Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Projektbericht

Alexander Mas

Pervasive Spine

Fakultät Technik und Informatik
Studiendepartment Informatik

Hamburg, den 28.Februar 2008

Inhaltsverzeichnis

1. Einleitung und Motivation

Das Projekt *Pervasive Spine* wurde im Rahmen des Masterstudiengangs im Fachbereich Informatik an der Hochschule für Angewandte Wissenschaften in Hamburg[1] durchgeführt. Grundlage des Themas ist neben den Interessensgebieten der einzelnen teilnehmenden Masterstudenten der Entwicklungsstand und die aktuelle Forschung auf dem Gebiet des *Pervasive*[2] *Computing* und *Pervasive Gaming*. Das Projektteam bestand aus 8 Studenten und 2 betreuenden Professoren. Der Zeitrahmen umfasste ein Semester, in dem Treffen von 8Std./Woche angesetzt waren.

Die Leistungsfähigkeit der heutigen mobilen Endgeräte steigt zunehmend und in immer kürzeren Intervallen an und ist in puncto Rechenkapazität und Speicherplatz schon soweit, dass die Hürden für die Entwicklung von komplexeren Anwendungen für mobile Geräte wesentlich geschrumpft sind. So besitzen sie u.a. die verschiedenste Möglichkeiten zur Kommunikation/Interaktion mit anderen Geräten oder der Umwelt. Hinzu kommen die fallenden Preise der Dienstanbieter und der Geräte selbst, was die Nachfrage der Endbenutzer nach diesen und immer neuen Anwendungen dafür, nur noch mehr steigen lässt und das unabhängig davon in welcher Alters- oder Anwendungsschicht. Diese Verbreitung und das Leistungsvermögen schaffen Möglichkeiten, neue und weitreichende Anwendungs- bzw. Spielideen für die kompakten Geräte umzusetzen, die ihre vielfältigen Fähigkeiten nutzen.

Damit neu entwickelte Spielkonzepte schnell und einfach umgesetzt und diese dann genauso schnell validiert werden können, benötigt man ein Framework[3], welches die Entwickler in ihrer Tätigkeit unterstützt. Ziel des Projektes war die Entwicklung eines solchen Frameworks für pervasive, mobile Spiele, das für die Erstellung solcher Spiele grundlegende Funktionen zur Verfügung stellt. Um die Funktionsfähigkeit des Frameworks zu untermauern sollten auch zwei Spielkonzepte als Proof-of-Concept Prototypen entwickelt und implementiert werden.

Jedoch beinhaltete dieses Ziel nicht ein konkretes Produkt zu entwickeln, sondern den Studenten mehr Erfahrungen bezüglich der Entwicklung von Anwendungen für mobile Endgeräte zu ermöglichen. Zusätzlich konnte das Projekt für manche Studenten als Hilfestellung oder Orientierung zur Themenfindung für die bevorstehende Masterarbeit dienen.

In den folgenden Kapiteln soll nun weiter auf das Projekt und seine Umsetzung eingegangen werden.
Im Kapitel 2 werden technische Grundlagen erläutert. Kapitel 3 beschreibt den Aufbau, Ablauf und konkretes zur Vision des Projektes. In Kapitel 4 wird auf die Realisierung der Aufgaben des Autors eingegangen. Zum Schluss steht in Kapitel 5 ein Fazit über das Projekt.

[1] Homepage HAW Hamburg Informatik
[2] Engl. für durchdringen, überall vorhanden.
[3] Engl. für Gerüst, Rahmen, Rahmenwerk.

2. Grundlagen

Dieses Kapitel geht in den anschliessenden Punkten grob auf einige technische Grundlagen ein, die im Projekt und im Rahmen der Aufgaben des Autors Thema waren.

2.1. Framework

Nach einer Definition von Erich Gamma [1] stellt ein Framework folgendes dar:

> *„A framework is a set of classes that makes up a reusable design for a specific class of software. A frameworks provides architectural guidance by partioning the design into abstract classes and defining their responsibilities and collaborations. A developer customizes the framework to a particular application by subclassing and composing instances of framework classes.“*

Es beinhaltet kooperierende Klassen, die ein wiederverwendbares Design darstellen, welches für eine spezielle Art von Software gilt. Um eine Anwendung zu erstellen passen die Entwickler das Framework auf die Weise an, dass sie Unterklassen von abstrakten Klassen des Frameworks ableiten. So bestimmt es die Architektur der Anwendung, d.h. den allgemeinen Aufbau, die Zusammenarbeit der Klassen und Objekte und den Kontrollfluss. Es nimmt dem Entwickler dahingehend Arbeit ab, dass dieser sich nicht um die grundlegenden und standardmäßigen Funktionen kümmern muss, sondern sich auf die speziellen Eigenschaften seiner Anwendung konzentrieren kann.

Eine ähnliche Beschreibung von [5] besagt, dass ein Framework aus statischen Teilen besteht, die die allgemeine Architektur der Anwendung angeben und bei jeder Wiederverwendung des Frameworks unverändert bleiben. Diese werden *frozen spots* genannt. Daneben gibt es *hot spots*, die von den Entwicklern bei jeder neuen Instanziierung an die neue Anwendung angepasst werden und ihre eigene Funktionalität implementieren.

Der entscheidende Unterschied zu einer reinen Klassenbibliothek besteht darin, dass diese ausschließlich Funktionalität kapselt. Die Anwendung ruft lediglich ihre Funktionen an der benötigten Stelle auf und benutzt diese, ohne wissen zu müssen, wie diese implementiert wird. Die Verantwortung an Architektur und Kontrollfluss liegt daher alleine bei der Anwendung, also beim Entwickler.
Ein Framework hingegen, wie oben beschrieben, gibt diese Eigenschaften vor und durch Ableiten der vorgegebenen Klassen stellt es fest, dass die Anwendungsfunktionalität an der richtigen Stelle vom Framework aufgerufen wird. Genannt wird dieses Prinzip *Inversion of Control*.

