Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bericht INF-M3 PO

Markus Dreyer

Pervasive Spine

Fakultät Technik und Informatik
Department Informatik

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1 Einleitung und Motivation

Die Leistungsfähigkeit mobiler Endgeräte nimmt ständig zu. Dadurch können neue Anwendungskonzepte realisiert werden. Gerade der Bereich pervasiver Spiele hat großes Potential und ist ein aktuelles Forschungsgebiet. Zur vereinfachten Implementierung neuer Anwendungen soll im Rahmen einer Projektveranstaltung des Masterstudienganges Informatik an der Hochschule für Angewandte Wissenschaften Hamburg (HAW), ein Framework entwickelt werden. Die Entwicklung soll anhand konkreter Anwendungen validiert werden. Ansonsten steht die Sammlung von Erfahrungen im Vordergrund. Diese Erfahrungen oder daraus resultierende Ideen, können anschließend in die Masterarbeit einfließen.

2 Projekt

Der Masterstudiengang Informatik der HAW bietet den Studierenden im dritten Semester eine Projektveranstaltung an. In dieser Zeit sollen die Studenten selbständig und eigenverantwortlich arbeiten. Zu den Lernzielen zählen die Ausarbeitung von fachlichen Anforderungen, sowie die Präsentation der erarbeiteten Konzepte und Lösungen. Zur Förderung der Teamarbeit und Qualitätssicherung sollen regelmäßige Besprechungen und moderierte Diskussionen durchgeführt werden. Die Zusammenarbeit verlangt von den Teilnehmern weiterhin eine gute Abstimmung über Teilgebiete der Konzepte für die spätere Integration, sowie Kompromissbereitschaft zur Konfliktlösung.

In diesem Semester besteht das Projekt aus acht Masterstudenten und zwei betreuenden Professoren. Die Projektdurchführung wird von den Studenten gestaltet, wobei von den Professoren lenkende Hilfen gegeben werden. Der zeitliche Umfang ist mit einem Personentag (PT) pro Woche und Student festgelegt, wodurch für das Projekt 128 PT zur Verfügung stehen.

Eine detaillierte Beschreibung der Abläufe findet sich in 2.6, während 2.5 die Aufgaben der Projektmitglieder aufzeigt. Als Einstieg werden kurz die Grundlagen erläutert, gefolgt von der Vision des Projekts. Diese wird anhand von Szenarien konkretisiert, aus denen die Anforderungen abgeleitet werden. Kapitel 3 befasst sich mit den Konzepten und der Realisierung der vom Autor bearbeiteten Teilbereiche. Den Abschluss bildet eine Zusammenfassung und ein Blick in die Zukunft.

2.1 Grundlagen

Das Projekt widmet sich dem Thema „Pervasives Spielen“. Dieser Ausdruck beschreibt einen relativ jungen Ansatz im Bereich der Spiele. Durch die Allgegenwart von Computern und Kommunikationsmöglichkeiten können Spiele eine Verbindung der realen zur virtuellen Welt herstellen. Die Technologien unterstützen den Spieler bei seinen Vorhaben.

Das pervasive Spielen ist eine Spezialisierung des pervasiven Rechnens, welches bereits 1993 von Marc Weiser geprägt wurde. Er stellte die Allgegenwart der Computertechnologien in den Vordergrund.

In dem Projekt soll ein Framework entwickelt werden. Ein Framework ist dabei eine Sammlung von Klassen. Diese zeichnet sich dadurch aus, dass sie für einen bestimmten Anwendungsbereich wiederverwendbare Komponenten bereitstellt. Die Architektur des Frameworks wird von dem Entwickler nicht beeinflusst, viel mehr entsteht die neue Anwendung aus der Nutzung vorgegebener Strukturen und Klassen, die an die konkrete Anwendung angepasst werden. Dies geschieht nicht durch Änderung des Codes an sich, sondern durch die Instanzierung von eigenen Klassen, die Eigenschaften von Frameworkklassen erben oder widerspiegeln.

2.2 Vision

Ziel des Projektes ist die Entwicklung eines Framework für pervasive Spiele. Dieses soll vorerst auf den Outdoorbereich beschränkt bleiben, um die Techniken und Technologien darauf basierend auszuwählen. Auf Grundlage des Frameworks sollen zwei Anwendungen realisiert werden, die zur Überprüfung der Funktionalitäten dienen. Die Basis sollen dabei bereits heute verfügbare Technologien stellen, die als Module in das Framework integriert werden. Damit stehen diese Technologien für unterschiedlichste Spielkonzepte zur Verfügung und eine spätere Erweiterung ist möglich.

Der Name des Frameworks soll „Pervasive Spine“ - kurz PS - lauten. Pervasive meint dabei die unter 2.1 beschriebenen Eigenschaften und Spine ist das Rückgrad. Somit soll eine stabile und dennoch flexible Software entstehen.

2.3 Szenario

Wie bereits in 2.2 erwähnt, sollen zwei Spiele zur Validierung des Frameworks implementiert werden. Sie dienen im ersten Schritt der Definition von fachlichen und technischen Anforderungen, denen das Framework gerecht werden muss. Die beiden Anwendungen werden im Folgenden vorgestellt.

2.3.1 PSCommunity

Wie der Name „Pervasive Spine Community“ schon sagt, wird die Anwendung von den Spielern getragen. Es soll ein Netzwerk von Benutzern aufgebaut werden, die den Inhalt selber verändern und ausbauen können. Die inhaltlichen Objekte beschränken sich für den Prototypen auf Orte und andere Personen. Zu diesen können von jedem Mitglied der Gemeinschaft Daten hinterlegt und geändert werden. Hierzu gehören Bilder und Texte, die anderen Teilnehmern zugänglich sind und automatisch durch die integrierte Positionierung angezeigt werden. So können die Benutzer besondere Orte oder Personen in ihrer aktuellen Umgebung ausfindig machen. Weiterhin ist eine Kommunikation zwischen den Mitspielern geplant. Diese soll dazu dienen, kurzfristige Treffen auszumachen oder einfach mit einander in Kontakt zu kommen.

2.3.2 PSOe

Lang ausgeschrieben lautet der Name des zweiten Spiels „Pervasive Spine Orientierungseinheit“. Es soll eine Technisierung der am Fachbereich stattfindenden Rallye ermöglichen. Diese wird im Rahmen des ersten Semesters für neue Studenten durchgeführt, um den Campus und wichtige Orte, wie das Studentensekretariat, den FH-Laden und die Kopierräume, kennenzulernen. Für die Teilnahme an dem Spiel soll sich eine Gruppe an einem Server

anmelden. Von diesem erhält sie nach dem Spielbeitritt Hinweise auf den zu findenden Ort. Um die Suche zu vereinfachen, soll auf einer Karte die aktuelle Position, sowie Richtung des Ziels dargestellt werden. Ist ein Ort gefunden worden, kann die benötigte Zeit zur Erstellung einer Highscoreliste an den Server übermittelt werden. Dieser stellt dann die Hinweise für das folgende Ziel bereit.

Das Spiel ist beendet, sobald alle vorgegebenen Orte erreicht wurden. Die Reihenfolge der Routenpunkte soll dabei sowohl dynamisch als auch statisch festgelegt werden können. Weiterhin muss das Spiel unterbrochen werden und temporär ohne Serverkommunikation arbeiten können.

2.4 Anforderungen

Anhand der unter 2.3 vorgestellten Szenarien werden folgende fachlichen und technischen Anforderungen festgelegt. Diese werden bewusst generell gehalten, da die grundlegenden Funktionen des Frameworks im Fokus stehen. Die in dieser Arbeit behandelten Anforderungen werden im Kapitel 3 detailliert erläutert.

2.4.1 Fachliche Anforderungen

Für die Umsetzung von PSCommunity werden mehrere Komponenten benötigt. Eine der wichtigsten ist die Verwaltung von Objekten auf dem mobilen Gerät. Dazu gehört das Erstellen und Bearbeiten von Orten oder Personen. Die so gespeicherten Daten sollen über den Server publiziert werden können.

