CARL VON OSSIETZKY universität OLDENBURG
FAKULTÄT II
DEPARTMENT FÜR INFORMATIK

Abschlussbericht der Projektgruppe „eXplorer“

Kontextsensitive Umgebungserkundung mit mobiler, multimodaler Unterstützung

Universität Oldenburg
Department für Informatik

Stand: 7. Dezember 2005
Verantwortliche: Stefan Andreßen, Arne Bartels, Frank Jellinghaus, Steffen Kruse, Olaf Lehde, Daniel Nüss, Michael Onken, Jens Peternel und Martin Pielot

Inhaltsverzeichnis

Einleitung

Dieser Abschlussbericht entstand im Rahmen der Projektgruppe eXplorer “Kontext-sensitive Umgebungserkundung mit mobiler, multimodaler Unterstützung“ und soll eine umfassende Übersicht über die geleisteten Arbeiten während der Laufzeit der Projektgruppe geben.

Aufgabenstellung und Zielsetzung

Die Aufgabenstellung der Projektgruppe war die Entwicklung einer kontextsensitiven, multimodalen Anwendung zur mobilen Umgebungserkundung. Diese Aufgabenstellung ließ für die Teilnehmer der Projektgruppe viel Raum für eigene Ideen, da die Art der Anwendung nur durch die Eckpunkte Kontextsensitivität, Multimodalität und mobile Umgebungserkundung beschränkt wurde.
Nach einer Phase der Ideenfindung hat sich die Projektgruppe für die Entwicklung eines Spiels entschieden, das die reale Welt mit virtuellen Inhalten verbindet. Die Spieler kommunizieren untereinander und interagieren mit dem Spiel mittels mobiler Endgeräte. Im Spielverlauf erfolgt die Auswahl der Interaktionsmodalitäten anhand des aktuellen Nutzerkontextes.

Aufbau des Abschlussberichts

Dieser Abschlussbericht besteht aus den folgenden Teilen:

- Teil I enthält die Seminarausarbeitungen,
- Teil II beinhaltet die Ergebnisdokumente der Softwareentwicklung. Diese umfassen neben dem Handbuch auch den Projektplan, die Anforderungsdefinition, den Entwurf sowie die Dokumente zur Technologistudie, der Implementierung, dem Qualitätsmanagement und zum Projektabschluss.
- Der Anhang umfasst die Ergebnisdokumente der Vorbereitungsphase sowie die Fallstudie als Entwurfsvorbereitung.

Danksagung

Die Projektgruppe „Kontextsensitive Umgebungserkundung mit mobiler, multimodaler Unterstützung“ möchte sich auf diesem Weg für (mit-)geleistete Arbeit und Unterstützung bei den folgenden Personen bedanken:

- Jun. Prof. Dr. Susanne Boll
- Dipl.-Inform. W. Heuten und

- Dipl.-Inform. P. Klante

Teil I

Seminararbeiten

Teil I des Abschlussberichts enthält die Seminararbeiten, die während der Seminarphase der Projektgruppe „eXplorer“ entstanden sind. Die Arbeiten vertiefen die gesetzten Schwerpunkte der Arbeit der Projektgruppe und bilden einen ersten thematischen Rahmen. Die Ergebnisse sind im weiteren Verlauf der Projektgruppe in den Entwicklungsprozess eingeflossen und finden sich je nach Anwendungsgebiet und Ausprägung in den einzelnen Projektphasen wieder.

Kapitel 1

Usability (Engineering)

Abstract Das Thema der Gebrauchstauglichkeit moderner Software gewinnt mit steigender Benutzeranzahl an Wichtigkeit. Heterogene, anspruchsvolle Benutzer und eine schnelllebige Zeit erhöhen zusätzlich den Bedarf an intuitiv bedienbarer Software. Bei der Entwicklung dieser muss die Notwendigkeit erkannt werden, auf die Gebrauchstauglichkeit zu achten und diese systematisch mit Hilfe neuer Techniken und Testszenarien zu verbessern und direkt an und mit dem Benutzer zu messen. Diese Ausarbeitung liefert eine Begriffsbestimmung der Gebrauchstauglichkeit von Software, stellt Möglichkeiten vor, diese zu messen und ingenieurmäßig zu verbessern und soll anregen und Lust wecken, Gebrauchstauglichkeit ernst zu nehmen.

Ein wenig betrachtetes Feld ist die Gebrauchstauglichkeit von Software auf mobilen Endgeräten, ein kleiner Einblick in die ganz speziellen Probleme und Herausforderungen in diesem Zusammenhang wird zum Abschluss gegeben.

1.1 Vorwort

Die vorliegende Arbeit wurde im Rahmen der Projektgruppe „Kontextsensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“, die semesterübergreifend vom Wintersemester 2004 bis zum Sommersemester 2005 stattfindet und von Juniorprofessorin Susanne Boll geleitet wird, an der Carl von Ossietzky Universität von Oldenburg angefertigt.

1.2 Einleitung

Die Zeiten, in denen relativ komplexe Software in wenigen Monaten von zwei bis drei Menschen geschrieben werden konnte, und diese dann hauptsächlich von ihren Entwicklern oder anderen, ähnlich qualifizierten Menschen genutzt wurde, sind endgültig vorbei. Der Erfolg moderner Software, also deren Verbreitung, wird stark durch ihre Gebrauchstauglichkeit[1] gegenüber dem Endbenutzer bestimmt, welcher in der Regel kein großes technisches Hintergrundwissen hat und auch nicht bereit ist, 500 Seiten Handbuch zu lesen, nur um die Basisfunktionalität der Software nutzen zu können. Mittlerweile nimmt der Anteil der Benutzerschnittstelle am Programmcode einen nicht unerheblichen Teil ein, welcher je nach Art der Software auf bis zu 50 Prozent wachsen kann.
Diese Tatsachen stellen die Entwickler und Designer moderner Software vor neue Herausforderungen. Die

[1] Deutsches Äquivalent von *Usability*, im folgendem immer anstatt *Usability* benutzt.

Zielgruppe muss analysiert und befragt werden, ebenso kommen neue Testszenarien hinzu, die nicht nur traditionell auf funktionale Korrektheit der Software, sondern auch auf deren Benutzbarkeit testen.

Erforderlich ist ein Umdenken nicht nur für Designer und Entwickler, sondern auch für das obere Management, da zusätzliche Testszenarien auch die initialen Projektkosten erhöhen. Um die Gebrauchstauglichkeit von Software zu verbessern, müssen also Techniken entwickelt werden, die

- eine ingenieurmäßige Vorgehensweise ermöglichen,
- eine messbare Verbesserung erzielen,
- die Nöte und Bedürfnisse der Benutzer berücksichtigen, damit sie produktiver arbeiten können und
- technisch realisierbar sind.

Dabei ist das Projektbudget immer ein limitierender Faktor und jede Vorgehensweise bzw. Technik muss sich einer Prüfung der finanziellen Machbarkeit unterziehen.

1.2.1 Was genau ist Gebrauchstauglichkeit?

Die Bedeutung des Terms Gebrauchstauglichkeit ist vielschichtig und damit auch schwer definierbar, einige Ansätze sollen im folgendem kurz vorgestellt werden.

Eine Definition von 1991, aufgestellt von der *„International Organisation for Standardisation (ISO)“* [urlf] und der *„International Electrotechnical Commission (IEC)“* [urle]:

> *„Usability: a set of attributes that bear on the effort needed for use, and on the individual assessment of such use, by a stated or implied set of users.“*

beschreibt Gebrauchstauglichkeit recht technisch und wenig aussagend als eine Menge von Attributen, die für die Nutzung und deren individuelle Beurteilung von einer ausgewiesenen oder implizit festgelegten Anzahl von Benutzern benötigt werden.

Eine etwas konkretere Definition gibt das *ISO* 1998 in 9241-11:

> *„Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.“*

Den Grad der Gebrauchstauglichkeit bestimmt, inwieweit es für bestimmte Benutzer möglich ist, mit einem Produkt ein bestimmtes Ziel effektiv, effizient und zufrieden stellend zu erreichten. Nachzulesen sind diese und weitere Definitionen unter [urll].

Ein letzter hier vorgestellter Definitionsansatz von [urlk] besticht durch seine Einfachheit und Kürze:

> *„Usability addresses the relationship between tools and their users.“*

, bedarf aber weiterer Erläuterungen. Gebrauchstauglichkeit beschreibt also die Beziehung zwischen Werkzeugen und deren Benutzer. [Nie93] identifiziert in diesem Zusammenhang fünf zentrale Punkte:

- Erlernbarkeit
- Effizienz
- Einprägsamkeit

- Fehlerbehandlung
- Zufriedenheit

Erlernbarkeit: Der erste Eindruck, den eine Software hinterlässt, ist wie so oft entscheidend. Zentrale Funktionalitäten sollten schnell erkennbar und einfach zu benutzen sein, da jeder Benutzer sich zu Beginn mit der Software vertraut machen muss. Allerdings muss hier etwas differenziert werden in Bezug auf die Art der Funktionalität und ihrer Frequentierung. Komplexe Simulationssoftware bspw. kann nur mit einem gewissen Vorwissen bedient werden und verlangt eine Einarbeitungsphase. Da die Software aber vermutlich lange und ausgiebig genutzt wird, lohnt sich die Einarbeitungsphase auch. Simple Informationssysteme hingegen werden generell nur selten frequentiert, nämlich immer genau dann, wenn eine Information benötigt wird, und diese sollte dann schnell und einfach ohne Einarbeitungsphase abrufbar sein. Die Lernkurven unterscheiden sich also, je nach Zielsetzung und Funktionalität der Software.

Wie schnell eine Software bedienbar ist, lässt sich leicht über Tests mit potentiellen Benutzern herausfinden. Der Testperson wird eine Aufgabe gestellt und die benötige Zeit zur Erfüllung dieser wird gestoppt. Je größer und vielschichtiger die Zusammensetzung bzw. der Umfang an Testpersonen ist, desto qualitativ besser ist die Aussage, wie schnell die getestete Funktionalität erlernbar ist.

Effizienz: Ein eingearbeiteter Benutzer erreicht ein kontinuierliches Maß an Produktivität mit der Software, welches sich als Nutzungseffizienz darstellt. Diese ist zum ersten abhängig davon, wie schnell der Benutzer an sich arbeiten kann und zum zweiten, welche Unterstützung ihm die Software dabei bietet. Ersteres kann der Softwaredesigner nicht beeinflussen, aber er kann dem Benutzer Möglichkeiten an die Hand geben, immer wiederkehrende Aktionen zu vereinfachen, zu beschleunigen und zu automatisieren. Tastaturbefehle[2] wie STRG-S zum Speichern eines Dokuments und die Möglichkeiten der Makrodefinition sind Möglichkeiten hierfür.

Um die Nutzungseffizienz zu messen geht man analog zum Punkt „Erlernbarkeit“ vor, es muss allerdings drauf geachtet werden, dass alle Testkandidaten ungefähr die gleiche Erfahrung mit der Software haben.

Einprägsamkeit: Wichtig für Benutzer, die eine bestimmte Software regelmäßig, aber mit größeren zeitlichen Abschnitten dazwischen nutzen, ist die Summe der Merkmale, die sie sich über diesen Zeitraum merken müssen. Der optimale Fall wäre eine Software, die einmal benutzt, völlig intuitiv erscheint und somit keinen erneuten Lernaufwand benötigt.

Diese Softwareeigenschaft ist schwerer zu messen als die der Erlernbarkeit und Effizienz, da die Merkfähigkeit von Mensch zu Mensch ebenso stark variiert, wie welche Details er sich merkt. Weiterhin können nur festgelegte Zeiträume überprüft werden, die wirklichen Nutzungszeiten sind schwer vorrausagbar. Eine Möglichkeit ist es, nicht routinierte Benutzer Aufgaben mit dieser Software erfüllen zu lassen und dann anhand von einem Fragenkatalog zu überprüfen, wie viel von den notwendigen Schritten der Benutzer behalten hat.

Fehlerbehandlung: Fehler treten in jeder Software auf, seien sie nun durch falsche oder unvollständige Benutzereingaben oder durch einen Programmierfehler entstanden. Dem Benutzer muss also mitgeteilt werden, das etwas nicht korrekt abläuft, das wann und wie ist hier entscheidend. Wichtig bei allen Fehlerarten ist es, den Benutzer sofort aufzuklären, dass etwas schief gelaufen ist, bspw. kann es sehr frustrierend sein, sich durch zehn Dialoge zu arbeiten und am Ende zu erfahren, dass im zweiten Dialog ein Eingabefehler vorliegt. Dieser sollte sofort behandelt werden.

Sehr vorteilhaft bei kleineren Fehlern ist es, wenn die Software dem Benutzer gleich einige Verbesserungsvorschläge macht, wie es bspw. die Rechtschreibkorrektur von MS Word tut oder ihm bei einer Suchoperation zu dem Suchbegriff verwandte Treffer liefert. Nicht sofort behebbare Fehler sollten nicht stillschweigend in einem undefinierten Zustand der Software enden, sondern sollten mit einer aussage-

[2] Engl.: Short Cuts

kräftigen Meldung angezeigt werden, die das Problem in einer für den Benutzer verständlichen Sprache beschreibt, Fehlercodes, wie in Abbildung 1.1 gezeigt, erfüllen diese Eigenschaft nicht!

Schon bei dem Design muss auf eine klare logische Ablaufsteuerung der einzelnen Funktionalitäten geachtet werden, um den Benutzer davor zu bewahren, Eingabefehler zu begehen und sich jeder Zeit darüber im Klaren zu sein, welcher Schritt gerade ausgeführt wird.

Abbildung 1.1: Nicht aussagekräftige Fehlermeldung

Die Qualität der Fehlerbehandlung ist quantitativ nicht adäquat zu messen, aber es können bestimmte Designrichtlinien, auch mit dem Benutzer zusammen, überprüft werden [Nie93]:

- Gibt es Soforthilfe/Unterstützung bei der Dateneingabe?
- Sie die Fehlermeldungen verständlich und schlagen sie dem Benutzer Lösungsmöglichkeiten vor?
- Wie viele Fehler macht der durchschnittliche Benutzer bei der Abarbeitung einer Aufgabe, gibt es alternative Ansätze, die zu weniger Fehlern verleiten?
- Ist der Ablauf einer Aufgabe logisch strukturiert und ein roter Faden erkennbar?

Zufriedenheit: Völlig subjektiv für den Einzelnen ist die Zufriedenheit mit der Software. Diese Zufriedenheit setzt sich aus dem Funktionsumfang, der Erlernbarkeit, der Effizienz, der Fehlerbehandlung, dem Layout und kleineren Aufmerksamkeiten der Software ihrem Benutzer gegenüber zusammen. Beispielsweise kann die Software den Benutzer beglückwünschen, wenn er eine Aufgabe erfolgreich zum Abschluss gebracht hat oder sie kann sich einfach passiv verhalten, also durch eine nicht vorhandene Fehlermeldung signalisieren, dass alles gut verlaufen ist. Ein automatischer *Updateservice* kann die Zufriedenheit ebenfalls erhöhen, insofern der Benutzer nicht gezwungen ist, jeden Tag viele *Megabytes* herunter zu laden. Als Leitsatz zur Steigerung der Zufriedenheit der Benutzer kann folgendes dienen: Die Software sollte sich dem Benutzer anpassen, nicht der Benutzer der Software.

Ob der Benutzer zufrieden mit der Software ist, kann eigentlich nur durch Befragungen in Erfahrung gebraucht werden, hier sind aber auch durchaus widersprüchliche Angaben zu erwarten, da jeder Benutzer auf andere Aspekte Wert legt, je nach seinen Anforderungen und seinem Geschmack. Falls aber ein allgemeiner Trend nicht ablesbar ist, so war entweder der Fragenkatalog ungeeignet oder der Fokus der Software ist zu ungenau bzw. allgemein gehalten. Hier kann eine detaillierte Auswertung der Ergebnisse helfen.

1.2.2 Zusammenhang zwischen Gebrauchstauglichkeit und Softwareakzeptanz

Der Trend, die Präferenzen und Forderungen des Benutzers ernst zu nehmen und sie in das Softwaredesign einfließen zu lassen, ist unumkehrbar. Um dem Leser einen Eindruck zu vermitteln, wo und inwieweit Gebrauchstauglichkeit zur Softwareakzeptanz beiträgt, soll Abbildung 1.2 betrachtet werden.

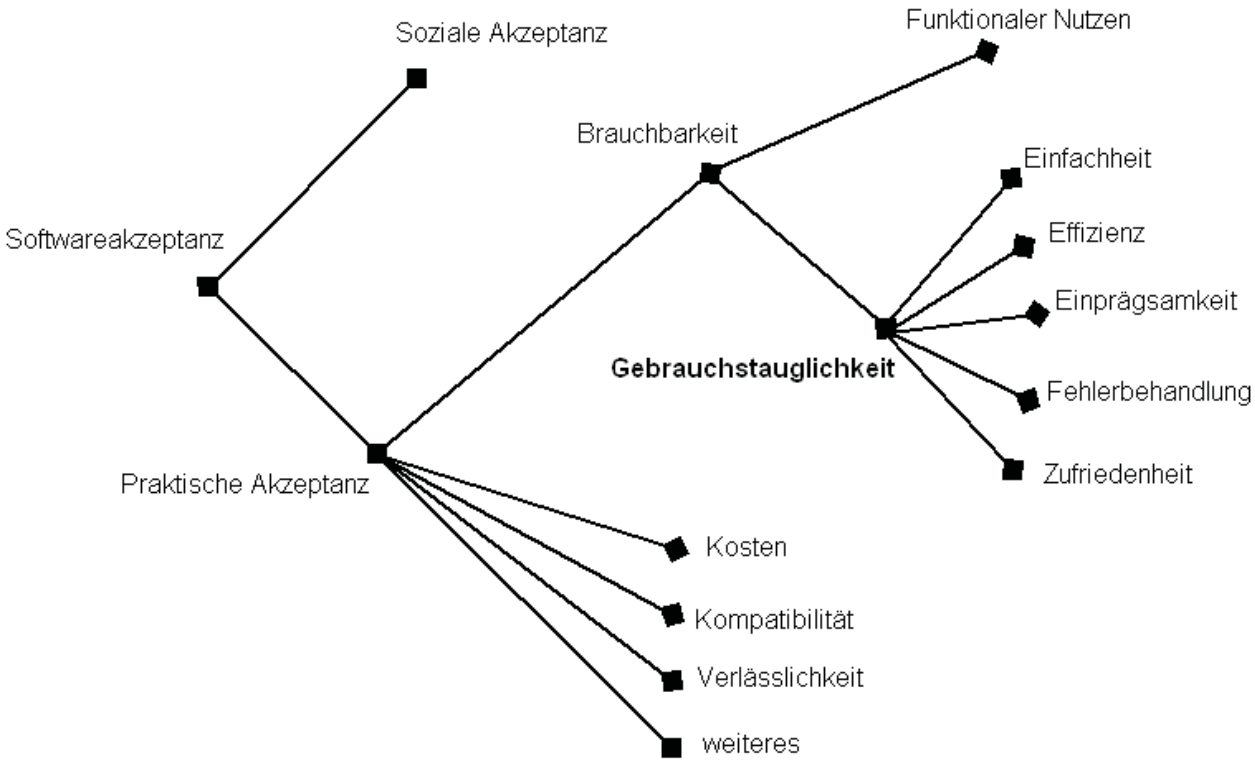


Abbildung 1.2: Softwareakzeptanz

Inhaltlich übernommen aus [Nie93] zeigt Abbildung 1.2, dass vor der Optimierung der Gebrauchstauglichkeit eindeutig technische Grundvoraussetzungen gegeben sein müssen. Die Software muss auf dem Zielsystem zu moderaten Kosten installierbar und anwendbar sein und die funktionalen Anforderungen erfüllen. Soziale Akzeptanz muss ebenfalls gegeben sein, das heißt, der Benutzer muss der Software im Umgang mit sensiblen Daten vertrauen und die Art der Softwareerstellung[3] und ihre Herkunft[4] gutheißen, erst dann ist für den Benutzer die Gebrauchstauglichkeit interessant. Diese entscheidet aber, ob die Software bei gegebenen Grundvoraussetzungen erfolgreich am Markt platziert werden kann oder nicht. Überall dort, wo Benutzer und Software in Interaktion miteinander treten, ist Gebrauchstauglichkeit gefragt und muss systematisch erarbeitet werden, mehr dazu ist in Kapitel 3 nachzulesen.

[3]Gemeint ist z. B. unter welchen Arbeitsbedingungen die Software erstellt worden ist.
[4]Schlechte bilaterale Beziehungen zwischen Ländern können wirtschaftliche Barrieren aufbauen.

1.3 Techniken zur Gebrauchstauglichkeit

Die Entwicklung von gebrauchstauglicher Software[5] ist ein strukturierter Prozess, der über die ganze Projektdauer stattfindet und zusätzlich zur Softwaretechnik auch psychologische Elemente beinhaltet, vgl. [May99]. Zahllose, sich ähnelnde und überschneidende Anleitungen zur Verbesserung der Gebrauchstauglichkeit von Software können in der Fachliteratur gefunden werden. Im Prinzip kann alles, was zur Verbesserung der Gebrauchstauglichkeit von Software beiträgt als Technik ausgelegt werden, nichts desto trotz sollen im folgenden drei sehr zentrale Techniken oder Aspekte vorgestellt werden, auf die später in Kapitel 4 Bezug genommen wird.

1.3.1 Konstruktionszyklus gebrauchstauglicher Software

Englische Fachliteratur bezeichnet diesen Prozess als *„Usability Engineering Lifecycle“*, siehe hierzu [May99] oder auch [Nie93]. Gemeint ist hiermit ein iterativer Vorgang, welcher der Theorie der Softwareentwicklung (Wasserfallmodell bzw. dessen Weiterentwicklungen, usw.), verglichen mit der Vorgehensweise, ähnelt. Mit dem Ziel, konzeptionell geplant und strukturiert eine hohe Gebrauchstauglichkeit zu garantieren, teilt sich der Zyklus nach [May99] in drei große Phasen auf.

1. **Anforderungsanalyse:** Hier werden grundlegende Entscheidungen getroffen, anhand von Benutzerprofilen, Aufgabenanalyse, technischen Möglichkeiten und generellen Designprinzipien. Alle diese Punkte müssen in Zusammenarbeit mit dem Kunden erarbeitet werden, siehe hierzu auch Abschnitt 2.2, da dieser ganz genau weiß, auf welchem Zielsystem die Software installiert werden soll und welchen Anforderungen und Funktionalitäten sie gerecht werden muss. Am Ende dieses Abschnitts steht ein Stilführer, an den sich der weitere Prozess der Softwareentwicklung anlehnt und orientiert.

2. **Entwicklung mit anschließenden Tests:** Basierend auf der Erstellung eines konzeptuellen Modelldesigns, unter Berücksichtigung des erarbeiteten Stilführers, beginnt der iterative Prozess der Entwicklung. Das Modell wird solange dem Kunden bzw. dem Benutzer vorgestellt, bis dieser keine Beanstandungen mehr hat. Mit Hilfe diese Modells und des Stilführers wird ein Bildschirm-Design-Standard entworfen und ein oder mehrere passende Prototypen entwickelt. In den Bildschirm-Design-Standard fließen plattformtechnische Möglichkeiten bzw. Beschränkungen, eventuell schon für die Zielplattform vorhandene Designrichtlinien und natürlich auch Kundenwünsche mit ein. Mit Hilfe dieses Bildschirm-Design-Standards wird die graphische Benutzerschnittstelle entworfen und zusammen mit dem Kunden bzw. Benutzer getestet und verbessert.

3. **Installation:** Entwickler und Kunde bzw. Benutzer sind zufrieden mit dem Resultat ihrer Bemühungen und die Software wird auf dem Zielsystem installiert. Dort wird sie unter realen Bedingungen im Einsatz getestet und erkennbare Probleme und Unzulänglichkeiten behoben, bis alle Beteiligten die Entwicklung für beendet erklären.

Anhand von Abbildung 1.3, die den Konstruktionszyklus zur Erstellung gebrauchstauglicher Software darstellt [May99], können deutlich die iterativen Vorgänge entdeckt werden, die von zentraler Bedeutung für den Konstruktionszyklus sind. Ohne ständiges Überprüfen und Verbessern der bisher erarbeiteten Resultate kann keine wirklich gebrauchstaugliche Software entwickelt werden. Die Entwickler selbst können nur in sehr geringem Maße beurteilen, ob ihre Software gebrauchstauglich ist, da sie über ein Wissen in Bezug auf die Software verfügen, das der normale Benutzer vermutlich niemals erringen wird. Dieses Wissen ist eine Einbahnstraße und kann nicht ignoriert werden, ein Verhalten oder eine Fehlermeldung

[5] Engl.: Usability Engineering

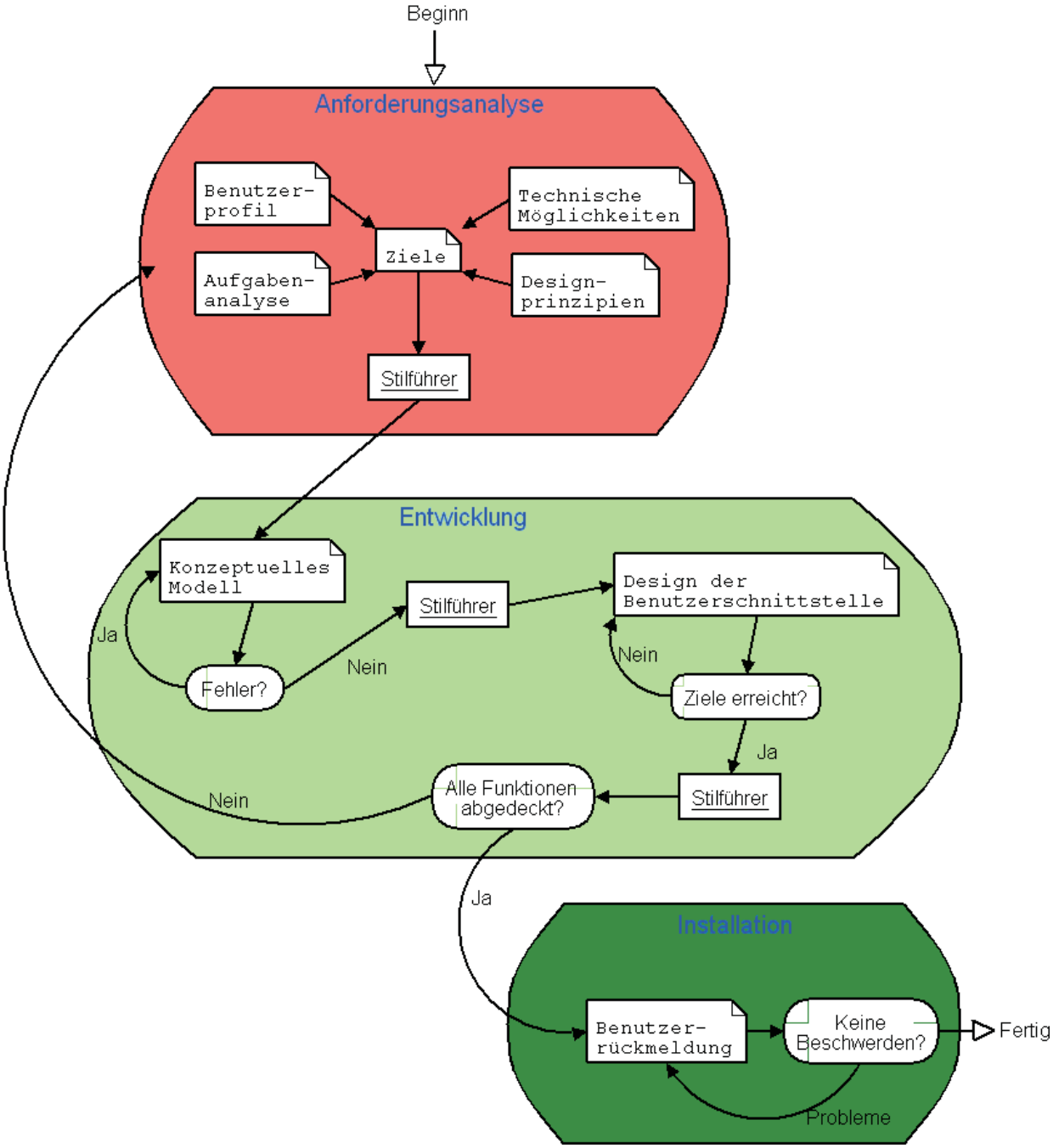


Abbildung 1.3: Vereinfachter Konstruktionszyklus gebrauchstauglicher Software

kann dem/den Entwickler(n) absolut logisch erscheinen, für einen Benutzer muss sie deshalb aber noch lange keinen Sinn ergeben. Der Kontakt zu demjenigen, der die Software später nutzen wird, ist darum unvermeidlich. Begrenzender Faktor der Iterationen ist sicherlich das Projektbudget, in den meisten Fällen von Softwareentwicklung wird es voll ausgeschöpft. Um ein Scheitern des Projektes zu verhindern, müssen Iterationen eventuell beschränkt und Kompromisse gefunden werden.

1.3.2 Kontextabhängiges Design

Unter kontextabhängigem Design[6] verstehen [HB98] den Ansatz, verschiedene benutzerorientierte Techniken für Soft- und Hardwaresysteme zu vereinen und zu nutzen. Basiskriterium um das System zu erstellen und zu strukturieren, sind gesammelte Benutzerdaten aus Befragungen bezüglich Arbeitsweise, Anforderungen und persönlicher Vorzüge. Der Benutzer bzw. die Informationen, die er geliefert hat, stehen im Mittelpunkt aller Entscheidungen. Traditionell musste der Benutzer sich mit der Software abfinden, die basierend von internen Entscheidungen der Entwickler erstellt worden ist. Das Prinzip des Kontextabhängigen Designs ist ein Ansatz, diesen Missstand zu beheben.

Im Zusammenhang des kontextabhängigen Designs fällt oft der Begriff Benutzerzentriertes Design[7]. Von der Natur der Sache her ist das kontextabhängige Design aber schon benutzerzentriert, deshalb geht benutzerzentriertes Design vollständig im kontextabhängigen Design auf.

Kontextabhängiges Design kann als Schablone angesehen werden, um die Vorgehensweise zum Design gebrauchstauglicher Systeme besser verstehen zu können und in geordneten Bahnen verlaufen zu lassen. [HB98] identifizieren drei zentrale Prinzipien, die im folgenden kurz aufgeführt und erläutert werden.

1. **Datenprinzip:** Alle Entscheidungen, welche die Benutzerschnittstelle der Software betreffen, müssen auf einem Fundament vertrauenswürdiger Daten beruhen, die durch kontextbezogene Befragungen des Benutzers gewonnen werden. Ein realistisches Versuchsszenario, welches die spätere Funktionsweise in dem zu entscheidenden Aspekt der Software widerspiegelt, ist dazu am besten, weil nur wenige Benutzer wirklich klar artikulieren können, was sie wünschen und was nicht. Normalerweise deuten die Aussagen eher in Richtung *„So nicht“* oder *„Eher in dieser Art“*, basierend auf dem, was der Benutzer vorgeführt bekommt.

   Es darf dabei nicht vergessen werden, dass die gesammelten Daten einem bestimmten Zweck dienen und nicht um ihrer selbst gesammelt werden. Ziel ist es daher nicht, eine möglichst große Datenmenge zu sammeln, sondern vielmehr die gezielte Gewinnung und Auswertung von Benutzerdaten bezüglich einer bestimmten, kontextbezogenen Fragestellung.

2. **Teamprinzip:** Selten trifft ein Entwickler alle designorientierten Entscheidungen allein. Um Leitlinien, Ergebnisse und zukünftig Vorgehensweisen abzusprechen, sind Teamsitzungen notwendig. Die Planung strukturierter Abläufe ist hier wichtig, da sonst die Gefahr besteht, zwar sehr lange Sitzungen abzuhalten, diese aber mit geringen Resultaten und unklaren Zielsetzungen zu beenden. Jeder Teilnehmer sollte deshalb eine bestimmte, zu ihm passende Rolle einnehmen und vertreten. Zu Beginn jeder Sitzung sollten alle Teilnehmer auf einen möglichst identischen Wissensstand gebracht werden, ein kurzer Bericht, eventuell durch Grafiken aufgewertet, über den aktuellen Stand der Dinge der Aufgaben jedes Teilnehmers, kann dazu beitragen. Eine Aufgabenformulierung am Ende der Sitzung für jeden Teilnehmer hilft jedem konkret zu wissen, wie der nächste Schritt für ihn aussieht. Deutlich gemacht werden sollten auch langfristige Ziele, damit diese über den Details nicht in Vergessenheit geraten.

