfälle:

Spielwelt - Die Spielwelt kennt die Zustände der Spielinstanzen, steuert den Spielablauf der einzelnen Spielinstanzen durch Benachrichtigungen bei spielrelevanten Ereignissen, ist für die Einhaltung der Spielregeln verantwortlich, verwaltet die Entitäten der einzelnen Spielinstanzen und ist für die Protokollierung des Spielverlaufs und die Punkteverteilung zuständig. Abgeleitet aus dem Domänenmodell aus Abbildung 4.3 aus Abschnitt 4.2.1 liegt der Spielweltkomponente folgendes Datenmodell zugrunde (siehe Abbildung 5.3).

Das Datenmodell spiegelt zum größten Teil das Domänenmodell wider. Die Spielweltkomponente verwaltet die Spielinstanzen, dargestellt durch die Klasse `Game`.

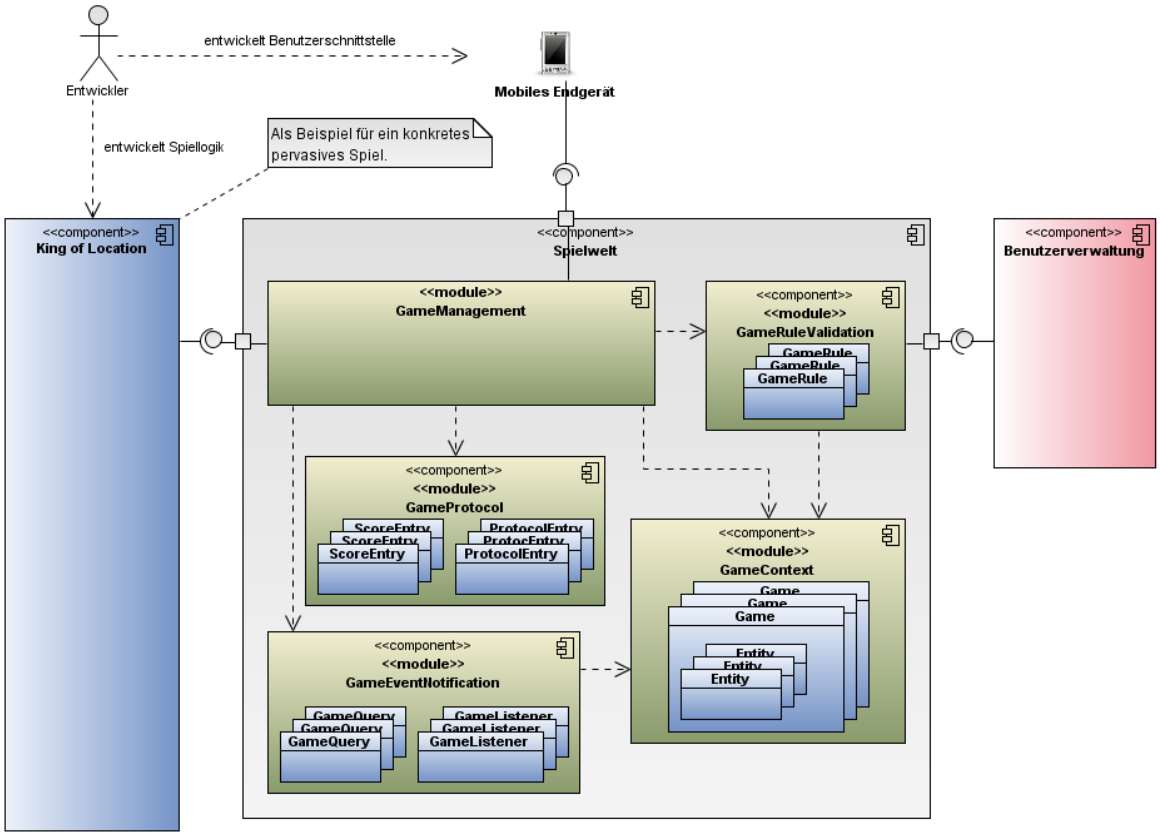

Abbildung 5.2.: Übersicht über die Anwendungsarchitektur

Eine Spielinstanz dient als Container für die Entitäten (`Entity`), die aktuell gültigen Sätze von Regeln (`GameRuleSet`) und die Attribute (`Attribute`), die eine Spielinstanz enthalten kann. Gleichzeitig gehört zu einer Spielinstanz auch eine aktuelle Rangliste (`ScoreBoard`), die einzelne Einträge mit den Punkten der einzelnen Entitäten enthalten kann (`ScoreEntry`). Ein Satz von Regeln stellt die zu einem bestimmten Zeitpunkt gültigen Regeln in einem Spiel dar und besteht wiederum aus einzelnen Regeln (`GameRule`). Durch Attribute können spielspezifische Eigenschaften einer Spielinstanz hinzugefügt werden. Eine Entität kann mehrere Aufgaben (`Challenge`) zugewiesen bekommen, die erfüllt werden müssen bzw. erfüllt werden können. Gleichzeitig hat eine Entität Zugriff auf mögliche Nachrichten (`Message`), die von anderen Entitäten an sie geschickt werden. Entitäten können wie Spiele auch Attribute enthalten, die spielspezifisch hinzugefügt werden können. Gruppen (`Group`) und Entitäten mit Positionen (`PositionedEntity`) sind Entitäten, die weitere Eigenschaften besitzen. Gruppen können eine Menge von Entitäten organisatorisch zusammenfassen, die dann spielspezifische Aufgaben gemeinsam lösen müssen oder denen Nachrichten gesendet werden, die dann an alle Entitäten innerhalb

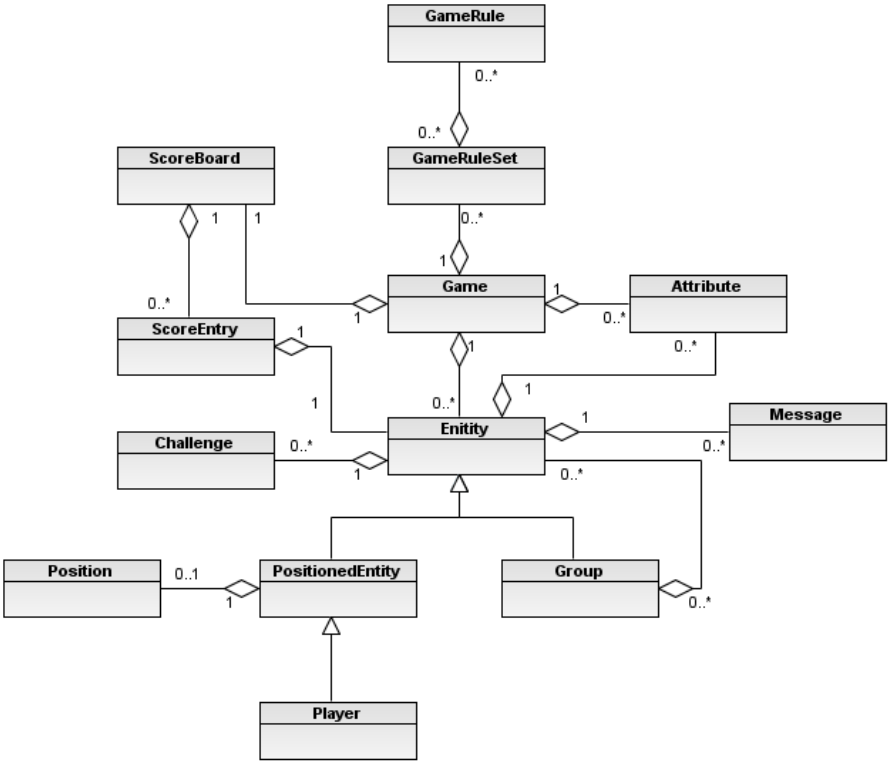

Abbildung 5.3.: Datenmodell der Spielweltkomponente

einer Gruppe verteilt werden. Gruppen bilden somit das Kompositum Entwurfsmuster (engl. composite pattern) nach Gamma u. a. (1994). Entitäten mit Positionen sind eine spezielle Ausprägung von Entitäten, die zusätzlich eine Position (`Position`) besitzen, die die zuletzt bekannte Position der Entität darstellt. Damit bilden Entitäten mit Position die elementare Entität innerhalb des Datenmodells. Spieler (`Player`) sind eine weitere Ausprägung von Entitäten mit Position, die die Spieler an den mobilen Endgeräten in einem konkreten Spiel darstellen. Im Gegensatz zu Entitäten, Entitäten mit Position und Gruppen sind die Spieler nicht von Beginn an einem festen Spiel zugeordnet, sondern werden erst durch das Beitreten zu einem Teil einer konkreten Spielinstanz.

Durch das Datenmodell wird die Datenhoheit der Spielweltkomponente festgelegt und die Zuständigkeiten für Manipulationen nach dem Gedanken der Komponentenorientierung eindeutig geklärt.

Benutzerverwaltung - Die Benutzerverwaltung verwaltet die Benutzer und ermöglicht die Registrierung und Authentifizierung. Auf ein Datenmodell für die Benutzerverwaltung wird an dieser Stelle bewusst verzichtet, da die Benutzerverwaltung ein externes System darstellt, von dem nur die Schnittstelle bekannt ist (vgl. die für das Rahmen-

werk übernommenen technischen Anforderungen K-TA-6 und K-TA-7 aus Abschnitt 4.1.3.2).

Die Spielwelt bildet den Zugangspunkt zum Rahmenwerk aus der Sicht eines Entwicklers eines konkreten Spiels basierend auf dem Rahmenwerk. Die Komponente für die Benutzerverwaltung wird durch die Spielwelt genutzt.

5.2.2. Außensicht

Bei der Außensicht werden die Schnittstellen der Komponenten der Anwendungsarchitektur vorgestellt und spezifiziert. Durch die Spezifikation der Schnittstelle weiß der Exporteur, was zu tun ist, und gleichzeitig weiß der Importeur, worauf er sich beim Einsatz der Schnittstelle verlassen kann. Dadurch entsteht ein Vertrag zwischen dem Exporteur und dem Importeur (Siedersleben, 2004). In den folgenden Abschnitten wird die angebotene Funktionalität der Schnittstellen der Komponenten vorgestellt.

5.2.2.1. Spielwelt

Die Spielweltkomponente stellt dem Entwickler operative Schnittstellen zur Verfügung, die den Zugriff auf die anwendungsspezifische Funktionalität des Rahmenwerks ermöglichen.

GameManagerAdministration

Die `GameManagerAdministration` Schnittstelle ermöglicht dem Entwickler die Verwaltung der Spielinstanzen innerhalb des Rahmenwerks (Abbildung 5.4).

Die Verwaltung der Spielinstanzen umfasst das Hinzufügen (`addGame()`) und Entfernen (`removeGame()`) von Spielinstanzen und das Setzen (`addAttribute()`)und Entfernen (`removeAttribute()`) von Attributen bei einzelnen Spielinstanzen. Außerdem kann der Zugriff auf eine konkrete Spielinstanz über den eindeutigen Bezeichner erfolgen (`getGame()`). Die Methode `setCurrentGameRuleSet()` setzt den aktuell gültigen Regelsatz in einer konkreten Spielinstanz, der durch die Methode `removeCurrentGameRuleSet` wieder entfernt werden kann.

Die `GameManagerAdministration` Schnittstelle erweitert die Schnittstelle `GameManagerEvent()`. Dadurch können über die Rückrufschnittstelle `GameListener` Ereignisse über Zustandsänderungen eines Spiels nach dem Beobachter-Entwurfsmuster (engl. observer pattern) (Gamma u. a., 1994) in einer Publish/Subscribe-Architektur abonniert werden (`registerListener()`), die eine übergebene Abfrage (`GameManagerQuery`)

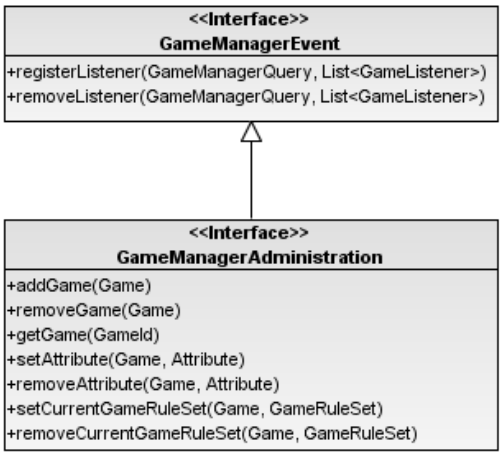

Abbildung 5.4.: `GameManagerAdministration` Schnittstelle der Spielweltkomponente

erfüllen (z.B. kann ein Ereignis ausgelöst werden, wenn eine bestimmte Anzahl an Spielern einer Spielinstanz beigetreten ist, damit das Spiel dann aktiv gestartet werden kann). Die `GameManagerQuery` Objekte repräsentieren somit Regeln, die erfüllt werden müssen, damit die jeweiligen Rückrufschnittstellen ausgeführt werden. Das Entfernen einer `GameListener` Rückrufschnittstelle für eine `GameManagerQuery` ist ebenso möglich, wenn eine Benachrichtung bei einem Ereignis nicht mehr gewünscht wird (`removeListener()`). Durch diesen Mechanismus kann auf Ereignisse, die den Spielablauf betreffen, spielspezifisch reagiert werden.

GameManager

Im Gegensatz zur `GameManagerAdministration` Schnittstelle stellt die Schnittstelle `GameManager` dem Entwickler eines konkreten pervasiven Spiels der betrachteten Klasse Methoden für die Suche nach Spielinstanzen und zum einfachen Beitreten und Verlassen einer konkreten Spielinstanz für einen Spieler bereit. Die Schnittstelle ist in Abbildung 5.5 dargestellt und wird im Folgenden erläutert.

Über die Schnittstelle `GameManager` kann ein Entwickler einen neuen Spieler registrieren (`registerPlayer()`) oder bereits registrierte Spieler authentifizieren (`login()`). Die über die administrative Schnittstelle `GameManagerAdministration` hinzugefügten Spielinstanzen können abgefragt (`getGames()`) und dann dem Spieler zur Auswahl präsentiert werden, wobei eine Filterung nach Kriterien (`GameCriteria`) möglich ist. Dadurch können z.B. nur Spielinstanzen zurückgeliefert werden, die physikalisch in der Nähe eines Spielers ablaufen. Nach der Auswahl einer Spielinstanz kann der Entwickler einen Spieler dann auf Wunsch die Spielinstanz beitreten lassen

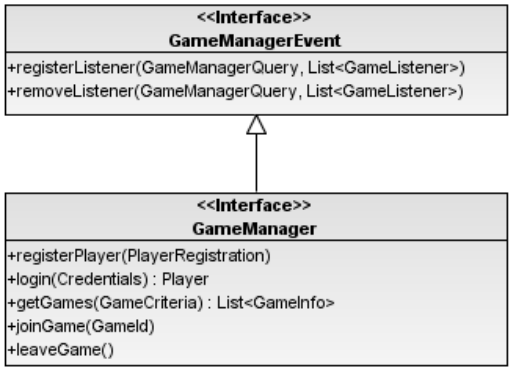

Abbildung 5.5.: `GameManager` Schnittstelle der Spielweltkomponente

(`joinGame()`). Der Spieler wird dann der aktiven Instanz des ausgewählten Spiels als Spieler zugeordnet. Ebenso kann der Entwickler einen Spieler eine Spielinstanz wieder verlassen lassen (`leave()`). Die `GameManager` Schnittstelle erweitert wiederum die Schnittstelle `GameManagerEvent()`. Dadurch können über die Rückrufschnittstelle `GameListener` Ereignisse über Zustandsänderungen einer Spielinstanz wiederum nach dem Beobachter-Entwurfsmuster abonniert werden (`registerListener()`), die eine übergebene Abfrage (`GameManagerQuery`) erfüllen (z.B. kann ein Ereignis ausgelöst werden, wenn andere Spieler Spiele betreten oder verlassen). Das Entfernen (`removeListener()`) einer `GameListener` Rückrufschnittstelle für eine `GameManagerQuery` ist wiederum jederzeit möglich.

GameWorld

Die Schnittstelle `GameWorld` ist für die Verwaltung der Entitäten innerhalb einer Spielinstanz verantwortlich. Eine Übersicht über die Methoden der Schnittstelle ist in Abbildung 5.6 dargestellt.

Entitäten mit Position (`PositionedEntity`) stellen den zentralen Bestandteil eines pervasiven Spiels der betrachteten Klasse dar, die abhängig von der eigenen Position Aufgaben lösen müssen. Entwicklern von konkreten pervasiven ortsabhängigen Spielen der betrachteten Klasse wird mit Hilfe der Schnittstelle `GameWorld` ermöglicht, Entitäten in einer Spielinstanz über bereitgestellte Methoden zu manipulieren. Entitäten können abhängig von der konkreten Ausprägung in einer Spielinstanz angefragt (`getEntity()`), hinzugefügt und aktualisiert (`setEntity()`) oder aber entfernt (`removeEntity()`) werden. Entitäten können Attribute zugewiesen (`setAttribute()`) und wieder entfernt werden (`removeAttribute()`), um spielspezifische Informationen mit Entitäten zu verknüpfen. Für die Ausprägung der Gruppe (`Group()`) sind zusätzlich noch Methoden für das Hinzufügen (`addEntityToGroup()`) und Entfernen (`removeEntityFromGroup()`) von

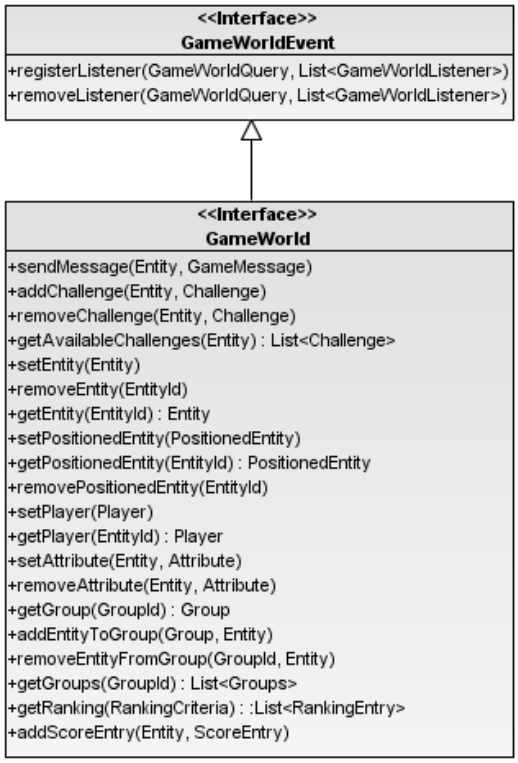

Abbildung 5.6.: `GameWorld` Schnittstelle der Spielweltkomponente

Entitäten in der Schnittstelle vorgesehen, damit Gruppen spielspezifisch durch den Entwickler manipuliert werden können. Durch dieses Konzept der Gruppen und die Möglichkeiten der Manipulation kann spielspezifisch eine „Spielwelt“ aus unterschiedlichen Entitäten und weiteren Gruppen hierarchisch zusammengesetzt werden. Der Entwickler kann sich bei der Entwicklung eines konkreten ortsabhängigen Spiels der betrachteten Klasse wie auch bei der `GameManager` und `GameManagerAdministration` Schnittstelle für spielspezifische Ereignisse (z.B. ausgelöst durch das Lösen einer Aufgabe durch einen anderen Spieler) in einem Spiel registrieren (`registerListener()`). Aber im Gegensatz zur `GameManager` und `GameManagerAdministration` Schnittstelle wird eine `GameWorldQuery` übergeben, die eine Nutzung auf die `GameWorld` Schnittstelle einschränkt. Wenn kein Interesse mehr an den Ereignissen besteht, kann die Zuordnung der Rückrufschnittstellen zu einer `GameWorldQuery` wieder entfernt werden (`removeListener()`). Nachrichten können in Form von `GameMessage` Objekten an andere Entitäten versendet werden (`sendMessage()`), wobei der Inhalt einer Nachricht nicht auf Textnachrichten begrenzt ist, sondern ein beliebiges Objekt darstellen kann. Für den Empfang muss eine `GameWorldListener` Rückrufschnittstelle wiederum mit einer vordefinierten `GameWorldQuery` registriert werden.

5.2.2.2. Benutzerverwaltung

Die Komponente der Benutzerverwaltung besitzt die schmale `UserManagement`-Schnittstelle, mit der Benutzer zurzeit nur registriert (`registerUser()`) und authentifiziert (`authenticateUser()`) werden können (siehe Abbildung 5.7).

Abbildung 5.7.: `UserManagement` Schnittstelle der Benutzerverwaltung

Allgemein besitzt eine Benutzerverwaltung noch weitere Funktionalitäten, wie z.B. die Aktualisierung der Benutzerinformationen, die aber für diesen Entwurf nicht benötigt werden und aus diesem Grund nicht abgebildet werden.

5.2.3. Innensicht der Spielweltkomponente

Die Spielweltkomponente ist die zentrale anwendungsspezifische Komponente, die dem Entwickler zur Verfügung gestellt wird. Im Folgenden wird die Innensicht der Spielweltkomponente aus Abbildung 5.2 aus der Sicht der Spiellogik eines konkreten Spiels im Detail in Form eines UML 2.1 Klassendiagramms beschrieben und dadurch eine mögliche Implementierung der Schnittstellen vorgestellt.

Die Schnittstellen der Außensicht aus Abschnitt 5.2.2 stellen wie beschrieben die Einstiegspunkte zum Rahmenwerk für den Entwickler bereit und werden dann in der `GameManagement` Schnittstelle zusammengeführt. Die Implementierung der `GameManagement` Schnittstelle bildet einen zentralen Zugangspunkt nach dem Fassaden-Entwurfsmuster (engl. façade pattern) und delegiert die einzelnen Anwendungsfälle an die zuständigen Module. Die `GameManagement` Schnittstelle ist zusätzlich für die Bereitstellung der Position der Spieler mit Hilfe der Schnittstelle `LocationProvider` zuständig und kann spielspezifische Benachrichtigungen über die `Communication` Schnittstelle an die Spieler übermitteln.