Ein weiterer Schwerpunkt besteht in der Lokalisierung des Anwenders und weiterer Teilnehmer, sowie Orte. Dafür wird auf dem Gerät eine Möglichkeit zur Positionsbestimmung benötigt, ebenso wie eine Relation vom eigenen Ort zu einem anderen Punkt. Die Entfernung soll sowohl auf dem Server, als auch auf dem Gerät selbst, bestimmt werden. Der Server hat so die Möglichkeit Neuigkeiten oder Beschreibungen zu nahegelegenen Orten bereitzustellen.

Für die Kommunikation mit anderen Spielern soll ein Chat und ein Nachrichtendienst realisiert werden. Diese soll direkt oder indirekt über den Server stattfinden. Das Spiel muss zu jeder Zeit benutzerabhängig unterbrochen und fortgesetzt werden können. Hierfür muss eine Verwaltung von Benutzerdaten integriert werden.

Das PSOe Spiel benötigt einen Mechanismus zum Verwalten von Spielen und Teilnehmern. Spieler sollen die Möglichkeit haben, eigene Rallyes zu erstellen oder vorhandenen beizutreten. Der Spielverlauf soll unterbrechbar sein und später fortgesetzt werden können. Auf dem Server werden Highscores gespeichert. Dazu muss das Erreichen eines Ortes verifiziert und die Daten an den Server übermittelt werden. Zur Orientierung soll die aktuelle Position und die Richtung zum nächsten Ziel auf einer Karte dargestellt werden. Diese wird

vom Server bereitgestellt, ebenso wie die Koordinaten des nächsten Ziels. Mit diesen Daten soll das mobile Gerät die Karte um die Positionen des Spielers darstellen.

Beide Spiele benötigen zum Teil ähnliche Komponenten. Die folgende Tabelle zeigt die Anforderungen und die Relevanz für das jeweilige Spiel.

	PSCommunity	**PSOe**
Objektübertragung (direkt / indirekt)	✓/ ✓	✓/ -
Spieleverwaltung (erstellen / beitreten / pausieren / fortsetzen)	- / ✓/ ✓/ ✓	✓/ ✓/ ✓/ ✓
Benutzerverwaltung (login / logout)	✓/ ✓	✓/ ✓
Lokalisierung (Nähe / Position)	✓/ ○	○ / -
Objektverwaltung	✓	-
Kartendarstellung	○	✓
Chat	✓	-
Nachrichtendienst	✓	-

-=nicht benötigt | ○=optional |✓=benötigt
Tabelle 1: Fachliche Anforderungen

2.4.2 Technische Anforderungen

In der Tabelle der fachlichen Anforderungen sind viele Überschneidungen erkennbar. Aus der Vereinigung lassen sich die technischen Anforderungen ableiten.

Zur zentralen Verwaltung der Spiele und Benutzer, sowie der Objekte, benötigt das mobile Gerät eine Möglichkeit, mit dem Server zu kommunizieren (WLAN, UMTS,usw.). Über diesen Weg lassen sich auch der Chat und der Nachrichtendienst realisieren. Für die direkte Kommunikation eignet sich Bluetooth oder WLAN.

Damit Spiele fortgesetzt werden können, müssen die Daten beim Pausieren auf dem Gerät persistiert und später geladen werden können. Gleiches gilt für die Serveranwendungen.

Zur Lokalisierung eines Teilnehmers bietet sich GPS an. Dafür müssen die Geräte mit den Koordinaten umgehen können. Für die Verifikation an einer Position können Barcodes eingesetzt werden, welche mit Hilfe der Kamera des mobilen Gerätes fotografiert werden. Für die Unterstützung mehrerer Anwendungen soll eine Konfigurationsdatei in XML-Format bereitgestellt werden, die den groben Ablauf eines Spiels repräsentiert.

Alle Implementierungen sollen die Funktionalitäten nach Möglichkeit kapseln, so dass ein Hardwareaustausch nicht eine komplette Reimplementierung bedeutet.

2.5 Aufgabenverteilung

Die unterschiedlichen Bereiche des Projektes werden auf die Teilnehmer aufgeteilt. Dabei werden die persönlichen Interessen und der Umfang der Aufgaben berücksichtigt. Es werden drei Gruppen gebildet. Das PSOe Team besteht aus Jan Schönherr und Leif Hartmann. Herr Schönherr implementiert hauptsächlich die Klientenseite, während Herr Hartmann die Serveranwendung entwickelt. Für das PSCommunity Spiel entwickeln Andreas Herglotz und Ralf Kruse die benötigten Komponenten. Das Frameworkteam besteht aus vier Studenten. Alexander Mas bearbeitet die XML Konfiguration und Basiskomponenten des Framework. Jan-Peter Tutzschke entwickelt große Teile der Framework-Komponenten und die Schnittstellen für die Basisdienste. Sven Vollmer implementiert die direkte Kommunikation exemplarisch mit Bluetooth. Markus Dreyer implementiert die Abstraktionsschicht für die direkte Kommunikation. Nähere Informationen zu den einzelnen Ausarbeitungen sind im Literaturverzeichnis zu finden.

2.6 Ablauf

Zu Beginn des Projektes findet eine Orientierungsphase statt, in der die Projektmitglieder in Teams aufgeteilt werden. Im Plenum wird über die zu benutzende Programmierplattform und die Hardware, sowie die Organisation abgestimmt. Zur Dokumentation wird ein Wiki eingerichtet und zum weiteren Vorgehen ein Projektplan erstellt. Zur Abstimmung zwischen den Teams werden wöchentlich Meetings abgehalten, bei denen auch die Zieldefinition und die Meilensteine abgeprüft werden. Zusätzlich werden Erfahrungen aus dem Vorsemester mit in die Entscheidungsfindungen einbezogen.

Als zweite Phase werden technische Möglichkeiten evaluiert und in kleinen Anwendungen getestet. Diese ergeben sich aus den Anwendungsfällen bzw. den Anforderungen. Ab diesem Zeitpunkt werden die Teams unabhängig ihre Komponenten entwickeln. Durch Meetings in dieser Phase kann das Framework iterativ weiterentwickelt und fertiggestellte Komponenten in die Anwendungen integriert werden. Am Ende sollen die Spiele in Betrieb genommen und real getestet werden. Abschließend werden die Projektergebnisse präsentiert.

3 Konzept

Am Anfang des Projektes werden die grundlegenden Technologien und Techniken festgelegt. Für die Implementierung wird Java als Programmiersprache und EclipseME als Programmierplattform ausgewählt.

Die Hardware soll nach Möglichkeit alle aktuellen JSR unterstützen, sowie die gängigsten Techniken bieten. Zu diesen gehören GPS, UMTS, sowie eine Kamera. Für die Umsetzung kann auf bereits vorhandene Nokia E70 zurückgegriffen werden. Zusätzlich wird ein Nokia N95 angeschafft.

Zur Implementierung der direkten Kommunikation wird eine Komponente benötigt die diese kapselt. Diese wird PeerCommunicationService (Kapitel 3.1.1 und 3.2.1) genannt und soll exemplarisch mittels Bluetooth umgesetzt werden. Für die Datenübertragung wird eine Serialisierung und Deserialisierung benötigt.

Damit das Framework zeitgesteuerte Ereignisse auslösen kann, wird ein TimerService (Kapitel 3.1.2 und 3.2.2) implementiert. Dieser soll aus der Konfiguration als auch aus der Anwendung direkt angesprochen werden können.

Im Laufe des Projektes wurde als weitere Anforderung die Kennzeichnung von veralteten Ereignissen erkannt. Diese Änderung und weitere Ergänzungen zum Framework werden in 3.1.3 und 3.2.3 behandelt.

