Diplomarbeit

Sven Oliver Friedburg

Entwicklung einer Plattform für Smartphone-basierte ortsabhängige Interaktionen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sven Oliver Friedburg

Entwicklung einer Plattform für Smartphone-basierte ortsabhängige Interaktionen

Diplomarbeit eingereicht im Rahmen der Diplomprüfung
im Studiengang Softwaretechnik
am Studiendepartment Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: Dipl.-Inform. Birgit Wendholt

Abgegeben am 26. August 2005

Entwicklung einer Plattform für Smartphone-basierte ortsabhängige Interaktionen

Stichworte Smartphone, Mobiltelefon, Bluetooth, J2ME, JABWT, Tracking, drahtlose Kommunikation, ortsabhängige Dienste, mobile Endgeräte

Zusammenfassung

Da sich das Handy längst bei der breiten Masse als ständiger Begleiter etabliert hat, ist die Idee nahe liegend, solche Geräte auch zur Unterstützung der Abläufe sowohl im Arbeitsleben, als auch in der Freizeit eines Benutzers heranzuziehen und diesem situations- und positionsabhängige Dienste anzubieten.
In dieser Arbeit wird eine Plattform entworfen, auf deren Basis mobile Endgeräte drahtlos mittels eines Ad-hoc-Netzwerkes in ein Gesamtsystem integriert werden und mit diesem interagieren. Dabei werden Dienste sowohl seitens des Systems, als auch seitens der mobilen Endgeräte angeboten. Die geografische Position eines mobilen Benutzers innerhalb des Systems wird verfolgt, um die Güte der Dienste zu erhöhen.

Development of a platform for location dependent smart phone based interaction

Keywords Smartphone, mobile phone, Bluetooth, J2ME, JABWT, tracking, wireless communication, location based services, mobile devices

Abstract

Since the mobile phone has been widely accepted as an everyday companion, it seems obvious to use such devices to support everyday workflows as well as leisure activities in a situation and location dependent manner.
This work devises a platform that serves as a basis allowing mobile devices to be integrated into and interact with a complete system via an ad hoc network. Services will be provided by the system as well as by the mobile devices. A mobile user's geographical position will be tracked within the system to enhance the quality of the services.

Inhaltsverzeichnis

1 Einleitung **6**
- 1.1 Motivation . . . 6
- 1.2 Zielsetzung . . . 7
- 1.3 Gliederung . . . 7

2 Grundlagen **8**
- 2.1 Bluetooth . . . 8
  - 2.1.1 Bluetooth-Stack-Architektur . . . 10
  - 2.1.2 Bluetooth-Protokolle . . . 11
- 2.2 J2ME . . . 13
  - 2.2.1 Konfigurationen . . . 15
  - 2.2.2 Profile . . . 16
  - 2.2.3 Optionale Pakete . . . 16
- 2.3 Java Community Process (JCP) . . . 17

3 Analyse **18**
- 3.1 Beispielszenario Ferienclub . . . 18
  - 3.1.1 Ankunft eines Gastes . . . 19
  - 3.1.2 Profilierung des Gastes . . . 19
  - 3.1.3 Veranstaltungskalender . . . 20
  - 3.1.4 Navigation innerhalb des Ferienclubs . . . 22
  - 3.1.5 Auffinden anderer Gäste . . . 22
  - 3.1.6 Mobiltelefon als Schlüssel . . . 23
  - 3.1.7 Statistiken für den Betreiber . . . 23
- 3.2 Grundlegende Anforderungen an das System . . . 24
- 3.3 Anforderungen an die Funktionalität . . . 26
  - 3.3.1 Systemanwendungsfälle . . . 26
  - 3.3.2 Ausgewählte Geschäftsanwendungsfälle . . . 30

4 Entwurf und Realisierung 35
4.1 Dienste . . . 36
4.1.1 Mobile Dienste . . . 38
4.1.2 Globale Dienste . . . 40
4.1.3 Lokale Dienste . . . 40
4.1.4 Dienstbereitstellung . . . 41
4.1.5 Dienstnutzung . . . 47
4.2 Clubsystem . . . 54
4.2.1 Mitgliederbeobachtung . . . 56
4.2.2 Cluborganisation . . . 58
4.2.3 Interne Kommunikation . . . 63
4.2.4 Schließfächer . . . 65
4.3 Mobiltelefon . . . 66
4.3.1 Persönliche Organisation . . . 68
4.3.2 Informationsdarstellung und Userinteraktion . . . 69
4.4 Laborumgebung . . . 71
4.4.1 Implementierungstools . . . 71
4.4.2 Verwendete Hardware . . . 72
4.4.3 Bluetooth-Stack und JABWT Implementation . . . 74
4.4.4 Ergebnisse der Evaluierung . . . 76
4.5 Zusammenfassung . . . 79

5 Fazit und Ausblick 81

A Klassendiagramme 83

Literaturverzeichnis 89

1 Einleitung

1.1 Motivation

Durch die rapide Weiterentwicklung der Hardware und der technischen Möglichkeiten für eine schnelle Datenübertragung bei gleichzeitigem Verfall der Kosten wird das Mobiltelefon in der heutigen Zeit nicht mehr nur zur reinen Kommunikation genutzt. Moderne Geräte ermöglichen die Verarbeitung komplexer digitaler Medien und stellen dem Benutzer diverse mehr oder weniger umfangreiche Anwendungen zur Selbstorganisation und zum Entertainment zur Verfügung. Trotz kleiner Abmessungen wird zusätzliche Technik integriert, so dass der Nutzer nur noch ein einziges Gerät benötigt, um hochwertige Fotos in Druckqualität aufzunehmen, E-Mails zu lesen, Musik zu hören, Internetinhalte anzuschauen, mobile TV-Dienste zu nutzen und vieles mehr. Die große Leistungsfähigkeit solcher im Vergleich kleinen und leichten sog. Smartphones[1] bewirken einen Trend in Richtung allgegenwärtiger und universeller Benutzung.

Da sich das Handy längst bei der breiten Masse als ständiger Begleiter etabliert hat, ist die Idee nahe liegend, solche Geräte auch zur Unterstützung der Abläufe sowohl im Arbeitsleben, als auch in der Freizeit eines Benutzers heranzuziehen. Die Vision könnte sein, das gesamte „Werkzeug“ des täglichen Lebens in einem kleinen Gerät zu vereinen, das dann als Kommunikations- und Informationscenter, Geldbörse und Schlüssel dient. Gerade die Tatsache, dass sich der Benutzer zu unterschiedlichen Zeiten an unterschiedlichen Orten aufhält, eröffnet das Anbieten von situations- und positionsabhängigen Diensten, deren Sinnhaftigkeit erst in diesem Moment entsteht. Die Eignung und damit auch die Akzeptanz einzelner Dienste ist aber sicherlich nicht nur von der Person selbst, sondern zusätzlich vom jeweiligen Aufenthaltsort dieser abhängig.

Für Nutzer und Betreiber von Institutionen vielfältiger Art ergeben sich hier Möglichkeiten, den Aufenthalt und die Benutzung zu optimieren, indem herkömmliche Ausweiskarten durch die Verwendung von mobilen Endgeräten ersetzt werden.

[1] zu deutsch etwa „schlaues Telefon“, auch Mobile Digital Assistant (MDA)

1.2 Zielsetzung

In dieser Arbeit wird eine Plattform entworfen, auf deren Basis mobile Endgeräte drahtlos über Zugangspunkte in ein Gesamtsystem integriert werden und mit diesem interagieren. Dabei werden Interaktionen sowohl von den Benutzern der Geräte, als auch von einer zentralen Verwaltung des Systems initiiert. Um mobile Benutzer gezielt anzusprechen und die Güte von angebotenen Diensten zu erhöhen, soll dem System die Position des Benutzers fortlaufend bekannt sein. Der Schwerpunkt dieser Arbeit liegt dabei auf der Kommunikation zwischen Infrastruktur und mobilen Endgeräten und den in diesem Zusammenhang angebotenen Diensten der Plattform.

Geeignete Technologien für eine Umsetzung wurden bereits in einer Studienarbeit des Verfassers recherchiert und bewertet (Friedburg 2005). Da ein solches mobiles Endgerät für eine derartige Verwendung mehr oder weniger ständiger Begleiter des Benutzers ist und ihn nur in möglichst geringem Maße bei seinen sonstigen Aktivitäten beeinflussen darf, wird der Fokus auf sog. Smartphones gelegt, da diese bezüglich Abmessungen und Gewicht eine besondere Eignung mit sich bringen. Für die drahtlose Kommunikation ist die Verwendung der Bluetooth Technologie vorgesehen, da diese bereits heute in den meisten Fällen von der fokussierten Geräteklasse unterstützt wird und abzusehen ist, dass in mittelfristiger Zeit wohl sämtliche Mobiltelefone mit dieser Technologie ausgestattet sein werden. Als Ergebnis der Arbeit soll eine prototypische Implementation entstehen, anhand derer eine Aussage getroffen werden kann, inwieweit eine tatsächliche Eignung der verwendeten Technologien für die Praxis vorliegt.

1.3 Gliederung

In Kapitel 2 (Grundlagen) werden zunächst einige für das Verständnis innerhalb des weiteren Verlaufs der Arbeit wichtige Grundlagen erörtert.

Beispielhaft für die Verwendung der Plattform wird in Kapitel 3 (Analyse) der Betrieb eines Ferienclubs herangezogen, um die behandelte Problematik verständlicher und greifbarer zu gestalten und die Anforderungen an das System genauer zu spezifizieren.

Kapitel 4 (Entwurf und Realisierung) beschäftigt sich mit dem Design und der Umsetzung der Gesamtarchitektur sowie der einzelnen Komponenten. Abschließend werden die Konstellation der Laborumgebung und die Ergebnisse der Evaluierung dargestellt.

Eine abschließende, die Ergebnisse bewertende Zusammenfassung der Arbeit findet der Leser in Kapitel 5 (Fazit und Ausblick).

2 Grundlagen

Dieses Kapitel soll in die begrifflichen und in die technologischen Grundlagen einführen, die u.U. dem einen oder anderen Leser noch nicht ausreichend bekannt sind. Dabei werden Umfang und Detail nur so weit behandelt, wie zum Verständnis des weiteren Verlaufs dieser Arbeit nötig ist. Für eine tiefergreifende Auseinandersetzung mit den Thematiken sei an dieser Stelle an entsprechende Fachliteratur verwiesen.

2.1 Bluetooth

Bluetooth ist ein Industriestandard für die drahtlose Vernetzung von Geräten über kurze Distanz und bietet eine drahtlose Schnittstelle, über die sowohl mobile Kleingeräte wie Mobiltelefone und PDAs[1] als auch Computer und Peripheriegeräte miteinander kommunizieren können. Die Technik dient vorrangig zur Realisierung von Ad-hoc-Pikonetzen, die eine sehr geringe räumliche Ausdehnung von wenigen Metern haben. Bluetooth nutzt wie viele andere Standards auch den unregulierten Frequenzbereich um 2,4 GHz, der mit gewissen nationalen Einschränkungen weltweit zur Verfügung steht[2].

Die Entwicklung von Bluetooth begann 1994 bei Ericsson, als man nach einer Möglichkeit suchte, die Kabel zwischen Mobiltelefonen und Zusatzgeräten zu ersetzen. Zusammen mit anderen Industriepartnern gründete man 1998 die *Bluetooth SIG (Bluetooth Special Interest Group)*, um Bluetooth als de-facto-Standard zu etablieren. Um dies zu erreichen, setzte man sich das Ziel, einen Transceiver mit geringen Herstellungskosten, flexiblen Einsatzmöglichkeiten, niedrigem Energieverbrauch, Robustheit gegenüber Störungen und der Fähigkeit, Daten für multimediale Anwendungen zu übertragen, zu spezifizieren. Dem Konsortium sind mittlerweile mehr als 1000 Unternehmen angeschlossen und der Bluetooth-Standard ist inzwischen von der Arbeitsgruppe der IEEE für WPANs als IEEE 802.15.1 adaptiert worden.

Bluetooth unterstützt seit der Version 1.0 zwei Sprach- und Datenübertragungsdienste.

---

[1] Personal Digital Assistant

[2] das ISM-Band (**I**ndustrial, Scientific, and **M**edical Band) ist ein Frequenzbereich für Hochfrequenz-Sendegeräte in Industrie, Wissenschaft und Medizin, der nicht der staatlichen Regulierung unterliegt und lizenzfrei genutzt werden darf

- Einen synchronen, verbindungsorientierten Dienst, der symmetrische, leitungsvermittelte Sprachkanäle bietet. Die Nutzung von parallel maximal drei Kanälen mit einer Datenübertragungsgeschwindigkeit bis zu 64 kbit/s.
- Einen asynchronen, verbindungslosen Dienst, der paketvermittelte Übertragung anbietet. Es sind asymmetrische Verbindungen mit Datenraten bis zu 721 kbit/s downstream und 57,6 kbit/s upstream, als auch symmetrische Verbindungen mit je 432 kbit/s möglich.

Der aktuelle Bluetooth-Standard ist Version 1.2. Die meisten zur Zeit im Handel befindlichen Geräte bauen jedoch noch auf dem Standard 1.1 auf. Die Versionen 1.1 und 1.2 sind zueinander kompatibel, Neuerungen in Version 1.2 sind das Frequenzsprung-Verfahren *Adaptive Frequency Hopping*, das Interferenzen mit anderen Funktechniken nochmals reduzieren soll, sowie eine bessere Fehlererkennung. Es gibt keine Verbesserung der Übertragungsgeschwindigkeiten zwischen den beiden Versionen. Mit der Version 2.0 - auch mit EDR (Enhanced Data Rate) betitelt - sollen Daten in Zukunft etwa dreimal so schnell übertragen werden können, also mit rund 2,2 Mbit/s.

Neben diesen Datendiensten gibt es auch Mechanismen zur Leistungssteuerung der Endgeräte, um Energie zu sparen. Zu diesem Zweck werden für die Endgeräte mehrere Zustände definiert, welche den Energieverbrauch entsprechend drosseln. Außerdem stellt Bluetooth bestimmte Schutzmechanismen zur Verfügung, die aus Verschlüsselungs- und Authentifizierungsverfahren bestehen.

Die theoretischen Reichweiten für Bluetooth-Geräte sind in Tabelle 2.1 dargestellt. Mit einem modifizierten Bluetooth-USB-Adapter mit Richtfunkantenne soll es sogar möglich sein, ein Bluetooth-Handy bei Sichtkontakt noch aus etwa 1,6 km Entfernung anzusprechen (Wikipedia 2005).

Bluetooth Klasse	**Sendeleistung**	**Reichweite**
Klasse III	1 mW	10 m
Klasse II	10 mW	50 m
Klasse I	100 mW	100 m

Tabelle 2.1: Theoretische Übertragungsreichweiten von Bluetooth

Bluetooth-Geräte, die sich in gegenseitiger Reichweite befinden, können eine Kommunikationsverbindung aufbauen, ohne dass weitere Geräte oder eine zentrale Administration notwendig sind. Ein solches auch als *Piconet* bezeichnetes Netzwerk kann aus zwei bis maximal acht aktiv kommunizierenden Teilnehmern bestehen, wobei sich die Geräte die Bandbreite teilen. Ein ausgezeichnetes Gerät (meist das Gerät, welches die Verbindungsaufnahme initiiert), der sog. *Master*, regelt den Zugriff auf die Funkschnittstelle. Die weiteren Geräte *(Slaves)* sind nicht direkt, sondern über den Master miteinander verbunden. Im Fall

einer Überlappung solcher Piconets ist es einem Gerät möglich, Teilnehmer mehrerer Netze zu sein und man spricht von einem *Scatternet*. Abbildung 2.1 zeigt solche Szenarien. Bis zu zehn Piconets können sich ohne gegenseitige Störungen überlagern, wobei allerdings die Leistungsfähigkeit der vorhandenen Netze sinkt (Schiller 2000).

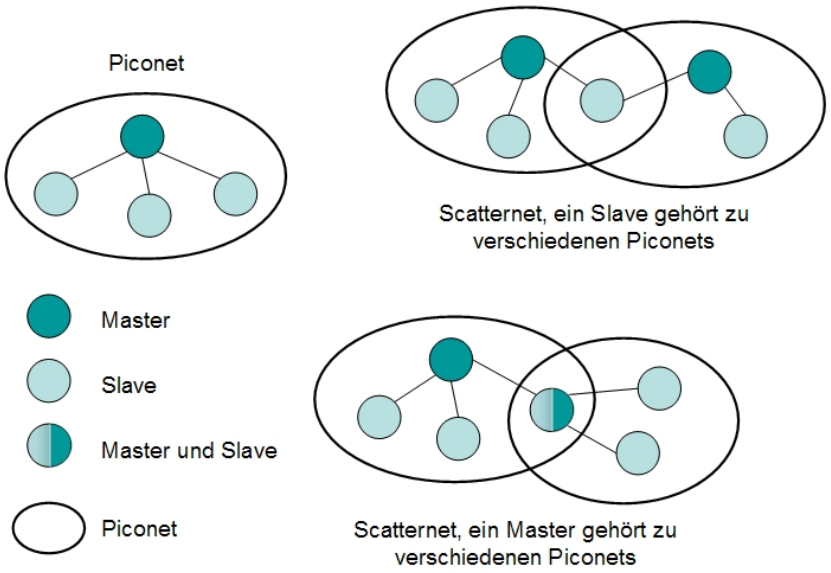

Abbildung 2.1: Beispiele für Piconets und Scatternets

Heutzutage ist der Einsatz von Bluetooth bereits enorm verbreitet. Die meisten modernen Mobiltelefone können mittels Bluetooth mit entsprechenden Freisprecheinrichtungen und Headsets, mobilen und stationären Computern, PDAs und anderen Mobiltelefonen verbunden werden, um Daten auszutauschen.

2.1.1 Bluetooth-Stack-Architektur

Dieser und die folgenden Abschnitte bieten einen kurzen Überblick über den Bluetooth-Protokoll-Stack. Der Bluetooth-Protokoll-Stack kann allgemein in zwei Komponenten eingeteilt werden: den Bluetooth Host und den Bluetooth Controller (oder auch Bluetooth Funkmodul). Das Host Controller Interface (HCI) stellt eine standardisierte Schnittstelle zwischen Host und Controller bereit. Diese Klassifikation wird in der Abbildung 2.2 illustriert.
Der Bluetooth Host wird auch als Upper-Layer-Stack bezeichnet und ist üblicherweise als Software implementiert und in die System-Software oder das Betriebssystem integriert. Bluetooth Profile setzen auf diese Protokolle auf. Beim Bluetooth Controller handelt es sich normalerweise um ein Hardwaremodul, das einen festen Bestandteil des Systems bildet oder

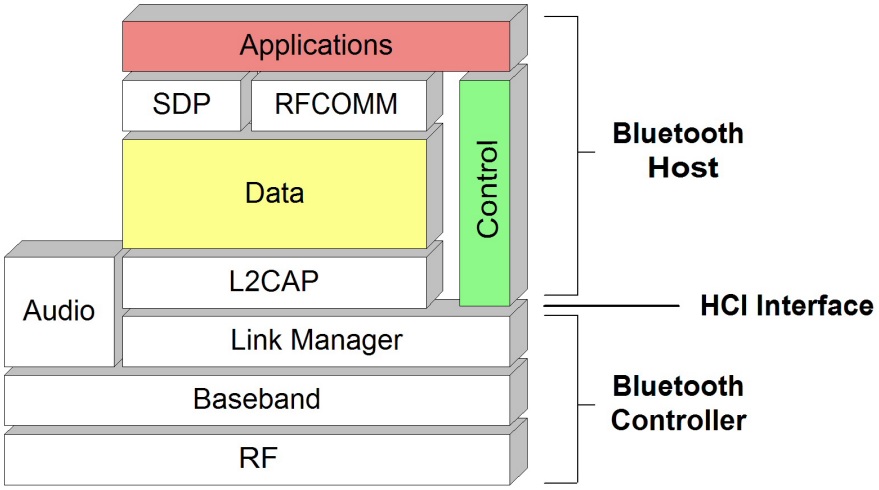

Abbildung 2.2: Bluetooth Host und Controller Klassifikation

über USB, PCMCIA, UART, etc. mit dem System verbunden wird. In einigen Geräten sind sowohl Controller als auch Host fest integriert und das HCI wird nicht für eine Verbindung dieser benutzt. Ein Beispiel hierfür sind die mittlerweile weit verbreiteten Headsets, die als Freisprecheinrichtung mittels Bluetooth an ein Mobiltelefon gekoppelt werden.
Einige der in Abbildung 2.2 dargestellten Bestandteile des Bluetooth-Stacks werden im folgenden Abschnitt näher erläutert.

2.1.2 Bluetooth-Protokolle

In der Bluetooth Spezifikation sind mehrere Protokolle definiert, Abbildung 2.3 zeigt die gebräuchlichsten als Blockdiagramm des Bluetooth Protokoll-Stacks. Die schattierten Blöcke repräsentieren die Protokolle, die von den Java APIs für *Bluetooth Wireless Technology (JABWT)* adressiert werden. Der Protokoll-Stack baut sich aus Bluetooth-spezfischen Protokollen, wie z.B. dem Service Discovery Protokoll (SDP), und anderen, adaptierten Protokollen, wie z.B. dem Object Exchange Protokoll (OBEX) auf. Die einzelnen Schichten und Protokolle sollen im Folgenden kurz beschrieben werden.

Bluetooth radio ist die unterste in der Bluetooth-Spezifikation definierte Schicht und bestimmt die Anforderungen des Bluetooth Sende- und Empfangsgerätes für den Betrieb auf dem 2,4-GHz ISM Band bezüglich Modulationen und Sendeleistungen. Diese Schicht entspricht im Wesentlichen der Bitübertragungsschicht des OSI-Modells[3].

[3]Das OSI-Modell (engl. Open Systems Interconnection Reference Model) ist ein offenes Schichtenmodell für

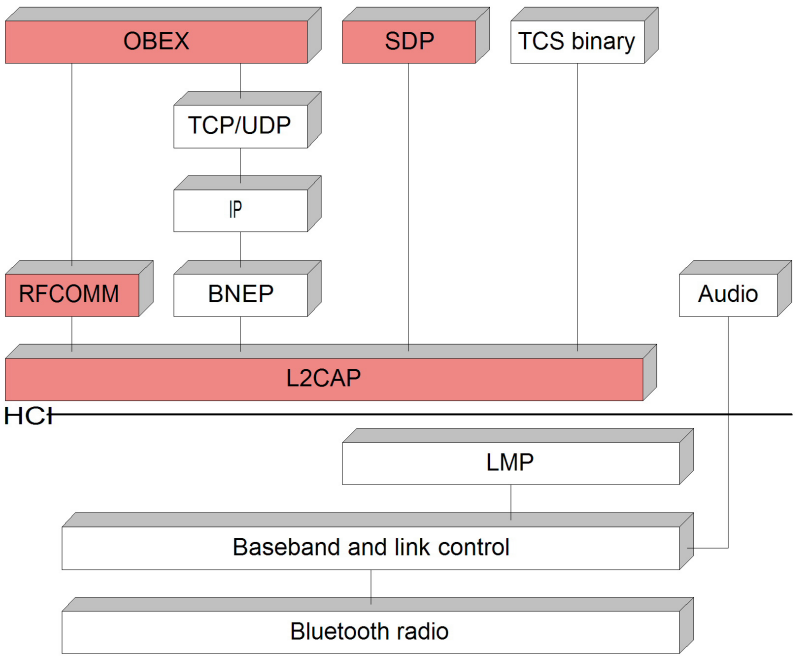

Abbildung 2.3: Bluetooth-Protokoll-Stack

Die Schicht *baseband and link control* ermöglicht die physikalische Funkverbindung zwischen Bluetooth-Einheiten. Sie ist zuständig für die Abwicklung der Funkkanäle und die zeitliche Koordinierung, setzt für die Übertragung die Pakete zusammen und synchronisiert sie. Dabei werden zwei Arten von physikalischen Verbindungen unterstützt: ein synchroner, verbindungsorientierter Anschluss (SCO, synchronous connection oriented) und ein asynchroner, verbindungsloser Anschluss (ACL, asynchronous connectionless)

Audio stellt keine Schicht des Protokoll-Stacks im eigentlichen Sinn dar, sondern wird bei der Bluetooth-Kommunikation eigenständig behandelt. Reine Audiodaten werden typischerweise über eine SCO-Verbindung direkt zu und von der *Baseband*-Schicht geleitet.

Das *Link Manager Protocol (LMP)* ist verantwortlich für Verbindungsaufbau und der Konfiguration einer solchen Verbindung zwischen Bluetooth-Geräten und handhabt Sicherheitsaspekte wie Authentisierung und Verschlüsselung.

Das *Logical Link Control and Adaptation Protocol (L2CAP)* schirmt in der Stack-Architektur weiter oben liegende Protokolle von den Details der unteren Protokolle ab und ermöglicht das Multiplexen zwischen verschiedenen logischen Verbindungen, die durch höher angesiedelte Schichten initiiert wurden.

die Organisation von Kommunikationstechnik. Es wird seit 1979 entwickelt und ist von der ISO standardisiert worden.

SDP stellt Applikationen ein Hilfsmittel bereit, um Bluetooth Services und deren Charakteristiken zu erkunden. Im Gegensatz zu einer üblichen LAN[4]-Verbindung, bei der zuerst eine Verbindung mit dem Netzwerk und dann das Auffinden von Geräten stattfindet, werden in einer Bluetooth-Umgebung zuerst die Geräte und erst dann die Dienste gefunden. Zusätzlich gestaltet sich das Angebot an Diensten dynamisch, wenn sich die Geräte in Bewegung befinden und sich die Umgebung dadurch verändert. *SDP* setzt auf *L2CAP* auf.

Serielle Ports sind eine der am weitesten verbreiteten Schnittstellen bei Computern und Kommunikationsgeräten. Mit dem *RFCOMM* Protokoll können solche seriellen Verbindungen über *L2CAP* emuliert werden. Mit *RFCOMM* sind parallele gleichzeitige Verbindungen zu einem Gerät und gleichzeitige Verbindungen zu mehreren Geräten möglich.

Damit Bluetooth-Geräte in einem gebildeten Netzwerk interagieren und Informationen austauschen können, muss ein allgemeines Paketformat definiert werden, um die unterschiedlichen Netzwerkprotokolle zu kapseln. Das *Bluetooth Network Encapsulation Protocol (BNEP)* kapselt Pakete diverser Netzwerkprotokolle und transportiert diese direkt über *L2CAP*. *BNEP* ist ein optionales Protokoll, das zeitlich nach der Bluetooth-Spezifikation Version 1.1 entwickelt wurde, aber auf dieser Version basiert.

Die Telephony Control Protocol Spezification (TCS binary) definiert die Kontrollsignale für den Aufbau, Abbau und Verlauf einer Verbindung bei Sprach- und Datenübertragungen zwischen Bluetooth-Geräten und setzt auf *L2CAP* auf.

Adaptierte Protokolle, wie *OBEX* und das *Internet Protocol (IP)*, basieren auf einem der bereits erwähnten Protokolle. So setzt *OBEX* auf *RFCOMM* und *IP* auf *BNEP* auf.

Die Anzahl der Bluetooth-Protokolle unterliegt zur Zeit einem ständigen Wachstum. Viele neue Protokolle werden durch die Bluetooth SIG spezifiziert, wobei die meisten von ihnen auf das *L2CAP* aufsetzen.

2.2 J2ME

Die *Java 2 Platform, Micro Edition*, abgekürzt J2ME, ist eine Umsetzung der Programmiersprache Java für so genannte „embedded consumer products“ wie etwa Mobiltelefone, PDAs, Pager, Set-Top Boxen oder Entertainment- und Navigationssysteme im Automobilbereich. Dabei ist J2ME eine von drei Editionen der Java 2 Plattform. Bei den anderen beiden handelt es sich um die *Enterprise Edition* (J2EE) für den hauptsächlichen Einsatz auf Servern und die *Standard Edition* (J2SE) mit der Zielgrupper der Desktop-Computer. Eine verwandte Technologie ist *Java Cards*, durch deren Spezifikation die Lauffähigkeit von Java-Programmen auf Smart-Cards und ähnlichen Geräten ermöglicht wird, deren verfügbarer

[4] **L**ocal **A**rea **N**etwork

Speicher noch limitierter ist, als bei einem Low End-Mobiltelefon. Diese Gruppierungen ermöglichen einen maßgeschneiderten Einsatz der Java-Technologie in den verschiedenen Sparten der heutzutage weit reichenden und breit gefächerten EDV-Industrie. In Abbildung 2.4 sind die Editionen der Java 2 Plattform und deren Zielmärkte im Überblick dargestellt.

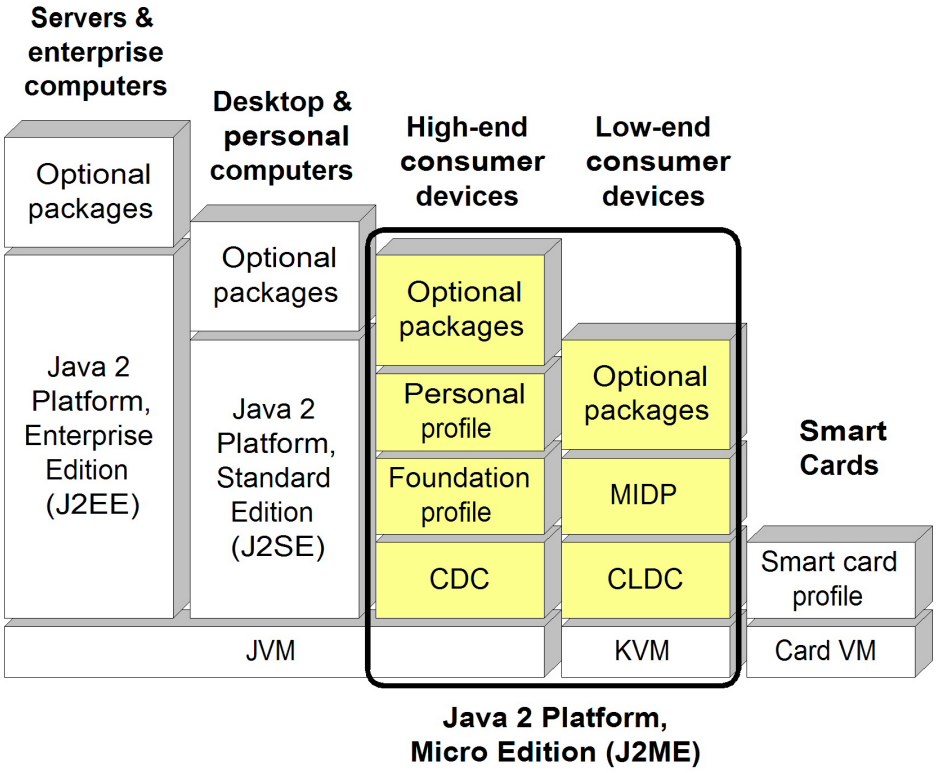

Abbildung 2.4: Java 2 Plattformen

Die J2ME Plattform ermöglicht es, die Leistung und Vorteile der Java-Technologie für eingebettete Geräte und allgemeine Endverbraucher-Geräte zu nutzen, wie z.B. die Portabilität von entwickeltem Code und Objektorientierte Programmierung. Eine der Hauptintentionen der J2ME ist dabei, Geräten den dynamischen Download von Applikationen zu ermöglichen. Die von der J2ME adressierten Gerätetypen weisen ein breites Spektrum bezüglich verfügbarem Speicher, Rechen- und damit auch Ausführungsgeschwindigkeit und Möglichkeiten der Ein- und Ausgabe auf. Um einer solchen Vielfalt in möglichst vielen Fällen gezielt gerecht zu werden, definiert die J2ME Architektur sog. *Konfigurationen*, *Profile* und *optionale Pakete*, die eine modulare Anpassung der Architektur ermöglichen. Der grundsätzliche Aufbau der J2ME Architektur ist in Abbildung 2.5 illustriert, die einzelnen Schichten werden in den folgenden Abschnitten betrachtet.

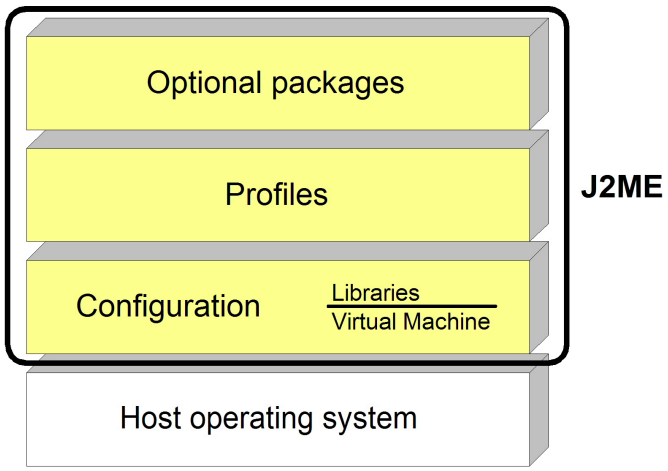

Abbildung 2.5: Komponenten der J2ME Architektur

2.2.1 Konfigurationen

Eine JVM (Java Virtual Machine) interpretiert den Bytecode, der entsteht, wenn ein Java-Programm kompiliert wird. Ein solches kompiliertes Java-Programm kann auf jedem Gerät ausgeführt werden, das eine geeignete virtuelle Maschine und den benötigten Umfang an Java-Klassenbibliotheken zur Verfügung stellt.
Konfigurationen im Sinne der J2ME bestehen aus genau diesen zwei genannten Bestandteilen, einer Java-fähigen virtuellen Maschine und einem bestimmten minimalen Umfang an Klassenbibliotheken. Eine Konfiguration definiert das Minimum an Funktionalität für eine bestimmte Kategorie oder Gruppierung von Geräten. Zur Zeit existieren zwei solcher Konfigurationen innerhalb der J2ME:

- *Connected, Limited Device Configuration*
  Die CLDC fokussiert low-end Geräte und ist die vom gebotenen Funktionsumfang kleinere der zwei Konfigurationen. Typische Geräte dieser Gruppe, wie z.B. Mobiltelefone, Pager und PDAs, verfügen meist nur über langsamere Prozessoren und sehr begrenzten Speicherumfang, werden mit Batterien betrieben und initiieren nur zeitweilige Netwerkverbindungen. Eine CLDC-Implementation inkludiert normalerweise eine sog. *Kilobyte Virtual Machine (KVM)*. Die KVM hat ihren Namen aufgrund der nur geringen Beanspruchung von Ressourcen im Bereich von Kilobytes und ist speziell für Geräte mit eingeschränktem Speicher entworfen. CLDC ist die Konfiguration, die normalerweise auf Endgeräten eingesetzt wird, die der Zielgruppe dieser Arbeit entsprechen.
- *Connected Device Configuration*
  Zielgruppe der CDC sind High End-Geräte, wie z.B. Set-Top Boxen. Solche Geräte

verfügen über mehr Speicherressourcen und schnellere Prozessoren, reichen aber diesbezüglich nicht an einen typischen Desktop Computer heran. Die CDC nutzt eine virtuelle Maschine, die die Spezifikation einer Java Virtual Machine vollständig erfüllt.