Zusätzlich existiert eine `GameManagementDecorator` Implementierung nach dem Dekorierer-Entwurfsmuster (engl. decorator patter) (Gamma u. a., 1994), damit die `GameManagement` Schnittstelle mit zusätzlicher Funktionalität erweitert werden kann, ohne dass die Implementierung der Klasse `GameManagement` modifiziert werden muss. Durch diesen Ansatz kann z.B. mit Hilfe der Implementierung `GameManagementValidation` die `GameManagement`-Implementierung dekoriert werden und so das Validieren der Spielregeln hinzugefügt werden. Gleichzeitig kann die

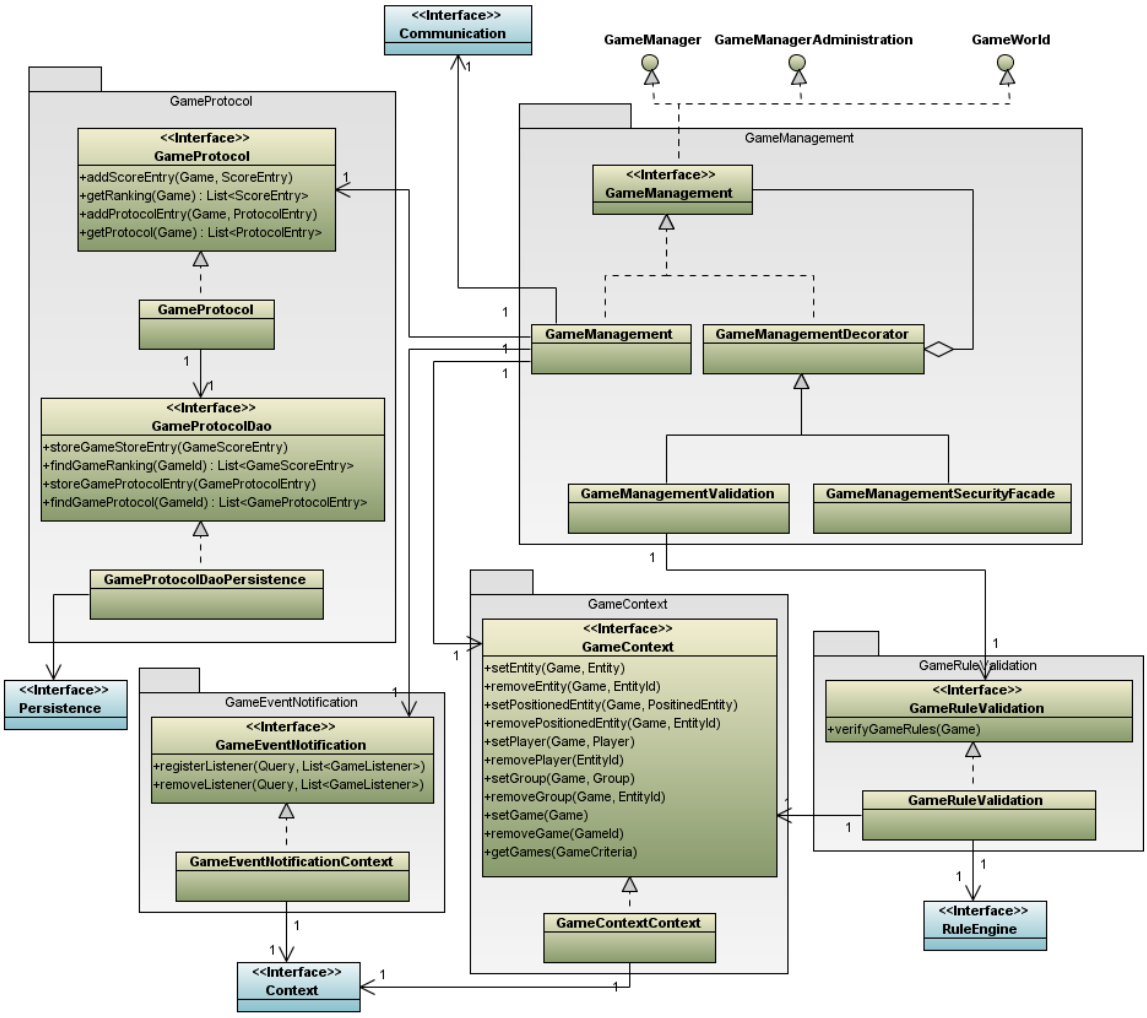

Abbildung 5.8.: Innensicht der Spielweltkomponente

`GameManagement` Implementierung auch mit einer Sicherheitsfassade dekoriert werden (`GameManagementSecurityFacade`), die nach Siedersleben (2004) einen zentralen Ansatz für die Behandlung von technischen Fehlern ermöglicht und bei Bedarf Reparaturmaßnahmen einleiten kann.

Damit das Rahmenwerk Ereignisbenachrichtungen auslösen kann, können wie bereits beschrieben an den Schnittstellen der Spielweltkomponenten Rückrufschnittstellen registiert werden, die dann durch die Implementierung der `GameManagement` Schnittstelle an die `GameEventNotification` Schnittstelle delegiert werden. Die `GameEventNotificationContext` Implementierung der Schnittstelle verwaltet die hinterlegten Rückrufschnittstellen mit den dazugehörigen Abfragen und führt mit Hilfe der `Context` 0-Schnittstelle die Methoden der registrierten Rückrufschnittstellen aus,

wenn sich der GameContext ändert und eine Abfrage erfüllt wird. Die Abfragen stellen somit Regeln dar, bei deren Erfüllung ein Ereignis ausgelöst und die Methode der Rückrufschnittstelle ausgeführt wird. Die Ereignisbenachrichtigung wird bei Veränderungen im GameContext angestoßen und überprüft, ob Ereignisse ausgelöst wurden.

Die GameContext Schnittstelle ist für die Verwaltung der Spielinstanzen und der Entitäten innerhalb einer Spielinstanz zuständig. Änderungen werden durch das GameEventNotification Modul erkannt und bei Bedarf wie beschrieben durch Ereignisse bekanntgegeben. Die GameEventNotificationContext Implementierung verwaltet die Kontextinformationen wiederum über die Context 0-Schnittstelle.

Das Überprüfen der Spielregeln wird durch die GameRuleValidation Schnittstelle bereitgestellt, die von der Klasse GameManagementValidation genutzt wird. Die GameRuleValidation Implementierung hat Zugriff auf den Kontext der einzelnen Spielinstanzen über die GameContext Schnittstelle. Die Regelsätze können dann mit den benötigten Kontextinformationen an die RuleEngine 0-Schnittstelle übergeben werden, die die konkrete Prüfung vornimmt.

Für die Vergabe von Punkten und die Protokollierung delegiert die Implementierung der GameManagement Schnittstelle die Aufrufe an die GameProtocol Schnittstelle. Die konkrete Implementierung der Schnittstelle benutzt ein Datenzugriffsobjekt (engl. data access object) nach dem gleichnamigen Entwurfsmuster, um Protokolleinträge oder Punkte persistent zu speichern(GameProtocolDao). Zum persistenten Speichern der Objekte nutzt die Implementierung GameProtocolDaoPersistence die Persistence 0-Schnittstelle.

Zusätzlich wird für die Komponenten der Anwendungsarchitektur ein Kompositionsmanager benötigt, der die Abhängigkeiten zu anderen Komponenten auflöst und die Implementierungen für die Schnittstellen zentral bereitstellt.

5.2.4. Schichtenverteilung

Durch die identifizierte Verteilung der Architekturschichten aus Abschnitt 5.1.3 ist auf dem mobilen Endgerät nur eine Präsentationsschicht notwendig. Damit jedoch das Rahmenwerk auf den mobilen Endgeräten eingesetzt werden kann, müssen auf den mobilen Endgeräten die Schnittstellen der Spielweltkomponente zur Verfügung gestellt werden. Ausgehend von der Systemarchitektur aus Abbildung 5.1, von dem Überblick über die Anwendungsarchitektur aus Abbildung 5.2 und der innensicht der Spielweltkomponente aus Abbildung 5.8 ergibt sich folgende funktionale Verteilung (Abbildung 5.9).

Auf der zentralen Instanz im Internet werden die Komponenten der Anwendungsarchitektur wie beschrieben ausgeführt, wobei die GameManagement Klasse nicht auf die

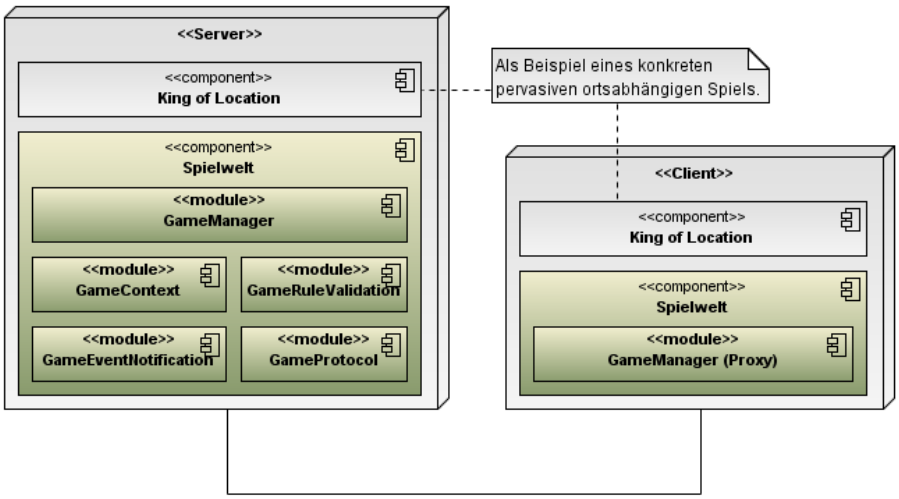

Abbildung 5.9.: Schichtenverteilung der Anwendungsarchitektur

`LocationProvider` Schnittstelle zugreift, da die Position der zentralen Instanz im Internet nicht spielrelevant ist. Die Schnittstellen der Spielweltkomponente werden auf den mobilen Endgeräten nur durch ein Stellvertreterobjekt (engl. proxy) implementiert, das die Anfragen stellvertretend entgegennimmt und dann über die `Communication` Schnittstelle an die zentrale Instanz im Internet weiterleitet. Zusätzlich benötigt die Klasse noch den Zugriff auf die `LocationProvider` Schnittstelle, um Positionsänderungen des mobilen Endgeräts zu registrieren und an die zentrale Instanz im Internet zu übertragen.

5.2.5. Dynamische Interaktion

Mit Hilfe von UML 2.1 Sequenzdiagrammen wird in diesem Abschnitt das Verhalten zur Laufzeit und damit die dynamische Interaktion zwischen den Komponenten und Modulen beschrieben. Dazu werden beispielhaft elementare Anforderungen und Anwendungsfälle betrachtet und beschrieben. Für einen besseren Überblick werden keine Fehlerfälle berücksichtigt.

Ereignisbenachrichtigung bei Positionsänderungen

Bei dem Szenario existieren zwei Spieler, die jeweils am System angemeldet und einer Spielinstanz beigetreten sind. Spieler 1 möchte bei Positionsänderungen von Spieler 2 benachrichtigt werden (Abbildung 5.10). Sobald sich ein Spieler am System angemeldet hat, werden Positionsänderungen des Spielers abonniert und an die `GameWorld` übertragen.

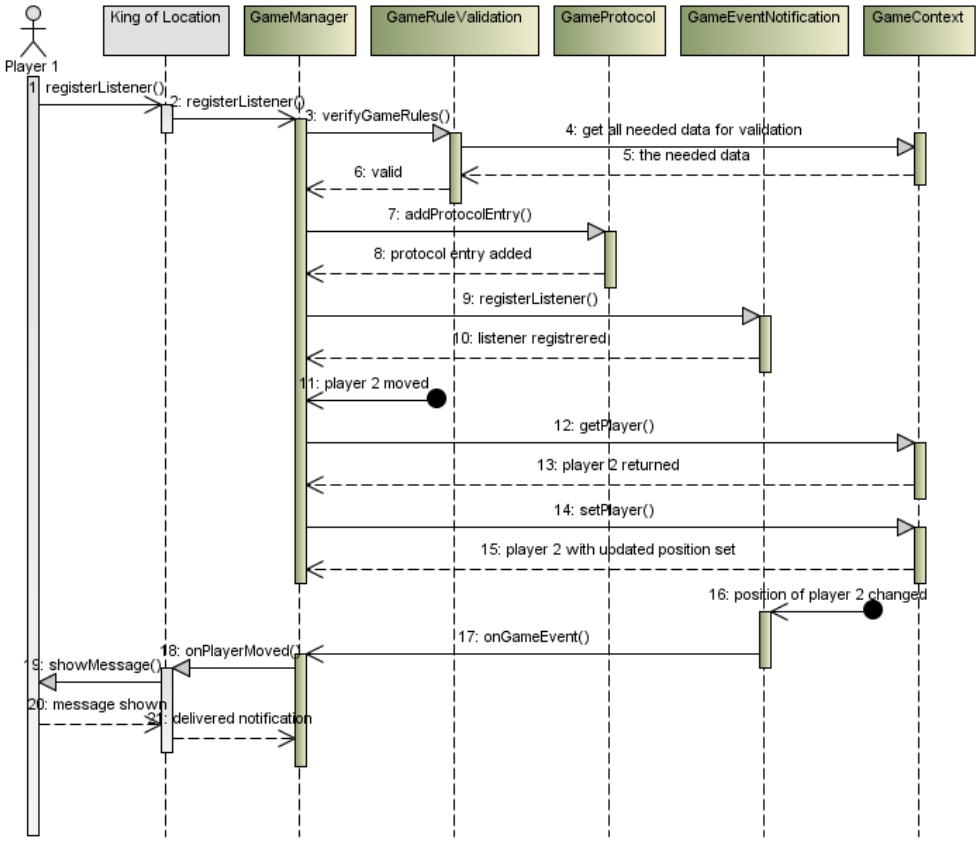

Abbildung 5.10.: Ereignisbenachrichtigung bei Positionsänderung eines Spielers

Spieler 1 registriert sich für Positionsänderungen von Spieler 2 (1,2). Das `GameManagement` Modul überprüft die Spielregeln und mit Hilfe des Moduls `GameRuleValidation` (3), das wiederum zur Validierung Zugriff auf die aktuellen Informationen der Spielinstanz über das `GameContext` Modul besitzt (4). Anschließend wird ein Protokolleintrag im `GameProtocol` Modul mit der Aktion des Registrierens vermerkt (7). Dann wird die Rückrufschnittstelle für eine Benachrichtigung im Falle des Eintretens des Ereignisses an das `GameEventNotification` Modul zur Verwaltung übergeben (9). Wenn sich Spieler 2 jetzt bewegt und seine Position verändert, wird das `GameManagement` Modul davon unterrichtet (11). Das `GameManagement` Modul holt den Spieler, dessen Position sich verändert hat, aus dem `GameContext` Modul (12), setzt die Position in dem Spieler und übergibt den aktualisierten Spieler wieder an das `GameContext` Modul zurück (14). Durch das Aktualisieren der Position im `GameContext` Modul wird ein Ereignis ausgelöst, dass die Position von Spieler 2 sich geändert hat (16). Dann führt das `GameEventNotification` Modul alle registrierten Rückrufschnittstellen aus, die sich für das Ereignis registriert haben (17) und Spieler 1 wird eine Nachricht mit der Ereignisbenachrichtigung angezeigt (19).

Erfüllen einer Aufgabe

Das Sequenzdiagramm in Abbildung 5.11 bildet den Anwendungsfall für das Auswählen und das Erfüllen einer Aufgabe ab.

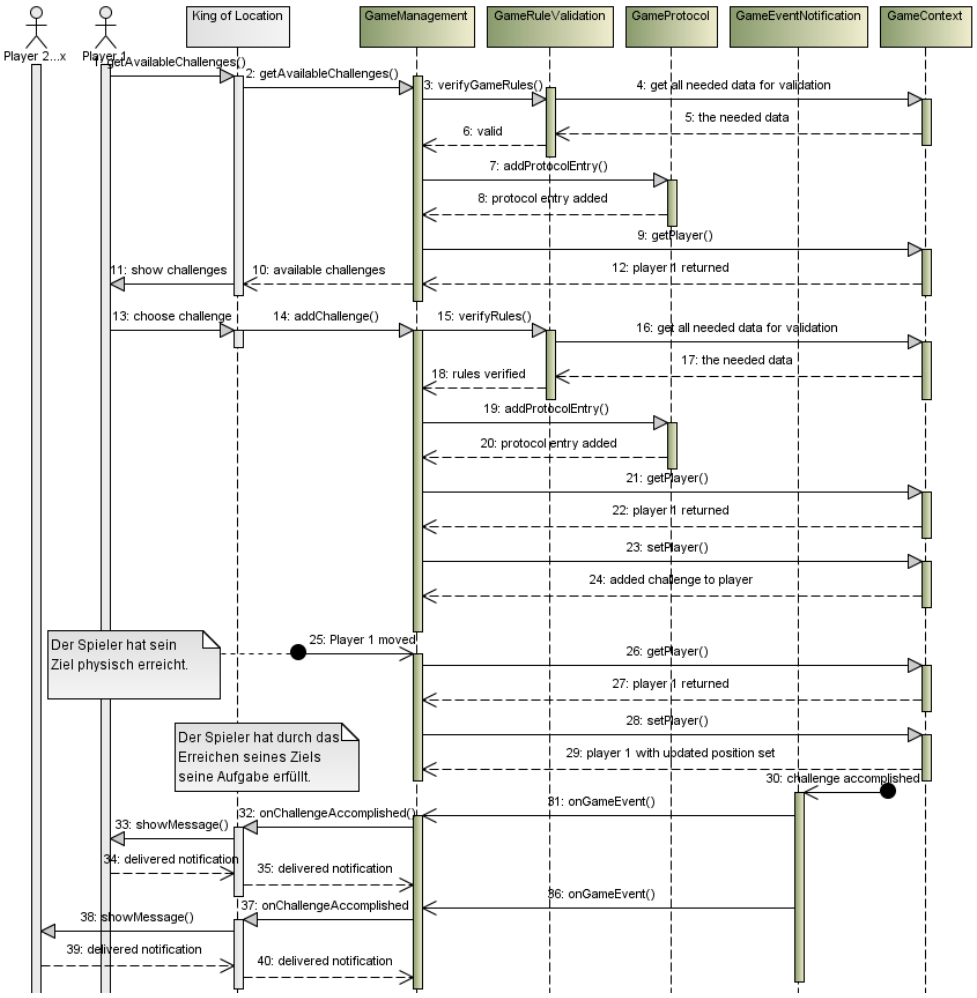

Abbildung 5.11.: Auswahl und Lösen einer Aufgabe

Ein Spieler möchte sich die zurzeit für den Spieler persönlich gerade gültigen Aufgaben anzeigen lassen, um eine Aufgabe auszuwählen, die er erfüllen möchte. Dazu wird das `GameManagement` Modul abgefragt (2). Das `GameManagement` Modul stößt eine Prüfung der Spielregeln durch das `GameRuleValidation` Modul an (3). Das `GameRuleValidation` Modul hat Zugriff auf das `GameContext` Modul, um die benötigten Kontextinformationen für die Überprüfung der Regeln abzufragen (4). Wenn der Zustand valide ist, wird durch das `GameProtocol` wiederum das `GameManagement` Modul angestoßen und ein Protokolleintrag mit der Aktion hinzufügt (7). Dann wird mit

Hilfe des GameContext Moduls der Spieler abgefragt und die verfügbaren Aufgaben dem Spieler angezeigt (11, 12). Wenn der Spieler eine Aufgabe z.B. mit dem Ziel, eine bestimmte Position zu erreichen, ausgewählt hat, werden die Spielregeln erneut geprüft (15, 16), ein Protokolleintrag erstellt (19) und dann der Spieler aus dem GameContext Modul geholt (21). Anschließend wird dem Spieler die ausgewählte Aufgabe zugewiesen (23).

Dann registrieren sich die Spieler 1 und 2 für eine Benachrichtigung, wenn Spieler 1 die Aufgabe erfüllt hat (nicht abgebildet). Wenn Spieler 1 seine Position verändert (25) wird das dem GameManagement Modul mitgeteilt, das den entsprechenden Spieler aus dem GameContext abfragt (26) und dann den Spieler mit der aktualisierten Position wieder dem GameContext Modul hinzufügt (28). Dadurch wird wiederum ein Ereignis beim GameEventNotification Modul ausgelöst, dass eine Aufgabe erfüllt wurde und Rückrufschnittstellen für das Ereignis vorhanden sind. Das GameEventNotification Modul führt die für das Ereignis registrierten Rückrufschnittstellen aus und benachrichtig über dasGameManagement Modul Spieler 1 und 2, die eine Nachricht angezeigt bekommen (31 - 40).

Versenden einer Nachricht

Beim folgenden Anwendungsfall übermittelt Spieler 1 eine Nachricht an Spieler 2, wobei Spieler 2 sich bereits für das Eintreffen von neuen Nachrichten registriert und eine Rückrufschnittstelle hinterlegt hat. Der Ablauf wird in Abbildung 5.12 dargestellt.

Spieler 1 übergibt die Nachricht und das Ziel (hier Spieler 2) über das beispielhaft aufgeführte konkrete Spiel an das GameManagement Modul (2). Das GameManagement Modul lässt die Spielregel von dem GameValidation Modul überprüfen (3), das wiederum die benötigten Kontextinformationen aus dem GameContext Modul abfragt (4). Wenn der Zustand valide ist, wird ein Protokolleintrag dem GameProtocol Modul hinzugefügt (7) und anschließend der Spieler 2 aus dem GameContext besorgt (9). Nachdem die Nachricht dem Spieler 2 hinzugefügt wurde (11) und damit eine Aktualisierung stattgefunden hat, wird ein Ereignis ausgelöst, dass Spieler 2 eine neue Nachricht bekommen hat (13). Das GameEventNotification Modul führt dann alle registrierten Rückrufschnittstellen aus, die sich für das Ereignis registriert haben (14). Spieler 2 wird z.B. die Nachricht dann angezeigt (16).

5.3. Architektur der technischen Infrastruktur

Die Architektur der technischen Infrastruktur wird durch die identifizierten technischen Anforderungen an ein Rahmenwerk für pervasive ortsabhängige Spiele der betrachteten Klasse

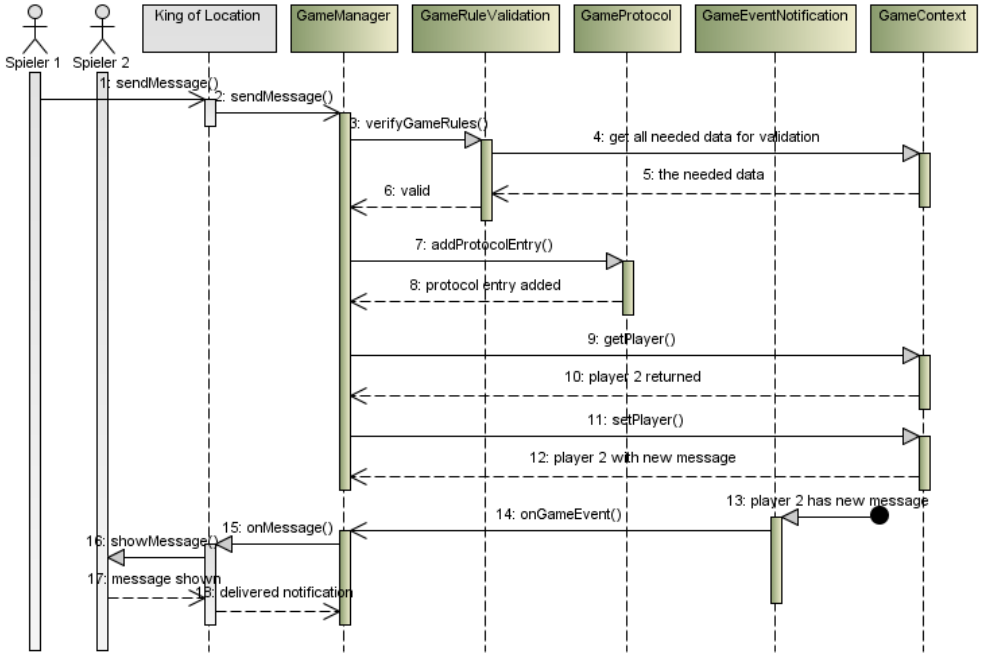

Abbildung 5.12.: Austausch von Nachrichten zwischen Spielern

aus Abschnitt 4.2.2.2 vorgegeben. Ein Überblick über die Architektur der technischen Infrastruktur ist in Abbildung 5.13 dargestellt.

Die technische Infrastruktur basiert wie bereits angeführt auf einer Client/Server-Architektur und zeigt die physikalische Verteilung des Systems und der Softwarekomponenten. Der Server repräsentiert die zentrale Instanz, die über das Internet jederzeit verfügbar ist und das Spiel zentral überwacht und steuert. Auf dem Server wird ein Applikationsserver ausgeführt, auf dem die Komponenten des Rahmenwerks und die pervasiven ortsabhängigen Spiele der betrachteten Klasse als Softwarekomponente ablaufen. Zusätzlich läuft auf dem Server eine Präsentationskomponente in Form einer Webanwendung, die als administrative Schnittstelle für die Erstellung und Überwachung eingesetzt werden kann oder aber Zugriff auf die Punkte der Spieler und die Ranglisten ermöglicht. Der Applikationsserver hat Zugriff auf eine Datenbank, die durch die Softwarekomponenten genutzt werden kann, um Daten persistent zu speichern.

Der Server kann über einen Webdienst auf eine externe Benutzerverwaltung zugreifen und ermöglicht damit die zentrale Registrierung und die Authentifizierung der Spieler.