3.1 Entwurf

In den folgenden Abschnitten werden die drei Hauptaufgaben und Lösungsansätze skizziert. Zur Einordnung der Komponenten wird an dieser Stelle kurz das Framework vorgestellt. Es besteht aus drei unabhängigen Ebenen. Der Kern (grün) beinhaltet die Ereignisverwaltung und definiert die Schnittstellen zu den Applikationen und Basisdiensten (blau). Durch diese Modularisierung bleibt das Framework an sich unabhängig von den eingesetzten Techniken. Damit ein Basisdienst genutzt werden kann, muss dieser Ereignisse an das Framework melden. Das Framework instanziert die Basisdienste anhand der Konfigurationsdatei, die der Anwendungsentwickler erstellt.

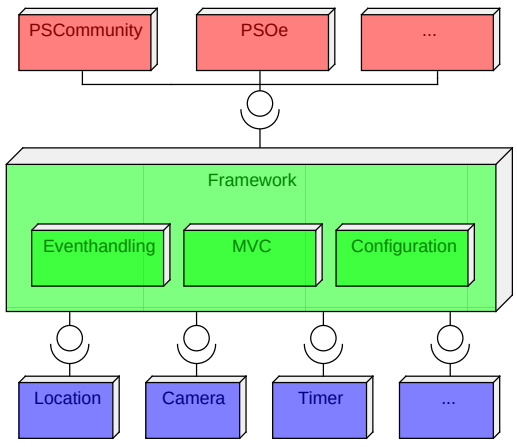

Abbildung 1: Frameworkübersicht

Für die Entwicklung eines Basisdienstes sind die Schnittstellen des Frameworks wichtig. Die Abbildung 2 zeigt alle relevanten Komponenten. Dazu gehört ein Event für das Framework, sowie ein Handler und Listener zur Erzeugung und Verarbeitung des Basisdienstes.

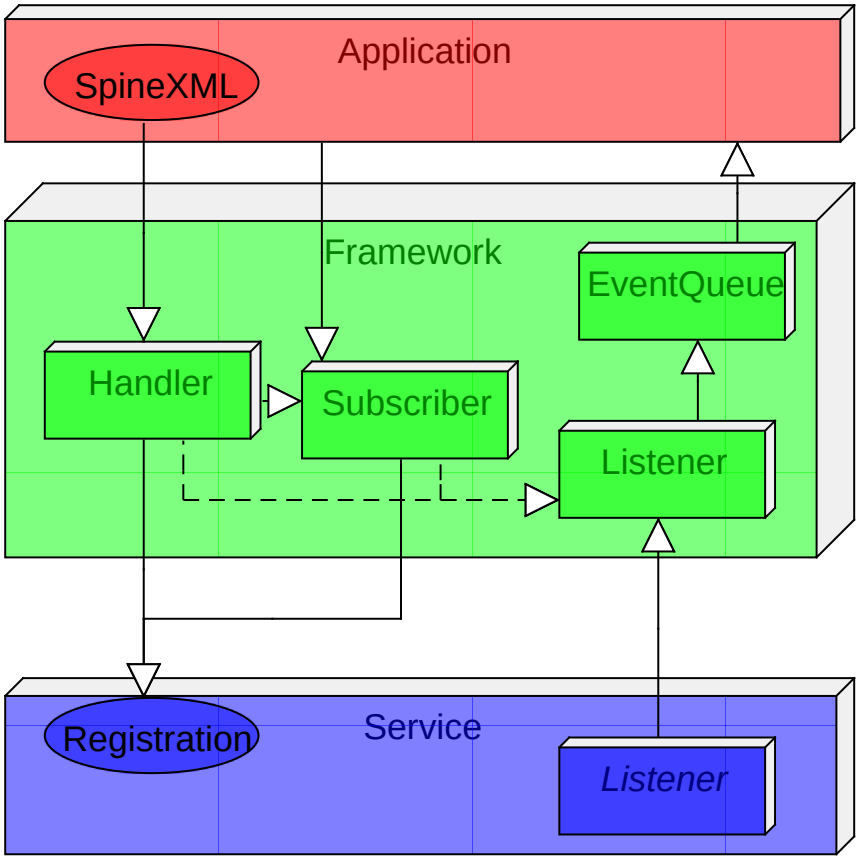

Abbildung 2: Serviceschnittstellen des Framework

Weiterhin wird ein Subscriber benötigt, über den der Basisdienst aus der Anwendung beeinflusst werden kann. Aus dem Event des Frameworks wird eine EventAction in der Applikation ausgeführt. Hier hat der Anwendungsentwickler die Möglichkeit, Aktionen auszuführen oder den Spielfluss zu verändern.

Für eine genauere Betrachtung des Frameworks sei an dieser Stelle auf die Arbeiten von Jan-Peter Tutzschke und Alexander Mas verwiesen.

3.1.1 PeerCommunicationService

Dieser Basisdienst dient der direkten Kommunikation zwischen verschiedenen mobilen Geräten. Hierzu kann auf unterschiedliche Techniken zurückgegriffen werden - z.B. Bluetooth, Infrarot, WLAN. Da es für die Anwendung meist nicht relevant ist, wie die Kommunikation stattfindet, soll diese gekapselt werden.

Der PeerCommunicationService - kurz PCS - muss Schnittstellen bereitstellen, die von der eigentlichen Übertragungsschicht implementiert werden. Welche Techniken zum Einsatz

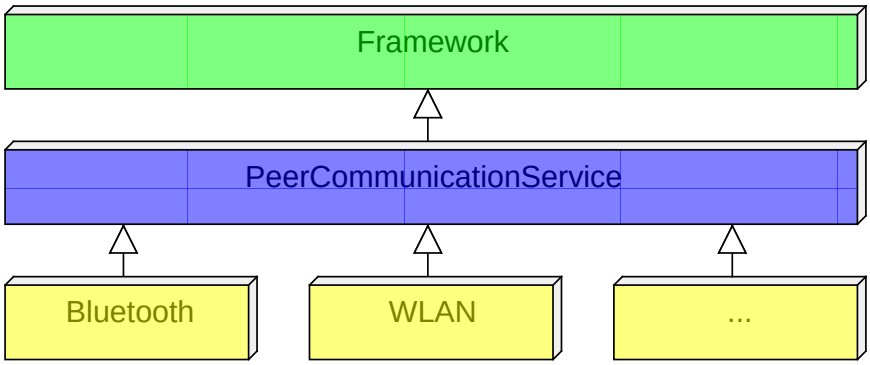

Abbildung 3: PeerCommunicationService Übersicht

kommen, kann in der Konfigurationsdatei festgelegt werden. Der Service kann über den Listener unterschiedliche Events an das Framework melden. Diese sind:

- Teilnehmer gefunden
- Teilnehmer verloren
- Antwort auf Anfrage

Damit diese Ereignisse ausgelöst werden können, muss der PCS die bekannten Teilnehmer verwalten und Objekte übertragen können. Für die Übertragung wird ein extra Interface benötigt, da J2ME keine Serialisierung unterstützt.

3.1.2 TimerService

Wie in Kapitel 3 erwähnt, benötigen viele Anwendungen eine Möglichkeit zeitgesteuert Aktionen auszuführen. Der TimerService soll diese Möglichkeit bieten. Für die Anwendungen sollen sowohl einmalige als auch wiederholende Ereignisse zur Verfügung gestellt werden. Der Service soll ebenso wie der PCS, sowohl über die Konfiguration als auch aus dem Programmcode beeinflusst werden können. Dazu müssen die Frameworkschnittstellen implementiert werden.

3.1.3 Erweiterung des Frameworks

Im Verlauf des Projektes stellt sich heraus, dass veraltete Ereignisse genauso verarbeitet werden wie aktuelle. Dies ist nicht immer gewünscht. Veraltete sind dabei jene, die zu einer Spielsituation erzeugt wurden, aber erst abgearbeitet werden, nachdem sich das Spiel weiterentwickelt hat. Sobald sich die Spielsituation ändert, werden alle vorhandenen Ereignisse als veraltet gekennzeichnet. Inwieweit diese abgearbeitet werden, um das Spiel zu beeinflussen, bleibt dem Anwendungsentwickler überlassen.