2.2. Pervasive Gaming\Computing

Pervasive Gaming ist ein Teilgebiet des *Pervasive Computing*. Beide Ausdrücke werden auch gleichgesetzt mit *Ubiquitous Gaming* bzw. *Ubiquitous Computing*. Den Begriffen *ubiquitous* und *pervasive* misst man die Bedeutung „überall vorhanden“ und „durchdringend“ bei.
Dieses Konzept beschreibt die nächste Stufe in der Entwicklung der Informationstechnologie. Vom einzelnen Desktop-PC hin zu der Allgegenwärtigkeit von vernetzbaren, intelligenten Gegenständen im Alltag. Die Interaktion mit diesen geschieht nicht wie gewohnt von einem spezifischen Terminal bzw. Zugang aus, wie ein Monitor mit Tastatur auf dem Schreibtisch. Diese intelligenten Geräte rücken in den Hintergrund und gliedern sich in den täglichen Ablauf des Nutzers ein, der mit ihnen über mobile Steuerungsobjekte, wie z.B. Handys, oder vorzugsweise Sprachsteuerung kommuniziert. Geräte agieren intelligent und sprechen auch auf Zustände und Änderungen in der Umwelt an und unterstützen automatisch den Nutzer in seinen Interessen oder Vorhaben. [10; 11]

Pervasive Gaming ordnet sich dahingehend ein, dass es den Aspekt der Verschmelzung von virtuellem Raum mit realem Raum mit einbringt. In der Welt des Spiels wird Bezug genommen auf die realen Gegebenheiten, in der sich der Spieler befindet und auch darauf reagiert. Die reale Umgebung bestimmt somit den virtuellen Spielablauf mit oder ist sogar Grundlage von diesem. In [6] wird dieses Verhalten von Pervasive Gaming beschrieben.

2.3. XML

Die Abkürzung XML [2; 12] steht für *Extensible Markup Language*[4] und ist eine Auszeichnungssprache für Textdokumente. In diesen Dokumenten wird ihre Syntax genutzt, um Daten hierarchisch und semantisch zu beschreiben. Dafür werden einfache und verständliche Marken, sogenannte *Tags*, verwendet, in welche die Daten eingebettet sind. In Listing 1 ist ein vereinfachtes Beispiel für eine XML-Datei zu sehen.
Ziel von XML ist es, Daten zu beschreiben und sie zu speichern bzw. zu transportieren. Im Gegensatz zur verwandten Sprache HTML (Hypertext Markup Language) [8], die ähnlich wie XML Daten beschreibt, deren Ziel jedoch die Art der Darstellung dieser im Webbrowser ist. Zudem ist XML eine Meta-Sprache. Das bedeutet, es gibt keine vorgegebenen Tags wie bei HTML. Der Entwickler ist frei in der Definition der Tags, wie er sie für seine Situation benötigt. Jedoch gibt es strenge Regeln wie diese Tags anzuordnen sind, wann welche auftreten dürfen, welche Namen gültig sind und wie Attribute an solche Tags angefügt werden können. Genügt ein Dokument dieser Grammatik, heißt es *wohlgeformt*. Das unterstützt die Maschinenlesbarkeit und erlaubt die Entwicklung von Parsern[5], die jedes XML-Dokument lesen und

[4] Engl. für erweiterbare Auszeichnungssprache
[5] Computerprogramm zur Analysierung von z.B. Textdateien

```
<?xml version="1.0" encoding="UTF-8" ?>
<ausarbeitung>
  <titel>Projektbericht</titel>
  <kapitel>
    <titel>Überschrift</titel>
    <text>Text</text>
    <unterkapitel>
      <titel>Unterüberschrift</titel>
      <text>Text</text>
    </unterkapitel>
  </kapitel>
</ausarbeitung>
```

Listing 1: Beispiel einer XML-Datei

verstehen können.
XML ist von der World Wide Web Consortium (W3C) standardisiert und ist frei zur Verwendung verfügbar. Es ist weit verbreitet und wurde ursprünglich zur Verteilung von Daten zwischen unterschiedlichen Informationssystemen, besonders über das Internet, entwickelt. Wegen der simplen Struktur und der einfachen Maschinenlesbarkeit, wird es heutzutage für weit mehr verwendet, wie z.B. zur Konfiguration von Anwendungen.

3. Projekt

In diesem Kapitel soll das Projekt genauer vorgestellt werden. Dabei wird zuerst auf den allgemeinen Aufbau eingegangen und im Weiteren dann auf die Inhalte und deren Umsetzung.

3.1. Ablauf

Schon vor dem Semester haben sich Studenten und Professor getroffen um das Thema im Allgemeinen zu besprechen und eine Idee auszuarbeiten, in welche Richtung das Ziel des Projektes gehen könnte. Nachdem die Zieldefinition feststand ging es am Anfang des Semesters daran, sich hinsichtlich der Aufgabe zu orientieren und eine Struktur für das Projekt zu erarbeiten. Es wurden allgemeine Fragen geklärt wie u.a. welche Programmierplattform und Programmiersprache für die Umsetzung gewählt werden soll oder auch welche Hardware, also welche mobilen Geräte, in Betracht kommen. Weiterhin wurden organisatorische Punkte besprochen, wie z.B. die Erstellung eines Projektplans, in dem der zeitliche Ablauf festgelegt wurde. Es wurden Termine für Teammeetings festgelegt und Meilensteine für die Entwicklung definiert.
Zu Beginn des Projekts gab es noch einen Termin, an dem die Studenten des Vorsemesters zu Besuch kamen, um über ihre Erfahrungen zu berichten und Fragen des Projektteams zu beantworten.
Der nächste Schritt war der Start der Entwicklungsphase, in dem die Arbeit am Basis-Framework startete und die Spezifikationen für die Beispielanwendungen festgelegt wurden. Fortschreitend wurden dann die Beispielanwendungen implementiert und eine iterative Weiterentwicklung des Frameworks vollzogen. Schlussendlich gab es eine Inbetriebnahme des Frameworks mit den zwei prototypischen Spielen und den Projektabschluss, der mit einer Präsentation endete.