3. **Designorientiertes Denken:** Allen am Design der Benutzerschnittstelle beteiligten Entwickler muss die Notwendigkeit bewusst gemacht werden, designorientiertes Denken zu entwickeln. Da iterative Prozesse wie das Design einer Benutzerschnittstelle es erfordern, auch hin und wieder einen Schritt zurück zu gehen, schärft designorientiertes Denken den Blick für Strukturiertheit, Klarheit und Konsistenz. Entwickler lernen dabei, Fehler im Design frühzeitig zu erkennen und sie zu beheben.

[6] Engl.: Contextual Design
[7] Engl.: User-centered Design

Interpretiert man den Abbildung 1.3 als grobe Ablaufrichtlinie, so stellt das kontextabhängige Design eine Verfeinerung der Richtlinie mit starkem Akzent auf die Metaebene des Designs dar. Entwickler sind gefordert, einiges über den Prozess der Entwicklung von gebrauchstauglicher Software selber zu lernen, um ihrerseits diese besser erstellen zu können. Die Theorie des kontextabhängigen Designs gibt Anleitung und Hinweise zu Fragen wie:

- Wie und zu welchem Thema/Aspekt/Funktion der Software sollten Benutzerdaten erhoben werden?
- Welche Informationen können diese Daten liefern, direkt oder implizit, welche nicht?
- Was ist nötig, um im Team produktiv zu arbeiten und wie präsentiert man seine Ergebnisse den Teammitgliedern bzw. den Benutzern?

Analog zum Konstruktionszyklus muss auch bei der Methodik des kontextabhängigen Designs die Vorgehensweise im Einklang mit dem Projektbudget stehen, insbesondere die umfangreichen Benutzerbefragungen und die damit anfallende Datenaufbereitung binden zu Beginn des Projekts viele Ressourcen. Trotzdem sind großangelegt Sparmaßnahmen hier nicht angebracht, da

- die Kosten möglicherweise nur verschoben werden, nämlich genau dann, wenn der Benutzer die fertige Software vorgestellt bekommt und mit ihr nicht zufrieden ist und
- Nachbesserungen generell kostenintensiver werden und qualitativ schlechtere Resultate liefern, als von Beginn an die Fehler zu beheben.

1.3.3 Heuristiken zur Gebrauchstauglichkeit

Unabhängig von der konkreten Software und ihren Benutzern sind einige Heuristiken oder auch Leitlinien, dessen Anwendung wahrscheinlich die Gebrauchstauglichkeit der zu erstellenden Software verbessern. Diese Leitlinien ersetzen aber nicht das kontextabhängige Design der Software oder erlauben die Einsparung einiger Konstruktionszyklen. Vielmehr stellen sie einen minimalen Nenner der Erkenntnisse aus vielen Entwicklungen von gebrauchstauglicher Software dar. Jakob Nielson hat unter [urlh] zehn konkrete Leitlinien aufgestellt:

1. Die **Sichtbarkeit des Systemstatus** soll für den Benutzer jederzeit gewährleist sein.
2. Je stärker der **Zusammenhang zwischen System und der wirklichen Welt** in Symbolik und Ausdrucksweise, desto besser versteht der Benutzer das System.
3. Der Benutzer muss die **Freiheit und Möglichkeit** besitzen, seine Aktionen auf dem System wieder rückgängig zu machen, bzw. das System zu verlassen.
4. **Konsistenz** von ähnlichen Funktionen/Aktionen und die Einhaltung **Plattform spezifischer Standards** verwirren den Benutzer nicht.
5. Jeder im **Vorfeld vermeidbare Fehler** ist ein Fehler weniger, um den sich der Benutzer sorgen machen muss.
6. Eine **Steigerung des Wiedererkennungswerts** von Dialogen und anderen sichtbaren Elementen entlastet den Benutzer.
7. **Flexibilität und Effizienz in der Benutzung** für Anfänger wie für Experten. Tastaturkürzel für häufig auftretende Aktionen bspw. verbessern die Effizienz.

8. Durch **Ästhetik und Beschränkung auf das Minimale** aller sichtbaren Elemente wird einer Ablenkung des Benutzers von dem Wesentlichen vorgebeugt.

9. **Unterstützung von Benutzern im Fehlerfall** ist essentiell.

10. **Hilfe und Dokumentation** sollten konkret Abläufe und Aufgaben, die mit dem System bewältigt werden können, in klaren Schritten beinhalten und eine gute Suchfunktion unterstützen.

Eine positive Ergänzung zu obigen Leitlinien ist die Berücksichtigung der Farbenlehre und Objektlehre. Nähere Informationen dazu sind unter [urld] zu finden. Das Strukturieren von Objekten nach den Gesetzen der Nähe, Ähnlichkeit, und weiteren lässt den Benutzer ohne große Erklärungen die logische Zusammengehörigkeit auf einen Blick erkennen. Eine geschickte Farbenwahl, beispielsweise grün für eine erfolgreich abgeschlossene Aktion oder rot für einen Fehler, erhöht die Klarheit der Systemrückmeldungen enorm. Notwendig für die Gestaltung jeder graphischen Oberfläche ist die Wahl von sorgfältig aufeinander abgestimmten Farben.

1.4 Entwicklung gebrauchstauglicher Anwendungen

Systematisches Anwenden der in Kapitel 1.3 vorgestellten Techniken ist Grundvoraussetzung, um im Hinblick auf Gebrauchstauglichkeit Software zu entwickeln, mit der Betonung auf einer genauen Analyse, für wen die Software entwickelt werden soll. Hier spielen Alter, Bildungsstand, Vorwissen, usw. des Benutzers eine Rolle. Je feiner die Software auf eine bestimmte Benutzergruppe abgestimmt wird, desto unwahrscheinlicher ist es, dass sie auch optimal für andere Benutzergruppen geeignet bleibt. Eine exakte Balance zwischen Spezialisierung und Generalisierung ist nicht immer leicht zu finden, und muss von Fall zu Fall, je nach Zielsetzung, neu abgewogen werden.

1.4.1 Standards und deren Anwendung

Drei Arten von Standards können unterschieden werden, nämlich nationale, internationale und unternehmensspezifische Standards. Diese Klassifizierung kann bei der Auswahl von unterstützenden Standards helfen, je nachdem für welche Benutzergruppe die Software erstellt wird. Nationale Standards werden bspw. vom *„American National Standards Institute (ANSI)“* für die Vereinigten Staaten oder für die Bundesrepublik Deutschland von der *„Deutschen Industrie Norm (DIN)“* vorgegeben. Diese Organisationen, vor allem *DIN*, beziehen sich aber größtenteils auf technische Details, die sogar oft einklagbar sind, und weniger auf designtechnische oder psychologische Aspekte. Im Zuge der Globalisierung erscheinen internationale Standards ohnehin deutlich wichtiger: Die *ISO* und andere haben eine Reihe Standards zur *„Human Computer Interaction (HCI)“* hervorgebracht. Diese Standards können die Entwicklung gebrauchstauglicher Software unterstützen, Empfehlungen für Multimedia-Schnittestellen können *ISO* 14915[8] und *IEC* 61997[9] liefern. Das Design einer Benutzerschnittestelle wird in *ISO* 9241 12-17 näher erläutert und *ISO/IEC* 9126 2-3[10] liefern Kriterien, wie die Benutzerschnittestelle bewertet werden kann.

Der *ISO* 9241-Standard beschäftigt sich mit Benutzungscharakteristika von Produkten und ergonomischen Voraussetzungen von visuellen Anzeigegeräten für die Büroarbeit, damit ist er eine gute Anlaufstelle für Fragen der Gebrauchstauglichkeit. Teil 11 beschreibt Leitlinien für Gebrauchstauglichkeit im Allgemeinen und kann helfen, den Kontext zu erkennen, in dem bestimmte Eigenschaften zur Förderung der

[8] Softwareergonomie für Multimedia-Benutzerschnittstellen
[9] Leitlinien für Benutzerschnittstellen von Multimedia-Equipment im allgemeinen Gebrauch
[10] Softwaretechnik - Produktqualität

Gebrauchstauglichkeit von Software notwendig sind. Die Teile 12 - 17 beschreiben wie Informationen in Menü- und Dialogstrukturen, und ähnlichem dargestellt werden können. Natur eines Standards ist immer die Generalisierung. Eigenschaften der konkreten Software, um Gebrauchstauglichkeit zu fördern, können Standards nicht liefern, sondern nur der Benutzer, die zu bewältigenden Aufgaben und die Zielumgebung genau kennt.

Unternehmensstandards beinhalten meistens einen Mix aus internationalen, plattformspezifischen und eigens für spezielle Produkte erarbeiteten Standards. Die Gewichtung liegt hier oft auf den plattformspezifischen Standards, da sie auf der Zielplattform Konsistenz in Benutzung von unterschiedlicher Software versprechen. Konsistenz zum Designstandard der Zielplattform ist ein wichtiger Faktor, zum einen, weil sich dieser Standard schon bewährt haben muss, zum anderen erwartet der Benutzer ihn. Die Einarbeitungszeit sinkt nach [Nie93] um bis zu 50% im Vergleich mit inkonsistenten Benutzerschnittstellen. Der Benutzer kann schneller produktiv mit der Software arbeiten, weil er viele Dinge schon kennt, wie Standarddialoge, Menüstruktur und ähnliches. Dadurch verringert sich die Fehlerquote und die Zufriedenheit mit der Software wird gesteigert. Diese Vorteile überwiegen so stark, dass die Einhaltung der Konsistenz selbst dann empfohlen wird, wenn diese zu kleinen Einbußen in der Gebrauchstauglichkeit der Software führt. Der Benutzer ist eher bereit, eine Funktion auf bekannte Art und Weise auszuführen, als einen neuen Weg zu erlernen, selbst wenn dieser langfristig die Produktivität etwas steigern würde. Nicht nur der Benutzer profitiert davon, sondern auch das Unternehmen. Zum ersten führen zufriedenere Kunden zu höheren Absatzzahlen, und zum anderen kann Software effizienter entwickelt werden, da Diskussionen über Schnittstellendesign auf die Kernpunkte reduziert werden können, da es einen Referenzstandard gibt.

Standards bergen aber auch Gefahren für das Unternehmen, schließlich kostet die Entwicklung eines Standards in den meisten Fällen Geld, welches oft erst mit Folgeprojekten wieder erwirtschaftet werden kann. Es besteht außerdem die Möglichkeit, dass standardisierte Benutzerschnittstellen nur den kleinsten Nenner mehrerer Aspekte in sich vereinen, so dass der Standard zu primitiv für komplexere Anwendungen ausgelegt ist. Der entgegengesetzt Fall ist ebenfalls denkbar, nämlich ein so kompliziertes Regelwerk an Vorschriften, dass basierend auf diesen die Entwicklung von Software deutlich erschwert wird. Zu guter letzt müssen die Entwickler sich bis zum einem gewissen Grad mit dem Standard identifizieren können, da dieser ansonsten schnell unterlaufen wird und die Entwickler mit ihrer eigenen Arbeitsleistung unzufrieden sind.

1.4.2 Das Messen und Testen von Gebrauchstauglichkeit

Um etwas messen zu können muss eine Bezugsbasis vorhanden sein und ein Wertesystem. Das „Messen“ von Gebrauchstauglichkeit ist also eher informell und von der Gewichtung der Zielsetzung abhängig, da es kein Messsystem für Gebrauchstauglichkeit gibt. Geschaffen werden kann von jedem Unternehmen natürlich eine eigene Bewertungsbasis bzw. ein Wertesystem, aber Aussagen wie: „Software xy ist zu 45% so gebrauchstauglich wie Software xyz“, sind immer nur bezogen auf ein spezielles Testsystem gültig, und von einem generellen Standpunkt eher bewertbar als: „Software xy ist deutlich gebrauchstauglicher als Software xyz“.
Im folgenden sollen zwei Methoden aus [Nie93], die Software auf ihre Gebrauchstauglichkeit testen und daraufhin auch in der Lage sind, bewertende Aussagen zu treffen, beleuchtet werden.

Die heuristische Bewertungsmethode beinhaltet die Untersuchung der Benutzerschnittstelle nach allgemeinen Kriterien (siehe hierzu Abschnitt 1.2.1), Intuition und Gefühl, welche Eigenschaften verbesserungswürdig sind. Jeder Testteilnehmer untersucht die Software vorerst alleine, um Beeinflussung durch andere zu unterbinden, später wird eine Sitzung abgehalten, bei der alle gefundenen Mängel zusammen-

getragen werden. Das Ergebnis kann schriftlich festgehalten werden, um zu dokumentieren und anderen Zwecken weiter dienen zu können. Optional einsetzbar ist ein Beobachter, der den Testkandidaten auf Nachfrage oder bei technischen Problemen unterstützen kann. Vorteile bei dieser Bewertungsmethode ist, dass sie schon in einem sehr frühen Entwicklungsstadium durchgeführt werden kann, da die Testkandidaten die Software nicht zwangsläufig in vollem Umfang nutzen, und darum nicht alle Details implementiert sein müssen. [Nie93] hält es sogar für möglich, nur auf Basis der Beschreibung der Benutzerschnittstellen nach Gebrauchstauglichkeitsproblemen zu suchen, der Autor glaubt allerdings nicht, dass dies zu aussagekräftigen Ergebnissen führen kann, allerhöchstens sehr große und offensichtliche Designfehler können erkannt werden und die sollte das geübte Auge der Schnittstellendesigner schon vorher gefunden haben. Heuristische Bewertungsmethoden liefern als Resultat eine Liste mit Problemen der Gebrauchstauglichkeit, systematische Analyse können sie aber nicht leisten, dafür sind sie aber in der Durchführung recht günstig. Laut [Nie93] finden schon 15 Testpersonen im Schnitt 90% der Probleme und es müssen keine umfangreichen Frage- bzw. Aufgabenkataloge erstellt werden. Die heuristische Bewertungsmethode ist von Unternehmensseite also als kosteneffizient anzusehen. Problematisch ist allerdings die Wahl der Testpersonen, [Nie93] unterscheidet hier drei Gruppen:

1. Anfänger, Vorwissen in der Computerbedienung
2. Einfache Experten, vertraut mit der heuristischen Bewertungsmethode
3. Experten, die sowohl mit der heuristischen Bewertungsmethode vertraut sind, als auch mit der Art der zu testenden Benutzerschnittstelle

Je mehr Vorwissen diese Testkandidaten haben, desto mehr Probleme können sie potentiell aufdecken, es ist darum die Aufgabe des Unternehmensm sich die Testpersonen geschickt auszusuchen. Denkbar ist es auch, eine gemischte Gruppe zusammenzustellen, da Anfänger oft andere Probleme finden als Experten, somit wird ein möglichst breites Spektrum abgedeckt.

Die zweite Methode, das explizite Testen von Gebrauchstauglichkeit anhand von wirklichen Benutzern, die echte Aufgaben mit der Software bewältigen, ist für einen systematischen Test auf Gebrauchstauglichkeit der Software nicht zu ersetzen. Zu Beginn wird ein Testplan aufgestellt, in dem festgehalten wird, was und wie getestet wird, welche Systeme eingesetzt werden, welche und wie viele Benutzer benötigt werden und nicht zuletzt, welche Ziele erreicht werden sollen. Ein explizites Testbudget definiert den finanziellen Rahmen.

Der Durchschnitt der Testpersonen sollte dem des durchschnittlichen zukünftigen Benutzers möglichst nahe kommen, dass heißt, die Software ist nicht von irgendeinem gerade zu Verfügung stehenden Menschen zu testen, nötig ist vielmehr eine sorgfältige Vorauswahl. Wenn andere Unternehmen die Auftraggeber der Software sind, können diese auch oft Testbenutzer stellen. Um Software für den generellen Markt zu testen bieten sich Studenten, ehemalige Mitarbeiter oder Zeitarbeitsfirmen mit ihrem breiten Spektrum an verfügbaren menschlichen Ressourcen an. Interessant ist auch die Abwägung zwischen Benutzer, denen die Software völlig unbekannt ist und Benutzern, die schon einige Erfahrungen mit dieser gesammelt haben. Unerfahrene Benutzer tendieren dazu, mit dem einfacheren und offensichtlicheren besser zurechtzukommen, als mit komplizierteren Benutzerschnittstellen, die aber einen größere Effizienz in der Aufgabenbearbeitung zulassen.

Es darf nie vergessen werden, dass die Testpersonen während des Testens von Software unter Druck stehen, da sie wissen, dass sie beobachtet werden und die Resultate ausgewertet werden, einen gute Vorstellung abzuliefern, sich also routiniert im Umgang mit der Software zu zeigen, selbst wenn ihnen vorher versichert wird, das nur die Software selbst getestet werden soll und nicht der Benutzer. Die gestellten Testaufgaben

sollten klar definiert, möglichst klein, repräsentativ für die Benutzung der Software sein und alle großen Aufgabenfelder abdecken. [Nie93] unterteilt den Test in vier Schritte:

1. **Vorbereitung:** Für die Durchführung des Tests werden die Computersysteme eingerichtet, Testmaterie wird ausgelegt, usw. Der Benutzer soll sofort mit dem Testen beginnen können und nicht noch erst Vorbereitungen seinerseits treffen oder treffen lassen.

2. **Einführung:** Der Testbenutzer wird mit den Testparametern vertraut gemacht und über das Ziel dieses Tests aufgeklärt. Vertrauensbildende Maßnahmen wie die Versicherungen, dass nicht der Tester getestet wird und Ergebnisse vertraulich gehandhabt werden, erhöhen den Realismus des Benutzerverhaltens.

3. **Test:** Während des eigentlichen Tests soll der Testbenutzer möglichst ohne Hilfe seine Aufgaben bearbeiten, ob er sie erfolgreich abschließen kann oder nicht. Eine Einmischung der Entwickler durch Hilfestellung verzerrt das Testergebnis.

4. **Nachbesprechung:** Hier kann die Testperson Kommentare abgeben und allgemeine Eindrücke darstellen. Alle Daten dieser Testperson müssen gezeichnet werden und so gespeichert werden, dass später eine systematische Auswertung möglich ist.

Das explizite Testen von Gebrauchstauglichkeit erlaubt genauere Aussagen über die wirkliche Gebrauchstauglichkeit der Software als die heuristische Methode. Durch Analyse der gewonnenen Daten können quantitative Aussagen über die durchschnittliche Dauer, eine bestimmte Aufgabe zu erfüllen, die Anzahl in einem Zeitrahmen erfüllter Aufgaben, Fehlerhäufigkeit, Anzahl der genutzten Funktionalitäten, usw. getroffen werden. Vergleicht man diese Ergebnisse mit anderen Prototypen dieser Software, so können Schlüsse auf Schwachpunkte und Stärken gezogen werden. Wie verlässlich und valide diese Aussagen sind, hängt hauptsächlich von den ausgewählten Testpersonen ab. Als generelle Regel kann angenommen werden, dass, je größer die Anzahl der Versuchspersonen und je repräsentativer diese Personen für den späteren tatsächlichen Benutzer sind, desto qualitativ besser die Testresultate.

1.5 Gebrauchstaugliche Software für mobile Endgeräte

Mobile Endgeräte[11], vor allem Handys, sind mittlerweile weit verbreitet. Trotzdem sind Aussagen zur Gebrauchstauglichkeit eher selten zu finden, vielmehr werden immer mehr technische Finnessen integriert, wie Handys mit eingebauter Kamera und *„Wireless Application Protocol (WAP)“* fähige Handys sind nur zwei Bespiele. Mobile Engeräte sind von Natur aus nicht mit den gewaltigen Ressourcen eines modernen Personal Computers ausgestattet, vor allem der Bildschirm ist um Dimensionen kleiner als bspw. ein 17“ Monitor. Mittlerweile gibt es durchaus schon Handys und *„Personal Digital Assistents (PDA)“* mit Hochauflösenden Farbbildschirmen. Erreicht wurde das aber eher durch Verkleinerung der Pixelgröße, als durch Vergrößerung des Bildschirms, was zwar zu größerer Bildschärfe führt, trotzdem aber die Anzahl der gleichzeitig darstellbaren Objekte nicht deutlich erhöht. Aus diesem Grund haben mobile Endgeräte oft eine tiefe Menustruktur, wie in Abbildung 1.4 anhand eines Handy-Bildschirms zu erkennen ist. Diese Menüstruktur verlangt aber von dem Benutzer, sich zu merken in welcher Ebene er sich gerade befindet. Hauptproblempunkt ist also die Darstellung von Informationen auf dem Bildschirm, hier haben [BFJ+] bei *WAP*-Handys interessante Untersuchungen darüber angestellt, wie Informationen, die nicht komplett

[11] Größere mobile Endgeräte, wie Notebooks und Tablet PCs, werden hier nicht betrachtet, sie können in Bezug auf Gebrauchstauglichkeit von Software prinzipiell mit dem Personal Computer gleichgesetzt werden.

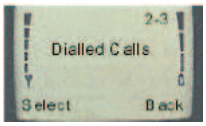


Abbildung 1.4: Traditioneller Handybildschirm

auf den Bildschirm passen, am besten abgerufen werden können. Horizontales- bzw. vertikales Bildschirmrollen und das Anzeigen auf einer neuen Seite werden verglichen. Das Bildschirmrollen hat sich gegenüber dem herkömmlich schlichten Seitentausch sowohl in der Anzahl der Fehler, als auch in der Zeit alle Informationen abzurufen, als überlegen gezeigt. Hier liegt also offensichtlich viel Optimierungspotential frei. Es gibt eine spezielle Version des Opera-Browsers für Geräte mit kleinen Bildschirmen an, dieser Browser nutzt die „Small-Screen Rendering“^{TM}, vgl. [urla]. Die Webseite wird so umstrukturiert, dass horizontales Bildschirmrollen unnötig wird, die Webseite wird gestreckt.

Abbildung 1.5: P503i von DoCoMo

Jakob Nielson hat in seiner *Alertbox* das P503i von DoCoMo (siehe Abbildung 1.5) aufgeführt, welches dem Benutzer einen Art Joystick anbietet um auf dem Bildschirm zweidimensional navigieren zu können. Items mit dem Eingabefokus werden hervorgehoben. Die meisten *PDAs* orientieren sich mit einem berührungssensitiven Bildschirm in eine ähnliche Richtung. Eine Verbesserung der Gebrauchtstauglichkeit ist auch eine Reduzierung der nötigen Tastenbetätigungen, zur schnellen Texteingaben für den *„Short Message Service (SMS)“* unterstützt beispielsweise Handybenutzer eine Funktion mit dem Namen *T9*, die eine Autovervollständigung von begonnenen Worten bietet. Textnachrichten können so deutlich schneller eingegeben werden.

Trotz einiger viel versprechender Ansätze ist die Gebrauchstauglichkeit mobiler Endgeräte eher als schlecht einzustufen. Die beschränkten Eingabenmöglichkeiten kombiniert mit dem durch den kleinen Bildschirm bedingten schlechteren Überblick, machen es dem Benutzer oft schwer das Gerät effizient zu benutzen. Helfen könnte eine Beschränkung der Funktionalität auf das Notwendige, ein Handy ist ein Telefon, keine Kamera. Wo am PC Ausführlichkeit gerne gesehen wird, ist auf kleineren Geräten sehr kompakte Informationsvermittlung nötig. Eine Anpassung der Software unter diesem Gesichtspunkt hilft dem Benutzer, sich auf das Wesentliche zu beschränken.

1.6 Fazit/Ausblick

Gebrauchstauglichkeit von Software ist keine optionale Eigenschaft. Traditionelle Softwaretechniken müssen erweitert werden, um diesem Punkt Rechnung zu tragen. Ein simpler Verweis auf die Dokumentation zur Erklärung einer Funktionalität ist nicht ausreichend, um dem Benutzer das stressfreie Arbeiten mit Software zu ermöglichen.

Umdenken ist nicht nur Sache der Entwickler, sondern auch des Managements. Ein zusätzlicher finanzieller Rahmen muss gegeben sein, um Software zukunftstauglich zu machen. In den heutigen Zeiten gibt es viel Software mit vergleichbarer Funktionalität, hier ist die Verbesserung der Gebrauchstauglichkeit der entscheidende Wettbewerbsvorteil. Die genaue Analyse, wer die Software wie nutzt, ist von großer Bedeu-

tung. Jeder Benutzer ist ein Mensch und möchte als ein solcher behandelt werden. Im Mittelpunkt muss also der Mensch stehen, nicht die Software. Diese hat die einzige Aufgabe, den Menschen zu unterstützen, nicht ihm das Leben schwer zu machen. Software ist ein Werkzeug, keine Zauberei.

Die Nachricht, dass Gebrauchstauglichkeit der Software auch in der mobilen Branche von entscheidender Bedeutung ist, scheint bei den großen Herstellern noch nicht vollständig angekommen zu sein. Möglichst viel Technik auf engstem Raum ist hier das Motto, ob jemand das sinnvoll nutzt, erscheint zweitrangig. Gerade für ältere Menschen ist das oft ein Grund, sich von technologischen Neuentwicklungen fern zu halten. Trotzdem ist der Prozess der Erkenntnis nicht mehr aufzuhalten, dass Software Gebrauchstauglichkeit sein muss. In Zukunft werden auch die mobilen Endgeräte davon zunehmend profitieren.

Kapitel 2

Human-Computer Interaction

abstract Technische Geräte und besonders Computer sind aus unserem Alltag nicht mehr wegzudenken und der problemlose Umgang mit dem Computer hat eine zentrale Bedeutung in der Entwicklung heutiger Computersysteme erreicht. Mit der zunehmenden Komplexität moderner Software wird es immer erforderlicher, dass deren Benutzung so weit wie möglich vereinfacht wird, um den Benutzer nicht durch ein umständliche Handhabung von seinem eigentlichen Ziel, dem Lösen von Aufgaben abzulenken. Nicht dem Benutzer steht die Rolle des Be-Dieners zu, sondern es ist die Aufgabe der Computer den Benutzer in seiner Tätigkeit zu unterstüzen.

2.1 Einleitung

Die vorliegende Seminararbeit befasst sich mit einer Einführung in die Begriffe Interaktion und Mensch-Computer-Interaktion und wurde im Rahmen der Projektgruppe: eXplorer, Kontext-sensitiven Umgebungserkundung mit mobiler multimodaler Unterstützung erstellt. Bei der Bedienung von Computern spielt die Interaktion eine zentrale Rolle. Die Möglichkeiten einen Computer zu bedienen haben sich in den letzten Jahren deutlich verändert. Anfänglich standen nur Lochkarten im Batch-Betrieb zur Verfügung, später dann Interaktion über zeilenorientierte Ein- und Ausgabetechniken. Heute sind es menü- und formulargesteuerte Dialogsysteme innerhalb grafischer Benutzeroberflächen, so genannte WIMP-Oberflächen (*Windows, Icons, Menüs, Pointer*), die mit Zeichengeräten wie der Maus bedient werden. Die Tabelle 1 gibt einen groben Überblick über die Entwicklung von Interaktionsmöglichkeiten. Die rasante Entwicklung im Bereich der Informations- und Kommunikationstechnologien, insbesondere der Einsatz immer kleiner werdender elektronischer Endgeräte wie Laptops, PDA und Handys, verkündet schon heute eine Interaktion über Sprache und Gestik in freien, dreidimensionalen virtuellen Räumen.

2.1.1 Aufbau der Arbeit

Der Inhalt der vorliegenden Arbeit ist in sechs Abschnitte aufgeteilt. Abschnitt eins enthält eine begriffliche Abgrenzung der Begriffe Interaktion und Mensch-Computer-Interaktion. Abschnitt zwei beschreibt die zwei grundsätzlichen Akteure in der Mensch-Computer-Interaktion, den Menschen und den Computer. Abschnitt drei stellt verschiedene Modelle der Interaktion vor, und beschreibt Entwurfsprinzipen für die Mensch-Computer-Interaktion. Abschnitt vier geht auf Interaktion in mobilen Systemen ein und Abschnitt fünf enthält ein Fazit.

Gerätetechnik	Ausgabemöglichkeiten (jeweils hinzukommend)	Eingabemöglichkeiten (jeweils hinzukommend)
Terminals an Großrechnern: Teletypes(Fernschreiber)	Text zeilenweise	Kommandos, Werte
Terminals an Großrechnern: alphanumerische Bildschirme	Text positioniert: Menüs, Masken	Positionierung des Cursors mit Tasten
Terminals: semigrafische Bildschirme, Positioniergeräte (Lightpen, Joystick)	einfache grafische Objekte, Schaltflächen, Boxen	direkte Positionierung und Auslösung
Grafische Bildschirme als Terminals und lokal (Workstations, später PCs)	Grafiken, Fenstersysteme	Drag 'n Drop „Direkte Manipulation“
Hochauflösende Grafik	Bilder	Bilder über Scanner, später Kamera
Audioverarbeitung	Klang, später Sprache	Sprache
Videoverarbeitung	Video (gespeichert / live)	bewegte Bilder, Gestik
VR-Techniken (Virtuelle Umgebungen)	3-D-Darstellung, fühlbare Ausgaben	Bewegungen, Kräfte

Tabelle 2.1: Entwicklung von Interaktionsmöglichkeiten [Hei]

2.1.2 Begriffliche Abgrenzung

Interaktion ist zu einem Schlagwort unserer Zeit geworden. Es gibt interaktives Fernsehen und interaktives Kino, interaktive Spiele und interaktive Bücher, ja sogar interaktive Kühlschränke werden uns in der heutigen Zeit angeboten. Besonders alles rund um das Medium Computer scheint uns als interaktiv suggeriert zu werden. Was bedeutet Interaktion und Interaktiv?

Sucht man im Internet in der freien Enzyklopädie Wikipedia[wik] nach dem Begriff Interaktion, findet man folgende Definition:

> *„Interaktion bezeichnet das wechselseitige aufeinander Einwirken von Akteuren oder Systemen. Der Begriff ist eng verknüpft mit dem der Kommunikation, manchmal werden diese beiden Begriffe sogar synonym verwendet.“*

Und weiter, ebenfalls in der freien Enzyklopädie Wikipedia[wik], unter Interaktionsbegriff in der Informatik:

> *„In der Informatik ist der Begriff der Interaktion mit dem Begriff der Kommunikation identisch: er befasst sich damit, wie einzelne Komponenten eines Systems sich gegenseitig beeinflussen.“*

Eine eindeutige Definition des Begriffs Interaktion gibt es nicht. Grundsätzlich kann aber gesagt werden, dass Interaktion und Kommunikation zwei eng miteinander verbundene Begriffe sind. Diese Arbeit beschäftigt sich mit der Kommunikation zwischen Mensch und Computer, der Mensch-Computer-Interaktion.

2.1.3 Mensch-Computer-Kommunikation oder Mensch-Computer-Interaktion?

Es stellt sich die Frage, inwieweit die Prozesse, die bei der Arbeit eines Menschen mit einem Computer ablaufen, bezeichnet werden können. Die Mensch-Computer-Interaktion (*MCI*) oder englisch human-computer-interaction, (*HCI*) ist ein relative junges Teilgebiet der Informatik. Sie entstand mit dem Aufkommen der ersten Großrechner. Es gibt z.Z. keine vereinbarte Definition des Begriffs Mensch-Computer-Interaktion. Prinzipiell beschäftigt sich die Mensch-Computer-Interaktion mit der benutzergerechten Gestaltung von interaktiven Systemen. Dabei ist die Mensch-Computer-Interaktion ein interdisziplinäres Fachgebiet. Neben Erkenntnissen der Informatik werden auch Erkenntnisse aus der Psychologie, der Arbeitswissenschaft, der Kognitionswissenschaft, der Ergonomie, der Soziologie und dem Design herangezogen.

Letztendlich hat das enorme Wachstum im Bereich der Personal Computer dazu geführt, dass der Verkauf von Computern eng mit der Qualität ihrer Schnittstelle zwischen Mensch und Computer abhängig ist. Auf Grund dieser Tatsache wird, heute und auch in Zukunft, die Mensch-Computer-Interaktion eine immer größere Rolle in der Entwicklung neuer Computersysteme spielen. Dabei ist das Ziel der Mensch-Maschine-Interaktion komplexe Wissen basierte Systeme, wie Datenbanken und Expertensysteme, mit einfachen und effizient zu benutzenden Schnittstellen zu verbinden. Neue Computer sollen nicht nur immer schneller werden, sondern sie sollen auch benutzerfreundlich und für jedermann leicht zu bedienen sein.