3.2 Realisierung

Bevor auf die Umsetzung der einzelnen Komponenten eingegangen wird, erfolgt hier die Darstellung der Funktionalität des Frameworks.

Entscheidend ist dabei die Konfigurationsdatei Spine.xml (siehe Anhang B). In dieser werden - für die eingesetzten Module - die Klassen der Handler angegeben. Diese sorgen für die Instanzierung der Dienste und die Registrierung der Listener, denen später die Events übergeben werden. Die Ereignisse werden in eine Queue eingefügt. Das Framework führt die dazugehörige EventAction, ebenfalls in der Spine.xml referenziert, aus. Diese Action wird vom Anwendungsentwickler erstellt und sorgt für das Fortschreiten der Anwendung. Der programmatische Zugriff auf die Services erfolgt über den entsprechenden Subscriber, der durch den Handler kreiert wird.

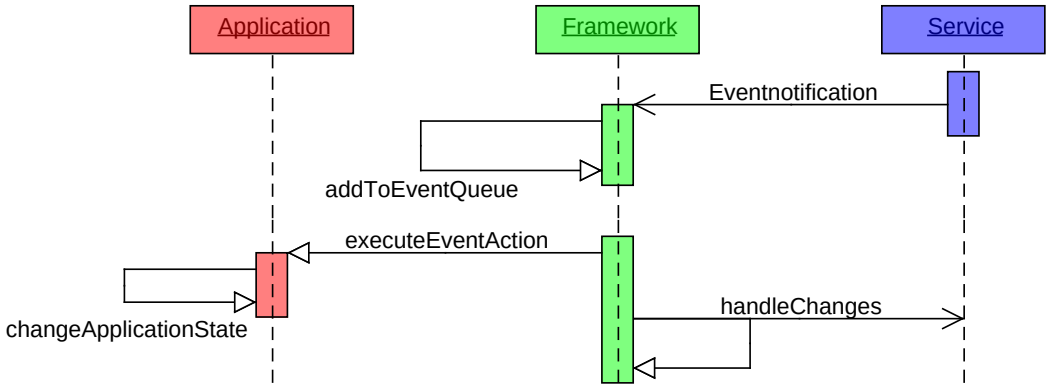

Abbildung 4: Ablauf der Eventverarbeitung

3.2.1 PeerCommunicationService

Zur Kapselung der direkten Kommunikation wird eine weitere Schicht erstellt. Diese wirkt zum Framework wie eine einzelne Komponente und besitzt eine eigene Konfigurationsdatei. In dieser sind die Klassen angegeben, die für die tatsächliche Kommunikation zuständig sind.

Sobald der PCS vom Framework instanziert wird, erstellt dieser die Kommunikationsinstanzen. Diese müssen das BasePeerCommunicationService - BPCS - Interface implementieren, während der PeerCommunicationService Listener der BasePeerCommunicationServices ist.

Die Suche nach anderen Teilnehmern kann automatisch durch den PCS erfolgen oder direkt durch den Anwendungsentwickler. Die Gefundenen werden in einer Map gespeichert, wodurch auch eine Zuordnung zu dem auffindenden BPCS geschieht. Anhand dieser Map

werden auch nicht mehr gefundene Teilnehmer zum Framework gemeldet. Für die Übertragung der Daten kann der Anwendungsentwickler beliebige PeerService oder PeerServiceRequest Objekte erstellen. PeerServices sind die von einem Teilnehmer angebotenen Dienste. Von diesen wird die *execute* Methode aufgerufen. Ein PeerServiceRequest Objekt ist eine Anfrage, die die auszuführende Logik enthält. Auch hier wird die *execute* Methode aufgerufen. Hierbei übergibt der Empfänger allerdings eine Referenz auf den eigenen Kontext.

Zum Aufruf eines entfernten Dienstes bietet der PeerCommunicationService die Methode *sendPeerRequest* an. Diese erwartet den Empfänger und den aufzurufenden Service. Der PCS sucht den zugehörigen Übertragungsdienst aus und überträgt die Anfrage über diesen. Sobald die Antwort eintrifft, wird diese als Event an das Framework weitergeleitet. Ein exemplarischer Übertragungsvorgang ist im Anhang als Sequenzdiagramm hinterlegt (Abbildung A).

3.2.2 TimerService

Der TimerService ist ebenso wie der PCS in das Framework integriert. Für jeden Timer wird ein eigener Listener instanziert. Damit jedem Listener eine EventAction zugeordnet werden kann, muss in der Spine.xml ein eigener Timername deklariert werden.

```
1  <event name="timerUpdated"
2    handler="de.haw.hamburg.pgf.mobile.spine.event.timer.handler.
       TimerEventHandler">
3    <parameters>
4      <parameter name="delay" value="0" />
5      <parameter name="interval" value="3000" />
6    </parameters>
7    <eventaction type="timerReached" class="de.haw.hamburg.mobile.
       application.event.event.TimerReachedEventActionImpl">
8      <view-refs>
9        <view-ref name="success" ref="timerReachedViewName" />
10     </view-refs>
11   </eventaction>
12 </event>
```

Listing 1: Konfigurationsauszug des Timer

Zu der Deklaration können Parameter mit angegeben werden. Soll ein Timer erst während des Programmablaufes genutzt werden, wird dieser ohne Parameter angegeben und kann später über den Subscriber instanziert werden. Der TimerService benutzt zur zeitlichen Steuerung intern den J2ME Timer und benachrichtigt beim Erreichen der gewünschten Zeit den Listener. Alle TimerServices können über den Subscriber abgeändert oder gestoppt werden.

3.2.3 Erweiterung des Framework

Das Dirty Flag wird in das Framework integriert, indem bei einem Wechsel des Anwendungspunktes alle vorhandenen Events in der Queue markiert werden. Bei diesen wird später von der EventAction nicht die *execute*, sondern die *executeDirty* Methode aufgerufen. Zur Vereinfachung für den Entwickler wird die abstrakte Klasse AbstractEventAction eingeführt. Diese implementiert die Methode *executeDirty* mit einem leeren Methodenrumpf. Dadurch werden diese Ereignisse ignoriert. Wenn der Entwickler auf veraltete Ereignisse reagieren will, kann er die Methode in der konkreten EventAction überschreiben.

4 Zusammenfassung und Ausblick

Jeder Teilnehmer des Projektteams kann seine persönlichen Schwerpunkte in die Arbeit einbringen. Die Projektorganisation fördert die Früherkennung von Verzögerungen im Projektablauf und sorgt dafür, dass beide Anwendungen zum Ende des Projektes fertig gestellt werden. Dadurch ist die Funktionalität des Frameworks verifiziert, kann also den gestellten Anforderungen gerecht werden.

Zur Steigerung der Effektivität der Projektmeetings sollen diese von den Teilnehmern besser vorbereitet werden. Das spart Zeit und sorgt für die Konzentration auf die wesentlichen Punkte. Ebenso sollte die Projektdokumentation nicht auf ein Wiki beschränkt bleiben. Dies eignet sich für iterative Dokumentation während des Projektes, ist aber schwer für eine abschließende Dokumentation zu verwenden, da es meist nicht auf dem aktuellen Stand gehalten wird.

Zu Beginn des Projektes sollte mehr Zeit in die Recherche investiert werden. Dadurch können Probleme eventuell umgangen oder vorzeitig erkannte werden.

Das entwickelte Framework stellt eine solide Basis für die Implementierung mobiler Anwendungen dar. Durch die Einstellung der Weiterentwicklung an J2ME seitens Sun ist eine zukünftige Erweiterung in dieser Umgebung nicht sinnvoll. Viel mehr ist dies der richtige Zeitpunkt auf jüngere Technologien umzusteigen. Dafür drängt sich Android, ein von Google entwickelte Framework, auf. Es basiert ebenfalls auf Java und stellt diverse Komponenten bereit, die nach dem MVC Prinzip realisiert sind. Darauf aufbauend lassen sich spannende Anwendungen entwickeln, die von den Erfahrungen aus dem Projekt profitieren werden.

