Hochschule für Angewandte Wissenschaften Hamburg
*Hamburg University of Applied Sciences*

# AW2 Projektbericht

Maik Weindorf

Pervasive Gaming Framework

*Fachbereich Elektrotechnik und Informatik*
*Department of Electrical Engineering and Computer Science*

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Kapitel 1

# Einleitung

Das Projekt *Pervasive Gaming Framework* wurde im Rahmen der AW2 Veranstaltung des Master Studiengangs Informatik an der HAW Hamburg durchgeführt. Am Projekt waren 6 Studierende, 1 betreuender Professor, sowie ein Projektassistent beteiligt. Über den Zeitraum eines Semesters wurde ein Framework für ortsbezogene Spiele auf mobilen Geräten entwickelt.

## 1.1 Motivation

Warum wurde ausgerechnet das Thema *Pervasive Gaming Framework* gewählt?

Im Projekt sollten sich die Interessen aller Beteiligten wiederfinden. Wunschthemen waren unter Anderem *Location bases Services*, *Transaktionen in verteilten Systemen*, *Dokumentenmanagement* und *Autonomic Computing*. Besonders aufgrund des Interesses an *Location bases Services*, sowie einem allgemeinen Interesse an mobilen Systemen und Spielen, fiel die Entscheidung für den Themenbereich *ortsbezogene Spiele*. Da die Entwicklung eines einzelnen konkreten Spiels dem Anspruch des Master Studiengangs nicht gerecht werden würde, wurde entschieden, ein *Pervasive Gaming Framework*[1] zu entwickeln und ein prototypisches Spiel zur Evaluierung auf Basis des Frameworks aufzubauen.

## 1.2 Spielidee

Für die Entwicklung und Evaluierung des Frameworks war es notwendig, eine Spielidee für einen Prototypen zu entwerfen. Abgesehen von einer Spielidee, musste dabei eine Klasse von Spielen identifiziert werden, die das Framework unterstützen sollte. Dabei war zu beachten,

[1] Eine nähere Betrachtung des Framework Begriffs erfolgt in Kapitel 2.1.2

dass ein zu allgemeingültiger Ansatz die Funktionaliät des Frameworks reduzieren würde, da sehr viel der konkreten Spiellogik in den auf dem Framework basierenden Spielen stecken müsste. Einen Ansatz lieferte dabei die Bachelorarbeit von Jan Napitupulu mit dem Thema *Entwicklung einer ortsabhängigen Spielesoftware für Pervasive Gaming*. Als Spielprinzip dient in dieser Arbeit eine ortsbezogene Handelssimulation mit mobilen Geräten. Dabei ist der Handel innerhalb des Spiels nur an bestimmten Orten der realen Welt möglich. Das Spielkonzept wird also mit der realen Welt verknüpft.

Als Metapher für die vom Framework zu unterstützenden Spiele wurde das Thema *Schnitzeljagd* gewählt. Das Prinzip hinter dieser Metapher ist ebenfalls, dass bestimmte Aktionen innerhalb des Spiels nur an bestimmten Orten der realen Welt möglich sind. Diese Metapher lässt sich leicht abstrahieren und auf weitere Klassen von Spielen oder Anwendungen, wie z.B. Rollenspiele und Städtetouren anwenden.

## 1.3 Projektverlauf

In der Projektgruppe wurde nach einer Phase der allgemeinen Themenfindung und Projektvorbereitung entschieden, das Projekt in zwei Phasen durchzuführen. In der ersten Phase wurde anhand von Technologie-Prototypen untersucht, welche Technologien und Konzepte sich für die endgültige Lösung verwenden lassen. Für diese erste Phase standen 2 Wochen zur Verfügung. In den 10 Wochen der zweiten Phase wurden sowohl das Konzept und die Architektur konkretisiert, als auch die Entwicklung des Frameworks und eines Spiel-Prototypen durchgeführt.

Zu Beginn des Projekts waren als Projektziele die Fertigstellung des Frameworks, sowie die Fertigstellung einer *OE-Spiel Version* definiert. Diese Spielversion sollte für die Orientierungseinheit der Erstsemester eingesetzt werden und eine Schnitzeljagd, bzw. ein Quiz zur HAW Hamburg darstellen. Im Laufe des Projekts wurde dieses Ziel verworfen und stattdessen eine Prototyp-Version einer *Glühwein-Edition* umgesetzt. Diese Spielversion hatte als Thema eine Schnitzeljagd durch Hamburg zu ausgesuchten Glühweinständen.

## 1.4 Schwerpunkt des Berichts

Innerhalb des Projektes gab es eine grobe Aufteilung in Server- und Client-Gruppe. Jedes Teammitglied hatte innerhalb dieser Teilgruppen verschiedenen Themenschwerpunkte. Der Author dieses Berichts war Mitglied der Client-Gruppe und hatte als Themenschwerpunkte, den Entwurf einer *Routen Beschreibungssprache*, sowie die Entwicklung zugehöriger Klassen und Methoden zum Parsen und Verarbeiten von Routen Informationen. Die *Routen Beschreibungssprache* war dabei nicht auf den Client beschränkt, sondern auch für die

Realisierung der Server Logik relevant. Darüber hinaus war der Author an der Entwicklung verschiedener GUI Prototypen beteiligt.

Der Schwerpunkt dieses Berichts liegt daher, abgesehen von der Beschreibung der Team- und Projekt-übergreifenden Grundlagen und konzeptionellen Entscheidungen, auf der Beschreibung und Erläuterung der *Routen Beschreibungssprache* in Verbindung mit den zugehörigen Klassen und Methoden (siehe Kapitel 4).