Im Bereich Mensch-Computer-Interaktion sind verschiedene Formen der Beschreibung üblich. Sie werden häufig synonym benutzt obwohl sie teilweise unterschiedliche Herangehensweisen beschreiben. Man spricht von:[Hei]

- Mensch-Computer-Interaktion
- Mensch-Maschine-Interaktion
- Mensch-Maschine-Kommunikation
- Mensch-Computer-Kommunikation

Der Begriff **Kommunikation** bezeichnet den Austausch von Nachrichten und Informationen. Dabei tauschen Sender und Empfänger Nachrichten aus. Es gibt aber auch die Form der einseitigen Kommunikation. Zu dieser zählen zum Beispiel die Massenmedien wie Fernsehen und Radio. Voraussetzung ist die Existenz von mindestens zwei Akteuren die, über ein gewisses gemeinsames Verständnis, Nachrichten und Informationen miteinander austauschen. Kommunikationsprobleme zwischen Mensch und Computer sind eine häufige Ursache für Benutzungsprobleme. Dabei liegt das Problem nicht in der Übertragung von Nachrichten, sondern in der Interpretation derselbigen. Fehler in der technischen Übertragung der Signale, so z. B durch eine defekte Maus oder Tastatur, sind kein Problem im Sinne der Mensch-Computer-Interaktion. Die Besonderheit in der Kommunikation zwischen Computer und Benutzer liegt darin, dass ein Computer empfangene Nachrichten eindeutig interpretiert und decodiert (denn dies wurde vom Entwickler vorher festgelegt). Im Gegensatz dazu kann ein und dieselbe Information von einem menschlichen Empfänger, dem Benutzer eines Computers, nicht nur auf verschiedene Arten interpretiert werden, sondern der Benutzer kann sogar etwas als Information betrachten und dieser somit eine Bedeutung zuordnen, wo ein anderer Benutzer gar kein Information erkennt. Ebenso suggeriert der Begriff Kommunikation eine Gleichheit zwischen Mensch und Computer, diese ist aber nicht gegeben. Eine normale verbale Kommunikation mit einem Computer ist mit dem heutigen Stand der Technik noch nicht möglich. Aus diesem Grund werden die Begriffe Mensch-Computer-Kommunikation oder Mensch-Maschine-Kommunikation nur noch

selten benutzt. Außerdem gilt immer das 1. Metakommunikatives Axiom nach Paul Watzlawick: *Man kann nicht nicht kommunizieren*. Selbst das Fehlen jeglicher Rückmeldung eines Computers auf die Eingabe eines Benutzers, kann vom Benutzer als Rückmeldung interpretiert werden.

Der Begriff **Interaktion** wird als jener Vorgang im Rahmen der Mensch-Maschine-Kommunikation verstanden, welcher die Betätigung von Interaktionsmedien sowie Anwendungsaktionen zum Inhalt hat. Daraus ergibt sich, dass Interaktion somit eine Spezialisierung des Begriffs Kommunikation ist.

Bei den Begriffen Mensch-Computer-Interaktion und Mensch-Maschine-Interaktion wird der Computer nicht als gleichwertiger Partner betrachtet. Vielmehr wird der Computer oder die Maschine als ein Gerät angesehen, das auf festgelegte Aktion mit vorherdefinierten Reaktionen antwortet. Der Mensch als Benutzer steht im Zentrum der Betrachtung. Nicht der Mensch muss sich den Anforderungen des Computers anpassen, sondern der Computer muss den Bedürfnissen des Menschen entsprechen. Die Begriffe Mensch-Computer-Interaktion und Mensch-Maschine-Interaktion unterscheiden sich nur insofern, dass in der Mensch-Maschine-Interaktion die gesamte Breite möglicher Maschinen und Geräte betrachtet wird und die Mensch-Computer-Interaktion sich auf das Verhalten zwischen dem Mensch und dem klassischen Computer bezieht.

2.2 Die Akteure - Mensch und Computer

In der Mensch-Computer-Interaktion gibt es zwei grundsätzliche Akteure. Den Menschen als Benutzer (User) und den Computer (System). Beide sprechen ihre eigene Sprache, die sich stark voneinander unterscheiden. Um miteinander zu kommunizieren und somit auch interagieren zu können, benötigen beide Akteure sogenannte Input- und Output Kanäle. Diese Input- und Output Kanäle bilden die Benutzerschnittstelle (User interface) zwischen Mensch und Computer, über die Informationen miteinander ausgetauscht werden. Abbildung 2.1 verdeutlicht dies. Auf der rechten Seite steht der Mensch, auf der linken Seite der Computer. Sie kommunizieren über ihre jeweiligen Input- und Output Kanäle. Diese sind in der Mitte aufgeführt und bilden die Benutzerschnittstelle (User interface). Jedem Output Kanal ist dabei ein zugehöriger Input Kanal zugeordnet, haptisch zu fühlen, visuell zu sehen u.s.w..

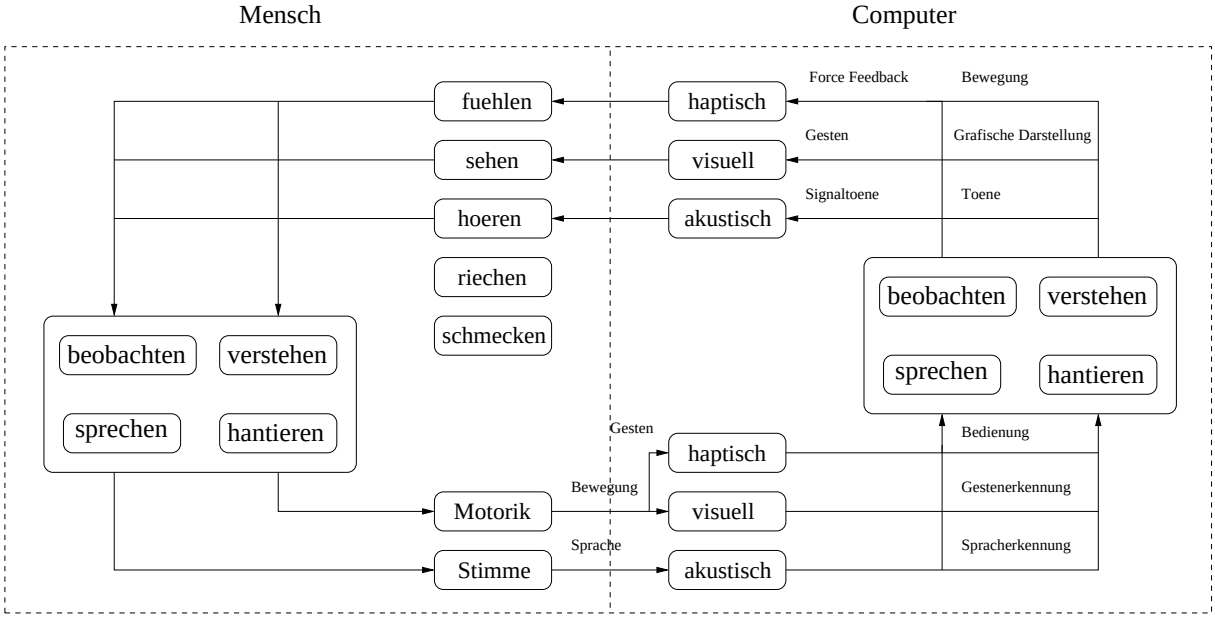


Abbildung 2.1: Gestaltungsrahmen nach *Operquelle*[OEO94]

2.2.1 Der Mensch

Der Kreis der potenziellen Benutzer umfasst die gesamte Bevölkerung. Trotz kultureller Unterschiede haben alle Menschen folgende Eigenschaften gemein. Der Mensch ist in seiner Fähigkeit Daten zu verarbeiten beschränkt und Informationen werden mittels einer Anzahl von Empfangskanälen (Input Kanäle) und Sendekanälen (Output Kanäle) empfangen und gesendet. Die grundsätzlichen Kanäle sind hierbei die fünf Sinneskanäle des Menschen, also Gehörsinn, Geruchssinn, Geschmackssinn, Sehsinn, und Tastsinn.

2.2.1.1 Die Sinne

Gesichtssinn
Der Sehsinn oder auch Gesichtssinn, *visueller kanal*, ist der führende Sinn des Menschen und über ihm werden am meisten Eindrücke aufgenommen. Er unterscheidet in der Wahrnehmung unter Farbe, Form, Bewegung und räumlicher Tiefe. Das sehen liefert nicht nur die differenziertesten Informationen, sondern weist auch die größte Reichweite auf.

Gehörsinn
Der Gehörsinn, *akustischer Kanal*, nimmt Schallschwingungen war. Es unterscheidet in der Wahrnehmung unter Hauptfrequenz, Klangfarbe und Intensität des Schalls. Dabei weist das Gehör eine große Reichweite auf und unterscheidet in Richtung, aus dem ein Signal kommt.

Geruchssinn
Der Geruchssinn, *olfaktorische Kanal*, ist ein eher störanfälliger Sinn. Gerüche können aus großen Entfernungen wahrgenommen werden, aber es gibt in der deutschen Sprache kaum entsprechende Bezeichnungen für Gerüche.

Tastsinn
Der Tastsinn, *haptischer Kanal*, nimmt Eindrücke über die gesamte Fläche de Haut auf. Er unterscheidet in der Wahrnehmung unter Druck, Feuchtigkeit und Temperatur. Die Reichweite ist durch die körperlichen Gegebenheiten stark begrenzt. Dabei stellt das Fühlen über die Haut einen Input Kanal und die Bewegung mit den Händen einen Output Kanal dar.

Geschmackssinn
Der Geschmackssinn, *gustatorischer Kanal*, ist ebenso wie der Geruchssinn ein sehr störanfälliger Sinn. Er erlaubt auch nur eine grobe Klassifizierung in süß, sauer, bitter und salzig. Die Reichweite des Geschmackssinnes ist stark begrenzt und hat von allen Sinnen die kleinste Reichweite.

2.2.1.2 Input Kanäle des Menschen

Alle fünf Sinne Gehörsinn, Geruchssinn, Geschmackssinn, Gesichtssinn, und Tastsinn können somit als Wahrnehmungskanäle oder auch Input-Kanäle verstanden werden. Innerhalb der Interaktion zwischen Mensch und Computer sind aber nur der Gehörsinn, Gesichtssinn, und Tastsinn interessant. Der Geruchssinn und der Geschmackssinn spielen, im Rahmen des heute technisch möglichen, noch keine Rolle. Das bedeutet aber nicht das sie es in Zukunft nicht werden.

2.2.1.3 Output-Kanäle des Menschen

Als Sendekanäle oder auch Output-Kanäle spielen das Berührung- und Bewegungsverhalten (haptischer Kanal und visuelle Kanal) sowie Sprache und Laute (akustischer Kanal) eine Rolle. Außerdem wirkt sich indirekt das Zeitverhalten (Chronemik) in Form von Dauer und Reihenfolge der Interaktion auf die Outputkanäle aus.

Normalerweise werden aber in der heutigen Zeit Informationen über die Augen und die Ohren empfangen und über die Hände, sei es mittels Tastatur oder Maus, gesendet.

2.2.1.4 Speicher

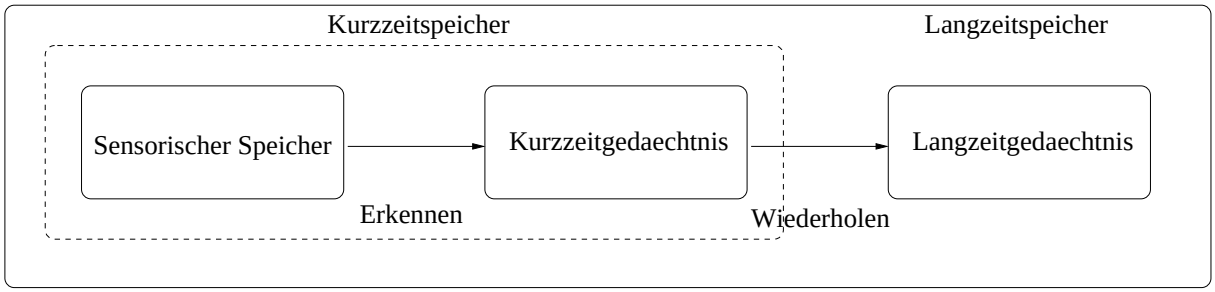


Abbildung 2.2: Struktur des menschlichen Speichers

Zur Speicherung der Informationen stehen dem Menschen zwei grundsätzliche Arten von „Speicher“ zur Verfügung, der Kurzzeitspeicher und der Langzeitspeicher. Der Kurzzeitspeicher besteht aus sensorischem Speicher und Kurzzeitgedächtnis und der Langzeitspeicher besteht aus dem Langzeitgedächtnis. Geht man beim Langzeitspeicher von einer unbegrenzten Speicherkapazität aus, ist die Speicherkapazität des Kurzzeitspeichers begrenzt. Die Speicherkapazität des Kurzzeitspeichers wird in der Regel in chunks angegeben. Der Begriff chunks wurde 1956 von G.A. Miler [Mil56] eingeführt. Millers Hypothese besagt, dass der Umfang der Speicherkapazität des Kurzzeitspeichers etwa 7±2 Chunks beträgt. Miller nennt dies die magische Nummer 7 (*magical number seven*). Dabei ist die Größe der Chunks abhängig von der jeweiligen Person, der Situation und der Art der Information die gespeichert werden soll. Die gespeicherten chunks müssen nicht unbedingt aus elementaren Objekten wie Ziffern oder Buchstaben bestehen. Durch das „chunking“ könne Informationen zusammengefasst werden. Eine Folge von sieben willkürlich gewählten bekannten Wörtern ist ebenso leicht oder schwer zu merken, wie eine Folge von sieben Ziffern oder sieben Buchstaben, obwohl die sieben Wörter zusammen wesentlich mehr als sieben Buchstaben umfassen.

Kurzzeitspeicher

Der **Sensorischen Speicher** ist eng mit den Sinnesorganen, den Empfangskanälen verknüpft. Dabei gibt es für jeden der Empfangskanäle einen eigenen Sensorischen Speicher. Er kann etwas 5 bis 9 Chunks speichern. Die Zeitdauer in der Informationen im Sensorischen Speicher beibehalten werden ist gering, sie spielt in einer Größe von 0.2 bis 0.5 Sekunden. Nach kurzer Zeit werden die Informationen durch Neue von den Empfangskanälen ersetzt. Durch Erkennungsprozesse werden Informationen aus dem sensorischen Speicher in das Kurzzeitgedächtnis übernommen

Das **Kurzzeitgedächtnis**, auch Arbeitsgedächtnis genannt, kann etwa 7 Chunks für eine Zeit von etwa 15 bis 30 Sekunden speichern, wobei das Speichern etwa 0.3 Sekunden dauert. Dabei wird die Korrektheit

der gespeicherten Informationen umso geringer, je mehr Chunks gespeichert sind. Vertraute Begriffe und bekannte Bilder lassen sich dabei am besten speichern. Ebenso funktioniert das Kurzzeitgedächtnis umso besser, je unterschiedlicher die einzelnen Chunks sind. 7 sehr unterschiedliche Bilder lassen sich viel leichter merken als 7 sehr ähnliche Bilder. Durch Wiederholungen werden Informationen ins Langzeitgedächtnis übertragen.

Langzeitspeicher

Das **Langzeitgedächtnis** scheint, so jedenfalls der heutige Stand des Wissens, unbegrenzte Speicherkapazität und unbegrenzte Speicherdauer zu besitzen. Es stellt die Hauptspeicherressource da. Informationen werden im Langzeitspeicher nicht gelöscht, allenfalls kann man gerade nicht auf sie zurückgreifen. Das Langzeitgedächtnis nimmt Informationen nur sehr langsam auf, es dauert pro Chunk minimal 8 Sekunden. Dagegen ist das Abrufen von Chunks deutlich schneller, pro Chunk dauert das Abrufen ca. 2 Sekunden.

Für die Mensch-Computer-Interaktion spielt die Speicherfähigkeit eines Menschen im dem Sinne eine Rolle, das ihre Begrenztheit es notwendig macht, den Menschen nicht durch eine Flut von Informationen und Möglichkeiten zu überlasten. Die gleichzeitig möglichen Interaktionsmöglichkeiten müssen begrenzt werden (hier spielt Millers magische Zahl 7 eine Rolle) ebenso wie das Feedback des Systems begrenzt werden muss.

2.2.1.5 Verarbeitung

Informationen werden vom Menschen nicht nur aufgenommen, sondern auch interpretiert und verarbeitet. Dabei werden vom menschlichen Gehirn verschieden Techniken eingesetzt. Zu diesen zählen Folgerung, Problemlösung, das Erwerben von Fertigkeiten und das Lernen aus Fehlern.

Folgerungen

Mittels Folgerungen wird bereits erworbenes Wissen eingesetzt um neue Zusammenhänge zu erfassen. Hierbei wird in *deduktives*, *inductives* und *abductives* Folgern unterschieden Deduktive Folgerungen basieren auf logische Schlussfolgerungen. Ein Beispiel:

A Hunde sind schwarz

B Bello ist ein Hund

C Bello ist schwarz(!)

Dabei ist wichtig zu beachten, dass keine Wertung der logischen Verknüpfung stattfindet. Auch die Aussage: Steine fallen zu Boden, Äpfel fallen zu Boden, Steine sind Äpfel ist eine korrekte deduktive Folgerung, auch wenn sie inhaltlich falsch ist.

Induktive Folgerungen basieren auf Hypothesen. Auf Grund von mehreren Einzelbeobachtungen wird auf einen allgemeinen Sachverhalt geschlossen. Ein Beispiel:

C Bello ist schwarz(!)

B Bello ist ein Hund

A Hunde sind schwarz (?)

Abduktive Folgerungen basieren auf Wahrscheinlichkeiten. Auf Grund bekannter Beobachtungen wird mit einer bestimmten Wahrscheinlichkeit auf etwas geschlossen. Ein Beispiel:

A Hunde sind schwarz

C Bello ist schwarz(??)

B Bello ist ein Hund

Problemlösung
Als Problemlösung wird der Prozess genannt, bei dem neue Lösungen für unbekannte Aufgaben gefunden werden ohne auf fertige Lösungen zurückgreifen zu können.

Erwerben von Fertigkeiten
Durch das Erwerben von Fertigkeiten steigt die Fähigkeit eines Menschen bekannte Systeme zu nutzen. Sie können je nach Grad der Fertigkeit schneller und effizienter genutzt werden

Fehler
Ist eine Aufgabe nicht durch Folgerungen oder Problemlösungen lösbar, ist der Mensch immer noch in der Lage aus resultierenden Fehlern zu lernen

2.2.2 Der Computer

Die Anzahl der Input und Output Kanäle bei einem Computersystem ist weit vielfältiger als beim Menschen. Hier spielt die schnelle Entwicklung im Bereich der Computertechnik eine große Rolle. Während die körperlichen Fähigkeiten des Menschen gleich geblieben sind, haben sich die Fähigkeiten der Computersysteme in den letzten Jahrzehnten rasant entwickelt und werden sich auch mit Sicherheit in Zukunft rasant weiterentwickeln und verändern.

Wie schon angegeben sind die möglichen Inputkanäle beim Computer vielfältig, deshalb wird hier nur auf die grundsätzlichen Kanäle eingegangen.

2.2.2.1 Input Kanäle des Computers

Input-Kanäle in Form von Texteingabe sind:

Tastatur
Die Tastatur ist mit Sicherheit die am stärksten verbreitete Form der Texteingabe. Die Form der Tastatur ist relativ einheitlich, wenn es auch geringe Unterschiede im Layout der Tastenbelegung gibt. Bei der Tastatur werden Informationen durch das Drücken von Tasten übermittelt.

Handschriftenerkennung
Die Handschriftenerkennung ist häufig im Bereich von PDA und Tablet PC zu finden. Obwohl bei der Handschriftenerkennung die Eingabe über Gesten und Bewegungen erfolgt, zählt sie zur Gruppe der Input-Kanäle in Form von Texteingabe. Grund dafür ist, dass sich die Handschriftenerkennung inhaltlich auf die Eingabe von Zeichen und somit von Texten beschränkt. Sie ersetzt oft die Eingabe über Tastatur und Maus, die Technik ist aber noch nicht ausgereift und fehleranfällig.

Spracherkennung
Die Spracherkennung wird, wie die Handschriftenerkennung, häufig in Bereichen eingesetzt, in der die

Eingabe mittels Tastatur und Maus nicht möglich ist. Über die Spracherkennung hinaus, wird die Möglichkeit der akustischen Eingabe noch nicht eingesetzt. Ebenso ist die Technik noch nicht ausgereift und fehleranfällig.

Inputkanäle in Form von Gesten und Bewegung sind:

Eingabe über Bewegung
Hierzu gehören z.B. Maus, Stift oder berührungsempfindliche Unterlagen. Neben der Tastatur ist auch die Maus ein sehr verbreitetes Eingabegerät. Auch hier gibt es eine Vielzahl von Variationen. Grundsätzlich wird aber die Bewegung der Hand in Steuerbefehle an den Computer ungewandelt. Die Eingabe mittels Stift und einer berührungsempfindlichen Fläche kommt seltener vor, häufig im Bereich von PDAs und Tablet PC oder in Form von TouchScreens in öffentlichen Bereichen wie z.B Bankautomaten. Es gibt in der Forschung eine Vielzahl weiterer Ansätze von haptischen Benutzerschnittstellen. So z.B blickgesteuerte oder gestengesteuerte Eingabegeräte in Form von Datenhandschuhen oder durch Kamerabeobachtung. Diese befinden sich aber noch in der Entwicklung und werden bisher sehr selten eingesetzt.

2.2.2.2 Outputkanäle des Computers

Outputkanäle im Sinne der Interaktion sind hauptsächlich visuelle sowie akustische Kanäle. Es gibt vereinzelt auch haptische Kanäle, die Bewegungen vermitteln z.B in Form von Force Feedback Joysticks. Auf Grund des geringen Informationsgehalt von haptischen Kanälen, sie lassen dem Benutzer einen großen Spielraum für eigene Interpretationen, setzt man sie meist in Kombination mit anderen Kanälen ein. So kann eine vibrierende Maus die Dringlichkeit einer Warnmeldung unterstreichen. Visuelle Kanäle kommen in Form verschiedener Arten von Bildschirmen vor, seien es CRT oder LCD Bildschirme. Akustische Kanäle in Form von Soundkarten und Lautsprechern

2.2.2.3 Speicher

Ebenso wie der Mensch besitzt der Computer auch zwei prinzipielle Arten von Speicher: Der flüchtige Speicher als äquivalent zum menschlichen Kurzzeitspeicher und der nichtflüchtige Speicher als äquivalent zum menschlichen Langzeitspeicher.

Flüchtiger Speicher
Als flüchtigen Speicher wird im Computer jener Speicher bezeichnet, der seine Daten verliert wenn der Computer ausgeschaltet wird. Er tritt in der Form von Hauptspeicher (RAM) und Zwischenspeicher (Cache). im Computer auf. Die Zugriffszeiten sind gering und die Speicherkapazität ist mittlerweile groß.

Nichtflüchtiger Speicher
Als nichtflüchtigen Speicher wird jede Form von Speicher bezeichnet, die auch nach Ausschalten eines Computers, ihre Daten behalten. Dazu gehören z.B Festplatten, Disketten und CDs. Die Speicherkapazität ist riesig und die Zugriffszeit, wenn auch um ein vieles langsamer als die von Flüchtigen Speicher, ist immer noch schnell.

2.2.2.4 Verarbeitung

Daten werden vom Computer mittels Software bearbeitet und angewandt. Dabei ist die Bearbeitung und Anwendung begrenzt durch die zur Verfügung stehende Rechenleistung, Speicherkapazität und durch die

eingesetzte Software.

2.2.3 Die Benutzerschnittstelle

Eine umfassende Definition der Benutzerschnittstelle findet man in den Ausführungen von Wandermacher: [Wan93]

> *„Unter der Benutzerschnittstelle versteht man diejenigen Komponenten und Aspekte der Mensch-Computer-Interaktion, mit denen der Benutzer begrifflich oder über Sinne und Motorik in Verbindung kommt. Zur Benutzerschnittstelle gehören damit auch das notwendige werkzeugunabhängige Aufgabenwissen und das werkzeugspezifische Wissen des Benutzers. Letzteres umfasst das begriffliche Wissen vom Computersystem, zu dem auch das mentale Modell des Systems gehört, sowie kognitive und sensumotorische Fertigkeiten zur Benutzung des Computersystems.„*

Die Benutzerschnittstelle besteht somit im weitesten Sinne aus allen Interaktionsmöglichkeiten zwischen einem Menschen und einem Computer. Sei es in Form der Input- und Output Kanäle des Menschen und Computers, oder in Form von Handbücher, Einführungsprogramme und Hilfefunktionen. Die Benutzerschnittstelle erfüllt eine Vermittlungsfunktion zwischen Mensch und Computer. Dabei muss eine Benutzerschnittstelle nicht nur reine Funktionalität zur Verfügung stellen, sie muss auch in der Lage sein den Benutzer zu motivieren. Die Benutzung muss dem Benutzer Spaß machen und für ihn interessant sein. Die vielfältigen Ansätze, die dabei verfolgt werden, können darin unterschieden werden, ob sie einem Konversationsmodell oder einem Weltmodell zugrunde liegen[Hut95]. Innerhalb des Konversationsmodells wird der Computer als gleichberechtigter Kommunikationspartner verstanden, welcher auf sprachliche Eingaben des Benutzers reagiert. Das Weltmodell setzt eine sichtbare Darstellung des Anwendungsmodells voraus. Die Interaktion erfolgt dabei durch Aktionen, die innerhalb der sichtbaren Darstellung des Anwendungsmodells ausgeführt werden können. Vorraussetzung für eine mögliche Interaktion über eine Benutzerschnittstelle ist ein Mindestmaß an gemeinsamen Symbolen (z.b. Zeichen, Gesten, etc.) auf beiden Seiten, über die eine Verständigung möglich wird. Ein häufiges Problem bei der Benutzung von Benutzerschnittstellen besteht darin, dass es meistens keinen direkten Kontakt zwischen Benutzer und Entwickler der Benutzerschnittstelle gibt. Entwickler müssen sich oft auf ihre Intuition verlassen und versuchen zu antizipieren, was dem Benutzer wichtig ist. Das führt nicht immer zu den gewünschten Ergebnissen, oft werden Benutzerschnittstellen am Benutzer vorbei entwickelt. Um dies zu verhindern ist eine Beteilung des Benutzers an der Entwicklung der Benutzerschnittstelle nötig.

2.3 Die Handlung

Die Anzahl der Möglichkeiten in der ein Mensch mit einem Computer interagieren kann ist vielfältig. Die Bandbreite reicht dabei von Batch-betrieb, bei dem der Benutzer alle Anweisungen auf einmal in das System gibt und auf die Ergebnisse wartet, bis hin zum hoch interaktiven System, die zu jedem Zeitpunkt eine direkte Manipulation ermöglicht.

2.3.1 Die Kluft der Ausführung und der Evaluation

Norman[Nor88] hat für die Interaktion mit technischen Geräten eine Theorie über verschiedene Phasen bei der Durchführung von Bedienhandlungen augestellt.

Norman nennt dies einen *interactive cycle*, einen interaktiven Zyklus. Innerhalb des interaktiven Zyklus wird in zwei Phasen unterschieden. Die Phase der Ausführung (*execution*) und die Phase der Auswertung (*evaluation*). Diese beiden Phasen bestehen nach Norman zusammen aus sieben Schritten:

1. Sich für ein Handlungsziel entscheiden (*establishing the goal*)
2. Die Intention formulieren (*forming the intention*)
3. Eine Handlung spezifizieren (*specifying the action seqquence*)
4. Die Handlung ausführen (*executing the action*)
5. Den Zustand der Welt wahrnehmen (*perceiving the system state*)
6. Den Zustand der Welt interpretieren (*interpreting the system state*)
7. Das Ergebnis auswerten (*evaluating the system state with respect to the goals intentions*)

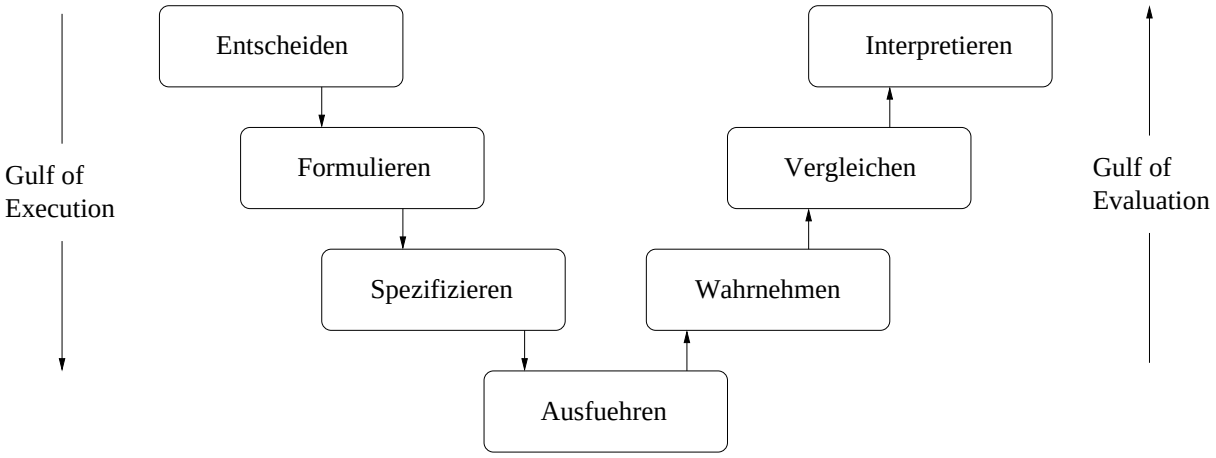


Abbildung 2.3: *Normans* Phasen der Bedienhandlung [DFGB93]

Die Abbildung 2.3 zeigt die sieben Phasen *Normans* [Pre99a] in zeitlicher Reihenfolge. Der Benutzer formuliert in der ersten Phase zuerst ein Handlungsziel und wählt einen Input-Kanal aus, der ihn bei der Erreichung seiner Ziele unterstützen soll. In Phase drei muss der Benutzer in der Lage sein, seine Handlungsziele zu spezifizieren. Um diese Spezifikation durchführen zu können muss dem Benutzer bekannt sein, welche Handlungen überhaupt möglich sind und welche Handlungen zum gewünschten Handlungsziel führen. Bei einer Anwendung (Software) auf einem Computer bedeutet dies normaler weise, dass dem Benutzer ein Anzahl von Menüs, Schaltflächen und Eingabefeldern zur Verfügung stehen, aus denen er diejenigen auswählen kann, die ihn seinem Handlungsziel näher bringen. Hat der Benutzer die Handlung ausgeführt (Phase vier), wird diese in den weiteren Phasen ausgewertet. In der fünften Phase nimmt er die Veränderung der Welt (System) war und interpretiert den neuen Zustand in Phase sechs. In der siebten Phase wird das Ergebnis ausgewertet und vom Benutzer mit seinen Erwartungen verglichen. Das Ergebnis ist meistens die Basis für ein neues Handlungsziel. So entsteht ein Zyklus bei dem sich Benutzer und System jeweils in der Initiative abwechseln.

Die Interaktion zwischen Mensch und Computer beginnt also schon vor der direkten Eingabe von Daten, somit bevor der Benutzer z.B eine Tastatur benutzt um Befehle in einen Computer einzugeben.

Nach *Norman* [Nor88] muss ein interaktives System alle sieben Phasen unterstützen und einen bequemen Übergang zwischen den einzelnen Phasen zu ermöglichen. Außerdem läst sich Normans Theorie gut dazu

benutzen, mögliche Fehlerquellen in der Interaktion aufzuzeigen. Ist es für den Benutzer schwer, seine formulierten Handlungsziele in eine Bedienhandlung umzusetzen, muss ein Benutzer erst angestrengt über die nötigen Kommandos und Parameter nachdenken oder sogar ein Handbuch konsultieren, so spricht Norman vom *Gulf of Execution*, der Kluft der Ausführung. Das kostet den Benutzer nicht nur Zeit sondern längt ihn auch von seinem Ziel ab. Ein einfaches und einprägsames Interaktionsangebot verhilft diese Kluft zu verringern

Treten Fehler in der Phase fünf bis sieben auf, so spricht Norman vom *Gulf of Evaluation* der Kluft bei der Evaluation. Hierzu gehören missverständliche Hinweis oder Fehlermeldungen oder im schlimmsten Fall gar keine Reaktion auf die Bedienhandlung des Benutzers. Ein Beispiel dafür sind Systemnachrichten, die den Benutzer im Unklaren darüber lassen, ob es sich um einen Bestätigung, Warnung oder gar Fehlermeldung handelt. Auch falsche Platzierung und Gestaltung von Systemnachrichten, kleine und unauffällige Nachrichten und schlecht zu lesender Text sind dem *Gulf of Evaluation* zuzuordnen

Hohmann[Hoh02] erweitert das Modell und teilt die Interaktionsangebote in 5 allgemeine Klassen ein:

1. Das Interaktionsangebot insofern bekannt, als dass ihm bestimmte Bedienhandlungen und Resultate zugeschrieben werden. Das Angebot wird entsprechend der Zuschreibung genutzt und die erwartete Reaktion tritt ein.