A UML

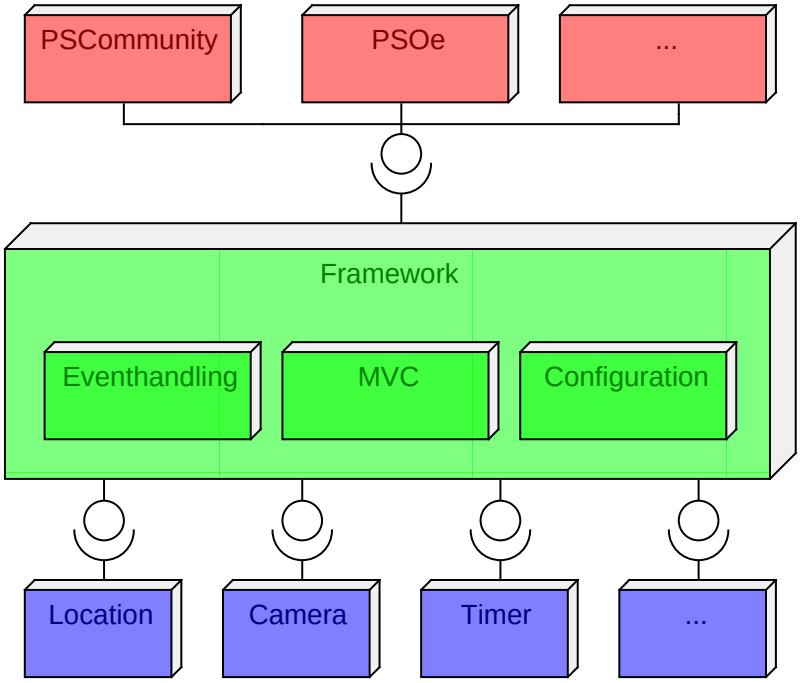

Abbildung A: Frameworkübersicht

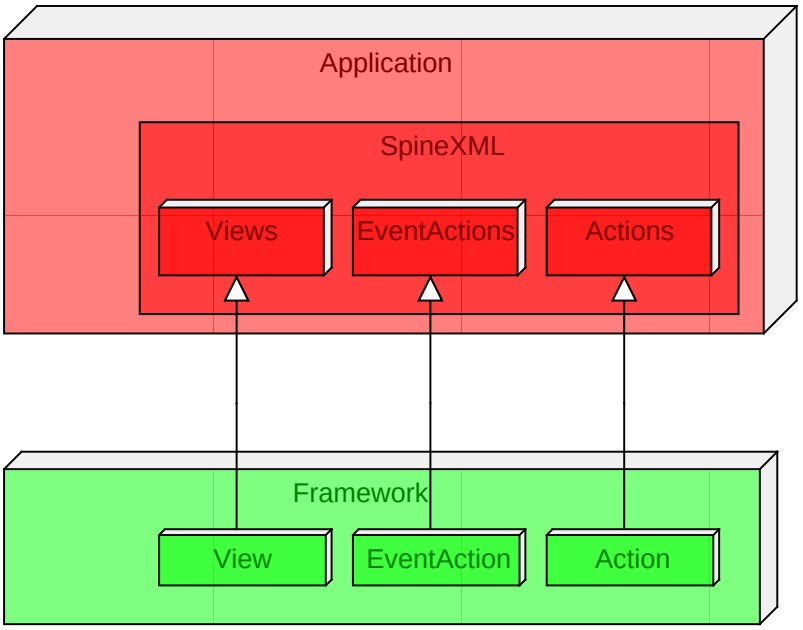

Abbildung B: Applikationsschnittstellen des Framework

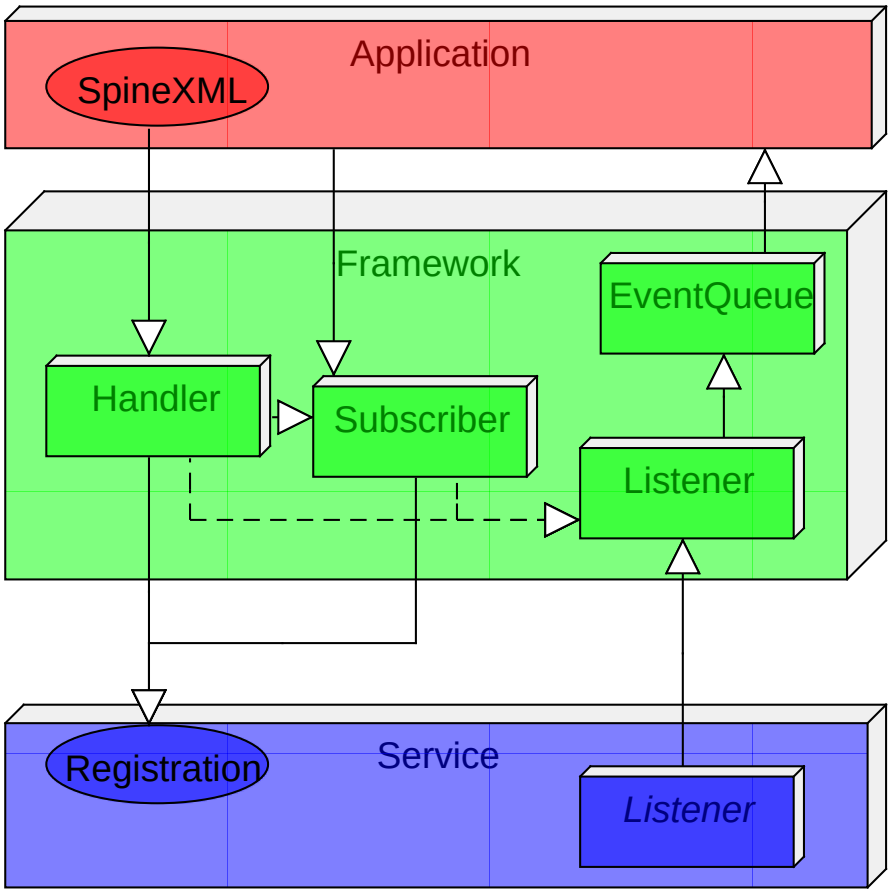

Abbildung C: Serviceschnittstellen des Framework

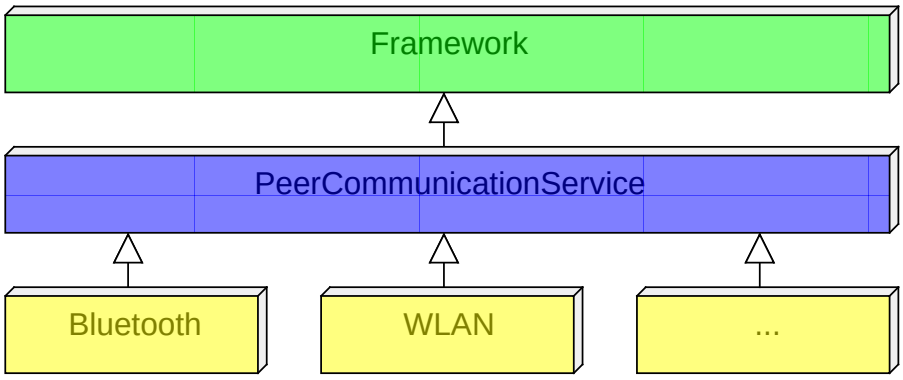

Abbildung D: PeerCommunicationService Übersicht

Abbildung E: PeerCommunicationService Kolaboration

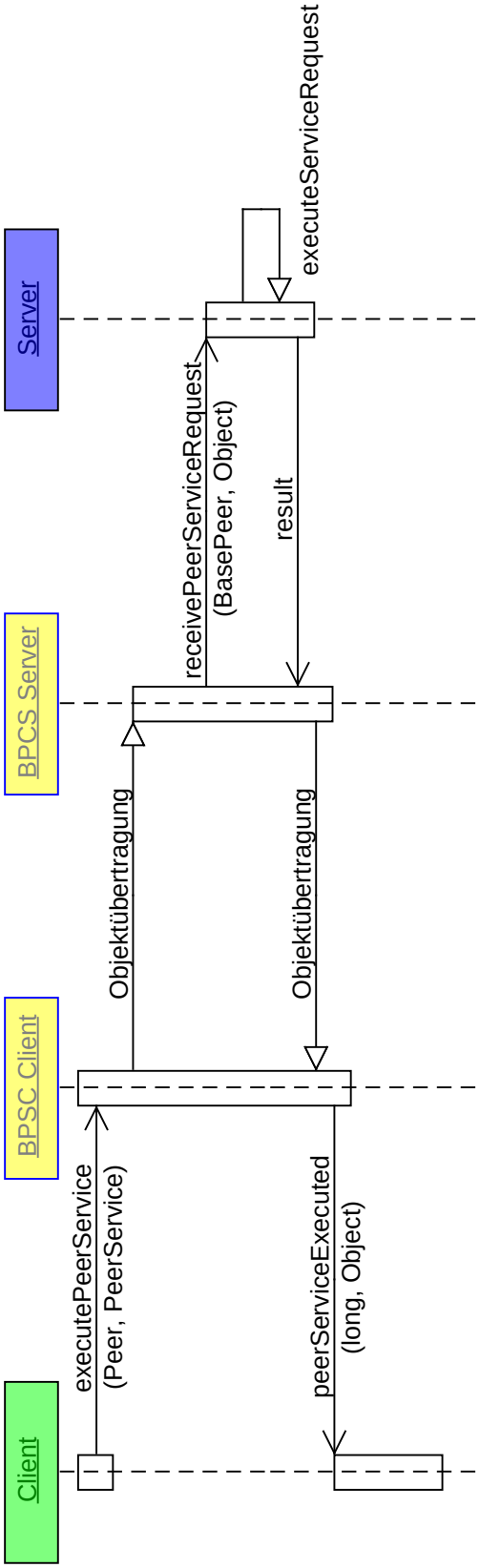

Abbildung F: PeerCommunicationService Sequenzdiagramm

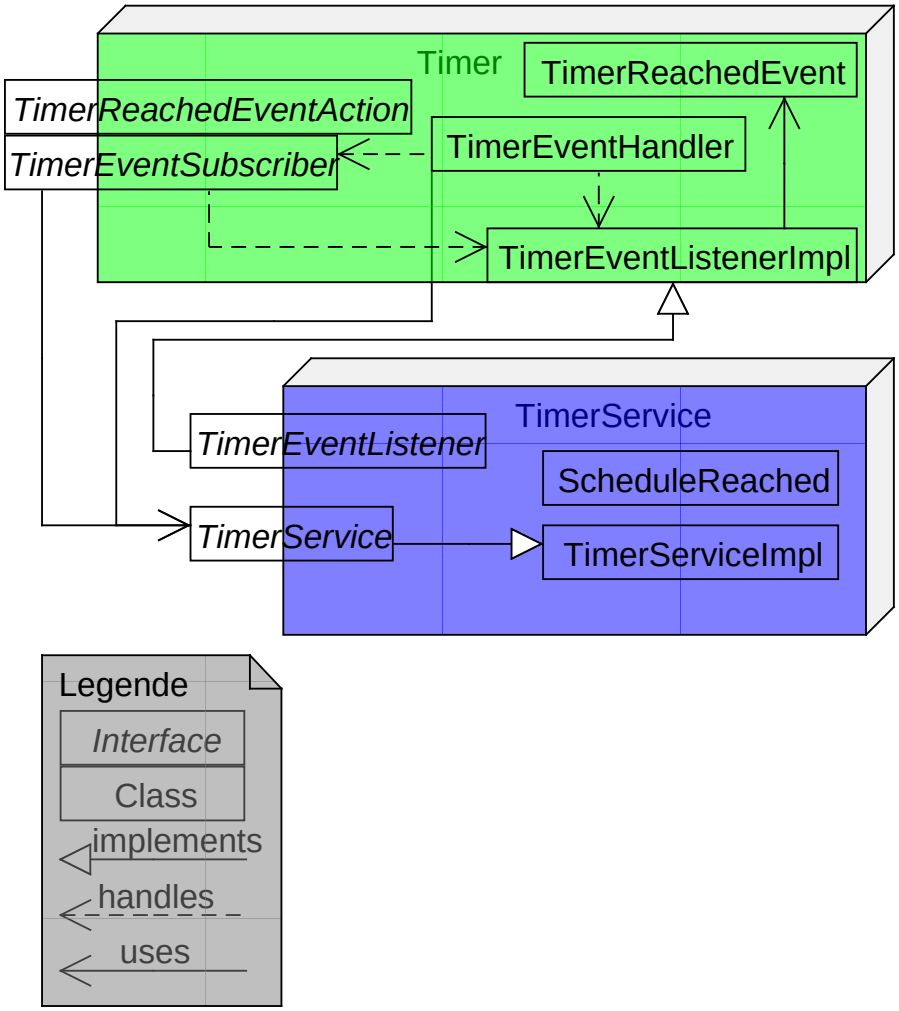

Abbildung G: TimerService

B Spine.xml

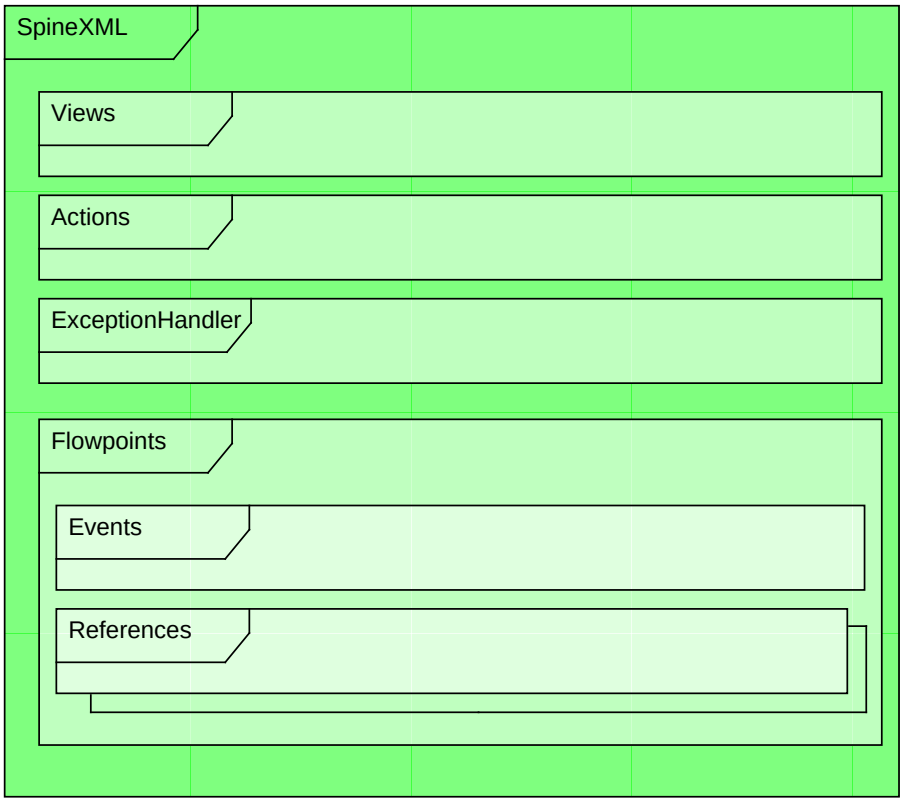

Abbildung H: Aufbau der Spine.xml Konfigurationsdatei

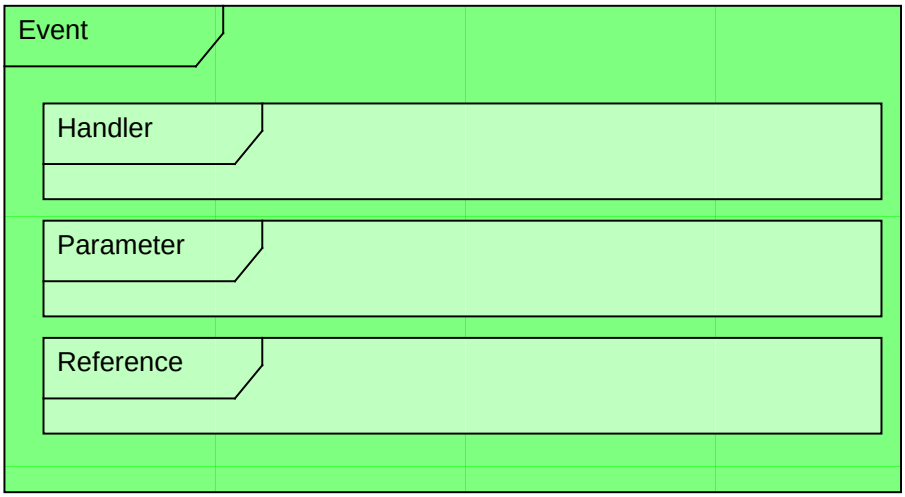

Abbildung I: Aufbau der Spine.xml Events

```
<?xml version="1.0" encoding="UTF-8"?>
<spine default="standard">
  <flowpoints>
    <flowpoint name="standard" default="MainMenuView">
      <action-refs>
        <action-ref name="PollResultsAction" ref="
           PollResultsAction">
          <view-refs>
            <view-ref name="success" ref="AlertView" />
            <view-ref name="failure" ref="AlertView" />
          </view-refs>
        </action-ref>
        <action-ref name="SubmitResultsAction"
          ref="SubmitResultsAction">
          <view-refs>
            <view-ref name="success" ref="AlertView" />
            <view-ref name="failure" ref="AlertView" />
          </view-refs>
        </action-ref>
        <action-ref name="ChooseGameAction" ref="ChooseGameAction"
           >
          <view-refs>
            <view-ref name="success" ref="ChooseGameView" />
            <view-ref name="failure" ref="AlertView" />
          </view-refs>
        </action-ref>
        <action-ref name="ChooseRouteAction" ref="
           ChooseRouteAction">
          <view-refs>
            <view-ref name="success" ref="ChooseRouteView" />
            <view-ref name="failure" ref="AlertView" />
          </view-refs>
        </action-ref>
        <action-ref name="LoginAction" ref="LoginAction">
          <view-refs>
            <view-ref name="failure" ref="AlertView" />
            <view-ref name="success" ref="CreateJoinGameMenuView"
               />
          </view-refs>
        </action-ref>
        <action-ref name="AbortGameAction" ref="AbortGameAction">
          <view-refs>
            <view-ref name="success" ref="MainMenuView" />
```

```
40            </view-refs>
41          </action-ref>
42          <action-ref name="ChooseHintAction" ref="ChooseHintAction"
               >
43            <view-refs>
44              <view-ref name="success" ref="ChooseHintView" />
45              <view-ref name="completed" ref="GoalReachedView" />
46            </view-refs>
47          </action-ref>
48          <action-ref name="CreateGameAction" ref="CreateGameAction"
               >
49            <view-refs>
50              <view-ref name="failure" ref="AlertView" />
51              <view-ref name="success" ref="DummyJoinGameActionView"
                    />
52            </view-refs>
53          </action-ref>
54          <action-ref name="JoinGameAction" ref="JoinGameAction">
55            <view-refs>
56              <view-ref name="failure" ref="AlertView" />
57              <view-ref name="success" ref="WaitForStartView" />
58            </view-refs>
59          </action-ref>
60          <action-ref name="GoalReachedAction" ref="
               GoalReachedAction">
61            <view-refs>