# Kapitel 2

# Grundlagen

## 2.1 Begriffe

### 2.1.1 Pervasive Gaming

*Pervasive Gaming* ist ein Unterthema von *Pervasive Computing* und wird häufig gleichgesetzt mit *Ubiquitous Computing*, bzw. *Ubiquitous Gaming*. Die Definitionen sind in diesem Bereich nicht eindeutig. Einen Ansatz zur Definition von *Pervasive Gaming* liefert folgende Definition:

> Pervasive gaming integrates the technical approaches of computer gaming with emerging interface, wireless and positioning technologies to create game experiences that combine both virtual and physical game elements. [Waern, 2004]

Der wichtigste Aspekt von *Pervasive Gaming* ist eindeutig, die Verknüpfung von physikalischer und virtueller Umgebung. Abbildung 2.1 zeigt eine Einordnung des Themenbereichs *Pervasive Gaming*, in Abhängigkeit von der Unterstützung durch Computer und der Verknüpfung von physikalischer und virtueller Umgebung.

### 2.1.2 Framework

Wie in Kapitel 1.1 beschrieben, wurde im Projekt entschieden, nicht nur ein konkretes Spiel zu erstellen, sondern ein Framework für eine Klasse von Spielen. Es ist daher wichtig, zu erläutern, was der Begriff *Framework* eigentlich bedeutet und wie er von anderen Ansätzen, wie beispielsweise einer Klassenbibliothek abgegrenzt werden kann.

Ein Framework gibt einen Kontrollfluss und eine Architektur vor [Gamma, 1996]. Ein Framework kann konkrete Funktionalität beinhalten, ist dabei aber keine eigenständige Anwendung. Auf dem Framework aufbauende Anwendungen, instantiieren das Framework

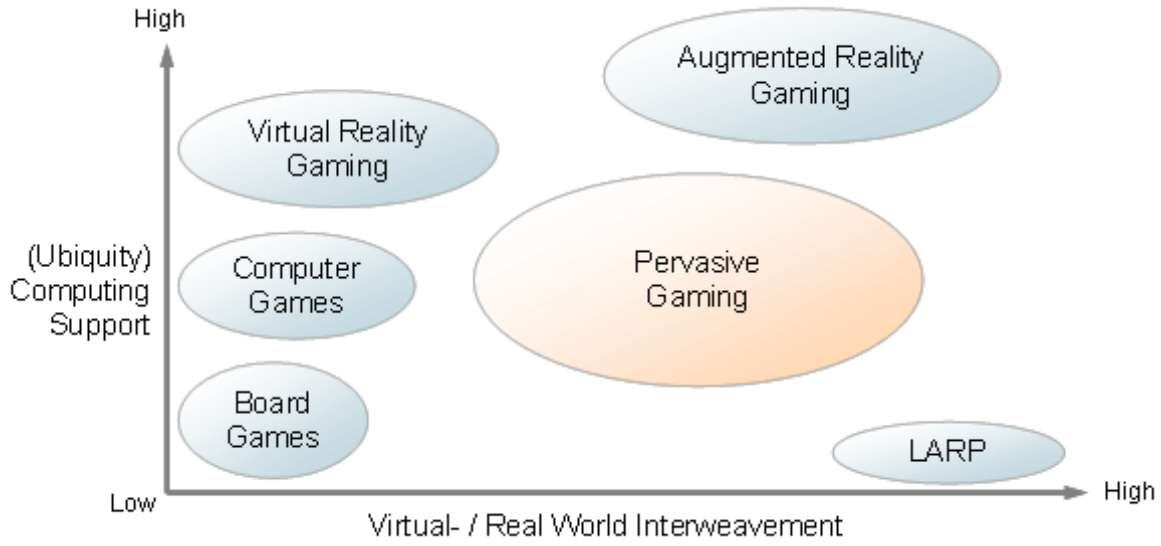



Abbildung 2.1: Pervasive Gaming [OKS, 2004]

und ergänzen es um ihre eigene Funktionalität. Da das Framework für den Kontrollfluss zuständig ist, werden Komponenten des Frameworks nicht durch die konkreten Anwendungen aufgerufen, sondern das Framework ruft Komponenten der Anwendungen auf. Dieses Prinzip wird auch *Inversion of Control* genannt.

Dieser Ansatz unterscheidet sich stark von einer Klassenbibliothek. Die Aufgabe einer Klassenbibliothek ist hauptsächlich die Kapselung von Funktionalität. Anwendungen, die eine Klassenbibliothek verwenden müssen sich nicht mit den Implementierungsdetails der von der Klassenbibliothek angebotenen Funktionalität auseinandersetzen. Die Verantwortung für den Kontrollfluss und die Architektur liegt jedoch bei der konkreten Anwendung.

## 2.2 Verwandte Arbeiten/Projekte

Das Thema *Pervasive Gaming* erfreut sich eines wachsenden Interesses. Es gibt dabei durchaus unterschiedliche Ansätze, sich diesem Thema anzunähern. Eine intensive Betrachtung relevanter Arbeiten und Projekte würde den Rahmen dieser Ausarbeitung sprengen. Daher werden an dieser Stelle lediglich einige Ansätze kurz vorgestellt. Für eine nähere Betrachtung sei auf die angegebenen Quellen verwiesen.