2.2.2 Profile

Konfigurationen bilden alleine keine komplette Lösung. Profile der J2ME fügen die Funktionalitäten und die APIs hinzu, die benötigt werden, um eine komplette und vollständig funktionierende Laufzeitumgebung für eine Klasse von Geräten zu erhalten. Geräte einer solchen Klasse haben typischerweise die gleichen oder zumindest ähnliche Benutzerschnittstellen (wie Bildschirm und Eingabegeräte) und Verfahren, Netzwerkverbindungen herzustellen und Daten zu speichern. Dabei kann ein einzelnes Gerät mehrere unterschiedliche Profile unterstützen.

Im Zusammenhang mit dieser Arbeit ist vor allem das *Mobile Information Device Profile (MIDP)* von Bedeutung, welches kombiniert mit der CLDC Applikationen auf mobilen Endgeräten Kernfunktionalitäten bezüglich Netzwerkfähigkeit, Persistierung von Daten und der Realisierung von Benutzerschnittstellen offeriert. Diese Kombination von Konfiguration und Profil steht auf den meisten aktuellen, Java-fähigen Smartphones zur Verfügung.
MIDP-Applikationen werden auch *MIDlets* genannt. MIDlet ist eine innerhalb des MIDP definierte Klasse, die als Superklasse für alle MIDP-Applikationen herangezogen wird. Der Name MIDlet lehnt sich an den Begriff *Applet* an, mit dem Java-Programme bezeichnet werden, die als Bestandteil einer Web-Seite innerhalb eines Internet-Browsers ausgeführt werden.

2.2.3 Optionale Pakete

Viele die J2ME unterstützende Geräte verfügen über zusätzliche Technologien, wie z.B. Bluetooth, Multimedia, Wireless Messaging oder Datenbank-Konnektivität. Optionale Pakete dienen dazu, diese Technologien mit Hilfe von standardisierten JAVA APIs zu nutzen. Ein solches „optional package“ wird also immer in Verbindung mit einer Konfiguration oder eines Profils verwendet. Es erweitert die Laufzeitumgebung, um Ressourcen unter Java nutzbar zu machen, die nicht universell genug sind, um als ein Bestandteil eines Profils definiert zu werden, oder deren Nutzung übergreifend auf unterschiedlichen Profilen nötig ist.

Zusätzlich zu den offiziellen Konfigurationen, Profilen und optionalen Paketen besteht für die Hersteller von Geräten die Möglichkeit, zusätzliche Java-Klassen zu definieren, um spezifische Möglichkeiten des Gerätes nutzbar zu machen. Solche Klassen werden *licensee open classes (LOCs)* genannt und sind für alle Entwickler vefügbar. Im Gegensatz dazu sind sog. *licensee closed classes (LCCs)* nicht für die Allgemeinheit nutzbar und unterliegen im

Gebrauch dem Hersteller des Gerätes. Da solche durch den Hersteller bereitgestellte Klassen keiner Standardisierung unterliegen, sind Applikationen, die solche Klassen verwenden, nicht oder nur schwierig zwischen Endgeräten portierbar, auch wenn diese dieselben Konfigurationen und Profile unterstützen.

2.3 Java Community Process (JCP)

Der Java Community Process ist eine offene Organisation von Java-Technologie-Entwicklern und Java-Lizenznehmern. Innerhalb dieser Organisation werden definiert durch einen formalen Prozess Spezifikationen für neue Java-Technologien, Referenz-Implementationen und Kompatibilitäts-Testpakete entwickelt, geprüft und verabschiedet. Die Erweiterungen werden *Java Specification Request (JSR)* genannt und einfach durchnummeriert.
Entwickelt hat sich der JCP aus einer Initiative von Sun, die aus den gescheiterten Standardisierungsbemühungen seitens internationalen Organisationen resultierte. Der JCP wird heute jedoch von Vertretern verschiedenster Organisationen der Java-Gemeinschaft gestaltet und gesteuert. Da Java eng mit dem Internet und mobilen Kommunikationstechnologien verbunden ist, arbeitet der JCP oft auch mit den entsprechenden Gremien wie dem W3C[5] und der Open Mobile Alliance[6] zusammen.

Die Existenz von Standards spielt eine entscheidende Rolle, um entwickelte Applikationen in einem hohen Maß zwischen einer großen Anzahl von Geräten portierbar zu machen und wieder verwendbare Komponenten zu entwickeln. Gerade bei mobilen Endgeräten besteht meist keine einfache Möglichkeit, Konfigurationen und Ergänzungen am Betriebssystem vorzunehmen, um eine Lauffähigkeit einer nicht von vornherein geeigneten Applikation zu ermöglichen.

Die mobilen Endgeräte, die in dieser Arbeit als Zielgruppe in Frage kommen, müssen J2ME-Standards bezüglich Konfiguration, Profil und dem optionalen Paket für die Bluetooth-Technologie genüge tun. In allen drei Punkten existieren durch die JCP etablierte Spezifikationen. CLDC in der Version 1.1 wird definiert von JSR-139, MIDP in der Version 2.0 von JSR-118 und die JABWT von JSR-82.

[5] World Wide Web Consortium, eine Vereinigung zur Weiterentwicklung von Standards für das WWW
[6] die Open Mobile Alliance (OMA) ist ein Industrieforum für die Entwicklung von Spezifikationen marktgetriebener, interoperabler End-zu-End-Dienste im Mobilfunk

3 Analyse

In diesem Kapitel sollen sowohl grundlegende Anforderungen, als auch die spezifischen Anforderungen an die Funktionalität, denen die zu entwickelnde Plattform genügen soll, zusammengestellt und diskutiert werden. In Abschnitt 3.1 wird der Betrieb eines Ferienclubs als Beispiel für die Problematik herangezogen, um die grundlegende Vision und damit die Anforderungen an den Funktionsumfang im Gesamtkontext darzulegen und anschließend diese Anforderungen kategorisieren zu können. Dieses Beispiel findet für die Entwicklung der Plattform im gesamten weiteren Verlauf der Arbeit Verwendung. Hierbei besteht aber natürlich eine Portierbarkeit auf andere, vom Grundsatz her ähnliche Szenarien. Die Beispielszenarien dienten bereits als Grundlage für eine Studienarbeit des Verfassers (Friedburg 2005) und sind an dieser Stelle zur besseren Verständlichkeit nochmals vollständig aufgeführt.

3.1 Beispielszenario Ferienclub

Wir betrachten im folgenden Beispielszenario einen fiktiven Ferienclub, der seinen Gästen ein breit gefächertes Freizeit- und Entertainmentangebot offeriert. Einige Angebote sind dabei ohne weitere Kosten für den Gast nutzbar, während für andere zusätzliche Gebühren erhoben werden.

Die Gäste nutzen während ihres Aufenthaltes ihr Handy als „intelligenten“ Clubausweis, d.h. neben der Identifizierung unterstützt dieses sie dabei, ihren Aufenthalt im Ferienclub zu gestalten. Ein Ziel dabei ist es, sowohl Organisation als auch Ablauf des Aufenthaltes für die Gäste zu vereinfachen, indem sie ihr Mobiltelefon als zentrales Instrument benutzen. Auf der anderen Seite profitiert auch der Betreiber des Ferienclubs, indem ihm ermöglicht wird, die Auslastung seiner Angebote zu optimieren, den Ablauf zu automatisieren und zusätzliche Informationen zur Verbesserung des Betriebes zu sammeln.

Das Szenario des Ferienclubs wurde bereits in ähnlicher Art im Kontext mobiler Endgeräte in den Diplomarbeiten von Lübke (2004), Seite 10ff und Bresch (2004), Seite 13ff betrachtet, allerdings hinsichtlich der Thematiken „Sicherheitsarchitektur zur Nutzung von Diensten“ bzw. „Optimierte Anzeige von Daten mittels JavaServerFaces“.

3.1.1 Ankunft eines Gastes

Beim Einchecken des Gastes wird auf seinem Mobiltelefon die benötigte Software installiert, um die erforderlichen Funktionalitäten für einen „intelligenten“ Clubausweis zur Verfügung zu stellen. Der Gast wurde bereits bei der Buchung seines Aufenthaltes im Ferienclub darüber informiert, welche technischen Anforderungen an das Mobiltelefon gestellt werden. Das Gerät des Gastes erfüllt diese Anforderungen. Wäre dies nicht der Fall gewesen oder würde er sein eigenes Gerät nicht benutzen wollen, hätte er vom Betreiber des Ferienclubs ein gegen Verschmutzung und Feuchtigkeit resistentes Gerät für die Dauer seines Aufenthaltes zur Verfügung gestellt bekommen.

Der Gast erhält eine Einweisung in die neuen den Ferienclub betreffenden Funktionalitäten des Mobiltelefons. Es soll ihm helfen, seine Teilnahme an angebotenen Aktivitäten zu organisieren und evtl. zusätzlich entstehende Kosten zu begleichen. Weiterhin wird ihm mitgeteilt, dass sein Mobiltelefon gleichzeitig als Schlüssel fungiert, sowohl für das durch den Gast bewohnte Zimmer als auch beim Zutritt in Bereiche des Ferienclubs, in denen zusätzliche Kosten entstehen. Auch können Umkleideschränke mittels des Mobiltelefons verschlossen und entriegelt werden.

3.1.2 Profilierung des Gastes

Um den Gast möglichst zielgerichtet mit Veranstaltungsangeboten über sein Mobiltelefon zu informieren, wird ihm empfohlen, seine Interessen und einige persönliche Eigenschaften in einem Profil zu hinterlegen. Die vom Ferienclub auf dem Mobiltelefon zur Verfügung gestellte Applikation ermöglicht dem Gast das Anlegen eines solchen Profils, wobei er jederzeit die Möglichkeit hat, dieses nach seinen Wünschen zu ergänzen oder zu aktualisieren.

Zusätzlich hat der Gast die Möglichkeit, sich nicht nur aktiv über Veranstaltungen zu informieren, sondern auch automatisch Veranstaltungsvorschläge auf sein Mobiltelefon übermittelt zu bekommen, ohne diese zu diesem Zeitpunkt explizit angefordert zu haben. In diesem Zusammenhang wird dem Gast empfohlen, sein Einverständnis zu einem oder allen folgenden Punkten zu erklären, um so eine weitere Verbesserung der Güte der ihm vorgeschlagenen Angebote zu erreichen.

- Der Betreiber darf die Position des Mobiltelefons und damit den Aufenthaltsort des Gastes innerhalb des Ferienclubs bestimmen und ggf. an andere weitergeben. Durch die Positionsbestimmung ergeben sich auch Möglichkeiten für die Teilnahme an weiteren Dienstangeboten.
- Die Anwendung auf dem Mobiltelefon darf auf dessen Kalender zugreifen, um Termine auszulesen und dort einzutragen. Die verwendeten Informationen beschränken sich

darauf festzustellen, ob in einem bestimmten Zeitintervall bereits Termine vorhanden sind.

Sämtliche Einwilligungen im Kontext der automatischen Unterbreitung von Veranstaltungsvorschlägen können vom Gast mittels der Anwendung auf dem Mobiltelefon zurückgezogen bzw. temporär außer Kraft gesetzt werden.

3.1.3 Veranstaltungskalender

Nach Installation und Initialisierung der Applikation auf dem Mobiltelefon kann der Gast, solange er sich im Zugangsbereich des lokalen Netzwerkes des Ferienclubs befindet, das geplante Veranstaltungsangebot abrufen.

Um die Übersicht zu behalten, hat er die Möglichkeit, das Angebot nach bestimmten Kriterien zu filtern, kennzeichnen zu lassen oder zu sortieren. Neben Kriterien wie z.B. Veranstaltungszeitpunkt sind auch auf die Person des Gastes abgestimmte Kategorisierungen sinnvoll, die in Abhängigkeit mit den Angaben und Einwilligungen des Gastes (siehe 3.1.2) stehen. Diese werden im Abschnitt 3.1.3.1 erläutert.

Des Weiteren ist es dem Gast möglich, detailliertere Informationen zu den einzelnen Veranstaltungen abzurufen und eine Veranstaltung zu buchen, also eine verbindliche Zusage bezüglich seiner Teilnahme abzugeben. Unter der Voraussetzung, dass das Einverständnis des Gastes vorliegt, auf den Kalender seines Mobiltelefons zuzugreifen, wird dort die Veranstaltung als Termin hinterlegt. Für den Fall, dass eine Teilnahme an einer Veranstaltung dem Gast zusätzliche Kosten verursacht, wird dieser darauf hingewiesen. Um den Vorgang der Teilnahmeerklärung abzuschließen begleicht er die Kosten mit Hilfe eines Zahlungsverfahrens mittels seines Mobiltelefons oder er bestätigt die Übernahme der Kosten, so dass diese auf eine Rechnung gesetzt werden, die der Gast bei seiner Abreise oder Verlassen des Ferienclubs begleicht. Bezüglich ePayment sei an dieser Stelle auf die Studienarbeit von Szensny (2005) verwiesen. Einige Veranstaltungen benötigen keine Zusage, weil sie weder dem Gast weitere Kosten verursachen, noch über eine begrenzte oder minimal erforderliche Teilnehmeranzahl verfügen. Auch die Zeiten dieser Veranstaltungen kann der Gast selektiv in seinen Terminkalender eintragen lassen.

Im Veranstaltungskalender erhält der Gast auch Einsicht in die Veranstaltungen, zu denen er bereits eine Zusage abgegeben hat. Hier erhält er ggf. die Möglichkeit, seine Teilnahme an einer Veranstaltung zurückzuziehen.

Der Kalender beinhaltet auch erst grob geplante Veranstaltungen, deren genauer Zeitpunkt noch nicht feststeht oder die mit einer gewissen Spontanität stattfinden, z.B. weil sie wetterabhängig sind. Für diese kann der Gast festlegen, dass er automatisch informiert wird, sobald die Durchführung der Veranstaltung konkretisiert wird.

3.1.3.1 Persönliche Kategorisierung

Je konkreter und genauer der Gast sein Profil definiert hat, desto höher ist die Wahrscheinlichkeit, ihm nur für ihn sinnvolle und interessante Vorschläge anzubieten. Die Güte eines Vorschlages kann durch Informationen bezüglich seiner Position innerhalb des Ferienclubs und seiner Kalendereinträge in seinem Mobiltelefon positiv beeinflusst werden. Auch seitens des Betreibers des Ferienclubs ergibt sich der Vorteil, dass er gezielt geeignete Teilnehmer für Veranstaltungen akquirieren kann. Es sind also nachfolgende Ursachen denkbar, warum einem Gast gerade ein bestimmter Veranstaltungsvorschlag unterbreitet wird:

- Die Art der Veranstaltung verfügt über ein hohes Matching mit den Interessen des Gastes. Je genauer die Vorlieben und Ansprüche des Gastes getroffen werden, desto wahrscheinlicher wird seine Teilnahme sein. Auch können hier persönliche Eigenschaften des Gastes von Bedeutung sein. Werden z.B. für einen Tanzkurs nur noch männliche Teilnehmer gesucht, ist es sinnlos, die Veranstaltung einem weiblichen Gast vorzuschlagen.
- Die Veranstaltung passt in den Zeitplan des Gastes, d.h. es besteht keine Kollision des Zeitfensters der Veranstaltung mit im Kalender des Gastes eingetragenen Terminen. Handelt es sich bei dem bereits eingetragenen Termin auch um eine Veranstaltung des Ferienclubs, könnte ein Vorschlag mit Zeitüberschneidung nur dann unterbreitet werden, wenn er aus Sicht des Gastes über eine bessere Eignung verfügt.
- Es besteht eine günstige räumliche Nähe des Standortes des Gastes zum Veranstaltungsort. Bei einer sehr kurzfristigen Angebotsunterbreitung, z.B. wenn ein andere Gast kurz vor Beginn seine Teilnahme an einer Veranstaltung absagt, ist es bei einer sehr weitläufigen Clubanlage sinnvoll, einen Ersatzteilnehmer zu finden, der sich so dicht am Veranstaltungsort aufhält, dass er diesen noch rechtzeitig erreichen kann.
- Beliebige Kombinationen der bisher aufgeführten Punkte sind denkbar. Im Idealfall fällt eine Veranstaltung in alle drei Kategorien.

3.1.3.2 Spontane Veranstaltungsvorschläge

Solange die Anwendung auf dem Mobiltelefon des Gastes so konfiguriert ist, dass diesem bei Bedarf Veranstaltungsvorschläge zugesandt werden (siehe 3.1.2), kann er jederzeit einen solchen Vorschlag erhalten. Der Gast ist also nicht dazu gezwungen, aktiv nach für ihn geeigneten neuen Angeboten im Veranstaltungskalender zu suchen, da er automatisch informiert wird.

Hier spielt die persönliche Kategorisierung der Veranstaltungen (siehe 3.1.3.1) eine wichtige Rolle. So kann verhindert werden, dass der Gast eine Flut von Vorschlägen erhält, denen er

schon grundsätzlich nicht geneigt ist. Er bleibt vor vielleicht störenden Benachrichtigungen mit für ihn wahrscheinlich wertlosem Informationsinhalt verschont. Durch die Gestaltung seines Interessensprofils kann er selbst den Umfang der Benachrichtigungen beeinflussen.

Erhält der Gast einen Vorschlag, kann er diesen ablehnen, seine Teilnahme zusagen oder aber grundsätzliches Interesse bekunden ohne zu diesem Zeitpunkt eine Zusage erteilen zu wollen, indem er vorläufig nicht auf den Vorschlag reagiert. Im letzteren Fall wird der Veranstaltungsvorschlag in einer Favoritenliste des Gastes vorgehalten, so dass er zu einem späteren Zeitpunkt leicht eine Buchung der Veranstaltung vornehmen kann. Sollten im Laufe der Zeit alle verfügbaren Teilnehmerplätze der Veranstaltung belegt worden sein, wird der Gast darüber informiert und die Veranstaltung aus seiner Favoritenliste entfernt.

3.1.4 Navigation innerhalb des Ferienclubs

Solange die wenn auch vielleicht nur ungefähre geografische Position des Gastes innerhalb des Ferienclubgeländes bestimmt werden kann, ist es möglich, ihm gezielt Hinweise und Hilfestellungen zu geben, um zu einer anderen Position zu gelangen.

Der Gast kann einen aus einer Liste der relevanten Orte auf dem Gelände auswählen und sich eine Wegbeschreibung anzeigen lassen, wie er auf optimale Weise diesen von seinem Standort aus erreichen kann. Um zu einer Veranstaltung zu gelangen, für die der Gast als Teilnehmer registriert ist bzw. die in seinem Mobiltelefonkalender als Termin eingetragen ist, kann jederzeit die Entfernung des Aufenthaltsortes des Gastes zum Veranstaltungsort ermittelt werden. So kann er zu genau dem Zeitpunkt eine Erinnerung an den Beginn der Veranstaltung erhalten, dass er den Veranstaltungsort noch pünktlich erreicht, wenn er sich in Kürze auf den Weg macht.

3.1.5 Auffinden anderer Gäste

Durch die Bestimmung der geografischen Position von Personen auf der Clubanlage wird es dem Gast ermöglicht, den Aufenthaltsort anderer Gäste zu ermitteln, sofern von diesen eine Einwilligung vorliegt.

Zum einen können dies ihm bekannte Personen sein, mit denen er zusammen seinen Urlaub verbringt, wie z.B. Familienmitglieder oder Freunde. Da für die Lokalisierung der Gäste kein aktives Eingreifen ihrerseits erforderlich ist, werden diese zum einen nicht gestört und können zum anderen auch geortet werden, wenn sie gerade nicht in der Lage sind, ihr Mobiltelefon zu bedienen. Eltern können sich so z.B. jederzeit und beliebig oft über den Verbleib ihrer Kinder informieren, ohne dass diese dem direkten Gefühl der Kontrolle ausgesetzt sind, wie es beispielsweise bei wiederholten Telefonanrufen der Fall wäre.

Zum anderen wird dem Gast als Service angeboten, andere ihm noch fremde Gäste im Ferienclub zu finden und kennen zu lernen, wenn er sich dort z.B. alleine aufhält und Gleichgesinnte für gemeinsame Aktivitäten sucht. Anhand seines definierten Interessen- und Eigenschaftsprofiles (siehe 3.1.2) kann er nicht nur an Gäste mit ähnlichen Profilen vermittelt werden, sondern diese dann auch gezielt zur Kontaktaufnahme aufsuchen. Ein ähnliches Szenario wurde von Babic (2003) in einer Diplomarbeit über die Entwicklung einer profilverarbeitenden ubiquitären Anwendung verwendet, allerdings im Kontext einer Ad-hoc-Vernetzung auf kurze Distanz.

3.1.6 Mobiltelefon als Schlüssel

Der Gast kann sein Mobiltelefon innerhalb der Ferienclubanlage als Schlüssel benutzen. Ähnlich der in bereits vielen Hotels üblichen Schlüsselkarten öffnet er damit seine Zimmertür, allerdings berührungslos. Bei der Installation der Anwendung auf dem Mobiltelefon wurde diese so konfiguriert, dass ihre Codierung nur die Tür des Zimmers öffnen kann, das der Gast bewohnt. Er löst die Funktion zur Türöffnung aus und wenn sich das Mobiltelefon nahe genug an dem Öffnungsmechanismus befindet, um mit diesem drahtlos in Kontakt zu treten, findet eine zeitweilige Entriegelung der Tür statt. Verlässt der Gast seine Räumlichkeiten, befindet sich die Eingangstür automatisch in einem verriegelten Zustand.

Auf ähnliche Weise erhält der Gast Zutritt oder Zugriff zu Bereichen des Ferienclubs, deren Nutzung er nur für einen bestimmten Zeitraum aufrechterhält. In einigen Fällen kann die Nutzung mit zusätzlichen Kosten verbunden sein, auf deren Anfallen der Gast direkt vor der Nutzung nochmals hingewiesen wird. Diese Kosten kann der Gast sogleich mittels seines Mobiltelefons begleichen. Als Beispiel sei hier die Benutzung eines Umkleideschrankes skizziert. Da der Gast sich für bestimmte Aktivitäten auf der Clubanlage seiner normalen Freizeitkleidung entledigen möchte, werden als Service an entsprechenden Stellen Umkleideschränke zur Verfügung gestellt, die von den Gästen mit ihrem Mobiltelefon ver- und entriegelt werden können. Befindet sich ein Gast an einer solchen Stelle, bekommt er mit Hilfe der Anwendung auf seinem Telefon eine Liste aller zurzeit nicht belegten Schränke und wählt sich aus dieser einen Schrank aus, deponiert dort seine Kleidung und verschließt den Schrank mittels eines Befehls über sein Telefon. Abgesehen vom Betreiber des Ferienclubs kann dieser Schrank nunmehr nur noch vom Mobiltelefon dieses Gastes geöffnet werden. Sobald er dies tut, steht anschließend der Umkleideschrank wieder zu freien Verfügung.

3.1.7 Statistiken für den Betreiber

Durch die Möglichkeit der Aufenthaltsortbestimmung der Gäste auf dem Ferienclubgelände wird dem Betreiber nicht nur das Potential eröffnet, positionsgerichtete Dienste anzubieten,

sondern diese Daten auch statistisch auszuwerten. Für Veranstaltungsangebote, die keine konkrete Zusage oder Buchung seitens der Gäste erfordern, liegen ihm zunächst keine Informationen über den tatsächlichen Umfang der Teilnahme an diesen Angeboten vor. Mittels einer mehr oder weniger dauerhaften Erfassung der Position der Gäste kann relativ einfach protokolliert werden, wie viele Teilnehmer an einer Veranstaltung teilnehmen und ob sie sich die gesamte Dauer dort aufgehalten haben. Auf diese Weise kann der Betreiber Angebote identifizieren, die grundlegend das Interesse der Gäste geweckt haben, aber dann doch nicht deren Erwartungen erfüllen, vielleicht weil die Güte der Veranstaltung an sich oder die Leistung der betreuenden Personen, also seiner Angestellten, nicht ausreichend war. Im Laufe der Zeit lassen sich Areale des gesamten Bereiches des Ferienclubs identifizieren, an denen sich Gäste bevorzugt aufhalten und zu welchen Tageszeiten. All diese Informationen können vom Betreiber verwendet werden, um den Betrieb zu optimieren, das Veranstaltungsangebot zu verbessern und Schwachstellen der Infrastruktur der Clubanlage zu entlarven.

3.2 Grundlegende Anforderungen an das System

Grundlegende Möglichkeiten und Technologien zur Verwirklichung der in Abschnitt 3.1 beschriebenen Szenarien wurden bereits detailliert in der Studienarbeit des Verfassers (Friedburg 2005) betrachtet. An dieser Stelle sollen die Verfahren, die sich dabei als am geeignetesten erwiesen haben, kurz begründet als Anforderung mit aufgenommen werden.

Ein grundsätzlich wichtiger Aspekt für den Betreiber ist eine möglichst geringe finanzielle Investition. Dieses Kriterium beeinflusst in gewissem Maße auch die Auswahl der nachfolgenden Anforderungen.

Voraussetzung für die in Abschnitt 3.1 und 3.3 beschriebenen Anwendungsfälle ist, dass entweder im gesamten geografischen Bereich oder wenigstens in den relevanten Bereichen der Clubanlage ein zumindest lokales Netzwerk existiert, in das der Gast mittels eines mobilen Endgerätes integriert werden kann. Gleichzeitig muss es möglich sein, die Position und damit den Aufenthaltsort eines Gastes ausreichend genau bestimmen zu können. Der Betreiber möchte gerne autonom von den Mobilfunknetzwerkbetreibern agieren. Außerdem sind die Funkzellen der Mobilfunknetze bedingt durch ihren großen Radius nicht für eine ausreichend exakte Lokalisierung der mobilen Endgeräte geeignet. Deshalb soll eine eigene kostengünstige Lösung für ein Netzwerk aufgebaut werden, die zum einen in möglichst großem Maße bereits vorhandene Infrastrukturen nutzt und zum anderen auf etablierten und verbreiteten Technologien basiert. Es soll einem breiten Publikum ermöglicht werden als Gast am System teilzunehmen, indem weit verbreitete mobile Endgeräte verwendet werden können und an diese keine übermäßig speziellen Anforderungen gestellt werden. Die ursprüngliche Funktionalität und Verwendung der Endgeräte sollte nicht eingeschränkt werden.

Um eine dauerhafte Architektur zu erstellen, sind grundlegende Anforderungen aus der Informatik, wie Modularität, Wiederverwendbarkeit, Wartbarkeit und Erweiterbarkeit, im bestmöglichen Maße zu erfüllen. Die zu entwickelnde Software soll außerdem ein hohes Maß an Portabilität besitzen, d.h. ohne große Anpassungen auf unterschiedlichster Hardware unterschiedlicher Hersteller lauffähig sein. Dies betrifft zum einen die Teile der Applikation, die auf den mobilen Endgeräten installiert werden, zum anderen auch die Komponenten, die die Zugangspunkte zum Ferienclub-Netzwerk realisieren und den Kern des Ferienclubsystems selbst.

Aufgrund dieser Überlegungen werden folgende Rahmenbedingungen mit in die Anforderungen aufgenommen:

- Aufgrund der großen Verbreitung und der heutigen Leistungsfähigkeit von Smartphones liegt der Fokus bezüglich der Zielgruppe der mobilen Endgeräte bei den Mobiltelefonen. Eine Anbindung anderer handlicher Geräte wie z.B. PDAs ist keine zwingende Voraussetzung, sollte aber nach Möglichkeit zusätzlich realisierbar sein.
- Die zu realisierende Software soll mit der von der Firma Sun Microsystems entwickelten, plattformunabhängigen und objektorientierten Programmiersprache Java implementiert werden[1]. Für die Komponenten des Systems, die für die mobilen Endgeräte entwickelt werden, findet die auf der Java 2 Plattform basierende Micro Edition (J2ME) Verwendung.
- Die bereits vorhandene Infrastruktur des Netzwerkes innerhalb des Ferienclubs wird als Grundlage für die Hardware des Systems verwendet. Da in der Clubanlage bereits für die tägliche Arbeit an den meisten Stellen miteinander vernetzte Computer positioniert sind, sollen diese soweit möglich die Funktion eines Zugangspunktes für die mobilen Endgeräte zum Netzwerk übernehmen.
- Die drahtlose Anbindung der mobilen Endgeräte an das Clubnetzwerk erfolgt mit Hilfe von Bluetooth[2], da diese Technologie im Gegensatz zu IEEE 802.11[3] bereits von einer Vielzahl von Endgeräten unterstützt wird. Weiterhin lassen sich fast alle stationären Computer mit geringem Aufwand um diese Technologie erweitern. Die Datenübertragungsraten der Technologie sind für diesen Einsatzzweck ausreichend und die Größe der Funkzellen erlaubt eine ausreichend genaue Standortbestimmung.

[1] http://java.sun.com

[2] https://www.bluetooth.org

[3] IEEE 802.11 (auch bekannt unter Wireless LAN, WLAN, WiFi) bezeichnet einen Industriestandard für drahtlose Netzwerkkommunikation. Herausgeber ist das **I**nstitute of **E**lectrical and **E**lectronics **E**ngineers.

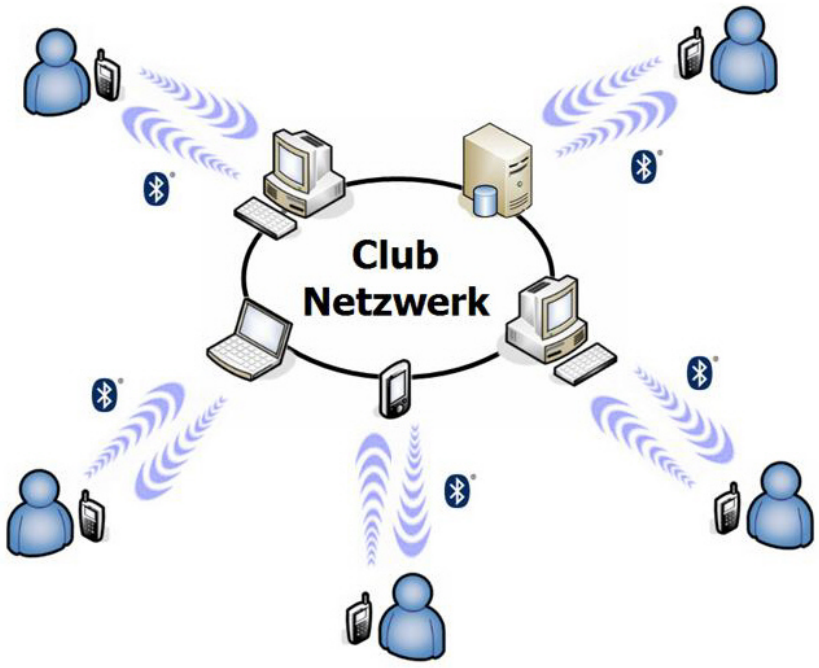

Abbildung 3.1: Grobszenario der Kommunikation im Ferienclubsystem

3.3 Anforderungen an die Funktionalität

Der mögliche Funktionsumfang des gesamten Systems wurde bereits in den Beispielszenarien in Abschnitt 3.1 beschrieben. Eine zusammengefasste, abstraktere Sicht möglicher Anwendungsfälle und deren Abhängigkeiten zeigt Abbildung 3.2.

3.3.1 Systemanwendungsfälle

Bei einer genaueren Betrachtung der Möglichkeiten der Interaktion wird deutlich, dass letztlich vier unterschiedliche grundlegende Systemanwendungsfälle für die Kommunikation zwischen dem Clubsystem und den mobilen Endgeräten der Gäste die Grundlage für die Vielzahl der Geschäftsanwendungsfälle bilden (siehe Abb. 3.3).

Diese Systemanwendungsfälle und die Abhängigkeiten bezüglich der Geschäftsanwendungsfälle sind in Abbildung 3.4 dargestellt und im folgenden näher erläutert. Die Systemanwendungsfälle sind in der Abbildung durch eine stärkere Umrisslinie hervorgehoben.