62              <view-ref name="success" ref="GoalReachedView" />
63            </view-refs>
64          </action-ref>
65        </action-refs>
66
67        <view-refs>
68          <view-ref name="LoginView" ref="LoginView" />
69          <view-ref name="MainMenuView" ref="MainMenuView" />
70          <view-ref name="CameraView" ref="CameraView" />
71          <view-ref name="ChooseGameView" ref="ChooseGameView" />
72          <view-ref name="ChooseRouteView" ref="ChooseRouteView" />
73          <view-ref name="ChooseHintView" ref="ChooseHintView" />
74          <view-ref name="CreateGameView" ref="CreateGameView" />
75          <view-ref name="HintReachedView" ref="HintReachedView" />
76          <view-ref name="CreateJoinGameMenuView"
77            ref="CreateJoinGameMenuView" />
78          <view-ref name="GoalReachedView" ref="GoalReachedView" />
```

```
79          <view-ref name="MapView" ref="MapView" />
80          <view-ref name="ShowHintView" ref="ShowHintView" />
81          <view-ref name="ShowScoreView" ref="ShowScoreView" />
82          <view-ref name="WaitForStartView" ref="WaitForStartView" />
83          <view-ref name="AlertView" ref="AlertView" />
84          <view-ref name="DummyJoinGameActionView"
85            ref="DummyJoinGameActionView" />
86        </view-refs>
87
88        <events>
89          <event name="locationUpdated"
90            handler="de.haw.hamburg.pgf.mobile.spine.event.location.
                handler.LocationEventHandler">
91            <eventaction type="update"
92              class="de.haw.hamburg.mobile.application.oeV2.event.
                  LocationUpdateEventActionImpl">
93              <view-refs></view-refs>
94            </eventaction>
95            <eventaction type="stateChanged"
96              class="de.haw.hamburg.mobile.application.oeV2.event.
                  LocationStateChangedEventActionImpl">
97              <view-refs></view-refs>
98            </eventaction>
99          </event>
100
101         <event name="proximity"
102           handler="de.haw.hamburg.pgf.mobile.spine.event.location.
                handler.ProximityEventHandler">
103           <eventaction type="proximity"
104             class="de.haw.hamburg.mobile.application.oeV2.event.
                  ProximityEventActionImpl">
105             <view-refs>
106               <view-ref name="success" ref="HintReachedView" />
107             </view-refs>
108           </eventaction>
109         </event>
110
111       </events>
112
113     </flowpoint>
114   </flowpoints>
115
```

```
116   <actions>
117     <action name="PollResultsAction"
118       class="de.haw.hamburg.mobile.application.oeV2.controller.
            PollResultsAction"
119       singleton="false" />
120     <action name="SubmitResultsAction"
121       class="de.haw.hamburg.mobile.application.oeV2.controller.
            SubmitResultsAction"
122       singleton="false" />
123     <action name="ChooseGameAction"
124       class="de.haw.hamburg.mobile.application.oeV2.controller.
            ChooseGameAction"
125       singleton="false" />
126     <action name="JoinGameAction"
127       class="de.haw.hamburg.mobile.application.oeV2.controller.
            JoinGameAction"
128       singleton="false" />
129     <action name="ChooseRouteAction"
130       class="de.haw.hamburg.mobile.application.oeV2.controller.
            ChooseRouteAction"
131       singleton="false" />
132     <action name="AbortGameAction"