**CRON** ist ein Gemeinschaftsprojekt des Fachgebiets *Offene Kommunikationssysteme*[1] der TU Berlin und des *Fraunhofer Institute for Open Communication Systems*. Es handelt

[1] http://www.oks.cs.tu-berlin.de/forschung/pervasive-gaming/

sich dabei um ein PDA basiertes Multiplayer fähiges pervasives Spiel, welches auf einem von den oben genannten Institutionen entwickelten Framework für kontextsensitive multimedia Applikationen in mobilen Umgebungen aufbaut. Ziel des Spiels ist es, WLAN Hotspots mit seinem Team zu "besetzen". [Linner et al., 2005]

**Botfighters** ist ein pervasives Spiel für GSM Mobiltelefone. Die Position der Spieler wird dabei grob durch die Cell-ID ermittelt. Jeder Spieler repräsentiert einen Roboter und kann andere Roboter in Reichweite bekämpfen[2].

**Uncle Roy All around You** wird sowohl von mobilen, als auch von online Spielern gespielt[3]. Die online Spieler müssen dabei nach Landmarken in einer virtuellen Umgebung suchen. Diese virtuelle Umgebung wird auf eine reale Umgebung gemappt. Ziel ist dann, die mobilen Spieler zu den entsprechenden Orten in der realen Umgebung zu lotsen.

**ARQuake** basiert auf *Augmented Reality*, der Verwendung von *Head-Mounted Displays* und *Umgebungserkennung*[4]. Das Spiel bildet den berühmten Egoshooter *Quake* auf die reale Welt ab. Der Spieler sieht über das *Head-Mounted Displays* Objekte der Spielwelt in die reale Welt eingeblendet.

## 2.3 Infrastruktur

An dieser Stelle werden die technischen Rahmenbedingungen des Projekts erläutert.

**Hardware** Als mobile Geräte kamen im Projekt Smartphones vom Typ *HP iPAQ HW6915* zum Einsatz. Diese Geräte verfügen unter Anderem über GPS, WLAN, Bluetooth und GPRS. Als Betriebssystem dient *Windows Mobile 5.0*. Als Plattform für die Serverkomponente diente ein einfacher PC mit Netzwerkanbindung. Zusätzlich wurden für Testzwecke das laboreigene LAN- und WLAN-Netz genutzt.

**Software** Wie schon erwähnt, wurde auf den mobilen Clients *Windows Mobile 5.0* eingesetzt. Für die Entwicklung der Client- und Server-Software wurden *Microsoft Visual Studio 2005*, das *.NET Compact Framework 2.0* und *C#* verwendet. Zusätzlich kam *Microsoft SQL Server 2005* zum Einsatz.

[2] http://www.botfighters.com
[3] http://www.uncleroyallaroundyou.co.uk/
[4] http://wearables.unisa.edu.au/projects/ARQuake/www/

# Kapitel 3

# Konzept

Für die Konzeption des Frameworks war es notwendig einige Annahmen über Rahmenbedingungen zu treffen und Entscheidungen zu fällen, die Auswirkungen auf die Architektur, die Use Cases und die gesamte Entwicklung hatten. Die wichtigsten Annahmen und Entscheidungen sind folgende:

- Clients können nicht immer Online sein
- Client und Server kommunizieren via Web Services
- Kommunikation wird ausschließlich von den Clients initiiert

## 3.1 Architektur

Die Abbildungen 3.1 und 3.2 zeigen die Grobarchitektur des Frameworks. Im Folgenden wird hauptsächlich auf die Client-Komponenten des Frameworks eingegangen. Für eine Betrachtung der Server-Komponenten sei an dieser Stelle auf die Berichte der Server-Gruppe verwiesen. (in Kürze zu finden auf [UbiComp])

Wie aus Abbildung 3.1 hervorgeht, sind die Client-Komponenten des Frameworks der *Administration Manager, Game Manager, Context Manager* mit integriertem *Session Manager und Task Manager*, sowie ein *Configuration Manager, Location Layer* und *Communication Layer*. Die Client GUI ist nicht Bestandteil des Frameworks und muss für jede Anwendung individuell entwickelt werden. Die im Rahmen dieser Ausarbeitung relevanten Komponenten und ihr Zusammenspiel werden im Folgenden beschrieben.

**Game Manager** Der *Game Manager* gibt den Kontrollfluss des Spiels vor und kapselt die Funktionalität der unter ihm liegenden Schichten. Über einen Event Listener erhält er Informationen vom *Context Manager*.