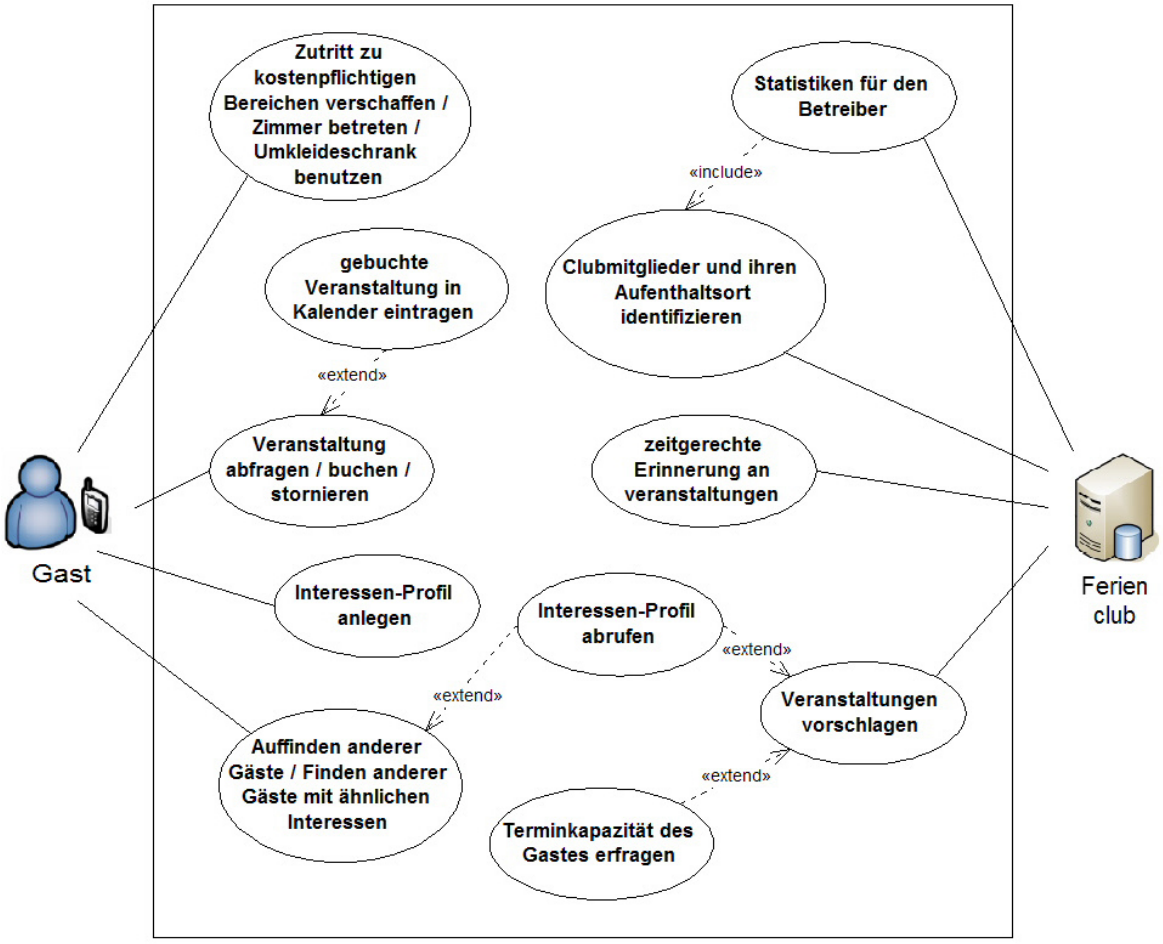

Abbildung 3.2: Übersicht möglicher Anwendungsfälle

3.3.1.1 Nutzung eines Dienstes auf einem beliebigen Zugangspunkt

Alle Anwendungsfälle mit dem Gast als Akteur, die direkt an durch das Ferienclubsystem zur Verfügung gestellte Funktionalitäten gekoppelt sind, erfordern eine zumindest temporäre Einbindung des mobilen Endgerätes des Gastes in das Clubnetzwerk. Der Standort des Gastes kann zwar als Information für den genutzten Dienst relevant sein, die Nutzung des Dienstes kann aber von einer beliebigen Position an einem beliebigen Zugangspunkt erfolgen.

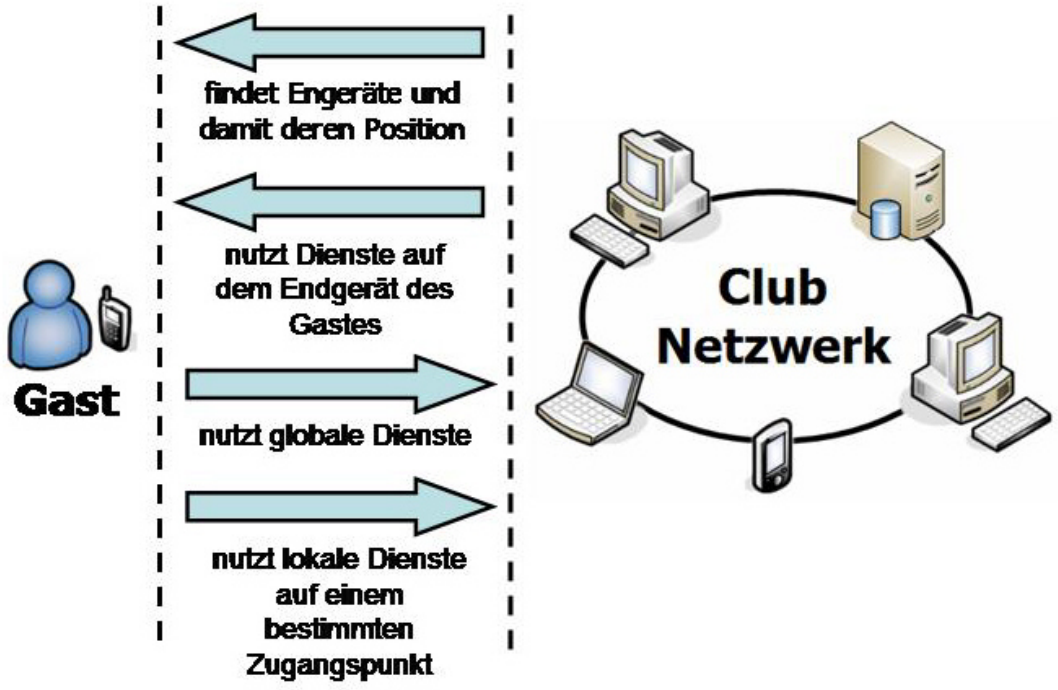

Abbildung 3.3: Interaktion zwischen Gast und Ferienclub

3.3.1.2 Nutzung eines Dienstes auf einem bestimmten Zugangspunkt

Die Dienste einiger Anwendungsfälle werden nur dann von einem Gast genutzt, wenn er sich an einer entsprechenden Position aufhält. Befindet sich der Gast außerhalb der Reichweite des entsprechenden Zugangspunktes, sind die entsprechenden Dienste nicht verfügbar. Für einige dieser Dienste ist es nicht zwingend notwendig, dass der Zugangspunkt in das Clubnetzwerk integriert ist, sondern er kann autonom agieren.

3.3.1.3 Nutzung eines Dienstes auf einem bestimmten mobilen Endgerät

Anwendungsfälle, die die Nutzung von Diensten auf der Seite der Gäste inkludieren und bei denen der Ferienclub als Akteur auftritt, haben hinsichtlich des Gastes einen individuellen Charakter und adressieren daher immer ein bestimmtes mobiles Endgerät.

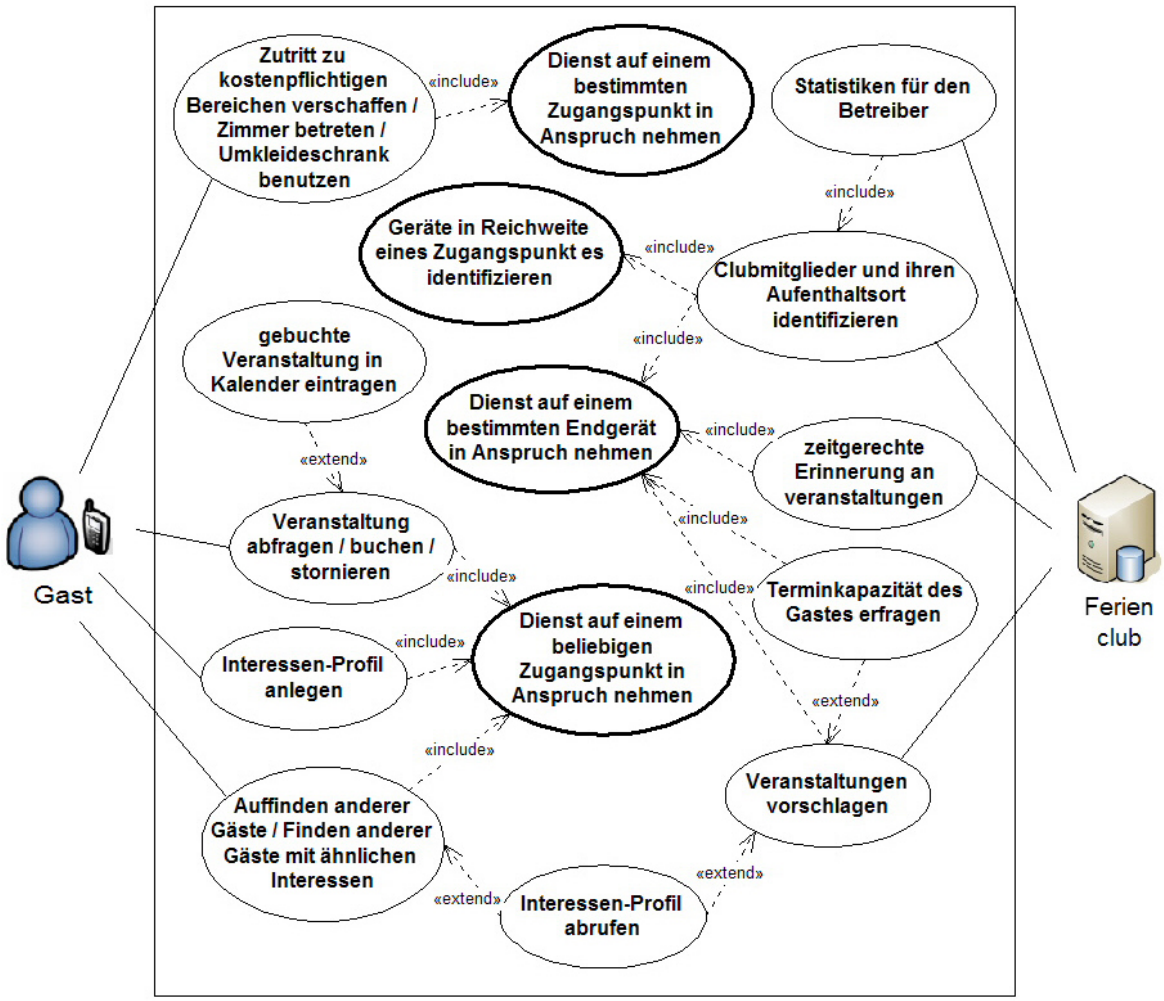

Abbildung 3.4: Abhängigkeiten zu den Systemanwendungsfällen

3.3.1.4 Identifikation mobiler Endgeräte in Reichweite eines Zugangspunktes

Das Wissen, im Bereich welcher Zugangspunkte sich ein (zumindest potenzieller) Benutzer des Ferienclubsystems zu einem bestimmten Zeitpunkt aufhält, stellt die Basis dar, um bei Bedarf Dienste auf dem mobilen Endgerät genau dieses Benutzers in Anspruch zu nehmen und ermöglicht statistische Auswertungen über das Aufenthaltsverhalten der Gäste.

3.3.2 Ausgewählte Geschäftsanwendungsfälle

Da der Schwerpunkt dieser Arbeit auf der Einbindung der mobilen Endgeräte in das Gesamtsystem liegt, sollen nicht alle möglichen Anwendungsfälle realisiert werden, sondern nur ein Teil des gesamten Ferienclubsystems exemplarisch zur Umsetzung kommen. Bereiche wie die Verwaltung und Organisation der Clubmitglieder sowie des Veranstaltungsangebotes und die damit verbundene Datenhaltung finden in diesem Rahmen keine detailliertere Betrachtung. In diesem Abschnitt werden einige der Anwendungsfälle ausgewählt und analysiert, die auf den vier grundlegenden Systemanwendungsfällen aus Abschnitt 3.3.1 basieren und die möglichen Sparten der Interaktion zwischen der Ferienclubinfrastruktur und den mobilen Endgeräten der Gäste abdecken. Abbildung 3.5 stellt die ausgewählten und geringfügig modifizierten Anwendungsfälle nochmals dar.

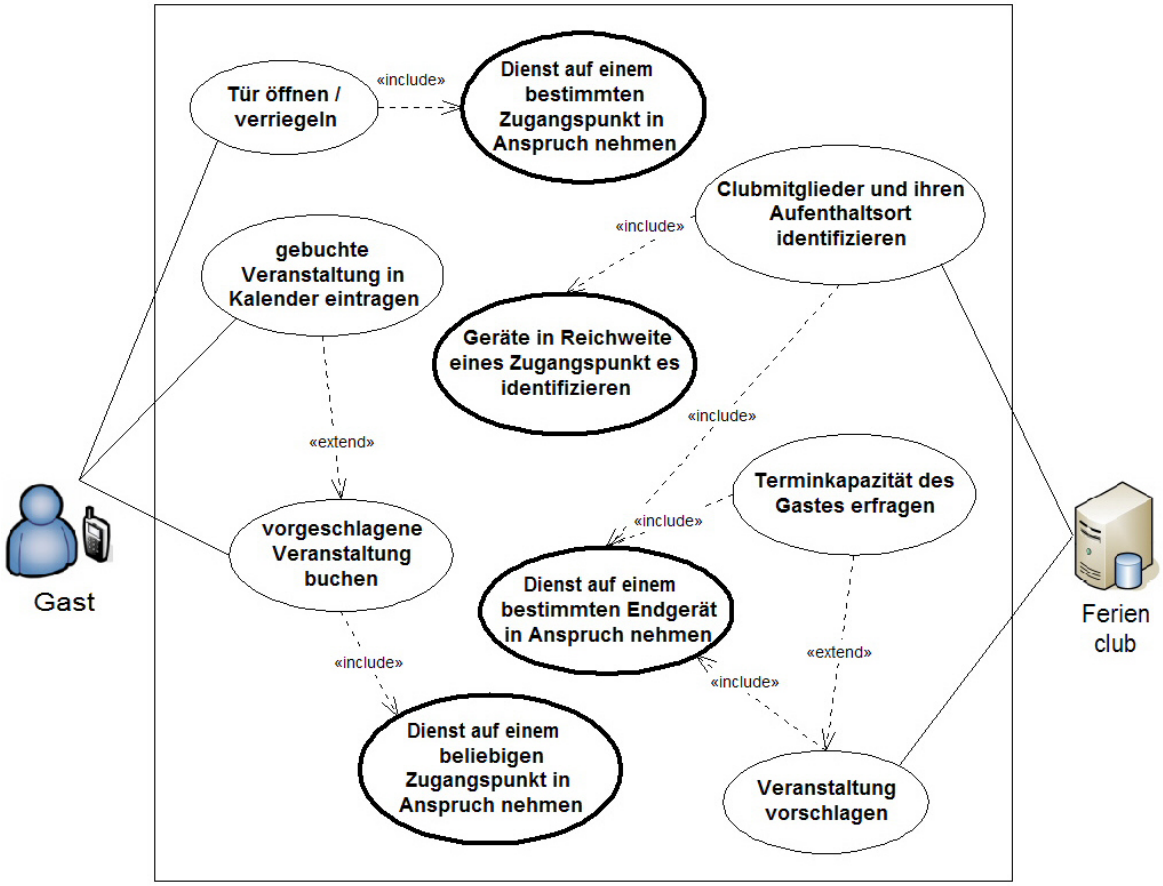

Abbildung 3.5: ausgewählte Anwendungsfälle für die Realisierung

3.3.2.1 Clubmitglieder und ihren Aufenthaltsort identifizieren

Über die drahtlosen Zugangspunkte des Clubnetzwerks werden kontinuierlich in ausreichend kurzen Zeitabständen mobile Endgeräte in deren Reichweite identifiziert. Da anhand der Anwesenheit des mobilen Endgerätes noch keine Aussage darüber getroffen werden kann, ob der Benutzer ein rechtmäßiges Mitglied des Ferienclubs ist, muss diesbezüglich eine weitere Identifikation stattfinden. Die Identifikation kann dabei stillschweigend erfolgen, d.h. das mobile Endgerät kann ohne weiteres Zutun des Benutzers agieren, indem es die hinterlegte Mitgliedsnummer des Gastes übermittelt. Wird das mobile Endgerät in einem der folgenden Zyklen an einem Zugangspunkt identifiziert, kann es mit dem entsprechenden Gast in Verbindung gebracht werden und es bedarf keiner erneuten zusätzlichen Identifikation als Mitglied.

Die Informationen, welcher Gast sich zu welchem Zeitpunkt in der Reichweite eines Zugangspunktes befunden hat, werden an eine zentrale Stelle weitergeleitet und verwaltet, um bei Bedarf einen geeigneten Zugangspunkt zur Kommunikation mit seinem mobilen Endgerät auszuwählen und Statistiken über das Aufenthaltsverhalten der Gäste erstellen zu können.

3.3.2.2 Veranstaltung vorschlagen

Voraussetzung für diesen Anwendungsfall ist, dass die Positionen der Gäste innerhalb der Clubanlage bekannt sind (siehe Anwendungsfall 3.3.2.1), um die Entfernung des Aufenthaltsortes zu anderen Orten abzuschätzen und gezielt mobile Endgeräte bestimmter Gäste ansprechen zu können.

Für eine Veranstaltung, für die noch kurzfristig ein Teilnehmer gesucht wird, soll spontan ein Gast akquiriert werden. Der Vorschlag soll nicht wahllos unterbreitet werden, sondern gezielt an Gäste gehen, die für eine Teilnahme besonders geeignet erscheinen.

Die grundsätzliche Eignung eines Gastes zeichnet sich primär dadurch aus, dass sein momentaner Aufenthaltsort auf dem Clubgelände dem Veranstaltungsort sehr nahe liegt. Er müsste also keinen weiten Weg zurücklegen, um noch an der Veranstaltung teilzunehmen. Eine Erhöhung der Eignung kommt dadurch zustande, dass der Gast für den Zeitraum der Veranstaltung bisher keine Aktivitäten für sich geplant hat (siehe Anwendungsfall 3.3.2.3). Befinden sich also mehrere Gäste zum Zeitpunkt des Vorschlags gleichermaßen günstig nah am Veranstaltungsort, werden diejenigen für den Vorschlag begünstigt, für die scheinbar für die Zeitspanne der Veranstaltung keine anderen Termine vorliegen.

Ist der Gast mit der besten Eignung gefunden, wird er auf seinem mobilen Endgerät über einen geeigneten Zugangspunkt über die Veranstaltung und deren Zeitrahmen benachrich-

tigt und erhält die Möglichkeit, eine Zu- oder Absage bezüglich seiner Teilnahme zu machen (siehe Anwendungsfall 3.3.2.4).

3.3.2.3 Terminkapazität des Gastes erfragen

Voraussetzung für diesen Anwendungsfall ist, dass die Positionen der Gäste innerhalb der Clubanlage bekannt sind (siehe Anwendungsfall 3.3.2.1), um gezielt mobile Endgeräte bestimmter Gäste ansprechen zu können. Für diesen Anwendungsfall ist keine Interaktion direkt mit dem Gast nötig.

Um die Güte des Vorschlags für eine Teilnahme an einer Veranstaltung zu verbessern, soll der Vorschlag möglichst nur Gästen unterbreitet werden, die grundsätzlich Zeit hätten, die Veranstaltung zu besuchen. Dieser Anwendungsfall erweitert den Anwendungsfall 3.3.2.2.

Soll für einen Gast bestimmt werden, ob er während einer definierten Zeitspanne keine weiteren Termine hat, werden einem Dienst auf seinem mobilen Endgerät über einen drahtlosen Zugangspunkt Beginn und Ende der Zeitspanne mitgeteilt. Im Gegenzug informiert das mobile Endgerät das Clubsystem, indem es auf den im Endgerät integrierten Kalender zugreift und feststellt, ob der Gast dort einen Termin hinterlegt hat, der mit der Zeitspanne kollidiert. Sind im Kalender des Endgerätes keine Termine für den besagten Zeitraum eingetragen, wird davon ausgegangen, dass der Gast dort prinzipiell bisher keine Aktivitäten geplant hat.

Für diesen Anwendungsfall ist keine Interaktion direkt mit dem Gast nötig, die Abfragen erfolgen durch ihn unbemerkt, so dass er nicht durch sie gestört wird.

3.3.2.4 Vorgeschlagene Veranstaltung buchen

Voraussetzung für diesen Anwendungsfall ist, dass sich der Gast mit seinem mobilen Endgerät in der Reichweite mindestens eines drahtlosen Zugangspunktes des Clubsystems befindet.

Des Weiteren schließt dieser Anwendungsfall an den Anwendungsfall 3.3.2.2 an. Der Gast hat also einen Vorschlag bezüglich einer Teilnahme an einer Veranstaltung auf sein mobiles Endgerät erhalten und entscheidet sich jetzt dazu, den letzten Vorschlag anzunehmen und seine Teilnahme verbindlich zuzusagen.

Nachdem er auf dem mobilen Endgerät die Daten über die Veranstaltung gelesen hat, wählt er aus einem Menü die Option für eine Zusage aus. An das Clubsystem wird die Information weitergegeben, dass genau dieser Gast an dieser Veranstaltung teilnehmen möchte. Der Gast wird im Clubsystem als Teilnehmer dieser Veranstaltung verzeichnet und erhält die

Rückmeldung über die erfolgreiche Buchung auf sein Endgerät. Abschließend wird der Gast gefragt, ob er die Zeitspanne der Veranstaltung in den in sein Endgerät integrierten Kalender als Termin eintragen lassen möchte (siehe Anwendungsfall 3.3.2.5).

3.3.2.5 Gebuchte Veranstaltung in Kalender eintragen

Dieser Anwendungsfall schließt an den Anwendungsfall 3.3.2.4 an. Für den Ablauf ist es jedoch nicht mehr erforderlich, dass sich das mobile Endgerät des Gastes in Reichweite eines Zugangspunktes zum Clubsystem befindet.

Nachdem der Gast seine Teilnahme an einer ihm vorgeschlagenen Veranstaltung an das Clubsystem übermittelt hat, erhält er die Möglichkeit, die Veranstaltung auf seinem Endgerät als belegte Zeit vermerken zu lassen. Macht er von dieser Option Gebrauch, werden Zeitspanne und Bezeichnung der Veranstaltung in den in sein Endgerät integrierten Kalender als Termin eintragen. Neben seinen manuell eingetragenen Terminen wird auch dieser Eintrag bei nachfolgenden Anfragen bezüglich der Terminkapazität berücksichtigt (siehe Anwendungsfall 3.3.2.3).

3.3.2.6 Tür öffnen/verriegeln

Voraussetzung für diesen Anwendungsfall ist, dass sich der Gast mit seinem mobilen Endgerät in der Reichweite eines ganz bestimmten drahtlosen Zugangspunktes befindet. Da die Dienste des Zugangspunktes hinsichtlich des übrigen Clubsystems autonom agieren, ist dabei nicht zwingend notwendig, dass dieser Zugangspunkt in das Clubnetzwerk integriert ist.

Für die Gäste des Ferienclubs sollen an einem festgelegten Ort virtuelle Schließfächer zur Verfügung stehen. Im Rahmen dieser Arbeit sind diese Schließfächer nicht real, sondern ihre Funktionalität wird durch Software simuliert. Befindet sich ein Gast in der Nähe der Schließfächer, kann er eine Liste mit allen Schließfächern auf sein mobiles Endgerät anfordern. Aus der Darstellung innerhalb der Liste geht hervor, ob ein Schließfach bereits benutzt wird oder frei verfügbar ist. Der Gast wählt ein freies Fach aus und verriegelt es. Hierzu wird er über eine Eingabemaske aufgefordert, einen beliebigen maximal zehnstelligen Code zu vergeben. Mit dem Absenden des Codes ist das virtuelle Schließfach verschlossen und steht damit anderen Gästen nicht mehr zur Verfügung. Ein Gast kann parallel mehrere Schließfächer am selben Ort nutzen. Um ein Schließfach wieder zu öffnen, fordert der Gast wiederum eine Liste der Schließfächer auf sein mobiles Endgerät an und wählt das entsprechende aus. Über eine Eingabemaske wird der Gast aufgefordert, seinen beim Verriegeln des Faches vergebenen Code einzugeben. Nach der Übermittlung des Codes vom Endgerät zum drahtlosen

Zugangspunkt wird die Übereinstimmung überprüft. Eine erfolgreiche Übereinstimmung öffnet das Fach und es steht wieder zur freien Benutzung, bei einem falschen Code bleibt es verschlossen. In beiden Fällen wird der Gast auf seinem Endgerät entsprechend informiert. Das Entriegeln eines Schließfaches ist nur abhängig von der korrekten Eingabe des Codes und kann von einem beliebigen mobilen Endgerät aus erfolgen. Auf diese Weise können zusammengehörige Gäste, z.B. die Mitglieder einer Familie, das selbe Schließfach nutzen, solange ihnen der Code bekannt ist.

4 Entwurf und Realisierung

Nachdem im vorangegangenen Kapitel die Anforderungen und der Funktionsumfang bezüglich der Plattform diskutiert wurden, soll in diesem Kapitel das technische Design für eine möglich Lösung erarbeitet werden.

Aus Gründen einer gesteigerten Wiederverwendbarkeit und um die Komplexität des Systems handhabbar zu machen, erfolgt eine Aufteilung in Komponenten, die jeweils ein Teilproblem der Gesamtfunktionalität lösen. Durch die Eigenständigkeit der Komponenten werden Seiteneffekte und Fernwirkungen auf ein Minimum reduziert, was für eine bessere Wartbarkeit der Software sorgt. Komponenten sind per Definition zu einem gewissen Grad wiederverwendbar und anpassbar, so dass neue Systeme aus bestehenden Komponenten aufgebaut werden können. Griffel (1998) und Szyperski (1999) geben einen guten Überblick über das komponentenorientierte Paradigma.

Zu beachten ist, dass die Komponenten, die die Bereitstellung und Nutzung von Diensten basierend auf der Bluetooth-Technologie realisieren, sowohl auf der Seite des Clubnetzwerkes, als auch auf den Endgeräten der mobilen Teilnehmer des Systems eingesetzt werden. Dies bedingt, dass die Bestandteile dieser Komponenten der CLDC-Spezifikation der J2ME in der Version 1.1 entsprechen. Dadurch wird erreicht, dass die Komponenten ohne weitere Anpassungen sowohl auf modernen Mobiltelefonen und PDAs, als auch auf JAVA 2 Standard Edition unterstützender Hardware lauffähig sind. Der Verzicht auf den Leistungsumfang der JAVA 2 Standard Edition wird für die Realisierung dieser Komponenten also bewusst in Kauf genommen. Zusätzlich müssen die auf mobilen Endgeräten eingeschränkten Ressourcen bei der Entwicklung berücksichtigt werden (Kroll und Steinhaus 2003). Die Komponenten sollten so klein wie möglich gehalten werden, um den zur Verfügung stehenden Speicher für die Installation zusätzlicher Anwendungen nicht zu überfordern. In diesem Zusammenhang bietet sich auch der Einsatz eines Obfuscators[1] an, um den Umfang der kompilierten Applikation für die mobilen Endgeräte in diesem Sinne zu optimieren. Ein geringer Umfang begünstigt des Weiteren den Zeitaufwand der Übertragung bei der Installation der Anwendung auf dem Endgerät. Auch der geringere Hauptspeicher für die Ausführung von Programmen muss beim Implementieren Berücksichtigung finden.

[1] Ein Obfuscator entfernt unnötige Informationen aus einem kompilierten Javaprogramm. Die eigentliche Intention dabei ist, ein Reverse-Engineering durch Entzug der Bezeichnersemantik zu erschweren. Als Seiteneffekt verkleinert sich auch das Kompilat.

Ein Überblick über die Komponenten, deren Abhängigkeiten untereinander und ihre Verteilung bezüglich des clubeigenen Netzwerkes und der mobilen Endgeräte ist in Abbildung 4.1 dargestellt.

Im weiteren Verlauf dieses Kapitels werden die Aufgaben der einzelnen Komponenten detaillierter dargestellt und die enthaltenen Klassen entworfen und implementiert, die die Funktionalität realisieren. Abschließend werden die Interaktionen der Komponenten untereinander für den Ablauf der Anwendungsfälle dargestellt.

Die Komponenten *Datenhaltung* für die Persistierung und *Reporting* für die Erstellung von Statistiken für den Betreiber des Ferienclubs sind der Vollständigkeit halber in den Diagrammen aufgeführt, finden im weiteren Verlauf aber keine weitere Berücksichtigung, um den Rahmen dieser Arbeit nicht zu sprengen. Die Komponenten *Cluborganisation*, *Persönliche Organisation* und *Userinteraktion* werden weniger detailliert betrachtet und nur rudimentär soweit entwickelt und implementiert, dass die Gesamtfunktionalität der Plattform anhand des Prototypen demonstriert werden kann.

4.1 Dienste

Die drahtlose Inanspruchnahme und damit auch die Bereitstellung von Diensten stellt eine Kernfunktionalität des Systems dar, auf der andere Funktionalitäten der Plattform aufbauen. Es wurde zwar die Verwendung der Bluetooth-Technologie für die drahtlose Kommunikation als Anforderung für das System formuliert, trotzdem werden die Klassen der Komponenten für die Benutzung von Diensten so entkoppelt, dass bei Bedarf ohne große Anpassungen der Implementation die Technologie durch eine andere ausgetauscht werden kann.

Obwohl auf den mobilen Endgeräten und ebenso auf den Zugangspunkten zum Clubnetzwerk grundsätzlich sowohl eine Dienstnutzung, als auch eine Dienstbereitstellung verwendet wird, sollen diese trotzdem als voneinander unabhängige Module realisiert werden, um bei Bedarf eine einzelne Verwendung zu ermöglichen. Auch für die Dienste erfolgt eine Einteilung in unterschiedliche Kategorien und damit Module. So könnten bei einem Einsatz auf sehr stark ressourcenbeschränkter Hardware nur wirklich benötigte Komponenten installiert werden.

Eine Dienstkategorie bündelt also eine Anzahl von Diensten, die an einer bestimmten Stelle erreichbar sind. Diese Stelle kann ein mobiles Endgerät, ein bestimmter Zugangspunkt oder ein beliebiger Zugangspunkt stellvertretend für ortsunabhängig nutzbare Dienste sein. Um einen Dienst universell ansprechbar zu machen, wird jede konkrete Implementierung von der Klasse *Service* abgeleitet und kann so abstrahiert angesprochen werden. Von der Verwendung eines reinen Interface statt einer abstrakten Klasse wird hier abgesehen, da die Klasse *Service* bereits Funktionalitäten für das Auslesen der einzelnen Parameter für einen

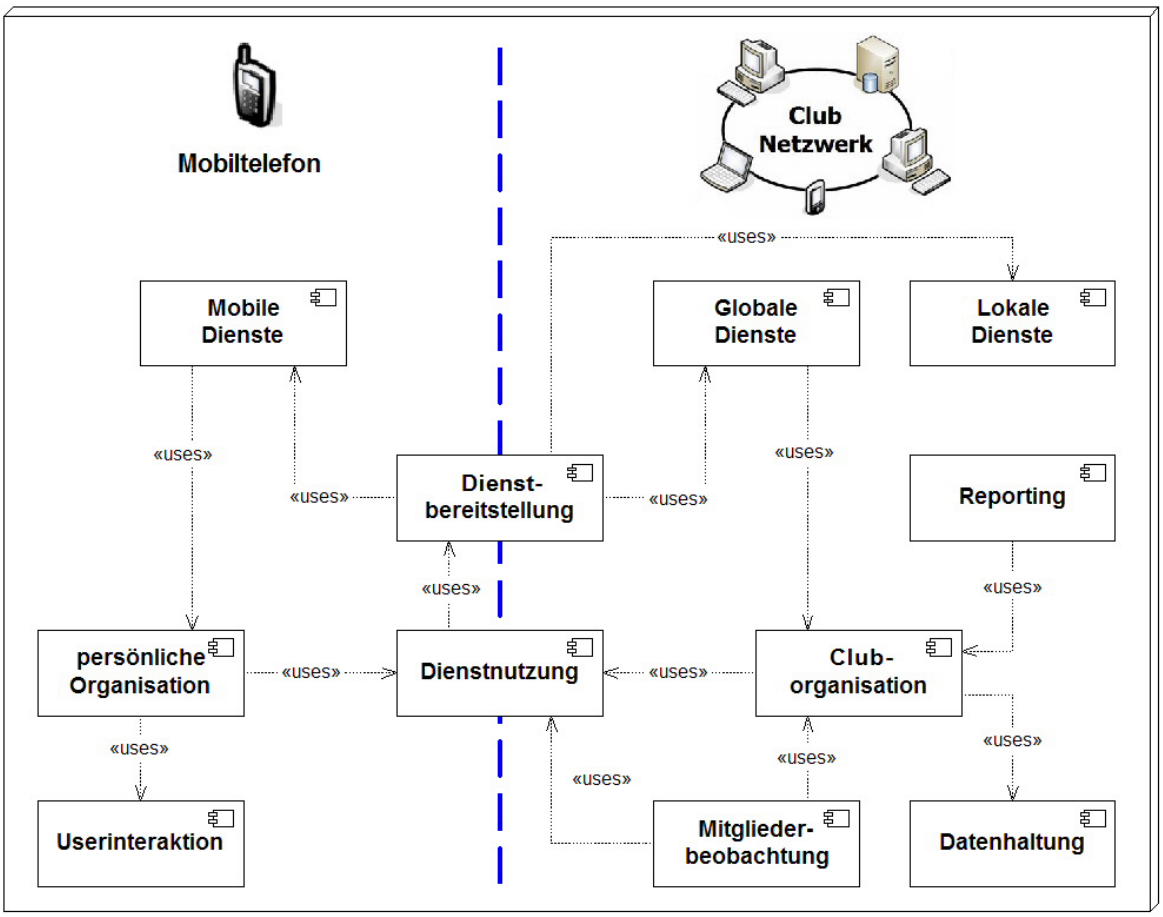

Abbildung 4.1: Komponenten der Plattform

Dienst aus der übergebenen Zeichenkette konkret implementiert und an die spezialisierten Dienst-Klassen vererbt.

Folgende Funktionalitäten werden dabei zur Verfügung gestellt:

- *getServiceID()*
  Gibt eine Zeichenkette zur eindeutigen Identifizierung des Dienstes zurück.
- *getServiceCategory()*
  Gibt die Kategorie zurück, der der Dienst zugeordnet ist.
- *doService(params)*
  Stößt die Ausführung des Dienstes an. Benötigt der Dienst weitere Informationen zur Ausführung, werden diese in einer Zeichenkette als Parameter übergeben. Für den

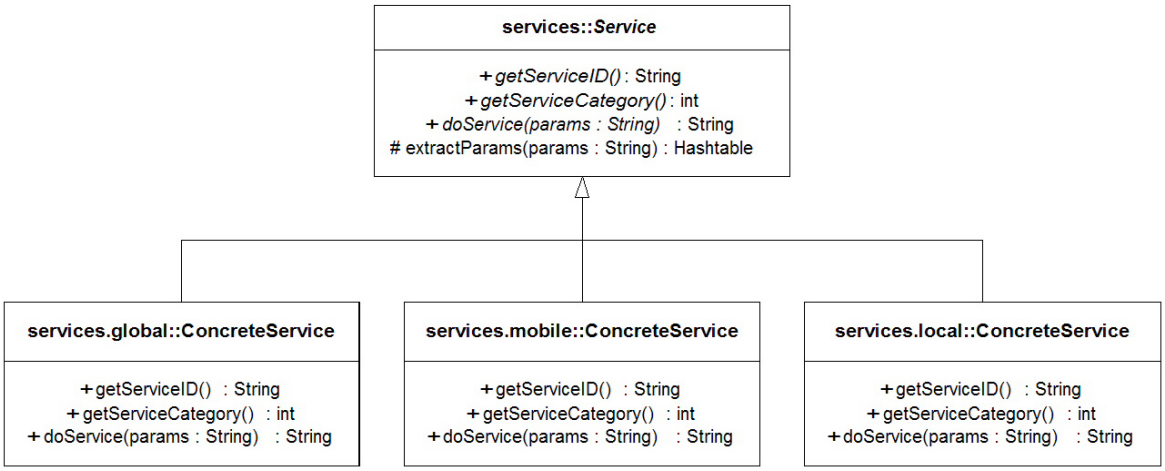

Abbildung 4.2: Abstrakte Klasse Service

Aufbau der Zeichenkette gilt dabei angelehnt an die Parameterübergabe bei einen HTTP-Request folgende Form:

parameterName=parameterWert&parameterName=parameterWert ...