3.2. Aufgabenverteilung

Innerhalb des Teams, welches insgesamt aus 8 Studenten bestand, wurden die Aufgaben je nach persönlichem Interesse aufgeteilt. Somit galt jeder Einzelne für seinen Bereich als Ansprechpartner und Verantwortlicher. Im Zuge des Projektes ergaben sich allerdings bei einzelnen Teammitgliedern freie Ressourcen, mit denen sie dann in anderen Bereichen mitarbeiten konnten.

Hier soll nur eine Grobe Übersicht über die einzelnen Themengebiete gegeben werden:
Leif Hartmann und Jan Schönherr waren zuständig für eine Spieleanwendung mit dem

Namen „PS[6] Orientierungseinheit“. Ralf Kruse und Andreas Herglotz waren für die zweite Spieleanwendung namens „PS Community Manager“ zuständig . Sven Vollmer beschäftigte sich mit der Bluetoothkommunikation. Markus Dreyer realisierte die Peer-to-Peer Kommunikationsschicht und war an weiteren Teilen des Frameworks beteiligt. Der Bereich von Jan-Peter Tutzschke war das Framework und er unterstützte noch in einigen anderen Bereichen. Die Ausarbeitungen dieser Teammitglieder sind zu Finden auf der Seite Ubicomp [9]. Die Aufgaben des Autors erstreckten sich in der Mitentwicklung des Frameworks, insbesondere in den Bereichen XMLParser, Kameraservice mit Barcodeerkennung und Audioservice.

3.3. Vision

Die Vision dieses Projektes war die Entwicklung eines Frameworks für pervasive Anwendungen, vorwiegend für den Outdoor-Bereich. Im Fokus standen hierbei Spiele für mobile Geräte. Dabei wurde sich an neueren Handymodellen und PDA's orientiert, die in erster Linie genügend Rechenleistung und eine weitgehende Unterstützung der neuen Technologien, wie diverse Kommunikationskomponenten (z.B. Bluetooth, GPS, UMTS o.Ä.), besitzen.
Das Ziel des Frameworks war, eine umfassende Grundstruktur und Abstraktionsebene für die Entwickler zu bieten. Das heisst, es sollte zum einen Teil aus Komponenten aufgebaut sein, die jeweils immer wiederverwendet werden können und aus einem Teil, der für jede Anwendung angepasst bzw. neu erstellt wird. Weiterhin sollte es die Grundfunktionen der Geräte so abstrahieren, dass eine Nutzung dieser einfach durchzuführen ist. Bestenfalls mit einer Unterstützung für eine große Vielfalt von Gerätemodellen und -marken. Somit würde ein flexibles Framework vorliegen, welches unterschiedlichste Geräte und Technologien unterstützt und die Möglichkeit bietet, eine Variation an Spielkonzepten umzusetzen.
Auf diese Weise ist es dem Entwickler möglich, sein Hauptaugenmerk uneingeschränkt auf die Konstruktion der Anwendung zu legen. Das bietet den Vorteil, dass diese sich nicht um grundlegende Funktionen kümmern und über Implementationen Bescheid wissen müssen, sondern dass sie diese Funktionen einfach über das Framework benutzen.

Zudem beinhaltete die Vision noch die Entwicklung zweier, auf dem erstellten Framework aufbauenden, Beispielanwendungen. Dies sollten zwei Spiele sein, die sich sowohl vom Inhalt, als auch von der Verwendung unterschiedlichster Funktionen\Technologien, nahe an der Realität ausrichteten. Im nächsten Kapitel sollen diese näher beschrieben werden.

[6] PS: Pervasive Spine

3.4. Szenarien

Mit Bezug auf die Vision vom Erstellen eines Frameworks für pervasive, mobile Spiele aus dem vorherigen Abschnitt (3.3), wurden in dem Projekt zwei unterschiedliche Szenarien erarbeitet, die das Framework in seiner Funktion validieren sollen. Als erstes wird hier das Spiel „PS Community Manager“ und danach das Spiel „PS Orientierungseinheit“ vorgestellt.

3.4.1. PS Community Manager

Das Spiel PS Community Manager beschäftigt sich mit dem Gedanken der Community und der Kommunikation. Es verfolgt die Idee, dass Personen sich im normalen Alltag bewegen und die Anwendung erkennt, ob sich eine bekannte Person oder eine Person mit den gleichen Interessen in der Nähe befindet. Der Nutzer kann entweder selbst nach solchen suchen, oder er erhält automatisch eine Benachrichtigung. Das kann auch in einem größeren Umkreis stattfinden, wie z.B. in einer ganzen Stadt. Mit diesen Personen kann man dann über den Community Manager in Kontakt treten und sich gegenseitig Nachrichten zukommen lassen.
Um Eigenschaften und Interessen der Teilnehmer festzuhalten, sind diese in einem Profil auf einem Server gespeichert, welches jederzeit bearbeitet werden kann. Bekannte oder mögliche Personen mit gleichen Interessen werden auf einer Karte angezeigt. Darauf können auch andere Objekte markiert sein, wie interessante Orte o.Ä. Dieses Konzept gibt dem Nutzer die Gelegenheit, neue Leute mit gleichen Interessen kennen zu lernen oder sich mit bekannten Personen spontan zu verabreden, von deren räumliche Nähe man sonst nicht gewusst hätte.