2. Das Interaktionsangebot insofern bekannt, als dass ihm bestimmte Bedienhandlungen und Resultate zugeschrieben werden. Das Angebot wird entsprechend der Zuschreibung genutzt und die erwartete Reaktion tritt nicht ein.

3. Das Interaktionsangebot insofern unbekannt, als dass ihm keine bestimmte Bedienhandlungen und Resultate zugeschrieben werden. Das Angebot wird *getestet*, die Bedienhandlung ist richtig (da eine Reaktion erfolgt), die Reaktion ist jedoch nicht wünschenswert, d.h. hilft dem Benutzer im Hinblick auf seine Intention nicht.

4. Das Interaktionsangebot insofern unbekannt, als dass ihm keine bestimmte Bedienhandlungen und Resultate zugeschrieben werden. Das Angebot wird *getestet*, die Bedienhandlung ist richtig (da eine Reaktion erfolgt) und die Reaktion ist wünschenswert, d.h. hilft dem Benutzer im Hinblick auf seine Intention.

5. Das Interaktionsangebot insofern unbekannt, als dass ihm keine bestimmte Bedienhandlungen und Resultate zugeschrieben werden. Das Angebot wird *getestet*, die Bedienhandlung ist falsch (da keine Reaktion erfolgt).

Der Fall, das ein Interaktionsangebot unbekannt ist und nicht genutzt wird spielt keine Rolle, da in diesem Fall keine Bedienhandlung ausgeführt wird.

Im ersten und vierten Fall kann man von einer gelungenen Interaktion sprechen. Im zweiten Fall hat der Benutzer aus seiner Sicht alles richtig gemacht, das erwartete Ergebnis ist aber nicht eingetreten. Anhand der Reaktion des Systems ist es ihm aber möglich, seine Handlungsweise zu korrigieren. Im dritten Fall hat der Benutzer keine Erwartungen an das Interaktionsangebot gestellt und konnte so auch nicht enttäuscht werden. Durch die Reaktion des Systems kann der Benutzer aber die Funktion des Interaktionsangebots erlernen. Der fünfte Fall stellt die schlimmste Klasse dar. Ohne Reaktion des Systems ist es dem Benutzer nicht möglich Rückschlüsse zu ziehen, ob seine Bedienhandlung richtig oder falsch war.

Anhand seiner Sieben Stadien der Interaktion hat Norman Designstrategien entworfen um möglichen Fehlern in der Interaktion vorzubeugen. Dabei sollen die Kluft der Evaluation und die Kluft der Auswertung durch folgende Strategien überwunden werden.

Sichtbarkeit
Zustand und Handlungsmöglichkeiten müssen für den Benutzer klar erkennbar sein.

Ein gutes konzeptuelles Modell
Das Systembild muss kohärent in der Darstellung und leicht verständlich sein.

Gute Mappings
Der Benutzer muss die Beziehungen zwischen Handlung und Ergebnis, sowie zwischen Bedienelementen und deren Auswirkungen erkennen.

Feedback
Der Benutzer erhält über alle seine Handlungen ein für ihn verständliches Feedback.

2.3.2 Interaction framework

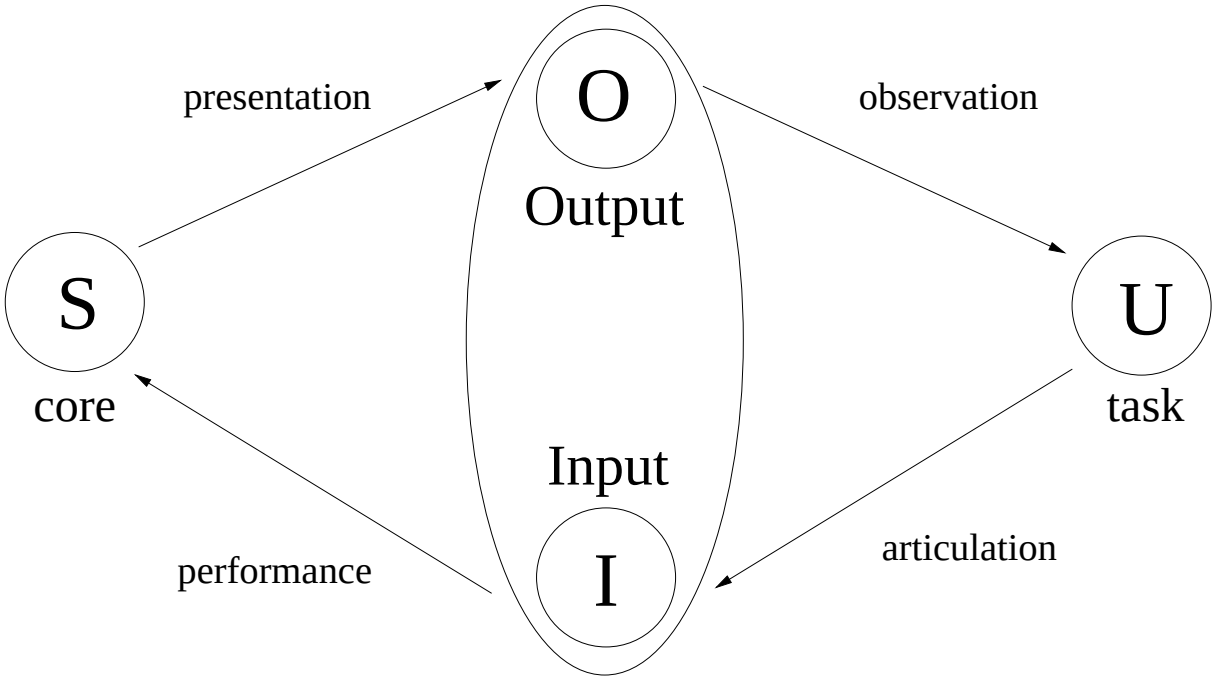


Abbildung 2.4: Modell des interaction framwork

Ein vereinfachtes und realistischeres Modell von Interaktion ist das Interaction Framework. Dabei wird die Interaktion in vier Hauptkomponenten aufgeteilt, Benutzer (*user*), Computer (*system*) und Input sowie Output. Jede Komponente spricht dabei ihre eigene Sprache. Die Sprache des Benutzers (*task language*) und die Sprache des Computers (*core language*) sind dabei unterschiedlich. Input und Output bilden das gemeinsame Interface, dessen Sprache sowohl dem User als auch dem System bekannt ist. Der Ablauf einer Interaktion ist dabei wie folgt: Die gesamte Kommunikation erfolgt über den Input und Output. Der Benutzer muss somit seine Aufgabe in die Input Sprache übersetzen. Die Input Sprache wird in die Sprache des Systems (*core language*) übersetzt. Das System bearbeitet die Aufgabe und präsentiert das Ergebnis über den Output, muss das Ergebnis also in die Output Sprache übersetzen. Der Output wird vom Benutzer beobachtet und in seine eigene Sprache (*task language*) übersetzt. Es gibt somit 4

Arten der Übersetzung: formulieren (*articulation*), ausführen (*performance*), präsentieren (*presentation*) und überwachen (*observation*). Der Benutzer formuliert seine Aufgabe in der Input Sprache. Die Input Sprache wird in die System Sprache übersetzt und ausgeführt. Das Ergebnis wird über die Output Sprache vom System präsentiert und der Output wird vom Benutzer überwacht. Jeder dieser Übersetzungsphasen stellt eine mögliche Fehlerquelle dar. Besonders die Phasen in denen der Benutzer involviert ist, sind anfällig.

2.3.3 Entwurfsprinzipien

Anhand der verschiedenen Modelle der Mensch-Computer-Interaktion lassen sich allgemeingültige Entwurfsprinzipen für Benutzerschnittstellen entwickeln. In der Literatur ist eine Vielzahl von Design-Grundsätzen, Checklisten und Styleguides zu finden. Die Reihenfolge ist beliebig mit einer Ausnahme: *Informiere dich über potentielle Benutzer und ihre Aufgaben* sollte immer am Anfang einer Entwicklung stehen. Die folgende Checklist ist stark an Preim [Pre99a] angelehnt.

Entwurfsprinzipien der Mensch-Computer-Interaktion

Informiere dich über potentielle Benutzer und ihre Aufgaben
Die Analyse von Benutzern ist Vorraussetzung für die Entwicklung interaktiver Systeme. Welche Fähigkeiten und Fertigkeiten besitzt die zukünftige Benutzergruppe? Welche Aufgaben sollen gelöst werden?

Hilf Benutzern, ein mentales Modell zu entwickeln
Konsistenz in der Bedienung erleichtert es dem Benutzer die Funktionen des Systems schnell zu erlernen.

Sprich die Sprache des Benutzers
Formulierungen müssen für den Benutzer verständlich und eindeutig sein.

Mache Systemzustände sichtbar und unterscheidbar
Es muss dem Benutzer möglich sein, jeder Zeit den Zustand des Systems zu erkennen.

Verdeutliche die jeweils möglichen Aktionen
Interaktionsmöglichkeiten die in einem bestimmten Zustand nicht sinnvoll sind, sollten vom Benutzer erst gar nicht auswählbar sein.

Strukturiere die Benutzerschnittstelle
Bedienelemente sollten in logische Funktionsgruppen zusammengefasst werden, um die Übersicht über den Funktionsumfang zu erhöhen.

Stelle erkennbare Rückkopplung sicher
Auf jede Bedienhandlung sollte eine erkennbare Rückkopplung folgen, um den Benutzern klar zu machen, das das System die Bedienhandlung angenommen hat.

Gestalte Schnittstellen adaptierbar
Es muss möglich sein die Schnittstelle den Fähigkeiten des Benutzers anzupassen. Ein erfahrener Benutzer benötigt weniger Hilfestellungen als ein Anfänger. Ebenso muss die Schnittstelle an unterschiedlichen Hardwareumgebungen anpassbar sein.

Kombiniere verschiedene Formen der Interaktion
Interaktionsmöglichkeiten sollten immer aus einer Kombination von Input- und Output Kanälen bestehen.

Vermeide, dass Benutzer sich zu viele Dinge merken müssen
Die Aufnahmekapazität von Benutzern ist durch seine Speicher- und Verarbeitungsmöglichkeiten beschränkt (siehe Kapitel Mensch). Er sollte nicht mit Interaktionsmöglichkeiten überfordert werden.

Ermögliche es, Aktion abzubrechen und rückgängig zu machen
Benutzer machen Fehler und probieren gerne Funktionen aus. Es muss ihnen jeder Zeit möglich sein, Aktionen rückgängig zu machen.

Erleichtere es, Fehler zu erkennen und zu beheben
Treten Fehler auf, sollte dies dem Benutzer deutlich mitgeteilt werden und ihm mögliche Lösungen und Alternativen angeboten werden.

Erkläre die Bedienung des Programms durch Beispiele und weniger durch Formalismen
Die Bedienung sollte nicht nur erklärt, sondern auch dem Benutzen mit Beispielen verdeutlich werden.

Alle hier aufgeführten Entwurfsprinzipien sind stark verallgemeinert. In der Praxis müssen sie an die konkrete Aufgabenstellung angepasst werden.

2.4 Interaktion in mobilen Systemen

Mobile Systeme gehören heute zum Alltag der Menschen. Handys, Laptops und PDAs erfreuen sich einer immer größeren Beliebtheit.

Mobile Systeme haben, im Gegensatz zu stationären Systemen, eine höhere Anforderungen an die Interaktion zwischen Mensch und Computer. Neben geringerer Rechenleistung und Speicherkapazität führt besonders die begrenzte Bandbreite, in Bezug auf die Auswahlmöglichkeiten von Input- und Output Kanälen, zu Problemen. Kaum ein Benutzer ist bereit eine vollständige Tastatur für sein Handy mitzunehmen und das kleine Display eines PDAs kann nicht die gleiche Menge an Informationen darstellen wie ein ausgewachsener Monitor. Entsprechend dieser Problemstellung, müssen mobile interaktive Systeme so entwickelt werden, dass sie den Benutzer mittels Assistenten in der Bedienung des Systems unterstützen und ihn mit Informationen nicht überfordern. Die Lösung für dieses Problem sind mobile Systeme mit multimodaler Unterstützung, deren Benutzerschnittstellen sich der Umgebung anpassen. Ein klassisches Beispiel für ein solches System ist ein Navigationssystem in einem Auto. Die Eingabemöglichkeiten beschränken sich meist auf wenige Bedienelemente. Bei älteren Systemen kann die Eingabe über ein berührungsempfindliches Display, bei modernen Systemen auch sprachgesteuert erfolgen. Der Fahrer muss in der Lage sein, auch während der Fahrt, den Anweisungen des Navigationssystems zu folgen. Zu diesem Zweck erfolgt die Ausgabe der Fahranweisungen nicht nur auf einem grafischen Display, sondern wird auch akustisch über die Lautsprecher ausgegeben. Diese multimodale Unterstützung erlaubt es dem Fahrer, den Anweisungen des Navigationssystems zu folgen ohne den Blick von der Straße zu nehmen.

2.5 Fazit

Es zeigt sich, dass das Gebiet der Mensch-Computer-Interaktion weit über das Gebiet der bloßen Entwicklung von Schnittstellen hinausgeht. Entwickler benötigen fundierte Kenntnisse in diesem Gebiet, damit Anwendungen auch tatsächlich anwendbar sind. Und dabei stehen wir erst am Anfang der Entwicklung der Mensch-Computer-Interaktion. Die Möglichkeit, einen Computer über Sprache und Gesten zu steuern, eröffnet ganz neue Wege innerhalb der Mensch-Computer-Interaktion, stellt aber auch eine große Herausforderung an die Entwickler dar. Dabei begrenzt sich die Entwicklung nicht nur auf die einseitige Kommunikation des Menschen mit dem Computer.
Mit den Robotern Kismet und Cog des MIT und dem Humanoiden Roboter Asimo von Honda, gibt es schon heute erste Exemplare von Robotern die in der Lage sind, über Mimik und Verhalten, Gefühle auszudrücken. Die Zukunft wird es bringen, dass sich der Computer vom leblosen Arbeitsgerät zum

lebendigen Interaktionspartner entwickelt.

Kapitel 3

Multimodalität

Abstract Multimodalität beschreibt im Rahmen der Wahrnehmung die Verarbeitung von Informationen über mehrere Sinnesorgane. Gleichzeitig wird aber auch auf die Eingabe von Informationen hingewiesen, die beim Menschen auch über mehrere Modalitäten erfolgen kann. Dazu werden in dieser Arbeit die physiologischen und psychologischen Grundlagen der Interaktion und Wahrnehmung behandelt. Dann wird versucht eine Begriffsdefinition zu finden und Beispiele aus dem Alltag anzuführen. Anschließend wird geklärt inwieweit Multimodalität in der Mensch-Maschine-Kommunikation eine Rolle spielt. Eine Erläuterung für welche Zielgruppen eine mobile und multimodale Informationsaufnahme und -weitergabe warum wichtig gibt der folgende Abschnitt. Zum Ende des Artikels werden momentane Schnittstellenforschungen in der multimodalen und mobilen Interaktion mit Computern betrachtet.

3.1 Einführung

Multimodalität als Begriff ist nicht schwer zu erfassen. Es bietet sich sogar eine relativ einfache Definition an (s. Abschnitt 3). Im Zusammenhang mit Informationsverarbeitung zwischen Mensch und Maschine ergeben sich weit reichende Forschungen im Grundlagenbereich der „Modalitäten“ und der Umsetzung zu multimodalen Schnittstellen.

Dieser Artikel wird sich zu Beginn der Klärung der dem Menschen zur Verfügung stehenden Modalitäten sowohl im sensorischen, als auch aktorischen Bereich widmen. Dabei werden die physiologischen und psychologischen Grundlagen im Bereich der Sensoren betrachtet. Bei den Interaktionsmöglichkeiten werden die Schwierigkeiten aus ihrem Gebrauch näher erläutert. Nachdem diese Begriffe näher erläutert wurden, kann auch eine Definition von „Multimodalität“ gegeben werden. Anhand der Definition kann man einfache Beispiele aus dem Alltag geben und zeigen, dass diese größtenteils im weiteren Sinne dem Begriff „Multimodalität“ standhalten.

Der letzte Abschnitt widmet sich der Multimodalität im Rahmen der Mensch-Maschine- Interaktion (Human-Computer-Interface (HCI)). Hier soll gezeigt werden, was momentan für multimodale Schnittstellen gegeben sind. Danach wird erläutert welche Zielgruppen für multimodale Anwendungen vorhanden sind und wieso diese überhaupt solche Schnittstellen benötigen. Zum Schluss werden aktuelle multimodale Schnittstellen aus der Forschung gezeigt und wie diese auch im mobilen Einsatz dargestellt werden können.

Dieser Artikel ist im Rahmen der Projektgruppe „eXplorer: Kontext-sensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“ im Wintersemester 2004/2005 entstanden.

3.2 Wahrnehmung und Interaktion

3.2.1 Wahrnehmung der Umwelt

Der Mensch nimmt seine Umwelt über seine Sinne wahr. Diese ermöglichen es ihm verschiedene physikalische und chemische Phänomene mehr oder weniger adäquat verarbeiten und registrieren zu können. Dies wären der visuelle (sehen), auditive/auditorische (hören), olfaktorische (riechen), gustatorische (schmecken) und der taktile/haptische (fühlen von Druck, Schmerz und Temperatur) Sinn. Zusätzlich können noch innere Zustände wie Lage der Gelenke und Extremitäten und die Lage des ganzen Körpers im Raum durch den vestibulären bzw. kinästhetischen Sinn wahrgenommen werden. Dabei wird der Begriff Sinnesmodalität als Gruppe ähnlicher Sinneseindrücke, die durch ein bestimmtes Organ vermittelt werden verstanden.

Der prinzipielle Ablauf der Wahrnehmung ist dabei unabhängig vom Phänomen. Der Reiz trifft auf ein dafür spezialisiertes Sinnesorgan auf. Dabei wird die dem Reiz zueigne Energie in eine andere Energieform umgewandelt, so dass sie im Körper weitergeleitet werden kann und während eines Verarbeitungsprozesses (der nicht ausschließlich im Gehirn stattfindet) zu einer bewussten Wahrnehmung führt. Dabei ist zusätzlich noch zu beachten, dass nicht alle Reize an das Gehirn weitergeleitet werden. Es muss eine Reizschwelle überschritten, so dass eine Weiterleitung vom Sinnesorgan erfolgt. Somit nimmt der Mensch nicht alle Einwirkungen auf ihn wahr. Dies ist insofern nützlich, da eine Aufnahme aller Sinnesreize zu einer Überforderung des Menschen führen würde.

Es sollen im Folgenden die kurzen physiologischen und anatomischen Besonderheiten der eigenen Sinnesorgane erläutert werden.

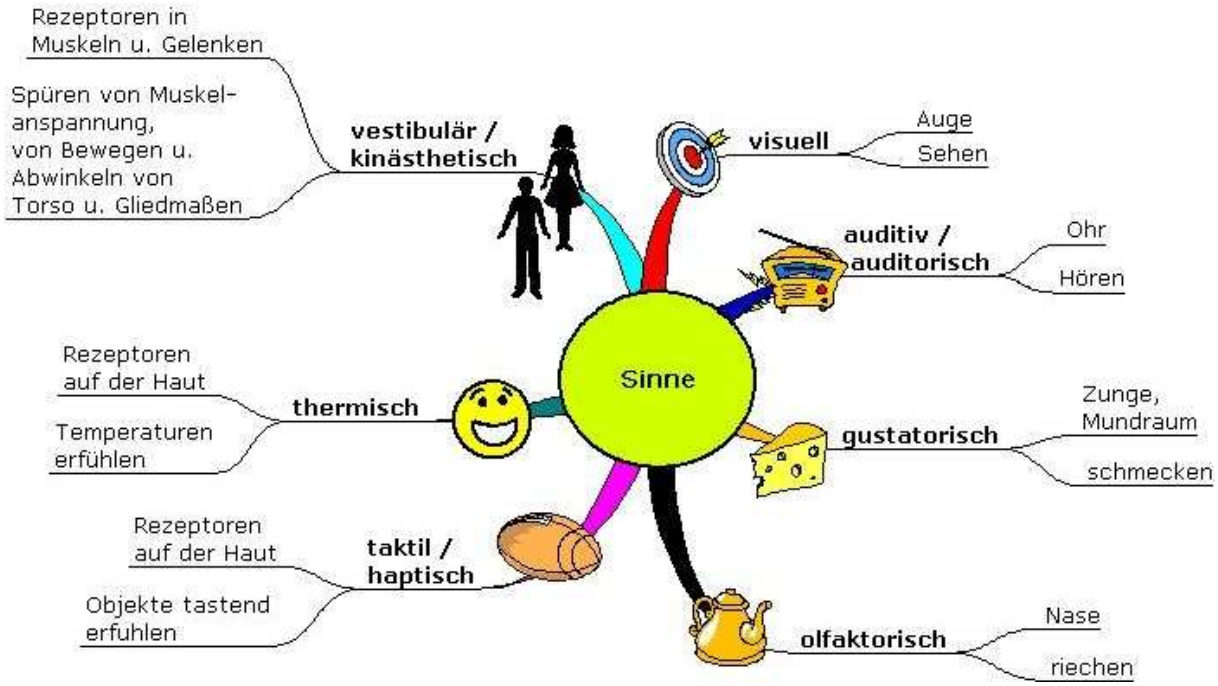


Abbildung 3.1: http://www.teachsam.de/psy/psy_wahrn/psy_wahrn_2_2_1_1.htm

Visuell Der Sehsinn stellt die Hauptaufnahmequelle für Informationen dar. Das Sinnesorgan hierfür ist das Auge, welches eine Ausstülpung des Gehirns darstellt. Dabei trifft das Reizsignal in Form von Licht

auf die Retina und wird aufgrund einer chemischen Reaktion in den Photorezeptoren (den Stäbchen und Zäpfchen) in elektrische Signale umgewandelt. Diese werden über entsprechende Nervenbahnen an die hintere Hirnrinde (visueller Cortex) geleitet. Dort findet die eigentliche Wahrnehmung des Bildes statt. Dazu werden verschiedene Areale im Gehirn zusammengeschaltet, was den letztendlichen Sinneseindruck prägt. Man beachte, dass das Reizsignal nicht unbedingt in Form von Licht auftreten muss. Es muss nur eine adäquate Reizung des Sinnesorgans stattfinden um eine Wahrnehmung auszulösen. Darum sieht man auch Sterne nach einem Faustschlag aufs Auge.

Thermisch Die Rezeptoren hierfür befinden sich in der Haut. Es wird zwischen Kälte und Wärmerezeptoren unterschieden. Diese sind in an verschiedenen Stellen des Körpers unterschiedlich dicht angeordnet.

Auditiv/Auditorisch Der Hörsinn wird über das Ohr realisiert. Dabei wird der Schall über das Außenohr aufgenommen in den Gehörgang geleitet. Im Mittelohr wird das Trommelfell in Schwingung versetzt. Die Schwingungen werden über die Gehörknöchelchen an das Schneckenorgan (Cochlea) übertragen. Die Flüssigkeit in diesem Organ wird bewegt und reizt die Sinnesrezeptoren in Form von kleinsten Härchen. Es findet wieder eine Umwandlung in ein elektrisches Signal statt, welches an die entsprechenden Gehirnareale weitergeleitet wird. Dort wird dann das Gehörte ausgewertet.

Gustatorisch Der Geschmackssinn wird durch Geschmacksknospen auf der Zunge repräsentiert. Hierbei können nur die folgenden Geschmacksrichtungen unterschieden werden: süß, sauer, bitter und salzig. Die Knospen sind charakteristisch auf der Zunge verteilt, so dass nicht jeder Geschmack überall auf der Zunge gleich gut wahrgenommen wird. Der Geschmackssinn wirkt in Kombination mit dem Geruchssinn, denn die Geschmackswahrnehmung ist vielfältiger als die vier Grundgeschmacksarten.

Olfaktorisch Gerüche werden in der so genannten Riechschleimhaut wahrgenommen. Dabei wird der Geruchsreiz durch die Riechzellen empfangen. Dies sind spezielle Nervenzellen, die kurze Ausläufer in die Nasenhöhle haben. Gleichzeitig haben sie Nervenendenverbindungen zum Bulbus olfactorius, welcher ein Hirnnerv ist. Somit ist eine fast direkte Verbindung vom Geruchsorgan zum Gehirn gegeben.

Taktil/Haptisch Der Tastsinn wird wie der Temperatursinn (und auch der Schmerzsinn (Nozizeption)) durch Rezeptoren in der Haut repräsentiert. Diese Rezeptoren nehmen den Druck, der auf sie ausgeübt wahr. Die Umwandlung in ein elektrisches Signal erfolgt, welches an das Gehirn weitergeleitet wird. Die Sinnesempfindung „Schmerz“ muss nicht unbedingt vom Gehirn wahrgenommen werden bevor der Körper darauf reagiert. Durch Rückschaltungen der Nervenbahnen im Rückenmark können Muskeln aktiviert werden bevor bewusst wahrgenommen wird, dass eine Schmerzreaktion aufgetreten ist. Dies wird auch als Reflex bezeichnet.

Vestibulär/Kinästhetisch Dieser Sinneseindruck bezeichnet die Tiefensensibilität und den Lagesinn des Menschen. Hiermit kann die Stellung der Gelenke, Extremitäten und die Lage des ganzen Körpers festgestellt werden. Die Tiefensensibilität wird dem Gehirn über Rezeptoren in Sehnen, Gelenken und Muskeln mitgeteilt. Der Lagesinn wird über das Vestibular-Organ im Innenohr registriert. Dabei werden Positionsänderungen in allen drei räumlichen Dimensionen festgestellt.

Es existiert dabei keine Einschränkung auf das bloße Erkennen des physikalischen Phänomens. Sinneswahrnehmungen werden, wie jeder aus Erfahrung bestätigen kann, differenziert wahrgenommen. Dies

ist entscheidend bei der Verarbeitung von Sinneseindrücken. Sie ist ein wesentlicher Abschnitt der die verarbeitenden Prozesse kennzeichnet, die letztendlich Handlungen beim Menschen auslösen.

Es wird von den so genannten Grunddimensionen der Wahrnehmung geschrieben. Dies wären im Folgenden:

Qualität Hier erfolgt eine Differenzierung innerhalb der Modalität. Dies wären z.B. beim visuellen Sinn die Wahrnehmung von Farben oder beim Geschmackssinn die Unterscheidung von süß und sauer.

Intensität Andere Begriffe hierfür sind Quantität oder Reizstärke. Ein Beispiel wäre die Lautstärke eines Tons oder wie stark der Druck auf eine Körperpartie ist.

Extensität Hiermit wird die zeitliche und räumliche Ausdehnung eines Reizes beschrieben.

Unter psychologischen Gesichtspunkten kann man die Gestaltgesetze betrachten, die vornehmlich visuell aufgenommene Informationen betrachten (aber nicht ausschließlich auf diese beschränkt sind):

Gesetz der Nähe Nahe beieinander liegende Gegenstände werden als zusammengehörig empfunden.

Gesetz der Gleichartigkeit Elemente mit gleichen oder ähnlichen Eigenschaften (Farbe, Form) werden als Gruppe wahrgenommen.

Gesetz der guten Gestalt bzw. Fortsetzung Schneidende Linien werden bevorzugt in Weiterführung ihres ursprünglichen Verlaufs gesehen.

Gesetz der Geschlossenheit Geschlossene Linienzüge werden als strukturelle Einheiten gesehen.

Gesetz des gemeinsamen Schicksals Gemeinsam gleichartige Veränderungen erfahrende Elemente werden als Gruppe aufgefasst.

3.2.2 Interaktion mit der Umwelt

Der Mensch kann je nach Kommunikationskontext mit verschiedenen Mitteln kommunizieren. In der Mensch-Mensch-Kommunikation können dabei Gestik, Mimik, Sprache und Schrift eingesetzt werden. Bei der Mensch-Maschine-Kommunikation werden dabei zum Teil durch technische Einschränkungen, Simulationen der Mensch-Mensch-Kommunikation eingesetzt. Dabei werden z.B. Gesten nicht direkt erkannt, sondern durch ein Zeigegerät wie der Maus durchgeführt. Dies widerspricht der natürlichen Kommunikationsform des Menschen.

Eine allgemeine Form der Kommunikation ist die Gestik. Einige Gesten sind dabei in fast allen Kulturen mit der gleichen Bedeutung behaftet, z.B. wird das Kopfnicken auf der ganzen Welt als ein Zeichen der Zustimmung verstanden. Es gibt jedoch auch gravierende Unterschiede was ein- und dieselbe Geste betrifft. Eine Geste der vollkommenen Zustimmung in einer Kultur können in einer anderen Kultur Teil von wüsten Beschimpfungen darstellen. Dabei gibt es eine sehr differenzierte Einteilung von Gesten, die aber den Rahmen dieser Ausarbeitung sprengen würde. Dabei seien die kodierten Gesten hervorgehoben, welche einen Übergang zu echten Sprachen darstellen. Unter diesen Gesten sind Signalsysteme zu verstehen, wie die Zeichensprachen von Taubstummen.

Weitere köperliche Kommunikationsformen, die jedoch bei weitem nicht so komplexe Begriffe kommunizieren können, umfassen die Körperhaltung und Körperkontakt. Auch gehören gar nicht bewußt kontrollierbare Körpersignale dazu, wie z.B. das Erröten in unangenehmen Situationen.

Die nächste natürliche Form der Kommunikation stellt die Sprache dar. Mit ihr ist es möglich, soweit die Gesprächspartner die gesprochen Sprache auch verstehen, komplexe Sachverhalte in relativ kurzer Zeit darzustellen. Sprache ist eine Leistung des Denkvermögens, sie wird bewußt ausgeführt, im Gegensatz zu Gesten und Körpersprache, die zum Teil sehr deutlich als unbewußt ausgeübt wahrzunehmen sind. Der große Nachteil von Sprachen ist, daß es eine solche Vielzahl von ihnen gibt. Damit werden Sprachbarrieren aufgebaut, die dazu führen, daß es zu Kommunikationsschwierigkeiten zwischen Menschen kommt. Außerdem spielen kulturgeschichtliche Aspekte innerhalb von Sprachen eine Rolle. Das kann zur Folge haben, dass eine Person, die eine Sprache gelernt hat, nicht alle Aspekte dieser Sprache versteht, da sie nicht aus dem Kulturkreis stammt. Sprache ist räumlich abgegrenzten Bereichen eine sehr einfach einzusetzende Kommunikationsform, jedoch durch soziokulturelle und interkulturelle Differenzen zum Teil erheblich eingeschränkt.

Während die gesprochene Sprache fast allen Menschen zur Verfügung steht, ist die Schriftsprache eine abstrakte Leistung, die erlernt wird. Damit ist die Schrift nicht allen Menschen zugänglich. Zwar gibt es in westlichen Ländern kaum Menschen, die dem Schreiben nicht mächtig sind. In Entwicklungsländern kann es genau umgekehrt aussehen. Schrift erlaubt es, genauso wie gesprochene Sprache, komplexe Sachverhalte näher zu bringen. Dabei ist durch die visuelle Darstellungsform eine Unabhängigkeit von der geringen Speicherfähigkeit bei gesprochenen Informationen gegeben. Schrift kann aber genauso vielfältig wie gesprochene Sprachen sein. Dabei seien auf die unterschiedlichen Zeichensysteme wie kyrillisch im slawischen Kulturraum, arabisch im nahen, mittleren und fernen Osten oder kanji, hiragana und katakana im asiatischen Sprachraum hingewiesen.

Betrachtet man nur einen gewissen kulturellen Rahmen, dann stehen dem Menschen unbegrenzte Möglichkeiten zur Verfügung um mit seiner Umgebung in Kontakt zu treten. Natürlich bietet sich ein ganz anderer Sachverhalt, wenn Kulturgrenzen überschritten werden sollen. Natürlich bietet sich aber gerade in den Unterschieden zwischen den Kulturen der Reiz des Unbekannten an, was aber nicht Gegenstand dieser Ausarbeitung ist.

3.3 Multimodalität

3.3.1 Definition

Das Wort Multimodalität an sich, bedeutet etwa „auf vielfältige Art und Weise“, da es aus dem Präfix „multi-“ (lat., viel) und dem Wort „Modus“ (lat., Art und Weise) zusammengesetzt ist. Mit Modalität ist, im Betrachtungsrahmen dieser Arbeit, sowohl eine Form der menschlichen Wahrnehmung, als auch der Interaktion im Zusammenhang mit Verarbeitung von Informationen gemeint. Welche Wahrnehmungsformen und Interaktionsmöglichkeiten es für den Menschen gibt wurde im vorherigen Abschnitt besprochen.