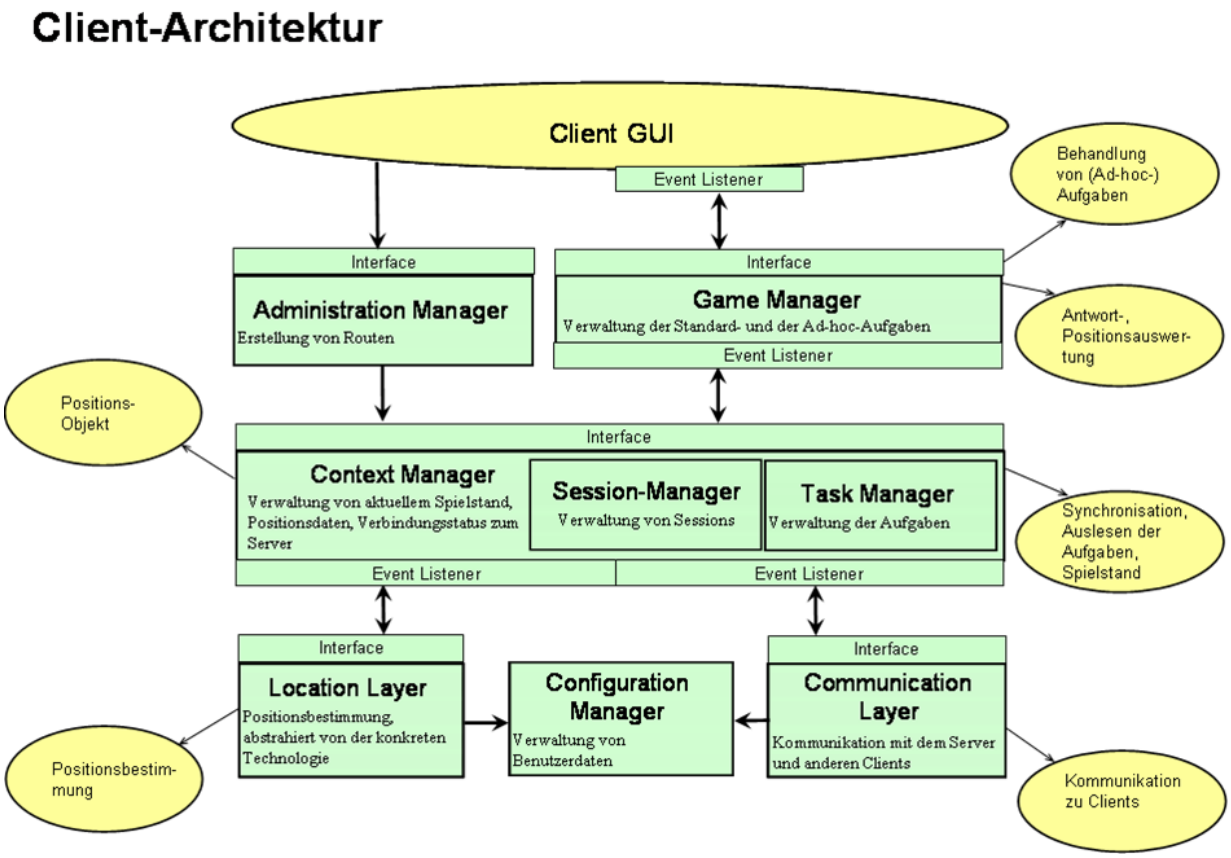



Abbildung 3.1: Pervasive Gaming Framework (Client Architektur)

**Task Manager** (siehe Kapitel 4.3)

**Context Manager** Der *Context Manager* ist die zentrale Komponente des Frameworks. Er nutzt den *Session Manager* und den *Task Manager* für eine Verwaltung von Sessions und Aufgaben. Zusätzlich greift er über Interfaces auf die Funktionalität des *Location Layers* und des *Communication Layers* zu. Der *Context Manager* hat außerdem die Aufgabe, über Event Listener Informationen vom *Location Layer* und *Communication Layer* entgegenzunehmen und diese an den *Game Manager* weiterzuleiten.

**Communication Layer** Der *Communication Layer* dient der Kommunikation mit dem Server über Web Services, sowie der Kommunikation mit anderen Clients. Außerdem kapselt er die konkrete Kommunikations-Technologie.

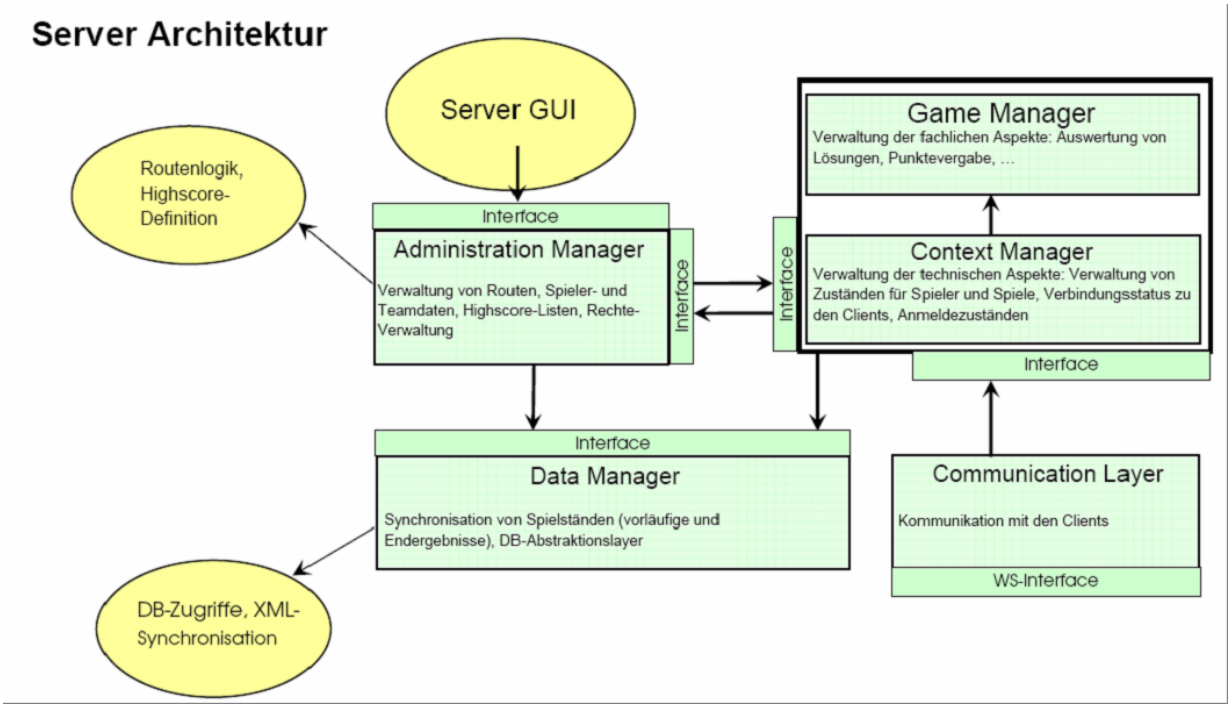



Abbildung 3.2: Pervasive Gaming Framework (Server Architektur)

## 3.2 Szenario

In Kapitel 1.2 wurde bereits die grobe Spielidee beschrieben. Für ein Konzept, welches als Grundlage für eine spätere Realisierung dienen soll, ist es jedoch notwendig, ein detaillierteres Szenario und zugehörige Ablaufdiagramme zu entwerfen.