3.4.2. PS Orientierungseinheit

PS Orientierungseinheit ist ein Spiel, das sich an der Art des bekannten Spielprinzips „Schnitzeljagd“ orientiert. Dabei meldet der Spieler sich mit seinem mobilen Gerät an einem Server an und kann dort ein gewünschtes Spiel auswählen. Beispielsweise kann der Spieler ein Spiel wählen, dass ihn entlang einer Route leitet, bei der er immer wieder an bestimmten Punkten anhand von Hinweisen den nächsten Routenpunkt erhalten kann. Das führt den Spieler somit zu einem speziellen Zielort oder entlang eines besonderen Weges. Zur Visualisierung des aktuellen Standortes wird dieser auf einer Umgebungskarte auf dem Gerät angezeigt zusätzlich zu der Richtung, in der der nächste Routenpunkt liegt.
Um dem Spiel einen Wettbewerbscharakter zu geben, werden die Ergebnisse auf den Server übertragen und dort in einer Art Highscore-Liste gespeichert, die von jedem Spieler auf

dem Geräte eingesehen werden kann. Eine weitere Eigenschaft des Spieles ist, das es unterbrechbar ist, sodass der Spieler es jederzeit später an diesem Punkt wieder aufnehmen kann.

3.5. Anforderungen

Auf Basis der Szenarien im Abschnitt 3.4, lassen sich für die einzelnen Spiele die einzelnen fachlichen und technischen Anforderungen definieren. Zuerst werden hier die fachlichen und dann die technischen Anforderungen aufgezeigt, welche an die Szenarien und an das Framework gestellt werden.

3.5.1. Fachliche Anforderungen

PS Community Manager:

- Eine Benutzerverwaltung und eine Spieleverwaltung, die zuständig sind für Anmeldung beim Spiel, Spielstände usw.
- Eine Verwaltung und Speicherung von Objekten auf dem mobilen Geräte um z.B. Orte oder Personen eingeben, bearbeiten und löschen zu können.
- Möglichkeiten zur Kommunikation mit anderen Personen über eine Chat-Funktion oder durch versenden von Nachrichten.
- Das Spiel soll Objekte in Abhängigkeit vom Ort erkennen können, wie Personen und Orte, und diese auf einer Karte darstellen.

PS Orientierungseinheit:

- Eine Benutzerverwaltung und eine Spieleverwaltung, die zuständig sind für Anmeldung beim Spiel, dem Spiel beitreten, Highscore-Listen, Spielstände usw.
- Visualisierung des aktuellen Standorts und der Route bzw. Richtung zum nächsten Routenpunkt auf einer aktuellen Karte.
- Ein Mechanismus, der überprüft, ob der gesuchte Ort erreicht wurde.
- Das Spiel soll zu jedem Zeitpunkt unterbrechbar sein, und ab diesem Punkt auch wieder fortgesetzt werden können.

3.5.2. Technische Anforderungen

Im vorherigen Abschnitt wurden die fachlichen Anforderungen aufgezeigt. Es lässt sich erkennen, dass sich einige innerhalb der beiden Spielen den überschneiden. Aus der vereinigten Menge sollen hier jetzt die technischen Anforderungen entwickelt werden.

- Eine zentrale Verwaltung für Spielzustand und Objekte auf einem Server.
- Die Möglichkeit, Daten auf dem Server sowie auf den mobilen Geräten persistent zu halten.
- Kommunikationsmöglichkeiten zwischen den mobilen Geräten und dem Server sowie zwischen den mobilen Geräten untereinander (P2P).
- Die Fähigkeit zur Lokalisierung.
- Um möglichst viele verschiedene Arten von Spielen zu unterstützen, muss es eine einfach Art der Konfiguration geben.

4. Konzept

Das im Projekt entworfene Konzept des Frameworks wird in Kapitel 4.1 erläutert. In Kapitel 4.2 werden die einzelnen Aufgabengebiete des Autors beschrieben.

4.1. Framework

Abbildung 1 zeigt den Aufbau des Frameworks und die darum liegende Struktur. Zu sehen sind 3 Schichten.

In der ersten Schicht (violett gefärbte Kästen) sind die *konkreten Anwendungen* zu sehen. In diesen werden die von dem Framework vorgegebenen Schnittstellen verwendet und implementiert. Die Konfiguration dieser einzelnen Klassen wird in einer XML-Datei vorgenommen, die für dieses Projekt den Namen *spine.xml* erhielt. Der Anwendungsentwickler legt in ihr den Zusammenhang der Klassen und die Verantwortlichkeiten fest.