133       class="de.haw.hamburg.mobile.application.oeV2.controller.
            AbortGameAction"
134       singleton="false" />
135     <action name="GoalReachedAction"
136       class="de.haw.hamburg.mobile.application.oeV2.controller.
            GoalReachedAction"
137       singleton="false" />
138     <action name="CreateGameAction"
139       class="de.haw.hamburg.mobile.application.oeV2.controller.
            CreateGameAction"
140       singleton="false" />
141     <action name="ChooseHintAction"
142       class="de.haw.hamburg.mobile.application.oeV2.controller.
            ChooseHintAction"
143       singleton="false" />
144     <action name="LoginAction"
145       class="de.haw.hamburg.mobile.application.oeV2.controller.
            LoginAction"
146       singleton="false" />
147   </actions>
148
```

```
149   <views>
150     <view name="LoginView"
151       class="de.haw.hamburg.mobile.application.oeV2.view.LoginView
              "
152       singleton="true" />
153     <view name="ChooseRouteView"
154       class="de.haw.hamburg.mobile.application.oeV2.view.
            ChooseRouteView"
155       singleton="true" />
156     <view name="MainMenuView"
157       class="de.haw.hamburg.mobile.application.oeV2.view.
            MainMenuView"
158       singleton="true" />
159     <view name="CameraView"
160       class="de.haw.hamburg.mobile.application.oeV2.view.
            CameraView"
161       singleton="true" />
162     <view name="ChooseGameView"
163       class="de.haw.hamburg.mobile.application.oeV2.view.
            ChooseGameView"
164       singleton="true" />
165     <view name="ChooseHintView"
166       class="de.haw.hamburg.mobile.application.oeV2.view.
            ChooseHintView"
167       singleton="true" />
168     <view name="CreateGameView"
169       class="de.haw.hamburg.mobile.application.oeV2.view.
            CreateGameView"
170       singleton="true" />
171     <view name="CreateJoinGameMenuView"
172       class="de.haw.hamburg.mobile.application.oeV2.view.
            CreateJoinGameMenuView"
173       singleton="true" />
174     <view name="GoalReachedView"
175       class="de.haw.hamburg.mobile.application.oeV2.view.
            GoalReachedView"
176       singleton="true" />
177     <view name="MapView"
178       class="de.haw.hamburg.mobile.application.oeV2.view.MapView"
179       singleton="true" />
180     <view name="ShowHintView"
181       class="de.haw.hamburg.mobile.application.oeV2.view.
            ShowHintView"
```

```
182       singleton="true" />
183     <view name="ShowScoreView"
184       class="de.haw.hamburg.mobile.application.oeV2.view.
             ShowScoreView"
185       singleton="true" />
186     <view name="WaitForStartView"
187       class="de.haw.hamburg.mobile.application.oeV2.view.
             WaitForStartView"
188       singleton="true" />
189     <view name="AlertView"
190       class="de.haw.hamburg.mobile.application.oeV2.view.AlertView
             "
191       singleton="true" />
192     <view name="DummyJoinGameActionView"
193       class="de.haw.hamburg.mobile.application.oeV2.view.
             DummyJoinGameActionView"
194       singleton="true" />
195     <view name="HintReachedView"
196       class="de.haw.hamburg.mobile.application.oeV2.view.
             HintReachedView"
197       singleton="true" />
198   </views>
199
200   <exception-handler name="globalException"
201     class="de.haw.hamburg.mobile.application.oeV2.
          OeExceptionHandler" />
202
203 </spine>
```