Bei den Klienten handelt es sich um heterogene mobile Endgeräte, die unterschiedliche Programmierplattformen bereitstellen können. Auf der Programmierplattform werden Teile der Komponenten für das Rahmenwerk, das pervasive ortsabhängige Spiel der betrachteten Klasse, das auf dem Rahmenwerk beruht, und die Präsentation ausgeführt. Über die Programmierplattform haben die Softwarekomponenten Zugriff auf die Position des mobilen

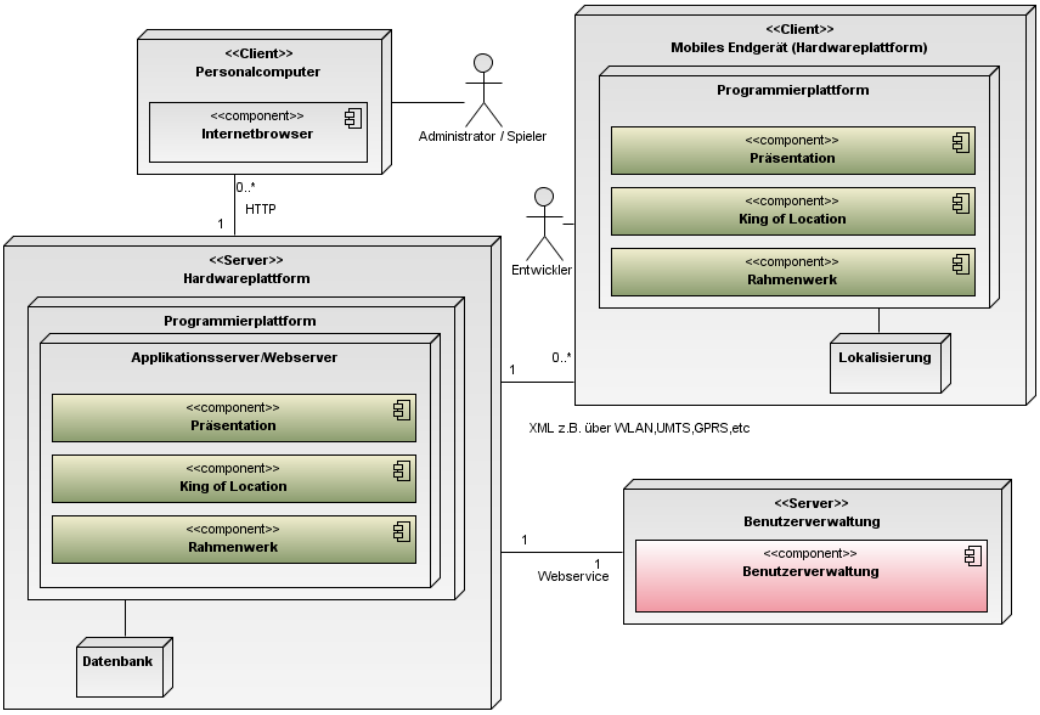

Abbildung 5.13.: Übersicht über die Architektur der technischen Infrastruktur

Endgeräts. Die Kommunikation mit dem Server erfolgt über den Austausch von XML Dokumenten über eine auf dem mobilen Endgerät verfügbare Kommunikationstechnologie.

Der Umfang der durch das Rahmenwerk auf dem mobilen Endgerät zur Verfügung gestellten Funktionalität hängt von den verfügbaren Ressourcen auf dem mobilen Endgerät ab und kann daher variieren.

Bei der Darstellung der Architektur der technischen Infrastruktur in Abbildung 5.13 wurden die einzelnen Komponenten und Beziehungen bewusst allgemein und abstrakt beschrieben und nicht auf konkrete Produkte und Technologien abgebildet. Durch dieses Vorgehen werden die technischen Anforderungen aus Abschnitt 4.2.2.2 vollständig erfüllt. Damit eine prototypische Realisierung im Rahmen dieser Arbeit durchgeführt und dadurch die Realisierbarkeit des Konzeptes bewiesen werden kann, werden an dieser Stelle zwei Szenarien vorgestellt, die die beschriebene abstrakte Architektur der technischen Infrastruktur beispielhaft instanziieren:

Szenario 1 - Eine mögliche Instanziierung setzt auf dem Server die JavaEE als Programmierplattform ein. Die Softwarekomponenten werden auf einem Tomcat Servlet Container ausgeführt und die Daten in einer Datenbank, wie z.B. der quelloffenen PostgrSQL-Datenbank, persistent gehalten. Die serverseitige Präsentation wird

über ein Web-Präsentationsrahmenwerk, wie z.B. das quelloffene Präsentationsrahmenwerk MyFaces[20] der Apache Software Foundation, realisiert[21], das für die Spezifikation JSR 127 JavaServer Faces als Referenzimplementierung angeboten wird. Die Softwarekomponente *King of Location* und das Rahmenwerk stellen die Geschäftslogik in einer klassischen Dreischichtenarchitektur dar und können z.B. duch das quelloffene Rahmenwerk Spring[22] unterstützt werden. Der Zugriff auf die Datenbank wird z.B. über einen objektrelationalen Mapper wie z.B. Hibernate ermöglicht. Die Kommunikation mit den mobilen Endgeräten erfolgt über den Austausch von XML Dokumenten. Für die Kommunikation mit der Benutzerverwaltung wird ein Endpunkt von einem Webdienst mit Hilfe von z.B. Apache Axis2 als standardisierter oder REST Webdienst bereitgestellt.

Auf eine Konkretisierung der Hardwareplattform des Servers wird an dieser Stelle verzichtet, da die Beschaffenheit der Hardware von anwendungsspezifischen Faktoren abhängig ist und somit individuell festgelegt werden muss.

Auf den modernen, leistungsstärkeren mobilen Endgeräten wird homogen die Android-Plattform eingesetzt, die eine Entwicklung in der Programmiersprache Java ermöglicht. Die Android Plattform ermöglicht den programmatischen Zugriff auf die Position des mobilen Endgeräts über das Global Positioning System (GPS) über eine proprietäre Programmierschnittstelle. Zusätzlich stellt die Android-Plattform Klassenbibliotheken und Rahmenwerke für die Softwarekomponenten bereit, die teilweise allgemein verbreitet sind (wie z.B. Apache Commons[23]) und z.B. die Kommunikation mit dem Internet über UMTS, GPRS oder WLAN ermöglichen und vereinfachen.

Szenario 2 Das zweite Szenario unterscheidet sich vom ersten Szenario nur durch die eingesetzten Produkte und Technologien des mobilen Endgeräts. Im Gegensatz zum ersten Szenario wird in diesem Szenario ein mobiles Endgerät mit eingeschränkten Ressourcen eingesetzt, das als Programmierplattform JavaME mit zusätzlichen Erweiterungen (wie z.B. JSR 179 Location API oder JSR 172 J2ME Web Services Specification) bereitstellt. Der Zugriff auf die Position des mobilen Endgeräts ist über GPS und die Kommunikation über Webdienste möglich.

Daneben sind noch weitere Szenarien denkbar, die z.B. das Microsoft .NET Framework[24] als Programmierplattform auf dem Server und Microsoft .NET Compact Framework auf den mobilen Endgeräten einsetzen oder aber noch weitere unterschiedliche Produkte, Technologien und Rahmenwerke verwenden. Durch die plattformunabhängige Kommunikation

[20] http://www.myfaces.org/
[21] http://www.apache.org/
[22] http://www.springframework.org/
[23] http://commons.apache.org/
[24] http://www.microsoft.com/net/

über XML Dokumente und Webdienste lassen sich auch Szenarien mit heterogenen Programmierplattformen und heterogenen mobilen Endgeräten realisieren, die aber im Rahmen dieser Arbeit nicht weiter betrachtet werden.

Die beiden Szenarien deuten die heterogene Vielfalt von mobilen Endgeräten an, die sich in unterschiedlichen Hardware- und Programmierplattform widerspiegeln, aber auch Unterschiede in der technischen Ausstattung und den verfügbaren Ressourcen bedeuten kann.

5.4. Technikarchitektur

Die Technikarchitektur stellt den sachgemäßen Umgang mit der Technik sicher und beschreibt die Komponenten des Systems, die von der Anwendung unabhängig sind (Siedersleben, 2004). Gleichzeitig hängt die Technikarchitektur von der Architektur der Infrastruktur ab und stellt technische Komponenten zur Verfügung, die über einfache Schnittstellen mit der Anwendung und untereinander kommunizieren können und deren Implementierung austauschbar ist (Siedersleben, 2004).

5.4.1. Übersicht

Die Technikarchitektur setzt sich aus fünf Komponenten zusammen, die in der Systemarchitektur in Abschnitt 5.1 identifiziert und als 0-Schnittstellen in der Innensicht der Anwendungsarchitektur in Abschnitt 5.2.3 aufgeführt wurden. Eine Übersicht über die Technikarchitektur ist in Abbildung 5.14 dargestellt.

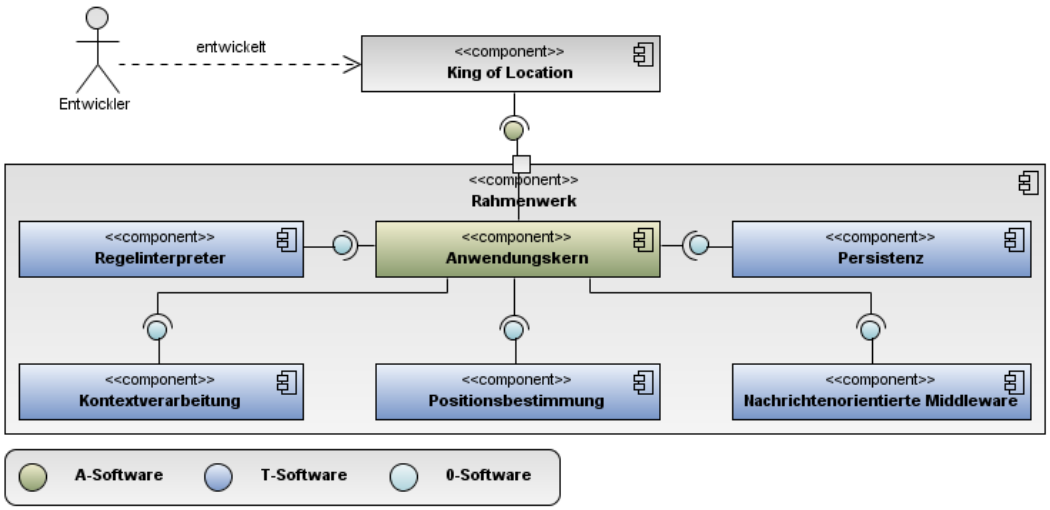

Abbildung 5.14.: Übersicht über die technische Architektur

Im Folgenden werden die einzelnen technischen Komponenten kurz beschrieben:

Regelinterpreter - Die Regelinterpreterkomponente validiert einen gegebenen Zustand durch einen Satz von definierten Regeln und gibt Rückmeldung über das Resultat.

Lokalisierung - Die Komponenten für die Lokalisierung stellt die aktuelle Position und Möglichkeiten der Benachrichtigung bei Ereignissen, z.B. bei Positionsänderungen, bereit.

Persistenz - Die Persistenzkomponente ist für die persistente Haltung der Daten zuständig. Persistente Daten können gespeichert und durch Abfragen wieder ausgelesen werden.

Nachrichtenorientierte Middleware - Die Komponente für die nachrichtenorientierte Middlware ermöglicht den Austausch von Nachrichten zwischen den mobilen Endgeräten und der zentralen Instanz im Internet. Nachrichten können anhand von Prädikaten veröffentlicht und abonniert werden.

Kontextverarbeitung - Die Komponente für die Kontextverarbeitung stellt Informationen aus dem Kontext zur Verfügung, erlaubt das Hinzufügen von Kontextinformationen und ermöglicht bei Bedarf die Benachrichtigung bei Änderungen.

Zusätzlich zu den beschriebenen Komponenten wird ein Kompositionsmanager benötigt, der die Komponenten verwaltet und die einzusetzende Implementierung für eine Komponenten-Schnittstelle instanziiert und zur Verfügung stellt (Siedersleben, 2004). Ebenso gehört die Verwaltung der Ausnahmebehandlung zu den Aufgaben des Kompositionsmanagers.

5.4.2. Außensicht

Im Gegensatz zur Außenansicht der Anwendungskomponenten werden in diesem Abschnitt die Schnittstellen der Technikarchitektur spezifiziert und beschrieben.

5.4.2.1. Nachrichtenorientierte Middleware

Die Schnittstellen der Komponente für die nachrichtenorientierte Middleware ermöglichen den Austausch von Nachrichten zur Kommunikation in einem verteilten System. In Abbildung 5.15 sind die Methoden der operativen Schnittstelle `Communication` abgebildet.

Bevor über die Komponente der nachrichtenorientierten Middleware Nachrichten abonniert oder veröffentlicht werden können, muss die Komponente über die Methode `initialize()` initialisiert werden. Bei diesem Prozess wird abhängig von der jeweiligen Implementierung z.B. eine Verbindung zu einem zentralen Nachrichtenvermittler aufgebaut und der Klient angemeldet. Ob die Komponente bereits initialisiert

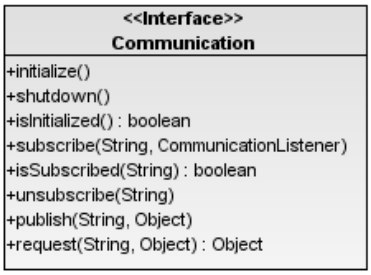

Abbildung 5.15.: `Communication` Schnittstelle der Komponente für die nachrichtenorientierte Middleware

worden ist, kann mit Hilfe der Methode `isInitialized()` jederzeit abgefragt werden. Mit der Methode `shutdown()` wird die Kommunikation definiert, z.B. über eine Abmeldung am zentralen Nachrichtenvermittler, beendet. Aufrufer können sich für Nachrichten eines bestimmten Kanals (symbolisiert durch ein Prädikat) registrieren und eine `CommunicationMessageListener` Rückrufschnittstelle hinterlegen, deren `onMessage()` Methode beim Eintreffen einer Nachricht auf dem Kanal ausgeführt wird. Der Methode wird der Inhalt der Nachricht übergeben und kann dann anwendungsspezifisch verarbeitet werden. Ebenso ist es möglich, Nachrichten in einem bestimmten Kanal zu veröffentlichen. Die Nachricht wird dann an alle Klienten übermittelt, die sich für Nachrichten dieses Kanals registriert haben. Bei den Methoden für das Veröffentlichen und das Abonnieren handelt es sich um asynchrone Aufrufe. Eine Nachricht kann jederzeit ohne explizites Anfordern eintreffen. Ebenso wird beim Veröffentlichen die Nachricht an den Kanal gesendet und nicht auf die Übermittlung an die Abonnenten gewartet. Die Kommunikation ist damit vollständig entkoppelt. Gleichzeitig bietet die Schnittstelle die Möglichkeit, direkt Nachrichten an einen bekannten Empfänger synchron zu übertragen und auf eine Antwort zu warten.

Zusätzlich zur operativen Schnittstelle existiert für die Komponente der nachrichtenorientierten Middleware eine schmale `CommunicationTransformationDefinition` Schnittstelle für die Transformation der Objekte in eine Transportrepräsentation und wieder zurück (Abbildung 5.16).

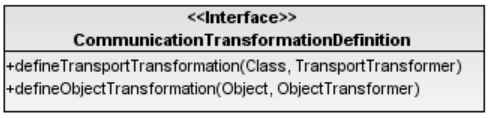

Abbildung 5.16.: `CommunicationTransformationDefinition` Schnittstelle der Komponente für die nachrichtenorientierte Middleware

Die Methode `defineTransportTransformation()` ermöglicht das Hinzufügen einer Transformationsregel, die ein Objekt einer bestimmten Klasse in eine Transportreprä-

sentation, in Abhängigkeit der jeweiligen Implementierung, überführt. Für die Rücktransformation kann durch die Methode `defineObjectTransformation()` eine definierte Transformationsregel hinterlegt werden, die aus der Transportrepräsentation wieder ein Objekt erstellt.

Gleichzeitig wird noch eine weitere administrative Schnittstelle für die Konfiguration der Komponente angeboten (Abbildung 5.17).

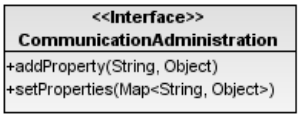

Abbildung 5.17.: `CommunicationAdministration` Schnittstelle der Komponente für die nachrichtenorientierte Middleware

Durch die Methoden `addProperty()` und `setProperties()` kann die Komponente implementierungsspezifisch konfiguriert werden und z.B. die Adresse des zentralen Nachrichtenvermittlers für den Aufbau einer Verbindung in der Komponente hinterlegt werden.

5.4.2.2. Kontextverarbeitung

Durch die `Context` Schnittstelle der Komponente für das Kontextbewusstsein können Kontextinformationen hierarchisch verwaltet und Ereignisbenachrichtigungen bei Änderungen abonniert werden. Ein Überblick über die Methoden der Schnittstelle ist in Abbildung 5.18) dargestellt.

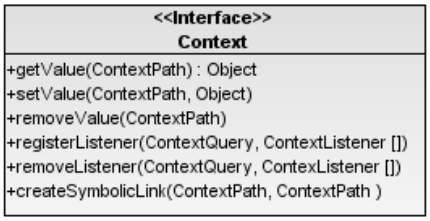

Abbildung 5.18.: `Context` Schnittstelle der Komponente für das Kontextbewusstsein

Die Methode `setValue()` ermöglicht das Hinzufügen von Objekten in den Kontext unter einem spezifischen Pfad (`ContextPath`), der unabhängig von der Implementierung einen hierarchischen Pfad innerhalb der Kontextinstanz darstellt. Unter dem gleichen Pfad können die Objekte mit Hilfe der Methode `getValue()` wieder aus dem Kontext geholt und bei Bedarf weiterverarbeitet werden. Das Löschen aus dem Kontext wird durch die Methode `removeValue()` bereitgestellt. Ereignisbenachrichtigungen können durch das

Hinterlegen einer `ContextListener` Rückrufschnittstelle in Kombination mit einer Abfrage (`ContextQuery`) durch die Methode `registerListener()` abonniert werden. Die Abfrage legt genau fest, wann die `update()` Methode der Rückrufschnittstelle ausgelöst werden soll und besteht aus einem eindeutigen Bezeichner und den für die Erstellung notwendigen Argumenten. Die Implementierung der Schnittstelle ist dann für die Transformation in eine implementierungsabhängige Repräsentation der Abfrage verantwortlich. Über die Methode `removeListener()` werden die Rückrufschnittstellen zur übergebenen Abfrage wieder entfernt und keine Benachrichtigungen mehr ausgelöst.

Die Definition der Abfragen und der Pfade erfolgt über die beiden Schnittstellen `ContextQueryDefinition` und `ContextPathDefinition`, die durch die Methoden `defineContextQuery()` und `defineContextPath()` das Hinzufügen von Abfragen und Pfaden ermöglichen (Abbildung 5.19). Die Definition erfolgt in Abhängigkeit von der jeweiligen Implementierung und der eingesetzten Technologie.

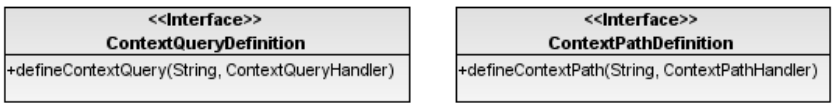

Abbildung 5.19.: `ContextPathDefinition` und `ContextQueryDefinition` Schnittstellen der Komponente für das Kontextbewusstsein

Der Benutzer der `Context` Schnittstelle benötigt durch diesen Ansatz keine Kenntnisse über die Implementierung und die technikabhängigen Angaben der Abfragen und Pfade.

5.4.2.3. Regelinterpreter

Die operative `RuleEngine` Schnittstelle der Regelinterpreterkomponente ermöglicht das Validieren eines Satzes von Regeln mit Hilfe der Methode `verifyRuleSet()` (Abbildung 5.20).

Abbildung 5.20.: `RuleEngine` Schnittstelle der Regelinterpreterkomponente

Ein Satz von Regeln wird unabhängig von der Technologie durch einen eindeutigen Bezeichner angegeben und zusammen mit einem Arbeitsbereich übergeben, der alle benötigten Informationen enthält, die für die Überprüfung der konkreten Regeln notwendig sind. Die Regelsätze können mit Hilfe der administrativen Schnittstelle `RuleSetDefinition` und der Methode `defineRuleSet()` hinzugefügt werden (Abbildung 5.21).

Abbildung 5.21.: `RuleSetDefinition` Schnittstelle der Regelinterpreterkomponente

Die Regelsätze werden mit einem eindeutigen Bezeichner assoziiert, der bei einem Aufruf auf der operativen Schnittstelle verwendet werden kann. Dadurch wird wiederum erreicht, dass für die Verwendung der operativen Schnittstelle kein implementierungsabhängiges Wissen benötigt wird.

5.4.2.4. Lokalisierung

Für die Bereitstellung der aktuellen Position und für die Ereignisbenachrichtigung steht auf den mobilen Endgeräten die `LocationProvider` Schnittstelle zur Verfügung.

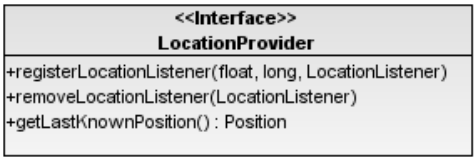

Abbildung 5.22.: `LocationProvider` Schnittstelle der Lokalisierungskomponente

Die Methode `getLastKnownLocation()` liefert die zuletzt bekannte Position zurück. Für die andauernde Benachrichtigung können über die Methode `registerLocationListener()` Rückrufschnittstellen hinterlegt werden. Die Ausführung wird dabei durch den minimalen Abstand zwischen zwei Positionsbestimmungen in Millisekunden und der minimal nötigen Positionsveränderung in Metern festgelegt. Durch diesen Ansatz kann die Anzahl der Positionsveränderungen und damit der Ereignisse verringert werden. Die Methode `removeLocationListener()` entfernt wiederum eine Rückrufschnittstelle, um die Benachrichtigung bei Positionsereignissen deaktivieren zu können.

5.4.2.5. Persistenz

Die `Persistence` Schnittstelle der Persistenzkomponente ist für die Verwaltung der persistenten Daten zuständig (Abbildung 5.23).

Die Methoden `persist()`, `get()`, `delete()` und `find()` bilden die grundlegenden Operationen zum Speichern, Laden, Löschen und Suchen ab. Damit bei der Verwendung

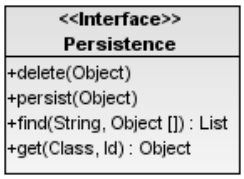

Abbildung 5.23.: `Persistence` Schnittstelle der Persistenzkomponente

der `find()` Methode kein Wissen über die konkrete Technologie nötig ist, wird wiederum ein eindeutiger Bezeichner für eine Abfrage und die benötigten Argumente übergeben. Die Definition der Abfragen für die Suche kann dann durch die `PersistenceDefinition` Schnittstelle erfolgen (Abbildung 5.24).

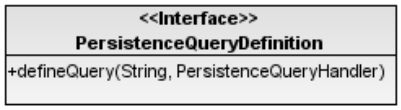

Abbildung 5.24.: `PersistenceQueryDefinition` Schnittstelle der Persistenzkomponente

Dadurch können Abfragen abhängig von der verwendeten Technologie definiert werden, die dann von der operativen Schnittstelle genutzt werden können.

5.4.3. Innensicht

Bei der Innensicht einer Komponente handelt es sich um eine konkrete Implementierung einer bereitgestellten Schnittstelle aus der Außensicht. Da die Architektur der technischen Infrastruktur aus Abbildung 5.13 in Abschnitt 5.3 einen sehr abstrakten Entwurf darstellt, wird kein konkreter Verweis auf ein konkretes Produkt oder eine Technologie vorgenommen.