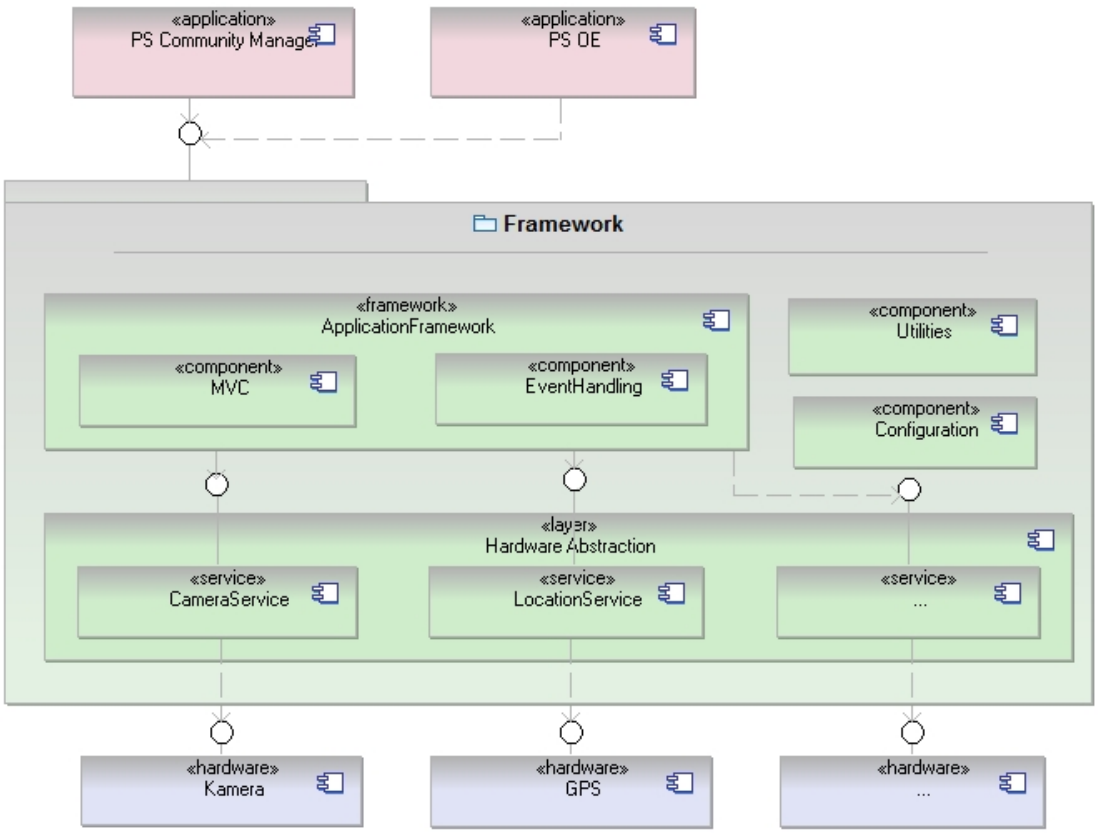

Abbildung 1: Übersicht des Frameworks

Die zweite Schicht stellt das *Framework* dar (grün gefärbte Kästen). Es setzt sich wiederum aus zwei Schichten und weiteren Komponenten zusammen.
Als Oberste ist die *Applikationsschicht* zu sehen. Darin enthalten ist die Komponente für die Trennung von Anwendungslogik und Darstellung. Die Umsetzung ist orientiert sich an dem MVC Pattern[7].
Die zweite Komponente ist verantwortlich für die Konfiguration und Handhabung der Events. Zur Konfiguration entnimmt sie ebenfalls Informationen aus der oben beschriebenen XML-Datei. Sie verwaltet die aktiven Events und führt bei deren Auftreten die festgelegte Funktionalität der Anwendung aus. Weiterhin besitzt sie eine Schnittstelle, um aus der Anwendung heraus nachträglich Events manuell registrieren zu können.
Unter der Applikationsschicht wird die *Abstraktionsschicht für die Hardware* dargestellt. Sie abstrahiert die vorhandene Hardware durch Schnittstellen, die so ihre Dienste anbieten. Diese sind so angelegt, dass sie nicht direkt die Technik darunter widerspiegeln, sondern ihre Funktionalität. Das heißt, für den Location-Service z.B. ist es nebensächlich, ob er die Position über GPS oder W-LAN Triangulation erhält.
Neben diesen zwei Schichten gibt es noch die Komponenten *Configuration* und *Utilities*.
Utilities beinhaltet zusätzliche Funktionalitäten. Dort findet man u.a. das Logging, welches sich an Log4J orientiert. Es ist möglich Logging-Informationen auf dem Desktop-Rechner zu sehen, aber auch auf dem mobilen Gerät. Sie enthalten ebenso die grundlegenden Klassen für den XML-Parser, der die XML-Datei zur Konfiguration der Anwendung analysiert.
Configuration ermöglicht es, Konfigurationsparameter in eine externe Properties-Datei auszulagern, sodass diese auf einfache Weise geändert werden können, ohne einen Eingriff in die Anwendung vornehmen zu müssen. Beispielsweise kann darüber das Logging auf dem mobilen Gerät aktiviert werden.

Die unterste Schicht stellt die konkrete Hardware dar (blau gefärbte Kästen), die vom mobilen Gerät zur Verfügung gestellt wird. Je nach Gerät können diese variieren.

Mit Abbildung 2 soll grob veranschaulicht werden, wie sich der Zusammenhang zwischen den konkreten Anwendungen und der Applikationsschicht verhält. Die Applikationsschicht hält in den einzelnen Komponenten Interfaces bereit, die in der Anwendung fest implementiert werden.
Auf der einen Seite ist die MVC Komponente zu sehen. Darin ist verdeutlicht, wie die Darstellung von der Anwendungslogik getrennt ist. Für die Anzeige auf dem mobilen Gerät sind die *Views* verantwortlich. Grundlegend implementiert man für jede neue Seite eine View in der Anwendung.
Die Anwendungslogik wird in den *Actions* implementiert. In der Konfiguration kann definiert werden, ob nachdem Ausführen einer Action eine weitere Action-Klasse oder eine View auf-

[7] Model View Controller Pattern nach [3].

gerufen wird.
In den *EventActions* wird festgelegt, welche Aktion ausgeführt werden soll, wenn ein gewisser Event aufgetreten ist. Das kann z.B. sein, wenn der Spieler einen bestimmten vordefinierten Ort erreicht hat und dies per Location-Service erkannt wird.