Der Inhalt der zurückgegebenen Zeichenkette hängt von den Eigenheiten des jeweiligen konkreten Dienstes ab.

4.1.1 Mobile Dienste

Mobile Dienste werden auf den mobilen Endgeräten zur Verfügung gestellt und im Auftrag des zentralen Ferienclubsystems von den Zugangspunkten des Clubnetzwerkes angesprochen. Daher müssen die Implementierungen dieser Dienste J2ME-konform sein. Für die Ausführung der Dienste werden Funktionalitäten der Komponente *Persönliche Organisation* (siehe Abschnitt 4.3.1) in Anspruch genommen, die Ergebnisse aufbereitet und als Antwort auf die Anfrage an den Dienstnutzer zurückgegeben. Stellvertretend für diese Kategorie werden im Rahmen dieser Arbeit die nachfolgenden Dienste benötigt.

4.1.1.1 MemberIdentificationService

Um ein Mitglied als solches zu identifizieren, ist die Tatsache hinreichend, dass ein mobiles Endgerät überhaupt mobile Dienste zur Verfügung stellt. Damit zwischen den Mitgliedern

differenziert werden kann, ist aber eine individuelle Identifikation nötig. Dieser Dienst benötigt keine weiteren Informationen als Parameter und gibt die eindeutige auf dem mobilen Endgerät hinterlegte Mitgliedsnummer zurück.

4.1.1.2 DateCapacityService

Mit diesem Dienst kann der interne Kalender des mobilen Endgerätes eines Gastes befragt werden, ob für einen bestimmten Zeitraum Termine eingetragen sind. Als Parameter werden Beginn und Ende des Zeitraumes benötigt, wobei jeder Zeitpunkt wie oft üblich jeweils durch einen Long-Wert repräsentiert wird.

dateBegin=1121172516401&dateEnd=1121172567114

Rückgabewert ist eine Zeichenkette mit dem Inhalt *TRUE*, wenn in dem Zeitraum kein eingetragener Termin gefunden wurde, mit dem Inhalt *FALSE*, wenn mindestens ein Termin gefunden wurde oder <*NULL*>, wenn während der Dienstausführung ein Fehler aufgetreten ist.

4.1.1.3 EventProposalService

Um einem bestimmten Mitglied eine Veranstaltung vorzuschlagen, wird dieser Dienst verwendet. Benötigte Parameter sind die eindeutige Identifizierung der Veranstaltung, der Veranstaltungsname, eine kurze Beschreibung und die Zeitpunkte von Beginn und Ende der Veranstaltung. Die Zeitpunkte werden dabei jeweils durch einen Long-Wert repräsentiert.

eventID=12345&eventName=Beachvolleyball&eventDescription=Turnier mit Preisen&eventBegin=1121172516401&eventEnd=1121172567114

Wenn der Dienst erfolgreich in Anspruch genommen wurde ist der Rückgabewert eine Zeichenkette mit dem Inhalt *TRUE*. Im Falle eines Fehlers während der Ausführung wird <*NULL*> zurückgegeben.

4.1.2 Globale Dienste

Globale Dienste stehen auf allen Zugangspunkten des Clubnetzwerks zur Verfügung und werden von den mobilen Endgeräten genutzt. Kennzeichnendes Merkmal solcher Dienste ist meist, dass sie auf zentrale Daten der Clubverwaltung angewiesen sind. Daher werden sie auf dem zentralen Server des Clubsystems zur Verfügung gestellt und die Implementierung ist deshalb nicht auf den Leistungsumfang der J2ME beschränkt, sondern kann die volle Funktionalität der JAVA 2 SE nutzen. Für die Ausführung der Dienste werden Funktionalitäten der Komponente *Cluborganisation* (siehe Abschnitt 4.2.2) in Anspruch genommen, die Ergebnisse aufbereitet und als Antwort auf die Anfrage an den Dienstnutzer zurückgegeben. Im Rahmen dieser Arbeit wird stellvertretend für diese Kategorie nur der eine nachfolgend beschriebene Dienst implementiert.

4.1.2.1 EventBookingService

Dieser Dienst ermöglicht einem Gast seine Zusage bezüglich einer Teilnahme an einer bestimmten Veranstaltung des Clubs abzugeben. Notwendige Parameter sind die Identifizierung der Veranstaltung und die Mitgliedsidentifizierung des Gastes.

eventID=12345&memberID=191269

Wenn der Dienst erfolgreich in Anspruch genommen wurde ist der Rückgabewert eine Zeichenkette mit dem Inhalt *TRUE*. Im Falle eines Fehlers während der Ausführung wird <*NULL*> zurückgegeben.

4.1.3 Lokale Dienste

Lokale Dienste stehen nur an bestimmten Stellen zur Verfügung und sind daher nur auf einem ausgewählten Zugangspunkt nutzbar. Falls für die Durchführung des Dienstes keine zentralen Daten der Clubverwaltung nötig sind, muss der Zugangspunkt nicht zwingend in das Clubnetzwerk integriert sein. Als Beispiel für diese Kategorie werden virtuelle Schließfächer implementiert, deren Verwendung impliziert, dass der Benutzer direkt vor Ort ist (siehe Abschnitt 4.2.4). Die Notwendigkeit einer J2ME-Konformität bei der Implementierung hängt von der verwendeten Hardware für den Zugangspunkt ab. Ist dieser ein handelsüblicher PC, wie bei der Umsetzung des Prototyps im Rahmen dieser Arbeit, kann der volle Funktionsumfang der JAVA 2 SE genutzt werden.

4.1.3.1 LockerListService

Mit Hilfe dieses Dienstes wird eine Liste der existenten Schließfächer angefordert. Der Dienst erfordert keine weiteren Informationen und benötigt keine Parameter. Die zurückgegebene Zeichenkette enthält alle Schließfächer, wobei ein solches durch eine Zuordnung des Zustands des jeweiligen Schließfaches zu der Schließfachidentifikation repräsentiert wird. Möglichkeiten für den Zustand können dabei entweder *LOCKED* für ein verschlossenes oder *AVAILABLE* für ein verfügbares Fach sein. Die Repräsentationen der Schließfächer werden innerhalb der Zeichenkette durch den Separator '&' getrennt.

01=LOCKED&02=AVAILABLE&03=AVAILABLE&04=LOCKED ...

4.1.3.2 LockerService

Über diesen Dienst werden die virtuellen Schließfächer ver- und entriegelt und er erwartet Parameter für die eindeutige Identifikation eines Schließfaches, einen maximal zehnstelligen Code sowie die auszuführende Aktion. Für die Aktion gibt es dabei zwei mögliche Werte. Mit dem Wert *LOCK* wird das Schließfach verschlossen, mit dem Wert *UNLOCK* wird versucht, das Schließfach mit dem übergebenen Code zu öffnen.

lockerID=01&lockerAction=LOCK&lockerCode=0123456789

Rückgabewert ist eine Zeichenkette mit dem Inhalt *TRUE*, wenn das Fach erfolgreich verschlossen oder geöffnet wurde oder *FALSE*, wenn die Aktion nicht ausgeführt werden konnte oder <*NULL*> wenn während der Dienstausführung ein Fehler aufgetreten ist. Die Darstellung des Codes erfolgt im Klartext, da davon ausgegangen wird, dass das zur drahtlosen Übermittelung herangezogene Protokoll für eine sichere Verschlüsselung sorgt.

4.1.4 Dienstbereitstellung

Ein Eckpfeiler der verwendeten Bluetooth-Technologie ist die Idee, standardisierte und damit universell nutzbare Dienste anzubieten. Hierzu werden in der Bluetooth-Spezifikation sog. Profile[2] definiert, so dass sich zwei ein solches Profil unterstützende Bluetooth-Geräte für bestimmte Anwendungsfälle ohne weitere Konfigurierungen verbinden und austauschen können.
Auch im Rahmen der Plattform müssen Dienstnutzer die zur Verfügung gestellten Dienste

[2] eine Liste aller Bluetooth-Profile ist unter http://www.bluetooth.org zu finden

und deren Nutzungsweise kennen, sie sollen aber ausschließlich für die Teilnehmer der Plattform und nicht für die Öffentlichkeit, also „plattformfremde“ Bluetooth-Geräte, verfügbar sein. Dies gilt sowohl seitens der mobilen Endgeräte, als auch der stationären Zugangspunkte. Dienste werden also nicht offiziell publiziert, sondern stillschweigend genutzt.

Zusätzlich zu den standardisierten als Bluetooth-Profile definierten Applikationen ist mit Hilfe der JABWT [3] die Entwicklung eigener Applikationen möglich. Hierbei kann die L2CAP API, die RFCOMM API oder die OBEX API Verwendung finden. Diese APIs setzen auf unterschiedliche Protokolle des Bluetooth-Stacks auf (zu den Protokollen des Bluetooth-Stacks siehe auch Abschnitt 2.1.1 und 2.1.2). In den meisten Fällen bieten die Stream-orientierten APIs der higher-level Protokolle RFCOMM und OBEX Vorteile, da das L2CAP-Protokoll paketorientiert arbeitet und weder Mechanismen für eine Flusskontrolle, noch für eine verlässliche Übermittlung der Pakete bereitstellt (Kumar u. a. 2004).Die OBEX API ist zwar Spezifikationsbestandteil des JSR-82, wird aber zur Zeit von den meisten die Spezifikation unterstützenden Smartphones noch nicht realisiert und kommt schon aus diesem Grunde als Grundlage für eine Implementierung nicht in Betracht. Ein weiterer Vorteil der RFCOMM API gegenüber der L2CAP API ist der, dass für das eigentliche Empfangen und Senden der Daten die Stream-Klassen der J2ME und keine Klassen der JABWT verwendet werden, wodurch eine weitere Entkopplung hinsichtlich der Bluetooth-Technologie realisiert werden kann. Aus diesen Gründen fällt die Entscheidung, für die Bluetooth-Kommunikation in diesem Fall die RFCOMM API zu verwenden.

Im Weiteren werden die wichtigsten für die Dienstbereitstellung benötigten Klassen und deren Funktionsweise beschrieben. Die eigentliche Dienstbereitstellung wird dabei nochmals von der Bluetooth-Technologie abstrahiert, um einen Austausch oder eine einfache Erweiterung mit einer anderen zusätzlichen Technologie zu ermöglichen. Das an den wichtigen Stellen detaillierte Klassendiagramm in Abbildung 4.3 gibt einen Überblick über die Abhängigkeiten und den Aufbau der Klassen. Ein vollständiges Klassendiagramm ist in der Abb. A.1 im Anhang zu finden.

4.1.4.1 Klasse ServiceServerImpl

Diese Klasse realisiert den eigentlichen Server, der sämtliche im Kontext der Plattform benötigten, drahtlos erreichbaren Dienste vorhält. Einer Instanz dieser Klasse können mit der Methode *addService(aService:Service)* beliebig viele Service-Objekte zugewiesen werden, wobei diese bezüglich der in Abschnitt 4.1 definierten Kategorien homogen sein müssen. Für die Verwaltung der Service-Objekte wird ein Container benötigt, mit dessen Hilfe ein direkter Zugriff mittels eines identifizierenden Schlüssels auf diese möglich ist. Da alle Klassen der Komponente für die Dienstbereitstellung auch auf den mobilen Endgeräten lauffähig

[3] Java APIs for Bluetooth Wireless Technology

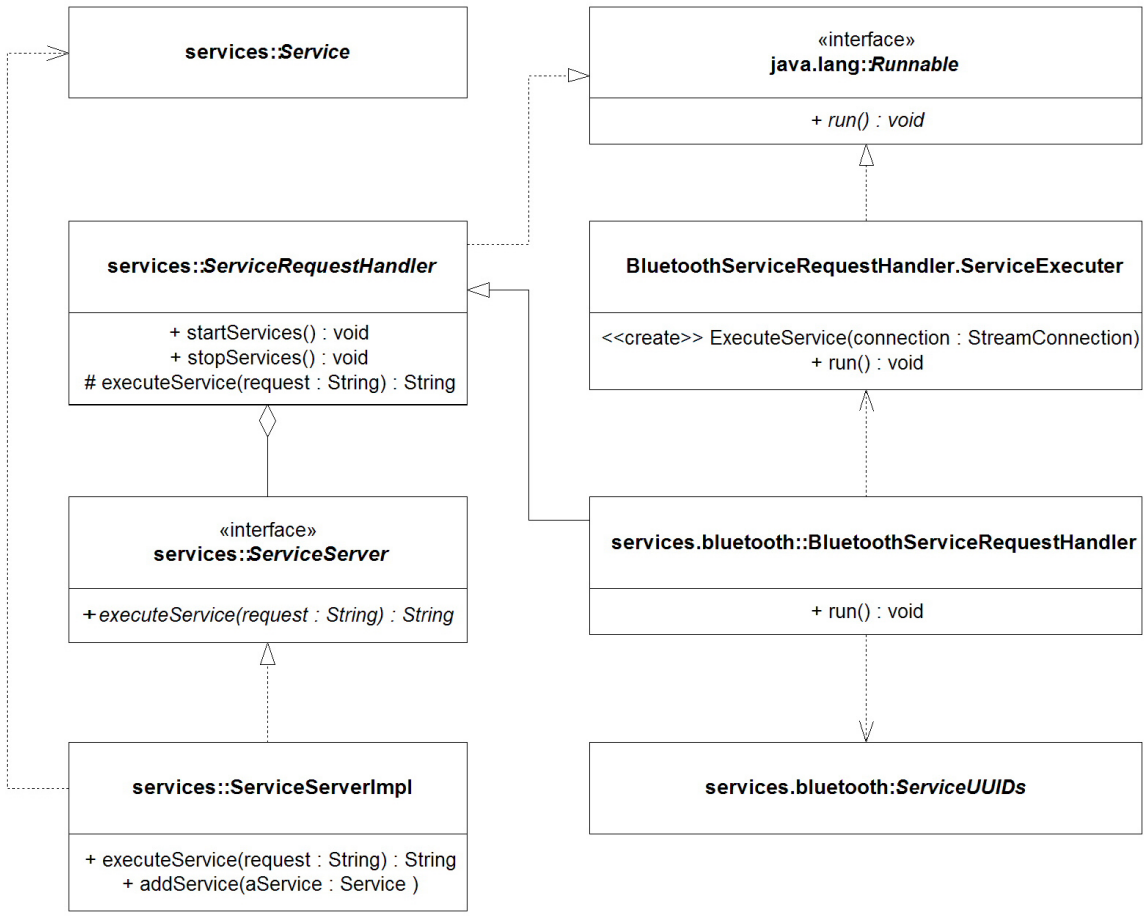

Abbildung 4.3: Klassendiagramm Dienstbereitstellung

sein sollen, wird für diese Aufgabe ein *java.util.Hashtable verwendet*, da dieser sowohl zum Umfang der JAVA 2 SE gehört, als auch Bestandteil der CLDC-Spezifikation der J2ME ist.

Die Klasse *ServiceServerImpl* implementiert das Interface *ServiceServer* und wird über dessen Methode *executeService(serviceRequest:String)* für die Ausführung eines Dienstes angesprochen. Dabei wird beim Aufruf der Methode der vollständige, wie bei den Diensten in Abschnitt 4.1 beschriebene Dienst-Request übergeben und aus diesem anschließend der Identifier für den gewünschten Dienst extrahiert. Anhand des Identifiers kann der entsprechende aus den vorgehaltenen Diensten gefunden werden und für die Ausführung die restliche Zeichenkette mit eventuellen Parametern an das Service-Objekt delegiert werden.

Die Instanzen der Klasse *ServiceServerImpl* verteilen sich innerhalb des Gesamtsystems wie folgt:

- für die globalen Dienste eine Instanz an zentraler Stelle
- für spezielle lokale Dienste jeweils eine Instanz auf dem jeweiligen Zugangspunkt
- für die mobilen Dienste jeweils eine Instanz auf jedem mobilen Endgerät

4.1.4.2 Klasse ServiceRequestHandler

Diese abstrakte Klasse definiert den funktionalen Rahmen für die Entgegennahme einer Dienstanfrage. Die eigentliche Realisierung der physikalischen Dienstbereitstellung ist Aufgabe einer konkreten abgeleiteten Subklasse. Im Falle der Entwicklung des Prototyps im Rahmen dieser Arbeit ist dies die im Abschnitt 4.1.4.3 behandelte Klasse *BluetoothServiceRequestHandler*, die hierfür die Bluetooth-Technologie verwendet.

Jede Instanz eines *ServiceRequestHandler* arbeitet mit einem Server zusammen, der die möglichen Dienste vorhält. Für die Anbindung des Servers wird hierbei das Interface *ServiceServer* benutzt, über dessen Methode *executeService(serviceRequest:String)* der Request bezüglich eines Dienstes an den Server weitergeleitet wird. Die Verwendung eines Interfaces statt einer konkreten Implementierung ermöglicht eine dynamische Anbindung beliebiger Klassen, die entweder direkt die Server-Funktionalitäten realisieren, oder für eine Kommunikation mit dem eigentlichen Server zuständig sind, falls dieser nicht in derselben virtuellen Maschine ausgeführt wird (siehe hierzu auch Abschnitt 4.2.3).

Die Methode *startServices()* startet einen neuen Prozess, in dem kontinuierlich Dienstanfragen entgegen genommen und bearbeitet werden. Bei der Verwendung von JAVA ist es für Instanzen, die in einem nebenläufig Prozess ausgeführt werden sollen, notwendig, das Interface *java.lang.Runnable* zu implementieren. Durch diese Vorgehensweise ist es möglich, mit einem Aufruf der Methode *stopServices()* den Prozess und damit die Entgegennahme von Dienstanfragen zu terminieren, auch wenn es innerhalb der Ausführung blockierende Methodenaufrufe gibt.

Die Instanzen der Subklassen von *ServiceRequestHandler* verteilen sich innerhalb des Gesamtsystems wie folgt:

- für die globalen Dienste jeweils eine Instanz auf jedem Zugangspunkt
- für spezielle lokale Dienste jeweils eine Instanz auf dem jeweiligen Zugangspunkt
- für die mobilen Dienste jeweils eine Instanz auf jedem mobilen Endgerät

4.1.4.3 Klasse BluetoothServiceRequestHandler

Mit dieser Klasse wird ein konkreter *ServiceRequestHandler* auf Basis der Bluetooth-Technologie realisiert. Es werden alle Dienste einer Kategorie des korrespondierenden Servers als Bluetooth-Service zur Verfügung gestellt. Um einen konkreten Service unter Bluetooth anzusprechen, werden für eine eindeutige Identifikation sog. UUIDs[4] verwendet. Ein solcher Service wird innerhalb der zu entwickelnden Plattform durch eine Dienstkategorie (also globale, mobile oder lokale Dienste) repräsentiert. Dementsprechend muss es eine gemeingültige, globale Definition geben, welche Dienstkategorien unter welchen UUIDs angeboten und auf der anderen Seite angesprochen werden. Die Klasse *ServiceUUIDs* hält diese Zuordnung vor und stellt die Funktionalität zur Verfügung, die entsprechende UUID für eine Dienstkategorie abzurufen.

Nachdem der Prozess für die kontinuierliche Entgegennahme von Dienstanfragen gestartet wurde (siehe Abschnitt 4.1.4.2), wird das lokale Bluetooth-Device so konfiguriert, dass es für alle anderen Bluetooth-Geräte sichtbar ist. Wenn ein Bluetooth-Gerät in Reichweite eine Suche nach anderen Bluetooth-Geräten durchführt, ist das lokale Bluetooth-Device für dieses auffindbar. Die Möglichkeit, Bluetooth-Geräte in der näheren Umgebung aufzufinden, ist elementarer Bestandteil für die Dienstnutzung (siehe hierzu auch Abschnitt 4.1.5).

Durch den Aufruf der Methode *open(url:String)* der Klasse *javax.microedition.io.Connector* wird ein Objekt erzeugt, welches das Interface *javax.microedition.io.Connection* implementiert und die grundsätzliche Verbindungsaufnahme mit einem entsprechenden Service auf dem lokalen Bluetooth-Device ermöglicht. Unter der übergebenen URL wird der Service erreichbar sein, sie hat dabei folgende Form:

btspp://localhost:<UUID>;authenticate=false

Die Belegung des Parameters *authenticate* bewirkt dabei, dass das anfragende Gerät für die Nutzung des Service nicht authentifiziert sein muss. Ein solche Authentifizierung kann zwar einmalig und dann zukünftig gültig erfolgen, betrifft aber immer nur die Kommunikation zwischen zwei bestimmten Bluetooth-Geräten. Demnach müsste eine Authentifizierung zwischen jedem mobilen Endgerät und jedem einzelnen Zugangspunkt stattfinden. Die Tatsache, dass die Nutzung des Service die Kenntnis der zugehörigen UUID vorraussetzt, soll hier in diesem Zusammenhang als Authentifizierung ausreichend sein.

Durch die Verwendung des Protokolls *btspp (bluetooth serial port protocol*) erzeugt der Aufruf ein Objekt für eine speziellere streambasierte Verbindungsmöglichkeit der Art *javax.microedition.io.StreamConnectionNotifier*, die eine Kommunikation auf Basis des

[4] Die Klasse javax.bluetooth.UUID definiert allgemeine eindeutige Identifier. Diese 128-bit unsigned Integer-Werte sind garantiert zu jedem Zeitpunkt an jedem Ort einzigartig

RFCOMM-Profils ermöglicht. Durch den Aufruf der Methode *acceptAndOpen()* wird der Prozess solange blockiert, bis eine Anfrage für einen Bluetooth-Service unter der propagierten UUID auf dem lokalen Bluetooth-Device eintrifft. In diesem Fall gibt der Methodenaufruf eine *javax.microedition.io.StreamConnection* zurück, über die mit dem anfragenden Bluetooth-Gerät kommuniziert werden kann. Für die weitere Kommunikation über diese Verbindung wird für jede Anfrage ein Objekt der Klasse *ServiceExecuter* instanziiert und diesem der geöffnete Kommunikationskanal übergeben. Anschließend kann sofort eine weitere Service-Anfrage entgegengenommen werden.

Die Klasse *ServiceExecuter* wird als innere Klasse des *BluetoothServiceRequestHandler* realisiert, da sie ausschließlich von diesem genutzt wird. Eine Instanz liest den eingehenden Dienst-Request als Zeichenkette aus und delegiert diese an den *ServiceRequestHandler*, der versucht, den gewünschten Dienst zu identifizieren und eine Ausführung anzustoßen. Der Rückgabewert der Dienstausführung wird vom *ServiceExecuter* über den Kommunikationskanal an das anfragende Bluetooth-Gerät übermittelt und dann abschließend die Verbindung geschlossen. Damit die Abarbeitung der Dienstanfragen parallel ablaufen kann, wird jeder *ServiceExecuter* in einem neuen Prozess gestartet.

Das Zusammenspiel mit den Klassen der JABWT und der CLDC ist nochmals grafisch als Klassen- bzw. Sequenzdiagramm in den Abbildungen 4.4 und 4.5 skizziert.

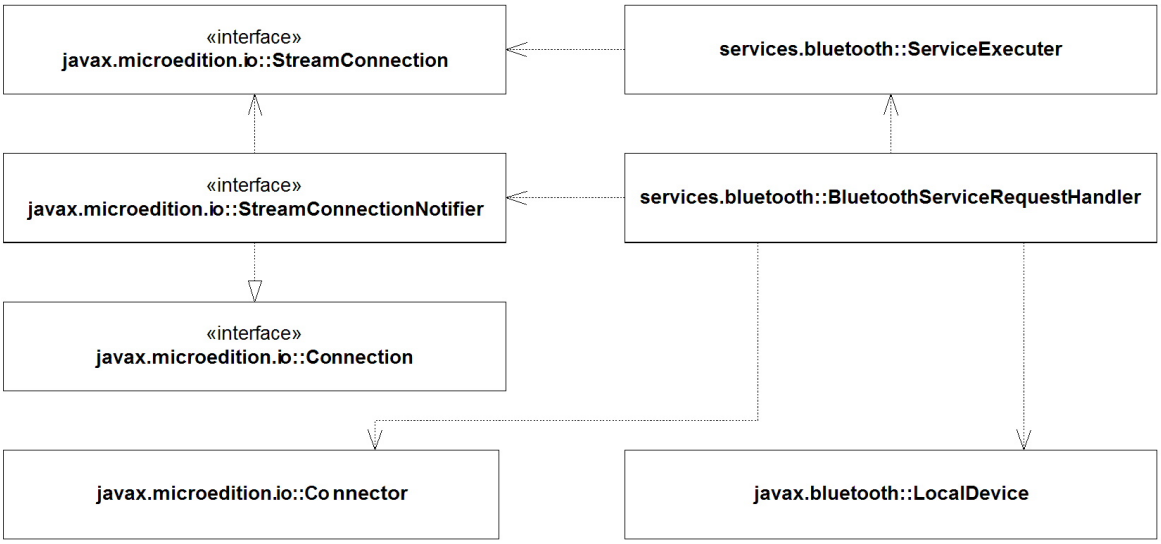

Abbildung 4.4: Klasse BluetoothServiceRequestHandler und JSR-82 und CLDC 1.1

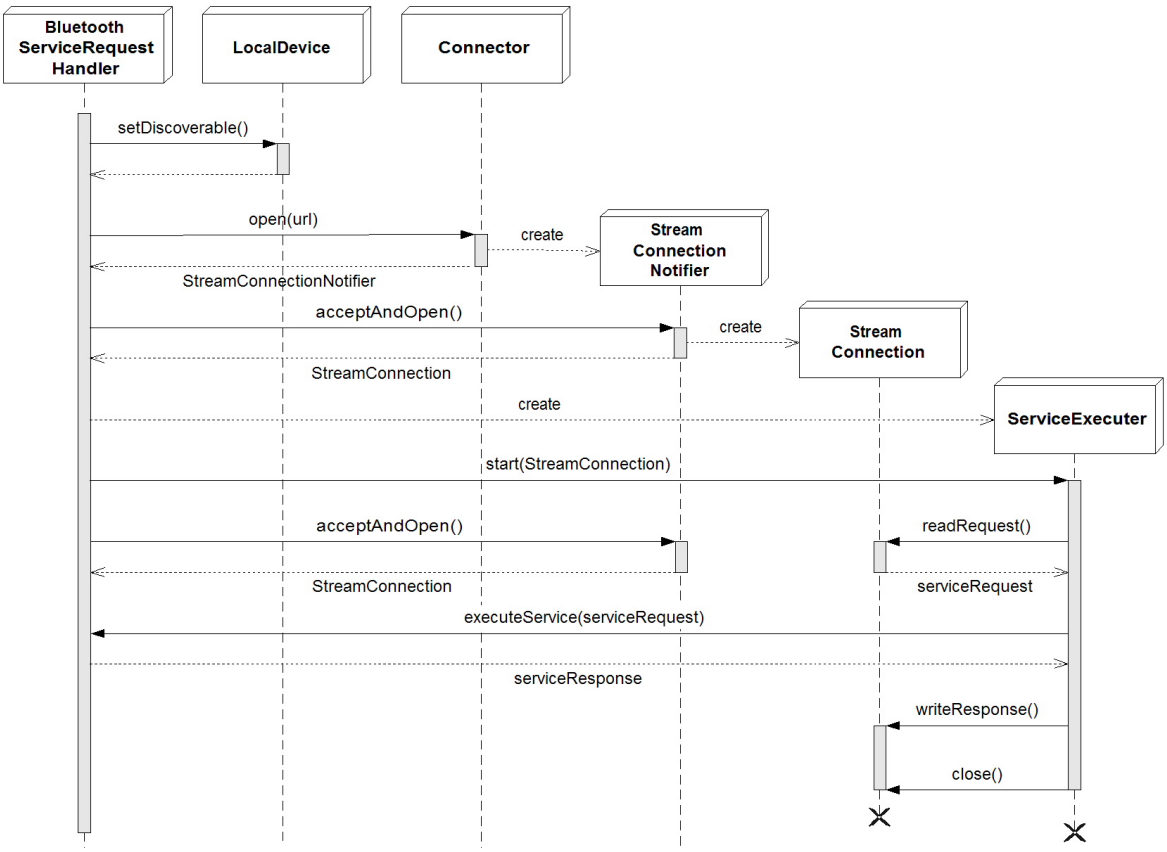

Abbildung 4.5: Sequenzdiagramm BluetoothServiceRequestHandler

4.1.5 Dienstnutzung

Auch die Komponente, die die Funktionalitäten für die Nutzung der in Abschnitt 4.1.4 beschriebenen angebotenen Dienste realisiert, muss eine Konformität bezüglich der J2ME vorweisen, da sie sowohl auf den Zugangspunkten, als auch auf den mobilen Endgeräten eingesetzt wird. Erst durch die Installation der Komponente wird die Verwendung von Diensten einer Kategorie ermöglicht, da diese nicht öffentlich publiziert werden, sondern für das Ansprechen die Kenntnis der Kategorie und des einzelnen Dienstes vorausgesetzt wird. Weil die Komponente als direktes Gegenstück zur Dienstbereitstellung agiert und dort die Verwendung des *btssp*-Protokolls festgelegt wurde, ist für die Bluetooth-Kommunikation auch hier die streamorientierte RFCOMM API zu verwenden (siehe Abschnitt 4.1.4.3).

Im Weiteren werden die für die Dienstnutzung wichtigen benötigten Klassen und deren Funktionsweise beschrieben. Die eigentliche Dienstnutzung wird dabei nochmals von der Bluetooth-Technologie abstrahiert, um einen Austausch oder eine einfache Erweiterung mit einer weiteren Technologie zu ermöglichen. Das an den wichtigen Stellen detaillierte Klas-

sendiagramm in Abbildung 4.6 gibt einen Überblick über die Abhängigkeiten und den Aufbau der Klassen. Ein vollständiges Klassendiagramm ist in der Abb. A.2 im Anhang zu finden.

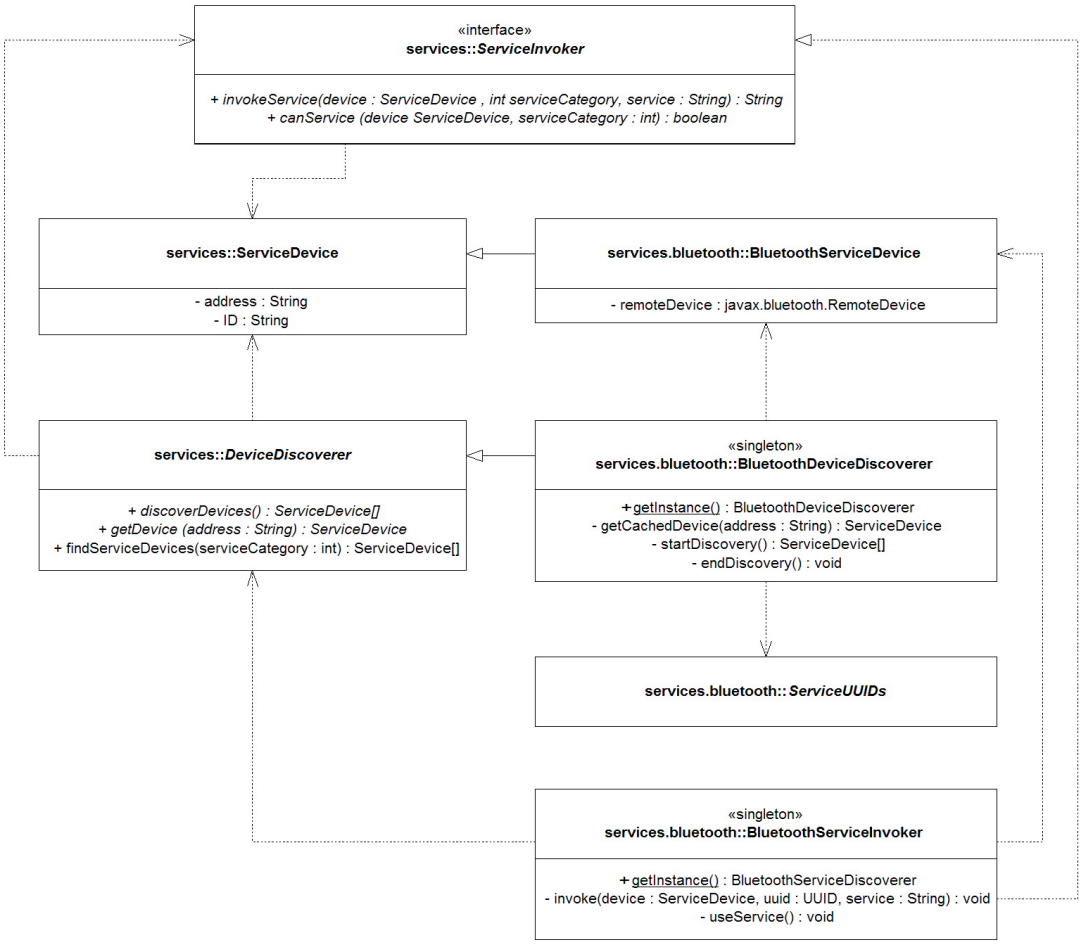

Abbildung 4.6: Klassendiagramm Dienstnutzung

4.1.5.1 Klasse ServiceDevice

Objekte dieser Klasse dienen zur Repräsentation eines mobilen oder stationären Gerätes, das grundsätzlich in der Lage ist, eine der drei in Abschnitt 4.1 beschriebenen Dienstkategorien für eine Nutzung anzubieten. Dabei kann es sich im Kontext der Plattform um einen Zugangspunkt oder ein mobiles Endgerät handeln. Jedes *ServiceDevice* besitzt eine Eigenschaft *address*, die mit einer eindeutigen Zeichenkette belegt ist und mit der es konkret angesprochen werden kann. Somit können aufgrund dieser Eigenschaft die repräsentierten Geräte auch eindeutig identifiziert werden. Eine weitere Eigenschaft *ID* wird genutzt, um ggf.

ein solches *ServiceDevice* einem Mitglied zuzuordnen, wenn es sich nach Inanspruchnahme des entsprechenden Dienstes (siehe Abschnitt 4.1.1.1) als ein an der Plattform teilnehmendes Gerät identifiziert hat.