Listing 2: Spine.xml vom PSOe Spiel

C Klassenverzeichnis

```
res
  \config\
    peerCommunication.properties
  \pgf.properties
src\de\haw\hamburg\pgf\mobile
  \communication\peer
    \base
      \BaseCommunicationListener.java
      \BaseCommunicationService.java
      \BaseCommunicationServiceFactory.java
      \BasePeer.java
      \BasePeerFactory.java
      \Serializable.java
      \SerializableHelper.java
    \impl
      \PeerCommunicationServiceImpl.java
      \PeerImpl.java
    \Peer.java
    \PeerCommunicationListener.java
    \PeerCommunicationService.java
    \PeerCommunicationServiceFactory.java
    \PeerFactory.java
    \PeerService.java
    \PeerServiceRequest.java
  \spine
    \action\Action.java
    \event
      \AbstractEvent.java
      \communication\peer
        \action
          \AbstractPeerFoundEventAction.java
          \AbstractPeerLostEventAction.java
          \AbstractPeerServiceExecutedEventAction.java
          \PeerFoundEventAction.java
          \PeerLostEventAction.java
          \PeerServiceExecutedEventAction.java
        \handler\PeerCommunicationEventHandlerImpl.java
        \listener\PeerCommunicationEventListenerImpl.java
```

```
        \PeerFoundEvent.java
        \PeerLostEvent.java
        \PeerServiceExecutedEvent.java
        \subscriber\impl\PeerCommunicationEventSubscriberImpl.java
        \subscriber\PeerCommunicationEventSubscriber.java
      \timer
        \action
          \AbstractTimerReachedEventAction.java
          \TimerReachedEventAction.java
        \handler\TimerEventHandler.java
        \listener\TimerEventListenerImpl.java
        \subscriber
          \impl
            \TimerEventSubscriberImpl.java
          \TimerEventSubscriber.java
      \TimerReachedEvent.java
      \impl\TimerServiceImpl.java
  \timer
    \TimerEventListener.java
    \TimerService.java
    \TimerServiceException.java
    \TimerServiceFactory.java
```

D Literaturverzeichnis

[Gamma u. a. 2005] GAMMA, E. ; HELM, R. ; JOHNSON, R. ; VLISSIDES, J.: *Design Patterns*. Addison-Wesley, 2005. – ISBN 0-201-63361-2

[Giguere] GIGUERE, Eric: *J2ME Serialization*. – URL http://j2me.synclastic.com/2006/07/25/revisiting-j2me-object-serialization/. – Stand 2008-02-19

[Google-Android] GOOGLE-ANDROID: *Android*. – URL http://code.google.com/android/. – Stand 2008-02-19

[Hartmann 2008] HARTMANN, Leif: *Pervasive Spine*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html. – Bericht INF-M3 PO

[Herglotz 2008] HERGLOTZ, Andreas: *Pervasive Spine*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html. – Bericht INF-M3 PO

[Kruse 2008] KRUSE, Ralf: *Pervasive Spine*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html. – Bericht INF-M3 PO

[Mas 2008] MAS, Alexander: *Pervasive Spine*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html. – Bericht INF-M3 PO

[Microsystems a] MICROSYSTEMS, Sun: *Bluetooth API*. – URL http://java.sun.com/javame/reference/apis/jsr082/. – Stand 2008-02-19

[Microsystems b] MICROSYSTEMS, Sun: *MID Profile*. – URL http://java.sun.com/javame/reference/apis/jsr118/. – Stand 2008-02-19

[Nieuwdorp 2007] NIEUWDORP, Eva: The pervasive discourse: an analysis. In: *Comput. Entertain.* 5 (2007), Nr. 2, S. 13

[Schönherr 2008] SCHÖNHERR, Jan: *Pervasive Spine*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html. – Bericht INF-M3 PO

[Tutzschke 2008] TUTZSCHKE, Jan-Peter: *Pervasive Spine*. 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html. – Bericht INF-M3 PO

[Vollmer 2008] VOLLMER, Sven: *Pervasive Spine*. 2008. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master07-08-proj/berichte.html`. – Bericht INF-M3 PO

[Weiser 1993] WEISER, Mark: Some computer science issues in ubiquitous computing. In: *Commun. ACM* 36 (1993), Nr. 7, S. 75–84. – ISSN 0001-0782

E Glossar

Barcode Maschinenlesbares System aus unterschiedlich dicken Strichen, zur Kodierung von Informationen.

Bluetooth Eine Technik zur drahtlosen Datenübertragung im Nahbereich.

EclipseME Auf Eclipse basierende Entwicklungsumgebung speziell für J2ME Anwendungen.

GPS Global Positioning System, ein satellitengestütztes System zur Positionsbestimmung.

J2ME Sun Java Micro Edition.

JSR Java Specification Request, Beschreibungen für die endgültigen Java Spezifikationen.

MVC Model View Controller, Architektur zur Softwareentwicklung. Dabei werden das Modell, die Präsentation und die Steuerung voneinander getrennt.

UMTS Universal Mobile Telecommunications System, Technik zur Übertragung von Daten für Mobiltelefone.

WLAN Wireless Local Area Network, ein drahtloses Netzwerk zur Verbindung mehrerer Teilnehmer.

XML eXtensible Markup Language, hierarchische Aufbereitung von Daten in Textform. Diese können in der Regel sowohl vom Menschen als auch von Maschinen interpretiert werden.