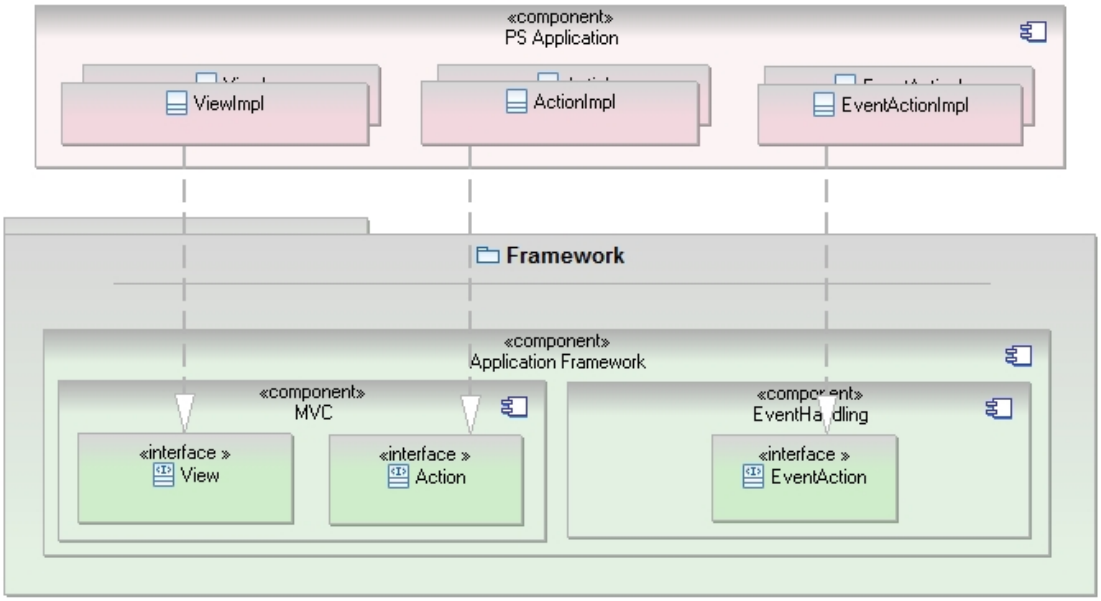

Abbildung 2: Übersicht Anwendung und Applikationsschicht

4.2. Realisierung

Hier werden kurz die einzelnen Aufgaben des Autors dargestellt. Für die Entwicklung hat sich das Projektteam auf die Programmiersprache Java geeinigt. Nach Überlegungen, welche Sprache denn am sinnvollsten wäre, ist man zur Entscheidung gekommen, dass Java aus Kompatibilitätssicht am geeignetsten ist und auch die heutigen Spielehersteller darauf setzen. Die Problematik bei Java ist, das die Java API's[8] von jedem Hersteller unterschiedlich implementiert und unterstützt werden. Es gibt jedoch schon Bemühungen, die die API's auf den verschiedenen Geräten mehr und mehr standardisieren (Stichwort Umbrella JSR[9]). Zum Ende des Projektes gab es sogar die Nachricht, dass wegen der steigenden Leistungsfähigkeit der neuen mobilen Geräte in weiterer Zukunft die J2ME[10] sogar ganz in die JSE[11] eingebunden werden soll.

[8] Engl. für Application Programming Interface, deutsch: Schnittstelle zur Anwendungsprogrammierung
[9] Java Specification Request
[10] Java Micro Edition. Entwickelt für „embedded-“ und mobile Geräte.
[11] Java Standard Edition

4.2.1. XML-Parser

Wie im vorherigen Abschnitt 4.1 bereits erwähnt, wird zur Konfiguration der Anwendung eine XML-Datei verwendet. Darin definiert der Entwickler den Zusammenhang und die Verantwortlichkeiten der einzelnen Klassen. Ein Beispiel für die XML-Datei findet man in Anhang A.1. Weiterführende Informationen zum Aufbau sind in der Ausarbeitung von Jan-Peter Tutzschke zu finden [9].
Der XML-Parser ist für das Auslesen der XML zuständig. Bei jedem Start der Anwendung durchläuft der Parser die Datei und liest aus ihr die Daten zeilenweise aus. Dabei interpretiert der Parser die Tags der Datei und wandelt diese an gegebener Stelle in Java-Klassen um. Mitgelieferte Attribute werden ebenso ausgelesen und an die Klassen übergeben. Zuvor müssen die Tags dem Parser als statische Variablen bekannt gemacht werden und die entsprechenden Klassen mit eventuellen Attributswerten zur Verfügung stehen.
Der Parser arbeitet nach dem Prinzip eines Zustandsautomaten. Ihm ist bekannt, welche Art von Tags eingelesen wurden und er weiss, welche oder welches Tag direkt darauf folgen darf. Stimmt das folgende Tag nicht damit überein, wird eine Fehlermeldung generiert und der Parser stoppt. Gleiches geschieht bei falschen bzw. fehlenden vorausgesetzten Attributen.
Somit entstehen alle nötigen Klassen, die dann zum Starten der Anwendung in den Speicher geladen werden. Nicht untersucht wurde, ob es zwecks Speicherverwaltung vorteilhafter wäre, die XML-Datei nur teilweise auszulesen und bei Bedarf den jeweiligen Teil nachzuladen, oder ob das ständige Öffnen und Parsen einer Datei den Vorteil wieder aufhebt.