4.1.5.2 Interface ServiceInvoker

Das Interface definiert den funktionalen Rahmen für die Inanspruchnahme eines durch ein *ServiceDevice* (siehe Abschnitt 4.1.5.1) zur Verfügung gestellten Dienstes. Die eigentliche Realisierung der physikalischen Dienstnutzung ist Aufgabe einer konkreten Klasse, die durch die Implementierung des Interfaces über dieses angesprochen wird. Im Falle der Entwicklung des Prototyps im Rahmen dieser Arbeit ist dies die im Abschnitt 4.1.5.5 behandelte Klasse *BluetoothServiceInvoker*, die hierfür die Bluetooth-Technologie verwendet.

An jeder Stelle, von der aus Dienste genutzt werden sollen, also sowohl Zugangspunkte, als auch mobile Endgeräte, wird ein *ServiceInvoker* benötigt. Mit ihm kann durch den Aufruf der Methode *invokeService()* ein bestimmter Dienst einer bestimmten Kategorie auf einen bestimmten *ServiceDevice* (siehe Abschnitt 4.1.5.1) angestoßen werden. Dabei werden als Parameter das *ServiceDevice*-Objekt, die Dienstkategorie repräsentiert durch einen int-Wert und die identifizierende Zeichenkette für den gewünschten Dienst übergeben. Der Rückgabewert des Methodenaufrufs ist die Zeichenkette, die der angesprochene Dienst als Antwort zurückgibt, oder *<NULL>*, falls das *ServiceDevice* nicht erreicht werden konnte oder den Dienst nicht zur Verfügung stellt. Um festzustellen, ob ein *ServiceDevice* überhaupt Dienste einer bestimmten Kategorie anbietet, kann dies durch die Benutzung der Methode *canService()* überprüft werden. Hier werden beim Aufruf das *ServiceDevice*-Objekt und ein int-Wert für die Dienstkategorie als Parameter übergeben. Rückgabewert ist *TRUE*, wenn das *ServiceDevice* Dienste der Kategorie zur Verfügung stellt, oder *FALSE* wenn dies nicht der Fall ist.

4.1.5.3 Klasse DeviceDiscoverer

Ein wichtiger Aspekt für den Betrieb der Plattform ist das Auffinden von Geräten, die auf Basis der verwendeten Technologie für die drahtlose Kommunikation ansprechbar sind. Die Nutzung eines Dienstes erfolgt entweder gezielt auf einem bestimmten Gerät, oder aber auf einem beliebigen erreichbaren Gerät, das den gewünschten Dienst bereitstellt. Demzufolge wird auf jedem dienstnutzenden Gerät ein *DeviceDiscoverer* benötigt. Durch die abstrakte Klasse *DeviceDiscoverer* wird zum einen der hierfür erforderliche Funktionsumfang durch die Definition abstrakter Methoden bestimmt, zum anderen werden konkrete implementierte Funktionalitäten an die Subklassen vererbt.

Die für die verwendete Drahtlos-Technologie spezifische Realisierung ist Aufgabe der jeweiligen von dieser Klasse abgeleiteten Subklassen. Basierend auf der im Rahmen des Prototypen verwendeten Bluetooth-Technologie ist das die in Abschnitt 4.1.5.6 behandelte Klasse *BluetoothDeviceDiscoverer*

Der Aufruf der Methode *discoverDevices()* erzeugt ein Array mit *ServiceDevices*, das die drahtlos ansprechbaren Geräte in Reichweite zu diesem Zeitpunkt repräsentiert. Unter diesen Geräten können auch solche sein, die zwar der verwendeten Drahtlos-Technologie mächtig sind, aber keine Dienste einer Kategorie im Sinne der Plattform anbieten. Mit der Methode *getDevice(address:String)* versucht der *DeviceDiscoverer* ein durch den Übergabeparameter eindeutig identifizierbares Gerät zu erreichen. Gelingt dies, wird stellvertretend für dieses Gerät ein *ServiceDevice*-Objekt zurückgegeben, andernfalls ist der Rückgabewert <*NULL*>.

Die konkret in dieser Klasse implementierte Methode *findServiceDevices()* dient zum Auffinden von Geräten in Reichweite, die eine bestimmte Dienstkategorie bedienen können. Als Übergabeparameter wird dabei ein die Kategorie repräsentierender int-Wert verwendet. Der *DeviceDiscoverer* sucht zuerst die Umgebung unter Verwendung der Funktionalität der *discoverDevices()*-Methode ab, um anschließend alle gefundenen Geräte dahingehend zu überprüfen, ob sie Dienste der gewünschten Kategorie zur Verfügung stellen. Hierfür bedient er sich der Funktionalitäten eines *ServiceInvoker*-Objektes (siehe Abschnitt 4.1.5.2). Rückgabewert ist ein Array mit den qualifizierenden *ServiceDevices* oder <*NULL*>, wenn kein entsprechendes Gerät gefunden werden konnte.

4.1.5.4 Klasse BluetoothServiceDevice

Diese Klasse spezialisiert die Klasse *ServiceDevice* aus Abschnitt 4.1.5.1 für die Verwendung von Bluetooth als Technologie für die drahtlose Kommunikation. Hier wird zusätzlich ein Objekt der Klasse *javax.bluetooth.RemoteDevice* gekapselt, welches bei der Verwendung der JAWBT ein konkretes entferntes Bluetooth-Gerät repräsentiert, das somit explizit angesprochen werden kann.

4.1.5.5 Klasse BluetoothServiceInvoker

Mit dieser Klasse wird ein konkreter *ServiceInvoker* auf Basis der Bluetooth-Technologie realisiert und dementsprechend dieses Interface aus Abschnitt 4.1.5.2 implementiert. Für das Ansprechen eines Bluetooth-Service wird der von JABWT zur Verfügung gestellte

javax.Bluetooth.DiscoveryAgent verwendet, der über das das lokale Bluetooth-Gerät repräsentierende Objekt der Klasse *javax.bluetooth.LocalDevice* referenzierbar ist. Für jede Serviceanfrage wird ein temporäres, als Piconet bezeichnetes Netzwerk mit der Hardware aufgebaut, die den Service zur Verfügung stellt. Ein solches Piconet kann aus zwei bis maximal acht aktiv kommunizierenden Teilnehmern bestehen, wobei sich die Geräte die Bandbreite teilen. Bis zu zehn Piconets können sich dabei ohne gegenseitige Störungen überlagern, wobei allerdings die Leistungsfähigkeit der vorhandenen Netze sinkt (Friedburg 2005). Da die Plattform sehr viele evtl. zeitgleich agierende Teilnehmer hat, sollten zumindest aus Sicht eines einzelnen Teilnehmers die Serviceanfragen sequenziell erfolgen, um einer möglichen Überlastung entgegenzuwirken. Aus diesem Grund findet das Singleton-Pattern[5] Verwendung, so dass alle Serviceanfragen über dasselbe Objekt stattfinden und dort synchronisiert werden können.

Um die beiden Funktionalitäten des *ServiceInvoker*-Interfaces zu realisieren, bedient sich das Objekt der privaten Methode *invoke(device:ServiceDevice, uuid:UUID, service:String)*. Der Parameter für die UUID des Bluetooth-Service wird vor dem Aufruf anhand der gewünschten Dienstkategorie mit Hilfe der Klasse *ServiceUUIDs* ermittelt. Für den Fall, dass das übergebene *ServiceDevice* kein wie in Abschnitt 4.1.5.4 beschrieben konkretes *javax.bluetooth.RemoteDevice* kapselt, wird mit Hilfe des *DeviceDiscoverers* (siehe Abschnitt 4.1.5.3) versucht, anhand der identifizierenden Adresse das entsprechende Gerät ausfindig zu machen. Mit der Referenz auf das *RemoteDevice* wird mit der Methode *searchServices()* des *DiscoveryAgents* eine Suche mit der Bluetooth-Hardware nach einem Service auf diesem Gerät gestartet. Dabei ist der Aufruf nicht blockierend, sondern wird in einem neuen parallelen Prozess gestartet. Ob ein mobiles Endgerät aufgefunden werden kann, ist hierbei allerdings vom Benutzer dieses Gerätes beeinflussbar. Es muss so konfiguriert sein, dass es für andere Bluetooth-Geräte „sichtbar“ ist. Die Benachrichtigung über etwaige Ergebnisse erfolgt dabei über das Interface *javax.bluetooth.DiscoveryListener*, das von der Klasse *BluetoothServiceInvoker* implementiert wird. Das *BluetoothServiceInvoker*-Objekt wird beim Aufruf als Listener mit übergeben und blockiert seine weitere Ausführung selbst, bis es die Mitteilung vom *DiscoveryAgent* erhält, dass die Suche abgeschlossen ist. Wird während der Suche ein korrespondierender Service gefunden, werden dem Listener über den *DiscoveryAgent* Referenzen vom Typ *javax.bluetooth.ServiceRecord* übergeben, die unter anderem die URL bereitstellen, mit denen der jeweilige Service angesprochen werden kann.

Ging es bei der Anfrage nicht nur um die Information, ob ein Service (eine Dienstkategorie) generell auf einem Gerät angeboten wird, sondern um die konkrete Nutzung eines Dienstes, wird nach Beendigung der Service-Suche mit der Methode *useService()* der vollständige Dienst-Request mit Hilfe einer *javax.microedition.io.StreamConnection* an das entfernte

[5] das Singleton-Pattern stellt sicher, dass immer nur eine einzige Instanz der Klasse in der virtuellen Maschine existiert (Gamma u. a. 2004)

Bluetooth-Gerät übermittelt. Die Funktionalität gestaltet sich dabei analog zu der Vorgehensweise bei der Dienstbereitstellung durch die Klasse *BluetoothServiceRequestHandler* aus Abschnitt 4.1.4.3.

Das Zusammenspiel mit den Klassen der JABWT und der CLDC ist nochmals grafisch als Klassen- bzw. Sequenzdiagramm in den Abbildungen 4.7 und 4.8 skizziert.

4.1.5.6 Klasse BluetoothDeviceDiscoverer

Die Klasse spezialisiert konkret einen in Abschnitt 4.1.5.3 beschriebenen *DeviceDiscoverer* auf Basis der Bluetooth-Technologie. Da die Ermittlung aller sich in Reichweite befindlichen Bluetooth-Geräte jedes Mal eine nicht unerhebliche Zeitspanne in Anspruch nimmt, werden immer die in der letzten Auffindungsphase gefundenen Geräte in einem Cache vorgehalten. Begründet wird dies durch die Annahme, dass es nicht unwahrscheinlich ist, dass sich ein Gerät auch noch geraume Zeit nach einer durchgeführten Auffindungsphase in Reichweite befindet. Soll innerhalb der Plattform nicht ein beliebiges, sondern ein bestimmtes *ServiceDevice* angesprochen werden, besteht so die Möglichkeit des Versuches, das Gerät zu kontaktieren, ohne zu diesem Zeitpunkt vorher nochmals alle Geräte im Umfeld zu ermitteln. Aus diesem Grunde wird für die Klasse das Singleton-Pattern verwendet, damit die Verwaltung der zuletzt gefundenen Geräte an einer zentralen Stelle erfolgt. Die von der Superklasse vorgegebenen hier implementierten Funktionalitäten *getDevice(address:String)* und *discoverDevices()* müssen somit synchronisiert sein, da sie auf denselben Cache zugreifen.

Die Methode *discoverDevices()* delegiert an die private Methode *startDiscovery()*, in der zuerst der Cache mit den Ergebnissen der letzten Auffindungsphase gelöscht wird. Für das Auffinden von Bluetooth-Geräten wird der von JABWT zur Verfügung gestellte *javax.Bluetooth.DiscoveryAgent* verwendet, der über das das lokale Bluetooth-Gerät repräsentierende Objekt der Klasse *javax.bluetooth.LocalDevice* referenzierbar ist. Mit der Methode *startInquiry()* des *DiscoveryAgents* wird eine Suche nach Bluetooth-Geräten gestartet, die sich in Kommunikationsreichweite befinden. Dabei ist der Aufruf nicht blockierend, sondern wird in einem neuen parallelen Prozess gestartet. Die Benachrichtigung über etwaige Ergebnisse erfolgt dabei über das Interface *javax.bluetooth.DiscoveryListener*, das von der Klasse *BluetoothDeviceDiscoverer* implementiert wird. Das *BluetoothDeviceDiscoverer*-Objekt wird beim Aufruf als Listener mit übergeben und blockiert seine weitere Ausführung selbst, bis es die Mitteilung vom *DiscoveryAgent* erhält, dass die Suche abgeschlossen ist. Wird während der Suche ein Bluetooth-Gerät gefunden, werden dem Listener vom *DiscoveryAgent* Referenzen vom Typ *javax.bluetooth.RemoteDevice* übergeben, mit denen die jeweilige Bluetooth-Geräte angesprochen werden können. Jedes gefundene *RemoteDevice* wird in

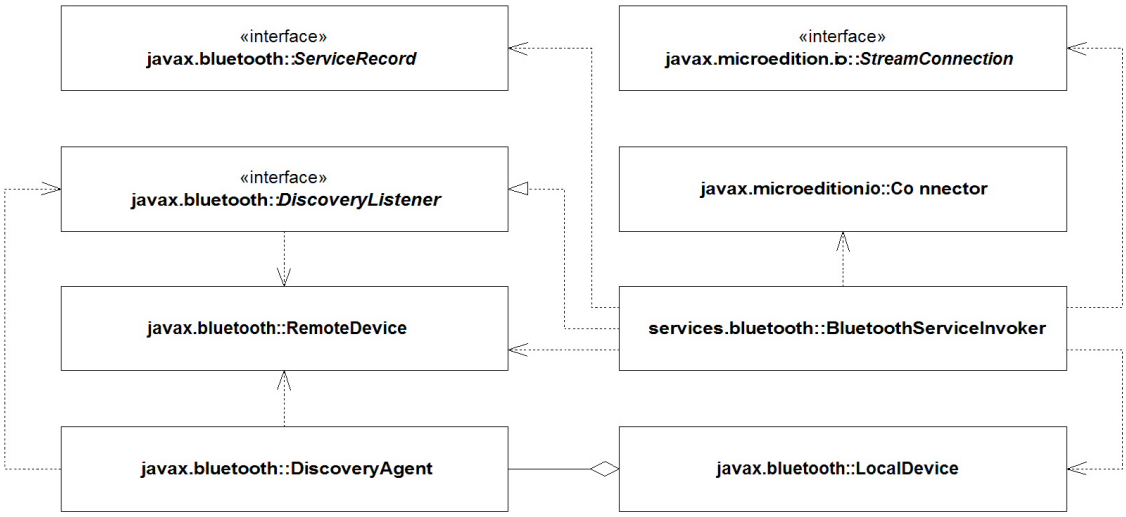

Abbildung 4.7: Klasse BluetoothServiceInvoker und JSR-82 und CLDC 1.1

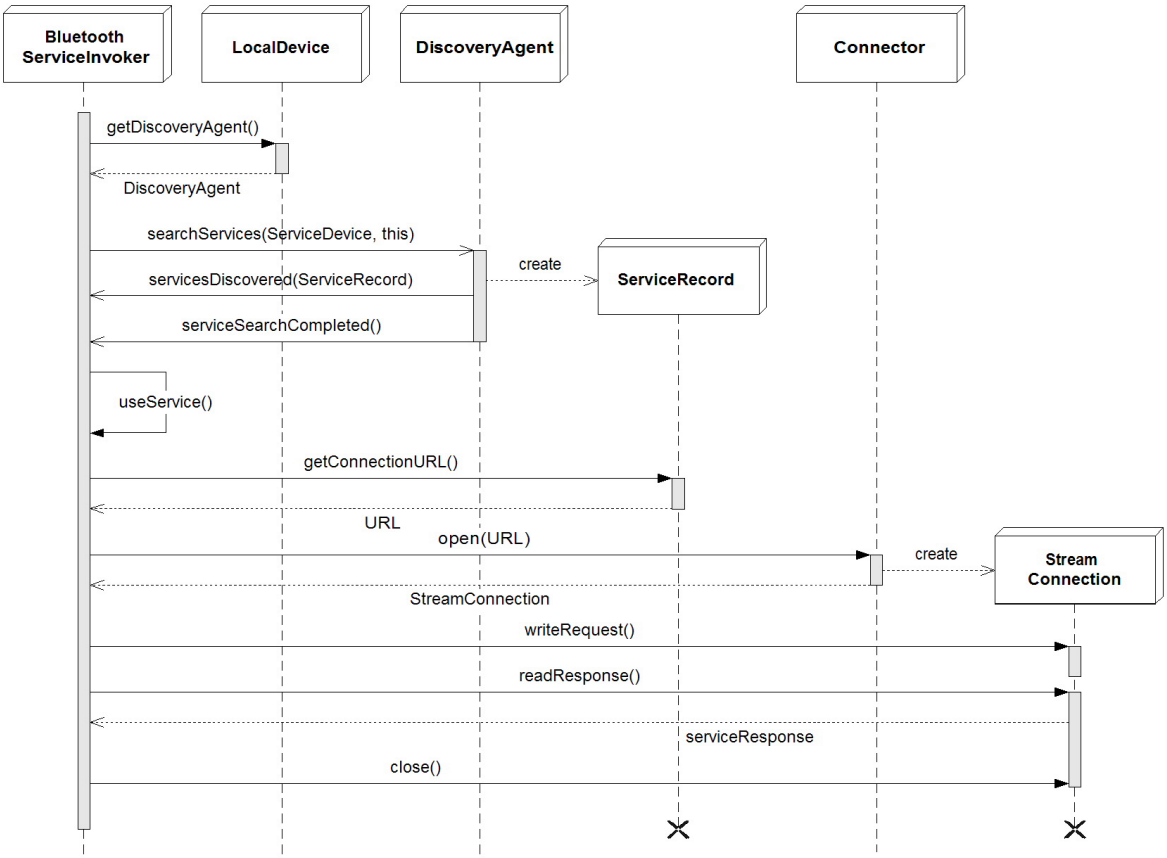

Abbildung 4.8: Sequenzdiagramm BluetoothServiceInvoker

einem *BluetoothServiceDevice* gekapselt und dem Cache hinzugefügt. Nach Abschluss der Suche werden alle *ServiceDevices* als Array zurückgegeben.

Das Zusammenspiel mit den Klassen der JABWT ist nochmals grafisch als Klassen- bzw. Sequenzdiagramm in den Abbildungen 4.9 und 4.10 skizziert.

Die Spezifikation JSR-82 sieht vor, dass ein *javax.bluetooth.DiscoveryAgent* auch die Methode *selectService(uuid:UUID, security:int, master:boolean)* anbietet, die eine URL zurückgibt, mit der ein gewünschter Bluetooth-Service auf einem beliebigen Gerät in Reichweite angesprochen werden kann. Diese Funktionalität wäre im Rahmen der Entwicklung des Prototyps der Plattform hilfreich gewesen, wurde aber zum Zeitpunkt der Entstehung dieser Arbeit vom verwendeten Bluetooth-Stack für die Zugangspunkte (siehe Abschnitt 4.4.3) zumindest noch nicht implementiert. Als Workaround wird deshalb die geerbte Funktionalität der Methode *findServiceDevices(serviceCategory:int)* verwendet, um mobile Endgeräte zu finden, die einen bestimmte Bluetooth-Service anbieten. Geht es um die Nutzung von globalen Diensten und sind Zugangspunkte so angeordnet, dass sie sich bezüglich ihrer Reichweite der drahtlosen Kommunikation überschneiden, kann von diesen Geräten dann ein beliebiges angesprochen werden, da die Ausführung des Dienstes sowieso an eine zentrale Stelle des Systems delegiert wird. Im Rahmen der mobilen Dienste wird diese Funktionalität nur bei der Mitgliederbeobachtung (siehe 4.2.1.1) eingesetzt, bei der dann alle ermittelten mobilen Endgeräte angesprochen werden. Wird nach einem Gerät mit einem lokalen Dienst gesucht, kann davon ausgegangen werden, dass von einem bestimmten Standort des Nutzers aus nur maximal ein Gerät gefunden wird, da jeder lokale Dienst innerhalb der durch das System abgedeckten Fläche einzigartig ist.

4.2 Clubsystem

Dieser Abschnitt behandelt die wichtigen Funktionalitäten, die auf der Seite des Clubnetzwerks realisiert werden müssen. Ob die jeweilige Implementation J2ME-konform gehalten werden muss, ist davon abhängig, auf welcher Art von Hardware die Klassen eingesetzt werden. Während für die Komponente *Cluborganisation* eine JAVA 2 SE vorausgesetzt wird, ist es denkbar, dass die anderen Komponenten sowohl auf einem handelsüblichen PC oder Notebook, als auch kleiner dimensionierter, nur der CLDC-Spezifikation genügender Hardware Verwendung finden. So wäre es z.B. möglich, an Stellen, an denen noch kein Computer für die tägliche Arbeit im Clubareal positioniert ist, einen Bluetooth-fähigen PDA mittels WLAN drahtlos in das Clubnetzwerk zu integrieren und als Zugangspunkt nutzbar zu machen. Um Möglichkeiten dieser Art offen zu halten, soll bei auf Zugangspunkten eingesetzten Komponenten eine Konformität bezüglich J2ME gewährleistet sein.

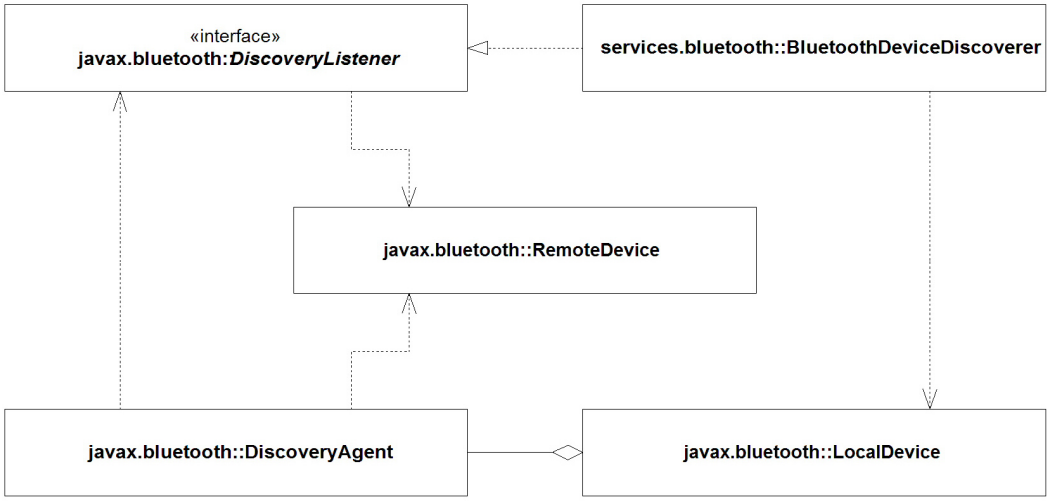

Abbildung 4.9: Klasse BluetoothDeviceDiscoverer und JSR-82

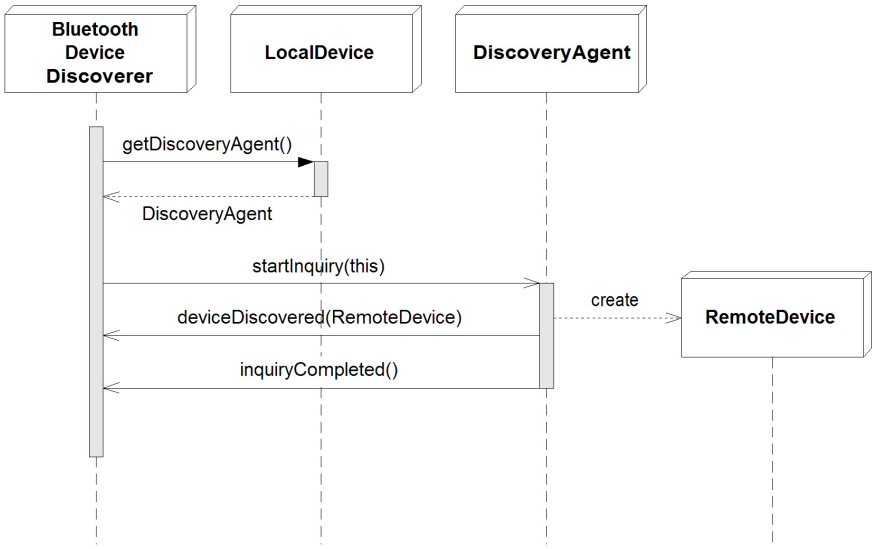

Abbildung 4.10: Sequenzdiagramm BluetoothDeviceDiscoverer

4.2.1 Mitgliederbeobachtung

Die Kenntnis des Aufenthaltsortes der mobilen Teilnehmer des Systems spielt eine entscheidende Rolle. Soll ein bestimmtes mobiles Endgerät angesprochen werden, um die Terminkapazität eines Mitglieds zu erfragen, oder diesem einen Veranstaltungsvorschlag zu unterbreiten (siehe Anwendungsfälle in den Abschnitten 3.3.2.3 und 3.3.2.2), erfolgt dies drahtlos über einen Zugangspunkt. Dafür ist es notwendig, dass sich das mobile Endgerät in der Reichweite dieses Zugangspunktes befindet. Da die Mitglieder ihre Position im Laufe der Zeit verändern, muss für die Nutzung der mobilen Dienste auf ihren Endgeräten zu einem bestimmten Zeitpunkt ein geeigneter Zugangspunkt verwendet werden. Um nicht wahllos die Kontaktaufnahme über alle Zugangspunkte „auszuprobieren“ zu müssen, soll von vornherein ein Zugangspunkt für den Versuch verwendet werden, bei dem die Wahrscheinlichkeit sehr hoch ist, dass sich das entsprechende Endgerät in seiner Reichweite befindet. Zusätzlich spielt die Position eines Mitglieds eine Rolle für die gezielte Unterbreitung von Veranstaltungsvorschlägen durch das zentrale Verwaltungssystem des Clubs.

Das System muss demnach einen Mechanismus für die Positionsbestimmung der Mitglieder bereitstellen, der eine möglichst aktuelle Information über den jeweiligen Aufenthaltsort geben kann. Primär muss dabei die Position dem zentralen Verwaltungssystem des Clubs und nicht dem einzelnen Mitglied gegenwärtig sein. Grundsätzlich können die Verfahren zur Positionsbestimmung in zwei Kategorien eingeteilt werden (Roth 2002). Hier wird das sog. *Tracking*-Verfahren angewendet, bei dem die Position von einem Sensorennetzwerk bestimmt wird und die ermittelte Position damit vorerst nur dem Positionierungssystem vorliegt. Das Verfahren des *Positioning*, bei dem der mobile Benutzer seine Position selbst ermittelt und dann für die weitere Verarbeitung an das System überträgt, ist für diesen Einsatzfall weniger geeignet. Die Endgeräte können aufgrund ihrer Ausstattung auch nur eine netzwerkgestützte Positionsbestimmung vornehmen. Zwar könnte mit der Inanspruchnahme eines einzigen Dienstes die Position sowohl dem zentralen System, als auch dem mobilen Endgerät publik gemacht werden, jedoch würde dadurch die vorherige Suche nach Geräten, die diesen Dienst zur Verfügung stellen, in den Zuständigkeitsbereich der mobilen Endgeräte fallen. Um die Aufenthaltsorte im System aktuell zu halten, muss die Ermittlung periodisch in relativ kurzen Zeitintervallen stattfinden. Da die Nutzung einer drahtlosen Kommunikation grundsätzlich einen erhöhten Energieaufwand bedeutet und die mobilen Endgeräte gegenüber den meisten Zugangspunkten bezüglich ihrer Energieressourcen benachteiligt sind, ist es sinnvoll, die Endgeräte in dieser Hinsicht soweit wie möglich zu entlasten.

Im Weiteren werden die für die Mitgliederbeobachtung wichtigen benötigten Klassen und deren Funktionsweise beschrieben. Für den Ablauf werden dabei einige Funktionalitäten der in Abschnitt 4.1.5 behandelten Komponente für die Dienstnutzung in Anspruch genommen. Das an den wichtigen Stellen detaillierte Klassendiagramm in Abbildung 4.11 gibt einen

Überblick über die Abhängigkeiten und den Aufbau der Klassen. Ein vollständiges Klassendiagramm ist in der Abb. A.3 im Anhang zu finden.

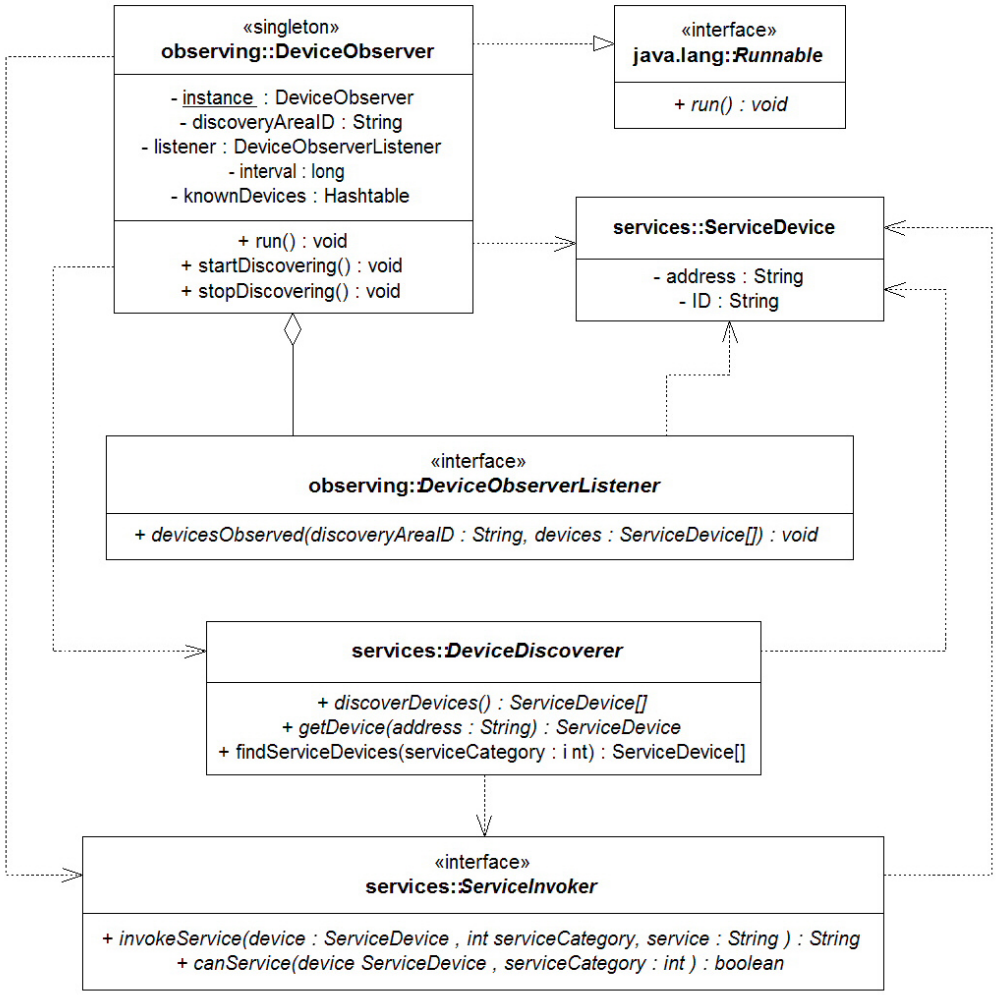

Abbildung 4.11: Klassendiagramm Mitgliederbeobachtung

4.2.1.1 Klasse DeviceObserver

Auf jedem Zugangspunkt des Clubnetzwerks ist eine Instanz dieser Klasse dafür zuständig, in einem definierten zeitlichen Abstand in Reichweite befindliche Bluetooth-Geräte aufzufinden. In diesem Zusammenhang sind nur solche Geräte interessant, die Teilnehmer der Plattform sind und den Service der in Abschnitt 4.1.1 beschriebenen mobilen Dienste anbieten. Anders formuliert sind dies also die mobilen Endgeräte der Mitglieder, deren Aufenthaltsort

ständig beobachtet werden soll. Die Zeitspanne in Millisekunden zwischen der Ausführung von Auffindungsprozessen kann dabei mit der Methode *setInterval(interval:long)* konfiguriert werden, um das Intervall Anhand der Bedürfnisse zu optimieren. Eine Ausdehnung des Intervalls reduziert den Gesamtumfang der drahtlosen Kommunikation innerhalb des Systems und den damit verbundenen Energieverbrauch, was insbesondere den mobilen Endgeräten zugute kommt. Das Intervall darf allerdings nicht so groß gewählt werden, dass die Aktualität der Aufenthaltsorte der Mitglieder leidet und die Information wertlos wird.

Mit dem Aufruf der Methode *startDiscovering()* wird ein neuer Prozess gestartet, der zuerst mit Hilfe der Funktionalität eines in Abschnitt 4.1.5.3 beschriebenen *DeviceDiscoverer*-Objektes alle *ServiceDevices* ermitteln lässt, die die mobilen Dienste anbieten. Anschließend wird auf jedem dieser gefundenen *ServiceDevices* der *MemberIdentificationService* (siehe Abschnitt 4.1.1.1) angestoßen, und die so ermittelte ID als Eigenschaft des *ServiceDevice*-Objekts gesetzt, um zukünftig dieses einem konkreten Mitglied zuordnen zu können. Dazu wird sich der Funktionalität des *ServiceInvokers* aus Abschnitt 4.1.5.2 bedient. Alle einmal identifizierten *ServiceDevices* werden anhand ihrer Adresse vorgehalten, so dass bei einem späteren, erneuten Auffinden keine weitere Dienstausführung auf dem mobilen Endgerät nötig ist. So wird der Umfang der drahtlosen Kommunikation vermindert und wiederum die Energieressourcen der Endgeräte geschont und die benötigte Zeit für einen Auffindungsprozess verringert. Ausgerichtet an dem Observer-Pattern[6] werden letztlich die mit den IDs der Mitglieder angereicherten *ServiceDevice*-Objekte an ein beliebiges Objekt über das Interface *DeviceObserverListener* als Array weitergegeben, das nachfolgend die Informationen über die in der Umgebung gefundenen und identifizierten mobilen Endgeräte nach der Benachrichtigung entsprechend verarbeiten kann. Nun wartet der Prozess so lange, bis die definierte Zeitspanne des Intervalls verstrichen ist, um anschließend wieder von vorne zu beginnen. Durch den Aufruf der Methode *stopDiscovering()* kann der Prozess und damit der Mechanismus der wiederkehrenden Auffindungsphasen jederzeit terminiert werden.