Damit lässt sich der Begriff Multimodalität in einer strengen Form so verstehen:

- Es ist mehr als eine sensorische Modalität für einen Informationskanal verfügbar (z.B. kann die Ausgabe am Computer, das ein Download beendet wurde visuell und/oder akustisch erfolgen) (Wahrnehmungsmodalität).
- Innerhalb eines Informationskanals kann eine Information in mehr als einer Form dargestellt werden (z.B. kann der Befehl zum Ausführen eines Programms mit der Maus angeklickt, gesprochen oder

getippt werden).

Eine weniger strenge Auslegung fordert, dass nur eine Richtung des Informationsaustausches über mehrere Modalitäten erfolgt.

3.3.2 Modalität im Alltag

Multimodale Schnittstellen begegnen uns in verschiedenen Bereichen des Alltags. Wobei meistens das Senden von Signalen an den Benutzer über mehrere Sinneseinwirkungen erfolgt. Es gibt auch vereinzelt Anwendungen, in denen mehr als eine Aktionsmodalität zur Eingabe benutzt wird.

Am häufigsten begegnen uns Fußgängerampeln, die das Signal zum überqueren einerStraße sowohl klassisch mit dem grünen Signalvermitteln, aber auch mit einem speziellen Warnton, der je nach Region unterschiedlich klingt. Dies kann ein hochfrequentes vibrieren sein, welches mehr erfühlt wird als gehört (div. Ampeln in Deutschland). Es treten auch Lautsprecher auf, die solange die Fußgänger ein grünes Signal haben, ein sich wiederholendes Klackgeräusch ausgeben (s. Fußgängerampeln am Julius-Mosen-Platz, Oldenburg, Niedersachsen). Diese zusätzlichen Modalitäten sind für sehbehinderte bzw. blinde Mitbürger ausgelegt.

Mit zunehmendem Einsatz von Bordcomputern innerhalb von Automobilen kommen auch Navigationssysteme, die dem Fahrer die Wegstrecke neben einem Display auch in sprachlicher Form mitteilen. Damit wäre gewährleistet, dass der Fahrer sich auf die Straße vor ihm konzentrieren kann, ohne ständig mit Blicken auf das Kartendisplay abgelenkt zu sein. Je nach Ausstattungsgrad der Autos erlebt man auch die Ausgabe von bestimmten Ereignissen sowohl visuell als auch akustisch (Anschnallen vergessen, Tür auf, Tür mit Fernbedienung abgeschlossen, usw.). Der bloße Vorgang des Autofahrens ist schon ein multimodaler Vorgang. Die Informationen um den Autofahrer herum werden visuell und akustisch wahrgenommen. Beschleunigungen, wie Anfahren oder Bremsen, werden auch kinästhetisch erfasst.

Mobiltelefone bieten schon seit 1994 neben dem akustischen Signal auch einen Vibrationsalarm an, der einen diskret auf die gewünschte Informationsaufnahme hinweist bzw. das Wahrnehmen einer solchen auch in Umgebungen erlaubt, in denen das normale Klingegeräusch nicht zu hören wäre. Auch die Eingabe erfolgt bei einigen Geräten multimodal. Neben den klassischen Zifferntasten, wird auch die Anwahl von Nummern über eine Spracheingabe ermöglicht. Somit hat man, z.B. beim Autofahren die Hände frei beim Telefonieren.

In modernen Fahrstühlen erfolgt die Information in welchem Stockwerk gehalten wird nicht nur visuell, sondern meistens auch akustisch über Sprachausgabe. Damit wird sehbehinderten oder blinden Benutzern die Möglichkeit der Orientierung gegeben. Daneben gibt es auch Fahrstühle, die nicht nur das Stockwerk ansagen, sondern auch mitteilen, ob die Türen geschlossen oder geöffnet werden (s. Fahrstühle in Bahnhöfen der Deutschen Bahn). Zusätzlich sind die Tasten noch mit Braille versehen. So ist auch eine haptische Informationsaufnahme gegeben. Damit sind diese Fahrstühle prinzipiell auch nichtvisuell zu bedienen.

In moderneren Küchengeräten wird der Funktionszustand zusätzlich meist visuell angezeigt. So ist bei Ceran-Herden nicht nur zu erfühlen ob die Kochplatte warm ist oder sich erhitzt. Zusätzlich gibt es eine Konsole vorne am Gerät, die zeigt, dass die entsprechende Stelle sich noch in einem erhitzten Zustand befindet. Bei Mikrowellen wird der erhitzende Zustand über ein Licht im Erhitzungsraum angezeigt und das Beenden des Erwärmungsvorgangs über ein charakteristisches akustisches Signal mitgeteilt. Nicht alltägliche Gegenstände wie Lichtbildautomaten verfügen neben einer visuellen Darstellung der Bedienung auch eine Sprachausgabe, die dem Benutzer eine korrekte Bedienung des Geräts näher bringt. Ein wirklich nicht alltäglicher, aber durchweg multimodaler, Gegenstand wäre ein Flugsimulator. Dieser Si-

mulator liefert dem Benutzer visuelle, akustische und vestibuläre (Sinken oder Steigen des Flugzeugs) Informationen.

Es wären noch viele Beispiele zu nennen, die der nicht zu strengen Definition des Begriffs Multimodalität genügen würden. Es bleibt jedoch anzumerken, dass kaum Gegenstände multimodal gesteuert werden. Dies liegt sehr oft noch an den Grenzen des technisch Erfüllbaren. So ist auf absehbare Zeit keine sprachgesteuerte Waschmaschine zu erwarten, da die Spracherkennungstechnologie noch nicht ausgereift ist. Andererseits ist für solche Gegenstände aber auch keine natürliche Entsprechung in Form von Gesten vorstellbar.

3.4 Interaktion mit dem Computer

3.4.1 Multimodale Interaktion

Im Rahmen der Computerinteraktion wird heutzutage hauptsächlich der visuelle Sinn angesprochen. Dabei besteht die klassische Eingabe- und Ausgabemodalitäten-Kombination aus Ausgabe am Bildschirm und die Eingabe per Tastatur und der Maus als Zeigegerät. Es besteht eine Diskrepanz zwischen empfangener Datenmenge vom Computer und anden Computer gesendeter Daten. Es werden viel mehr Daten aufgenommen als gesendet. Das liegt auch daran, dass zwar Menschen eine hohe Kapazität für visuelle Aufnahme haben. Jedoch eine entsprechend hohe Datenmenge erst mit Gesten oder Sprache generieren können. Dafür sind Computersysteme aber noch nicht ausgelegt. Es gibt Diktiersysteme, bei denen durch Spracheingabe das gesprochene Wort in ein Textverarbeitungssystem eingearbeitet wird. Diese Systeme sind sehr fehleranfällig und müssen in einem aufwändigen Lernprozess an einen einzelnen Benutzer gewöhnt werden. Dabei muss der Anwender entweder unnatürlich deutlich und langsam sprechen um eine hohe Wortwiedererkennungsrate zu erzielen. Ansonsten erreichen solche Systeme Wortschätze von 50000 - 120000 Wörtern und bei 140 Wörtern pro Minute eine Wiedererkennungsrate von 90 Prozent. Dabei ist zu beachten, dass diese Systeme überhaupt keine inhaltliche Analyse des gesprochenen anbieten, sondern nur das Erkennen von Wörtern.

Mit dem zunehmenden Trend der multimedialen Verwendung ist der Einsatz von akustischen Ausgaben fast selbstverständlich geworden. Moderne PC-Hardware bietet grundsätzliche Unterstützung dieser Ausgabeart an. Auditive Signale werden jedoch zum größten Teil dazu benutzt, um die Aufmerksamkeit des Benutzers auf die visuelle Ausgabe zu ziehen.

Wünschenswert wäre es also, diese Eingabeformen zu verbessern bzw. neue Wege der Eingabe zu finden. Dabei müssen Bedingungen wie erhöhte Eingabekapazität und Natürlichkeit der Eingabe beachtet werden. Die Eingabe sollte den gewünschten Intentionen und Gedankengängen so nahe wie möglich stehen. Die umständlichen technischen Hindernisse sollten umgangen werden. Beispielhaft wäre da das Eingabetablett zu nennen um digitale Zeichnungen anzufertigen. Der Stift als Zeigegerät, welcher auf unterschiedliche Druckstärken mit entsprechend veränderter Ausgabe reagiert, ermöglicht einen viel natürlicheren Vorgang des Zeichnens als die Eingabe mit der Maus. Weitere Eingabetechniken, die auf die natürlichen Bewegungsvorgänge des Menschen abzielen zeigen sich in Form von Virtual Reality Systemen. Dabei wird dem Benutzer mit einem Stereodisplay bzw. einer entsprechenden Brille eine virtuelle dreidimensionale Welt präsentiert. Hiermit wird der Tiefensensibilität des Menschen Rechnung getragen. Die Eingabe erfolgt mit 3D-Mäusen oder mit Datenhandschuhen, welche die haptischen Fähigkeiten des Menschen anspricht. Dabei ist eine realistische und detaillierte Darstellung mit hohen Rechenleistung verbunden. Anwendungen dieser Technologien finden sich z.B. im Automobilbau in der Forschung und Entwicklung.

Zwar nicht sehr natürlich, dafür aber schnell und flexibel ausführbar sind so genannte Mouse Gestures, die

während der Navigation im Opera Browser (und mittlerweile als Plugin in anderen Browsern) eingesetzt werden können. Prinzipiell aktiviert der Benutzer mit der gedrückten rechten Maustaste diesen Modus und führt ein Kommando mit einer entsprechenden Bewegung aus und lässt die rechte Maustaste los, z.B. kann in der Besuchshistorie zurücknavigiert werden, in dem eine Mausgeste nach links ausgeführt wird. Vorwärts geht es mit einer Mausgeste nach rechts. Ein neues Fenster wird mit einer Bewegung nach unten geöffnet. Die Navigation ist für die meisten Befehle nicht intuitiv, jedoch ist nach kurzer eine viel schnellere Navigation möglich, als dies mit der klassischen Benutzung der Maus oder Tastaturbefehle möglich wäre.

Die Frage nach Einsatzmöglichkeiten von multimodalen Ein- und Ausgaben richtet sich auch nach den Bedürfnissen der abgezielten Personengruppen. Diese sollen im folgenden Abschnitt näher gebracht werden.

3.4.2 Zielgruppen

Wie schon im Abschnitt zu multimodalen Schnittstellen im Alltag war zu bemerken, dass generell Personenkreise, die eine Einschränkung in einer Modalität erfahren haben, auf multimodale Schnittstellen angewiesen sind. Dies ermöglicht im gewissen Maße ein Kompensation. Beispielhaft wären blinde Personen zu nennen, die auf eine alternative Ausgabeform vom Bildschirminhalten angewiesen sind. Dies besteht in der Tat in Form von haptischen Braille-Displays. Dabei ist es generell nicht nur möglich textuelle Informationen aufzubereiten. Es ist möglich Informationen grafischer Natur wie Diagramme in einer nicht-visuellen Art darzustellen. Dabei ist natürlich zu beachten, dass evtl. nicht direkt von multimodaler Ein- und Ausgabe gesprochen werden kann, wenn nur eine Modalität durch eine andere ersetzt wird.

Auch Personen, die keine Einschränkungen unterliegen können durch den Einsatz von multimodalen Informationsaufnahme eine sinnvolle Ergänzung zu bestehenden Interaktionsformen erfahren. Es wurde gezeigt, dass die haptische Informationsaufnahme eine geringere mentale Belastung mit sich brachte. Dies könnte dahingehend eingesetzt werden, dass größere Informationsmengen bewältigt werden können oder auch mit höherer Konzentrationsfähigkeit gearbeitet werden kann. Somit bietet sich ein Transfer von momentanen Benutzungsschnittstellen zu multimodalen Formen der Interaktion an, da die Datenmengen im alltäglichen Bereich immer mehr zunehmen. Außerdem könnte somit in Bereichen wie z.B. Autofahren eine erhöhte Sicherheit zu Tage treten. Die Anwendungsbereiche für multimodalen Einsatz liegen bei Benutzergruppen, deren klassische Sensoren, wie Sehsinn, schon belegt sind, die jedoch zusätzliche Signale zur besseren Funktion oder Kommunikation einsetzen wollen. Dies können auditive Signale (dabei nicht wie im Computerbereich aufmerksam auf sich ziehende Signale, sondern z.B. Stereosignale, die auf die Richtung eines bestimmten Ziels hindeuten) sein oder Anzüge, die haptische Signale an entsprechende Körperstellen leiten (näheres im Abschnitt Schnittstellen). Personen die auf Navigationshilfen angewiesen sind, denen aber zusätzliche Modalitäten angeboten werden müssen, da die klassischen Modalitäten schon belegt sind oder ergänzt werden müssen.

Im Rahmen mobiler Anwendungen kommen auch Navigationshilfen mit zusätzlicher Erläuterung der näheren Umgebung in den Sinn. Dafür bestehen mannigfalte Anwendergruppen, wie z.B. Touristen, Museumsbesucher, Supermarktbesucher. Diesen könnten mit Hilfe mobiler Endgeräte über ein visuelles Display ihre momentane Lage dargestellt werden. Gleichzeitig könnte ein Audiokommentar darstellen wo sich z.B. der Tourist befindet und welche Sehenswürdigkeiten sich in der Nähe befinden. In Supermärkten könnten sich diese Endgeräte am Einkaufswagen befinden.

Auch in verschiedenen Berufsgruppen können vor allem mobile multimodale Schnittstellen und Endgeräte von Nutzen sein. Ein Beispiel wäre das Wartungspersonal von Flugzeugen, welche generell komplexes technisches Wissen bereithalten müssen. Diese könnten über eine entsprechende Apparatur in Form einer

Brille, in welcher ein Prismendisplay eingefasst ist, zusätzlicher Informationen anzeigen (z.B. „nächste Schraube 30 Nm nach rechts drehen“). Eventuell könnte dies durch Feststellen der Blickrichtung unterstützt werden. Gleichzeitig sollte eine Eingabe um Daten zu erhalten nicht durch Tastatur oder Maus erfolgen, da in diesem Fall die Hände für andere Aufgaben benötigt werden. Eingabe über Gesten oder Sprache wäre wünschenswert.

Auch sind Szenarien von Feuerwehrleuten vorstellbar, die sich über ein grafisches Display in mit normalen Sinnen schwer zu manövrierenden Umgebungen zurechtfinden. Dabei könnte ein mobiler Computer einen Grundriss von einem brennenden Gebäude einblenden. Dabei hätte der Anwender seine Hände frei für seien eigentlichen Aufgaben. So was wäre auch für Sicherheitskräfte vorstellbar, die zusätzlich zu ihrer normalen Sicht über ein solches Display Informationen über den Verbleib ihrer Kollegen einblenden lassen können. Somit wäre eine gezieltere Koordination möglich.

Ein Beispiel wie solche Schnittstellen in etwa aussehen könnten liefert Steve Mann. In seiner Brille ist ein transparentes Computerdisplay und seine Sicht der Realität wird mit Computerdaten ergänzt. Gleichzeitig zeichnet eine Kamera seine visuelle und auditive Wahrnehmung auf. Auf diese Daten kann er wahlfrei zugreifen. Somit hat er die Möglichkeit z.B. bei Konversationen mit Personen, Hintergrundinformationen zu diesen aufzurufen. Waren die dazugehörenden Apparaturen nur sehr umständlich in die Kleidung integriert bzw. mehr Last als Nutzen, zeichnen sich mit der fortschreitenden Miniaturisierung immer leistungsfähigerer Elektronik realistische Einsatzgebiete ab. Mehr Informationen gibt es unter http://www.wearcomp.org und http://wearcam.org/mann.htm.

3.4.3 Schnittstellen

Multimodale Schnittstellen bieten die Möglichkeit einer effektiven Interaktion mit dem Computer an. Dabei sollten die natürlichen Kommunikationsformen des Menschen Beachtung finden, wie z.B. Sprache und Gestik. Die Maschine sollte sich dem Menschen anpassen und nicht umgekehrt. Dies würde den Weg zu einer breiteren Benutzergruppe öffnen und den Einzug von Computertechnologie in alltäglichere Bereiche ermöglichen. In diesem Abschnitt sollen exemplarisch einige Schnittstellen vorgestellt, die neue Wege der Interaktion zwischen Mensch und Maschine ermöglichen, dabei soll der Blick auf bestehende Schnittstellen nicht verwehrt werden. Es sollen aber auch Ideen zu solchenSchnittstellen erwähnt werden, die demonstrieren, wie solche Interaktionsformen aussehen könnten. Dabei sollen auch mobile Anwendungen näher betrachtet werden.

Die prinzipielle Idee für Eingaben in den Computer ist der Transport von Information vom Gehirn des Benutzers in den Computer. Dabei ist zu beachten, dass eine hohe Bandbreite an Information gefordert ist und eine Natürlichkeit der Eingabe. Dies würde Fähigkeiten berücksichtigen, die sich im Laufe der Evolution entwickelt haben und auch auf dem individuellen Erfahrungsschatz basieren. Sie stehen damit im Gegensatz zur antrainierten Benutzung von Computern. Außerdem muss der Kontext des Einsatzes der Schnittstelle beachtet werden. Wie bei den obigen Beispielen für Zielgruppen multimodaler Anwendungen müssen Nichtverfügbarkeit von bestimmten Köperteilen in Kauf genommen werden. So könnte ein Pilot, dessen Hände für die Steuerung des Flugzeugs eingesetzt werden, keine Eingaben mit seinen Händen durchführen.

Nichtsdestotrotz stellen die Hände die hauptsächliche Eingabemodalität für Computer. Sie bedienen die Tastaturen, die normalerweise im „QWERTY/QWERTZ“-Layout vorhanden sind. Im Rahmen der Minitiarisierung von Computern blieb die Tastatur der unhandlichste Teil. Somit wurden Bemühungen wach um Alternativen zur Tastatur zu finden. Eine Alternative zur klassischen Tastatur sind Chord-Tastaturen, die mit einer Hand bedient werden und eine Taste für jeden Finger besitzt. Die ursprüngliche Idee war, dass der Benutzer mit der linken Hand die Chord-Tastatur bedient und mit der anderen Hand

die Maus. Es war möglich Zeichen mit Hilfe kombinierter Tastendrücke zu erzeugen. Diese Idee hat sich nicht durchgesetzt, wird aber im Rahmen der Minitiarisierung in Betracht gezogen.

Während Tastaturen jeglicher diskrete Eingaben aufzeichnen, sind aber auch kontinuierliche Eingaben mit Hilfe der Hände möglich. Dabei können die kontinuierlichen Signale auf vielfältige Weise unterschieden werden (Art der Bewegung, absolute oder relative Positionsmeldung, welche physikalische Eigenschaft wird festgestellt (Position oder Kraft), Anzahl der Dimensionen innerhalb derer die Eingabe erfolgt, direkte oder indirekte Kontrolle). Beispiel für so einen kontinuierlichen, eindimensionalen und der Bewegungsart nach auf Rotation beruhenden Eingabemechanismus wäre der rotary knob der innerhalb der iDrive-Technologie von BMW eingesetzt wird http://www.bmw.com/e65/id14/3_a91_idrive.jsp. Das bekannteste Beispiel wäre die klassische Computermaus, die zweidimensional und linear bewegt wird. Dabei erfolgt die Kontrolle aber indirekt, was im Gegensatz zu einem Touchscreen steht. Dreidimensionale Eingabegeräte bedienen sich dreier Magnetspulen, die die jeweiligen Dimensionen repräsentieren. Es gibt auch Schnittstellen zum Erfassen von Gesten. Diese sind aber auf sehr gute Technologie zum Erkennen von Dreidimensionalität. Außerdem ist zu beachten, dass Gesten sehr unpräzise und nicht exakt wiederholbar ausgeführt werden. Zum Einsatz kommen dieselben Techniken wie bei dreidimensionalen Eingabegeräten und Kameratechnologien.

Füße dienen auch der Eingabe. Simple Schnittstellen befinden sich in Autos und in Musikinstrumenten. Diese Form der Eingabe können analog an Computern angewandt werden um diskrete oder kontinuierliche einfache Informationen aufzunehmen.

Die Kopfpositionen können durch dreidimensionale Ortungsgeräte erfasst werden und ermöglichen so z.B. eine Positionierung des Eingabefokus. Virtual Reality Systeme setzen dazu 3D-Helme ein, die aufgrund der Kopfposition das Bild, das dem Benutzer präsentiert wird, zoomen und drehen. Dabei werden zur Eingabe noch Datenhandschuhe benutzt.

Sprache wurde weiter oben schon behandelt. Es bietet aber eine alternative Eingabemöglichkeit, wenn andere Eingabekanäle des Menschen durch anderer Tätigkeiten beschäftigt sind. Denkbar sind auch Differenzierung von gesagtem in Kombination mit Gesten, quasi das zeigen auf einen Gegenstand und dem Sprechen der Aktion, die auf den Gegenstand ausgeführt werden soll.

Augenbewegungen können auch der Eingabe von Informationen dienen. Dabei kann die Position der Pupille bestimmt werden und an den Computer gereicht werden. Damit wäre eine schnelle und natürliche Form der Eingabe gegeben. Die Aktionen vom Computer müssen jedoch mit Bedacht gewählt werden, da Augenbewegungen unwillkürlich erfolgen und somit in einem unpassenden Moment ungewünschte Befehle ausgeführt werden.

Bevor sensorische Interfaces angesprochen werden, soll noch als Übergang zwischen Eingabe und Ausgabe das Verständnis von „haptischen Displays“ nach angerissen werden. Hierunter wird dem Begriff „haptic“ neben dem taktilen auch die propiorezeptorischen Wahrnehmungen verstanden. Ein haptisches Display liefert demnach Informationen vom Benutzer und liefert aber programmabhängige Informationen an den Benutzern zurück. Eine normale Computermaus würde, wenn sie ein haptisches Display wäre, würde abhängig von den Gesten die mit ihr durch den Benutzer ausgeführt werden, verschiedene 'haptische' Signale an den Benutzern zurückschicken. Sie wäre somit als Informationskanal geeignet. Zudem wird noch eine Einteilung von haptischen Displays in aktive und passive Formen vorgenommen. Passive Displays zeichnen sich durch z.B. programmierbare Bremsen aus oder sollten eine Veränderung ihrer Elastizität erlauben (härter oder weicher werden). Durch die Programmierbarkeit dieser Geräte soll eine Kontrolle durch den Computer ermöglicht werden. Bei aktiven Displays ist durch das Feedback vom User über das Gerät an den Computer eine Interaktionsmöglichkeit gegeben. Dabei kann das Gerät die Position messen und eine Kraft ausüben (isotonisch) oder aber als Positionsquelle dienen und die Kraft, die

auf das Gerät ausgeübt wird messen (isometrisch). Als Beispiel eines aktiven haptischen Displays ist die Simulation der Steuerung des Lenkrads eines Rennautos gegeben. Zusammenfassend kann man sagen, dass haptische Displays auf zwei Wegen kommunizieren, also haptische Eingaben und Ausgaben erlauben. Die Voraussetzung dafür jedoch ist die Programmierbarkeit der Geräte.

Beispiele für diese haptischen Displays sind:

- Exoskelette: Diese Geräte werden „angezogen“ und liegen dem Körper von außen an. Dabei wird die menschliche Biomechanik genutzt um Geräte zu steuern. Gleichzeitig geben Sie dem Körper mit Hilfe von Hydraulik die Zustände der benutzten Geräte wieder.

Abbildung 3.2: Quelle: http://rn01.rednova.com/news/stories/3/2004/03/11/story001.html

- Force-Feedback-Joysticks: Der Benutzer verwendet das Gerät für Eingaben am Computer. Bei bestimmten Ereignissen (z.B. Lattenschuss im Fußballspiel) gibt der Joystick ein Rücksignal in Form von Vibration.

- Der vorher schon erwähnte rotary knob für BMW iDrive.

Abbildung 3.3: Quelle: http://www.bmw.com/e65/id14.jsp

- Ein Anzug bestehend aus vibro-taktilen Impulsgebern. Dieser dient nur zum Erfahren der Schönheit dieser Wahrnehmung.

Eine weitere Form der schnellen Informationsübergabe über haptische Schnittstellen wird durch die so genannten „Tactons“ eingeführt. Diese sind unter haptischen Displays in den Bereich der Ausgabe zuzuordnen. Der Ansatz ist hier nicht kinästhetische, haptische Reize anzusprechen, sondern auf der Ebene der vibro-taktilen Informationsaufnahme zu arbeiten. Die dazu vorhandenen Aktoren bestehen aus kleinen Stiften, die feldartig angeordnet sind. Sehr feine Stifte sind für empfindliche Stellen wie Fingerspitzen vorgesehen, gröbere Stifte dienen der Benutzung in Aktoren, die um den Körper schnallbar sind (in Form von Westen oder Jacken). Diese Geräte sind einfach zu benutzen und zu steuern. Diese Geräte werden dazu benutzt konzeptuell komplexe Informationen an den Benutzer in Form von „Tactons“ mitzuteilen. In der visuellen Welt sind Icons der schnelle Gegenpart zu Text. Hier würden dem Tactons und Braille gegenüber stehen. Dabei stehen dem Reizen des vibro-taktilen Sinns verschiedene Parameter zur Verfügung die verändert werden können um unterschiedliche Informationen darzubringen. Dies wären im Folgenden:

- Frequenz: Der vibro-taktile Sinn kann Frequenzen von 20 - 1000 Hz wahrnehmen. Maximal können so neun verschiedene Frequenzabstufungen unterschieden werden.

- Amplitude: Mit der Intensität kann auch eine Darreichung von verschiedenen Informationen erfolgen. Hierbei ist es möglich bis zu 55 dB über der wahrnehmbaren Schwelle an verschieden empfundenen Intensitäten zu stimulieren. Größere Werte führen aber zur Schmerzwahrnehmung. Untersuchungen haben gezeigt, dass Wertänderungen zwischen 0.4 - 3.2 dB als unterschiedliche Reize wahrgenommen werden.

- Dauer: Reize in unterschiedlicher Länge können Informationen kodieren. Stimulusdauern von 0.1 sec wurden als kurzer Stoß wahrgenommen wobei länger anhaltende Stimuli als sanft fließender Übergang wahrgenommen wurden.

- Rhythmus: Durch Variation der Dauer können verschiedene rhythmische Einheiten generiert werden, die somit der Informationsweitergabe dienen.

- Ort der Stimulation: Als zusätzlicher Informationsträger kann noch der Ort der Stimulation dienen. Dabei spielt noch die unterschiedliche Reizbarkeit unterschiedlicher Körperstellen eine Rolle (z.B. ist der Rücken erheblich unempfindlicher als der Unterarm). Fingerspitzen sind durch ihre hohe Empfindlichkeit prädestiniert für vibrotaktische Displays. Jedoch sind die Finger meist mit anderen Aufgabe beschäftigt. Darum bieten sich für die Stimulation durch Vibrationen der Rücken, der

Bauch und die Oberschenkel an. Versuche haben gezeigt, dass Personen, die an Vibrationsmuster an einem dieser Orte gewöhnt wurden, dieses Muster auch sofort an einem anderen Körperteil wieder erkannt haben.

- Räumlich Zeitliche Ausdehnung der Signale: Muster können sozusagen auf den Körper gemalt werden bzw. sie können auf dem Körper herumwandern.

Mit Hilfe dieser Modulationen der Modalität können verschiedene abstrakte Informationen an den Benutzer über den vibrotaktischen Sinn näher gebracht werden. So könnte ein hochfrequenter Impuls abstrakt für Erstellung eines Objekts dienen. Und ein bestimmtes Impulsmuster könnte einen Ordner repräsentieren. Zusammen kombiniert würden sie „Ordner erstellen“ darstellen. Somit wären vielfältige Kombinationsmuster gegeben, was zu einer Vielfalt an Informationsdarstellungen führen kann.

Tactons können hierarchisch organisiert werden. Somit könnte anhand eines Rhythmus festegestellt werden, wo in der Hierarchie ein Objekt steht.

„Transformationelle“ Tactons hätten verschiedene Eigenschaften. Beispielhaft könnte man eine Datei in einem Betriebssystem darstellen. Der Dateityp wurde über den Rhythmus, die Größe durch die Frequenz und das Erstellungsdatum über die Körperlokation vermittelt. Somit würden die Dateien in einem durchschnittlichen Betriebssystem allesamt einen einzigartigen Reiz auf das vibrotaktische System ausüben.

Tactons können als Erweiterung des normalen Desktopgebrauchs benutzt werden. Dabei haben sie gegenüber auditiven Signalen den Vorteil diskret stattzufinden. Andere Benutzer, z.B. in Großraumbüros, werden nicht gestört. Versuche haben gezeigt, dass der Einsatz von Earcons (Icons für das Ohr) zu weniger Fehlern, verkürzter Zeit zur Erfüllung von Aufgaben und verringertem Arbeitsaufwand geführt haben. Ein Grund Tactons zur Erweiterung des Desktops zu benutzen wäre die Verringerung der visuellen Komplexität von Benutzeroberflächen. Somit stünde eine höhere Konzentrationsfähigkeit in der Orientierung innerhalb solcher Oberflächen zur Verfügung.

Tactons sind auch für den Einsatz mit Braille-Displays vorstellbar um sehbehinderte oder blinde Benutzer zu unterstützen. Damit könnte Information sehr viel effizienter übertragen werden. Ein weiterer Nutzen wäre graphische Informationen nicht-visuell darzustellen. Damit könnten Blinde Fähigkeiten erlangen, die einem normalen Benutzer fast wie selbstverständlich erscheinen.

Abbildung 3.4: Quelle: http://wearcam.org/steve5.jpg

Eine weiteres Anwendungsgebiet finden Tactons in mobilen und tragbaren Geräten. Der Nachteil von mobilen Endgeräten ist, dass die visuellen Displays schnell mit Informationen überflutet werden. Dies

macht ein vernünftiges Design von Interfaces auf so kleinem Raum schwierig. Außerdem ist die visuelle Ablenkung hinderlich im mobilen Kontext, da man somit von der Umgebung abgelenkt ist. Ein Weg diese Schwierigkeiten zu umgehen besteht in dem Einsatz von anderen Darstellungsmodalitäten. Dabei ist Sprache und akustische Interaktion als ein Weg angesehen worden. Dieser Weg bietet viele Vorteile. Nachteile sind jedoch, dass in lauten Umgebungen der Einsatz von auditiven Informationen so gut wie wirkungslos ist. Außerdem können solche Ausgaben andere Personen in der Nähe stören. Bei blinden Menschen werden durch den Einsatz von Kopfhörern wichtige Umgebungsgeräusche blockiert. Eine Lösung für diese Probleme besteht im Einsatz von haptischen Displays. Anstatt dem primitiven Vibrationsalarm für Handys könnten taktile Displays Informationen über den Anrufer liefern. Taktile Displays könnten auch in der Navigation im Menü solch eines Endgerätes dienen. Der Einsatz in mobilen Geräten könnte als anziehbare Geräte geschehen. Ein einfacher Anwendungsfall könnte im Liefern von Richtungsinformationen dienen. Ein Impuls könnte wie ein Kompass eingesetzt werden, der die Richtung angibt in die gegangen werden muss um sein Ziel zu erreichen. Mit Tactons könnten solche Displays auch kontextabhängig die Umgebung beschreiben. Welche Typen von Gebäuden in der Nähe sind (Geschäft, Bank, etc.), was für ein Geschäft in der Nähe ist, in welcher Preislage ein Geschäft sich befindet. Für sehbehinderte Menschen könnten Informationen geliefert werden, wie z.B. wieviele Stufen noch bis zum Erreichen der Treppe vorhanden sind.

Die vorhin erwähnte Weste zum Erzeugen von vibrotaktilen Stimuli gehört in den Bereich der „Wearable Computer“. Ein wearable Computer ist nach Definition „ein Computer der in den persönlichen Raum des Benutzers eingebunden ist, vom Benutzer kontrolliert wird, [...], er ist immer eingeschaltet und einsatzbereit“. Im Extremfall geht dies soweit, dass der Benutzer quasi zum Cyborg wird, also halb Mensch und halb Maschine. Dies kann man am Beispiel von Prof. Steve Mann (http://wearcam.org/mann.htm). Die Frage ist, inwieweit Benutzer diesen doch radikalen Eingriff zum Zwecke der mobilen computerbasierten Unterstützung bei der Umgebungserkundung tolerieren.