4.2.2. Kamera

Um in den Anwendungen die Kamera nutzen und gegebenenfalls Bilder aufnehmen zu können, wurde ein Camera-Service implementiert. Ziel war es, diese Funktionen so einfach wie möglich nutzbar zu machen. Dazu wurde der konkrete Service (`CameraServiceImpl()`)mit einfachen Funktionen ausgestattet, welche die detaillierte Implementierung vor dem Entwickler verbirgt. Zur Anzeige des aktuellen Kamerabilds muss lediglich die Methode `showCamera(..)` aufgerufen und der Funktion ein Objekt mitgeliefert werden, auf dem das Bild angezeigt werden soll. Das Aufnehmen eines Bildes läuft genauso einfach ab, in dem man die Methode `getSnapshotAsByteArray()` oder `getSnapshotAsImage()` aufruft, je nachdem in welchem Format man die Bilddaten zurück geliefert bekommen möchte. Zum Beenden reicht der Aufruf von `discardPlayer()`.

4.2.3. Barcode

Als Weiteres wurde die Möglichkeit zur Erkennung von Barcodes implementiert. In diesem Projekt wurde die Erkennung von Semacodes[12] realisiert. Es ist ein 2D Barcode und steht samt den Bibliotheken frei zur Verfügung.

Das Dekodieren eines Semacode kann mit Verwendung des eigenen Camera-Service vollbracht werden. Wurde ein Bild eines Codes aufgenommen, ruft man in der Klasse `SemacodeServiceImpl` einfach `decode(..)` mit den Bildinformationen als Parameter auf. Die Methode kann die Informationen als Byte-Array oder als Image Objekt verarbeiten. Zurückgeliefert wird der dekodierte Wert als String.

4.2.4. Audio

Das Abspielen von Audiodateien sollte in gleicher Weise einfach anwendbar sein wie der Camera-Service. Der Audio-Service stellt hierzu zwei Methoden zur Verfügung. Die erste Methode der Klasse `AudioServiceImpl` heißt `playAudio()` und erhält als Parameter den Pfad zu der Audiodatei. Diese erkennt anhand der Dateiendung um welches Format es sich handelt und spielt den Ton ab. Die zweite Methode ist eine überladene Methode der Ersten und ruft diese intern auf. Davor wird allerdings noch die Lautstärke gesetzt, die in dieser Methode als zusätzlicher Parameter mitgegeben werden kann.

4.2.5. Weitere Aufgaben

Sobald während des Projektes etwas Leerlauf entstand, konnte man seine freie Kapazität dazu nutzen in anderen Themengebieten Unterstützung anzubieten, wenn durch die stetige Weiterentwicklung nicht neue Anforderungen aufgekommen sind. Der Autor hatte hierbei noch Mitarbeit beim Thema Location-Service und Timer-Service geleistet.
Der Location-Service bezog sich hauptsächlich darauf, Positionsangaben über GPS zu Empfangen und diese der Anwendung zur Verfügung zu stellen bzw. Events zu generieren, wenn eine bestimmt Position mit dem Gerät erreicht wurde.
Beim Timer-Service handelte sich es um Konfiguration und Generierung von Events zu einem bestimmten Zeitpunkt oder Zeitintervall.
Für weitere Informationen sei auf die Arbeiten der jeweiligen Kommilitonen verwiesen [9].

[12] Semacode Homepage

5. Fazit

Mit diesem abschließenden Kapitel gibt der Autor eine Zusammenfassung zu dem Projekt. Als Ergebnis kann gesagt werden, dass das Projekt mit Erfolg abgeschlossen wurde. Bis auf Kleinigkeiten wurde ein grundlegend einsatzfähiges Framework für pervasive Spiele entwickelt, das mit zwei prototypischen Anwendungen validiert wurde.

Zu Anfang wurde die Größe des Teams für etwas groß befunden, was die Gefahr von zu großem Verwaltungsoverhead, im Sinne von zu vielen Diskussionen über Entscheidungen mit sich bringen könnte. Doch hat eine gute Planung und Aufgabenverteilung in kleinen Gruppen gezeigt, das dem nicht so war und es sich eher positiv auf die Ideenvielfalt auswirkte. In Betracht der zur Verfügung stehenden Zeit von 8Std./Woche in einem Semester und des angestrebten Projektziels war die Teamstärke gewiss angemessen. Dazu beigetragen haben sicherlich auch die wöchentlichen Teammeetings, in denen der Stand dargelegt und weiteres Vorgehen besprochen wurde. Wichtig ist jedoch, diese in Zeit und Themen nicht ausufern zu lassen.

Die Entscheidung für Java als Programmiersprache hatte einige Hürden mit sich gebracht, wie in Abschnitt 4.2 kurz beschrieben. Somit war es anfangs nicht so einfach ein geeignetes Gerät für unsere Anforderungen zu finden. Jedoch hatte sich in Gesprächen mit dem Vorsemester, die auf .Net gesetzt hatten, herausgestellt, das es auch mit der Sprache manche Schwierigkeiten gab. Für die Realisierung bat die Entwicklungsumgebung EclipseME (Mobile Edition) eine sehr gute Unterstützung. Getestet werden konnte auch in einem Simulator, obwohl man nicht immer davon aus gehen konnte, dass der Code dann auch auf dem mobile Gerät wie gewünscht lief. Das Testen nahm durch das lange hinüber kopieren zum Geräte und Starten viel Zeit in Anspruch.

Das Framework stellt durch seinen modularen Aufbau und die Erweiterbarkeit ein gelungenes Konzept dar. Sicherlich gibt es jedoch noch einige Punkte die optimiert werden können. Bezüglich der Ressourcennutzung und Performanceoptimierung wurden keine Tests gemacht. Dies wäre sicher ein Punkt, der noch ausgearbeitet werden kann.
Sinnvoll wäre auch die Erstellung einer Schemadatei (XSD) für das XML. Aufgrund der ständigen Veränderung der spine.xml wurde dies nach einiger Zeit zurückgestellt. Dies verringert aber die Fehleranfälligkeit beim Aufstellen der XML-Datei und der Parser muss nicht alle Fehlermöglichkeiten einzeln abfangen, da sie durch die XSD schon angezeigt werden.