4.2.2 Cluborganisation

Die Komponente *Cluborganisation* wird im Rahmen dieser Arbeit nur rudimentär entwickelt. Um die Komplexität der Komponente möglichst gering zu halten, beschränkt sich der Funktionsumfang im Wesentlichen auf ein Maß, mit dem die Funktionstüchtigkeit des Prototypen der Plattform demonstriert werden kann. Die elementaren Bestandteile des Clubnetzwerkes sind eine zentrale Instanz der Klasse *Club*, mehrere Instanzen der Klasse *Servicepoint*, die jeweils auf einem Zugangspunkt des Clubnetzwerks für den dort ablaufenden Funktionsumfang verantwortlich sind, sowie für jeden Zugangspunkt eine Instanz der Klasse *Zone*, die

[6] Das Observer-Pattern definiert eine 1-zu-n Abhängigkeit zwischen Objekten, so dass die Änderung des Zustands eines Objektes dazu führt, dass alle abhängigen Objekte benachrichtigt und automatisch aktualisiert werden (Gamma u. a. 2004).

das Areal repräsentiert, das durch die Reichweite bezüglich der drahtlosen Kommunikation von einem Zugangspunkt abgedeckt wird. Eine Zone referenziert dabei über das Interface *ServiceInvoker* die zugehörige *Servicepoint*-Instanz, um gezielt mobile Dienste auf Endgeräten in Reichweite des entsprechenden Zugangspunktes ansprechen zu können. Die einzelnen Aufgaben und die Zusammenarbeit dieser Klassen während des Betriebes des Systems wird in den folgenden Abschnitten verdeutlicht. Im Vorwege ist in Abbildung 4.12 dargestellt, über welche implementierte Mechanismen die Objekte der Klassen kommunizieren.

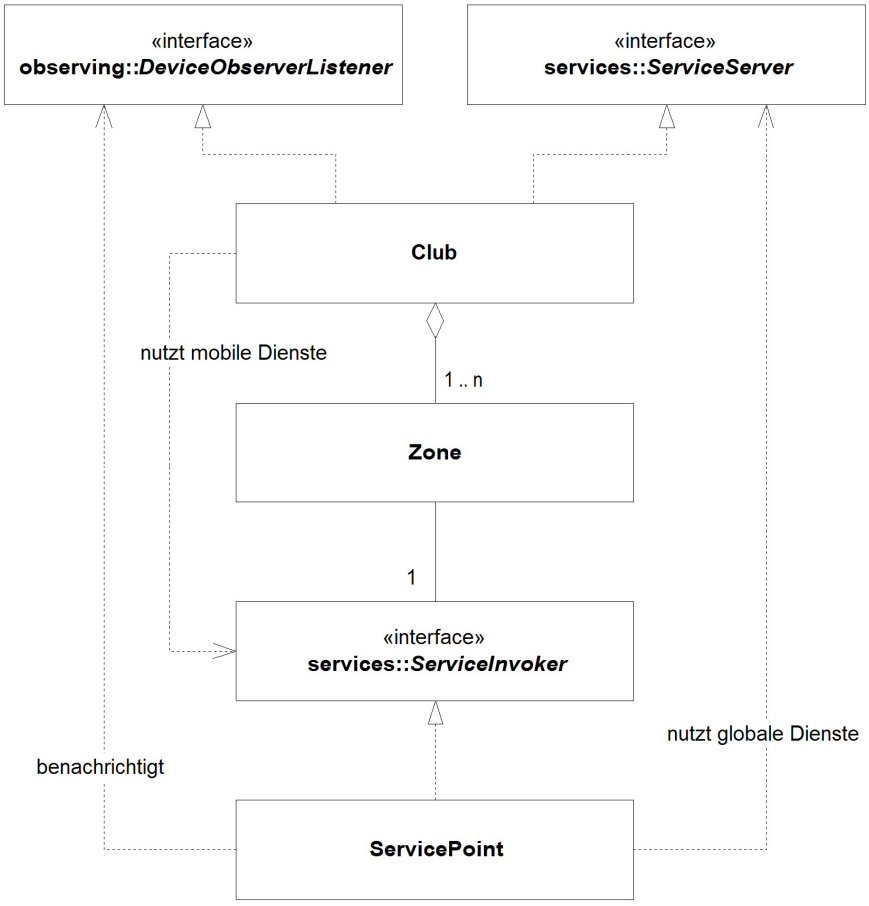

Abbildung 4.12: Verbindung zwischen Zentralsystem und Zugangspunkten

4.2.2.1 Mitglieder- und Veranstaltungsverwaltung

An einer zentralen Stelle des Clubnetzwerks werden sowohl die Mitglieder des Clubs, als auch das Programm der angebotenen Veranstaltungen gepflegt. Dieses Zentrum wird durch die Klasse *Club* repräsentiert. Exemplarisch werden Klassen für Mitglieder (*Member*) und

Veranstaltungen (*Event*) sowie für deren Verwaltung implementiert. Das Clubareal ist in verschiedene Zonen unterteilt, in denen sich die Mitglieder aufhalten können. Dabei wird davon ausgegangen, dass sich je nach Verteilung der Zugangspunkte die Flächen der Zonen theoretisch auch überschneiden könnten. Ein Mitglied kann sich demnach zum selben Zeitpunkt in mehreren Zonen aufhalten, so dass im Rahmen einer späteren Weiterentwicklung der Plattform eine noch filigranere Bestimmung des Aufenthaltsortes ermöglicht wird. Eine Zone dient auch als Referenzpunkt für den Ort, an dem eine Veranstaltung stattfindet. Die Assoziationen der Klassen untereinander sind in der Abbildung 4.13 skizziert. Auf eine detailliertere Beschreibung der dabei simpel zu realisierenden Funktionalitäten wird an dieser Stelle verzichtet.

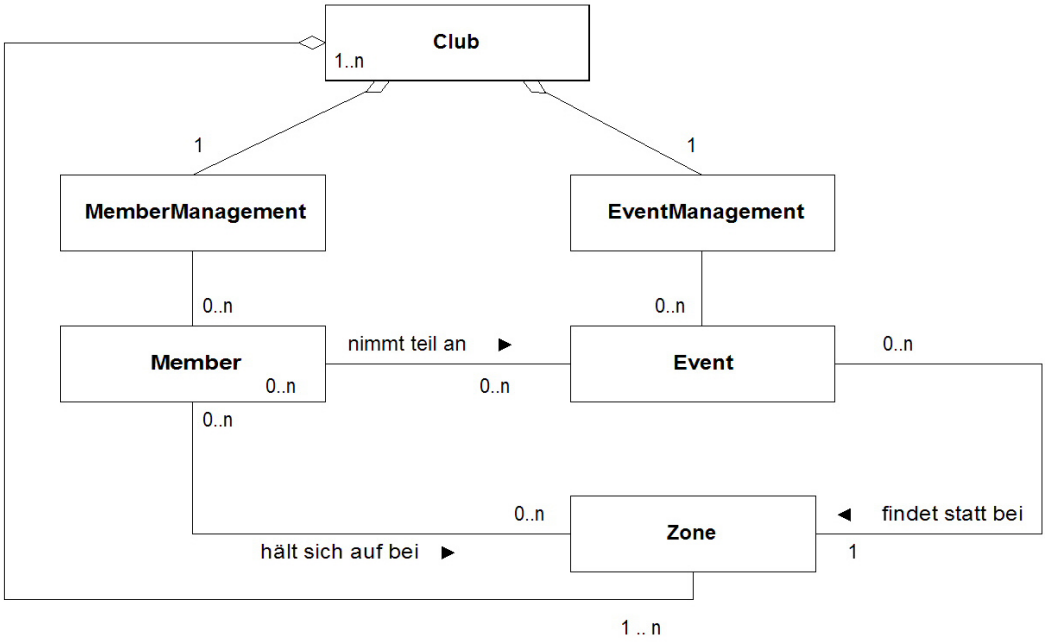

Abbildung 4.13: Klassendiagramm Clubverwaltung

4.2.2.2 Tracking der Mitglieder

Jede Instanz von *ServicePoint* verfügt über einen *DeviceObserver* und einen *DeviceDiscoverer*, die in den Abschnitten 4.2.1.1 und 4.1.5.3 beschrieben sind. Die Instanz der Klasse *Club* implementiert das Interface *DeviceObserverListener* und wird jedem *ServicePoint*-Objekt als zentrale Anlaufstelle für die Benachrichtigung über entdeckte mobile Endgeräte mitgeteilt und dort als Listener des *DeviceObservers* verwendet. Nach jeder

Benachrichtigung durch den *DeviceObserver* auf dem Zugangspunkt muss das repräsentierende *Zone*-Objekt auf einen aktuellen Stand gebracht werden. Jede Zone beinhaltet eine Liste der Mitglieder, deren Anwesenheit bei der letzten, in dieser Zone durchgeführten Mitgliederbeobachtung festgestellt wurde. Zusätzlich wird mit Hilfe eines Timestamps der Zeitpunkt der Benachrichtigung in der Zone festgehalten, so dass die Aktualität der Information über Aufenthaltsorte verschiedener Zonen verglichen werden kann. Anhand der IDs der bei der Benachrichtigung übergebenen *ServiceDevices* wird über die Mitgliederverwaltung das korrespondierende Mitglied herausgesucht und bei diesem eine Referenz auf das *ServiceDevice* hinterlegt. Damit kann künftig dieses konkrete mobile Endgerät angesprochen werden, wenn eine Dienstnutzung bei einem bestimmten Mitglied erforderlich ist. Die für das Tracking verwendeten Klassen und deren Verteilung auf Zugangspunkt und Zentralsystem sind nochmals in der Abbildung 4.14 dargestellt.

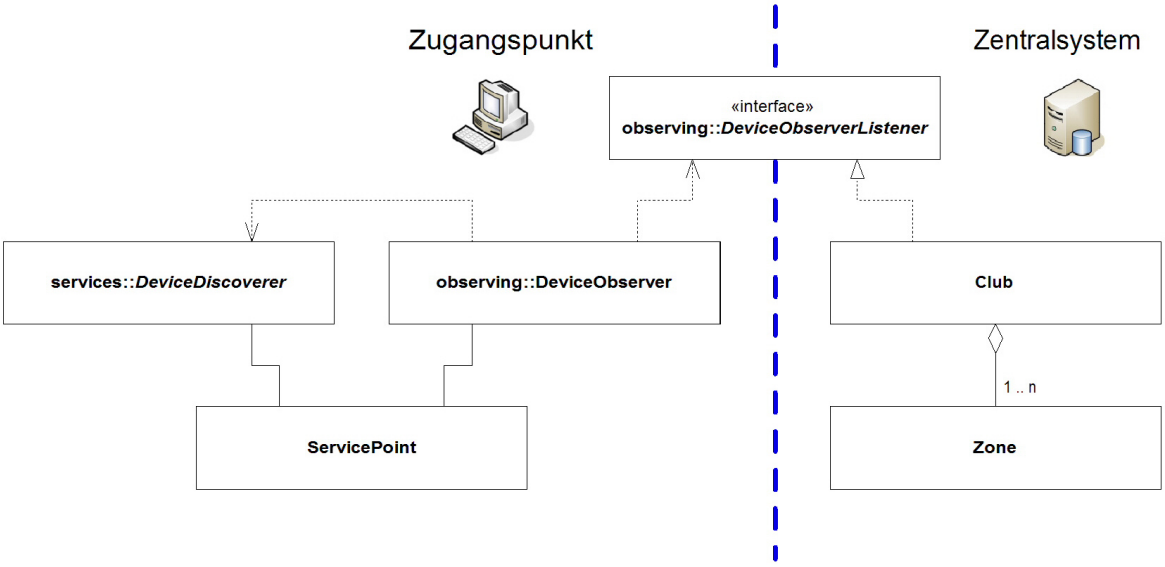

Abbildung 4.14: Klassendiagramm Tracking

4.2.2.3 Bereitstellung und Ausführung globaler Dienste

Die Nutzung globaler Dienste steht auf allen Zugangspunkten des Clubnetzwerks zur Verfügung und wird von den mobilen Endgeräten in Anspruch genommen (siehe Abschnitt 4.1.2). Kennzeichnendes Merkmal solcher Dienste ist, dass sie auf zentrale Daten der Clubverwaltung angewiesen sind. Daher erfolgt auch hier eine räumliche Trennung von *ServiceRequestHandler* und der eigentlichen Implementation des *ServiceServer*. Die Funktionsweise und Zusammenarbeit dieser Klassen im Rahmen der Dienstbereitstellung wurde

bereits in Abschnitt 4.1.4 erläutert. Auf jedem Zugangspunkt beinhaltet die Instanz der Klasse *ServicePoint* einen *ServiceRequestHandler* für die Entgegennahme von Anfragen für globale Dienste. Diesem wird für die Zusammenarbeit keine konkrete Implementation eines *ServiceServer* zugeordnet, sondern stellvertretend das Objekt der Klasse *Club*, welches dieses Interface implementiert. Die tatsächliche Realisierung in Form eines *ServiceServerImpl* ist Bestandteil des zentralen *Club*-Objektes und kann somit über dieses von allen Zugangspunkten aus angesprochen werden.

Stellvertretend für die Kategorie der globalen Dienste wurde der in Abschnitt 4.1.2.1 beschriebene Dienst implementiert, mit dem ein Mitglied über sein mobiles Endgerät seine Zusage bezüglich einer Teilnahme an einer bestimmten Veranstaltung des Clubs machen kann. Anhand der Parameter im Servicerequest kann der Dienst eine von *Club* zur Verfügung gestellte Funktionalität nutzen, mit der das entsprechende Mitglied der entsprechenden Veranstaltung als Teilnehmer hinzugefügt wird.

Die für die Bereitstellung und Ausführung der globalen Dienste verwendeten Klassen und deren Verteilung auf Zugangspunkt und Zentralsystem sind nochmals in der Abbildung 4.15 dargestellt.

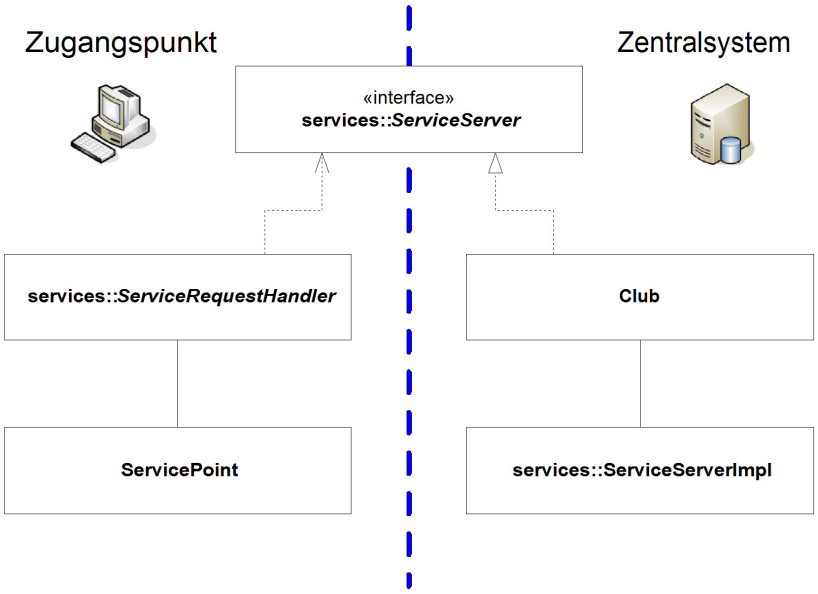

Abbildung 4.15: Klassendiagramm Bereitstellung globaler Dienste

4.2.2.4 Nutzung mobiler Dienste

Auch für die Nutzung der mobilen Dienste auf den Endgeräten der Mitglieder erfolgt eine Aufteilung der benötigten Klassen auf das Zentralsystem und die Zugangspunkte. Der Bedarf

für die Inanspruchnahme dieser Dienste entsteht zu bestimmten Zeitpunkten im Organisationsablauf während des Betreibens des Clubs, die Nutzung wird durch die Instanz der Klasse *Club* initiiert. Am Beispiel des hier entwickelten Prototyps ist dies eine Situation, in der für eine Veranstaltung mit vakanten Plätzen noch Mitglieder als Teilnehmer gesucht werden. Um ein geeignetes Mitglied zu finden, wird der in Abschnitt 4.1.1.2 behandelte mobile Dienst verwendet, mit dem auf dem Endgerät des Mitglieds festgestellt werden kann, ob dort für den Veranstaltungszeitraum noch keine Termine im Kalender des Endgerätes vermerkt sind[7]. Um einem Mitglied schließlich die Teilnahme an einer Veranstaltung vorzuschlagen, wird der mobile Dienst aus Abschnitt 4.1.1.3 angesprochen. Der dritte implementierte mobile Dienst zur Identifikation eines Mitglieds (siehe Abschnitt 4.1.1.1) wird nicht vom Zentralsystem, sondern direkt von den einzelnen Zugangspunkten im Zuge der in Abschnitt 4.2.1 beschriebenen Mitgliederbeobachtung genutzt.

Um einen Dienst auf dem mobilen Endgerät eines bestimmten Mitglieds zu nutzen, kann stellvertretend über die Zonen der entsprechende konkrete *ServiceInvoker* des *ServicePoints* auf einem Zugangspunkt angesprochen werden. Dabei wird eine Zone ausgewählt, von der aufgrund der Mitgliederbeobachtung angenommen wird, dass sich das Mitglied in dieser aufhält. Die Funktionsweise des *ServiceInvoker* wurde in Abschnitt 4.1.5.2 ausführlich beschrieben. Das für den Funktionsaufruf benötigte *ServiceDevice*-Objekt kann über die Mietgliedsverwaltung von dem das Mitglied repräsentierenden *Member*-Objekt erlangt werden.

Die für die Nutzung der mobilen Dienste verwendeten Klassen und deren Verteilung auf Zugangspunkt und Zentralsystem sind nochmals in der Abbildung 4.16 dargestellt.

4.2.3 Interne Kommunikation

Wie aus dem vorhergehenden Abschnitt ersichtlich, können die Klassen der Komponenten, die auf der Seite des Clubnetzwerkes eingesetzt werden, räumlich voneinander getrennt sein, indem sie auf Hardware an unterschiedlichen Orten eingesetzt werden. Die zentrale Verwaltung des Clubs arbeitet für das Tracking der Mitglieder, die Bereitstellung globaler Dienste des Clubs und die Nutzung mobiler Dienste auf den Endgeräten der Mitglieder mit den auf dem Clubareal verteilten Zugangspunkten zusammen. Die Anbindung der Klassen untereinander wird dabei grundsätzlich über ein Interface realisiert (siehe hierzu auch die Abbildungen 4.14, 4.15 und 4.16). Im Rahmen der Entwicklung des Prototyps wird für die Tests innerhalb einer Laborumgebung vorerst nur ein einziger Zugangspunkt verwendet, der mit derselben Hardware realisiert ist, auf der auch das Zentralsystem eingesetzt ist (siehe

[7] Die Eignung eines Mitglieds für einen Veranstaltungsvorschlag kann selbstverständlich auch von dessen Position abhängig gemacht werden. Für die Bewertung einer Eignung sind unterschiedliche Vorgehensweisen denkbar, deren detaillierte Diskussion nicht Bestandteil dieser Arbeit ist.

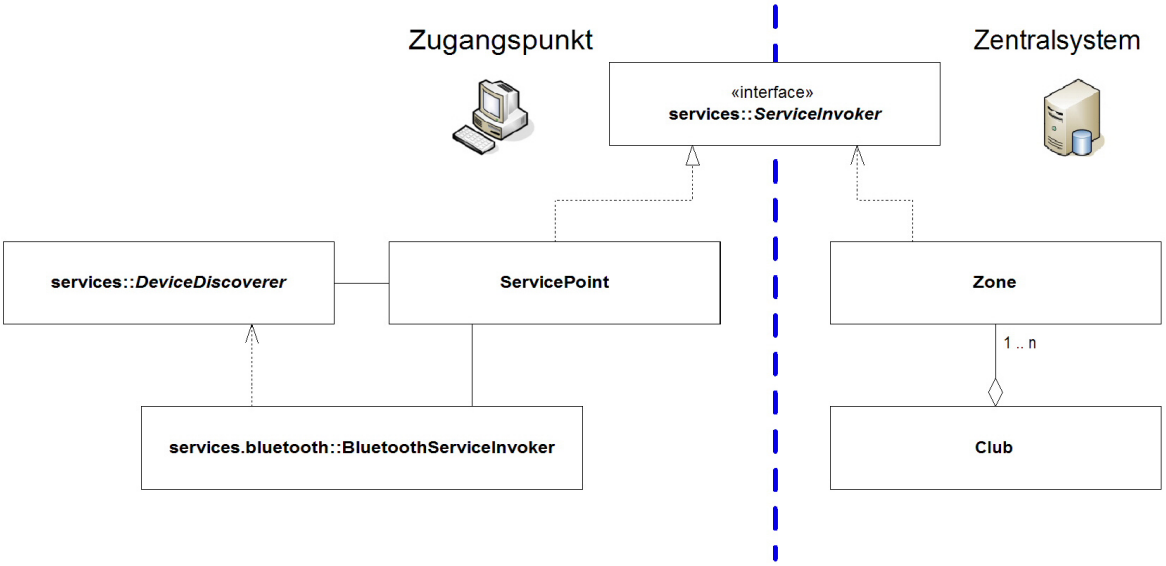

Abbildung 4.16: Klassendiagramm Nutzung mobiler Dienste

Abschnitt 4.4.2). Dadurch, dass die die Interfaces implementierenden Klassen in derselben virtuellen Maschine ausgeführt werden, können diese direkt referenziert werden. Für einen tatsächlichen Einsatz auf unterschiedlichen Punkten innerhalb eines Netzwerks, müssen die Klassen auch über dieses Netzwerk kommunizieren. Eine konkrete Realisierung dieser Kommunikation ist nicht Bestandteil dieser Arbeit, die Verwendung eines proprietären Protokolls über HTTP[8] oder der Einsatz von SOAP[9] oder RMI[10] sind nur einige von etlichen denkbaren Möglichkeiten.

Durch die Verwendung der Interfaces wird auf einfache Art und Weise ermöglicht, statt einer direkten Referenzierung beliebige Mechanismen zur Kommunikation zwischen die kooperierenden Klassen zu schieben, ohne dass weit reichende Änderungen innerhalb dieser durchgeführt werden müssen. Eine beispielhafte Darstellung des Prinzips ist aus Abbildung 4.17 ersichtlich. Die clientseitige Realisierung der Kommunikationskomponente implementiert dabei das Interface und gibt die Aufrufe an die serverseitige Realisierung mit Hilfe der verwendeten Technologie weiter. Dort wird über dasselbe Interface der konkrete Empfänger

[8] *Hypertext Transfer Protocol*, ein zustandsloses Protokoll zur Übertragung von Daten, primär wird es im Rahmen des World Wide Web zur Übertragung von Webseiten verwendet

[9] ein leichtgewichtiges Protokoll für den Austausch strukturierter Informationen in einer dezentralisierten, verteilten Umgebung (http://www.w3.org/TR/soap12-part1/)

[10] *Remote Method Invocation* ist der Aufruf einer Methode eines entfernten (auf einer anderen Virtuellen Maschine befindlichen) Java-Objekts und realisiert die Java-eigene Art eines sog. RPC (Remote Procedure Call)

des Aufrufs angesprochen und eventuelle Rückgabewerte an den Kommunikations-Client übermittelt, der diese nachfolgend an das eigentlich aufrufende Objekt zurückgibt.

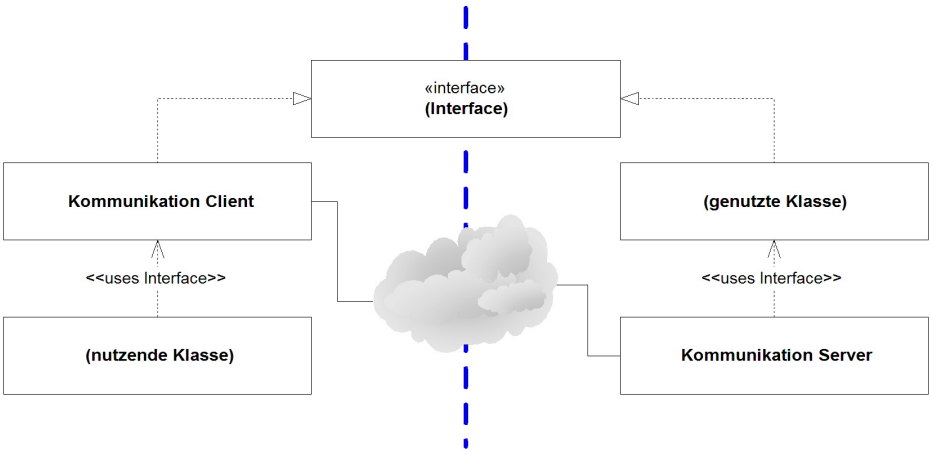

Abbildung 4.17: Indirekte Kommunikation zwischen verteilten Objekten

Grundsätzlich muss bei der Wahl der Kommunikationsmechanismen beachtet werden, dass diese konform zu den Spezifikationen der J2ME realisiert werden müssen, wenn die Hardware für die Zugangspunkte nicht kompatibel zur JAVA 2 SE ist. Dies wäre z.B. der Fall, wenn PDAs oder vielleicht sogar speziell für diesen Einsatzzweck entwickelte Set-Top-Boxen als Zugangspunkte an Stellen verwendet werden, an denen noch kein vollwertiger Personal Computer für die tägliche Arbeit positioniert ist, oder dies nicht sinnvoll oder sogar unmöglich ist.

Die Verteilung der Komponenten bezüglich des clubeigenen Netzwerkes und der mobilen Endgeräte wurde bereits am Anfang des Kapitels in Abbildung 4.1 dargestellt. Durch die Trennung von Zentralsystem und Zugangspunkten mit einer Komponente für die interne Kommunikation im Clubnetzwerk können die Komponenten dann auf drei verschiedene, räumlich und logisch voneinander getrennte Systembestandteile verteilt werden. Diese Verteilung und die Kommunikation zwischen den Systembestandteilen ist in Abbildung 4.18 ersichtlich.

4.2.4 Schließfächer

Für die Umsetzung der in dem Anwendungsfall in Abschnitt 3.3.2.6 beschriebenen Funktionalität zur Benutzung von Schließfächern wird eine Komponente entwickelt, die rudimentär einen Ort mit mehreren solchen Schließfächern auf virtueller Basis simuliert. Der Einsatz dieser Komponente geschieht stellvertretend für die Kategorie eines lokalen Dienstes (siehe Abschnitt 4.1.3), bei dem ein beliebiges System über entsprechende Schnittstellen von den

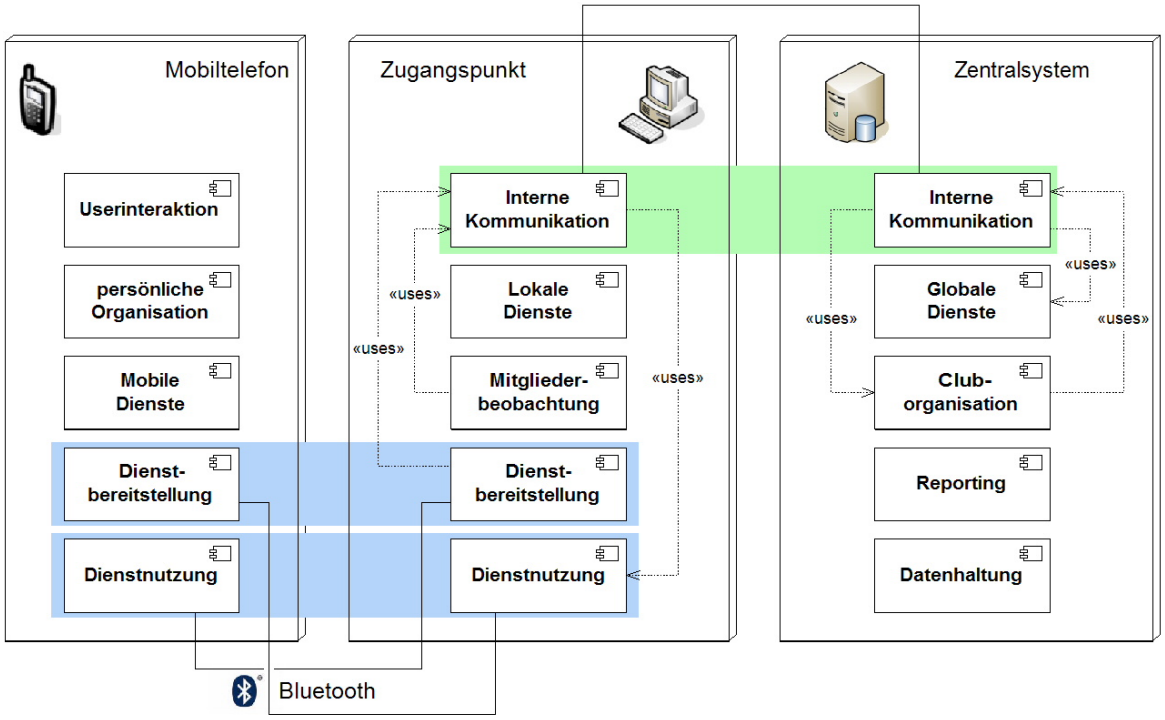

Abbildung 4.18: Verteilung der Komponenten auf Systembestandteile

zugehörigen Diensten gesteuert wird. Wie in Abbildung 4.18 dargestellt, sind im Unterschied zu den globalen Diensten hierbei die Komponenten der grundsätzlichen Dienstbereitstellung und der Dienstausführung nicht räumlich voneinander getrennt.*ServiceRequestHandler* und *ServiceServer* laufen somit innerhalb derselben virtuellen Maschine und werden auf einem Zugangspunkt durch dieselbe Instanz von *ServicePoint* zur Verfügung gestellt. Über die Klasse *Lockers* kann eine Liste aller vorhanden Schließfächer angefordert werden, um diese dann gezielt zu ver- oder entriegeln. Die Klasse spezialisiert die in Abschnitt 4.1.4.1 behandelte Klasse *ServiceServerImpl* und dient somit gleichzeitig als Server für die zugehörigen Dienste.

4.3 Mobiltelefon

In diesem Abschnitt werden die Komponenten behandelt, die ausschließlich auf den mobilen Endgeräten eingesetzt werden. Der Leistungsumfang der bei der Realisierung zu verwendenden JAVA-Bibliotheken ist dadurch strikt durch die Spezifikationen CLDC 1.1 und MIDP 2.0 eingegrenzt. Durch den Einsatz auf Geräten mit im Vergleich zu handelsüblichen Personal Computern stark eingeschränktem Volumen an Ausführungs- und Programmspeicher,

ist bei der Entwicklung noch mehr Wert darauf zu legen, den Umfang des Codes möglichst gering zu halten und innerhalb der Programmabläufe nicht mehr benötigte Ressourcen so frühzeitig wie möglich wieder freizugeben. Dabei muss abgewogen werden, inwieweit geschätzte Paradigmen der objektorientierten Programmierung verletzt und die Übersicht und Wartbarkeit des Quellcodes in Mitleidenschaft gezogen werden.

Das Klassendiagramm in Abbildung 4.19 skizziert die nur auf dem mobilen Endgerät einzusetzenden Klassen und deren Abhängigkeiten zu den Klassen aus der MIDP-Spezifikation. Eine genauere Beschreibung der Klassen und deren Interaktionen erfolgt in den weiteren Unterabschnitten.

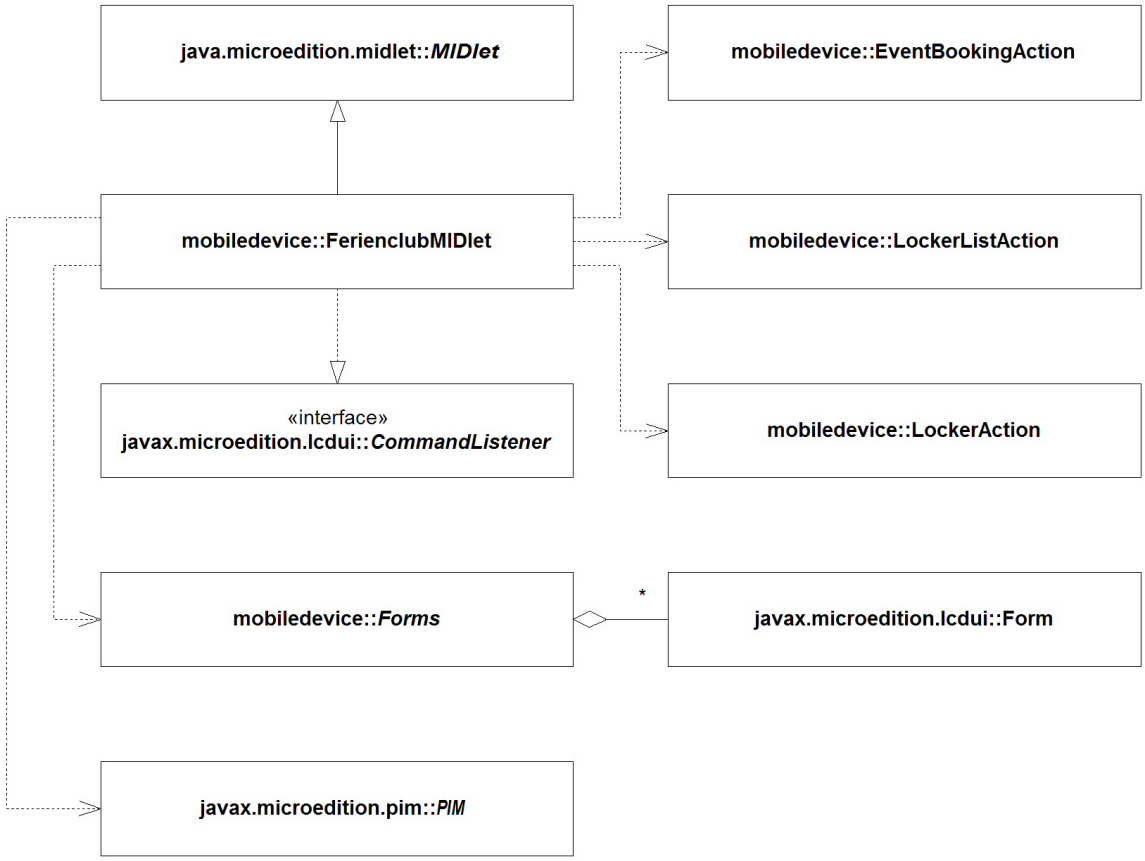

Abbildung 4.19: Klassendiagramm FerienclubMIDlet

4.3.1 Persönliche Organisation

In dieser Komponente werden die Funktionalitäten realisiert, die den Ablauf der Teilnahme des mobilen Nutzers am System auf dem mobilen Endgerät steuern. Hauptbestandteil und zentraler Punkt ist die Klasse *FerienclubMIDlet*, die von der in der J2ME enthaltenen Klasse *java.microedition.midlet.MIDlet*[11] abgeleitet ist. Eine solches MIDlet kann nach der Installation auf dem mobilen Endgerät gestartet und ausgeführt werden. Weiterhin implementiert das *FerienclubMIDlet* das Interface *javax.microedition.lcdui.CommandListener* und kann somit an Objekte, die der Benutzerinteraktion dienen, gekoppelt werden, um auf Aktionen und Eingaben am mobilen Endgerät zu reagieren (siehe hierzu auch Abschnitt 4.3.2).