5.4.3.1. Nachrichtenorientierte Middleware

Die operative Schnittstelle `Communication` und die administrative Schnittstelle `CommunicationAdministration` werden durch eine Klasse implementiert (Abbildung 5.25).

Die Implementierung nutzt die Programmierschnittstelle einer konkreten Technologie oder eines konkreten Produkts. Die Konfiguration erfolgt dabei durch Konfigurationseigenschaften, die durch die `CommunicationAdministration` Schnittstelle hinzugefügt werden können. Die Transformation zwischen der Transport- und der Objektre-

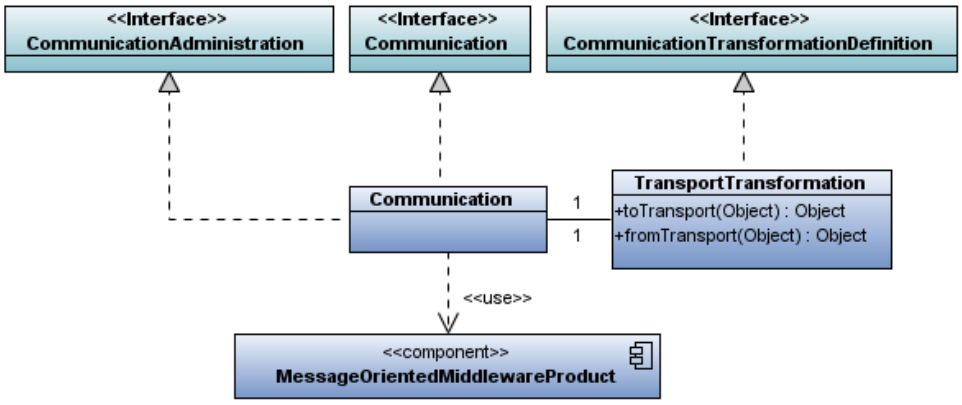

Abbildung 5.25.: Innensicht der Komponente für die nachrichtenorientierte Middleware

präsentation und umgekehrt wird durch die Klasse `TransportTransformation` übernommen. Die einzelnen Transformationsregeln müssen mit der Hilfe der `CommunicationTransformationDefinition` Schnittstelle hinzugefügt werden.

5.4.3.2. Kontextverarbeitung

Bei der Innensicht der Komponente für das Kontextbewusstsein wird die `Context` Schnittstelle mit Hilfe von einem konkreten Produkt oder einer konkreten Technologie implementiert (Abbildung 5.26).

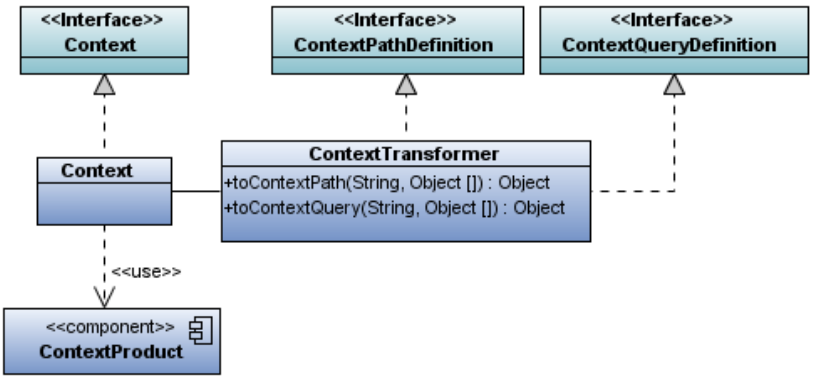

Abbildung 5.26.: Innensicht der Komponente für das Kontextbewusstsein

Die produktabhängige Transformation der Abfragen und Pfade wird durch die `ContextTransformer` Klasse realisiert, die durch die beiden Schnittstellen

`ContextPathDefinition` und `ContextQueryDefinition` administriert werden kann, um die jeweiligen Transformationsregeln hinzuzufügen.

5.4.3.3. Regelinterpreter

Die Regelinterpreterkomponente implementiert in einer Klasse die operative Schnittstelle `RuleEngine`, die wiederum eine produktspezifische Programmierschnittstelle verwendet (Abbildung 5.27).

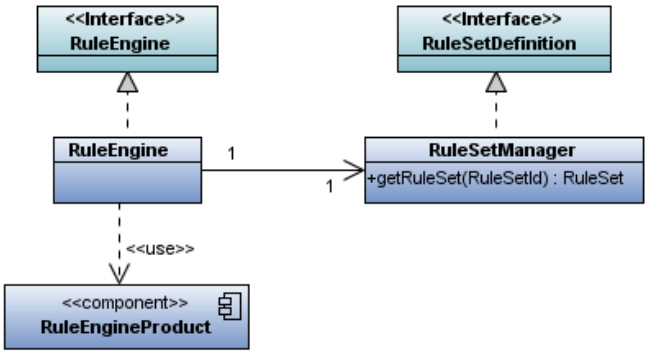

Abbildung 5.27.: Innensicht der Regelinterpreterkomponente

Die Regelsätze werden über eine administrative Schnittstelle `RuleSetDefinition` definiert, die von der Klasse `RuleSetManager` implementiert wird. Die Klasse verwaltet die hinzugefügten Regelsätze und gibt durch die Methode `getRuleSet()` den gewünschten technikspezifischen Regelsatz für den eindeutigen Bezeichner zurück.

5.4.3.4. Lokalisierung

Die `LocationProvider` Schnittstelle wird wiederum in von Abhängigkeit einer auf dem mobilen Endgerät verfügbaren Programmierschnittstelle in einer konkreten Klasse implementiert (Abbildung 5.28).

5.4.3.5. Persistenz

Die `Persistence` Schnittstelle der Persistenzkomponente wird mit Hilfe einer konkreten Persistenztechnologie realisiert, wobei die Suchanfragen in Abhängigkeit der eingesetzten Technologie vorher mit Hilfe der administrativen Schnittstelle `PersistenceQueryDefinition` definiert werden müssen (Abbildung 5.29).

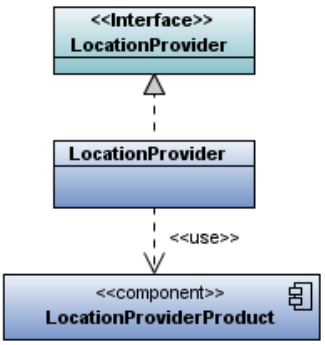

Abbildung 5.28.: Innensicht der Lokalisierungskomponente

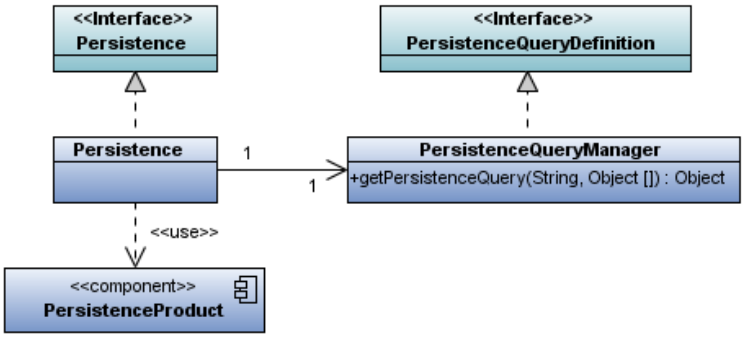

Abbildung 5.29.: Innensicht der Persistenzkomponente

Die definierten Suchanfragen können dann anhand des eindeutigen Bezeichners in der `Persistence` Schnittstelle verwendet werden.

5.5. Testkonzept

Durch die entwurfsmuster- und komponentorientierte Architektur mit definierten Schnittstellen und der klaren Trennung von anwendungsspezifischen und technischen Zuständigkeiten kann das Gesamtkonzept mehrstufig getestet werden. Auf Klassen- und Komponentenebene werden mit Modul- bzw. Komponententests z.B. die Schnittstellen der spezifizierten Komponenten getestet. Wenn Komponenten Schnittstellen von anderen Komponenten importieren werden, werden diese Schnittstellen durch eine vorgetäuschte Implementierung (engl. mock object) ersetzt. Das erlaubt das isolierte Testen von Komponentenschnittstellen. Wenn die Modultests erfolgreich abgeschlossen werden, wird das Gesamtsystem abschließend integriert getestet. Dafür wird das Gesamtsystem als Ganzes unter realen Bedingungen getestet. Da zum Zeitpunkt der Erstellung dieser Arbeit keine realen mobilen Endgeräte mit

der Android-Plattform verfügbar sind, wird der Integrationstest auf Emulatorebene in einer Laborumgebung durchgeführt.

5.6. Zusammenfassung

In diesem Kapitel wurde auf der Grundlage der Analyse aus Kapitel 4 das Konzept für ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, entwickelt. Ausgehend von dem Überblick über die grundlegende Systemarchitektur aus Abschnitt 5.1 wurde anhand der Anforderungen das System als Push-System klassifiziert und die Realisierung in Form einer nachrichtenorientierten Middleware festgelegt. Anschließend wurde die Verteilung der Architekturschichten anhand von drei Mobilitätsgraden als Online-Dienst mit dem höchsten Bewegungsgrad in der Benutzer- und Gerätemobilität klassifiziert und festgelegt, wobei in diesem Entwurf auf den mobilen Endgeräten neben der Präsentationsschicht auch Teile der Geschäftslogik ausgeführt werden. Nach den konzeptionellen Entwurfsentscheidungen für einen komponenten- und entwurfsmusterorientierten Ansatz wurde eine Anwendungsarchitektur entwickelt, die Architektur der technischen Infrastruktur bestimmt und die Technikarchitektur entworfen und spezifiziert. Abschließend wurde ein Testkonzept skizziert, das die Qualität im Rahmen einer Realisierung sichern soll.

6. Realisierung

In diesem Kapitel wird die Tragfähigkeit des vorgestellten Entwurfs aus Kapitel 5 für ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, durch eine prototypische Realisierung sichergestellt und verifiziert. Dafür wird der Umfang der Realisierung abgesteckt, Realisierungsdetails präsentiert und abschließend die Durchführung von Qualitätssicherungsmaßnahmen beschrieben.

6.1. Realisierungsumfang

Die Grundlage der Realisierung bildet das im Rahmen der Architektur für die technische Infrastruktur vorgestellte Szenario 1. Dadurch wird eine Konkretisierung der abstrakt beschriebenen technischen Architektur vorgenommen und die Rahmenbedingungen für eine Realisierung festgelegt. Wie beschrieben wird das Rahmenwerk serverseitig mit JavaEE implementiert und auf einem Apache Tomcat Server in der Version 5.5.27 ausgeführt. Auf die Implementierung einer serverseitigen Präsentationsschicht in Form einer Webanwendung wurde verzichtet, da die Funktionalität und die Anforderungen spielspezifisch sind und keinen funktionalen Beitrag für die Entwicklung eines Rahmenwerks liefern.

Die Spielweltkomponente der Anwendungsarchitektur als zentrale funktionale Komponente wird aufgrund des begrenzten Zeitrahmens sowohl server- als auch klientenseitig nicht vollständig implementiert. Die elementaren funktionalen Anforderungen, die durch die Schnittstellen `GameWorld`, `GameManager` und `GameManagerAdministration` abgedeckt werden, werden somit nur prototypisch umgesetzt. Die Funktionalität einiger Methoden, wie z.B. `setEntity()`, werden implizit durch die Realisierung einer anderen Methode (`setPositionedEntity()`) bewiesen und müssen dadurch nicht explizit realisiert werden.

Auf eine Realisierung der Persistenzkomponente wurde im Rahmen der prototypischen Realisierung wiederum aufgrund des begrenzten Zeitrahmens bewusst verzichtet, da das persistente Speichern von Daten keinen direkten funktionalen Beitrag leistet und ein Standardproblem mit etablierten Lösungsansätzen in der Informatik darstellt. Aus diesem Grund wird auf eine dedizierte Installation einer Datenbank auf dem Server verzichtet.

Im Gegensatz dazu wird die Schnittstelle `RuleEngine` der Regelinterpreterkomponente zwar durch eine Klasse konkret implementiert, jedoch ohne weitere Funktionalität. Jeder Aufruf auf den Schnittstellen der Spielweltkomponente wird als valide angesehen und niemals abgewiesen.

6.2. Realisierungdetails

In diesem Abschnitt werden die Realisierungsdetails der einzelnen Komponenten des Konzepts der Anwendungs- und Technikarchitektur vorgestellt und Abweichungen vom Konzept diskutiert.

6.2.1. Abweichungen vom Konzept

Die Realisierung der Benutzerverwaltung erfolgt im Rahmen der prototypischen Realisierung als vorgetäuschtes Objekt (engl. mock object) und wird nicht wie spezifiziert über einen Adapter über einen Webdienst entfernt aufgerufen. Die Implementierung der Schnittstelle `UserManagement` verhält sich der Spezifikation entsprechend, aber der Zustand der Benutzerverwaltung wird nicht persistent gespeichert. Bei jeder Instanziierung der Komponente wird ein definierter Zustand wieder hergestellt, der die Durchführung von automatisierten Tests des gesamten Systems unterstützt und somit die Testbarkeit des prototypisch realisierten Entwurfs erhöht.

Innerhalb der prototypisch realisierten Spielweltkomponete ist aufgrund der Entscheidung, die Persistenzkomponente nicht zu implementieren, bisher keine Möglichkeit vorgesehen, Punkte und Protokollinformationen mit Hilfe der jeweiligen Datenzugriffsobjekte (engl. data access objects) persistent zu speichern.

Auch die zentrale Behandlung von Fehlern durch eine Sicherheitsfassade, wie im Entwurf spezifiziert, wird für die prototypische Realisierung im Rahmen dieser Arbeit nicht umgesetzt, da die Behandlung von Fehlern keinen direkten funktionalen Nutzen darstellt.

6.2.2. Spielweltkomponente der Anwendungsarchitektur

Die Implementierung der Schnittstellen der Spielweltkomponente erfolgt auf Basis des Entwurfs server- sowie klientenseitig. Auf dem Server werden dem Entwurf entsprechend die Schnittstellen `GameManagerAdministration`, `GameManager` und `GameWorld` implementiert und für die serverseitige Implementierung eines konkreten pervasiven ortsabhängigen Spiels der betrachteten Klasse bereitgestellt. Die serverseitige Spielweltkomponente kann von den mobilen Endgeräten über die Komponenten für

die nachrichtenorientierte Middleware aufgerufen werden. Dafür wird eine generische `CommunicationMessageListener` Rückrufschnittstelle in der `Communication` Schnittstelle für den Kanal des Servers registriert, die anhand der aus den Nachrichten transformierten Objekte eine Verteilung an dedizierte registrierte Implementierungen der `CommunicationMessageListener` Schnittstelle vornimmt. Wenn eine dieser Implementierungen ein Ergebnis zurückliefert, wird dieses als Nachricht über die `Communication` Schnittstelle an den Kanal des jeweiligen Klienten publiziert.

Auf dem mobilen Endgerät werden dem Entwickler nur die beiden Schnittstellen `GameManager` und `GameWorld` wie spezifiziert mit den beschriebenen Einschränkungen aus Abschnitt 6.1 zur Verfügung gestellt. Da die Verteilung der Architekturschichten wie in Abschnitt 5.1.3 hauptsächlich von dem Grad der Dienstkonnektivität abhängig ist und durch die Anforderungen der Grad der Dienstkonnektivität als Online-Dienst eingestuft wurde, werden die Schnittstellen wie spezifiziert nach dem Stellvertreter-Entwurfsmuster (engl. proxy pattern) (Gamma u. a., 1994) implementiert. Über die `Communication` Schnittstelle werden die Methodenaufrufe nachrichtenorientiert an den Server delegiert. Damit Nachrichten vom Server mit Antworten zu Anfragen oder andere spielspezifische Benachrichtigungen über Ereignisse auf dem mobilen Endgerät empfangen werden können, werden bei der Initialisierung der Spielweltkomponente auf dem mobilen Endgerät Nachrichten über die `Communication` Schnittstelle subskribiert und für den klientenspezifischen Kanal eine dem Entwurf entsprechende generische Rückrufschnittstelle hinterlegt, die Kenntnis über die Verarbeitung der einzelnen Benachrichtigungen hat und die Verteilung an die jeweils in der `GameManager` oder `GameWorld` Schnittstelle registrierten Rückrufschnittstelle vornimmt.

Wenn ein Spieler über die `GameManager` Schnittstelle erfolgreich am System angemeldet wird, wird automatisch die Lokalisierungskomponte auf dem mobilen Endgerät initialisiert und eine Rückrufschnitstelle hinterlegt, die in einem definierten Intervall die aktuelle Position des Spielers an den Server überträgt. Um eine Überflutung mit Nachrichten von häufigen Positionsaktualisierungen der Spieler zu vermeiden, wird die Rückrufschnittstelle wie beschrieben nur in einem definierten Zeitintervall ausgeführt.

6.2.3. Komponenten der Technikarchitektur

Aufgrund der funktionalen Fokussierung auf die Unterstützung der Entwicklung von ortsabhängigen pervasiven Spielen in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, und einem begrenzten Zeitrahmen erfolgt im Rahmen dieser Arbeit die prototypische Realisierung der Komponenten der Technikarchitektur ohne eine explizite Behandlung von technischen Fehlern.

6.2.3.1. Nachrichtenorientierte Middleware

Die Komponente für die nachrichtenorientierte Middleware wird mit dem frei verfügbaren und quelloffenen Produkt XmlBlaster[25] realisiert. Die Nachrichten werden als XML-Dokumente übertragen, wobei die Nutzlast nicht auf XML-Dokumente begrenzt ist, sondern auch anders kodierte Daten enthalten kann. XmlBlaster kann sowohl warteschlangenbasiert als auch durch das Veröffentlichen und Abonnieren von Nachrichten genutzt werden. Obwohl es sich bei XmlBlaster um einen proprietären Ansatz handelt, werden unterschiedliche Programmiersprachen, wie z.B. Java, C# oder Python, durch geeignete Programmierschnittstellen unterstützt. Gleichzeitig kann der Transport über eine Vielzahl von Kommunikationsprotokollen, wie z.B. CORBA, RMI, XmlRpc, HTTP oder SOAP (als Plugin), erfolgen. Das ermöglicht die Anbindung von Klienten mit heterogenen Programmierplattformen und unterschiedlichen Programmiersprachen über die jeweils verfügbaren Kommunikationsprotokolle.

Die Transformation von der Objekt- in eine XML-Repräsentation und umgekehrt wird durch die quelloffene Bibliothek XStream[26] durch eine generisch Implementierung der `XmlTransformer` Schnittstelle sowohl server- als auch klientenseitig realisiert. Ohne weitere Konfiguration ist das transformierte XML nicht vollständig plattformunabhängig. Für eine prototypische Realisierung ist der gewählte Ansatz jedoch ausreichend, da grundsätzlich XML-Dokumente übertragen werden. Auf dem mobilen Endgerät wird eine Portierung der Bibliothek für die Android-Plattform verwendet, die leicht modifiziert wurde.

Für die prototypische Realisierung wurde die `Communication` Schnittstelle, wie im Entwurf spezifiziert umgesetzt. Der Server subskribiert sich zu Beginn für eingehende Nachrichten auf einem dedizierten Kanal, an die die mobilen Endgeräte Nachrichten senden können. Dadurch wird eine logische Ordnung eingehender Nachrichten erreicht. Die Nachrichten vom Server werden an die Kanale von jedem einzelnen mobilen Endgeräte gesendet, wobei jedes mobile Endgerät nur die Nachrichten mit registrierten Ereignissen oder Antworten auf Anfragen erhält.

Für die serverseitige Implementierung der `Communication` Schnittstelle wird die von XmlBlaster bereitgestellte Programmierschnittstelle ohne eine dedizierte Konfiguration eingesetzt. Die Kommunikation erfolgt nicht direkt mit den einzelnen mobilen Endgeräten, sondern lose gekoppelt über einen zentralen Nachrichtenvermittler (engl. message broker), der als zusätzlicher Server im Netzwerk ausgeführt wird. Auf der Android-Plattform erfolgt die Implementierung der `Communication` Schnittstelle aufgrund der begrenzt verfügbaren Ressourcen, wie z.B. Prozessorleistung, Speicherplatz und Energie, auf den mobilen Endgeräten durch eine plattformspezifische Portierung der XmlBlaster JavaME HTTP Programmierschnittstelle. Im Gegensatz zur serverseitig verwendeten Programmierschnittstelle wird durch die Portierung, durch die geringere Größe der Programmierschnittstelle,

[25] http://www.xmlblaster.org/
[26] http://xstream.codehaus.org/

Speicherplatz eingespart und eine Programmierschnittstelle eingesetzt, die speziell für den Einsatz auf mobilen Endgeräten konzipiert wurde. Damit die Kommunikation über HTTP erfolgen kann, wird zusätzlich zum Nachrichtenvermittler ein Stellvertreter (engl. proxy) im Netzwerk benötigt, der die HTTP-Verbindungen der mobilen Endgeräte verwaltet und dann an den Nachrichtenvermittler weiterleitet.

Durch die Komponentenorientierung des vorgestellten Entwurfs stellt die Implementierung mit der nachrichtenorientierten Middleware XmlBlaster nur eine Realisierungsmöglichkeit dar. Daneben existieren noch andere kommerzielle und frei verfügbare Produkte, wie z.B. Apache ActiveMQ[27] oder IBMs MQSeries[28], die eine Implementierung der Schnittstelle ermöglichen würden. Im Kontext von Webdiensten als Kommunikationsprotokoll existiert z.B. die WS-Notification Spezifaktion[29], die einen standardisierten Ansatz bereitstellt und damit die Produkt- und Plattformunabhängigkeit verbessern könnte.

6.2.3.2. Kontextbewusstsein

Die `Context` Schnittstelle der Komponente für das Kontextbewusstsein wird mit Hilfe des anwendungsspezifischen Rahmenwerks WildCAT2[30] implementiert. WildCAT2 ist die Weiterentwicklung des in Abschnitt 4.3.2 vorgestellten Lösungsansatzes und steht frei und quelloffen zur Verfügung. Dabei handelt es sich um ein Rahmenwerk für kontextbewusste Anwendungen, das Kontextinformationen hierarchisch verwaltet und eine an SQL angelehnte Sprache (Event Query Language (EQL)) für die Selektion von Ereignissen in Abhängigkeit der Kontextinformationen bereitstellt. Die Kontextinformationen werden an WildCAT übergeben und unter einem hierarchischen und eindeutigen Pfad abgelegt (Listing 6.1).

```
1 // WildCAT2 Context instanziieren
2 Context context = new BasicContext("gamecontext");
3 // Kontextinformationen unter einem eindeutigen Bezeichner
4 // hinzufügen
5 context.setValue("self://games/players#hans", player);
```

Listing 6.1: Kontextinformationen hinzufügen

Das Beispiel zeigt, wie ein WildCAT2 `Context` Objekt instanziiert wird und anschließend beispielsweise ein Spieler unter einem eindeutigen Ressourcenbezeichner hinzugefügt wird. Um über Änderungen des Spielers benachrichtigt zu werden, ist die Registrierung einer Rückrufschnittstelle in Verbindung mit einer Regel möglich, die festlegt, welche Ereignisse von Interesse sind (Listing 6.2).

[27] http://activemq.apache.org/
[28] http://www.ibm.com/de/
[29] http://www.oasis-open.org/specs/#wsnv1.3
[30] http://wildcat.objectweb.org/index.html

```
// registrieren einer Rückrufschnittstelle, wenn die Ressource
// unter dem Pfad self://games/players#hans geändert wird
context.registerListeners("select * from WAttributeEvent(source='
   self://games/players#hans')", new UpdateListener() {
    void update (EventBean[] nevents, EventBean[] oevents) {
        System.out.println("Ereignis ausgelöst!");
    }
});
// durch das erneute Setzen wird das Ereignis ausgelöst und
// die Methode der Rückrufschnittstelle ausgeführt
context.setValue("self://games/players#hans", player);
```

Listing 6.2: Registrieren für Kontextereignisse

Durch diese Funktionalität deckt WildCAT2 die geforderte Funktionalität der Komponente für das Kontextbewusstsein vollständig ab. Alternative Implementierungen ,z.B. auf der Grundlage des semantischen Webs, sind durch die Komponentenorientierung denkbar, wurden im Rahmen dieser Arbeit aber nicht weiter verfolgt.