Insgesamt hat das Praktikum viel Spaß gemacht. Die Organisation und Entwicklung im Team waren spannend und aufschlussreich. Einziger Kritikpunkt war der knappe Zeitrahmen.

Literatur

[1] GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Design Patterns, Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. – ISBN 0-201-63361-2

[2] HAROLD, Elliotte R. ; MEANS, W. S.: *XML In a Nutshell*. O'Reilly, 2001. – ISBN 0-596-00058-8

[3] KRASNER, Glenn E. ; POPE, Stephen T.: A cookbook for using the model-view controller user interface paradigm in Smalltalk-80. In: *J. Object Oriented Program.* 1 (1988), Nr. 3, S. 26–49. – ISSN 0896-8438

[4] OESTEREICH, Bernd: *Objektorientierte Softwareentwicklung : Analyse und Design mit der UML 2.0*. 6. Auflage. Oldenburg Verlag, 2005. – ISBN 3-486-57654-2

[5] PREE, W.: *Meta patterns - a means for capturing the essentials of reusable object-oriented design*. in M. Tokoro and R. Pareschi (eds), Springer-Verlag, proceedings of the ECOOP, Bologna, Italy: 150-162. 1994

[6] WALTHER, Bo K.: Atomic actions – molecular experience: theory of pervasive gaming. In: *Comput. Entertain.* (2005). – URL http://doi.acm.org/10.1145/1077246.1077258. – ISSN 1544-3574

[7] WIKIPEDIA: *Software framework*. – URL http://en.wikipedia.org/wiki/Software_framework. – Zugriffssdatum: 2008-02-16

[8] WIKIPEDIA: *HTML*. – URL http://en.wikipedia.org/wiki/HTML. – Zugriffssdatum: 2008-02-16

[9] HAW HAMBURG: *UbiComp Projekte*. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projects.html

[10] WEISER, Marc: *The Computer for the 21st Century*. 1991. – URL http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html. – Zugriffssdatum: 2008-02-16

[11] WIKIPEDIA: *Ubiquituos Computing*. – URL http://en.wikipedia.org/wiki/Ubiquitous_computing. – Zugriffssdatum: 2008-02-16

[12] WIKIPEDIA: *XML*. – URL http://en.wikipedia.org/wiki/XML. – Zugriffssdatum: 2008-02-16

A. Anhang

A.1. Spine.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<spine default="standard">
  <actions>
    <action name="startUpAction"
      class="de.haw.hamburg.pgf.mobile.community.action.
       StartUpAction"
      singleton="true" />
    <action name="loginActionName"
      class="de.haw.hamburg.pgf.mobile.community.action.
       LoginAction"
      singleton="false" />
    <action name="mainListSelectItemAction"
      class="de.haw.hamburg.pgf.mobile.community.action.
       MainListSelectItemAction"
      singleton="false" />
    ...
  </actions>

  <views>
    <view name="loginView"
      class="de.haw.hamburg.pgf.mobile.community.view.LoginView"
      singleton="true" />
    <view name="mainView"
      class="de.haw.hamburg.pgf.mobile.community.view.MainView"
      singleton="false" />
    <view name="searchForNewBuddyView"
      class="de.haw.hamburg.pgf.mobile.community.view.
       SearchForNewBuddyView"
      singleton="true" />
    <view name="searchResultNewBuddyView"
      class="de.haw.hamburg.pgf.mobile.community.view.
       SearchResultNewBuddyView"
      singleton="true" />
    ...
  </views>

    <exception-handler name="globalException"
      class="de.haw.hamburg.pgf.mobile.community.eventtest.
       TestException" />
```

```
34
35  <flowpoints>
36    <flowpoint name="mainFlowpoint" default="startUpAction">
37      <action-refs>
38        <action-ref name="startUpAction" ref="startUpAction">
39          <view-refs>
40            <view-ref name="main" ref="mainView" />
41          </view-refs>
42        </action-ref>
43        <action-ref name="logOutAction" ref="logOutAction">
44          <view-refs>
45            <view-ref name="logOut" ref="
               forwardToStandardFlowpoint" />
46          </view-refs>
47        </action-ref>
48        <action-ref name="locationInRangeAction" ref="
           locationInRangeAction">
49          <view-refs>
50            <view-ref name="main" ref="forwardToStandardFlowpoint"
               />
51          </view-refs>
52        </action-ref>
53        ...
54      </action-refs>
55
56      <view-refs>
57        <view-ref name="main" ref="mainView" />
58        <view-ref name="displayBuddyInvitation" ref="
           displayBuddyInvitationView" />
59        <view-ref name="searchForNewBuddy" ref="
           searchForNewBuddyView" />
60        ...
61      </view-refs>
62
63      <events>
64        <event name="locationUpdated" handler="de.haw.hamburg.pgf.
           mobile.spine.event.location.handler.LocationEventHandler"
           >
65          <parameters>
66            <parameter name="interval" value="4"/>
67            <parameter name="timeout"  value="-1"/>
68            <parameter name="maxage"   value="-1"/>
69          </parameters>
```

```
70          <eventaction type="update" class="de.haw.hamburg.pgf.
             mobile.community.action.LocationInRangeAction">
71            <view-refs>
72              <view-ref name="main" ref="mainView" />
73            </view-refs>
74          </eventaction>
75          <eventaction type="stateChanged" class="de.haw.hamburg.
             pgf.mobile.community.eventtest.
             LocationStateChangedEventActionImpl">
76            <view-refs>
77              <view-ref name="locationStateChanged" ref="
                 locationUpdatedView" />
78            </view-refs>
79          </eventaction>
80        </event>
81        ...
82      </events>
83    </flowpoint>
84    ...
85  </flowpoints>
86 </spine>
```