Das Szenario sieht vor, dass Spieler entweder alleine oder in Teams am Spiel teilnehmen können. Die Spieler nutzen zum spielen PDAs. Sie können mittels der auf den PDAs installierten Spielesoftware Routen für eine Schnitzeljagd herunterladen. Diese Spiele müssen vorher für die Spieler, bzw. Teams auf dem Server eingerichtet werden. Ziel des Spiels ist es, festgelegte Orte in der realen Welt zu finden. Wenn die Spieler einen Ort erreicht haben, bekommen sie Aufgaben, bzw. Fragen zum jeweiligen Ort gestellt. Die Aufgaben sind erst verfügbar, wenn der Ort erreicht wurde. Wenn die Aufgaben gelöst wurden, wird der nächste Zielort freigeschaltet. Es ist dabei egal, ob die Aufgaben richtig oder falsch gelöst werden. Wenn alle Zielorte erreicht und alle Aufgaben gelöst wurden, kann das Endergebnis zum Server übertragen werden.

Während des Spiels können theoretisch Ad-Hoc Verbindungen zu anderen Clients aufgebaut werden um beispielsweise den Spielstand zu synchronisieren.

Die zugehörigen Ablaufdiagramme sind unter Anhang A zu finden. Sie zeigen einerseits Szenarien der Client-Server und Client-Client Interaktion, sowie den Client internen Ablauf.

## 3.3 XML Schema

Das *Pervasive Gaming Framework* und die zu entwickelnden Spiel-Prototypen sind aufgrund der *Schnitzeljagd* Metapher auf Routen-Informationen angewiesen. Die Routen, bzw. Routen-Informationen sollen dabei mindestens eine Sequenz von Routenpunkten und zugehörigen Aufgaben (Tasks) abbilden. Für die Erstellung und Übertragung von Routen, bzw. Routen-Informationen, ist es notwendig eine *Routen Beschreibungssprache*, bzw. eine angemessene Datenstruktur zu entwickeln.

Potentiell sind dafür verschiedene strukturierte Datenformate denkbar, wie z.B. CSV, HTML, XML oder proprietäre Formate. Für das *Pervasive Gaming Framework* Projekt wurde XML als Datenformat gewählt. XML bietet sich an, da die generelle Struktur durch eine XSD (XML Schema Description) beschrieben werden kann und es etablierte Standards und Werkzeuge für das Zusammenspiel von XSD, XML und Programmcode (u.a. Java, C#) gibt.

Abbildung 3.3 zeigt das konzeptionelle Schema für das *Pervasive Gaming Framework*. Für konkrete Spiele kann dieses Schema bei Bedarf ergänzt oder ersetzt werden. Wie in Kapitel 4.1 beschrieben, war es im Laufe des Projekts notwendig, das konzeptionelle Schema aufgrund der Unzulänglichkeit verfügbarer Werkzeuge anzupassen. Die angepasste Version unterscheidet sich lediglich syntaktisch vom konzeptionellen Schema.

An dieser Stelle ist anzumerken, dass es bereits Ansätze für XML basierte *Routen Beschreibungssprachen* gibt. Der am weitesten verbreitete Vertreter ist *GPX*[1] (GPS Exchange Format). Da *GPX* erweiterbar ist, wäre eine Verwendung für die Zwecke des *Pervasive Gaming Framework* theoretisch möglich gewesen. Der Aufwand wäre jedoch höher gewesen, als die Entwicklung einer eigenen *Routen Beschreibungssprache*. Zumal *GPX* nicht problemlos mit den unter 4.1 beschriebenen Werkzeugen zu verwenden ist.