6.2.3.3. Lokalisierung

Die `LocationService` Schnittstelle der Lokalisierungskomponente wird plattformspezifisch für die Android-Plattform nach der Spezifikation des Entwurfs implementiert und bereitgestellt. Durch die bereits durch die Android-Plattform zur Verfügung gestellt Programmierschnittstelle besteht die Implementierung fast ausschließlich aus Delegationsaufrufen und wird daher nicht im Detail beschrieben.

6.3. Validierung und Qualitätssicherung

Um die Tragfähigkeit des Entwurfs für ein Rahmenwerk für pervasive ortsabhängige Spiele in einer vermischten Realität, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, zu validieren, wird in diesem Abschnitt eine Testanwendung vorgestellt, allgemein die Durchführung von weiteren Qualitätsmaßnahmen in Form von Klassen- und Komponenten-Tests beschrieben und einzelne geforderte nicht-funktionale Anforderungen an die Architektur des System validiert.

6.3.1. Testanwendung

Die Tragfähigkeit des Entwurfs wird durch eine einfache an das Szenario *King of Location* angelehnte Testanwendung sichergestellt, die einzelne Anwendungsfälle und die damit verbundenen funktionalen Anforderungen aus Abschnitt 4.2.2.1 realisiert. Zur besseren Visualisierung werden im Folgenden elementare Quellcodefragmente beispielhaft angeführt. Der vollständige Quellcode ist auf der beiliegenden CD-ROM abgelegt und kann bei Bedarf eingesehen werden.

Auf dem Server wird das konkrete pervasive ortsabhängige „Spiel“ `TestGame` erstellt und über die Schnittstellen der Spielweltkomponente initialisiert. Dafür besorgt sich der Entwickler des konkreten Spiels zu Beginn Zugriff auf die Schnittstellen `GameWorldServer` und `GameManagerAdministration` der Spielweltkomponenten über die Schnittstelle `CompositionManagerServer`, die über die Fabrik `CompositionManagerServerFactory` bereitgestellt wird (Listing 6.3).

```
1 // Schnittstelle des Kompositionsmanagers über die Fabrik
2 CompositionManagerServer compositionManagerServer =
     CompositionManagerServerFactory.getCompositionManagerServer();
3 // Schnittstelle der Spielwelt-Komponente
4 GameWorldServer gameWorldServer = compositionManagerServer.
     getGameWorldServer();
5 GameManagerServerAdministration gameManagerServerAdministration =
      compositionManagerServer.getGameManagerServerAdministration()
     ;
```

Listing 6.3: Bereitstellung der Schnittstellen der Spielwelt-Komponente

Anschließend wird in der Spielwelkomponente eine neue Spielinstanz über die Schnittstelle `GameManagerAdministration` hinzugefügt (Listing 6.4).

```
1 // eine neue Spielinstanz erstellen
2 Game game = new Game("test");
3 // das Spiel der Spielweltkomponente hinzufügen
4 GameToken gameToken = gameManagerServerAdministration.addGame(
     game);
```

Listing 6.4: Ein Spiel initialisieren

Nachdem die konkrete Spielinstanz in Form eines `Game` Objekts der Spielweltkomponente hinzugefügt wurde, wird z.B. eine Entität mit Position erstellt und wiederum der Spielinstanz hinzugefügt. Dafür wird serverseitig ein `GameToken` benötigt, das eine eindeutige Zuordnung zu dem ausführenden der Aktion sicherstellt (Listing 6.5).

```
// eine Entität mit Position erstellen
PositionedEntity entity = new PositionedEntity("btc");
// die Entität der Spielinstanz hinzufügen
gameWorldServer.setPositionedEntity(gameToken, entity);

// eine Position für die Entität festlegen
Position position = new Position();
position.setLatitude(53.55403042753172f);
position.setLongitude(10.023120045661926f);
// die Position der Entität in der Spielinstanz zuweisen
gameWorldServer.setPosition(entity, position);

// ein Attribute für die Entität erstellen
Attribute attribute = new IntegerAttribute("spieler gesehen", 0);
// das Attribut der Entität in der Spielinstanz hinzufügen
gameWorldServer.setAttribute(entity, attribute);
```

Listing 6.5: Einen Spieler hinzufügen

Die Entität mit Position ist somit Teil der erstellten Spielinstanz mit dem Namen „test“ und besitzt eine Position und ein Attribut mit der Anzahl der Spieler, die bereits die Entität erreicht haben.

Auf dem mobilen Endgerät kann ein Spieler an das System über die `GameManagerClient` Schnittstelle angemeldet werden und dann der Spielinstanz „test“ beitreten. Über die `GameWorldClient` Schnittstelle kann nun dem Spieler in der Testanwendung z.B. das Erreichen der auf dem Server erstellten Entität mit Position als eine Aufgabe zugewiesen werden (Listing 6.6).

```
// die Aufgabe erstellen, wobei die Aufgabe als erreicht gilt,
// wenn sich der Spieler in einem Umkreis von 10 Metern befindet
Challenge challenge = new ReachEntityChallenge(entity, 10);
// die Aufgabe dem Spieler zuordnen
gameWorldClient.addChallenge(player, challenge);
```

Listing 6.6: Eine Aufgabe zuweisen

Damit Änderungen über den Zustand der Aufgabe auf dem mobilen Gerät erfasst werden können, wird eine Rückrufschnittstelle über die `GameWorldClient` Schnittstelle in Verbindung mit einer Abfrage registriert, die eine Regel für das Ausführen der Rückrufschnittstelle darstellt (Listing 6.7).

```
// eine Abfrage für Zustandsänderungen der Aufgabe für Änderungen
```

```
// erstellen
EntityChallengeUpdateQuery query = new EntityChallengeUpdateQuery
    (player, challenge);

// beispielhafte Implementierung der Rückrufschnittstelle
GameListener listener = new GameListener() {

    public void onGameEvent(GameEvent gameEvent) {
        // spielrelevante Logik ausführen, z.B. eine Nachricht
        // anzeigen
        showMessage("Zustand der Aufgabe hat sich geändert!")
    }
});

// die Rückrufschnittstellen für eine übergebene Abfrage
// registrieren
List<GameListener> listenerList =
        new ArrayList<GameListener>();
listenerList.add(listener);
gameWorldClient.registerListener(query, listenerList);
```

Listing 6.7: Ereignisbenachrichtigung abonnieren

Erreicht der Spieler mit dem mobilen Endgerät die Entität mit der Position, die durch die Aufgabe erreicht werden soll, wird auf dem Server ein Ereignis ausgelöst und der Spieler über das mobile Endgerät durch die Ausführung der registrierten Rückrufschnittstelle darüber informiert. Zusätzlich können auch andere mobile Endgeräte den Zustand der Aufgabe über den gleichen Mechanismus beobachten und spielspezifisch auf bestimmte Ereignisse reagieren.

Im Rahmen der Testanwendung sind noch weitere Anwendungsfälle, wie z.B. die Benachrichtigung über die Änderung der Position einer Entität mit Position oder aber die Benachrichtigung über Änderungen von Attributen einer Entität realisiert worden, die aber an dieser Stelle nicht weiter ausgeführt werden.

6.3.2. Testdurchführung

Der komponenten- und entwurfsmusterorientierte Ansatz nach der Quasar-Methode ermöglicht ein modulares Testen einzelner isolierter Schnittstellen von implementierten Komponenten. So werden z.B. anwendungsspezifische Komponenten im Rahmen dieser Arbeit isoliert getestet und die verwendeten Schnittstellen der Kategorie 0 durch vorgetäuschte

Implementierungen (engl. mock objects) ersetzt, die sich jedoch spezifikationsgetreu verhalten.

Für die Implementierung der Testfälle auf Klassen- und Komponentenebene wird das Testrahmenwerk JUnit[31] eingesetzt, damit die Tests automatisiert ausgeführt werden können. Der Integrationstest erfolgt im Rahmen dieser Arbeit lokal auf einem Rechner mit Hilfe des Android-Emulators und der vorgestellten Testanwendung aus Abschnitt 6.3.1.

Positionsänderungen auf dem Android-Emulator können für das Testen der Testanwendung durch vordefinierte Routen über geeignete Dateien, die durch das GPS Austauschformat (GPX) oder mit der Auszeichnungssprache Keyhole Markup Language (KML) beschrieben und simuliert werden. Dadurch kann die Funktionalität der Positionsänderung innerhalb des Rahmenwerks bereits auf Emulator-Ebene auch ohne real verfügbares Geräte mit Positionsbestimmungstechnologie sichergestellt werden.

6.4. Zusammenfassung

In diesem Kapitel wurde die Tragfähigkeit des entwickelten Konzepts aus Abschnitt 5 durch eine prototypische Realisierung sichergestellt. Dazu wurden elementare funktionale Anwendungsfälle der Anwendungsarchitektur implementiert und im vorgestellten Szenario 1 der Architektur der technischen Infrastruktur auf der prototypisch realisierten Technikarchitektur ausgeführt.

Durch den komponenten- und entwurfsmusterorientierten Entwurf nach der Quasar-Methode konnten die einzelnen Komponenten isoliert und anschließend integriert mit Hilfe einer Testanwendung mit den anderen funktionalen und technischen Komponenten nach dem vorgestellten Testkonzept aus Abschnitt 5.5 getestet und validiert werden.

[31] http://www.junit.org/

7. Evaluation

In diesem Kapitel wird das Konzept aus Kapitel 5 und die prototypische Realisierung aus Kapitel 6 mit den funktionalen, technischen und nicht-funktionalen Anforderungen aus Abschnitt 4.2.2 abgeglichen und einer kritischen Betrachtung unterzogen.

7.1. Anforderungsabgleich

In diesem Abschnitt werden die spezifizierten Anforderungen an ein Rahmenwerk für ortsabhängige pervasive Spiele in einer vermischten Realität aus Abschnitt 4.2.2, die ortsdiskrete, aber zeitkontinuierliche Jagd- und Strategiespiele darstellen, mit dem Konzept und der prototypischen Realisierung abgeglichen und dadurch Lücken im Konzept oder in der prototypischen Realisierung für eine Analyse aufgezeigt.

7.1.1. Funktionale Anforderungen

In Tabelle 7.1.1 werden die funktionalen Anforderungen aus Abschnitt 4.2.2.1 aufgeführt und dann jeweils die Erfüllung im Konzept und in der prototypische Realisierung betrachtet.

Anforderung	**Konzept**	**Realisierung**
R-FA-1 (Einhaltung der Regeln, Ablaufsteuerung, Protokollierung)	$\surd$	($\surd$)
R-FA-2 (Eigenschaften von Spielen hinzufügen, entfernen, aktualisieren)	$\surd$	($\surd$)
R-FA-3 (Ereignisbenachrichtigung bei Änderungen von Eigenschaften eines Spiels)	$\surd$	$\surd$
R-FA-4 (Nachrichten an beliebige Entitäten übertragen)	$\surd$	-

R-FA-5 (Auslösung von Ereignissen durch das Rahmenwerk an registrierte Klienten)	√	√
R-FA-6 (Zuweisen und Entfernen von Aufgaben zu Entitäten)	√	(√)
R-FA-7 (Bereitstellung von passenden Aufgaben dem Spielablauf entsprechend)	√	-
R-FA-8 (Benachrichtigung der Klienten über den Status der Aufgaben)	√	√
R-FA-9 (Eigenschaften von Entitäten hinzufügen, entfernen, aktualisieren)	√	(√)
R-FA-10 (Ereignisbenachrichtigung bei Änderungen von Eigenschaften einer Entität)	√	√
R-FA-11 (Hinzufügen und Entfernen von Entitäten zu/von einer Gruppe)	√	-
R-FA-12 (Ereignisbenachrichtung beim Hinzufügen oder Entfernen)	√	(√)
R-FA-13 (Spieler registrieren und authentifizieren)	√	√
R-FA-14 (Ereignisbenachrichtigung bei Positionsänderungen von Entitäten mit Position)	√	√
R-FA-15 (Bereitstellung des Spielstands)	√	-

Tabelle 7.1.: Konzeptrealisierung der funktionalen Anforderungen

Tabelle 7.1.1 zeigt, dass alle funktionalen Anforderungen an das Rahmenwerk für ortsabhängige Spiele der betrachteten Klasse im Konzept berücksichtigt wurden. Aufgrund des begrenzten Zeitrahmens wurden jedoch nicht alle funktionalen Anforderungen prototypisch realisiert (vgl. Abschnitt 6.1). Die prototypische Realisierung deckt jedoch die elementarsten funktionalen Anforderungen ab. Durch die Möglichkeit der Reaktion auf spielrelevante Ereignisse wird die Ablaufsteuerung durch das Rahmenwerk vorgegeben. Der Mechanismus der Ereignisbenachrichtigung ist Bestandteil von mehreren Anforderungen, bezieht sich jedoch auf unterschiedliche Anwendungsfälle. Teilweise wurden funktionale Anforderungen nicht vollständig realisiert, z.B. wurden nicht alle Operationen beim klassischen Erstellen,

Lesen, Aktualisieren und Löschen (engl. create, read, update, delete (CRUD)) umgesetzt. Das Austauschen von Nachrichten zwischen Entitäten ist ein elementarer Bestand von pervasiven ortsabhängigen Strategiespielen. Jedoch ist die Realisierbarkeit implizit durch z.B. das Hinzufügen von Aufgaben oder Eigenschaften abgedeckt und wurde deswegen nicht extra in der prototypischen Realisierung berücksichtigt. Das Gruppenmanagement wurde ebenso nicht explizit realisiert, da die Funktionalität der Ereignisbenachrichtigungen bereits auf der Ebene der Entitäten umgesetzt wurde. Das Gruppenmanagement als solches stellt kein neues Problemfeld dar. Auch die Bereitstellung des Spielstands und die Protokollierung von Spielergebnissen und Zwischenständen bietet keinen direkten Nutzen und keine neuen Erkenntnisse, da die Anforderung keine neuen Problemstellungen umfasst. Daher wurde dies nicht realisiert.

7.1.2. Technische Anforderungen

Die technischen Anforderungen an das Rahmenwerk wurden, wie in Abschnitt 4.2.2.2 angeführt, aus den technischen Anforderungen aus Abschnitt 4.1.3.2 an das konkrete pervasive ortsabhängige Spiel *King of Location* übernommen. In Tabelle 7.1.2 werden die technischen Anforderungen aufgeführt und mit dem Konzept und der Realisierung abgeglichen.

Anforderung	**Konzept**	**Realisierung**
K-TA-1 (Ausführbar auf heterogenen mobilen Endgeräten)	√	√
K-TA-2 (Positionsbestimmung der mobilen Endgeräte über Sensoren)	√	(√)
K-TA-3 (Fähigkeit der Kommunikation mit zentraler Instanz im Internet)	√	(√)
K-TA-4 (Eingabe und Ausgabe der mobilen Endgeräte)	√	√
K-TA-5 (Überwachung eines Spiels durch eine zentrale Instanz im Internet)	√	√
K-TA-6 (Informationsaustausch zwischen zentraler Instanz und mobilen Endgeräten)	√	√
K-TA-7 (Kommunikationsmöglichkeiten zwischen zentraler Instanz und einer Benutzerverwaltung)	√	-

Tabelle 7.2.: Konzeptrealisierung der technischen Anforderungen

Obwohl das Rahmenwerk klientenseitig ausschließlich das Szenario 1 der Architektur für die technische Infrastruktur realisiert und damit auf der Android-Plattform basiert, wird die technische Anforderung K-TA-1 nach der Ausführbarkeit auf heterogenen mobilen Endgeräten erfüllt, da die Android-Plattform auf heterogenen mobilen Endgeräten ausführbar sein wird[32]. Die Positionsbestimmung und die Kommunikation konnten bisher nur auf Emulatorebene bewiesen und getestet werden und müssen bei Verfügbarkeit der ersten mobilen Endgeräte erneut in der Praxis überprüft werden. Die Anbindung der Benutzerverwaltung erfolgt lokal und nicht über einen entfernten Webdienst. Bei Webdiensten handelt es sich jedoch um ein Standardproblem in der Informatik, das bereits häufig gelöst wurde und technisch keine neuen Erkenntnisse in Bezug auf die Entwicklung des Rahmenwerks liefert.

7.1.3. Nicht-funktionale Anforderungen

In Tabelle 7.1.3 werden die nicht-funktionalen Anforderungen aufgeführt und wiederum die Umsetzung mit dem Konzept und der Realisierung abgeglichen.

Anforderung	**Konzept**	**Realisierung**
R-NFA-2 (Portierungsanforderung der Programmierplattformunabhängigkeit)	√	√
R-NFA-3 (Wartbarkeitsanforderungen zur Erweiterbarkeit und Fehlerbehebung)	√	√
R-NFA-4 (Benutzbarkeitsanforderung der Konzentration auf die Fachlichkeit)	√	√
K-NFA-5 (Bereitstellung von Komponenten, Schnittstellen und abstrakten Klassen für die Entwicklung)	√	√
K-NFA-6 (Geringe Einarbeitungszeit der Entwickler durch Entwurfsmuster- und Komponentenorientierung)	√	√

Tabelle 7.3.: Konzeptrealisierung der nicht-funktionalen Anforderungen

[32] Sobald die ersten Geräte von unterschiedlichen Herstellern verfügbar sind.

Die nicht-funktionale operationelle Anforderung R-NFA-1 stellt Anforderungen an die Eigenschaften, die der Ort erfüllen muss, damit ein Spiel stattfinden kann und muss daher für jedes konkrete Spiel überprüft werden. Die Leistungsanforderungen können im Rahmen der prototypischen Realisierung nicht überprüft werden, da durch das Fehlen von realen mobilen Endgeräten für die Android-Plattform ausschließlich auf Emulatorebene getestet wurde und auf Emulatorebene keine realistischen Testszenarien und Testbedingungen erzeugt werden können. In dem Zusammenhang müsste dann auch das Verhalten bei einer steigenden Anzahl an mobilen Endgeräten (Skalierbarkeit) überprüft werden.

Die Möglichkeit der Portierbarkeit ist grundsätzlich durch die Kommunikation über XML-Dokumente gegeben. Jedoch ist einschränkend anzumerken, dass der Aufwand einer klientenseitigen Portierung steigt, wenn z.B. für die Implementierung der `Communication` Schnittstelle keine Implementierung der verwendeten Middleware in der verwendeten Programmiersprache existiert.

Die Anforderungen der Wartbarkeit, Erweiterbarkeit und der Fehlerbehebung hängen von der Architektur des Rahmenwerks ab und können z.B. durch Architekturbewertungsmethoden wie der szenariobasierten Architekturanalysemethode (engl. scenario-based architecture analysis method (SAMM)) bewertet und überprüft werden. Die Anforderungen der Wartbarkeit und Erweiterbarkeit sind besonders bei den variablen Bereichen (engl. hot spots) (vgl. Abschnitt 2.1) in einem Rahmenwerk von elementarer Bedeutung und müssen explizit betrachtet werden. Beim Entwurf des Rahmenwerks in Kapitel 5 wurde aus diesen Gründen besonders viel Wert auf einen komponenten- und entwurfsmusterorientierten Ansatz gelegt, der Zuständigkeiten kapselt und bewährte und allgemein bekannte Lösungsstrategien abbildet. So konnte z.B. eine konkrete Aufgabe, eine bestimmte Entität mit Position zu erreichen, einfach implementiert und einfach in das Rahmenwerk integriert werden.

Die Benutzbarkeitsanforderungen konnten durch die Entwicklung einer konkreten Testanwendung im Rahmen der Realisierung in Kapitel 6 ansatzweise überprüft werden.

7.2. Kritische Betrachtung

Im Rahmen der prototypischen Realisierung wurden keine Android spezifischen Konzepte für die Bereitstellung von Systemdiensten verwendet. Die Android-Plattform stellt ein proprietäres Service-Konzept bereit, das einen Dienst , wie z.B. eine Middleware, im Hintergrund weiter ausführt, auch wenn eine andere Applikation den Fokus erhält, wie z.B. bei einem eingehenden Anruf. Auch wenn die Tragfähigkeit des Entwurfs sichergestellt wurde, müssen bei einer Weiterverfolgung des Szenarios 1 aus der Architektur der technischen Infrastruktur aus Abschnitt 5.3 die Konzepte der Android-Plattform konsequent in der Realisierung berücksichtigt und eingesetzt werden.

Um die Integrität der Daten auf dem Server zu gewährleisten, muss ein Transaktionskonzept in den Entwurf für das Rahmenwerk integriert werden. Für das vorgestellte Szenario 1 der Architektur für die technische Infrastruktur kann z.B. per aspektorientierter Programmierung und der Injizierung von Abhängigkeiten mit Spring gelöst werden. So wäre nur eine deklarative Änderung in der Spring-Konfiguration notwendig, um die Transaktionsgrenzen zu setzen, und kein programmatischer Eingriff in den Quellcode.

Durch den Einsatz einer nachrichtenorientierten Middleware müssen in dem präsentierten Entwurf die Transformationen von der Objekt- in eine XML-Repräsentation und umgekehrt für jedes zu übertragende Objekt explizit implementiert werden. Im Rahmen der prototypischen Realisierung konnte zwar der Aufwand der Implementierung durch den Einsatz der Bibliothek XStream für die Programmiersprache Java reduziert werden, das transformierte XML erfüllt allerdings nicht die nicht-funktionale Anforderung der Portierbarkeit. Um die Anforderung zu erfüllen müssen die Transformationsregeln einzeln ausimplementiert werden, was einen erhöhten Aufwand bedeuten kann und damit unter Umständen auch die nicht-funktionalen Anforderungen der Wartbarkeit und die Erweiterbarkeit beeinflussen kann.

7.3. Methodische Abstraktion

Die durchgeführten Qualitätssicherungsmaßnahmen aus Abschnitt 6.3.1 haben zwar die Tragfähigkeit des Entwurfs bewiesen und die ausgeführten Testfälle haben die Qualität sichergestellt, aber im Gegensatz zu klassischen Testkonzepten für statische verteilte Systeme oder mobile Anwendungen ergeben sich für pervasive Anwendungen neue Herausforderungen:

- Durch die Integration von Kontextinformationen in ein pervasives Spiel müssen neben der reinen Testfallentwicklung auch die Sensoren der mobilen Endgeräte durch vorgetäuschte Objekte (engl. mock objects) ausgetauscht werden.
- Für einen Akzeptanztest im Rahmen eines Testkonzepts sind eine Menge an Ressourcen (Tester) notwendig, die nach dem Testkonzept definierte Testfälle überprüfen und nachstellen. Durch den Einsatz von Sensoren und die Möglichkeit von Verbindungsabbrüchen und Sensorungenauigkeiten zu jeder Zeit ist eine wiederholbare Durchführung von Testszenarien mit einem identischen Ablauf und reproduzierbaren Testergebnissen kaum realisierbar.

Neben der erhöhten Komplexität bei der Entwicklung von pervasiven Spielen und Anwendungen ist auch die Komplexität im Rahmen des Testens gestiegen. Deswegen ist ein Umdenken im Bereich des Testens notwendig. Klassische Integrationstests mit definierten Testdaten und einem definierten Testablauf müssen für den Bereich des pervasiven Rechnens neu definiert werden, um aussagekräftige Testergebnisse zu erhalten.

7.4. Zusammenfassung

In diesem Kapitel wurden die funktionalen, technischen und nicht-funktionalen Anforderungen mit dem Entwurf und der prototypischen Realisierung abgeglichen. Dabei hat sich gezeigt, dass der Entwurf die geforderten Anforderungen abbildet, die prototypische Realisierung jedoch nur Teile davon umsetzt. Anschließend wurden Schwächen beim Entwurf und bei der Realisierung identifiziert und eine Bewertung und eine methodische Abstraktion vorgenommen.