Das Endgerät stellt mobile Dienste zur Verfügung und nutzt globale und lokale Dienste. Die grundsätzliche Funktionsweise und die Verwendung der entsprechenden Klassen geschieht dabei analog zu der innerhalb der Cluborganisation, welche in den Abschnitten 4.2.2.3 und 4.2.2.4 beschrieben wurde. Bei der Dienstbereitstellung erfolgt hier im Unterschied zu den globalen Diensten auf der Seite des Clubsystems keine räumliche Trennung von *ServiceRequestHandler* und der eigentlichen Implementation des *ServiceServer*. Die initiale Vorbereitung der Klassen für die Dienstbereitstellung erfolgt bei Beginn der Ausführung des *FerienclubMIDlet*. Werden mobile Dienste von den Zugangspunkten des Clubsystems angesprochen, stellt das *FerienclubMIDlet* den Dienstklassen folgende Funktionen zur Verfügung, um die Ausführung der Dienste zu vollziehen:

- *getMemberID()*
  Die Methode stellt für den *MemberIdentificationService* aus Abschnitt 4.1.1.1 die Mitgliedsnummer des Eigentümers des mobilen Endgeräts zur Verfügung. Im Rahmen der Umsetzung des Prototypen ist diese Information vorerst im Quellcode hart codiert, für jedes Mitglied müsste demnach die Klasse verändert und dann erneut kompiliert werden. In einer weiteren Ausbaustufe müsste die Möglichkeit gegeben sein, die Mitgliedsnummer individuell im Speicher des Endgerätes abzulegen, ohne Veränderungen an der Applikation vorzunehmen.

- *checkForDates(begin:long, end:long)*
  Mit dieser Methode kann der *DateCapacityService* aus Abschnitt 4.1.1.2 überprüfen, ob innerhalb einer Periode zwischen zwei durch einen long-Wert repräsentierten Zeitpunkten im Kalender des mobilen Endgerätes eingetragene Termine vorhanden sind. Hierfür werden die APIs des für die J2ME optionalen PIM[12]-Package nach der Spezifikation JSR-75 genutzt, die es ermöglichen, auf die Termin-, Kontakt und Aufgabenverwaltung eines mobilen Gerätes zuzugreifen.

[11] der Name *MIDlet* lehnt sich an den Begriff *Applet* an, mit dem Java-Programme bezeichnet werden, die als Bestandteil einer Web-Seite innerhalb eines Internet-Browsers ausgeführt werden

[12] Personal Information Management

- *proposeEvent(id:String, name:String, begin:long, end:long)*
  Um einen Gast mittels dem EventProposalService aus Abschnitt 4.1.1.3 auf einen Vorschlag für die Teilnahme an einer Veranstaltung hinzuweisen und die nötigen Informationen anzuzeigen, wird diese Methode zur Verfügung gestellt.

Für die Nutzung der über die Zugangspunkte angebotenen Dienste wird je Dienst eine Hilfsklasse implementiert, die unter Verwendung der Instanz des *BluetoothDeviceDiscoverer* nach einem sich in Reichweite befindlichen Dienstanbieter sucht und bei Erfolg anschließend über die Instanz des *BluetoothServiceInvoker* die Ausführung des entsprechenden Dienstes anstößt. Für die Ausführung wird dabei jeweils ein neuer paralleler Prozess gestartet, hierfür implementieren diese Klassen das Interface *java.lang.Runnable*. Eine Auslagerung dieser Funktionalitäten in einzelne Klassen und Prozesse schien notwendig, da das Ansprechen des Bluetooth DiscoveryAgents auf dem Smartphone direkt aus dem *FerienclubMIDlet*-Prozess heraus bei diesem einen Stillstand verursachte. Eine Aussage, ob es sich hierbei um eine generelle Problematik handelt, oder dies ein spezifisches Phänomen des verwendeten Smartphones[13] ist, ließ sich vorerst nicht treffen, da nur dieses eine Modell für die Tests zur Verfügung stand. Für den Prototypen sind dementsprechend folgende drei Hilfsklassen zu implementieren.

- *EventBookingAction*
  für das Ansprechen des globalen *EventBookingService* aus Abschnitt 4.1.2.1
- *LockerListAction*
  für das Ansprechen des lokalen Dienstes *LockerListService* aus Abschnitt 4.1.3.1
- *LockerAction*
  für das Ansprechen des lokalen Dienstes *LockerService* aus Abschnitt 4.1.3.2

Die Methode *createEventInCalendar(String name, long begin, long end)* ist eine weitere vom *FerienclubMIDlet* zur Verfügung gestellte Funktionalität und ermöglicht den Eintrag eines Termins in den internen Kalender des mobilen Endgerätes. Auch hier ist die Benutzung der APIs der Spezifikation JSR-75 erforderlich. Diese Funktionalität wird von der Klasse *EventBookingAction* genutzt, um nach erfolgreicher Dienstausführung die Veranstaltung in den Terminkalender einzutragen.

4.3.2 Informationsdarstellung und Userinteraktion

Die Beschränkungen bezüglich Bedienung und Anzeigemöglichkeiten eines handlichen mobilen Endgeräts bedingen nicht zu komplexe und einfach zu handhabende Anwendungen

[13] eine nähere Beschreibung des verwendeten Modells befindet sich in Abschnitt 4.4.2

(Friedburg 2005). Dies muss nicht unbedingt als Nachteil zu sehen sein. Betrachtet man Anwendungsfälle aus dem Kapitel 3, will ein Nutzer Dienste sicherlich schnell und unkompliziert in Anspruch nehmen.

Durch den Aufbau und die Funktionsweise der zu entwickelnden Plattform agiert das mobile Endgerät nicht ausschließlich als Thin-Client und es ist ein direktes Zusammenspiel der Benutzeroberfläche mit auf dem Gerät vorhandenen Komponenten erforderlich. Aus diesem Grund werden für die Realisierung der GUI die Möglichkeiten der MIDP-Spezifikation innerhalb der J2ME genutzt. Diese stellt vorgefertigte GUI-Elemente, wie z.B. Textfelder, Eingabefelder oder Auswahllisten zur Verfügung. Leider ist dabei der Einfluss auf die Darstellung der einzelnen Elemente sehr beschränkt. Lediglich die Reihenfolge der Elemente kann wirksam beeinflusst werden. Wie die Überschrift zu einem Textfeld angeordnet ist oder welcher Button des mobilen Endgerätes die verschiedenen Aktionen auslöst, kann der Entwickler nicht beeinflussen. Zudem existieren keine Möglichkeiten, Texte fett oder kursiv anzeigen zu lassen. Dies alles ist von der jeweiligen konkreten Implementierung von J2ME auf dem mobilen Endgerät abhängig. So kann die Anwendung von Endgerät zu Endgerät verschieden aussehen (Kruse 2005).

Für den Prototypen werden die drei folgenden Screens als *javax.microedition.lcdui.Form* rudimentär implementiert, die von der abstrakten Klasse *Forms* aggregiert werden und über diese zugreifbar sind. Einen Eindruck über die Ansicht der Screens vermittelt die Abbildung 4.20.

- Eine Hauptanzeige für Informationen über den grundsätzlichen Ablauf während der Ausführung der Applikation, wie z.B. Erfolgs- und Fehlermeldungen. Der Screen verfügt über ein Befehlsmenü, mit dem das Formular für die Bedienung der Schließfächer aufgerufen werden, sowie die Ausführung der Applikation beendet werden kann.
- Ein Screen, um eine Auswahlliste von Schließfächern und deren Zustand anzuzeigen. Dieser wird gleichzeitig dazu verwendet, eine Aktion auf ein ausgewähltes Schließfach zu initiieren. Dafür steht ein Passwort-Eingabefeld für den dazu erforderlichen Code zur Verfügung. Das Befehlsmenü beinhaltet Einträge für das Verriegeln und Entriegeln eines Schließfachs sowie einen Abbruch mit Rückkehr zur Hauptanzeige.
- Für die Unterbreitung eines Veranstaltungsvorschlags ein Screen, auf dem alle Informationen zu dieser Veranstaltung dargestellt sind. Bei der Anzeige des Screens wird, sofern das mobile Endgerät dies unterstützt, für kurze Zeit der Vibrationsalarm aktiviert. Im Befehlsmenü sind Einträge für eine Annahme oder Ablehnung des Vorschlags zu finden.

Um eine Trennung von View und Controller zu erreichen, wird die Instanz des *FerienclubMIDlet* über das Interface *javax.microedition.lcdui.CommandListener* an die Formulare gekoppelt und verarbeitet die durch den Benutzer ausgelösten Aktionen. Die

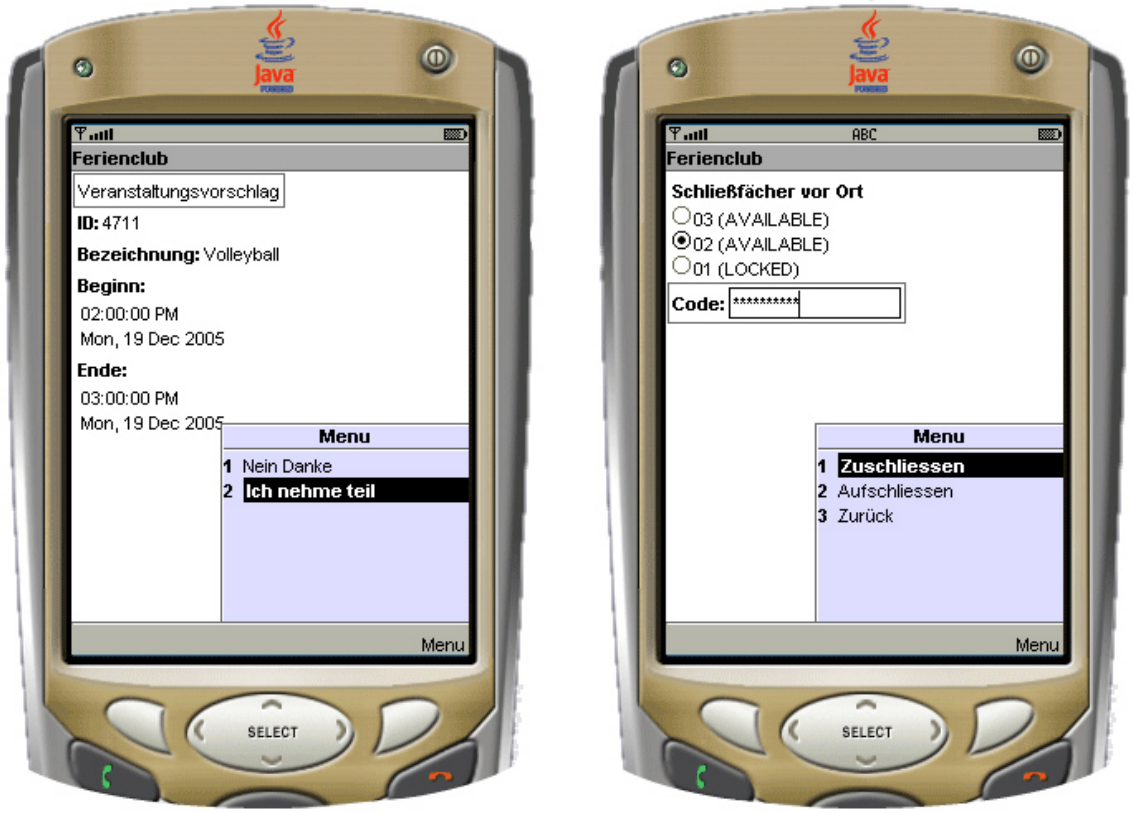

Abbildung 4.20: GUI für Veranstaltungsvorschlag (links) und Schließfachbenutzung (rechts), dargestellt mit einem Smartphone-Emulator des J2ME Wireless Toolkit 2.2

Klasse *Forms* bietet weiterhin Methoden an, um die entsprechenden Inhalte von Eingabefeldern und Auswahllisten der Formulare für eine weitere Verarbeitung auszulesen oder diese zu befüllen und auf dem Display des mobilen Endgerätes anzuzeigen.

4.4 Laborumgebung

4.4.1 Implementierungstools

Für die Implementierung der Java-Klassen wurde hauptsächlich die IDE[14] *IntelliJ IDEA* von JetBRAINS[15] in der Version 4.5.1 verwendet, da diese dem Autor zur Verfügung stand und er über weit reichende Erfahrung in der Entwicklung mit dieser IDE verfügt. Diese „intelligente“

[14] *Integrated Development Environment* bezeichnet ein Anwendungsprogramm zur Entwicklung von Software
[15] http://www.jetbrains.com/

Java IDE legt einen intensiven Fokus auf die Produktivität und stellt eine robuste Kombination weit reichender Entwicklungstools zur Verfügung.

Da IntelliJ IDEA erst ab der Version 5.0 eine integrierte Unterstützung für die Entwicklung von J2ME-Applikationen anbietet, wurde für die Implementierung der Klassen der für das mobile Endgerät vorgesehenen Komponenten zusätzlich die IDE *Sun Java Studio Mobility 6*[16], Version 2004Q3 in Verbindung mit dem *J2ME Wireless Toolkit 2.2*[17] herangezogen, welche kostenlos von der Firma Sun Microsystems bezogen werden können. Mit diesen Tools kann eine MIDlet-Suite für die Installation auf einem J2ME-fähigen Gerät zusammengestellt werden, die alle für die Applikation benötigten Klassen und weitere Informationen enthält. Durch den Einsatz eines Obfuscators bei der Kompilierung der Klassen wird nicht nur ein Reverse Engineering erschwert, sondern zusätzlich der Umfang der Klassen nochmals reduziert.

Ein weiteres wichtiges Merkmal ist die Bereitstellung von universellen Emulatoren für mobile Endgeräte und deren Integration in die Entwicklungsumgebung, so dass implementierter Code schnell und unkompliziert unter Verwendung des Debuggers getestet werden kann. Das J2ME Wireless Toolkit 2.2 unterstützt die Java APIs für Bluetooth nach der JSR-82-Spezifikation, so dass mit zwei zur selben Zeit laufenden Emulatoren eine Bluetooth-Verbindung simuliert werden kann. Hierbei kann dann allerdings nur J2ME-konformer Code verwendet werden, wodurch ein vollständiges Testen der mit den Möglichkeiten der J2SE entwickelten Klassen mit diesen Emulatoren unmöglich wird. Der *Impronto Simulator* von Rococo[18] ermöglicht es, sowohl J2ME- als auch J2SE-Applikationen in einer simulierten Bluetooth-Umgebung ausführen zu können, so dass bei der Entwicklung Tests und Konfigurationen ohne reale Bluetooth-Hardware stattfinden können. Dieses und vergleichbare, zum Zeitpunkt der Erstellung dieser Arbeit verfügbare Tools bedingen einen meist nicht unerheblichen Betrag für die Anschaffung, was einen Einsatz innerhalb der Entwicklung des Prototypen ausschloss. Trotz einer Möglichkeit der Verwendung von Emulatoren sind selbstverständlich Tests unter realen Bedingungen obligatorisch, um das tatsächliche Zusammenspiel zwischen Applikation und Hardware zu verifizieren und Aussagen über die Performance einer Applikation treffen zu können (Hopkins und Antony 2003).

4.4.2 Verwendete Hardware

Für die Entwicklung und die Durchführung der Tests standen für das Clubsystem und die Zugangspunkte drei Hardwarekonfigurationen zur Verfügung. Genauere technische Daten sind der Tabelle 4.1 zu entnehmen. Im Rahmen des Prototyps wurde zwar die Integration von Mechanismen zur Kommunikation zwischen den Zugangspunkten und der zentralen

[16] http://www.sun.com/software/products/jsmobility/
[17] http://java.sun.com/products/sjwtoolkit/index.html
[18] http://www.rococosoft.com

Clubverwaltung angedacht, jedoch nicht konkret implementiert (siehe Abschnitt 4.2.3). Für die Sinnhaftigkeit und einen realen Einsatz der Plattform ist eine für den Einsatzzweck ausreichende Abdeckung mit Zugangspunkten erforderlich und auch vorgesehen, aber für ein „Proof of Concept“ ist die Existenz eines einzelnen Zugangspunktes jedoch ausreichend. Ein Testszenario besteht somit immer aus einem einzigen Zugangspunkt, dessen Komponenten in derselben virtuellen Maschine ausgeführt werden wie die der clubinternen Verwaltung.

Typ	Betriebssystem	Prozessor	RAM
PC	Windows XP Professional SP 2	Intel Pentium 4, 3 GHz	1 GB
PC	Linux Fedora Core 3	Intel Pentium 3, 866 MHz	512 MB
Laptop	Windows XP Home Edition SP 2	Intel Pentium M, 1.7 GHz	512 MB

Tabelle 4.1: Hardwarekonfigurationen der Laborumgebung

Als Bluetooth-Hardware für die Personal Computer fanden Bluetooth USB-Adapter der Firma Belkin[19] vom Typ F8T008 und der Firma SITECOM[20] vom Typ CN-502 Verwendung, die durch einfaches Einstecken in einen USB-Port auf beliebige Hardware installiert werden können. Beide sind Adapter der Klasse I mit einer theoretischen Reichweite von bis zu 100 m und entsprechen dem Bluetooth-Standard der Version 1.2. Der für die Laborumgebung verwendete Laptop verfügt über einen in das Gerät integrierten monolithischen Bluetooth-Chip vom Typ BCM2035 des Herstellers Broadcom[21]. Auch hier wird der Bluetooth-Standard 1.2 unterstützt, konkrete Angaben über die maximal zu erzielende Reichweite konnten jedoch nicht ermittelt werden.

Eine Referenzhardware seitens der mobilen Endgeräte muss für die Lauffähigkeit der entwickelten Komponenten zum einen im Allgemeinen Bluetooth- und Java-fähig sein, zum anderen aber auch explizit den Standard JSR-82 für die Verwendung der JAWBT sowie den Standard JSR-75 für die Zugriffe auf das PIM des Gerätes unterstützen. Insbesondere der erstgenannte Standard wird bisher erst von einer sehr geringen Anzahl der erhältlichen Smartphones zur Verfügung gestellt. Dem Autor stand während der Verfassung dieser Arbeit leider nur ein Gerät zur Verfügung, welches die nötigen Qualifikationen erfüllt. Dabei handelt es sich um das Modell 6630 des Herstellers Nokia[22] (Abbildung 4.21), das mit dem Betriebssystem Symbian OS und 10 MB verfügbarem Speicher ausgeliefert wird.

[19] http://www.belkin.com
[20] http://www.sitecom.com
[21] http://www.broadcom.com
[22] http://www.nokia.de

Abbildung 4.21: Smartphone Nokia 6630

4.4.3 Bluetooth-Stack und JABWT Implementation

Der Betrieb einer Bluetooth-Hardware erfolgt über einen Protokoll-Stack, der grundlegend in Bluetooth Host und Bluetooth Controller aufgeteilt werden kann. Die Teile des Controllers befinden sich meist direkt auf der Bluetooth-Hardware und kommunizieren mittels eines standardisierten Interfaces mit den Schichten des Hosts, die normalerweise als Software implementiert sind (siehe Abschnitt 2.1.1). Einige dieser Schichten werden von den JABWT adressiert und ermöglichen so Zugriff und Kontrolle bezüglich der Bluetooth-Hardware mittels der Programmiersprache Java.

Für den Einsatz der Plattform wird ein solcher Bluetooth-Stack mit JABWT Implementation benötigt, damit dieser von den entwickelten Komponenten für die Ausführung der drahtlosen Kommunikation verwendet werden kann. Smartphones, die den Standard JSR-82 unterstützen, verfügen bereits über einen Bluetooh-Stack dieser Art. Die im vorherigen Abschnitt erwähnte Bluetooth-Hardware für die Realisierung der Zugangspunkte wird zwar auch mit einem vollständigen Bluetooth-Stack geliefert, diese Treiber bieten jedoch nur eine Integration in das entsprechende Betriebssystem und eine Ansteuerung über eine meist mitgelieferte Software. Hier besteht Bedarf für den Einsatz eines speziellen JABWT-fähigen Stacks, mit

dem die Bluetooth-Hardware bedient werden kann. Es sind diverse solche Implementationen erhältlich, von denen die bei der Recherche als potentiell einsetzbar erschienenen im Folgenden aufgeführt sind.

- *The JavaBluetooth Stack*[23], eine reine, nicht-native Java-Implementation der Bluetooth-Spezifikation 1.1 mit der Möglichkeit einer Nutzung der Protokolle L2CAP und SDP. Eine Unterstützung der Protokolle RFCOMM und TCS ist in Planung. Der Stack implementiert die APIs nach JSR-82 und ist als Open Source[24] bei SourceForge.net[25] als Download beziehbar.
- *Blue Cove*[26], eine Open Source Implementation der JSR-82 Bluetooth API für Java, die auf den Windows XP SP2 Bluetooth-Stack aufsetzt. Eine Unterstützung weiterer Betriebssysteme und anderer Stacks unter Windows ist zu gegebener Zeit geplant. Auch diese Implementation ist über SourceForge.net erhältlich.
- *aveLink*[27], von der Bluetooth Special Interest Group zertifizierter, den Standard JSR-82 erfüllender Stack von der Firma Atinav. Dieser ist im Rahmen eines Software Development Kit für J2ME oder J2SE erhältlich.
- *BlueJ*[28], ein Java Protokoll Stack, der auf BlueZ[29], dem offiziellen Bluetooth-Stack für Linux aufsetzt. Die Implementation ist als Open Source bei SourceForge.net verfügbar, entspricht aber nicht dem Standard nach JSR-82.
- *avetanaBluetooth*[30], eine JSR-82 Implementation der Firma avetana für Desktop PCs, die mit einem Windows-, MacOS- oder Linux-Betriebssystem laufen. Es gehört kein eigener Bluetooth-Stack zum Lieferumfang, die Implementation setzt für Windows auf dem Widcomm-Stack[31] und für Linux auf dem BlueZ-Stack auf. Nur die Linux-Version ist nicht kostenpflichtig und kann über SourceForge.net bezogen werden.

Die Verwendung einer Implementation, die nicht dem Standard JSR-82 entspricht, konnte schnell ausgeschlossen werden. Dies hätte bedeutet, dass Teile der Komponenten für die drahtlose Kommunikation aus den Abschnitten 4.1.4 und 4.1.5 nicht ohne weit reichende Anpassungen sowohl auf den mobilen Endgeräten, als auch auf den Zugangspunkten zum

[23] http://www.javabluetooth.org
[24] Software, deren Lizenzvertrag vorsieht, dass der Quellcode in lesbarer Form vorliegt und die Software in dieser und auch in einer veränderten Form beliebig kopiert, verbreitet und genutzt werden darf
[25] http://www.sourceforge.net, die weltgrößte Website für die Entwicklung von Open Source mit einer Bereitstellung von mehr als 100.000 Projekten und über 1.000.000 registrierten Benutzern
[26] http://sourceforge.net/projects/bluecove/
[27] http://www.avelink.com/Bluetooth/Products/JSR-82/index.htm
[28] http://sourceforge.net/projects/bluej
[29] http://www.bluez.org
[30] http://www.avetana-gmbh.de/avetana-gmbh/produkte/Readme.xml
[31] http://www.broadcom.com, Widcomm wurde 2004 von Broadcom übernommen

Clubnetzwerk einzusetzen gewesen wären. Da auch eine monetäre Investition außer Frage stand, kamen nur frei verfügbare Realisierungen in Betracht. Für eine Evaluation in der Laborumgebung, in der hauptsächlich Konfigurationen mit dem Betriebssystem Windows XP zur Verfügung standen, fiel letztlich die Wahl auf *Blue Cove*. Die Entwicklung dieser Implementation befand sich zum Zeitpunkt der Erstellung dieser Arbeit zwar noch im Beta-Stadium, aber die Installation und die ersten funktionalen Tests liefen problemfrei und zufrieden stellend. Des Weiteren wurden bereits alle für eine Umsetzung der Plattform ggf. benötigten Bluetooth-Protokolle unterstützt. Im weiteren Verlauf stellte sich allerdings heraus, dass die JSR-82-Spezifikation an wenigen Stellen noch nicht vollständig implementiert ist. Dieser Missstand konnte aber mit geringem Aufwand durch einen Workaround (siehe hierzu Abschnitt 4.1.5.6) entkräftet werden, so dass kein Austausch durch eine andere Implementation zwingend indiziert gewesen ist. Für zusätzliche Tests auf einem Linux-System kam die *avetanaBluetooth*-Implementation zum Einsatz.

4.4.4 Ergebnisse der Evaluierung

Wie bereits in Abschnitt 4.4.2 erwähnt, besteht ein Aufbau für den Test der grundlegenden Funktionalität der Plattform aus einem einzelnen Zugangspunkt, dessen Komponenten in derselben virtuellen Maschine ausgeführt werden, wie die des Zentralsystems zur clubinternen Verwaltung und dem einzigen zur Verfügung stehenden mobilen Endgerät. Alle Ergebnisse, die sich direkt auf das mobile Endgerät beziehen, müssen also nicht zwingend allgemeingültig sein, sondern können in gewissem Maße einen auf das Modell des Smartphones spezifischen Charakter aufweisen. Die Verfügbarkeit weiterer Testgeräte wäre in diesem Zusammenhang interessant und hilfreich gewesen.
Für die Ausführung sämtlicher Komponenten auf den Zugangspunkten wurde das SUN Java 2 SDK Standard Edition in der Version 1.4.2 verwendet. Die Verwendung des PCs mit Linux-Betriebssystem und der avetanaBluetooth-Implementation wurde nach einiger Zeit aufgrund massiver Probleme eingestellt. Während das reine Auffinden anderer Bluetooth-Geräte reibungslos funktionierte, verursachte sowohl das Anbieten von Diensten, als auch die Dienstnutzung eine kontinuierliche Fehlermeldung und einen Absturz des Systems. Eine erfolgreiche Ermittlung der genauen Ursache konnte nicht in angemessener Zeit erfolgen, deshalb wurde der Fokus für die Tests auf die Windows-Systeme mit dem Blue Cove Bluetooth-Stack gelegt, auf dem die selben Klassen problemlos ihren Dienst verrichteten.
Das Deployment der MIDlet-Suite mit den Klassen für das mobile Endgerät konnte problemlos vom Entwicklungsrechner durchgeführt werden, indem das entsprechende JAR[32]-Archiv

[32] Bei einer JAR-Datei (kurz für **J**ava **AR**chiv) handelt es sich um eine komprimierte oder auch auch unkomprimierte ZIP-Datei, die zusätzliche Metadaten in einer immer vorhandenen Datei META-INF/MANIFEST.MF enthält. JARs werden vor allem zur Verteilung von Java-Libraries und -Programmen eingesetzt.

mit Hilfe der Bluetooth-Technologie drahtlos auf das Endgerät übertragen wurde. Anschließend konnte die Applikation nach einer Einverständniserklärung durch den Benutzer endgültig auf dem Smartphone installiert werden. Das JAR-Archiv der endgültigen Fassung des Prototyps wies dabei einen Umfang von 26 Kilobyte auf. Eine Größe, die weder bezüglich der Übertragungszeit, noch dem auf dem mobilen Endgerät verfügbaren Speicher auch nur annähernd kritisch ist.

Bei den endgültigen Tests verlief sowohl das Auffinden des mobilen Endgerätes, die Nutzung der mobilen Dienste auf dem Smartphone, als auch die Nutzung der globalen und lokalen Dienste auf dem Zugangspunkt im Wesentlichen erfolgreich. Zu erwähnen ist, dass sich der Benutzer des mobilen Endgerätes dem Tracking durch das Clubsystem entziehen kann. Den Zugangspunkten ist es nicht möglich, einen Benutzer als Mitglied zu identifizieren und zu lokalisieren, wenn dessen Endgerät ganz ausgeschaltet ist, die Applikation auf dem Endgerät nicht ausgeführt wird oder die Bluetooth-Hardware durch den Benutzer so konfiguriert wurde, dass sie nicht von anderen Bluetooth-Geräten aufgefunden werden kann. Im letzten Fall wäre es dem Benutzer trotzdem möglich, aktiv globale und lokale durch den Club angebotene Dienste zu nutzen, sofern diese durch ihn initiierbar sind. Eine Beeinträchtigung anderer Funktionalitäten des Smartphones während der Ausführung der Applikation konnte nicht festgestellt werden. Es waren Zugriffe auf die internen Applikationen möglich, weitere Java-Applikationen konnten gestartet und Telefongespräche geführt werden, während das Smartphone weiterhin kontinuierlich durch die Zugangspunkte identifiziert wurde.

Die Vorgehensweise bei der Kommunikation via Bluetooth ist innerhalb der Plattform so realisiert, dass Verbindungen zwischen Zugangspunkten und mobilen Endgeräten nur kurzfristig für die Dauer einer Dienstausführung bestehen, da über die Bluetooth-Hardware nur max. sieben gleichzeitige drahtlose Verbindungen unterstützt werden, wobei sich die Geräte die Gesamtbandbreite teilen. Je nach Anzahl der in der Reichweite eines Zugangspunktes befindlichen mobilen Endgeräte und des Nutzungsverhaltens deren Besitzer bezüglich der globalen Dienste kann ein Zugangspunkt möglicherweise einer Überlastung ausgesetzt sein. Zeitmessungen während der Testläufe ergaben, dass die eigentliche Ausführung eines Dienstes auf einem spezifischen Gerät inklusive der drahtlosen Übertragung der Daten meistens unter einer Sekunde lag. Hierbei ist zu berücksichtigen, dass die Ausführung der globalen Dienste direkt auf den Zugangspunkten stattfand, bei einem Einsatz des Gesamtsystems mit mehreren Zugangspunkten sind noch Zeitaufwände für die Kommunikation zum Zentralsystem innerhalb des Clubnetzwerkes hinzuzurechnen. Weitaus kritischer ist die Zeitspanne von 10 bis 15 Sekunden zu beurteilen, die von der Bluetooth-Hardware für die Suche nach in Reichweite befindlichen Geräten benötigt wird (die Länge der Zeitspanne ist nach JSR-82 abhängig von der jeweiligen Implementation). Diese Funktionalität wird regelmäßig in Intervallen seitens der Zugangspunkte und der mobilen Endgeräten bei jeder Inanspruchnahme von globalen oder lokalen Diensten genutzt. Bei den Tests stellte sich heraus, dass sich Device-Discovery und die parallele Ansprache eines Bluetooth-Service auf demselben

Bluetooth-Gerät gegenseitig stören, was ein fehlerhaftes Verhalten verursacht. Hier besteht ein Verbesserungspotential für den entwickelten Prototypen: die Verwendung der Funktionalitäten für das Auffinden anderer Geräte sollte größtmöglich reduziert werden. Beim Tracking der Mitglieder über die Zugangspunkte ist die Verwendung zwingend notwendig, jedoch sollte die Zeitspanne des Intervalls so groß wie möglich gewählt werden. Diese Größe wird wiederum durch die Anforderung an eine ausreichende Aktualität der Positionen der Mitglieder nach oben begrenzt. Die Aktualität des Trackings könnte durch das Einfließen der Information verbessert werden, dass ein Benutzer zu einem bestimmten Zeitpunkt einen Dienst über einen bestimmten Zugangspunkt in Anspruch genommen hat. Für die Nutzung globaler und lokaler Dienste durch das mobile Endgerät sollte zuerst durch die Applikation der Versuch unternommen werden, den letzten Zugangspunkt, über den eine erfolgreiche Dienstnutzung stattfand, anzusprechen. Ein solcher Versuch nimmt bei einem Fehlschlag nicht mehr als eine Sekunde in Anspruch. Befindet sich der Gast aber noch in Reichweite dieses Zugangspunktes, kann auf den zeitraubenden Auffindungsprozess verzichtet werden.
Eine weitere Herangehensweise für eine Reduzierung des Problems wäre, im Zuge des Trackings die Zugangspunkte bei der Nutzung des Dienstes zur Identifikation deren Bluetooth-Adresse übermitteln zu lassen, so dass auf dem mobilen Endgerät eine Auswahl an Zugangspunkten vorgehalten werden kann, für die die Wahrscheinlichkeit hoch ist, dass sie sich in Reichweite befinden. Bevor eine globale Suche nach Zugangspunkten angestoßen wird, kann dann zuerst versucht werden, gezielt einen Zugangspunkt anzusprechen.

Die für die Bluetooth-Hardware angegebene maximale Reichweite für eine Funkübertragung konnte in der Laborumgebung bei weitem nicht erreicht werden. Eine Verwendung innerhalb von Gebäuden schränkte diese teilweise auf ein Zehntel des theoretisch möglichen Wertes ein, so dass nur noch ein Radius von ca. 10 Metern um einen Zugangspunkt bedient werden konnte. Für die Praxis ist dies ein relativ kritischer Wert, da im Gegensatz zum Laboraufbau mit nur einem Zugangspunkt in der Realität für viele Einsatzzwecke eine möglichst vollständige Flächenabdeckung angestrebt werden würde, für die dann ein Vielzahl von geschickt positionierten und miteinander vernetzten Zugangspunkten geschaffen werden müsste. Bei einer Positionierung der Zugangspunkte ist es dabei zusätzlich durchaus denkbar, dass bezüglich ihrer Radien der Reichweite Überschneidungen auftreten könnten. Dies würde bedeuten, dass bei der Suche nach mobilen Endgeräten im Zuge des Trackings nicht nur solche, sondern eventuell auch andere Zugangspunkte aufgefunden werden würden. Dies stellt für die Plattform kein primäres Problem dar, da die Zugangspunkte keine mobilen Dienste (insbesondere den für die Mitgliedsidentifikation) zur Verfügung stellen. Trotzdem wird natürlich ein lokalisierter Zugangspunkt durch die Anfrage, ob er einen solchen Dienst bereitstellt, belastet. Um dem entgegenzuwirken, sollte auf jedem Zugangspunkt eine Liste mit den Bluetooth-Adressen aller Zugangspunkte propagiert werden, um diese identifizieren zu können und somit eine konkrete Dienstanfrage von vornherein unterdrückt werden kann.