[1] http://www.topografix.com/gpx.asp

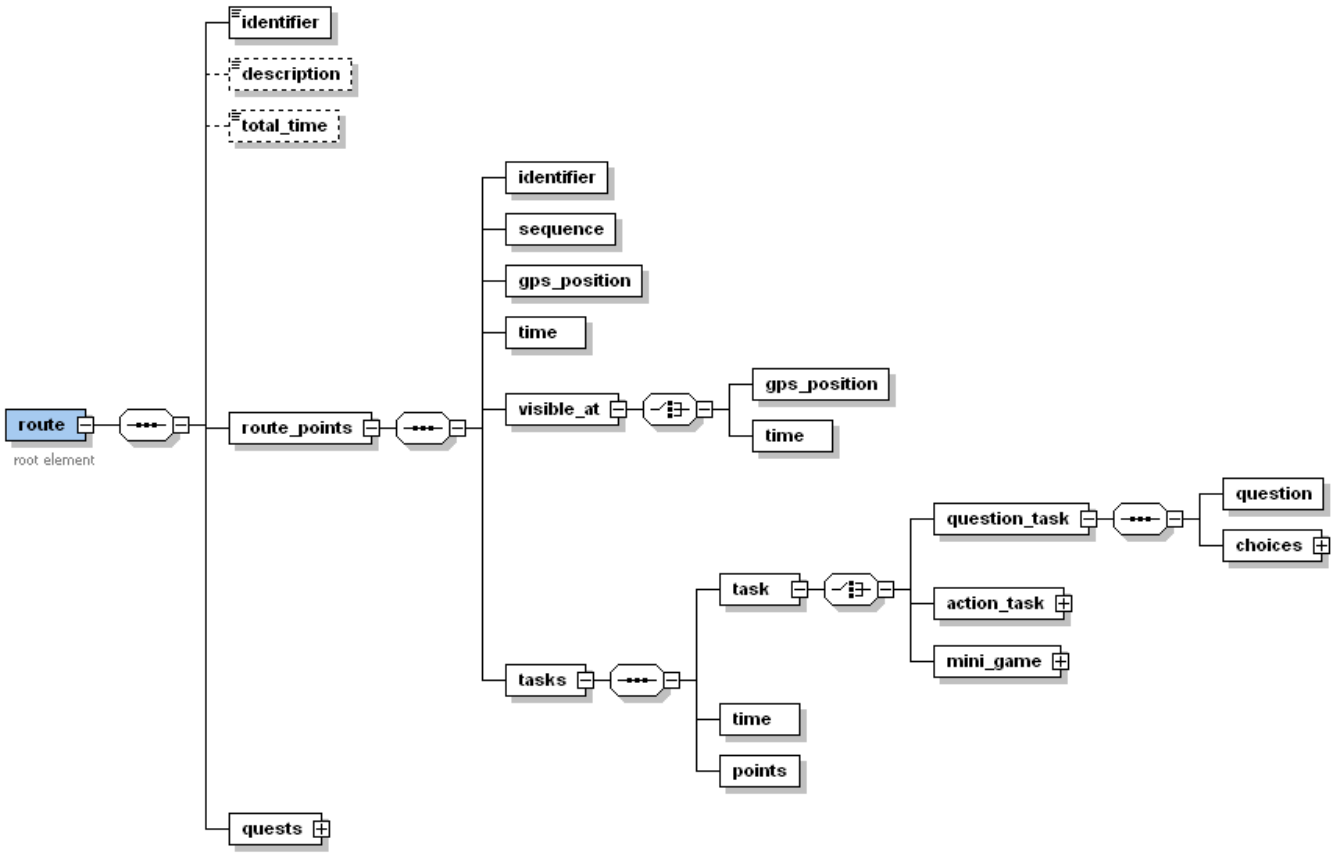



Abbildung 3.3: Route Schema

# Kapitel 4

# Realisierung

In diesem Kapitel werden die Teilprojekte beschrieben, die vom Author dieses Berichts entworfen und realisiert wurden. Für die Beschreibungen der restlichen Teilprojekte sei an dieser Stelle auf die Berichte der anderen Projektteilnehmer verwiesen. (in Kürze zu finden auf [UbiComp])

## 4.1 XML Parser

In Kapitel 3.3 wurde beschrieben, dass die *Routen Beschreibungssprache* mit Hilfe eines XML Schemas realisiert wurde. Für das Einlesen und Verarbeiten von konkreten Routen, mussten im Framework entsprechende Klassen und Methoden vorgesehen werden. Es bietet sich dabei an, diese Klassen und Methoden nicht per Hand zu erstellen, sondern geeignete Werkzeuge zu deren Generierung zu verwenden. Aus dem Java Bereich ist für diesen Zweck *JAXB* (Java Architecture for XML Binding) ein Begriff. Mithilfe von *JAXB* lassen sich auf Basis einer XSD, Klassen und Methoden erzeugen, mit deren Hilfe es sehr einfach und elegant möglich ist, XML Dateien, die nach Vorgabe der XSD erstellt wurden, zu parsen und Objekte aus ihnen zu erzeugen.

Für .Net, bzw. C# existieren qualitativ vergleichbare Produkte nur im kommerziellen Umfeld. Da diese Produkte den finanziellen Rahmen des Projekts gesprengt hätten, kam für das *Pervasive Gaming Framework* Projekt nur der dem *Microsoft Visual Studio 2005* beiliegende Parser und ein Open Source Produkt namens *dingo*[1] infrage. Der *Microsoft Visual Studio 2005* Parser *xsd* lieferte keine sinnvoll verwendbaren Ergebnisse. Mit Hilfe von *dingo* waren nach einigem Aufwand brauchbare Ergebnisse zu erzielen. Wie in Kapitel 3.3 beschreiben, musste das XSD Schema zu diesem Zweck syntaktisch angepasst werden.

Die von *dingo* generierten Klassen sind nicht Bestandteil des Frameworks, sondern dienen

[1] http://dingo.sourceforge.net/

hauptsächlich dem eleganten Einlesen von von Routen-Informationen im XML Format. Zusätzlich muss ein Mapping auf die entsprechenden Klassen des Frameworks erfolgen (siehe 4.2). Ein weiterer wichtiger Aspekt der generierten Klassen ist, dass sie serialisierbar sind. Somit lassen sie sich bei der über Web Services realisierten Kommunikation zwischen Client und Server verwenden.

## 4.2 XML Mapper

Das Framework beinhaltet Interfaces und Klassen, die alle generellen Komponenten von Routen abbilden. Diese Interfaces und Klassen korrespondieren mit den Komponenten der unter 3.3 beschriebenen *Routen Beschreibungssprache*. Wie das XML Schema lassen sich die konkreten Klassen des Frameworks erweitern. Damit die Basisfunktionalität des Frameworks stets erhalten bleibt, ist es notwendig, dass die Interfaces dabei nicht verändert werden. Wie in Kapitel 4.1 beschrieben, werden mit Hilfe von *dingo* Klassen erzeugt, die zum Einlesen von Routen-Informationen im XML Format dienen. Diese Klassen implementieren aus technischen Gründen nicht die vom Framework vorgegebenen Interfaces. Es muss also ein Mapping auf die entsprechenden Klassen des Frameworks erfolgen.