8. Zusammenfassung und Ausblick

In diesem Kapitel wird der Inhalt dieser Arbeit zusammengefasst und die wesentlichen Ergebnisse werden aufgeführt. Abschließend wird ein Ausblick auf mögliche Erweiterungen oder Weiterführungen dieser Arbeit gegeben und einzelne Gebiete werden beispielhaft angeführt.

8.1. Zusammenfassung

Um die Komplexität bei der Entwicklung von pervasiven Spielen zu reduzieren, die sich durch die Anreicherung der physischen Welt mit virtuellen und pervasiven Elementen ergeben, wurde im Rahmen dieser Arbeit ein Rahmenwerk für die Unterstützung der Entwicklung entworfen und prototypisch realisiert.

Um einen Überblick über das Problemfeld zu bekommen, wurden zu Beginn dieser Arbeit (siehe Kapitel 2) grundlegende Themenfelder wie z.B. Middleware, Rahmenwerk, pervasives und kontextbewusstes Rechnen und pervasives Spielen betrachtet, diskutiert und Herausforderungen herausgearbeitet, die im Rahmen eines eigenen Entwurfs für ein Rahmenwerk allgemein beachtet werden müssen.

In Kapitel 3 wurde im Anschluss das konkrete Szenario *King of Location* als Beispiel für ein pervasives Spiel vorgestellt und als ortsabhängiges Spiel in einer vermischten Realität als ortsdiskretes, aber zeitkontinuierliches Jagd- und Strategiespiel klassifiziert. Ausgehend von der Analyse des Szenarios (Kapitel 4) wurden die funktionalen, technischen und nichtfunktionalen Anforderungen an ein Rahmenwerk für pervasive Spiele der betrachteten Klasse methodisch erfasst und für einen Entwurf strukturiert spezifiziert. Auf der Grundlage der funktionalen, technischen und nicht-funktionalen Anforderungen wurde in Kapitel 5 dann ein Entwurf für ein Rahmenwerk für pervasive Spiele der betrachteten Klasse präsentiert und Entwurfsentscheidungen begründet.

Das Rahmenwerk basiert auf einer klassischen Client-Server-Architektur und stellt auf den mobilen Endgeräten sowie auf der zentralen Instanz funktionale Schnittstellen von Komponenten bereit, die von konkreten pervasiven Spielen genutzt werden können, um spielspezifische Logik und Abläufe zu implementieren. Durch den komponentenorientierten Ansatz wird ein wiederverwendbarer modularer Entwurf erreicht, der die Erweiterbarkeit und

Änderbarkeit sicherstellt. Die Kommunikation und Interaktion zwischen den mobilen Endgeräten und der zentralen Instanz basiert auf einem Push-System Ansatz. Dadurch kann die Benachrichtigung bei spielrelevanten Ereignissen ohne ständiges Anfragen der mobilen Endgeräte an die zentrale Instanz realisiert und gleichzeitig die beschränkten Energieressourcen und die Verbindungskosten der mobilen Endgeräte geschont werden. Der Spielkontext wird auf der zentralen Instanz erfasst und spielrelevante Ereignisse kontextbewusst an die mobilen Endgeräte in Abhängigkeit von spielspezifisch konfigurierten Regeln verteilt.

Um die Tragfähigkeit des Entwurfs sicherzustellen, wurde anschließend der Entwurf prototypisch in einer Laborumgebung mit aktuell verfügbaren Technologien realisiert und die Funktionalität durch eine auf dem Rahmenwerk aufsetzende Testanwendung validiert (siehe Kapitel 6), wobei die Funktionalität der Testanwendung aus den Anwendungsfällen und den funktionalen Anforderungen aus der Analyse aus Kapitel 4 abgeleitet wurde.

Durch das entwickelte Rahmenwerk können neue Spielideen und -konzepte mit der Konzentration auf die funktionalen Aspekte realisiert werden, ohne einen Großteil der Entwicklungszeit auf die technischen und nicht-funktionalen Aspekte zu verwenden. Dadurch wird die angeführte Komplexität bei der Entwicklung von pervasiven Spielen reduziert und die Realisierung von neuen Spielideen und -konzepten beschleunigt.

8.2. Ausblick

Bei der Entwicklung von Rahmenwerken können getroffene Entwurfsentscheidungen erst durch wiederholte Anwendung innerhalb der Problemdomäne bestätigt werden (Pree (1994) und (Fach, 2001)). Ein erster Schritt besteht in der vollständigen Realisierung des in dieser Arbeit vorgestellten und prototypisch realisierten Entwurfs und Tests auf real verfügbaren mobilen Endgeräten unter realistischen Einsatzbedingungen. Auch wenn eine grundlegende Tragfähigkeit des Entwurfs im Rahmen dieser Arbeit nachgewiesen wurde, muss das Rahmenwerk iterativ über die Zeit weiter entwickelt und vernachlässigte Aspekte, wie z.B. Leistungs- und Sicherheitsanforderungen, zusätzlich berücksichtigt und integriert werden. Gleichzeitig müssen durch die Entwicklung von konkreten pervasiven Spielen mögliche Lücken im Entwurf des Rahmenwerks identifiziert und durch einen iterativen Entwicklungszyklus beseitigt werden.

Um die Komplexität bei der Entwicklung von pervasiven Spielen weiter zu reduzieren, ist die Entwicklung von weiteren Komponenten denkbar, die auf dem Rahmenwerk aufbauen und die z.B. zusätzlich Elemente der Präsentationsschicht umfassen und z.B. die Darstellung auf einer Landkarte vereinfachen. Die Entwickler von konkreten pervasiven Spielen können dann auf diese Komponenten zurückgreifen und diese anforderungsspezifisch zusammenfügen.

Um die Komplexität bei der Entwicklung von pervasiven Spielen weiter zu reduzieren, ist die Entwicklung einer domänenspezifischen Sprache (engl. domain specific language) denkbar. Auf der Grundlage der Domänenanalyse aus Abschnitt 4.2.1 könnte eine domänenspezifische Sprache für pervasive Spiele der betrachteten Klasse entwickelt werden, die dann wiederum im Rahmen der modellgetriebenen Softwareentwicklung (engl. Model-Driven Software Development (MDSD)) zusammen mit dem Rahmenwerk und entsprechenden Codegeneratoren und Interpretern für die Generierung von lauffähiger Software eingesetzt werden könnte. Darauf aufbauend sind grafische Werkzeuge realisierbar, die von Spieleentwicklern mit dem Fokus auf die Funktionalität, jedoch auch mit nur geringem technischen Verständnis, bedient werden könnten.

Nach Fetter u. a. (2007) können die Erkenntnisse aus der Entwicklung und der Verwendung eines Rahmenwerks für pervasive Spiele verwendet werden, um z.B. den Umgang und die Verarbeitung von Kontextinformationen auf angrenzende Forschungsgebiete wie dem pervasiven und kontextbewussten Rechnen zu übertragen. Durch den pragmatischen Einsatz in einem eingegrenzten Problembereich werden erste praktische Ergebnisse gesammelt, die dann wiederum in die Forschung einfließen können.

A. Inhalt der CD-ROM

Dieser Arbeit liegt eine CD-ROM mit folgender Verzeichnisstruktur bei:

- **[Ausarbeitung]** beinhaltet diese Arbeit im PDF-Format.
- **[Literatur]** beinhaltet in dieser Arbeit verwendete Literatur.
- **[Quellcode]** beinhaltet den Quellcode des prototypisch entwickelten Rahmenwerks und der Testanwendung.
- **[Software]** beinhaltet viele der eingesetzten und frei verfügbaren Bibliotheken und Softwareprodukte.

B. Anwendungsfälle

In diesem Abschnitt werden die identifizierten Anwendungsfälle aus dem Systemkontext aus Abbildung 4.1 durch jeweils eine Anwendungsfallbeschreibung beschrieben und teilweise durch Aktivitätsdiagramme verfeinert. Die Anwendungsfälle bilden die Grundlage für die funktionalen, technischen und nicht-funktionalen Anforderungen (siehe Abschnitt 4.1.3.1, Abschnitt 4.1.3.2 und Abschnitt 4.1.3.3).

B.1. Spiel beitreten

Name	**Spiel beitreten**
Kurzbeschreibung	Der Spieler bekommt eine Auswahl von Spielen in seiner Umgebung angezeigt und kann dann einem Spiel beitreten.
Verwendete Anwendungsfälle	-
Akteure	Spieler
Vorbedingung	Der Spieler ist für das Spiel *King of Location* angemeldet.
Ergebnis	Der Spieler ist dem ausgewählten Spiel beigetreten.
Nachbedingung	Der Spieler kann in ein Team eingeteilt werden.

Standard Ablaufschritte	1. Die Spiele in der Umgebung des Spielers werden abgefragt. 2. Die existierenden Spiele werden dem Spieler angezeigt. 3. Der Spieler wählt ein Spiel aus den angebotenen Spielen aus und tritt dem Spiel bei. 4. Dem Spieler wird eine Erfolgsmeldung eingeblendet. 5. Dem Spieler wird die Spielübersicht angezeigt.
Alternative Ablaufschritte	3.1. Der Spieler betätigt den Schaltknopf „Verlassen“. 4. Der Spieler beendet und verlässt das Spiel *King of Location*. 4.1. Der Spieler bekommt eine Fehlermeldung eingeblendet. 5. Die Spiele in der Umgebung des Spielers werden erneut abgefragt und der Anwendungsfall wird erneut von Beginn an durchlaufen

Tabelle B.1.: Anwendungsfallbeschreibung „Spiel beitreten“

B.2. Ziel auswählen

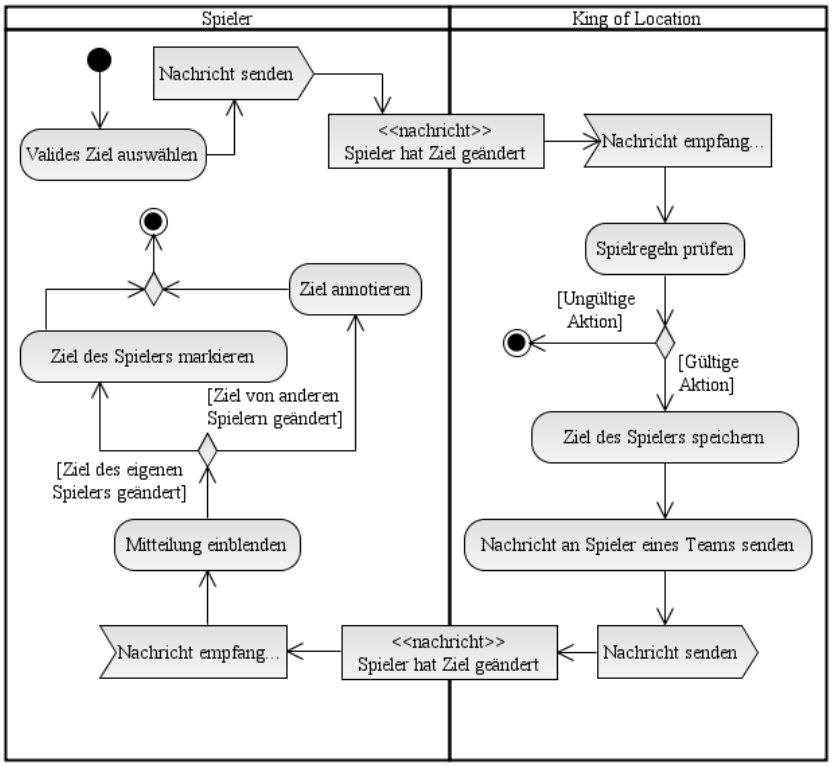

Abbildung B.1.: Aktivitätsdiagramm zum Anwendungsfall „Ziel auswählen“

Name	**Ziel auswählen**
Kurzbeschreibung	Der Spieler wählt ein valides Ziel des aktuellen Spiels aus, das er physisch erreichen möchte.
Verwendete Anwendungsfälle	-
Akteure	Spieler
Vorbedingung	Eine Spielrunde wurde gestartet und die Spielübersicht wird angezeigt.
Ergebnis	Das Ziel, das der Spieler erreichen möchte wird farblich markiert oder bei anderen Mitgliedern aus dem Team durch Zahlen annotiert.
Nachbedingung	Die Spielübersicht wird angezeigt.

Standard Ablaufschritte	1. Der Spieler ändert das Ziel, das er physisch erreichen möchte. 2. Die Spielregeln werden geprüft. 3. Das Ziel das der Spieler physische erreichen möchte wird persistent gespeichert. 4. Die Mitteilung mit dem Ziel, das der Spieler erreichen möchte, wird an alle Mitglieder des Teams übermittelt. 5. Dem Spieler wird eine Mitteilung eingeblendet. 6. Das Ziel, das der Spieler erreichen möchte wird farblich markiert.
Alternative Ablaufschritte	3.1. Die Spielregeln wurden verletzt. 6.1. Das Ziel, das ein Mitglied des Teams erreichen möchte, wird annotiert.

Tabelle B.2.: Anwendungsfallbeschreibung „Ziel auswählen“

B.3. Mitteilung an Teammitglieder schreiben

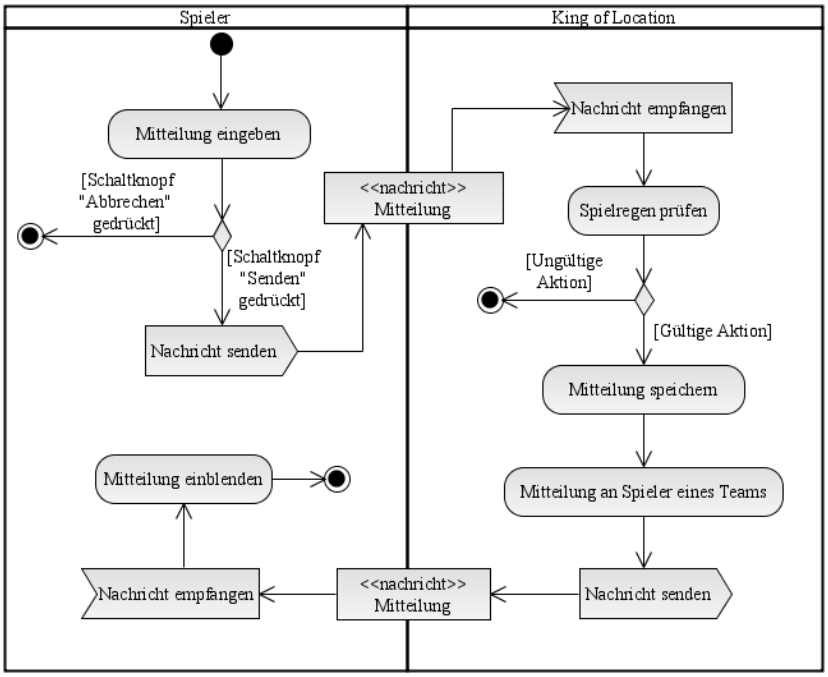

Abbildung B.2.: Aktivitätsdiagramm zum Anwendungsfall „Mitteilung an Teammitglieder schreiben“

Name	**Mitteilung an Teammitglieder schreiben**
Kurzbeschreibung	Der Spieler kann eine Mitteilung an die Mitglieder des eigenen Teams schreiben.
Verwendete Anwendungsfälle	-
Akteure	Spieler
Vorbedingung	Die Spielübersicht wird angezeigt und der Schaltknopf „Mitteilung schreiben“ betätigt.
Ergebnis	Die Mitteilung ist an alle Teammitglieder übermittelt worden.
Nachbedingung	Die Spielübersicht wird angezeigt.

Standard Ablaufschritte	1. Der Spieler gibt eine Mitteilung ein. 2. Die Spielregeln werden geprüft. 3. Die Mitteilung des Spielers wird persistent gespeichert. 4. Die Mitteilung des Spielers wird an allen Mitgliedern eines Teams übermittelt. 5. Die Meldung mit der Mitteilung des Spielers wird allen Mitgliedern einblendet.
Alternative Ablaufschritte	2.1. Die Spielregeln wurden verletzt. 3.1. Die Mitteilung des Spielers wird verworfen.

Tabelle B.3.: Anwendungsfallbeschreibung „Mitteilung an Teammitglieder schreiben“

B.4. Spielübersicht anzeigen

Name	**Spielübersicht anzeigen**
Kurzbeschreibung	Dem Spieler wird die Spielübersicht angezeigt.
Verwendete Anwendungsfälle	-
Akteure	Spieler
Vorbedingung	Eine Spielrunde wurde gestartet.
Ergebnis	Die Spielübersicht wird angezeigt.
Nachbedingung	-
Standard Ablaufschritte	1. Dem Spieler wird eine Karte mit den Positionen aller Mitgliedern seines Teams und der Ziele angezeigt.
Alternative Ablaufschritte	-

Tabelle B.4.: Anwendungsfallbeschreibung „Spielübersicht anzeigen“

B.5. Positionsänderung melden

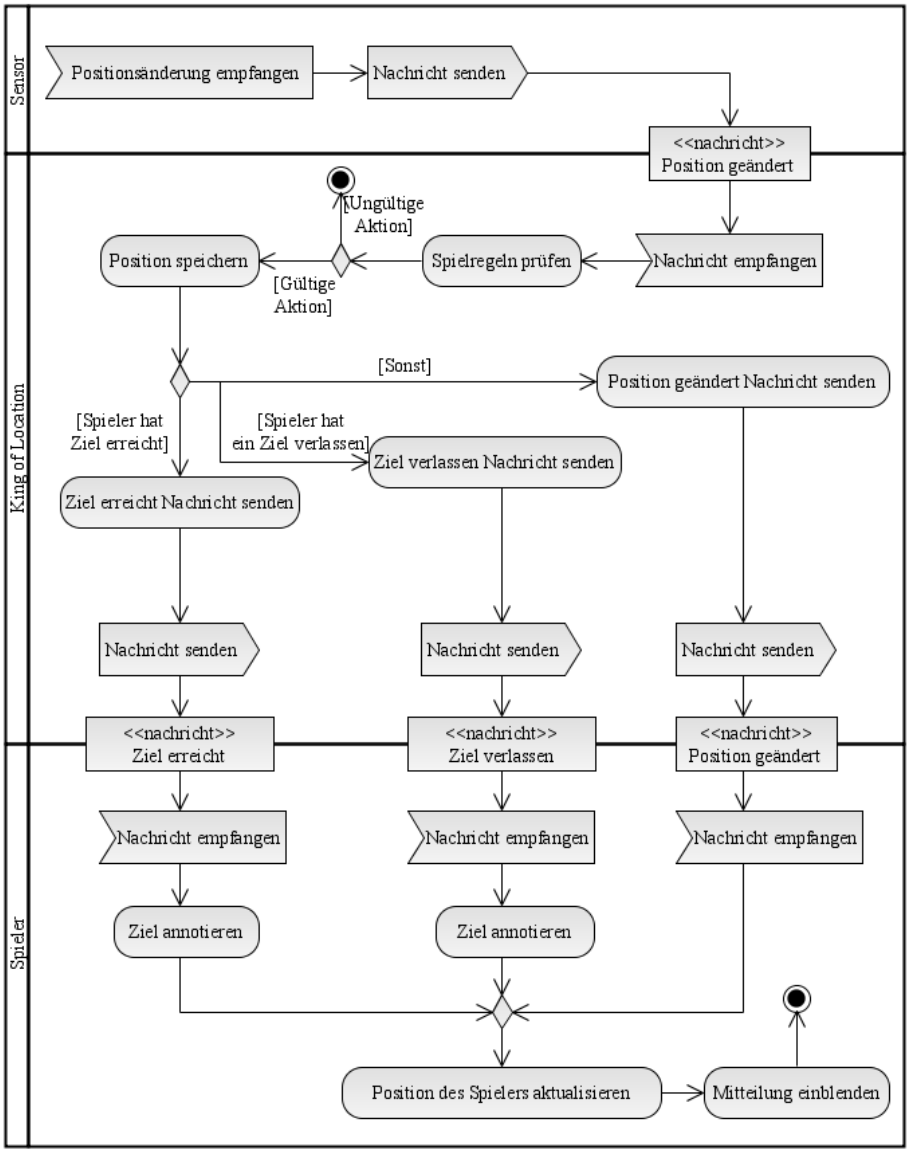

Abbildung B.3.: Aktivitätsdiagramm zum Anwendungsfall „Positionsänderung melden“

Name	**Positionsänderung melden**
Kurzbeschreibung	Positionsänderung des Spielers wird vom Sensor gemeldet und den Spielern eines Teams mitgeteilt.
Verwendete Anwendungsfälle	-

Akteure	Spieler, Sensor
Vorbedingung	Eine Spielrunde wurde gestartet.
Ergebnis	Die Positionsänderung eines Spielers ist an alle Teammitglieder übermittelt worden.
Nachbedingung	-
Standard Ablaufschritte	1. Eine Positionsänderung des Spielers wird vom Sensor gemeldet. 2. Die Spielregeln werden geprüft. 3. Die Position des Spielers wird persistent gespeichert. 4. Die Positionsänderung wird an alle Mitglieder eines Teams gemeldet. 5. Positions des Spielers wird aktualisiert. 6. Die Meldung der Positionsänderung des Spielers wird den Mitgliedern eines Teams eingeblendet.

Alternative Ablaufschritte	3.1. Die Meldung der Positionsänderung des Spielers wird verworfen. 5.1. Das Ziel, das der Spieler erreicht hat, wird annotiert. 6. Die Position des Spielers wird aktualisiert. 7. Die Meldung der Erreichung eines Ziels wird den Mitgliedern eines Teams eingeblendet. 5.2. Das Ziel, das der Spieler verlassen hat, wird annotiert. 6. Die Position des Spielers wird aktualisiert. 7. Die Meldung des Verlassens eines Ziels wird den Mitgliedern eines Teams eingeblendet.

Tabelle B.5.: Anwendungsfallbeschreibung „Positionsänderung melden“

B.6. Punktestand anzeigen

Name	**Punktestand anzeigen**
Kurzbeschreibung	Der Spieler bekommt den Gesamtpunktestand angezeigt.
Verwendete Anwendungsfälle	-
Akteure	Spieler
Vorbedingung	Die Spielübersicht wird angezeigt und der Schaltknopf „Punktestand anzeigen“ betätigt.
Ergebnis	Dem Spieler wird der Gesamtpunktestand angezeigt.
Nachbedingung	-
Standard Ablaufschritte	1. Dem Spieler wird der Gesamtpunktestand angezeigt. 2. Der Spieler betätigt den Schaltknopf „Verlassen“. 3. Dem Spieler wird die Spielübersicht angezeigt.
Alternative Ablaufschritte	-

Tabelle B.6.: Anwendungsfallbeschreibung „Punktestand anzeigen“

B.7. Spieler registrieren

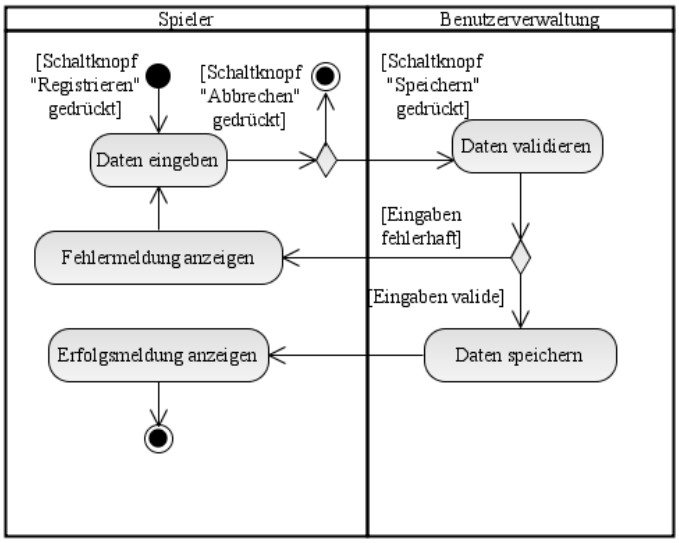

Abbildung B.4.: Aktivitätsdiagramm zum Anwendungsfall „Spieler registrieren“

Name	**Punktestand anzeigen**
Kurzbeschreibung	Der Spieler registriert sich für das Spiel *King of Location* mit Benutzername und Passwort
Verwendete Anwendungsfälle	-
Akteure	Spieler, Benutzerverwaltung
Vorbedingung	Der Spieler betätigt den Schaltknopf „Registrieren“ auf dem Startbildschirm.
Ergebnis	Der Spieler ist registriert.
Nachbedingung	Ein Spieler mit dem gleichen Benutzernamen kann nicht mehr registriert werden.