Eine den Einsatz des entwickelten Systems stark einschränkende Problematik wurde durch

das für die Tests verwendete Smartphone aufgezeigt. Zusätzlich installierten Applikationen ist es von vornherein nicht ohne weiteres möglich, Verbindungen jeglicher Art zu initiieren oder auf interne Funktionalitäten des Telefons zuzugreifen. Im Zusammenhang der Plattform betrifft dies in erster Linie das Etablieren von Bluetooth-Verbindungen und den Zugriff auf den internen Kalender des mobilen Endgerätes. Über das Betriebssystem hat der Benutzer die Möglichkeit, für die einzelnen Applikationen die Erlaubnis für eine solche Nutzung zu erteilen. Das Testgerät bietet dabei für diverse Kategorien unterschiedliche Möglichkeiten der Aufhebung solcher Einschränkungen, die zwischen einem grundsätzlichen Verbot und einer uneingeschränkten Verwendung rangieren. Während für die telefoninterne Bluetooth-Hardware eine jederzeitige Nutzung konfiguriert werden konnte, gab es für das Auslesen und Bearbeiten von Benutzerdaten nur die Wahl, dieses vollständig zu verhindern oder vor jeder Durchführung eines solchen Vorgangs die Zustimmung des Benutzers über eine Eingabeaufforderung einzuholen. Im Zuge von Anwendungsfällen, bei denen der Benutzer sowieso zu diesem Zeitpunkt eine Bedienung des Gerätes vornimmt (wie z.B. das Eintragen des Veranstaltungszeitraumes in den Kalender direkt nach einer Zusage zu einer Veranstaltung, siehe Abschnitt 3.3.2.5) gestaltet sich dies noch unproblematisch. Für die Ausführung von mobilen Diensten auf dem Gerät, die stillschweigend ohne eine Interaktion mit dem Benutzer ablaufen sollen, ist es hingegen inakzeptabel. Hier wird insbesondere der Dienst zur Überprüfung der Terminkapazität eines Benutzers untergraben (siehe auch Anwendungsfall in Abschnitt 3.3.2.3), der gerade dazu dienen soll herauszufinden, ob es lohnenswert erscheint, diesen mit einer Nachricht auf sein Gerät zu „belästigen“. Ein grundsätzlicher Schutz der Benutzerdaten ist sicherlich sinnvoll und erwünscht. Wie sich gezeigt hat, gibt es jedoch Anwendungsfälle, die eine fortwährend gültige Freigabe für den Zugriff erfordern.

Eine weitere Besonderheit bei den für die Laborumgebung verwendeten Konfigurationen war bei dem Bluetooth Geräte-Manager des Windows-Betriebssystems zu beachten. Ist dort ein Bluetooth-Gerät mit Hilfe des Assistenten einmal aufgefunden und hinzugefügt worden, wird dieses bei einem durch den Windows Bluetooth-Stack durchgeführten Device-Discovery gefunden, auch wenn es sich nicht in Reichweite befindet oder sogar ausgeschaltet ist. Für eine Verwendung als Zugangspunkt empfiehlt es sich deshalb, alle Kopplungen mit Bluetooth-Geräten aus dem Geräte-Manager zu entfernen, um ein erfolgloses Ansprechen dieser Geräte im Rahmen des Trackings der Mitglieder im Vorwege zu unterbinden.

–> Test bezüglich des Energieverbrauchs der mobilen Endgeräte ??!!

4.5 Zusammenfassung

Trotz der der erfolgreichen Realisierung des Prototyps ist eine Eignung für einen tatsächlichen Einsatz einer solchen Plattform in der Praxis kritisch zu bewerten. Damit beliebige Smartphones als mobile Teilnehmer des Systems verwendet werden können, müssen diese

über Bluetooth-Hardware verfügen und bestimmte Standards bezüglich ihrer Java-Fähigkeit, wie die Spezifikationen JSR-82 und JSR-75, unterstützen. Zum Zeitpunkt der Verfassung dieser Arbeit waren auf dem Markt nur wenige Geräte verfügbar, die diese Qualifikationen mit sich brachten. Die Verbreitung der Technologie und allgemeingültige Standards sind hier Eckpfeiler, die noch nicht ausreichend etabliert sind. Ein weiteres wichtiges Merkmal ist die Möglichkeit, zusätzlich auf dem mobilen Endgerät installierten Applikationen uneingeschränkten Zugriff auf die Benutzerdaten zu gewähren. In diesem Punkt scheinen die zurzeit erhältlichen Geräte Defizite aufzuweisen.

Es zeichnet sich jedoch ab, dass diese Problematik in naher Zukunft in großem Maße entschärft werden wird. Nach Aussagen der Firmen Sun und Nokia ist der Siegeszug von Java auf mobilen Geräten unaufhaltbar, dieses Jahr sind dort zum ersten Mal mehr JVMs[33] installiert als auf PCs und Servern. Dem herrschenden API[34]-Chaos und den daraus resultierenden Schwierigkeiten bei der Entwicklung portabler mobiler Anwendungen scheint mit der *Mobile Service Architektur*[35]-Spezifikation eine Ende gesetzt zu werden. Diese definiert Pakete mit standardisierten APIs und soll bereits Anfang 2006 von einem Großteil der Hersteller und Provider eingesetzt werden (Puder und Röwekamp 2005). Geht man zusätzlich von einer weiterhin rasanten technischen Weiterentwicklung der Hardware aus, ist zu vermuten, dass viele, während der Erstellung dieser Arbeit aufgetretene und einen Einsatz in der Praxis behindernde Kritikpunkte zukünftig entkräftet sein werden.

Der Prototyp der Plattform wurde an vielen Stellen nur rudimentär entwickelt und für einige Bereiche besteht Potential für eine vertiefende und weiterführende Betrachtung. Die für die drahtlose Datenübertragung verwendete Bluetooth-Technologie verfügt zwar über Verschlüsselungsmechanismen, welche Anforderungen jedoch an die Sicherheit gestellt werden sollten und ob diese erfüllt werden könnten, ist sicherlich ein Punkt, der spätestens im Zuge der Ermöglichung von Zahlungsvorgängen mittels der mobilen Endgeräte noch genauer betrachtet werden müsste. Auch eine Verbesserung oder Verfeinerung der Positionsbestimmung der Plattformteilnehmer könnte angestrebt werden. Je nach Fortschritt der technologischen Entwicklung und des Einsatzortes der Plattform könnte hier alternativ oder additiv die Verwendung von GPS[36] zum Einsatz kommen, da es nicht unwahrscheinlich ist, dass in endlicher Zeit alle modernen Mobiltelefone mit dieser Technologie ausgerüstet sein werden (Glahn 2005b). Mechanismen für die interne geografische Darstellung der durch die Plattform verwalteten Zonen sowie Algorithmen für Entfernungsbestimmung zwischen diesen mit Hilfe der Graphentheorie stellen eine weitere interessante Herausforderung dar.

[33] **J**ava **V**irtual **M**achine, Laufzeitumgebung für Java-Applikationen

[34] **A**pplication **P**rogramming **I**nterface

[35] JSR-248 und JSR-249

[36] **G**lobal **P**ositioning **S**ystem. Ein Verfahren zur Positionsbestimmung mit Hilfe von Satelliten, die Radiosignale ausstrahlen, aus denen ein Empfänger seine Position errechnen kann.

5 Fazit und Ausblick

Im Rahmen dieser Arbeit wurde eine Plattform entworfen, auf deren Basis mobile Endgeräte drahtlos mit Hilfe der Bluetooth-Technologie über Zugangspunkte in ein Gesamtsystem integriert werden und mit diesem interagieren, wobei das System die jeweilige geografische Position eines mobilen Teilnehmers beobachtet. Eine Bereitstellung von nutzbaren Diensten sollte dabei sowohl seitens der Zugangspunkte, als auch seitens der mobilen Endgeräte möglich sein. Anhand des konstruierten Beispiels der Betriebs eines Ferienclubs wurden Anforderungen an Funktionalität und Aufbau einer solchen Plattform definiert und ein Prototyp exemplarisch implementiert.

Der „Proof of Concept“ kann insgesamt als gelungen bezeichnet werden, alle gestellten Anforderungen konnten im Wesentlichen unter Verwendung der verfügbaren Technologien erfüllt werden. Die Kombination von Bluetooth als drahtloses Kommunikationsmedium und Java als Implementierungssprache erwies sich als durchaus realisierbar. Die Ergebnisse der Evaluierung und dort aufgetretene Problematiken wurden in den Abschnitten 4.4.4 und 4.5 als Abschluss des Kapitels 4 dargelegt und diskutiert.

In der Vergangenheit hat sich gezeigt, dass sich die für die Interaktion mit der Plattform geeigneten Geräte zusehens zu einem ständigen Begleiter entwickeln. Dadurch sind auf dem Gerät gespeicherte Informationen immer dort verfügbar, wo sich der Besitzer des Gerätes aktuell aufhält und können ad-hoc unterschiedlichsten Systemen zur Verfügung gestellt werden. Von besonderer Bedeutung im Hinblick auf mögliche zukünftige Einsatzgebiete der Plattform sind speziell Benutzerprofile, also Daten, die in einer individuellen Ausprägung bezüglich jedes Anwenders vorliegen. Das lokale Speichern der Daten auf dem Gerät eines Benutzers hat neben der allgegenwärtigen Verfügbarkeit zudem den Vorteil in Hinblick auf die Akzeptanz des Anwenders, dass dieser eine bessere Kontrolle über seine personenbezogenen Daten hat und nach Bedarf auch einen Zugriff verwehren kann.
Die Kommunikation zwischen dem mobilen Gerät und dem entwickelten System ist für kurze Distanzen konzipiert. Der Anwender befindet sich also immer vor Ort, wenn er mit einem Zugangspunkt zum System interagiert. In Kombination mit der Möglichkeit, die Position des Anwenders zu bestimmen, können dadurch Dienste ortsabhängig zur Verfügung gestellt werden. Die Positionsbestimmung bietet zudem die Möglichkeit, den Weg des mobilen Gerätes zu verfolgen und auf diesem Weg Informationen für statistische Erhebungen zu sammeln.

Seit geraumer Zeit ist ein Trend zur Personalisierung von Produkten, Dienstleistungen und Angeboten zu beobachten. Der Kunde profitiert hierbei von einer zielgerechten Ansprache, Unternehmen nutzen diesen Trend um wertvolle Daten über ihre Kunden zu gewinnen und so sehr viel spezifischer Marktforschung betreiben zu können, als dies noch vor einigen Jahren der Fall war. Basis für jede Form von Personalisierung und Marktforschung sind Daten über Kunden und deren Einkaufsverhalten, die zurzeit hauptsächlich über sog. Rabattkarten ermittelt werden, die vom Kunden im Zuge des Bezahlvorgangs vorgelegt werden.
Durch den Einsatz von an die in dieser Arbeit entwickelten Plattform angelehnten Systemen könnte die Qualität und Quantität der zur Verfügung stehenden Daten erheblich erhöht werden, wobei gleichzeitig eine deutliche Senkung des Aufwands zur Erhebung der Daten erzielbar wäre. Solche Rabattkarten könnten durch eine Applikation auf den mobilen Endgeräten realisiert werden, wodurch eine sehr viel detailliertere Ermittlung der Daten möglich wäre. Die Erhebung der Daten erfolgt nicht an einem zentralen Punkt (der Kasse), sondern kann an jedem Ort, an dem ein Zugangspunkt zur Verfügung steht, auch ohne einen konkreten Kauf durchgeführt werden. Dadurch können zusätzliche ortsabhängige Daten gesammelt werden - z.B. die genaue Wegstrecke eines Kunden durch einen Supermarkt oder seine Aufenthaltsdauer vor einem Regal oder Schaufenster.
Das Erheben von personenbezogenen Daten und das Anbieten von personalisierten Informationen ist im Internet inzwischen weit verbreitet und akzeptiert, man denke hier an Partnerbörsen oder den mittlerweile in fast jedem Online-Shop zum Standard gehörenden Hinweis „Kunden die Produkt A kauften, kauften auch Produkt B.“. Unter Verwendung der diskutierten Technologien könnten diese Konzepte von der virtuellen Welt des Internets auf die reale Welt übertragen werden. Zusätzlich ergäbe sich dann die Möglichkeit der Erweiterung der Konzepte, Dienste und Informationen nicht nur zu personalisieren sondern auch automatisch ortsabhängig anzubieten.

Diese technischen Möglichkeiten bergen natürlich auch die Gefahr einer völligen Datentransparenz, ständiger Eingriffe in die persönliche Privatsphäre und auch des Missbrauchs der Daten. Hier müssen geeignete gesetzliche Regelungen und entsprechende technische Schutzmechanismen entwickelt werden, um der Problematik des „Gläsernen Menschen“ entgegenzuwirken.

Es gibt eine Vielzahl an Szenarien, für die die Verwendung einer solchen Plattform Vorteile bietet oder deren Realisierung erst durch diese ermöglicht wird. Der Einsatz ist immer dann besonders sinnvoll, wenn Informationen sowohl personen- als auch ortsabhängig präsentiert oder abgerufen werden sollen. Eine rapide Verbreitung und Verbesserung der in dieser Arbeit verwendeten Technologien und die Weiterentwicklung von relevanten Standards kündigen sich bereits an, so dass die hier skizzierten Konzepte und Szenarien schon bald alltäglich sein werden.

A Klassendiagramme

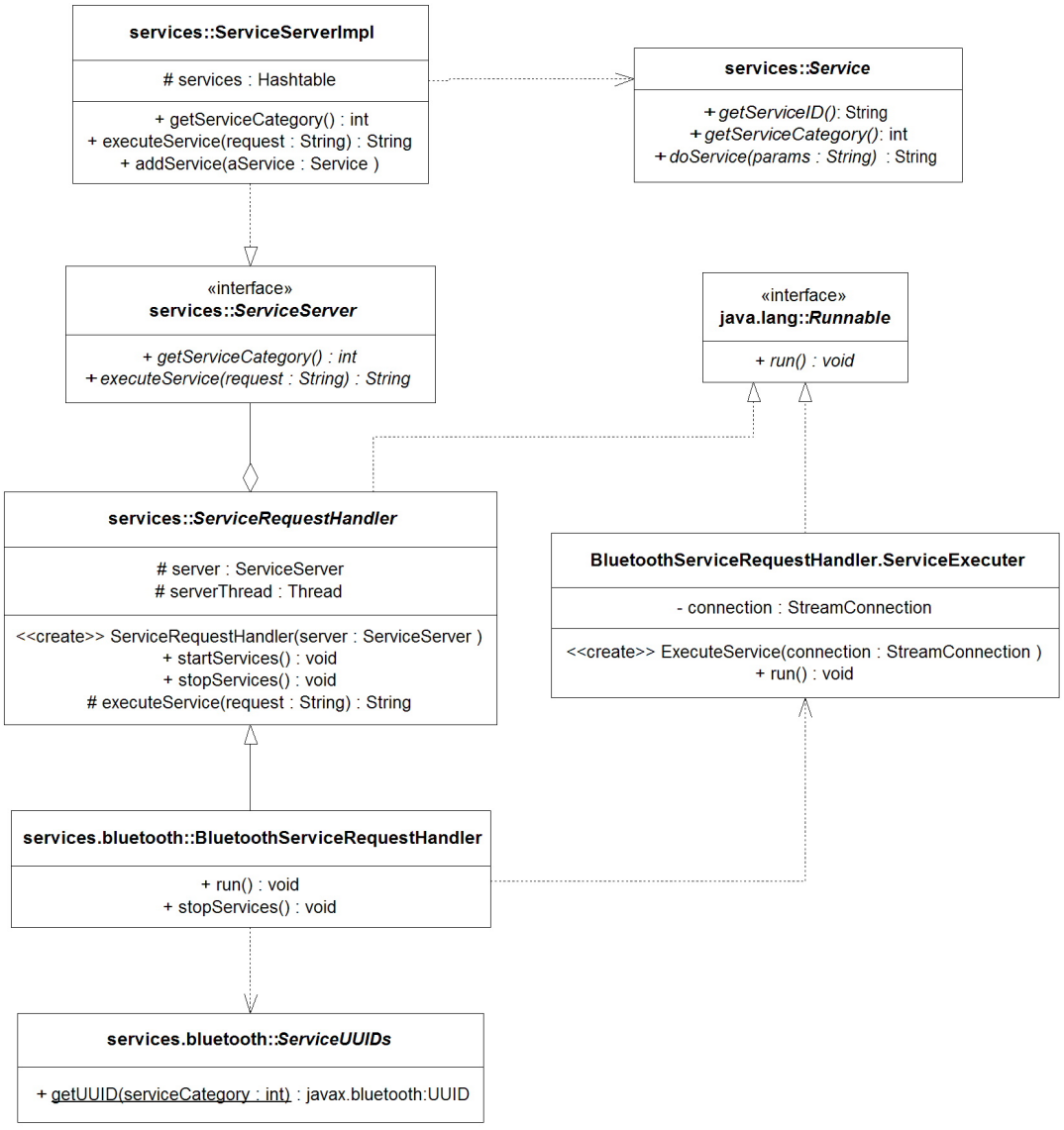

Abbildung A.1: Klassendiagramm Dienstbereitstellung

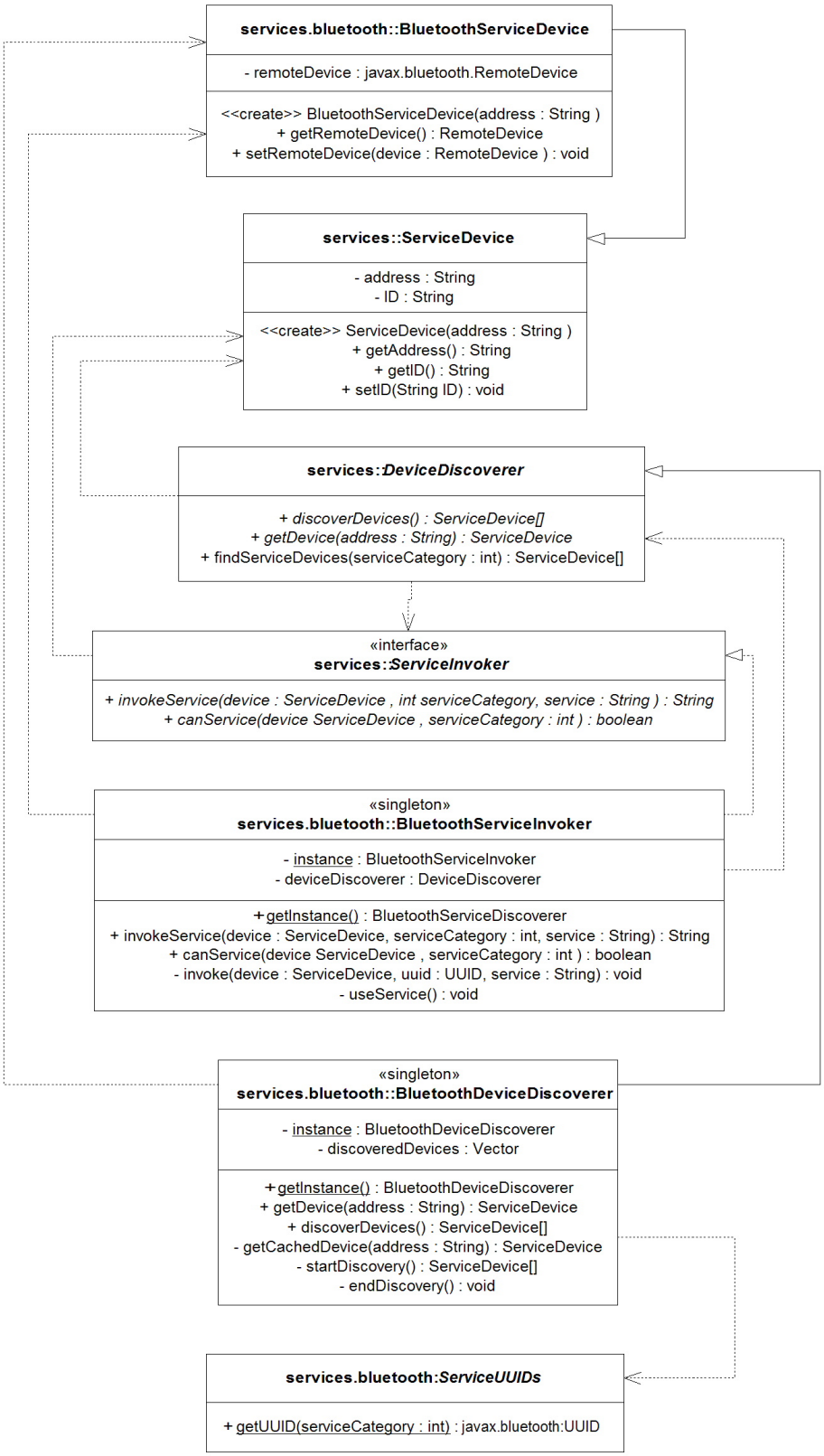

Abbildung A.2: Klassendiagramm Dienstnutzung

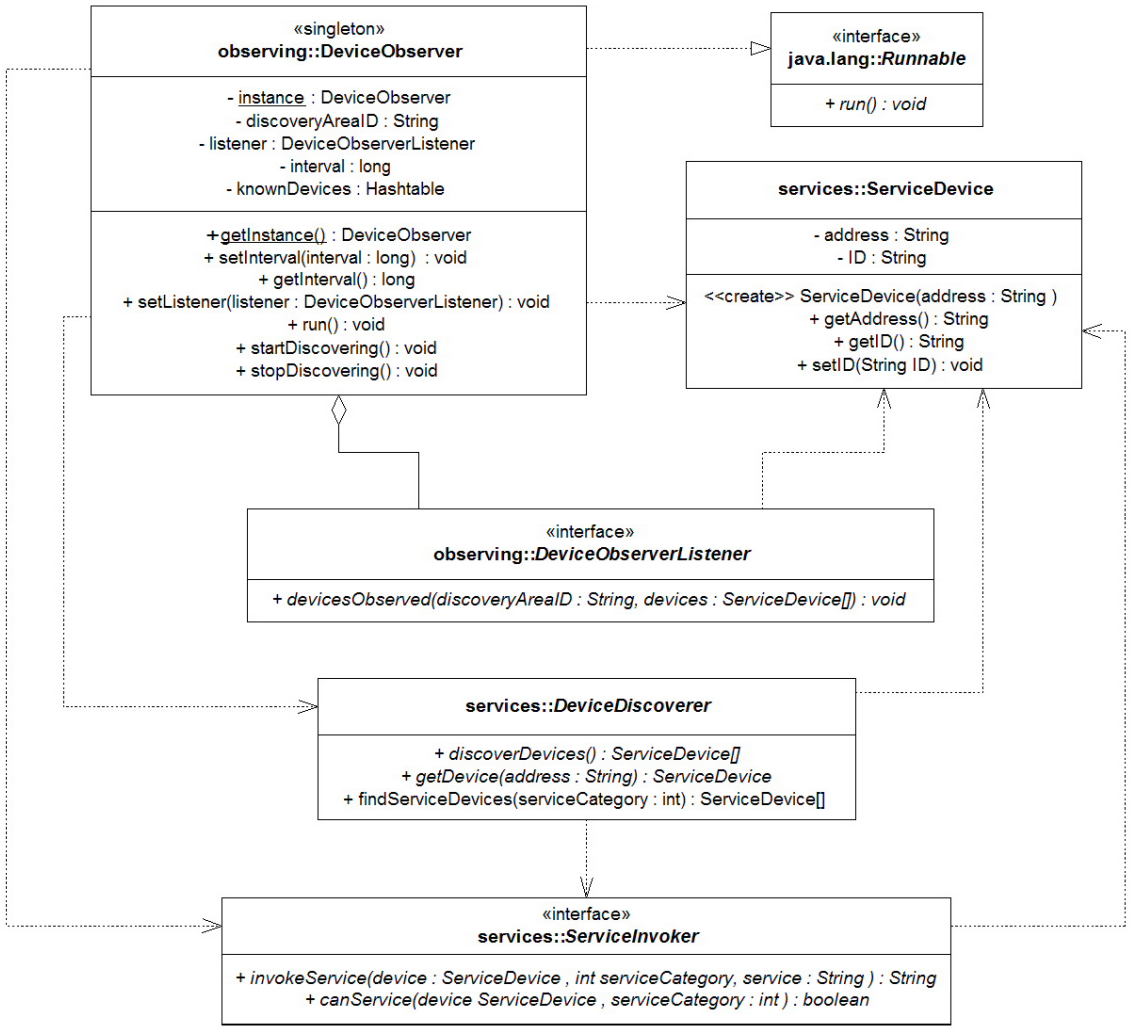

Abbildung A.3: Klassendiagramm Mitgliederbeobachtung

Tabellenverzeichnis

2.1 Theoretische Übertragungsreichweiten von Bluetooth 9

4.1 Hardwarekonfigurationen der Laborumgebung 73

Abbildungsverzeichnis

2.1 Beispiele für Piconets und Scatternets 10
2.2 Bluetooth Host und Controller Klassifikation 11
2.3 Bluetooth-Protokoll-Stack 12
2.4 Java 2 Plattformen 14
2.5 Komponenten der J2ME Architektur 15

3.1 Grobszenario der Kommunikation im Ferienclubsystem 26
3.2 Übersicht möglicher Anwendungsfälle 27
3.3 Interaktion zwischen Gast und Ferienclub 28
3.4 Abhängigkeiten zu den Systemanwendungsfällen 29
3.5 ausgewählte Anwendungsfälle für die Realisierung 30

4.1 Komponenten der Plattform 37
4.2 Abstrakte Klasse Service 38
4.3 Klassendiagramm Dienstbereitstellung 43
4.4 Klasse BluetoothServiceRequestHandler und JSR-82 und CLDC 1.1 46
4.5 Sequenzdiagramm BluetoothServiceRequestHandler 47
4.6 Klassendiagramm Dienstnutzung 48
4.7 Klasse BluetoothServiceInvoker und JSR-82 und CLDC 1.1 53
4.8 Sequenzdiagramm BluetoothServiceInvoker 53
4.9 Klasse BluetoothDeviceDiscoverer und JSR-82 55
4.10 Sequenzdiagramm BluetoothDeviceDiscoverer 55
4.11 Klassendiagramm Mitgliederbeobachtung 57
4.12 Verbindung zwischen Zentralsystem und Zugangspunkten 59
4.13 Klassendiagramm Clubverwaltung 60
4.14 Klassendiagramm Tracking 61
4.15 Klassendiagramm Bereitstellung globaler Dienste 62
4.16 Klassendiagramm Nutzung mobiler Dienste 64
4.17 Indirekte Kommunikation zwischen verteilten Objekten 65
4.18 Verteilung der Komponenten auf Systembestandteile 66
4.19 Klassendiagramm FerienclubMIDlet 67

4.20 Bespielhafte Darstellung der GUI 71
4.21 Smartphone Nokia 6630 74

A.1 Klassendiagramm Dienstbereitstellung 83
A.2 Klassendiagramm Dienstnutzung 84
A.3 Klassendiagramm Mitgliederbeobachtung 85

Literaturverzeichnis

Babic 2003 BABIC, Alexander: *Entwicklung einer profilverarbeitenden ubiquitären Anwendung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, Februar 2003

Borriello u. a. 2005 BORRIELLO, Gaetano ; CHALMERS, Matthew ; LAMARCA, Anthony ; NIXON, Paddy: Delivering Real-World Ubiquitous Location Systems. In: *Communications Of The ACM* 40, No.3 (2005), März, S. 37–41

Bresch 2004 BRESCH, Marco: *Erweiterung des JavaServer Faces Framework für den Einsatz in mobilen Anwendungen*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, Oktober 2004

Eschweiler 2003 ESCHWEILER, Sebastian: *Java 2 Micro Edition*. 1. Auflage. Bonn : verlag moderne industrie Buch AG & Co. KG, 2003

Friedburg 2005 FRIEDBURG, Sven O.: *Möglichkeiten ortsabhängiger Interaktionen auf Basis mobiler Endgeräte*, Hochschule für Angewandte Wissenschaften Hamburg, Studienarbeit, Mai 2005

Gamma u. a. 2004 GAMMA, Erich ; HELM, Richard ; JOHNSON, Ralph ; VLISSIDES, John: *Entwurfsmuster - Elemente wiederverwendbarer objektorientierter Software*. Addison Wesley, 2004

Glahn 2005a GLAHN, Kay: MIDlets ganz easy. In: *JavaMagazin* 4 (2005), April, S. 67–77

Glahn 2005b GLAHN, Kay: Mobility Special - Starter. In: *Javamagazin* 7 (2005), Juli, S. 59

Glatschke 2005 GLATSCHKE, Christian: Wireless Java und JCP 2.6. In: *JavaSPEKTRUM* 2 (2005), April/Mai, S. 51–52

Griffel 1998 GRIFFEL, Frank: *Componentware - Konzepte und Techniken eines Software-Paradigmas*. Heidelberg : dpunkt-Verlag, 1998

Hahn 1998 HAHN, Dr. J.: Wenn Bits und Bytes fliegen lernen / Mobile Datenübertragung - technisch und wirtschaftlich eine Herausforderung. In: *Elektronik* 20 (1998), S. 94–101

Hopkins und Antony 2003 HOPKINS, Bruce ; ANTONY, Ranjith: *Bluetooth for Java*. New York : Springer-Verlag, 2003

Kahlbrandt 1998 KAHLBRANDT, Bernd: *Software-Engineering*. Springer-Verlag, 1998

Klingsheim 2004 KLINGSHEIM, André N.: *J2ME Bluetooth Programming*, University of Bergen, Diplomarbeit, Juni 2004

Kroll und Steinhaus 2003 KROLL, Michael ; STEINHAUS, Stefan: *J2ME Developer's Guide*. München : Markt+Technik Verlag, 2003

Kruse 2005 KRUSE, Volker: *Konzeption und Realisierung der Anbindung mobiler Endgeräte an eine Vorgangsmanagementanwendung*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, Juli 2005

Kumar u. a. 2004 KUMAR, C B. ; KLINE, Paul J. ; THOMPSON, Timothy J.: *Bluetooth Applikation Programming with the JAVA APIs*. San Francisco : Morgan Kaufmann, 2004

LaMarca u. a. 2005 LAMARCA, Anthony ; CHAWATHE, Yatin ; CONSOLVO, Sunny ; HIGHTOWER, Jeffrey ; SMITH, Ian ; SCOTT, James ; SOHN, Tim ; HOWARD, James ; HUGHES, Jeff ; POTTER, Fred ; TABERT, Jason ; POWLEDGE, Pauline ; BORRIELLO, Gaetano ; SCHILIT, Bill: *Place Lab: Device Positioning Using Radio Beacons in the Wild*. 2005. – URL `www.placelab.org/publications/pubs/pervasive-placelab-2005-final.pdf`. – Zugriffsdatum: 2005-04-26

Lübke 2004 LÜBKE, Andre: *Entwurf einer Sicherheitsarchitektur für den Einsatz mobiler Endgeräte*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, April 2004

Lehner 2003 LEHNER, Franz: *Mobile und drahtlose Informationssysteme*. Berlin Heidelberg : Springer Verlag, 2003. – ISBN 3-540-43981-1

Mahmoud 2003a MAHMOUD, Qusay H. ; SUN DEVELOPER NETWORK (Hrsg.): *Part II: The Java APIs for Bluetooth Wireless Technology*. April 2003. – URL `http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth2/`. – Zugriffsdatum: 2005-06-15

Mahmoud 2003b MAHMOUD, Qusay H. ; SUN DEVELOPER NETWORK (Hrsg.): *Wireless Application Programming with J2ME and Bluetooth*. Februar 2003. – URL `http://developers.sun.com/techtopics/mobility/midp/articles/bluetooth1/`. – Zugriffsdatum: 2005-06-15

Oestereich 2005 OESTEREICH, Bernd: *Die UML 2.0 Kurzreferenz für die Praxis*. Oldenbourg, 2005

Ortiz 2004 ORTIZ, C. E. ; SUN DEVELOPER NETWORK (Hrsg.): *Using the Java APIs for Bluetooth Wireless Technology, Part 1 - API Overview*. Dezember 2004. – URL `http://developers.sun.com/techtopics/mobility/apis/articles/bluetoothintro/index.html`. – Zugriffsdatum: 2005-06-16

Ortiz 2005 ORTIZ, C. E. ; SUN DEVELOPER NETWORK (Hrsg.): *Using the Java APIs for Bluetooth, Part 2 - Putting the Core APIs to Work*. Februar 2005. – URL `http://developers.sun.com/techtopics/mobility/apis/articles/bluetoothcore/index.html`. – Zugriffsdatum: 2005-06-16

Puder und Röwekamp 2005 PUDER, Arno ; RÖWEKAMP, Lars: Flipper am Horizont, 10. Java-One in San Francisco. In: *iX* 8 (2005), August, S. 14

Roth 2002 ROTH, Jörg: *Mobile Computing : Grundlagen, Technik, Konzepte*. 1. Auflage. Heidelberg : dpunkt.verlag GmbH, 2002. – ISBN 3-89864-165-1

Schiller 2000 SCHILLER, J.: *Mobilkommunikation - Techniken für das allgegenwärtige Internet*. München : Addison-Wesley Verlag, 2000

Szensny 2005 SZENSNY, Jan: *ePayment in Ferienclubs: Entwurf eines ePaymentsystems unter Nutzung von PDAs*, Hochschule für Angewandte Wissenschaften Hamburg, Studienarbeit, April 2005

Szyperski 1999 SZYPERSKI, Clemens: *Component Software - Beyond Object-Oriented Programming*. New York : Addison-Wesley, 1999

Wehrmann 2001 WEHRMANN, Jens: *Situationsabhängige Dienste - Grundlagen ihrer Entwicklung*, Universität Erlangen, Diplomarbeit, Juli 2001

Wikipedia 2005 WIKIPEDIA (Hrsg.): *Die freie Enzyklopädie*. 2005. – URL `de.wikipedia.org/wiki/`

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 26. August 2005	
Ort, Datum	Unterschrift