Kapitel 4

Accessibility

Abstract Moderne Informations- und Kommunikationstechnologie ist nicht für alle Menschen gleichermaßen zugänglich. Accessibility ist ein wissenschaftliches Feld, das sich mit der Optimierung der Zugänglichkeit von Informationssystemen auseinander setzt. Dabei wird ein besonderes Augenmerk auf die speziellen Bedürfnisse einiger Personengruppen wie z.B. Menschen mit Behinderungen bei der Nutzung von Anwendungen gelegt. Diese Ausarbeitung stellt die Bedürfnisse im Einzelnen, die den behinderten Menschen zur Computernutzung zur Verfügung stehenden Hilfsmittel sowie die gegenwärtigen Standards und Gesetze vor. Außerdem werden im Hinblick auf die Aufgabenstellung der Projektgruppe, in deren Rahmen die Ausarbeitung entstanden ist, die Möglichkeiten zur Gewährleistung der Zugänglichkeit auf mobilen Endgeräten untersucht.

4.1 Einleitung

4.1.1 Motivation

Moderne Informations- und Kommunikationstechnologie nimmt einen stetig steigenden Stellenwert in unserem alltäglichen Leben ein. Im Internet können auf bequeme Art und Weise jederzeit Informationen abgerufen werden, es kann Kontakt zu den verschiedensten Menschen hergestellt werden oder Angebote wie Online-Shopping oder e-Banking können in Anspruch genommen werden. Es gibt aber eine nicht geringe Anzahl an Personengruppen, die aus den verschiedensten Gründen Probleme beim Zugang zu Informationen aus dem Internet haben. Dabei ist das Internet mittlerweile ein wichtiger Teil der gesellschaftlichen Teilhabe, der Gleichstellung und Selbstbestimmung und somit sollte es jedem Besucher möglich sein, dieses grundsätzlich ohne Hilfe nutzen zu können. Insbesondere ältere Menschen oder Menschen mit Behinderungen stoßen aufgrund ihrer Bedürfnisse und speziellen Einschränkungen auf Barrieren, die ihnen den Zugang zum Internet und die Nutzung der Angebote erschweren oder sogar verwehren.

Das Problem liegt dabei sowohl in technischen Schwierigkeiten als auch oftmals an der mangelhaften Informationsaufbereitung und -darstellung. Die Zugänglichkeit von Informationen muss für alle Menschen gleichermaßen gewährleistet sein. Die existierenden Barrieren müssen durch neue Technologien und die adäquate Aufbereitung von Informationen überwunden werden.

Das Ziel dieser Ausarbeitung ist es deswegen, die Projektgruppenteilnehmer für das Thema *Accessibility* zu sensibilisieren und aufzuzeigen, in wie fern *Accessibility* im weiteren Verlauf der Projektgruppe Beachtung finden kann und sollte.

4.1.2 Übersicht

Diese Ausarbeitung entstand im Rahmen der Projektgruppe „Kontext-sensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“ im Wintersemester 2004/2005 im Fachbereich Informatik an der Universität Oldenburg und beschäftigt sich mit dem Thema *Accessibility*. Dazu werde ich im ersten Teil dieser Arbeit die grundlegenden Begriffe *Accessibility* und *Usability* definieren. Danach findet in Kapitel 3 eine Sensibilisierung für das Thema statt, indem die menschliche Seite der Zugänglichkeit beleuchtet wird. In Kapitel 4 wird dann auf die technische Seite der Accessibility eingegangen, samt gesetzlicher Regelungen und anderweitigen Standardisierungen. Kapitel 5 beschreibt, inwiefern sich Accessibility in mobilen Anwendungen beachten lässt. In Kapitel 6 werden dann die wichtigsten Aspekte des Themas zusammengefasst und das Thema in den Kontext der Projektgruppe eingeordnet.

4.2 Begriffsklärung

In diesem Kapitel möchte ich die Begrifflichkeiten klären und darstellen, welche Grundgedanken und Ziele hinter Accessibility und den damit verbundenen Überlegungen stehen. Außerdem soll der Begriff der Usability, welcher im Kontext von Accessibility des Öfteren auftaucht, definiert und deren Zusammenhang dargestellt werden.

Accessibility

Allgemein kann man das englische Wort Accessibility mit Zugänglichkeit bzw. Barrierefreiheit ins Deutsche übersetzen. Doch was bedeutet *Barrierefreiheit*? Unter Barrierefreiheit versteht der Gesetzgeber, entgegen der sonst im Internet vorherrschenden Meinung nicht nur Internetseiten behindertenfreundlich zu gestalten.

> *Barrierefrei sind bauliche und sonstige Anlagen, Verkehrsmittel, technische Gebrauchsgegenstände, Systeme der Informationsverarbeitung, akustische und visuelle Informationsquellen und Kommunikationseinrichtungen sowie andere gestaltete Lebensbereiche, wenn sie für behinderte Menschen in der allgemein üblichen Weise, ohne besondere Erschwernis und grundsätzlich ohne fremde Hilfe zugänglich und nutzbar sind.*[1]

Ich werde im restlichen Teil dieser Arbeit Barrierefreiheit im Bezug auf Informations- und Kommunikationstechnologie folgendermaßen verstehen: Eine Software ist barrierefrei, wenn sie von jedem Benutzer unabhängig von persönlichen Fähigkeiten und Kenntnissen mit jeglicher technischer Ausstattung bedienbar ist.
Eine barrierefreie Webseite ist demnach eine Internetseite, die für jeden Benutzer mit jedem beliebigen Browser und jeder beliebigen technischen Ausstattung im vollen Umfang zugänglich und nutzbar ist.

Usability

Usability kann mit Benutzbarkeit, Benutzer- oder Bedienfreundlichkeit ins Deutsche übersetzt werden und beschreibt, in welchem Maße ein Produkt (z.B. Software oder Webseite) für die vorgesehene Aufgabe geeignet ist.

In der internationalen Norm ISO 9241 (Richtlinien für die ergonomische Gestaltung von Bildschirmarbeitsplätzen) wird Usability wie folgt beschrieben: *„Usability eines Produktes ist das Ausmaß, in*

[1] § 4 Blindengleichstellungsgesetz (BGG)

dem es von einem bestimmten Benutzer verwendet werden kann, um bestimmte Ziele in einem bestimmten Kontext effektiv, effizient und zufriedenstellend zu erreichen“
Je schneller und leichter ein Benutzer den zielgerichteten Gebrauch eines Produktes erlernen und anwenden kann, ohne dabei frustriert zu werden, desto höher ist also die Usability des Produktes.

Wie hängen jetzt aber Usability und Accessibility zusammen?
Usability und Accessibility sind zwei eng miteinander verknüpfte Konzepte. Die erste wichtige Beziehung zwischen ihnen ist, dass eine Verbesserung der Zugänglichkeit für Menschen mit Behinderungen oftmals auch zu einer Verbesserung der Benutzerfreundlichkeit für alle Benutzer führt. Dies ist jedoch nicht zwingend so, weil z.B. der Verzicht auf Dropdown-Menüs im Zuge der Barrierefreiheit durchaus die Benutzerfreundlichkeit beeinträchtigen kann.
Die zweite bedeutende Verbindung zwischen Accessibility und Usability ist, dass man sich bei der Entwicklung von Benutzungsoberflächen, die Menschen mit Behinderungen entgegenkommen auf Performanz, einfache Nutzbarkeit und einfache Erlernbarkeit fokussiert. Es reicht nicht, dass Menschen der Zugang zu Anwendungen ermöglicht wird, sondern die Benutzung muss außerdem einfach, schnell, angenehm und so wenig fehleranfällig wie möglich sein.
([SR03], [Nie03])

4.3 Menschen mit Behinderungen

Die aktuelle Situation von Menschen mit Behinderungen in Deutschland wurde durch einen Paradigmenwechsel in der deutschen Behindertenpolitik geprägt. Ausgehend von Artikel 3 Absatz 3 Grundgesetz („Niemand darf wegen seiner Behinderung benachteiligt werden.“) hat der Gesetzgeber mit dem Sozialgesetzbuch IX[2] (Rehabilitation und Teilhabe behinderter Menschen) und dem Behindertengleichstellungsgesetz (siehe 4.4.3.2) das neue Bild von einem Bürger mit Behinderung in das Gesetz aufgenommen. Der Mensch mit Behinderung wird nicht mehr als Objekt betrachtet, sondern als Bürger mit gleichen Rechten.

So wurden und werden in vielen Bereichen des alltäglichen Lebens Barrieren für Menschen mit Behinderungen abgebaut. So wurden beispielsweise die Eingangsbereiche von vielen öffentlichen Gebäuden mit Rollstuhlrampen ausgestattet und die Einganstüren verbreitert, akustische Signale wurden an Ampeln angebracht, Busse und Bahnen (zumindest teilweise) wurden behindertengerecht gestaltet und es wurden eine Menge von Behindertenparkplätzen geschaffen.

Der nächste logische Schritt ist, auch die Informations- und Kommunikationstechnologie barrierefrei zu gestalten. Bei dieser Problemstellung spielen Menschen mit Behinderungen eine gewaltige Rolle. Um die vorhandenen Barrieren in der Informations- und Kommunikationstechnologie, insbesondere die im Internet, zu reduzieren, muss man sich im Klaren darüber sein, welche besonderen Bedürfnisse Menschen mit Behinderungen bei der Benutzung von Anwendungen haben. Erst wenn man die Bedürfnisse von Behinderten im Bezug auf die Nutzung von Computern verstehen lernt, so kann es gelingen auf diese Bedürfnisse einzugehen und somit existierende Barrieren abzubauen bzw. von vornherein gar nicht erst aufkommen zu lassen.

Aus diesem Grund werde ich in diesem Kapitel auf die aktuelle Situation von Menschen mit Behinderungen in Deutschland anhand von einigen ausgewählten Zahlen eingehen und die eben angedeuteten Bedürfnisse und Probleme von Menschen mit Behinderungen näher beleuchten.

[2] siehe `http://www.behindertenbeauftragter.de/gesetzgebung/sozialgesetzbuchix`

4.3.1 Statistisches

Laut der letzten statistischen Erhebung des Statistischen Bundesamtes Ende 2001, leben in Deutschland rund 6,7 Millionen schwerbehinderte Menschen, was im Vergleich zum Jahresende 1999 einen Anstieg von 1,2% bedeutet. Bezogen auf die gesamte Bevölkerung war in Deutschland jeder zwölfte Einwohner (8,1%) schwerbehindert. Als schwerbehindert gelten in Deutschland Personen, denen von den Versorgungsämtern ein Behinderungsgrad von mindestens 50 zuerkannt wurde.

Älteren Menschen sind besonders betroffen von Behinderungen: So war gut die Hälfte (52%) der schwerbehinderten Menschen 65 Jahre und älter; ein knappes Viertel (23%) gehörte der Altersgruppe zwischen 55 und 65 Jahren an und „nur“ 2,5% der Schwerbehinderten waren Kinder und Jugendliche unter 18 Jahren.

Die Behinderungen wurden überwiegend (in 85% der Fälle) durch eine Krankheit verursacht, 4,7% der Behinderungen waren angeboren und 2,5% waren auf einen Unfall oder eine Berufskrankheit zurückzuführen.

Am häufigsten litten schwerbehinderte Menschen unter Funktionsbeeinträchtigungen der inneren Organe bzw. Organsysteme (27%). Bei 15% der Personen waren Arme und Beine in ihrer Funktion eingeschränkt, bei weiteren 14% Wirbelsäule und Rumpf. Auf geistige oder seelische Behinderungen entfielen zusammen 8%, auf zerebrale Störungen ebenfalls 8%. In 5% der Fälle lag Blindheit oder Sehbehinderung vor.
([Bun03])

4.3.2 Behinderte und Computer

Unter dem Begriff „Behinderung“ ist eine Vielzahl von unterschiedlichen Schädigungen zusammengefasst, welche jeweils die Computernutzung in unterschiedlichem Maße beeinflussen und unterschiedliche Voraussetzungen an Webseiten bzw. Software im Allgemeinen stellen.
Im Folgenden sollen nun auf diese verschiedenen Anforderungen eingegangen werden, die jede einzelne Behinderung mit sich bringt und wie diese z.B. bei der Webseitengestaltung beachtet werden können.

4.3.2.1 visuelle Behinderungen

Visuelle Behinderungen umfassen alle Schädigungen der Sehfähigkeit und werden unterteilt in Sehbehinderung, wie z.B. die Rot/Grün-Sehschwäche, und Blindheit
Aufgrund der Einschränkungen der Sehfähigkeit sind visuell beeinträchtigte Menschen besonders auf unterstützende Hilfsmittel zur Darstellung des Bildschirminhaltes angewiesen. (Siehe Abschnitt 4.4.1)
Die besonderen Anforderungen dieser Personengruppe lassen sich grob in 3 Kategorien einteilen:

Strukturiertheit:
Blinde Menschen können den Bildschirminhalt im Gegensatz zu sehenden Menschen nur linear verarbeiten und benötigen daher klar strukturierten Text anhand von Überschriftenhierarchien, einheitlicher Navigationsstruktur usw., damit dieser von den im Kapitel 4.4.1 beschriebenen Hilfsmitteln angemessen aufbereitet werden kann. Insbesondere eine Trennung von Inhalt und Layout sorgt hierbei für eine bessere Auslesbarkeit einer Webseite durch Screenreader und somit zu einer besseren Lesbarkeit durch den Anwender.

alternative Beschreibungen:
Da Blinde keine Informationen aus Bildern und Grafiken gewinnen können, müssen deren Inhalte durch

Textalternativen beschrieben werden. Dies kann in HTML mittels ALT-Angabe im Zuge des IMG-Tags geschehen oder allgemein in Form von Bildunterschriften, aus denen eine Erklärung bzw. eine Erläuterung des Grafikinhaltes hervorgeht. Auch Videos erfordern eine alternative Inhaltsbeschreibung, sei es in Form von Audiobeschreibungen des aktuellen Videoinhaltes oder mittels textueller Zusammenfassung der Videogeschehnisse, ähnlich der Bildunterschriften.

Navigierbarkeit:
Da blinde Menschen Anwendungen nicht mit der Maus bedienen können, muss eine Navigation innerhalb der Applikation mittels Tastaturkürzel gewährleistet sein, optimalerweise sogar mit selbst definierbaren Tastaturkürzeln. Im Bezug auf eine Webseite bedeutet dies z.B., dass jeder Navigationspunkt einer Webseite mittels Tabulatortaste in einer sinnvollen Reihenfolge angewählt werden kann. Außerdem sollte von der Möglichkeit Gebrauch gemacht werden, allen Verweisen einer Internetseite ein Tastaturkürzel zuzuweisen. Dadurch wird es dem Benutzer ermöglicht, bestimmte Teile der Webseite (z.B. Startseite oder Gästebuch) mittels eines einheitlichen Tastaturkürzels anzuspringen und somit die Navigation in erheblichem Maße zu vereinfachen und zu beschleunigen. Dies kann in HTML mittels des ACCESSKEY-Attributes im Rahmen des A-Tags geschehen.
Im Zuge der Vereinfachung der Navigierbarkeit einer Webseite ist es außerdem zu empfehlen, alle externen (und auch internen) Links mit selbsterklärenden Link-Bezeichnungen auszustatten. Link-Bezeichnungen wie „hier klicken“ liefern für blinde Menschen keinen Informationsgewinn und verschlechtern die Navigierbarkeit enorm.
Im Allgemeinen sollte darauf geachtet, werden jegliche Elemente einer Anwendung mit aussagekräftigen Bezeichnern auszustatten, um die Navigation innerhalb der einzelnen Elemente zu vereinfachen. Sei es nun die Navigation zwischen verschiedenen Frames oder zwischen einzelnen Formularelementen einer Webseite, aussagekräftige Bezeichner führen immer dazu, dass sich der Benutzer leichter zurechtfinden kann.
Ferner ist es sehr ungünstig, wenn die Breite einer in der Anwendung dargestellten Tabelle mehr als 80 Zeichen einnimmt, denn eine Braillezeile (siehe 4.4.1.2) kann nicht mehr als 80 Zeichen gleichzeitig darstellen. Tauchen jedoch sehr breite Tabellen auf, so wird von der Braillezeile ein zusätzlicher Zeilenumbruch erzeugt, was natürlich zu einem erheblichen Maß an Unübersichtlichkeit für den sehgeschädigten Benutzer führt.

Außerdem gibt es noch weitere Anforderungen, die sich keiner der Kategorien zuordnen lassen und im Folgenden kurz erläutert werden sollen.

sonstiges:
Während für blinde Menschen keine spezielle Schriftart, Schriftgröße sowie Vorder- bzw. Hintergrundfarbe von Nöten ist, haben Sehbehinderte Menschen im Bezug auf diese Einstellungen andere Bedürfnisse.
Menschen mit einer Sehschwäche, darunter insbesondere auch ältere Menschen, können kleinere Zeichen sehr häufig nicht mehr lesen und benötigen somit eine skalierbare Schriftgröße. Deshalb sollten z.B. bei der Gestaltung einer Webseite relative anstatt absolute Größenangaben verwendet werden.
Farbenblinde bzw. Menschen mit einer Rot-Grün-Sehschwäche brauchen eine kontrastreiche Kombination von Vorder- und Hintergrundfarben, da sonst keine klare Erkennung der einzelnen Bildbereiche möglich ist. Auf Farbkombinationen wie Rot-Grün Variationen sowie Rot und Grün als Erkennungsmerkmale (z.B. als Bereichstrennung oder anstatt Button-Beschriftungen) sollte aus diesem Grund verzichtet werden. Außerdem sollte die voreingestellte Farbkombination veränderbar sein, sowohl von Schrift als auch von Vorder- bzw. Hintergrund, damit der Benutzer die für ihn angenehmste Farbkombination auswählen kann.

Eine Anforderung, die speziell an Webseiten gestellt werden muss, ist der Verzicht auf einen automatischen Seitenrefresh, welcher z.B. im Zuge von zeitgesteuerten Werbebannern häufig anzufinden ist. Screenreader (siehe 4.4.1.1) werden durch automatischen Seitenrefresh nämlich dazu veranlasst, die Webseite vom Seitenanfang an neu zu bearbeiten. Dies erschwert auf jeden Fall die Arbeit unnötig und macht es im schlechtesten Fall für einen Blinden unmöglich, bis zu dem gewünschten Inhalt durchzudringen.
Zu guter Letzt gilt es im Zuge der Lesbarkeit auf blinkende oder animierte Texte zu verzichten, da diese von Menschen mit einer Sehschwäche als unangenehm zu lesen empfunden werden.
([Men03], [See03], [Nie96], [Hae04], [Win03])

4.3.2.2 auditive Behinderungen

Das Feld der auditiven Behinderungen umfasst Hörschädigungen jeglicher Art, sei es nun eine eingeschränkte Hörfähigkeit oder sogar Taubheit.

Insbesondere die von Geburt an Gehörlosen haben große Probleme mit der Schriftsprache, die für sie im Gegensatz zu ihrer „Muttersprache“, der Gebärdensprache fast eine Fremdsprache darstellt. Aus diesem Grund haben die meisten Gehörlosen, so sie nicht nach Erlernen der natürlichen Sprache ertaubt sind, einen viel kleineren Wortschatz als hörende Personen. Viele Gehörlose haben daher große Schwierigkeiten Anglizismen, lange Texte sowie komplexe Schachtelsätze inhaltlich zu verstehen. Mittels Gebärdensprache hingegen ist jedem Gehörlosen jeder Text inhaltlich verständlich. Optimal wäre daher, auf Webseiten oder in Anwendungen vorhandene Texte zusätzlich als Video in Gebärdensprache anzubieten.

Menschen mit Hörschädigung haben meist Schwierigkeiten mit Audio-Dateien. Für sie müssen Töne, Geräusche oder Stimmen, auf die eine Anwendung nicht verzichten kann, auf einem alternativen Weg dargestellt werden und zwar in einer äquivalenten visuellen Form.
Als naheliegenste Möglichkeit bietet sich da natürlich die Darstellung aller Inhalte von Audio-Dateien als Text an, aber auch die Darstellung der Inhalte mittels Gebärdensprache würde die Bedürfnisse von hörgeschädigten Menschen befriedigen.
Gleiches gilt auch für die Audioinhalte von Videos und Multimedia-Präsentationen. Auch diese müssen textuell aufbereitet werden, sei es in Form von Untertiteln, als ergänzender bzw. erklärender Text zu dem Video oder in Form von Gebärdensprache.
Außerdem müssen Fehlermeldungen, welche nur durch ein Audiosignal erkennbar sind, auch auf eine andere Weise präsentiert werden. Eine textuelle Anzeige der Fehlermeldung, wie zum Beispiel in den verschiedenen Windows-Versionen, ist hierbei vollkommen ausreichend.
([BL03], [See03], [Win03])

4.3.2.3 körperliche Behinderungen

Körperliche Behinderung ist eigentlich nur ein Oberbegriff für sämtliche Erscheinungsformen und Schweregrade körperlicher Beeinträchtigungen. Die häufigsten Erscheinungsformen sind Schädigungen des Zentralen Nervensystems wie Querschnittslähmung, Spastik, Multiple Sklerose oder Parkinson, Schädigungen bzw. Fehlbildungen des Skelettsystems wie Rückgratverkrümmungen bzw. Klumphand sowie Fehlbildungen des Rumpfes wie z.B. fehlende Gliedmaßen.

Während sich für einen Rollstuhlfahrer wenige Probleme beim Umgang mit dem PC ergeben, versucht ein Mensch ohne Hände oder mit Fehlbildungen an den Händen anhand von Hilfsmitteln wie z.B. Mund, Kopf, o.ä. den Computer zu bedienen. Diese Menschen sind darauf angewiesen, sich per Tastatur durch eine Anwendung navigieren zu können.
Für Menschen, die nur eine Hand benutzen können sind Tastaturkürzel, bei denen drei Tasten gleichzeitig

gedrückt werden müssen, oder Shortcuts bestehen aus zwei Tasten (wie Strg+S für Speichern, Strg+F für Suchen) eine Gymnastikübung. Aus diesem Grund sollten sich daher Shortcuts individuell einrichten lassen können.

Menschen mit feinmotorischen Störungen fällt die ruhige und punktgenaue Steuerung der Maus besonders schwer. Aber auch das gleichzeitige Drücken mehrerer Tasten zum Auslösen von Shortcuts bereitet ihnen Schwierigkeiten. Demzufolge benötigt diese Personengruppe ausreichend große Buttons und Links, um sie trotz ungenauer Mausführung anklicken zu können. Aber auch die Navigierbarkeit der Anwendung mit all ihren Bestandteilen durch die Tastatur statt der Maus muss gewährleistet sein. Vor allem eine logische Tabulatorreihenfolge zur Ansteuerung aller Interaktionspunkte ist wichtig.
([Hue04], [See03], [Win03])

4.3.2.4 geistige Behinderungen

Geistige Behinderung ist der Oberbegriff für sämtliche Erscheinungsformen und Schweregrade geistiger Beeinträchtigungen wie Dyslexie (Lese- und Verständnisschwäche), Legasthenie (Lese-Rechtschreib-Schwäche) oder Demenz (Erinnerungsschwäche).

Menschen mit geistiger Behinderung haben Schwierigkeiten mit langen und unüberschaubar aufbereiteten Informationen. Aus diesem Grund sollte die Struktur und die Navigation einer Anwendung so einfach und verständlich wie möglich gestaltet sein. Ferner sollten in allen Texten kurze, prägnante Sätze in einer klaren, einfachen Sprache verwendet werden. Text sollte durch Bilder oder Grafiken ergänzt werden, wenn das Verstehen des Textes dadurch einfacher wird.

Demenzerkrankte benötigen vor allem einen klaren, übersichtlichen und eindeutigen Aufbau der Navigation und aller Inhalte. Insbesondere muss für diese Personengruppe ersichtlich sein, welche Bereiche einer Webseite bereits von ihnen besucht wurden. Aus diesem Grund sollten vor allem bereits besuchte Links in einer anderen Farbe dargestellt werden, als noch nicht besuchte.

Auf Personen mit Rechtschreibschwäche ist vor allem bei der Eingabe von Daten in Suchmasken Rücksicht zu nehmen. Diese sollen mit einer ausreichenden Fehlertoleranz wie beispielsweise einer Rechtschreibprüfung ausgestattet sein. Nur so kann gewährleistet werden, dass sinnvolle und hilfreiche Ergebnisse ausgegeben werden.

Epileptiker reagieren unter Umständen auf heftiges Blinken und Flackern des Bildschirmes mit einem epileptischen Anfall. Folglich muss auf jedes Blinken, Bildschirmflackern und schnelle Animationen verzichtet werden.
([Nie96], [See03], [Win03])

4.4 Accessibility

In diesem Kapitel geht es vor allem um die technische Seite der Zugänglichkeit. Dazu werden zunächst Hilfsmittel vorgestellt, die blinden und sehbehinderten Menschen die Benutzung des Computers ermöglichen, um danach die Fragestellung zu beantworten, ob durch diese Hilfsmittel alle Anwendungen für diese Personengruppe zugänglich sind. Im Anschluss daran werden die gegenwärtigen Standards und deutschen Gesetzte im Bezug auf Zugänglichkeit vorgestellt, bevor schlussendlich Werkzeuge und Verfahren zur Überprüfung der Zugänglichkeit einer Webseite bzw. Anwendung aufgeführt werden.

4.4.1 Hilfsmittel zur Bedienung von Computern und Webseiten

Blinde und Sehbehinderte können den Inhalt eines Computerbilschirms nicht oder nur schwer sehen und außerdem, und in Zusammenhang damit auch keine Maus betätigen. Somit benötigen diese Personengruppen neben den Standardein- und -ausgabegeräten wie Maus, Tastatur und Bildschirm Alternativen für die Ausgabe des Bildschirminhaltes und die Bedienung des Computers.

Sehbehinderten Menschen, die noch über genügend Sehrest verfügen, um die Informationen auf dem Bildschirm erkennen zu können, genügen meist Vergrößerungsprogramme. Diese Programme vergrößern den Bildschirminhalt ausschnittsweise, sorgen für eine Glättung der vergrößerten Schrift und bieten Orientierungshilfen an. Ist der Sehrest jedoch nicht mehr ausreichend, so kommen so genannte *assistive Technologien* zum Einsatz.

Im Folgenden sollen die 2 verbreitetsten dieser Hilfsmittel für blinde und sehbehinderte Menschen vorgestellt werden, denn nur wer über die Funktionsweise dieser Hilfsmittel Bescheid weiß, kann diese der Softwareentwicklung berücksichtigen.

4.4.1.1 Screenreader

Ein Screenreader schlägt eine Brücke zwischen der Computeranwendung und den Ausgabemedien wie Braillezeile (siehe 4.4.1.2) und Sprachausgabe, indem er zunächst den Inhalt des aktuellen Bildschirmfensters ausliest und den Textinhalt, die Bedeutung der grafischen Symbole sowie den Aufbau und die Struktur des aktuellen Bildschirmfensters analysiert und interpretiert. Ist die Phase der Analyse und Interpretation abgeschlossen, so werden die extrahierten Informationen an die Ausgabeeinheiten wie Braillezeile oder Sprachausgabe weitergegeben und können dann an über diese Ausgabemedien entweder in Blindenschrift oder in synthetischer Sprache an den Benutzer ausgegeben werden.

Ein Screenreader verwendet dabei einen eigenen Zeiger, ähnlich dem Mauszeiger jedoch unabhängig von diesem, der nicht sichtbar ist und zeigt, wo sich der Screenreader gerade befindet. Mittels der Pfeiltasten wird bei den meisten Screenreadern die Navigation innerhalb von einer Anwendung und dabei insbesondere das zeilenweise Abarbeiten der aufbereiteten Informationen ermöglicht.

Mittlerweile gibt es Screenreader für die gängigsten Betriebssysteme. Unter Windows ist vor Allem *JAWS*[3] (Job Access With Speech) weit verbreitet, der jedoch je nach Ausführung zwischen 900 und 1100$ kostet. Auch kostenpflichtige Produkte wie *VIRGO*[4] oder *BLINDOWS*[5] werden von blinden oder sehbehinderten Menschen unter Windows häufiger genutzt.
Für die verschiedenen Macintosh Betriebssysteme stehen *Voice over*[6] für Mac OS X oder *outSPOKEN*[7] für ältere Macintosh Betriebssysteme zur Verfügung.
Unter Linux ist vor allem *Suse Blinux*[8] zu erwähnen, welches seit SUSE-LINUX 7.0 fester Bestandteil der Distribution ist und auch sonst kostenlos heruntergeladen werden kann. Außerdem gibt es unter Linux auch noch kostenpflichtige Screenreader wie z.B. *UXDOTS*[9], dessen Preis in der Personal Edition 1000 Euro beträgt.

Ein wesentlicher Unterschied zwischen den verschiedensten Screenreaderm besteht darin, dass manche Screenreader an eine bestimmte Braillezeile gebunden sind, andere dagegen eine größere Auswahl zulassen. Jedes Produkt verfolgt seine eigene Bedienstrategie, wobei die verschiedenen Ansätze sich langsam

[3] http://www.freedomscientific.com/fs_products/software_jaws.asp
[4] http://www.virgo4.de/
[5] http://www.audiodata.de/de/blindows/index.php
[6] http://www.apple.com/accessibility/spokeninterface/
[7] http://www.synapseadaptive.com/alva/outspoken/outspoken_for_mac.htm
[8] http://www.suse.de/de/private/support/online_help/howto/blinux/
[9] http://www.c-lab.de/insb/uxdots.htm

aufeinanderzu entwickeln. Deutliche Qualitätsunterschiede bestehen im Zusammenspiel von Sprachausgabe und Braillezeile, das einige Screenreader in komplementärer Weise beherrschen, während andere eines der beiden Ausgabemedien bevorzugen.
Demnach muss bei der Auswahl eines Sceenreaders genau auf die zu bewältigende Aufgabe und die dafür eingesetzte Person Rücksicht genommen werden.
([Inc04a], [Hae04], [Win03])

4.4.1.2 Braillezeilen

Die Ausgabe bzw. Übersetzung von Bildschirminformationen in Blindenschrift erfolgt über eine Braillezeile, die meist in Form einer Leiste vor der Tastatur des PC liegt. Auf einer Braillezeile sind je nach Ausführung 20 bis 80 Zeichen, so genannte Module, angeordnet. Eine feine Mechanik steuert kleine Stifte innerhalb dieser Module, die sich heben oder senken und somit die einzelnen Buchstaben des Bildschirminhalts in Blindenschrift darstellen. Durch diese maximal 80 darstellbaren Zeichen kann natürlich maximal auch nur eine entsprechend lange Textzeile angezeigt werden. Dadurch erfolgt somit nur die Darstellung eines Bildschirmausschnittes, bestehend aus maximal 80 Zeichen einer Bildschirmzeile.
Um innerhalb des Bildschirminhaltes navigieren zu können, besitzt die Braillezeile verschiedene Steuertasten zur Verschiebung des Ausschnittes in alle vier Himmelsrichtungen.

Außerdem existiert bei der neusten Generation der Braillezeilen das so genannte Cursorrouting. Hierbei befindet sich ober- oder unterhalb jedes Braillemoduls ein kleiner Knopf (siehe Abbildung 4.1).

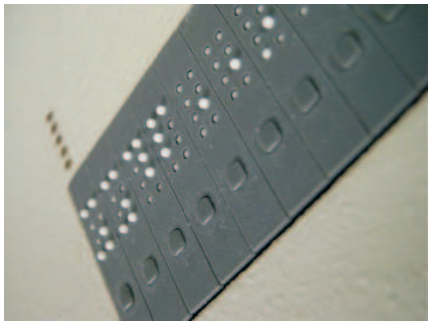


Abbildung 4.1: Ausschnitt einer Braillezeile mit Cursorrouting

Durch Drücken dieses Knopfes wird der Cursor auf dem Bildschirm an die Stelle gesetzt, wo sich das dem Knopf zugeordnete übersetzte Zeichen befindet. Dieses Verfahren dient im Besonderen der Emulation der Maus. So kann man mittels dieser Knöpfe den Mauszeiger navigieren und einfache bzw. doppelte Klicks auslösen.