Standard Ablaufschritte	1. Der Spieler gibt Benutzername und Passwort ein. 2. Die Benutzerverwaltung validiert die Informationen. 3. Die Benutzerverwaltung speichert die Informationen. 4. Dem Spieler wird eine Erfolgsmeldung eingeblendet.
Alternative Ablaufschritte	2.1. Der Spieler betätigt den Schaltknopf „Abbrechen“. 3 Dem Spieler wird der Startbildschirm angezeigt. 3.1. Dem Spieler wird eine Fehlermeldung einblendet. 4 Der Anwendungsfall wird erneut durchlaufen.

Tabelle B.7.: Anwendungsfallbeschreibung „Spieler registrieren“

B.8. Spieler anmelden

Name	**Spieler anmelden**
Kurzbeschreibung	Der Spieler meldet sich am Spiel *King of Location* an.
Verwendete Anwendungsfälle	-
Akteure	Spieler, Benutzerverwaltung
Vorbedingung	Der Spieler betätigt den Schaltknopf „Anmelden“
Ergebnis	Der Spieler ist am Spiel *King of Location* angemeldet.
Nachbedingung	Der Spieler kann einem Spiel beitreten.
Standard Ablaufschritte	1. Der Spieler gibt Benutzername und Passwort ein. 2. Die Benutzerverwaltung validiert die Informationen. 3. Dem Spieler wird eine Erfolgsmeldung eingeblendet.
Alternative Ablaufschritte	2.1. Der Spieler betätigt den Schaltknopf „Abbrechen“. 3 Dem Spieler wird der Startbildschirm angezeigt. 2.2. Der Spieler hat Benutzername und/oder Passwort nicht eingegeben. 3 Dem Spieler wird eine Fehlermeldung eingeblendet. 3.1. Dem Spieler wird eine Fehlermeldung eingeblendet. 4 Der Anwendungsfall wird erneut durchlaufen.

Tabelle B.8.: Anwendungsfallbeschreibung „Spieler anmelden“

C. Funktionale Anforderungen (*Spiel King of Location*)

Anforderungen an die Aktivität „Spiel beitreten“:

K-FA-10 [Wenn der Spieler am Spiel *King of Location* angemeldet ist], soll das Spiel *King of Location* die Spiele in der Umgebung des Spielers auf dem mobilen Endgerät in einer Liste anzeigen.

F-KA-11 [Falls der Spieler den Schaltknopf „Verlassen“ auf dem mobilen Endgerät betätigt hat], soll das Spiel *King of Location* den Startbildschirm auf dem mobilen Endgerät anzeigen.

K-FA-12 [Nachdem der Spieler ein Spiel aus der Liste ausgewählt hat], soll das Spiel *King of Location* den Spieler als Spieler des ausgewählte Spiel speichern und eine Erfolgsmeldung auf dem mobilen Endgerät anzeigen.

K-FA-13 [Falls das Spiel *King of Location* einen Fehler bei der Auswahl eines Spiels oder beim Speichern des Spielers für das ausgewählte Spiel registriert], soll das Spiel *King of Location* eine Fehlermeldung anzeigen.

K-FA-14 [Falls das Spiel *King of Location* eine Fehlermeldung angezeigt hat], soll das Spiel *King of Location* die Spiele in der Umgebung des Spielers erneut auf dem mobilen Endgerät in einer Liste anzeigen.

Anforderungen an die Aktivität „Teamaufteilung“:

K-FA-15 [Nachdem das Spiel *King of Location* eine Erfolgsmeldung beim Beitreten zu einem Spiel angezeigt hat], soll das Spiel *King of Location* die Liste mit allen verfügbaren Teams für das ausgewählte Spiel und die Anzahl der bisher für das Spiel gespeicherten Spieler auf dem mobilen Endgerät anzeigen.

K-FA-16 [Wenn das Spiel eine definierte Anzahl an Spieler für ein Spiel gespeichert wurde], soll das Spiel *King of Location* dem Spieler die Möglichkeit geben, ein Team aus der Liste von Teams auf dem mobilen Endgerät auszuwählen.

K-FA-17 [Nachdem der Spieler ein Team aus der Liste ausgewählt hat], soll das Spiel *King of Location* eine gleichmäßige Verteilung der Spieler auf die verfügbaren Teams sicherzustellen.

K-FA-18 [Falls der Spieler den Schaltknopf „Spiel verlassen“ betätigt hat], soll das Spiel *King of Location* die Liste mit den Spielen in der Umgebung des Spielers auf dem mobilen Endgerät erneut anzeigen.

Anforderungen an die Aktivität „Spielrunde starten“:

K-FA-19 [Nachdem die Spieler auf die Teams verteilt sind], soll das Spiel *King of Location* nach einem definierten Zeitraum eine neue Spielrunde starten.

Anforderungen an die Aktivität „Spielübersicht anzeigen“;

K-FA-20 [Nachdem das Spiel *King of Location* eine neue Spielrunde gestartet hat], soll das Spiel *King of Location* die Spielübersicht anzeigen.

K-FA-21 [Während das Spiel *King of Location* die Spielübersicht anzeigt], soll das Spiel *King of Location* die Positionen der Spieler eines Teams bei Veränderungen aktualisieren.

K-FA-22 [Wenn die Spielübersicht angezeigt wird und der Spieler den Schaltknopf „Spiel verlassen“ betätigt], soll das Spiel *King of Location* das aktuelle Spiel verlassen und den Startbildschirm anzeigen.

Anforderungen an die Aktivität „Positionsänderung melden“:

K-FA-23 [Während das Spiel *King of Location* die Spielübersicht auf dem mobilen Endgerät anzeigt], soll der Sensor Positionsänderungen des Spielers an das Spiel *King of Location* übertragen.

K-FA-24 [Nachdem das Spiel *King of Location* die Positionsänderung des Spielers vollständig empfangen hat], soll das Spiel *King of Location* die Position des Spielers für das ausgewählte Spiel speichern.

K-FA-25 [Nachdem das Spiel *King of Location* die Positionsänderung des spielers für das ausgewählte Spiel gespeichert hat], soll das Spiel *King of Location* eine Meldung auf den mobilen Endgeräten der Spieler anzeigen.

K-FA-26 [Falls das Spiel *King of Location* eine Meldung anzeigt und der Spieler ein Ziel physisch erreicht hat], soll das Spiel *King of Location* das Ziel annotieren und die Anzahl der Spieler des eigenen Teams an diesem Ziel um eins inkrementieren.

K-FA-27 [Falls das Spiel *King of Location* eine Meldung anzeigt und der Spieler ein Ziel physisch verlassen hat], soll das Spiel *King of Location* das Ziel annotieren und die Anzahl der Spieler des eigenen Teams an diesem Ziel um eins dekrementieren.

Anforderungen an die Aktivität „Mitteilung an Teammitglieder schreiben“:

K-FA-28 [Wenn das Spiel *King of Location* die Spielübersicht auf dem mobilen Endgerät anzeigt und der Spieler den Schaltknopf „Mitteilung schreiben“betätigt hat], soll das Spiel *King of Location* dem Spieler die Möglichkeit geben, eine Mitteilung auf dem mobilen Endgerät einzugeben.

K-FA-29 [Nachdem der Spieler eine Mitteilung eingegeben hat], soll das Spiel *King of Location* die Mitteilung speichern und auf dem mobilen Endgerät den Spielern des gleichen Teams anzeigen.

Anforderungen an die Aktivität „Ziel auswählen“:

K-FA-30 [Wenn das Spiel *King of Location* die Spielübersicht auf dem mobilen Endgerät anzeigt und der Spieler ein valides Ziel auf der Spielübersicht auswählt], soll das Spiel *King of Location* das ausgewählte Ziel des Spielers speichern.

K-FA-31 [Nachdem das Spiel *King of Location* das Ziel des Spielers gespeichert hat], soll das Spiel *King of Location* eine Meldung mit dem neuen Ziel des Spielers auf den mobilen Endgeräten der Spieler eines Teams anzeigen.

K-FA-32 [Während das Spiel *King of Location* die Meldung mit dem neuen Ziel des Spielers auf dem mobilen Endgerät des Spielers (der das Ziel ausgewählt hat) anzeigt] soll das Spiel *King of Location* das ausgewählte Ziel farblich markieren.

K-FA-33 [Während das Spiel *King of Location* die Meldung mit dem neuen Ziel des Spielers auf dem mobilen Endgerät der anderen Spieler des Teams anzeigt], soll das Spiel *King of Location* das ausgewählte Ziel annotieren.

Anforderungen an die Aktivität „Punktestand anzeigen“:

K-FA-34 [Wenn das Spiel *King of Location* die Spielübersicht auf dem mobilen Endgerät anzeigt und der Spieler auf den Schaltknopf „Punktestand anzeigen“ betätigt hat], soll das Spiel *King of Location* den Punktestand anzeigen.

K-FA-35 [Wenn das Spiel *King of Location* den Punktestand anzeigt und der Spieler den Schaltknopf „Verlassen“ betätigt hat], soll das Spiel *King of Location* die Spielübersicht erneut anzeigen.

Anforderungen an die Aktivität „Spielrunde beenden“:

K-FA-36 [Wenn die definierte Spielzeit abgelaufen ist], soll das Spiel *King of Location* die aktuelle Spielrunde beenden.

Anforderungen an die Aktivität „Ergebnis der Spielrunde anzeigen“:

K-FA-37 [Nachdem das Spiel *King of Location* die aktuelle Spielrunde beendet hat], soll das Spiel *King of Location* das Ergebnis der Spielrunde auf dem mobilen Endgerät anzeigen.

K-FA-38 [Wenn das Spiel *King of Location* das Ergebnis der Spielrunde anzeigt und der Spieler den Schaltknopf „Verlassen“ betätigt], soll das Spiel *King of Location* die Spiele in der Umgebung des Spielers auf dem mobilen Endgerät erneut in einer Liste anzeigen.

Anforderungen an die Aktivität „Spieler registrieren“:

K-FA-39 [Nachdem der Spieler den Schaltknopf „Spieler registrieren“ auf dem Startbildschirm auf dem mobilen Endgerät betätigt hat], soll das Spiel *King of Location* fähig sein, die Eingabe des Benutzernamens und des Passworts des Spielers auf dem mobilen Endgerät zu ermöglichen.

K-FA-40 [Falls der Spieler den Schaltknopf „Abbrechen“ bei der Eingabe des Benutzernamens und des Passworts betätigt hat], soll das Spiel *King of Location* den Startbildschirm anzeigen.

K-FA-41 [Wenn der Spieler den Schaltknopf „Speichern“ nach der Eingabe des Benutzernamens und des Passworts betätigt hat], soll das Spiel *King of Location* den Benutzernamen und das Passwort an die Benutzerverwaltung übertragen.

K-FA-42 [Nachdem die Benutzerverwaltung den Benutzernamen und das Passwort vollständig empfangen hat], soll die Benutzerverwaltung den Benutzernamen und das Passwort des Spielers validieren.

K-FA-43 [Wenn die Benutzerverwaltung den Benutzernamen und das Passwort validiert hat], soll die Benutzerverwaltung eine Erfolgs- oder Fehlermeldung an das Spiel *King of Location* übertragen.

K-FA-44 [Falls der Spieler den Schaltknopf „Speichern“ betätigt hat und kein Benutzername und/oder kein Passwort eingeben wurde], soll das Spiel *King of Location* eine Fehlermeldung auf dem mobilen Endgerät anzeigen.

K-FA-45 [Nachdem das Spiel *King of Location* die Erfolgs- oder Fehlermeldung vollständig von der Benutzerverwaltung empfangen hat], soll das Spiel *King of Location* die Meldung anzeigen.

K-FA-46 [Falls das Spiel *King of Location* eine Fehlermeldung anzeigt], soll das Spiel *King of Location* dem Spieler die Möglichkeit geben, den Benutzernamen und das Passwort erneut einzugeben.

Damit sichergestellt werden kann, dass die definierten Spielregeln eingehalten werden, wird zusätzlich noch folgende allgemeine funktionale Anforderung aufgenommen:

F-FA-47 Das Spiel *King of Location* soll die Einhaltung der definierten Spielregeln sicherstellen und überwachen.

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

Glossar

Augmented Reality
Unter einer erweiterter Realität (engl. Augmented Reality) werden computergestützte Erweiterung der Realitätswahrnehmung verstanden.

Geografische Informationssysteme
Ein Geographisches Informationssystem (GIS) ist ein rechnergestütztes Informationssystem, das aus Hardware, Software, Daten und den Anwendungen besteht und mit dem raumbezogene Daten digital erfasst und redigiert, gespeichert und reorganisiert, modelliert und analysiert sowie alphanumerisch und grafisch präsentiert werden können.

Head Mounted Display
Ein Head Mounted Display ist ein auf dem Kopf getragenes visuelles Ausgabegerät, das am Computer erzeugte Bilder auf einem augennahen Bildschirm darstellt oder direkt auf die Netzhaut projiziert.

Mixed Reality
Unter einer vermischten Realität (engl. mixed reality) werden Umgebungen oder Systeme zusammengefasst, die die physische Welt mit einer virtuellen Realität vermischen.

Semantisches Web
Die Idee des semantischenWebs (engl. semantic web) reichert das weltweite Web (World Wide Web (WWW)) um formal beschriebenen semantische Informationen an, die durch Maschinen automatisiert verarbeitet werden können.

SOAP
SOAP (bis Version 1.1 war SOAP das Akronym für Simple Access Object Protocol) ist ein leichtgewichtiges Protokoll gedacht für den Austausch von strukturierten Informationen in dezentralen und verteilten Umgebungen. SOAP nutzt XML für die Beschreibung von Nachrichten und ermöglicht deren Austausch über eine Vielzahl von zugrunde liegenden Protokollen.

Abkürzungsverzeichnis

AI	Artificial Intelligence.
API	Application Programming Interface.
AR	Augmented Reality.
CORBA	Common Object Request Broker Architecture.
CRUD	Create, Read, Update, Delete.
EQL	Event Query Language.
GPRS	General Packet Radio Service.
GPS	Global Positioning System.
GPX	GPS Exchange Format.
HMD	Head Mounted Display.
HTTP	Hypertext Transfer Protocol.
ISO	International Organization for Standardization.
JSR	Java Specification Request.
KML	Keyhole Markup Language.
LAN	Local Area Network.
LBG	Location Based Games.
LGPL	GNU Lesser General Public License.
MAR	Mobile Augmented Reality.
MDSD	Model-Driven Software Development.
MOM	Message-Oriented Middlware.
MR	Mixed Reality.
MUPE	Mulit-User Publishing Environment.
POT	Physical Object Tagger.

PUD	Public Display.
REST	Representational State Transfer.
RMI	Remote Method Invocation.
RPC	Remote Procedure Call.
SAAM	Scenario-Based Architecture Analysis Method.
SIC	Silent Communicator.
SOA	Service-Oriented Architecture.
SOM	Service Oriented Middleware.
UML	Unified Modeling Language.
UMTS	Universal Mobile Telecommunications System.
URI	Uniform Resource Identifier.
WLAN	Wireless Local Area Network.
XML	Extensible Markup Language.

Literaturverzeichnis

[Abowd u. a. 1999] ABOWD, Gregory D. ; DEY, Anind K. ; BROWN, Peter J. ; DAVIES, Nigel ; SMITH, Mark ; STEGGLES, Pete: Towards a Better Understanding of Context and Context-Awareness. In: *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag, 1999, S. 304–307

[Akman und Surav 1996] AKMAN, Varol ; SURAV, Mehmet: Steps Toward Formalizing Context. In: *AI Magazine* 17 (1996), Nr. 3, S. 55–72

[Baldauf u. a. 2007] BALDAUF, Matthias ; DUSTDAR, Schahram ; ROSENBERG, Florian: A survey on context-aware systems. In: *Int. J. Ad Hoc Ubiquitous Comput.* 2 (2007), Nr. 4, S. 263–277

[Balzert 1996] BALZERT, Helmut: *Lehrbuch der Software-Technik: Teil 1: Software-Entwicklung*. Heidelberg, Germany : Spektrum Akademischer Verlag, 1996

[Banavar und Bernstein 2002] BANAVAR, Guruduth ; BERNSTEIN, Abraham: Software infrastructure and design challenges for ubiquitous computing applications. In: *Commun. ACM* 45 (2002), Nr. 12, S. 92–96

[Bartelme 2005] BARTELME, Norbert: *Geoinformatik: Modelle, Strukturen, Funktionen*. 4. vollständig überarbeitete Aufl. Springer, Berlin, 2005

[Benford u. a. 2006] BENFORD, Steve ; CRABTREE, Andy ; FLINTHAM, Martin ; DROZD, Adam ; ANASTASI, Rob ; PAXTON, Mark ; TANDAVANITJ, Nick ; ADAMS, Matt ; ROWFARR, Ju: Can you see me now? In: *ACM Trans. Comput.-Hum. Interact.* 13 (2006), Nr. 1, S. 100–133

[Benford u. a. 2005] BENFORD, Steve ; MAGERKURTH, Carsten ; LJUNGSTRAND, Peter: Bridging the physical and digital in pervasive gaming. In: *Commun. ACM* 48 (2005), Nr. 3, S. 54–57

[Bernstein 1996] BERNSTEIN, Philip A.: Middleware: a model for distributed system services. In: *Commun. ACM* 39 (1996), Nr. 2, S. 86–98

[Bhaskar und Ahamed 2007] BHASKAR, Pankaj ; AHAMED, Sheikh I.: Privacy in Pervasive Computing and Open Issues. In: *ARES '07: Proceedings of the The Second International Conference on Availability, Reliability and Security*. Washington, DC, USA : IEEE Computer Society, 2007, S. 147–154

[Bisgaard u. a. 2005] BISGAARD, Jesper J. ; HEISE, Morten ; STEFFENSEN, Carsten: *How is Context and Context-awareness Defined and Applied? A Survey of Context-awareness*. 2005

[Björk u. a. 2001] BJÖRK, S. ; FALK, J. ; HANSSON, R. ; LJUNGSTRAND, P.: Pirates! - Using the Physical World as a Game Board. In: *Proc. Interact IFIP TC.13 Conference on Human-Computer Interaction*. Tokyo, Japan, 2001

[Björk u. a. 2003] BJÖRK, Staffan ; LUNDGREN, Sus ; HOLOPAINEN, Jussi: Game Design Patterns. In: *DIGRA Conf.*, 2003

[Book u. a. 2005] BOOK, Matthias ; GRUHN, Volker ; HÜLDER, Malte ; SCHÄFER, Clemens: Der Einfluss verschiedener Mobilitätsgrade auf die Architektur von Informationssystemen. In: HAMPE, J. F. (Hrsg.) ; LEHNER, Franz (Hrsg.) ; POUSTTCHI, Key (Hrsg.) ; RANNENBERG, Kai (Hrsg.) ; TUROWSKI, Klaus (Hrsg.): *MCTA* Bd. 59, GI, 2005, S. 117–130

[Bosch u. a. 2000] BOSCH, Jan ; MOLIN, Peter ; MATTSSON, Michael ; BENGTSSON, PerOlof: Object-oriented framework-based software development: problems and experiences. In: *ACM Comput. Surv.* (2000), S. 3

[Bregman] BREGMAN, Marc: Interview With Marc Bregman. The Convenience of Small Devices: How Pervasive Computing Will Personalize EBusiness.

[Brown 1996] BROWN, M.: Supporting User Mobility. In: *IFIP World Conference on Mobile Communications*, 1996, S. 69–77

[Buschmann u. a. 1998] BUSCHMANN, Frank ; MEUNIER, Regine ; ROHNERT, Hans ; SOMMERLAD, Peter ; STAL, Michael: *Pattern-orientierte Software-Architektur . Ein Pattern-System.* 2. Auflage. Addison-Wesley, 1998

[Chalmers 2005] CHALMERS, M.;Brown B.;Hall M.;Sherwood S.;Tennent P.: Gaming on the Edge: Using Seams in Pervasive Games. In: *Pervasive 2005 workshop on Pervasive Games (PerGames 2005)*, 2005

[Chen und Kotz 2000] CHEN, Guanling ; KOTZ, David: A Survey of Context-Aware Mobile Computing Research. Hanover, NH, USA : Dartmouth College, 2000. – Forschungsbericht

[Cheok u. a. 2004] CHEOK, Adrian D. ; GOH, Kok H. ; LIU, Wei ; FARBIZ, Farzam ; FONG, Siew W. ; TEO, Sze L. ; LI, Yu ; YANG, Xubo: Human Pacman: a mobile, wide-area entertainment system based on physical, social, and ubiquitous computing. In: *Personal Ubiquitous Comput.* 8 (2004), Nr. 2, S. 71–81

[Cooperstock u. a. 1995] COOPERSTOCK, Jeremy R. ; TANIKOSHI, Koichiro ; BEIRNE, Garry ; NARINE, Tracy ; BUXTON, William A. S.: Evolution of a reactive environment. In: *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM Press/Addison-Wesley Publishing Co., 1995, S. 170–177

[Crawford 2003] CRAWFORD, Chris: *Chris Crawford on Game Design*. Thousand Oaks, CA, USA : New Riders Publishing, 2003

[Crowley u. a. 2002] CROWLEY, James L. ; COUTAZ, Joëlle ; REY, Gaeten ; REIGNIER, Patrick: Perceptual Components for Context Aware Computing. In: *UbiComp '02: Proceedings of the 4th international conference on Ubiquitous Computing*. London, UK : Springer-Verlag, 2002, S. 117–134

[Csikszentmihalyi 1990] CSIKSZENTMIHALYI, Mihaly: *Flow the Psychology of Optimal Experience*. Harpercollins Publisher, 1990

[David und Ledoux 2005] DAVID, Pierre-Charles ; LEDOUX, Thomas: WildCAT: a generic framework for context-aware applications. In: *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA : ACM, 2005, S. 1–7

[Dey u. a. 2001] DEY, A. ; SALBER, D. ; ABOWD, G.: *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*. 2001

[Dey 1998] DEY, Anind K.: Context-aware computing: The CyberDesk project. In: *AAAI 1998 Spring Symposium on Intelligent Environments*. Palo Alto : AAAI Press., 1998, S. 51–54. – URL `http://www.cc.gatech.edu/fce/cyberdesk/pubs/AAAI98/AAAI98.html`. – Abruf: 01.07.2008

[Dustdar u. a. 2003] DUSTDAR, Schahram ; GALL, Harald C. ; HAUSWIRTH, Manfred: *Software-Architekturen für Verteilte Systeme*. Springer Verlag GmbH, 2003

[Ebert 2005] EBERT, Christof: *Systematisches Requirements Management*. Heidelberg : Dpunkt verlag, 2005

[Elrod u. a. 1993] ELROD, Scott ; HALL, Gene ; COSTANZA, Rick ; DIXON, Michael ; RIVIÈRES, Jim des: Responsive Office Environments. In: *Commun. ACM* 36 (1993), Nr. 7, S. 84–85

[Emmerich 2000] EMMERICH, Wolfgang: Software engineering and middleware: a roadmap. In: *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*. New York, NY, USA : ACM, 2000, S. 117–129