Die Klasse *XMLMapper* übernimmt diese Aufgabe. Die im Framework enthaltene Implementierung dieser Klasse, ermöglicht ein Mapping der dem Standard XML Schema entsprechenden *dingo* Klassen auf die Standard Routen Klassen des Frameworks. Für konkrete Anwendungen oder Spiele, die das Standard XML Schema oder die Standard Routen Klassen erweitern, muss die *XMLMapper* Klasse ebenfalls erweitert, bzw. ersetzt werden.

Außerdem bietet die *XMLMapper* Klasse die Möglichkeit, ein *Rückwärts-Mapping* der Framework Klassen auf die generierten Klassen vorzunehmen. Diese Funktionalität wird unter Anderem benötigt, um Ergebnisse in ein serialisierbares Format zu bringen, damit sie per Web Service zum Server übertragen werden können.

## 4.3 Task Manager

Der *Task Manager* (vgl. 3.1) bildet einen Großteil des eigentlichen Spielflusses ab. Er bietet folgende Funktionalitäten:

- mithilfe des *XML Mappers* Routen laden, speichern oder serialisieren
- den aktuellen, nächsten oder letzten Routen-Punkt zurückliefern
- die aktuelle, nächste oder letzte Aufgabe zurückliefern
- Ergebnisse und Zwischenergebnisse verwalten und zurückliefern

Der Frameworkanteil des *Task Managers* steckt dabei in der Klasse *AbstractTaskManager*. Konkrete Anwendungen, bzw. Spiele können von dieser Klasse erben und die Funktionalität erweitern.

# Kapitel 5

# Fazit

**Ergebnis** Bis auf einige Details wurden die Ziele des Projekts erreicht und es konnte zum Abschluss ein auf dem Framework basierendes Spiel präsentiert werden. Somit ist das Projekt als Erfolg anzusehen.

**Kritik** Der Kommunikationsaufwand innerhalb des Projektteams war über einen langen Zeitraum des Projekts höher als erwartet. Die eigentliche Implementierungsphase verkürzte sich daher, was zu einem hohen Zeitdruck zum Ende des Projekts führte.

Im Projekt stand hauptsächlich das Gesamtergebnis im Vordergrund. Dies hatte den Vorteil, dass zum Ende des Projekts ein vorzeigbares Produkt entstand. Allerdings konnten unter diesen Umständen nicht alle individuellen Interessen der Projektteilnehmer gefördert werden.

Insgesamt stellte sich der Zeitraum von einem Semester als sehr gering für ein Projekt dieser Größenordnung heraus. Ein Zeitraum von zwei Semestern wäre rückblickend angemessener gewesen und hätte mehr Raum für die Umsetzung individueller Ideen und Konzepte gelassen.

**Ausblick** Das Ergebnis des Projekts bietet eine gute Ausgangssituation für anschließende Arbeiten. Es wurde anhand von Prototypen gezeigt, dass sich auf Basis der aktuellen Version des Frameworks, lauffähige pervasive Spiele aufbauen lassen. Es ist daher naheliegend, aufwändigere Spiele auf dem Framework aufzubauen, die über einen Prototyp-Status hinausgehen.

Bisher unterstützt das Framework lediglich GPS Positionierung. Interessant wäre daher eine Erweiterung der unterstützen Positionierungs-Technologien um beispielsweise Indoor-Positionierungs-Technologien. Ein weiterer Aspekt, der im Framework zwar konzeptionell vorgesehen ist, jedoch noch nicht implementiert wurde, ist die Kommunikation von Cient zu Client via Bluetooth.

# Literaturverzeichnis


[Gamma, 1996] Erich Gamma et al.: **Entwurfsmuster - Elemente wiederverwertbarer objektorientierter Software.**, Addison-Wesley (1996)

[Gamma, 1996] Erich Gamma et al.: **Entwurfsmuster - Elemente wiederverwertbarer objektorientierter Software.**, Addison-Wesley (1996)

[Linner et al., 2005] David Linner et al.: **Context-aware Multimedia Provisioning for Pervasive Games**, Fraunhofer Institute for Open Communication Systems (2005)

[UbiComp] UbiComp @ Informatik.HAW Hamburg. Im Internet zu finden unter <http://users.informatik.haw-hamburg.de/ ubicomp/projects.html> (Februar 2007)