Eine Braillezeile kann mittels paralleler oder serieller Schnittstelle sowie über die USB Schnittstelle an den Computer angeschlossen werden. Die Stromversorgung einer Braillezeile kann, je nach Typ der Braillezeile, entweder über ein externes Steckernetzteil oder über das Netzteil des verwendeten Computers erfolgen.
Durch transportable, akkubetribene Braillezeilen wird auch die Nutzung von Laptops für blinde und sehbehinderte Menschen ermöglicht.
([Hae04], [fG04])

4.4.2 Zugänglichkeit von Anwendungen

Nun stellt sich natürlich die Frage, ob alle Anwendungen für Menschen mit Behinderungen zugänglich sind. Dies wäre jedoch vermessen zu behaupten, da es schon eine Reihe von Anwendungen gibt, die für nichtbehinderte Menschen aufgrund der verschiedensten Tatsachen (z.B. unübersichtliche Menüführung usw.) unzugänglich sind.
Auffällig ist jedoch die Tatsache, dass die Zugänglichkeit von Anwendungen immer in engem Zusammenhang mit den eingesetzten Hilfsmitteln steht. Während z.B. einige Screenreader mit bestimmten Anwendungen sehr gut kooperieren können, kann die Bedienung der selben Anwendung mit anderen

Screenreadern nahezu unmöglich sein. Das macht es somit schwer möglich eine konkrete Aussage zu treffen, ob eine bestimmte Anwendung für Menschen mit Behinderungen, insbesondere für Blinde, zugänglich ist. Ich möchte mich daher bei der Beantwortung der Frage nach Zugänglichkeit einer Windowsanwendung an den Erfahrungen von Matthias Hänel [Hae04] und einem Testbericht des „Informationspool Computerhilfmittel für Blinde und Sehbehinderte“ [Inc04b] orientieren.
Matthias Hänel beschreibt auf seiner Homepage einige Windows-Anwendungsprogramme, die von Blinden aufgrund der vorhandenen Hilfsmittel relativ problemlos genutzt werden können. Zu diesen Programmen zählen hauptsächlich die Standard-Microsoft Produkte, wir *Word*, *Excel*, *Internet Explorer* oder *Outlook*. Jedoch muss man auch bei diesen Anwendungen noch einige Abstufungen vornehmen. Während in *Word* die Textverarbeitung, abgesehen von einigen kleineren Schwierigkeiten beim Bearbeiten von Dokumenten, problemlos ist, gibt es bei *Excel* noch ein paar größere Probleme (z.B. bei der Formularbearbeitung).

Generell kann man sagen, dass die grundlegenden Standardanwendungen unter Windows im Großen und Ganzen recht problemlos zugänglich sind. Dadurch wurde eine solide Basis geschaffen, die es aber zukünftig zu verbreitern gilt. Einen guten Ansatz dafür bietet die von Microsoft zur Verfügung gestellte Microsoft Active Accessibility (MSAA)[10] Schnittstelle, die es für zukünftige Anwendungen zu implementieren gilt, um ein optimales Zusammenspiel mit der Hilfsmitteltechnologie zu gewährleisten und somit ein höchstes Maß an Zugänglichkeit erreichen zu können.

Unter Linux/Unix sind dank vorhandener Screenreadertechnologie alle textbasierten Anwendungen, wie z.B. Pine oder emacs auch für Menschen mit Sehbehinderungen zugänglich. Durch den Screenreader Gnopernicus[11] wird zudem der Zugriff auf alle Gnome-Programmen, Mozilla und Open Office unterstützt [fbusS04]. Auch unter Linux/Unix gibt es Bestrebungen, die Zugänglichkeit von grafischen Oberflächen zu verbessern. Zu diesem Zweck wurden u.a. das Gnome Accessibility Project[12] und das KDE Accessibility Project[13] ins Leben gerufen. Es bleibt jedoch abzuwarten, in wieweit diese Projekte Früchte tragen werden.
([Hae04], [Inc04b], [fbusS04], [Hel02])

4.4.3 Gesetze und Standards

Die USA haben bereits im Jahr 1998 im „Rehabilitation Act“ und der darin enthaltenen „Section 508“ alle Bundesbehörden der USA gesetzlich dazu verpflichtet, ihre Angebote basierend auf Informations- und Kommunikationstechnologie für Menschen mit Behinderungen zugänglich zu machen.
Die Europäische Gemeinschaft hat mit dem von allen Mitgliederstaaten ratifizierten e-Europe Aktionsplan festgeschrieben, dass alle Web-Auftritte der öffentlichen Hand der Mitgliederstaaten und der europäischen Institutionen bis zum Jahresende 2005 barrierefrei gestaltet sein müssen, um sicherzustellen, dass behinderte Menschen Zugang zu den Informationen bekommen und die Vorteile des Potentials des E-Governments voll nutzen können.

In diesem Abschnitt sollen zunächst die Web Content Accessibility Guidelines als Grundlage für alle Gesetze, die sich mit der Zugänglichkeit von Informations- und Kommunikationstechnologie beschäftigen, dargestellt werden und dann auf die rechtlichen Umsetzung dieser Kriterien in Deutschland eingegangen werden. ([fIA04], [wik04])

[10] MSAA ist eine Schnittstelle von Microsoft, über die Hilfsmittelprogramme auf Informationen von Anwendungen zugreifen können.
[11] `http://www.baum.ro/gnopernicus.html`
[12] `http://developer.gnome.org/projects/gap/`
[13] `http://accessibility.kde.org/`

4.4.3.1 Web Content Accessibility Guidelines (WCAG)

Die Web Accessibility Initiative (WAI), eine Arbeitsgruppe des World Wide Web Consortiums W3C hat am 5.Mai 1999 Spezifikationen zur barrierefreien Webgestaltung („Web Content Accessibility Guidelines 1.0 “) veröffentlicht, um die Zugänglichkeit Web-Inhalten zu fördern.
Dieser Spezifikation besteht aus 14 Richtlinien, von denen jede einzelne mit einem oder mehreren von insgesamt 66 Checkpunkten in Zusammenhang steht. Jedem Checkpunkt wurde wiederum von der WAI eine von 3 Prioritätsstufen zugeordnet, je nach Abhängigkeit seines Einflusses auf die Zugänglichkeit.
Prioritätsstufe 1 enthält alle Checkpunkte, die von Entwicklern von Web-Inhalten erfüllt werden *müssen*, da anderenfalls eine oder mehrere Gruppen von der Benutzung dieser Web-Inhaltes ausgeschlossen wären. Inhalte, die diese Prioritätsstufe erfüllen, besitzen die Konformitätsstufe A.
Die Prioritätsstufe 2 beinhaltet alle „*Soll*-Anforderungen“, d.h. Entwickler von Web-Inhalten sollten diese Checkpunkte erfüllen, da sonst eine oder mehrere Gruppen Schwierigkeiten beim Zugriff auf diese Web-Inhalt haben. Inhalte, die diese Prioritätsstufe und die der Stufe 1 erfüllen, besitzen die Konformitätsstufe AA.
In der Prioritätsstufe 3 sind alle Checkpunkte zusammengefasst, die von Entwicklern von Web-Inhalten erfüllt werden *können*, um einer oder mehreren Gruppen den Zugriff auf diese Inhalte zu erleichtern. Inhalte, die alle Prioritätsstufe erfüllen, besitzen die Konformitätsstufe AAA.

Die oben beschriebenen Richtlinien behandeln im Groben folgende Themengebiete:[14]

1. Stellen Sie äquivalente Alternativen für Audio- und visuellen Inhalt bereit
2. Verlassen Sie sich nicht auf Farbe allein
3. Verwenden Sie Markup und Stylesheets und tun Sie dies auf korrekte Weise
4. Verdeutlichen Sie die Verwendung natürlicher Sprache
5. Erstellen Sie Tabellen, die sich flexibel transformieren lassen
6. Sorgen Sie für Rückwärtskompatibilität ihrer Seiten im Bezug auf die eingesetzte Technologie
7. Sorgen Sie für eine Kontrolle des Benutzers über zeitgesteuerte Änderungen des Inhalts
8. Sorgen Sie für direkte Zugänglichkeit eingebetteter Benutzerschnittstellen
9. Wählen Sie ein geräteunabhängiges Design
10. Verwenden Sie Interim-Zugänglichkeitslösungen, damit assistive Technologien und ältere Browser korrekt funktionieren
11. Verwenden Sie W3C-Technologien und -Richtlinien
12. Stellen Sie Informationen und Hilfen zum Kontext und zur Orientierung bereit
13. Stellen Sie klare Navigationsmechanismen bereit
14. Sorgen Sie dafür, dass Dokumente klar und einfach gehalten sind

[14]deutsche Übersetzung der WCAG erreichbar unter `http://www.w3c.de/Trans/WAI/webinhalt.html`

An dieser Stelle wurde bewusst darauf verzichtet, die WCAG in ihrer ausführlichen Form wiederzugeben, da diese sonst den Rahmen dieser Arbeit sprengen würden. Dem interessierten Leser sei die offizielle Seite der WCAG 1.0 [15] oder deren deutsche Übersetzung[16] nahegelegt, um die Richtlinien in all ihren Details zu verfolgen.

Da seit der Veröffentlichung der WCAG in der Version 1.0 im Jahr 1999 schon einige Zeit vergangen ist, entsprechen diese in einigen Punkten nicht mehr dem Stand der Technik. Die Arbeitsgruppe der WCAG hat diese Problematik erkannt und arbeitet seit geraumer Zeit an einem Nachfolger. Die WCAG 2.0[17] liegen im Moment nur als Arbeitsversion (Working Draft) vor, ihre Bearbeitung ist also noch nicht so weit fortgeschritten, als dass sie als Empfehlung veröffentlicht werden könnten.
Im Gegensatz zu den WCAG 1.0, die einen Ansatz zur korrekten Verwendung von HTML und CSS verfolgen, wird in der WCAG 2.0 das Ziel angestrebt, die Prinzipien und Kriterien unabhängig von heutigen und zukünftigen Technologien zu formulieren. Dabei werden die WCAG 1.0 jedoch nicht durch die WCAG 2.0 ersetzt, sondern fortgeschrieben und ergänzt. Die 14 Richtlinien der WCAG 1.0 werden in der neuen Version nur noch den 4 folgenden Hauptrichtlinien zugeordnet.[18]

- **Wahrnehmbarkeit:** Inhalte müssen in einer für alle Nutzer wahrnehmbaren Form angeboten werden
- **Bedienbarkeit:** Alle Bedienelemente im Inhalt müssen von allen Anwendern benutzt werden können
- **Verständlichkeit:** Inhalte und Bedienelemente müssen so einfach verständlich, wie möglich sein
- **Robustheit der Technik:** Inhalte müssen ausreichend robust sein, damit sie mit heutigen und zukünftigen Technologien funktionieren

Eine Verabschiedung des Dokuments durch die WAI ist im Moment leider noch nicht absehbar.

4.4.3.2 Gesetz zur Gleichstellung behinderter Menschen (BGG)

Am 1.Mai 2002 trat das Gesetz zur Gleichstellung behinderter Menschen (Behindertengleichstellungsgesetz - BGG) in Deutschland in Kraft. Ziel des Bundesgesetzes ist es, *"die Benachteiligung von behinderten Menschen zu beseitigen und zu verhindern sowie die gleichberechtigte Teilhabe von behinderten Menschen am Leben in der Gesellschaft zu gewährleisten und ihnen eine selbstbestimmte Lebensführung zu ermöglichen."* (§1, Gesetzesziel [BGG02]).
Dabei geht es nicht nur um die Beseitigung von Barrieren für Rollstuhlfahrer/Innen sowie für gehbehinderte Menschen sondern auch um barrierefreie Kommunikation blinder, seh- oder hörbehinderter Menschen, um die Teilnahme blinder und sehbehinderter Menschen an Wahlen sowie um Nutzungsmöglichkeiten elektronischer Medien. Die Herstellung barrierefrei gestalteter Lebensbereiche stellt demnach das Kernstück des Gesetzentwurfes dar.

Der im Rahmen der Informatik interessanteste Gesetzesteil[19] ist §11 über barrierefreie Informationstechnik. Hierin wird festgelegt, dass *„Träger öffentlicher Gewalt[20] [...] ihre Internetauftritte und -angebote*

[15] `http://www.w3.org/TR/WAI-WEBCONTENT/`
[16] `http://www.w3c.de/Trans/WAI/webinhalt.html`
[17] `http://www.w3.org/TR/WCAG20/`
[18] Eine detailliertere Zuordnung der WCAG 1.0 Checkpunkte zum WCAG 2.0 Working Draft findet sich unter `http://www.w3.org/WAI/GL/2004/07/26-mapping.html`
[19] Anm.: Dies ist in keinstem Fall abwertend gegenüber den anderen Paragraphen des BGG gemeint, nur behandelt diese Ausarbeitung Accessibility insbesondere im Zusammenhang mit Informations- und Kommunikationstechnologie.
[20] Diese Definition umfasst u.a. alle Bundesministerien, die Bundesanstalt für Arbeit, die Bundeszentrale für gesundheitliche Aufklärung sowie das Umweltbundesamt

sowie die von ihnen zur Verfügung gestellten grafischen Programmoberflächen, die mit Mitteln der Informationstechnik[21] *dargestellt werden schrittweise technisch so (gestalten), dass sie von behinderten Menschen grundsätzlich uneingeschränkt genutzt werden können.*[BGG02]
Frei von dieser gesetzlichen Vorgabe sind demnach alle gewerblichen Anbieter von Internetseiten und grafischen Programmoberflächen. Der Gesetzgeber empfiehlt jedoch eine freiwillige Umsetzung durch die Unternehmen.

Im BGG wird jedoch nicht geregelt, welche Maßnahmen zum Umsetzen dieses Paragraphen des Gleichstellungsgesetzes erforderlich sind. Dies machte eine Rechtsverordnung des Bundes erforderlich, welche diese Maßnahmen im Detail regelt.

4.4.3.3 Barrierefreie Informationstechnik-Verordnung (BITV)

Die zum Bundesgleichstellungsgesetz (BGG) gehörige Barrierefreie Informationstechnik-Verordnung (BITV) trat am 17. Juli 2002 in Kraft. Sie regelt die praktische und die zeitliche Umsetzung des Paragraphen §11 im Bezug auf Behörden der Bundesverwaltung und bezieht sich auf deren Internetauftritte und -angebote, Intranetauftritte und -angebote, die öffentlich zugänglich sind, sowie deren „mittels Informationstechnik realisierte graphische Programmoberflächen“, die öffentlich zugänglich sind.

Die Anlage 1[22] der Rechtsverordnung enthält keine Vorgaben zur grundlegenden Technik, die für die Bereitstellung von elektronischen Inhalten und Informationen verwendet wird (Server, Router, Netzwerkarchitekturen und Protokolle, Betriebssysteme usw.), sondern listet Anforderungen auf, die sich an den Richtlinien der WCAG 1.0 (siehe 4.4.3.1) orientieren.
Diese insgesamt 28 Anforderungen mit über 60 zu erfüllenden Bedingungen sind in 2 Prioritätsstufen unterteilt, wobei die Prioritäten 1 und 2 der WCAG zur Prioritätsstufe I zusammengefasst wurden; die Prioritässtufe II im BITV entspricht der Priorität 3 der WCAG. Standards der Prioritätsstufe I sind zwingend einzuhalten, um wesentliche Barrieren zu beseitigen, wohingegen solche der Stufe II nur auf zentralen Navigations- und Einstiegsangeboten, wie etwa Portalen, Sitemaps, Übersichtsseiten oder Suchseiten zwingend erforderlich sind.
Öffentliche Angebote, die die unter Priorität I genannten Anforderungen und Bedingungen erfüllen, würden bei den WCAG 1.0 des W3C die Konformität AA erreichen. Eine Konformitätsstufe von AAA würden öffentliche Angebote erreichen, die sowohl die Prioritäten I als auch II erfüllen.

Für die Anpassung bestehender Angebote ist eine Übergangsfrist bis zum 31. Dezember 2005 vorgesehen, solange diese Angebote sich nicht speziell an Menschen mit Behinderungen richten[23]. Solche Angebote mussten bis zum 31. Dezember 2003 an die festgelegten Standards angepasst werden.
Für Angebote im Internet, die seit dem Zeitpunkt des Inkrafttretens der Verordnung ganz oder im Wesentlichen neu gestaltet werden, sind die vorgeschriebenen Standards sofort einzuhalten. Als Veränderung oder Anpassung wesentlicher Bestandteile gilt jede Änderung, die über rein redaktionelle Änderungen hinausgeht.

4.4.3.4 Kriterien zur Gestaltung zugänglicher Systeme

Im Gegensatz zu den Standards zur Gestaltung von barrierefreien Webinhalten gibt es zur Gestaltung zugänglicher Anwendungen zur Zeit noch keine offiziellen Standards. Zwar hat die WAI auch Richtlinien

[21] Unter „mittels Informationstechnik realisierte grafische Programmoberflächen“ sind insbesondere CD-ROMs, DVDs oder vergleichbare Medien zu verstehen. [BGG02]
[22] online erreichbar unter `http://www.bmgs.bund.de/download/gesetze/behinderung/anlage\_1\_bitv.htm`
[23] z.B. das Angebot des Beauftragten der Bundesregierung für die Belange behinderter Menschen, entsprechende Seiten des Bundesministeriums für Arbeit und Sozialordnung oder der Bundesanstalt für Arbeit

wie die User Agent[24] Accessibility Guidelines (UAAG)[25] oder Authoring Tool[26] Accessibility Guidelines (AATG)[27] veröffentlicht, doch zielen diese alle darauf hin, Web-Inhalt für alle Menschen zugänglich zu machen. Mehrere Unternehmen, wie z.B. IBM oder Microsoft bieten jedoch auf ihren Webseiten Checklisten zur Erstellung von zugänglicher Software an.
Laut Microsoft[28] besteht die Idee eines zugänglichen Softwareentwurfs aus den 5 Grundprinzipien

- *Flexibilität der Benutzeroberfläche:* Bereitstellung einer anpassbaren Benutzeroberfläche (z.B. einstellbare Schriftgröße, Farbe, Übernahme der Systemeinstellungen)
- *Flexibilität der Eingabemethoden:* Die Anwendung muss mit verschiedenen Eingabegeräten gleichermaßen bedienbar sein; Shortcuts sollten individuell anpassbar sein
- *Flexibilität der Ausgabemethoden:* Bereitstellung von verschiedenen Ausgabekombinationen aus Audio, Optik, Text und Grafik, die individuell zusammenstellbar sind
- *Konsistenz:* Die Anwendung muss in einer konsistenten und vorhersehbaren Weise mit anderen Anwendungen und Systemstandards (z.B. MSAA) zusammenarbeiten
- *Kompatibilität mit Eingabehilfen:* Erstellung der Anwendung unter Verwendung von standardmäßigen und bekannten Benutzeroberflächen (z.B. dem bei Microsoft üblichen Menü), die mit Eingabehilfen kompatibel sind

Die anderen Empfehlungen zum zugänglichen Softwareentwufs ähneln der von Microsoft, lediglich der Detaillierungsgrad der Beschreibungen unterscheidet sich. Dem interessierten Leser sind hier die Ausführungen von IBM[29] und des „Deutschen Vereins der Blinden und Sehbehinderten in Studium und Beruf“[30] zu empfehlen.
Da die Gestaltung zugänglicher Software noch nirgends gesetzlich geregelt ist, sind Menschen mit Behinderungen an dieser Stelle auf den guten Willen der Entwickler angewiesen. Vor allem die schon beschriebenen Großunternehmen gehen hierbei mit gutem Beispiel voran. Es bleibt jetzt nur noch zu wünschen, dass möglichst viele Entwickler diesen Beispielen folgen.

4.4.4 Werkzeuge und Verfahren zur Überprüfung von Accessibility

In der Praxis gibt es eine ganze Reihe von Werkzeugen, um die Einhaltung der Verschiedensten Standards im Bezug auf Accessibility überprüfen. Einer der berühmtesten Vertreter dieser Kategorie ist *Bobby*[31]. *Bobby* ermöglicht die Überprüfung einer Webseite wahlweise auf Einhaltung der WCAG 1.0 oder der U.S. Section 508 Guidelines und verhängt, je nach Abschneiden einer Webseite, in Anlehnung an die Konformitätsstufen der WCAG die Prädikate „Bobby A“, „Bobby AA“„Bobby AAA approved“ oder „not approved“.
Ein kostenfreies Programm in deutscher Sprache, welches auch die Überprüfung von Offlineinhalten erlaubt ist *A-Prompt*[32]. Dieses Programm kann nicht online ausgeführt, sondern muss auf einem Windows-Rechner installiert werden. Dafür bietet es nicht nur die Überprüfung von Webseiten anhand der BITV,

[24] Ein Benutzeragent ist eine Software zum Zugriff auf Internetinhalte
[25] http://www.w3.org/TR/UAAG10/
[26] Ein Autorentool ist eine Software zur automatischen Generierung von Internetinhalten
[27] http://www.w3.org/TR/ATAG20/
[28] http://msdn.microsoft.com/library/deu/vsent7/html/vxconbasicprinciplesofaccessibledesign.asp
[29] http://www-3.ibm.com/able/guidelines/index.html
[30] http://www.dvbs-online.de/download/fit/web.htm
[31] http://www.cast.org/bobby
[32] http://wob11.de/publikationen/aprompt/programm.html

WCAG oder Section 508, sondern zusätzlich noch Funktionen zur schrittweisen Korrektur der Webseite. Es gibt noch eine Reihe von weiteren Werkzeugen zur Überprüfung der verschiedenen Aspekte der Zugänglichkeit von Webseiten, so z.B. Programme um zu testen, wie Webseiten von Menschen mit Sehschädigungen (z.B. Rot-Grün-Schwäche) wahrgenommen werden. Eine Umfassende Sammlung von Links zu diesem Thema findet sich unter [33].

Beim Einsatz dieser Werkzeuge ist jedoch zu beachten, dass die Überprüfung von Webseiten auf Barrierefreiheit lediglich auf syntaktischer Ebene geschehen kann. Für die Software ist es z.B. nicht feststellbar, wenn bestimmte Elemente einer Sprache missbräuchlich eingesetzt werden. Die Barrierefreiheit auf der Ebene der Wahrnehmung durch den User, wie z.B. eingängige Navigation und verständliche Texte ist nur durch Tests mit „echten“ Benutzern überprüfbar. Auch für diese Überprüfung auf Zugänglichkeit, die über die syntaktische Analyse hinausgeht hat wie WAI im Anhang zu den WCAG 1.0 ein Verfahren vorgestellt, dass an dieser Stelle kurz aufführen möchte[34].

- Verwendung eines automatisierten Zugänglichkeits-Tools
- Syntaxvalidierung (z.B. HTML, XML usw.)
- Stylesheetvalidierung (z.B. CSS)
- Verwendung eines Textbrowsers oder Emulators
- Verwendung mehrerer Grafikbrowser: mit aktiviertem Ton und Grafik, ohne Grafiken, ohne Ton, ohne Maus, mit deaktivierten Frames, Scripts, Stylesheets und Applets.
- Verwendung verschiedener Browsergenerationen
- Verwendung eines sprachgenerierenden Browsers, eines Screenreader, von Vergrößerungs-Software, eines kleines Display usw.
- Rechtschreib- und Grammatikprüfung des Inhaltes
- Überprüfung des Dokuments auf Klarheit und Einfachheit
- Fordern Sie Behinderte auf, Dokumente zu überprüfen

Um die Zugänglichkeit von AWT- oder Swing-basierten Java-Anwendungen zu testen, stellt Sun den Java Accessibility Helper Early Access v0.6[35] bereit. Dieser generiert, angewendet auf eine AWT- bzw. Swing-basierte Anwendung, einen Bericht, der auf eine Liste von Problemen bzw. möglichen Problemen mit der Anwendung hinweist. (z.B. wird überprüft, ob alle Eingabefelder mit der Tastatur erreichbar sind)

Für die allgemeine Überprüfung der Zugänglichkeit von Anwendungen gibt es meines Wissens im Moment noch keine Werkzeuge. Die Firma IBM hat jedoch eine Checkliste entwickelt, anhand derer man Software Zugänglichkeit überprüfen kann[36]. Diese Checkliste ist ähnlich der WCAG 1.0 aufgebaut und teilt die Überprüfung auf Zugänglichkeit in 7 Bereiche, welche wiederum auch denen der WCAG 1.0 ähneln. So wird zum Beispiel der alternative Zugriff auf alle Bedienelemente per Tastatur, sowie die Bereitstellung alternativer Inhalte zu Grafiken, Sounds und Videos gefordert, um die Zugänglichkeit einer Anwendung zu erreichen. Dabei ist es natürlich am sinnvollsten, wenn die einzelnen Punkte dieser Checkliste nicht erst nach der Entwicklung der Software überprüft werden, sondern die Zugänglichkeit schon in der Planungsphase eine gewichtige Rolle spielt und anhand dieser Checkliste prozessbegleitend überprüft wird.

[33] `http://www.w3.org/WAI/ER/existingtools.html`
[34] siehe `http://www.w3c.de/Trans/WAI/webinhalt.html`
[35] `http://java.sun.com/developer/earlyAccess/jaccesshelper/`
[36] Das Original befindet sich unter `http://www-3.ibm.com/able/guidelines/software/accesssoftware.html`, die deutsche Übersetzung ist unter `http://wob11.de/publikationen/ibmguidelines/index.html` zu finden

4.5 Accessibility und mobile Anwendungen

Bevor ich auf die Frage eingehen kann, in wieweit Accessibility Beachtung in mobilen Anwendungen findet, möchte ich erstmal den Begriff der mobilen Anwendung definieren. Eine mobile Anwendung ist eine Anwendung, die auf einem mobilen Endgerät wie z.B. PDA oder Mobiltelefon läuft.

Zur Zeit gibt es weder Standards noch Richtlinien im Bezug auf die Entwicklung von zugänglichen mobilen Anwendungen. Außerdem besteht auch keine Unterstützung von Seiten der verfügbaren Betriebssysteme, wie dies z.B. unter Windows mit der MSAA-Schnittstelle zu finden ist. Teile der Konzeptionen von zugänglichen Anwendungen lassen sich jedoch auch auf die mobile Ebene übertragen.
So sollte auch hier die Gestaltungsprinzipien im Hinblick auf Farbgestaltung sowie Schriftgröße individuell anpassbar sein, um Menschen mit Behinderungen die Benutzung von mobilen Anwendungen zu vereinfachen. Im Hinblick auf geistige und auditive Behinderungen sollten die Inhalte so einfach und strukturiert wie möglich dargeboten werden. Außerdem sollte auch hier von der Möglichkeit Gebrauch gemacht werden, Inhalte auf verschiedenen Sinneskanälen zu transportieren.

Vor gar nicht allzu langer Zeit waren sowohl Handys as auch PDAs für Menschen mit Sehbehinderungen nicht zugänglich. Doch im Zuge der immer weiteren Verbreitung mobiler Endgeräte wurden auch in diesem Bereich einige Lösungen präsentiert, die Basisfunktionen der mobilen Endgeräte auch für Menschen mit Sehbehinderungen zugänglich zu machen. Diese Lösungen lassen sich in 2 Kategorien klassifizieren:

- **Hardwarelösungen:** Verschiedenste Hersteller haben Organizer bzw. PDAs eigens für sehbehinderte Menschen hergestellt. Diese Geräte sind anstatt mit einer Tastatur direkt mit einer Braillezeile ausgestattet und verzichten in den meisten Fällen sogar auf ein Display. Mit Hilfe der Datenausgabe in Brailleschrift können blinde Menschen z.B. auf das Telefonbuch zugreifen sowie Kurzmitteilungen und E-Mails schreiben bzw. lesen. Außerdem besteht bei diesen Geräten die Möglichkeit Texte zusätzlich von einer integrierten Sprachausgabe vorgelesen zu bekommen. Auch die Nutzung des Telefonbuchs mit Adressverwaltung sowie eines Kalenders mit Terminverwaltung stellt auf diesen Geräten kein Problem mehr dar. Als Vertreter dieser Gattung wäre z.B. das ALVA MPO[37] zu nennen.

- **Softwarelösungen:** Neben den Hardwarelösungen haben einige Hersteller auch Screenreader für Mobiltelefone entwickelt. Mit Hilfe dieser Screenreader wird es blinden Menschen ermöglicht, die Basisfunktionalitäten eines Handys zu verwenden. Dazu wird der aktuelle Inhalt des Handydisplays von der integrierten Sprachausgabe vorgelesen. Prominenteste Vertreter der Softwarelösungen sind Mobile Accessibility 2.0[38], TALKS[39] oder VIAS[40]

4.6 Zusammenfassung und Einordnung in den Kontext der Projektgruppe

Das Feld der Accessibility beschäftigt sich mit der Zugänglichkeit von Anwendungen. Wie von mir dargestellt haben Menschen mit Behinderungen besondere Bedürfnisse bei der Nutzung von Software. Um für blinde und sehbehinderte Menschen überhaupt nutzbar zu sein, muss diese die aktuelle Hilfsmitteltechnologie wie Screenreader und Braillezeilen unterstützen. Außerdem muss sie u.a. komplett mit der Tastatur zu bedienen sein, damit Menschen, die bei der Interaktion mit einer grafischen Benutzeroberfläche keine

[37] http://www.imobile.com.au/WhatsNew/default.asp?ID=whatmar0301
[38] http://www.marland.de/produkte2.php?caretec=d95313a357725688793356ae1a6bb3a8&id=394
[39] http://www.talx.de/index.shtml
[40] http://www.sprachhandy.de/vias1/index.html

Maus bedienen können, problemlos auf alle Funktionalitäten zugreifen können.
Wie gesehen gibt es mittlerweile offizielle Standards zur barrierefreien Webseitenentwicklung, die u.a. in Deutschland auch gesetzlich verankert sind, wenn auch nur für Bundesanstalten und Angebote, die sich speziell an Menschen mit Behinderungen wenden.
Auch zur Gestaltung zugänglicher Software gibt es mittlerweile eine Reihe von Kriterien, doch wurden diese beinahe ausschließlich von Unternehmen veröffentlicht. Eine Standardisierung zugänglichen Softwareentwurfes ist nicht abzusehen, geschweige denn eine gesetzliche Verankerung.
Auch auf dem Gebiet der mobilen Endgeräte wird die Zugänglichkeit von Anwendungen noch stiefmütterlich behandelt. Hier müssen sich Menschen mit Sehbehinderungen mit Hilfsmitteln bedienen, um die Basisfunktionalitäten von mobilen Endgeräten wie Handys und PDAs nutzen zu können.
Für die Zukunft bleibt zu hoffen, dass Barrierefreiheit ein gesteigertes öffentliches Interesse findet, damit die Mehrzahl der Anwendungen für alle Benutzergruppen gleichermaßen zugänglich ist.

Im Hinblick auf die Arbeit in der Projektgruppe, sollten Aspekte der Zugänglichkeit beachtet werden, wann immer dies möglich ist. Zu empfehlen ist, die Barrierefreiheit der zu entwickelnden Anwendung schon in den Entwurfsprozess einzubeziehen und bei der Implementierung der Anwendung stetig prozessbegleitend zu überprüfen. Hierbei sollte besonderes Augenmerk darauf gelegt werden, dass die Benutzung der Anwendung mit verschiedenen Eingabemedien möglich. Der in der Nutzung von PDAs häufig verwendete „Stift“ sollte nicht die einzige Möglichkeit der Interaktion mit unserer Anwendung sein. Ein Zugriff auf die wichtigsten Programmelemente mittels Kurztasten würde die Benutzerfreundlichkeit und Zugänglichkeit der Anwendung ebenfalls steigern.
Auch ein Audiofeedback von Benutzereingaben wäre eine Möglichkeit, die Zugänglichkeit unserer Anwendung für blinde und sehbehinderte Personen zu erhöhen. Außerdem sollte die farbliche Gestaltung der Benutzungsoberfläche sowie die Schriftgröße der Anwendung individuell einstellbar sein. Dazu wäre es zu überlegen, von vornherein verschiedene Farbprofile zur Verfügung zu stellen, die vom jeweiligen Anwender jedoch noch anpassbar sein sollten.

Aus der Aufgabenstellung der Projektgruppe ist ersichtlich, dass die Ausgabe unserer Anwendung multimodal sein soll, also dazu ausgelegt ist, verschiedene Sinne anzusprechen. Im Zuge der Zugänglichkeit kann dies nur von Vorteil sein, da die Bereitstellung von Informationen auf unterschiedlichen Sinneskanälen ein Schlüsselkonzept von Accessibility ist. So wäre es z.B. ratsam, dass auch im Bezug auf die Präsentation der Inhalte der Benutzer Präferenzen festlegen kann. Ein blinder Benutzer profitiert von der visuellen Darstellung der Inhalte genausowenig wie ein tauber Benutzer von einer akustischen Präsentation.
Am Rande sei noch erwähnt, dass wir auch im Zuge der Barrierefreiheit unsere Projekthomepage zugänglich gestalten sollten.

Kapitel 5

Personalisierung

Abstract Die zunehmende Verbreitung digitaler Geräte und deren Vernetzung nehmen einen immer größer werdenden Stellenwert in unserem alltäglichen Leben ein. Um die damit verbundene Zunahme an Informationsflüssen kontrollierbar zu machen, werden personalisierte Anwendungen eingesetzt. Die Standardisierung der dafür notwendigen Technologien ist bereits weit vorangeschritten und wird ständig weiterentwickelt und verbessert, um den vielfältigen Anforderungen gerecht zu werden. Die Erstellung von persönlichen Profilen ist eine Grundvoraussetzung um orts- und kontextabhängige Dienste anbieten zu können. Die Industrie verwendet diese Informationen, um gezielt Werbung zu betreiben und ihren Kunden personalisierte Angebote unterbreiten zu können.