[Fach 2001] FACH, Peter W.: Design Reuse through Frameworks and Patterns. In: *IEEE Softw.* 18 (2001), Nr. 5, S. 71–76

[Fayad und Schmidt 1997] FAYAD, Mohamed ; SCHMIDT, Douglas C.: Object-oriented application frameworks. In: *Commun. ACM* 40 (1997), Nr. 10, S. 32–38

[Fayad u. a. 1999] FAYAD, Mohamed E. ; SCHMIDT, Douglas C. ; JOHNSON, Ralph E.: *Building application frameworks: object-oriented foundations of framework design*. New York, NY, USA : John Wiley & Sons, Inc., 1999

[Fetter u. a. 2007] FETTER, Mirko ; ETZ, Markus ; BLECHSCHMIED, Heiko: Mobile chase - towards a framework for location-based gaming. In: *GRAPP (AS/IE)*, INSTICC - Institute for Systems and Technologies of Information, Control and Communication, 2007, S. 98–105

[Fickas u. a. 1997] FICKAS, Stephen ; KORTUEM, Gerd ; SEGALL, Zary: Software Organization for Dynamic and Adaptable Wearable Systems. In: *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1997, S. 56

[Floréen u. a. 2005] FLORÉEN, P. ; PRZYBILSKI, M. ; NURMI, P. ; KOOLWAAIJ, J. ; TARLANO, A. ; WAGNER, M. ; LUTHER, M. ; BATAILLE, F. ; BOUSSARD, M. ; MROHS, B. ; LAU, S.: Towards a Context Management Framework for MobiLife. In: *IST Mobile Communications Summit*. Dresden, Germany, 2005

[Franklin 2001] FRANKLIN, Michael J.: Challenges in Ubiquitous Data Management. In: *Informatics - 10 Years Back. 10 Years Ahead.* London, UK : Springer-Verlag, 2001, S. 24–33

[Fullerton 2008] FULLERTON, Tracy: *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. 2. edition. Morgan Kaufmann, 2008

[Gamma u. a. 1994] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design patterns: elements of reusable object-oriented software*. Addison-Wesley Professional, 1994

[Gelernter 1985] GELERNTER, David: Generative communication in Linda. In: *ACM Trans. Program. Lang. Syst.* 7 (1985), Nr. 1, S. 80–112

[Greenhalgh u. a. 2007] GREENHALGH, C. ; BENFORM, S. ; DROZD, A. ; FLINTHAM, M. ; HAMPSHIRE, A. ; OPPERMANN, L. ; SMITH, K. ; TYCOWICZ, C. von: *EQUIP2: A Platform for Mobile Phone-Based Game Development*. Bd. 1. S. 153–178. In: MAGERKURTH, Carsten (Hrsg.) ; RÖCKER, Carsten (Hrsg.): *Concepts and technologies for Pervasive Games - A Reader for Pervasive Gaming Research* Bd. 1. Aachen, Germany : Shaker Verlag, 2007

[Gupta u. a. 2002] GUPTA, Sandeep Kumar S. ; LEE, Wang-Chien ; SATYANARAYANAN, Mahadev: Editorial for special issue on pervasive computing. In: *Mob. Netw. Appl.* 7 (2002), Nr. 4, S. 255–257

[Gwizdka 2000] GWIZDKA, Jacek: What's in the Context? (2000)

[Harris u. a. 2004] HARRIS, Eric ; FITZPATRICK, Geraldine ; ROGERS, Yvonne ; PRICE, Sara ; PHELPS, Ted ; RANDELL, Cliff: From snark to park: lessons learnt moving pervasive experiences from indoors to outdoors. In: *AUIC '04: Proceedings of the fifth conference on Australasian user interface*. Darlinghurst, Australia, Australia : Australian Computer Society, Inc., 2004, S. 39–48

[Henricksen u. a. 2001] HENRICKSEN, Karen ; INDULSKA, Jadwiga ; RAKOTONIRAINY, Andry: Infrastructure for Pervasive Computing: Challenges. In: *GI Jahrestagung (1)*, 2001, S. 214–222

[Hinske u. a. 2007] HINSKE, Steve ; LAMPE, Matthias ; MAGERKURTH, Carsten ; RÖCKER, Carsten: *Classifying Pervasive Games: On Pervasive Computing and Mixed Reality*. Bd. 1. S. 11–38. In: MAGERKURTH, Carsten (Hrsg.) ; RÖCKER, Carsten (Hrsg.): *Concepts and technologies for Pervasive Games - A Reader for Pervasive Gaming Research* Bd. 1. Aachen, Germany : Shaker Verlag, 2007

[Hofer u. a. 2003] HOFER, Thomas ; SCHWINGER, Wieland ; PICHLER, Mario ; LEONHARTSBERGER, Gerhard ; ALTMANN, Josef ; RETSCHITZEGGER, Werner: Context-Awareness on Mobile Devices - the Hydrogen Approach. In: *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9*. Washington, DC, USA : IEEE Computer Society, 2003, S. 292.1

[Holleis u. a. 2006] HOLLEIS, Paul ; KRANZ, Matthias ; WINTER, Anneke ; SCHMIDT, Albrecht: Playing with the Real World. In: *Journal of Virtual Reality and Broadcasting* 3 (2006), Nr. 1

[Holler und Illing 2005] HOLLER, Manfred J. ; ILLING, Gerhard: *Einführung in die Spieltheorie*. 6., überarbeitete Auflage. Springer, Berlin, 2005

[Hong und Landay 2001] HONG, J. ; LANDAY, J.: *An Infrastructure Approach to Context-Aware Computing*. 2001

[Huang u. a. 1999] HUANG, Andrew C. ; LING, Benjamin C. ; PONNEKANTI, Shankar: Pervasive computing: what is it good for? In: *MobiDe '99: Proceedings of the 1st ACM international workshop on Data engineering for wireless and mobile access*. New York, NY, USA : ACM, 1999, S. 84–91

[Huizinga 1938] HUIZINGA, Johan: *Homo Ludens: Vom Ursprung der Kultur im Spiel*. 19. Auflage. Rowohlt Tb., 1938

[Hull u. a. 1997] HULL, R. ; NEAVES, P. ; BEDFORD-ROBERTS, J.: Towards situated computing. In: *Wearable Computers, 1997. Digest of Papers., First International Symposium on* (1997), Oct, S. 146–153

[Issarny u. a. 2007] ISSARNY, Valerie ; CAPORUSCIO, Mauro ; GEORGANTAS, Nikolaos: A Perspective on the Future of Middleware-based Software Engineering. In: *FOSE '07: 2007 Future of Software Engineering*. Washington, DC, USA : IEEE Computer Society, 2007, S. 244–258

[Jegers 2007] JEGERS, Kalle: Pervasive game flow: understanding player enjoyment in pervasive gaming. In: *Comput. Entertain.* 5 (2007), Nr. 1, S. 9

[Johnson und Foote 1988] JOHNSON, R. ; FOOTE, B.: Designing reusable classes. In: *Journal of Object-Oriented Programming*, 1988, S. 22–35

[Johnson 1997] JOHNSON, Ralph E.: Components, frameworks, patterns. In: *SSR '97: Proceedings of the 1997 symposium on Software reusability*. New York, NY, USA : ACM, 1997, S. 10–17

[Karlsson 1995] KARLSSON, Even-André (Hrsg.): *Software reuse: a holistic approach*. New York, NY, USA : John Wiley & Sons, Inc., 1995

[Kiefer u. a. 2006] KIEFER, P. ; MATYAS, S. ; SCHLIEDER, C.: Systematically Exploring the Design Space of Location-based Games. In: *Pervasive 2006 Workshop Proceedings, Poster presented at PerGames2006*. Dublin, Ireland, May 2006, S. 183–190

[Kiefer u. a. 2007] KIEFER, Peter ; MATYAS, Sebastian ; SCHLIEDER, Christoph: *Geogames Integrating Edutainment Content in Location-Based Games*. Bd. 1. S. 127–152. In: MAGERKURTH, Carsten (Hrsg.) ; RÖCKER, Carsten (Hrsg.): *Concepts and technologies for Pervasive Games - A Reader for Pervasive Gaming Research* Bd. 1. Aachen, Germany : Shaker Verlag, 2007

[Kindberg und Fox 2002] KINDBERG, Tim ; FOX, Armando: System Software for Ubiquitous Computing. In: *IEEE Pervasive Computing* 1 (2002), Nr. 1, S. 70–81

[Kirk u. a. 2005] KIRK, Douglas ; ROPER, Marc ; WOOD, Murray: Identifying and Addressing Problems in Framework Reuse. In: *IWPC '05: Proceedings of the 13th International Workshop on Program Comprehension*. Washington, DC, USA : IEEE Computer Society, 2005, S. 77–86

[Kirk u. a. 2007] KIRK, Douglas ; ROPER, Marc ; WOOD, Murray: Identifying and addressing problems in object-oriented framework reuse. In: *Empirical Softw. Engg.* 12 (2007), Nr. 3, S. 243–274

[Klabbers 2003] KLABBERS, Jan H. G.: The gaming landscape: a taxonomy for classifying games and simulations. In: *DIGRA Conf.*, 2003

[Kuikkaniemi u. a. 2006] KUIKKANIEMI, Kai ; TURPEINEN, Marko ; SALOVAARA, Antti ; SAARI, Timo ; VUORENMAA, Janne: Toolkit for user-created augmented reality games. In: *MUM '06: Proceedings of the 5th international conference on Mobile and ubiquitous multimedia*. New York, NY, USA : ACM, 2006, S. 6

[Landin und Niklasson 1995] LANDIN, Niklas ; NIKLASSON, Axel: *Development of Object-Oriented Frameworks*. Lund, Schweden, Lund University, Diplomarbeit, 1995

[Lieberman und Selker 2000] LIEBERMAN, H. ; SELKER, T.: Out of context: computer systems that adapt to, and learn from, context. In: *IBM Syst. J.* 39 (2000), Nr. 3-4, S. 617–632

[Lindley 2003] LINDLEY, Craig A.: Game Taxonomies: A High Level Framework for Game Analysis and Design, 2003

[Linthicum 2004] LINTHICUM, David: *Next Generation Application Integration*. Boston : Addison-Wesley, 2004

[Lyytinen und Yoo 2002] LYYTINEN, Kalle ; YOO, Youngjin: *Issues and challenges in ubiquitous computing*. 2002

[Magerkurth u. a. 2005] MAGERKURTH, Carsten ; CHEOK, Adrian D. ; MANDRYK, Regan L. ; NILSEN, Trond: Pervasive games: bringing computer entertainment back to the real world. New York, NY, USA : ACM, 2005

[Markopoulos u. a. 2005] MARKOPOULOS, Panos ; RUYTER, Boris de ; PRIVENDER, Saini ; BREEMEN, Albert van: Case study: bringing social intelligence into home dialogue systems. In: *interactions* 12 (2005), Nr. 4, S. 37–44

[Matos und Fernandes 2006] MATOS, S.N. ; FERNANDES, C.T.: Early Definition of Frozen and Hot Spots in the Development of Domain Frameworks. In: *Fourteenth ACM SIGSOFT Symposium on Foundations of Software Engineering* (2006)

[Mattern 2005] MATTERN, Friedemann: Allgegenwärtige und verschwindende Computer. In: *Praxis der Informationsverarbeitung und Kommunikation (PIK)* 28 (2005), Januar, Nr. 1, S. 29–36

[Mattern 2007] MATTERN, Friedemann: Was bedeuten Pervasive und Ubiquitous Computing? In: *asut-Bulletin* (2007), Nr. 4, S. 33

[Mattern 2008] MATTERN, Friedemann: *Allgegenwärtige Informationsverarbeitung – Technologietrends und Auswirkungen des Ubiquitous Computing*. S. 3–29. In: ROSSNAGEL, Alexander (Hrsg.) ; SOMMERLATTE, Tom (Hrsg.) ; WINAND, Udo (Hrsg.): *Digitale Visionen - Zur Gestaltung allgegenwärtiger Informationstechnologien*. Berlin Heidelberg New York : Springer, April 2008

[Mattsson u. a. 1999] MATTSSON, Michael ; BOSCH, Jan ; FAYAD, Mohamed E.: Framework integration problems, causes, solutions. In: *Commun. ACM* 42 (1999), Nr. 10, S. 80–87

[Montola 2005] MONTOLA, Markus: Exploring the edge of the magic circle. Defining pervasive games. In: *In Proceedings of the 2005 DAC Conference*, 2005

[Moore 1965] MOORE, Gordon E.: Cramming more components onto integrated circuits. In: *Electronics* 38 (1965), S. 114–117

[Nakajima 2003] NAKAJIMA, Tatsuo: Pervasive Servers: A framework for creating a society of appliances. In: *Personal Ubiquitous Comput.* 7 (2003), Nr. 3-4, S. 182–188

[Nicklas u. a. 2001] NICKLAS, Daniela ; PFISTERER, Christoph ; MITSCHANG, Bernhard: Towards Location-based Games. In: SING, Alfred Loo W. (Hrsg.) ; MAN, Wan H. (Hrsg.) ; WAI, Wong (Hrsg.) ; NING, Cyril T. (Hrsg.): *Proceedings of the International Conference on Applications and Development of Computer Games in the 21st Century: ADCOG 21; Hongkong Special Administrative Region, China, November 22-23 2001*. Hong Kong : Division of Computer Studies, City University of Hong kong, Hong Kong SAR, China, November 2001, S. 61–67

[Nieuwdorp 2007] NIEUWDORP, Eva: The pervasive discourse: an analysis. New York, NY, USA : ACM, 2007, S. 13

[Nurmi und Floreen 2004] NURMI, Petteri ; FLOREEN, Patrik: *Reasoning in Context-Aware Systems*. 2004

[O. Davidsson 2004] O. DAVIDSSON, S.Björk: Game Design Patterns for Mobile Games. In: *Project Report to Nokia Research Center* (2004)

[Papazoglou und Georgakopoulos 2003] PAPAZOGLOU, M. P. ; GEORGAKOPOULOS, D.: *Service Oriented Computing*. 2003

[Pascoe u. a. 2000] PASCOE, Jason ; RYAN, Nick ; MORSE, David: Using while moving: HCI issues in fieldwork environments. In: *ACM Trans. Comput.-Hum. Interact.* 7 (2000), Nr. 3, S. 417–437

[Pascoe 1998] PASCOE, Mr. J.: Adding Generic Contextual Capabilities to Wearable Computers. In: *ISWC '98: Proceedings of the 2nd IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 1998, S. 92

[Payne und Macdonald 2004] PAYNE, R. ; MACDONALD, B.: Ambient Technology — Now You See It, Now You Don't. In: *BT Technology Journal* 22 (2004), Nr. 3, S. 119–129

[Peitz u. a. 2007] PEITZ, Johan ; SAARENPÄÄ, Hannamari ; BJÖRK, Staffan: Insectopia: exploring pervasive games through technology already pervasively available. In: *ACE '07: Proceedings of the international conference on Advances in computer entertainment technology*. New York, NY, USA : ACM, 2007, S. 107–114

[Pree 1994] PREE, Wolfgang: Meta Patterns - A Means For Capturing the Essentials of Reusable Object-Oriented Design. In: *ECOOP '94: Proceedings of the 8th European Conference on Object-Oriented Programming*. London, UK : Springer-Verlag, 1994, S. 150–162

[Prekop und Burnett 2003] PREKOP, Paul ; BURNETT, Mark: Activities, Context and Ubiquitous Computing. In: *Computer Communications* 26 (2003), S. 1168

[Rekimoto u. a. 1998] REKIMOTO, J. ; AYATSUKA, Y. ; HAYASHI, K.: Augment-able reality: situated communication through physical and digital spaces. In: *Wearable Computers, 1998. Digest of Papers. Second International Symposium on* (1998), Oct, S. 68–75

[Reussner und Hasselbring 2006] REUSSNER, Ralf ; HASSELBRING, Wilhelm: *Handbuch der Software-Architektur*. dpunkt Verlag, 2006. – ISBN 3-89864-372-7

[Robertson und Robertson 2006] ROBERTSON, Suzanne ; ROBERTSON, James: *Mastering the Requirements Process (2nd Edition)*. Addison-Wesley Professional, 2006

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

[Rothermel u. a. 2003] ROTHERMEL, Kurt ; BAUER, Martin ; BECKER, Christian: Digitale Weltmodelle - Grundlage kontextbezogener Systeme. In: MATTERN, Friedemann (Hrsg.): *Total vernetzt - Szenarien einer informatisierten Welt*. Berlin, Heiderlberg, New York : Springer-Verlag, Mai 2003 (Xpert.press), S. 123–141

[Rupp 2007] RUPP, Chris: *Requirements-Engineering und -Management*. 4., aktualisierte und erweiterte Auflage. Carl Hanser Verlag, 2007

[Salen und Zimmerman 2003] SALEN, Katie ; ZIMMERMAN, Eric: *Rules of Play: Game Design Fundamentals*. The MIT Press, 2003

[Satyanarayanan 2001] SATYANARAYANAN, M.: Pervasive computing: Vision and challenges. In: *IEEE Personal Communications*, 2001, S. 10–17

[Schafer 1999] SCHAFER, S.: *Ten dimensions of ubiquitous computing.* 1999

[Schilit u. a. 1994] SCHILIT, Bill ; ADAMS, Norman ; WANT, Roy: Context-Aware Computing Applications. In: *IEEE Workshop on Mobile Computing Systems and Applications*. Santa Cruz, CA, US, 1994

[Schilit und Theimer 1994] SCHILIT, Bill ; THEIMER, M.: Disseminating Active Map Information to Mobile Hosts. In: *IEEE Network* 8 (1994), Nr. 5, S. 22–32

[Schlieder u. a. 2006] SCHLIEDER, Christoph ; KIEFER, Peter ; MATYAS, Sebastian: Geogames: Designing Location-Based Games from Classic Board Games. In: *IEEE Intelligent Systems* 21 (2006), Nr. 5, S. 40–46

[Schmidt 2002] SCHMIDT, Albrecht: *Ubiquitous Computing - Computing in Context*, University of Lancaster, PhD, 2002

[Schmidt u. a. 1999a] SCHMIDT, Albrecht ; AIDOO, Kofi A. ; TAKALUOMA, Antti ; TUOMELA, Urpo ; LAERHOVEN, Kristof V. ; VELDE, Walter V. de: Advanced Interaction in Context. In: *HUC '99: Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing*. London, UK : Springer-Verlag, 1999, S. 89–101

[Schmidt u. a. 1999b] SCHMIDT, Albrecht ; BEIGL, Michael ; GELLERSEN, Hans-W.: There is more to context than location. In: *Computers and Graphics* 23 (1999), Nr. 6, S. 893–901

[Schmidt u. a. 2006] SCHMIDT, Albrecht ; SPIEKERMANN, Sarah ; GERSHMAN, Anatole ; MICHAHELLES, Florian: Real-World Challenges of Pervasive Computing. In: *IEEE Pervasive Computing* 5 (2006), Nr. 3, S. 91–93, c3

[Schneider und Kortuem 2001] SCHNEIDER, J. ; KORTUEM, G.: How to Host a Pervasive Game - Supporting Face-toFace Interactions in Live-Action Roleplaying, 2001

[Siedersleben 2004] SIEDERSLEBEN, Johannes: *Moderne Software-Architektur: Umsichtig planen, robust bauen mit Quasar*. 1. Auflage. Dpunkt Verlag, 2004

[Siedersleben (Hrsg.) 2003] SIEDERSLEBEN (HRSG.), Johannes: *Quasar: Die sdm Standardarchitektur. Teil 1 und 2*. 2. Auflage. sdm Research, 2003

[De Souza e Silva 2004] SILVA, Adriana De Souza e: Location based mobile games: Blurring the borders between physical and virtual spaces. In: *Proceedings of the 12th International Symposium of Electronic Art (ISEA)* Bd. 3166, 2004

[Singh u. a. 2006] SINGH, Sachin ; PURADKAR, Sushil ; LEE, Yugyung: Ubiquitous computing: connecting Pervasive computing through Semantic Web. In: *Inf. Syst. E-Business Management* 4 (2006), Nr. 4, S. 421–439

[Suomela u. a. 2004] SUOMELA, Riku ; RÄSÄNEN, Eero ; KOIVISTO, Ari ; MATTILA, Jouka: Open-Source Game Development with the Multi-user Publishing Environment (MUPE) Application Platform. In: *ICEC* Bd. 3166, Springer, 2004, S. 308–320

[Sweetser und Wyeth 2005] SWEETSER, Penelope ; WYETH, Peta: GameFlow: a model for evaluating player enjoyment in games. In: *Comput. Entertain.* 3 (2005), Nr. 3

[Szyperski 2002] SZYPERSKI, Clemens: *Component Software: Beyond Object-Oriented Programming*. Boston, MA, USA : Addison-Wesley Longman Publishing Co., Inc., 2002

[Tanenbaum und van Steen 2003] TANENBAUM, Andrew S. ; STEEN, Maarten van: *Verteilte Systeme*. PEARSON STUDIUM, 2003

[Thackara 2001] THACKARA, John: The design challenge of pervasive computing. In: *interactions* 8 (2001), Nr. 3, S. 46–52

[Tugui 2004] TUGUI, Alexandru: Calm technologies in a multimedia world. In: *Ubiquity* 5 (2004), Nr. 4, S. 1–1

[Wallmüller 2001] WALLMÜLLER, Ernest: *Software - Qualitätsmanagement in der Praxis*. 2. Auflage. Carl Hanser Verlag, 2001

[Walther 2005a] WALTHER, Bo K.: Atomic actions – molecular experience: theory of pervasive gaming. In: *Comput. Entertain.* 3 (2005), Nr. 3, S. 4–4

[Walther 2005b] WALTHER, Bo K.: Notes on the Methodology of Pervasive Gaming. In: KISHINO, Fumio (Hrsg.) ; KITAMURA, Yoshifumi (Hrsg.) ; KATO, Hirokazu (Hrsg.) ; NAGATA, Noriko (Hrsg.): *ICEC* Bd. 3711, Springer, 2005, S. 488–495

[Want und Pering 2005] WANT, Roy ; PERING, Trevor: System challenges for ubiquitous & pervasive computing. In: *ICSE '05: Proceedings of the 27th international conference on Software engineering*. New York, NY, USA, 2005, S. 9–14

[Weiser u. a. 1999] WEISER, M. ; GOLD, R. ; BROWN, J. S.: The origins of ubiquitous computing research at PARC in the late 1980s. In: *IBM Syst. J.* 38 (1999), Nr. 4, S. 693–696

[Weiser] WEISER, Mark: *Mark Weiser's Homepage*. – URL `http://www.ubiq.com/weiser/`. – Abruf: 01.07.2008

[Weiser 1991] WEISER, Mark: The computer for the 21th century. In: *Scientific American*, 1991

[Weiser 1993] WEISER, Mark: Some computer science issues in ubiquitous computing. New York, NY, USA : ACM, 1993, S. 75–84

[Weiser und Seely Brown 1995] WEISER, Mark ; SEELY BROWN, John: Designing Calm Technology. In: *Xerox PARC* (1995). – URL `http://www.ubiq.com/hypertext/weiser/calmtech/calmtech.htm`. – Abruf: 01.07.2008

[Widjaja und Balbo 2005] WIDJAJA, Ivo ; BALBO, Sandrine: Spheres of role in context-awareness. In: *OZCHI '05: Proceedings of the 19th conference of the computer-human interaction special interest group (CHISIG) of Australia on Computer-human interaction*. Narrabundah, Australia, Australia : Computer-Human Interaction Special Interest Group (CHISIG) of Australia, 2005, S. 1–4

[Winograd 2001] WINOGRAD, Terry: Architectures for Context. In: *Human-Computer Interaction* 16 (2001), Nr. 2-4

[Zeidler 2007] ZEIDLER, Andreas: *Event-based Middleware for Pervasive Computing. Foundations, Concepts, Design*. Vdm Verlag Dr. Müller, 2007

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 5. Dezember 2008	
Ort, Datum	Unterschrift