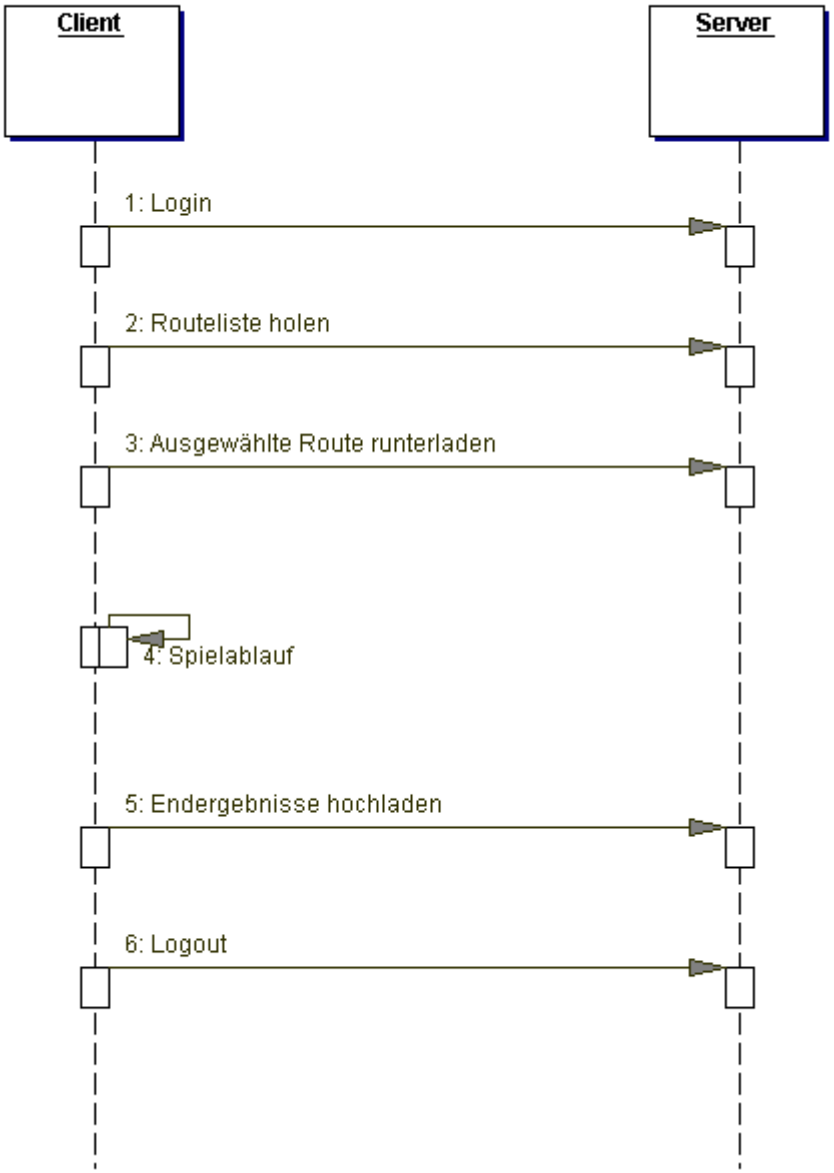
[Waern, 2004] Annika Waern: **Working with Pervasive Games**, 2004. Im Internet zu finden unter <http://pergame.blogspot.com> (Februar 2007)

[OKS, 2004] Offene Kommunikationssysteme OKS, TU Berlin: **Cron Game Concept**, 2006. Im Internet zu finden unter <http://www.oks.cs.tu-berlin.de/forschung/pervasive-gaming/> (Februar 2007)

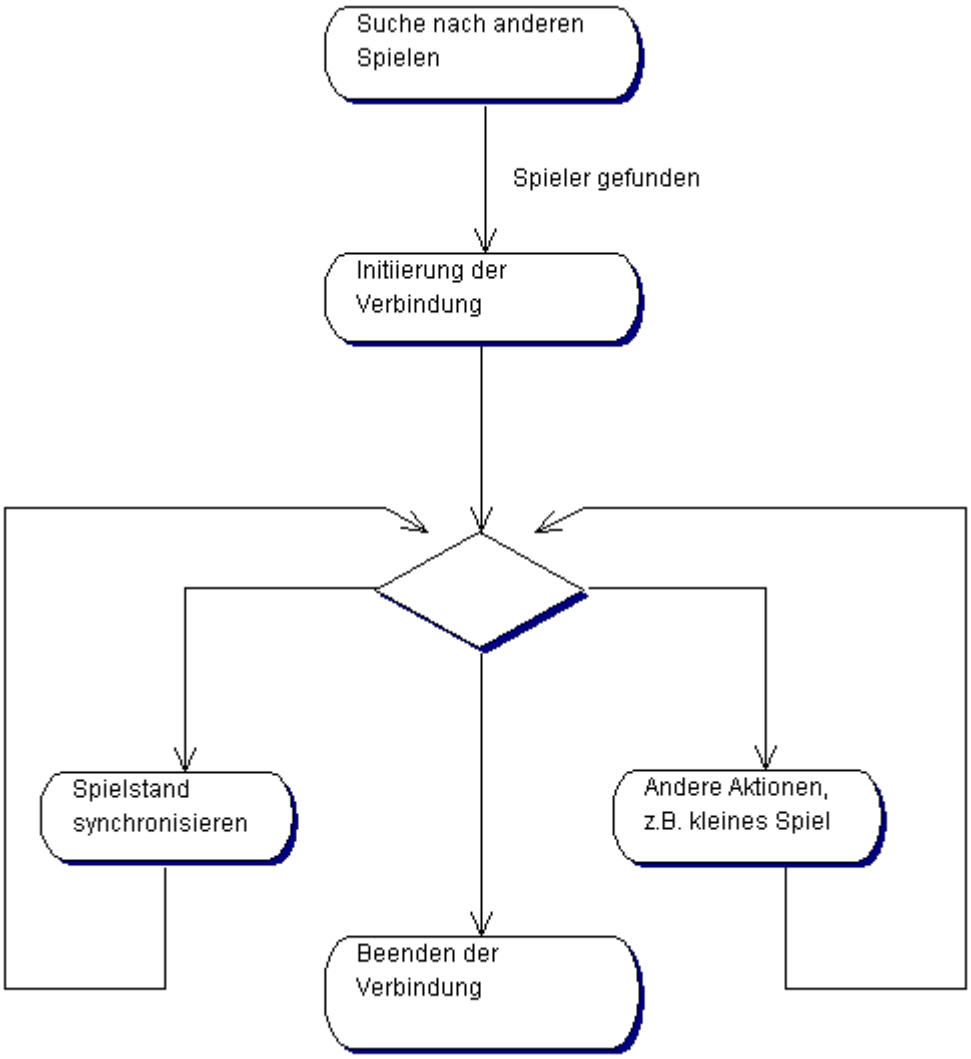
# Anhang A

# Ablaufdiagramme

## Client-Server



## Client-Client



## Client intern