5.1 Einleitung

Die vorliegende Arbeit wurde angefertigt im Rahmen der Projektgruppe „Kontextsensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“, die semesterübergreifend im Wintersemester 20-04/2005 und im Sommersemester 2005 an der Carl von Ossietzky Universität Oldenburg stattfand. Sie beschäftigt sich mit dem Thema „Personalisierung“ in Bezug auf Dienstleistungen, die einem Benutzer über das Internet zur Verfügung gestellt werden. Nutzbar sind diese Dienste durch Anwendungen, die sowohl auf stationären PCs, als auch auf mobilen Geräten, wie zum Beispiel PDAs und Handys, eingesetzt werden.

Der Einstieg in dieses Thema beginnt in Kapitel 5.2 mit einem Beispiel, gefolgt von einer allgemeinen Beschreibung des Nutzens personalisierter Anwendungen. Das folgende Kapitel behandelt die Grundlagen für die Anbindung mobiler Geräte an das Internet und geht auf die technischen Beschränkung dieser Geräteklasse ein. Der letzte Abschnitt behandelt die Technologien, die für die Personalisierung von Anwendungen eingesetzt werden, wie beispielsweise die Frameworks „Composite Capabilities / Preferencs Profile“ und „User Agent Profile“. Kapitel 5.4 befasst sich mit dem Thema Sicherheit und Schutz personenbezogener Daten. Es wird hier gezeigt, welche Angriffspunkte es für Dritte gibt und welche Techniken vor einem unerlaubten Zugriff auf vertrauliche Informationen schützen. Der Zusammenhang, der zwischen personalisierten Anwendungen und Multimodalität besteht, ist Gegenstand des Kapitels 5.5. Im ersten Abschnitt wird erläutert, welche Bedeutung Multimodalität in Bezug auf die Informatik hat, bevor der zweite Abschnitt den Bezug zwischen Multimodalität und Personalisierung herstellt. Kapitel 5.6 beschäftigt sich mit den Anwendungsgebieten personalisierter Dienste. Anhand dieser Kategorisierung lässt sich schon erkennen, dass die meisten personalisierten Anwendungen kommerzielle Dienste zur Verfügung stellen. Auf diesen Aspekt geht Kapitel 5.7 näher ein. Hier wird die Bedeutung personalisierter

Anwendungen für den elektronischen Handel aufgezeigt. Die Ausführungen basieren auf einer Studie der „Hong Kong University of Science and Technology“ aus dem Jahr 2003. Ziel der Untersuchung war es herauszufinden, unter welchen Bedingungen Benutzer mobiler Endgeräte zu einem Anbieter wechseln, der personalisierte Dienste vertreibt.

5.2 Was ist Personalisierung?

5.2.1 Ein Beispiel

Um den Einstig in das Thema „Personalisierung“ anschaulich zu gestalten, bietet es sich an, ein Beispiel für eine personalisierte, webbasierte Anwendung aufzuzeigen. Bei [WBHK02] wird ein Szenario dargestellt, wie einem Benutzer der Umgang mit alltäglichen Problemen durch den Gebrauch personalisierter Dienste erleichtert werden kann.

Die Ausgangssituation ist hier, dass ein Geschäftsmann zu einem Treffen in einer fremden Stadt mit dem Flugzeug unterwegs ist. Bei der Ankunft auf dem Flughafen steht für ihn ein bereits reserviertes Mietfahrzeug bereit. Die Feststellung seiner Identität sowie die Autorisierung seiner Reservierung bei der Autovermietung erfolgt dabei über sein Handy. Sobald der Mann in sein Auto steigt, synchronisiert sich sein PDA[1] mit der Elektronik des Wagens, die die bevorzugten Einstellungen wie Sitzposition, Spiegelausrichtung und Heizungstemperatur von dem PDA übermittelt bekommt. Ebenso wird von dem PDA das Navigationssystem des Autos erkannt und die Daten über Zeit und Ort des Geschäftstreffens an dieses übermittelt. Daraufhin werden entsprechend die benötigten Straßenkarten automatisch vom Navigationssystem geladen, um eine optimale Route zum Fahrtziel unter Berücksichtigung der aktuellen Verkehrsinformationen zu berechnen. Weil das System ermittelt, dass das Fahrtziel noch deutlich vor Beginn des Geschäftstreffens erreicht werden kann, entscheidet sich der Mann, bei einer nahe gelegenen Bank vorher noch Geld zu wechseln. Dazu ruft er einen Dienst ab, der ihm den Weg zu einer Bank in der näheren Umgebung erklärt, die die geringsten Umtauschkosten für den Geldwechsel berechnet. Mit diesen Informationen kann der Weg zu dem Geschäftstreffen neu berechnet werden, auf dessen Strecke eine geeignete Bank liegt. Dabei muss ein Kompromiss gefunden werden zwischen einer Fahrtzeit, die nicht überschritten werden darf, um den Beginn des Treffens nicht zu verpassen, sowie einer geeigneten Strecke zu einer möglichst nahe gelegenen Bank, die die Transaktion zu geringen Umtauschkosten anbietet. Ein weiterer, aus dem Internet abrufbarer Dienst ermittelt einen in der Nähe der Bank gelegenen Parkplatz.

In dem Gebäude angekommen, in dem das Treffen stattfindet, synchronisieren sich Handy, Laptop und PDA umgehend mit der vorhandenen und zur Verfügung gestellten Elektronik. Sobald erkannt wird, dass die Konferenz beginnt, werden die persönlichen Einstellungen der Geräte automatisch aktualisiert. Beispielsweise werden eingehende Anrufe nur signalisiert, wenn es sich um einen Notruf handelt, eingehende E-Mails werden lautlos angekündigt.

Da ein Teilnehmer des Treffens aufgrund eines Notrufs dieses verlassen wird, werden die Terminpläne aller Beteiligten an eine zentrale Terminverwaltungssoftware übermittelt, die dann einen geeigneten Zeitpunkt für eine Fortsetzung des Treffens aufzeigen kann. Währenddessen möchte sich der Mann mit anderen Personen treffen, die in dem Gebäude arbeiten. Dazu kann sein Laptop über das W-LAN des Gebäudes einen Arbeitsplan abrufen, der Auskunft über die derzeitige Beschäftigung oder freie Zeiten der Angestellten gibt, ebenso wie über den Ort des Büros oder die Telefonnummer.

[1] Abkürzung für: „*Personal Digital Asstistant*“

5.2.2 Personalisierung im Alltag

Dieser erdachte Ablauf verdeutlicht durch die vielen kleinen Beispiele für zukünftige oder bereits alltägliche Vorgänge im Wesentlichen einen wichtigen Aspekt der Personalisierung, nämlich die Unterstützung und Bewältigung von kleinen Hindernissen und Problemen in verschiedenen Situationen des alltäglichen Lebens durch unterschiedliche elektronische Geräte. Dabei sind es vor allem automatisierte Abläufe, die dem Benutzer Arbeit abnehmen und Zeit sparen können.

In dem Beispiel beginnt die Unterstützung, indem das Handy alle Informationen zur Identifikation seines Besitzers bereithält und die Autorisierung der Autoreservierung so in wenigen Sekunden ablaufen kann. Im Auto werden dann die bevorzugten und gewohnten Einstellungen des Fahrers, wie die Position des Sitzes, automatisch von seinem PDA an die Elektronik des Fahrzeugs übertragen. Dabei muss der Benutzer keinen Vorgang per Hand initialisieren. Sein PDA erkennt die neue Umgebung, baut automatisch eine Verbindung zu den Systemen des Fahrzeugs auf und transferiert die Daten, die für die Einstellungen der Fahrzeugausstattung relevant sind. In jedem Auto, das der Mann fährt, kann die Einrichtung des Wagens an seine persönlichen Vorgaben automatisch angepasst werden.

Um die optimale Route zum Treffpunkt zu errechnen, lädt das Navigationssystem die relevanten Karten selbständig in Abhängigkeit von den Informationen über das Treffen aus dem Terminplan des PDAs. Hierbei ist zu beachten, dass das System lediglich die Absicht des Benutzers kennen kann, auf einem möglichst schnellen Weg zu dem Treffen mit den Geschäftspartnern zu gelangen. Denkbar wäre auch die Situation, das der Mann vor dem Treffen noch eine Besichtigungsfahrt durch die Stadt unternehmen möchte. Wäre dies noch eine Option oder gar das Primärziel der Reise gewesen, hätte der PDA entsprechende Daten darüber enthalten müssen.

Aber nicht nur die Unterstützung des Benutzers durch das Anbieten spezieller Dienste, sondern auch die Beeinflussung des Empfindens im Umgang mit elektronischen Geräten und digitalen Medien ist ein wichtiger Gesichtspunkt, der durch Personalisierung beeinflusst werden kann. Gerade in diesem Bereich sind im Internet viele Beispiele zu finden, meist in Form von Plattformen, die sich ein Benutzer im Bezug auf Design und Layout selbst gestalten kann. Nach einem „Login“ mit einer persönlichen Kennung und einem Passwort kann sich der Benutzer die Webseiten eines Portals nach seinen Vorstellungen, Wünschen und Bedürfnissen einrichten. Dabei ist die Personalisierung nicht nur auf Äußerlichkeiten beschränkt, auch inhaltlich lassen sich Webseiten personalisieren. Ein gutes Beispiel ist in diesem Zusammenhang das Internetportal „*Yahoo!*“. Dieser Anbieter stellt mit „*My Yahoo!*“ einen Bereich zur Verfügung, in dem sich jeder angemeldete Benutzer seine personalisierte Webseite zusammenstellen kann. Inhalte werden in Form von Modulen angeboten, die sich mit verschiedenen Themen befassen, beispielsweise Nachrichten, Börsenkursen oder einer Wettervorhersage. Diese lassen sich zum einen bezüglich ihres Inhalts anpassen, so dass ein Benutzer auf Wunsch hin nur Informationen über das Wetter in seiner Region oder Nachrichten aus dem Bereich Politik erhält. Zum anderen können die Module in ihrer Größe verändert sowie beliebig auf der Seite platziert werden. Die ganze Webseite kann dann noch farblich gestaltet werden. Der Benutzer kann sich dabei entscheiden, ob er eigene Farbkombinationen für die verschiedenen Elemente der Seite erstellen möchte, oder ob er sich für ein vorgefertigtes Design entscheidet.

5.3 Personalisierung im Zusammenhang mit mobilen Anwendungen

5.3.1 Voraussetzungen für die Personalisierung mobiler Anwendungen

Eine wichtige Voraussetzung für die Realisierung von personalisierten Diensten auf mobilen Endgeräten wird an zwei Stellen des Beispiels deutlich. Sowohl im Auto als auch in dem Gebäude, in dem das Treffen stattfindet, erkennen die mobilen Geräte des Benutzers die Elektronik der neuen Umgebung und können mit den Systemen eine Verbindung aufbauen, um Daten auszutauschen. Möglich gemacht wird dieses erst durch die immer weiter fortschreitende Vernetzung digitaler Medien auf der ganzen Welt. Dabei spielt nicht nur die Kommunikation zwischen wenigen Geräten über eine kurze Entfernung eine Rolle. Personalisierte Dienste werden dadurch attraktiv, indem sie über das Internet an den verschiedensten Orten zur Verfügung stehen. Für die Anbindung mobiler Geräte an das „World Wide Web“ wird momentan der Mobilfunkstandard GSM (Global System for Mobile Communication) eingesetzt. Dieses Netzt ist ein System der zweiten Generation, ermöglicht eine Kommunikation über Kontinente hinweg und bietet dabei eine Übertragungsgeschwindigkeit von 9,6 kbis/s [Mat03]. Für Sprachübertragung ist diese Kanalkapazität ausreichend, für die Übertragung von Multimediadateien und Videoströme werden Standards mit höheren Transferraten das GSM-System ablösen. Geeignete Kandidaten sind GPRS und UMTS mit Bandbreiten von 144 kbit/s bzw. 2 MBit/s.

5.3.2 Hindernisse und Beschränkungen

Der Abruf von Diensten über das Internet gestaltet sich schwierig hinsichtlich der Darstellung auf mobilen Geräten, insbesondere wenn komplexe Webseiten angezeigt werden sollen. In dem Beispiel könnte dies der Fall sein, wenn der Benutzer einen Dienst über sein Handy anfordert, der einen in der Nähe der Bank gelegenen Parkplatz suchen und den Weg dorthin beschreiben soll. Die Größe des Handydisplays ist in erster Linie dafür verantwortlich, dass die Informationen erst in ein geeignetes Format umgewandelt werden müssen, damit sie entsprechend ihres Informationsgehalts adäquat darstellbar sind. Handys bieten üblicherweise nur Platz für drei bis vier Zeilen Text, PDAs haben meistens eine etwas größere Displayfläche auf denen auch Grafiken dargestellt werden können. Ebenfalls begrenzt ist der Speicherplatz auf mobilen Endgeräten. Aufgrund dieser Gegebenheiten muss ein Anbieter, der einen Dienst zur Verfügung stellt, die Daten in der entsprechenden Form bereitstellen. Aus diesem Grund ist es notwendig, neben der Anforderung von Daten ebenfalls noch Informationen über die technische Austattung des Gerätes mitzuschicken.

Den Anschluss mobiler Geräte an das Internet ermöglicht seit 1999 der WAP-Standard (Wireless Application Protocol). Dieses Protokoll berücksichtigt generell die Beschränkungen wie Displaygröße und eine langsame Übertragungsgeschwindigkeit zwischen einem Server und einem Client, so dass die Inhalte den Geräten in einer entsprechenden Weise zur Verfügung gestellt werden. Die Anfrage eines WAP-Clients wird an einen WAP-Gateway geleitet, der die Kommunikation mit einem Webserver durchführt. Weil dieser die Daten im für das WAP-Gerät unlesbaren HTML-Format sendet muss der Gateway diesen Code in die WML-Sprache (Wireless Marup Language) übersetzen.

Das WAP-Protokoll bietet jedoch nur die Grundlage, um mobile Geräte überhaupt mit dem Internet vernetzen zu können. Um personalisierte Dienste zu realisieren, müssen weitere Techniken eingesetzt werden, mit denen sich der folgende Abschnitt beschäftigt.

5.3.3 Techniken und Technologien zur Vernetzung mobiler Endgeräte

Die Grundlegende Infrastruktur, die eine Kommunikation zwischen personalisierten Anwendungen und mobilen Endgeräten überhaupt erst möglich macht, ist das Internet, basierend auf der Internet Protocoll (IP) Technologie. Auf dieser mächtigen Plattform findet eine Kommunikation zwischen Geräten, Netzwerken, Anwendungen gleicher Art oder Struktur ebenso wie zwischen heterogenen Geräten, Netzwerken und Anwendungen[2] statt. Verschiedene Endgeräte bringen in Bezug auf Hardware, Software, Ein- und Ausgabegeräte sowie Netzwerkfähigkeit[3] unterschiedliche Voraussetzungen mit. Personalisierung bedeutet in dieser Hinsicht, dass ein Webserver, der eine Anfrage von einem mobilen Gerät erhält, die Daten in einem Format an dieses zurückschickt, in dem sie ohne Probleme ausgegeben werden können. Techniken, die diese Aufgaben erleichtern sollen, sind das vom W3C (World Wide Web Consortium) entwickelte „Composite Capabilities / Preferences Profile“ (CC/PP) und das von der Open Mobile Alliance (ehemals WAP-Forum) entworfene „User Agent Profile“ (UAProf). Diese beiden Standards sind Frameworks, die einem Server Daten über die technischen Voraussetzungen des Gerätes eines Benutzers liefern. Diese Metadaten werden in Form eines XML- oder RDF-Schemas übermittelt. XML und das darauf basierende RDF-Format wurden entwickelt, um über das Web übertragenen Informationen eine semantische Bedeutung zu geben. Ziel dieser Anreicherung von Daten mit zusätzlichen Informationen ist es, dass sowohl Menschen, in erster Linie aber auch Maschinen diese Daten interpretieren können. In einer Anforderung einer Abfrage eines Internetdienstes können auf diesem Weg zusätzliche Informationen über die technischen Gegebenheiten eines Gerätes codiert werden.

Neben den Daten, die das verwendete Gerät beschreiben, muss der Server auch noch Kenntnis über die Bedürfnisse des Kunden haben, um einen Dienst personalisiert anbieten zu können. Informationen über einen Benutzer werden auf dessen Gerät in einem „Personal Profile“ gespeichert. Ein „Personal Profile“ übernimmt eine Aufgabe ähnlich den bisher verwendeten „Cookies“. Es enthält benutzerspezifische Informationen wie zum Beispiel „*Name, Adresse oder persönliche Neigungen und Interessengebiete*“ [HZ97], die einem Webserver zur Verfügung gestellt werden können, um eine personalisierte Antwort zu erzeugen. Um die Effizienz der Übertragung eines „Personal Profile“ an einen Server zu steigern, kommen „Default Profiles“ zum Einsatz. Ein „Default Profile“ ist ein allgemeines Profil eines Benutzerstyps. Für die Personalisierung einer Anwendung sind dann nur noch die Informationen über einen Benutzer zu übertragen, die nicht im „Default Profile“ enthalten sind. In dem Beispiel kennzeichnet das Standardprofil den Geschäftsmann als Teilnehmer des Treffens und eingehende Anrufe werden automatisch abgeblockt. Nur im Falle eines Notrufs wird er benachrichtigt, denn diese Einstellung ist Teil seines persönlichen Profils. Für den Fall, dass ein Dienst angefordert wird, für den ein persönliches Profil zu konkrete Vorgaben enthält, können die Informationen mit Hilfe eines „Usage Pattern“ verallgemeinert werden, bis der entsprechende Dienst zur Verfügung gestellt werden kann. Abbildung 5.1 auf Seite 84 zeigt den Ablauf, wenn eine personalisierte Anwendung aufgerufen wird.

[2] Vergleiche [Rie00b], Seite 1.
[3] Vergleiche [WBHK02], Seite 2.

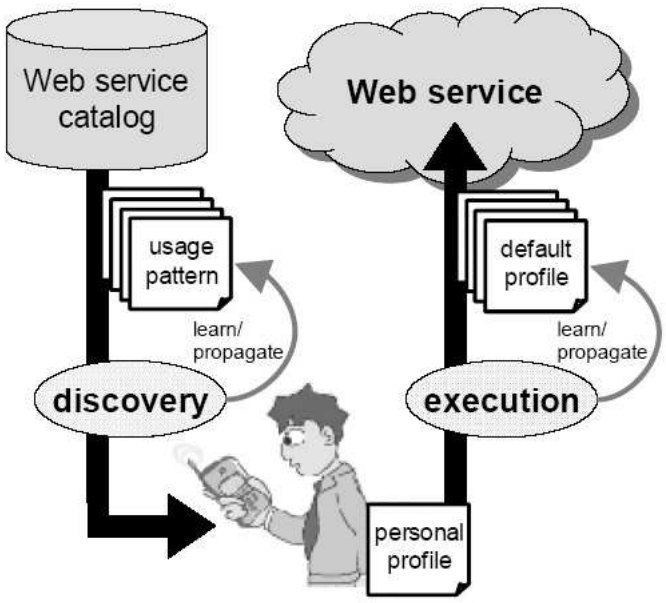


Abbildung 5.1: Anforderung eines personalisierten Dienstes

Eine Neuerung gegenüber der Cookie-Technologie ist ein Mechanismus, durch den ein Benutzer die Kontrolle darüber behält, welcher Webserver auf welche Daten zugreifen darf. Dieser Standard wurde vom „World Wide Consortium“ (W3C) im „Platform for Privacy Preferences Project“ (P3P) definiert, und soll das Vertrauen der Anwender bezüglich des Umgangs und des Schutzes privater Daten stärken. Dazu wird auf einem Server ein P3P-Dokument definiert in dem die Datenverarbeitungspraktiken abgelegt sind. Der Benutzer macht in seinem Browser neben den persönlichen Daten auch Angaben gemäß seinen Vorstellungen, wie ein Server mit diesen Informationen umgehen soll. Bei einem Zugriff werden die Datenschutzeinstellungen des Servers mit denen im Browser verglichen und der Benutzer bei einer Abweichung informiert[4].

5.4 Schutz und Sicherheit personenbezogener Daten

Ein kritisch zu betrachtender Aspekt der Personalisierung von Anwendungen entsteht durch die Verwendung persönlicher Daten eines Benutzers. Dritte können aus verschiedenen Gründen ein Interesse daran haben, personenbezogene Daten anderer in Erfahrung zu bringen. Ein möglicher Angriffspunkt, um an vertrauliche Informationen zu gelangen, ergibt sich während einer Kommunikation zwischen dem Gerät eines Benutzers und dem Server eines Dienstanbieters. Dieser Vorgang muss so gesichert sein, dass der Transfer nicht abgehört werden kann. Ebenfalls inakzeptabel ist eine Manipulation der Daten während der Übertragung. Dabei können zwei Fälle unterschieden werden. Zum einen könnten die Daten absichtlich verfälscht werden, so das sie uninterpretierbar beim Empfänger eintreffen. Dieser Eingriff kann lediglich als störend empfunden werden, allenfalls muss lediglich die Kommunikation unterbunden und neu initiiert werden. Der weitaus schlimmere Fall tritt ein, wenn die Daten derart verändert werden, dass sie eine neue Bedeutung erhalten. Das kann zu ungewollten Ergebnissen führen und stellt gerade bei sicherheitskritischen Anwendungen, zum Beispiel beim Online-Banking, eine große Gefahr dar. Gegen diese Beeinflussung von Außen müssen geeignete Maßnahmen und Vorkehrungen getroffen werden,

[4] Vergleiche [Möl03], Abschnitt „*Wie funktioniert P3P?*“.

denn sonst würde das Vertrauen der Benutzer in diese Technologie erheblich erschüttert werden. Mechanismen, die den Schutz der Daten bei der Übertragung garantieren sollen, sind XML-Signaturen und XML-Verschlüsselung[5].

Ein weiterer Aspekt, der das Vertrauen der Benutzer in verschiedene, über das Internet angebotene Dienste beeinträchtigen könnte, ist die Art des Umgangs des Dienstanbieters mit vertraulichen Daten. Um das Vertrauen zu stärken, muss ein Server die Möglichkeit bieten, eine gesicherte Verbindung zu einem Client aufzubauen, wenn sensible Daten, wie beispielsweise Kreditkarteninformationen, übertragen werden. Eine gängige Verschlüsselungstechnik auf HTTP-Ebene ist SSL (Secure Socket Layer), die auch für die sichere Übertragung von WML-Dokumenten eingesetzt werden kann.

Eine mächtige Kontrollinstanz für einen Benutzer bietet die von dem World Wide Web Consortium (W3C) entwickelte und standardisierte Plattform P3P (Platform for Privacy Preferences Projekt). Diese Plattform bietet die Möglichkeit, dass die Sicherheitseinstellungen einer Webseite, an die Informationen übermittelt werden sollen, für einen Benutzer sichtbar gemacht werden können. Er entscheidet dann für jedes einzelne Datum selbst, ob es an den Server geschickt werden soll oder nicht.

5.5 Multimodalität und Personalisierung

5.5.1 Der Begriff „*Multimodalität*“

Um zu erläutern, welche Bedeutung Multimodalität für die Personalisierung im Zusammenhang mit mobilen Endgeräten hat, muss man wissen, was unter diesem Stichwort zu verstehen ist. In der Psychologie bezeichnet der Begriff Modalität die Wahrnehmung von Reizen zur Informationsaufnahme über ein Sinnesorgan. Der Mensch verfügt über viele verschiedene Sinneskanäle, die mit unterschiedlicher Intensität Reize aus der Umwelt aufnehmen können. Der Informationsflut entsprechend sind einige Sinne für den Menschen nahezu unverzichtbar. In erster Linie sind dies der visuelle (Sehen), auditive (Hören), taktile (Fühlen) olfaktorische (Riechen/Schmecken) und vestibulare (Orientierung) Sinn. Multimodalität bedeutet in diesem Zusammenhang, dass eine Reizaufnahme über mehrere Kanäle gleichzeitig stattfindet und die empfangenen Informationen parallel verarbeitet werden. Im Bezug auf die Informatik wird der Begriff „Multimodalität“ eingesetzt, um die Schnittstelle zwischen Mensch und Maschine zu definieren. In dem Beispiel aus Abschnitt 5.2.1 wäre die Informationsausgabe multimodal, wenn das Navigationssystem den Weg zu einem Ziel auf einem Bildschirm ausgeben und gleichzeitig Kommandos geben würde, in welche Richtung der Fahrer das Fahrzeug steuern müsste.

Die heutzutage in der Praxis eingesetzten Kommunikationsmöglichkeiten beschränken sich auf die visuelle, auditive und taktile Informationsübertragung, beispielsweise auf die Eingabe von Daten und Befehlen an eine Maschine über die Tastatur oder einen Stift bei einem Touch-Screen-Display, sowie die Ausgabe von Informationen über ein Display oder in akustischer Form.

5.5.2 Die Bedeutung multimodaler Anwendungen

Ein Bereich, in dem Multimodalität für die Personalisierung eine wichtige Rolle spielt, ist der Abruf von Diensten über ein mobiles Gerät, zum Beispiel ein Handy. Um die Mobilität dieser Geräteklasse zu wahren, werden sie möglichst klein gehalten und kompakt gebaut. Daraus resultiert, das auch die Schnittstelle, die die Kommunikation zwischen Mensch und Maschine ermöglicht, nur wenig Platz in Anspruch nehmen kann. Displays von PDAs, die schon zu der mittelgroßen Kategorie mobiler Endgeräte

[5]Für weitere Informationen siehe [W3Cc].

zählen, bieten meistens eine Auflösung von nicht mehr als 640x480 Pixel. Im Gegensatz zu Handydisplays, auf denen lediglich drei bis vier Zeilen Text darstellbar sind, können auf einem PDA sowohl Texte, als auch Bilder und Tabellen ausgegeben werden. Die Eingabe von Informationen und Daten erfolgt über einen Touch-Screen, der in der Regel mit Hilfe eines Stifts bedient wird[6]. Eine Handytastatur hingegen besteht lediglich aus einer Zehnertastatur und einigen Funktionstasten. Da über diese Eingabeschnittstelle auch Texte verfasst werden müssen, ist jede Taste mit mehreren Zeichen belegt. Die Auswahl eines Zeichens wird durch die Häufigkeit bestimmt, mit der eine Taste gedrückt wurde. Unter diesen Gegebenheiten ist es sinnvoll und teilweise sogar notwendig, alle Interaktionswege und -möglichkeiten effizient auszuschöpfen. Eine Ausgabe in Form von akustischen Signalen oder aber die Eingabe von Befehlen und Daten in der natürlichen Sprache ist heute schon in einigen Bereichen in der Praxis möglich.

Der Einsatz einer multimodalen Interaktion ist aber nicht nur in Bereichen sinnvoll, in denen die Kommunikationsmittel begrenzt sind. Die „Verlagerung der Informationsübertragung auf mehrere Sinne, wodurch die Beanspruchung eines Sinnes und des Benutzers insgesamt zu reduzieren ist“ [TJK02] führt zu einer für den Menschen natürlicheren Kommunikation im Ungang mit dem benutzten Gerät. Multimodalität kann daher ebenso als Komfortmerkmal angesehen werden. Nützlich ist beispielsweise der Einsatz eines Vibrationsalarms am Handy, so das der Besitzer über einen eingehenden Anruf informiert wird, ohne dass die Menschen in der näheren Umgebung dies unbedingt mitbekommen. Der Benutzer kann entscheiden, ob das Handy nur klingeln, nur vibrieren, oder aber beide Signale verwenden soll.

Auch der Sicherheitsaspekt darf nicht vernachlässigt werden. Aus diesem Grund ist es inzwischen (in Deutschland) verboten, dass ein Straßenverkehrsteilnehmer während der Fahrt sein Handy benutzt, ohne eine Freisprecheinrichtung zu verwenden. Seine Wahrnehmungsfähigkeit soll dadurch nicht unnötig hoch auf die Bedienung des Telefons fixiert, sondern in erster Linie auf das Geschehen auf der Straße gerichtet sein. In diesem Bereich ist daher auch die Steuerung des Mobilen Telefons, wie zum Beispiel die Aus- und Anwahl von Rufnummern durch Spracheingabe weit vorangeschritten.

Eine ganz wesentliche Bedeutung bekommen multimodale Anwendungen in Systemen, die für Personengruppen mit eingeschränkten Wahrnehmungsfähigkeiten zur Verfügung gestellt und von ihnen in Anspruch genommen werden können. Personalisierung heißt in diesem Zusammenhang, dass keine Person aufgrund einer Einschränkung von der Benutzung eines Dienstes ausgeschlossen wird. So ist über das Internet beziehbare Software erhältlich, die blinden Menschen die Inhalte einer Webseite vorliest. Voraussetzung ist allerdings, dass bei der Gestaltung der Seite Standards eingehalten wurden, die der Software die Interpretation der Inhalte ermöglicht.

5.6 Einsatzbereiche personalisierter (mobiler) Anwendungen

Eine Plattform, durch die personalisierte Anwendungen einen erheblichen Fortschritt in ihrer Entwicklung gemacht haben, ist das Internet und damit die zunehmende Vernetzung stationärer und mobiler Geräte bis in die privaten Haushalte. Eine weite Verbreitung haben die Systeme aber in erster Linie in kommerziellen Geschäftsfeldern und für wirtschaftliche Prozesse gefunden. Kostenpflichtige Dienste werden zum Beispiel in den Bereichen der elektronischen Unterhaltung und des virtuellen Einkaufs angeboten. Dabei können Produkte nicht nur über das Handy angeboten, sondern auch gleich von diesem aus bezahlt werden. Ein großer Vorteil gegenüber traditionellen Systemen ist die Möglichkeit, Dienste auf mobilen Geräten orts- und kontextbezogen anbieten zu können.

Ein weiteres Gebiet, auf dem personalisierte Anwendungen vermehrt zum Einsatz kommen, ist zum Beispiel das Gesundheitswesen. Ein Arzt kann beispielsweise seinen PDA als Informationsquelle nutzen, um

[6] Vergleiche [Mat03], Seite 89.

Informationen über „Medikamente, Checklisten, Therapieschema, etc.“[7] von einem zentralen Server abrufen zu können. Ganze Lehrbücher lassen sich so als Nachschlagewerke archivieren und sind bequem von überall aus abrufbar. Die Arbeit eines Arztes lässt sich auf diese Art aus dem Büro in die Krankenzimmer verlagern, „wodurch mehr Zeit für die persönliche Betreuung von Patienten geschaffen werden kann“[8]. Durch den zunehmenden Einsatz elektronischer Geräte und deren Vernetzung untereinander kann eine zunehmend bessere Überwachung der Patienten realisiert werden. Im folgenden sind noch einmal die wesentlichen Einsatzbereiche personalisierter mobiler Anwendungen aufgezeigt und mit Beispielen verdeutlicht:

- **Web-Portale**
  Internet-Portale sind traditionelle Anwendungen, die von stationären PCs aus genutzt werden können. Benutzer können auf diesen Plattformen verschiedene personalisierte Dienste vereinen. Für den mobilen Bereich sind Portale aufgrund ihres Umfangs und der damit verbundenen Datenkapazität jedoch bisher ungeeignet. Ein Beispiel für ein Internet-Portal ist der personalisierte Teil von „Yahoo!“, der unter der Domain „my.yahoo.com“ erreichbar ist. Der Aufbau einer *My Yahoo!*-Webseite ist in Abbildung 5.2 auf Seite 88 dargestellt.

- **Finanzgeschäfte**
  Bankgeschäfte und Aktienhandel über das Internet zu tätigen wird von vielen Banken bereits angeboten. Der Zugang erfolgt über einen stationären PC. In Zukunft werden Kunden über ihr Handy ihren Kontostand abrufen, Überweisungen tätigen und Rechnungen bezahlen können. Gerade im Bereich des Aktienhandels ist es wichtig, auf Veränderungen schnell reagieren zu können. Die mobile Nutzung dieses Dienstes ermöglicht einen ortsunabhängigen Zugriff auf dieses Geschäft zu jeder Zeit.

- **Tickets**
  Das Angebot, Reise und Tickets im Internet zu buchen wird bereits heute vielfach wahrgenommen. Informationen über Sonderangebote zu jeder Zeit über das Handy zu empfangen wird diese Anwendung noch attraktiver machen. In Kombination mit einem Dienst, der die Position eines Kunden bestimmen kann, wird die Möglichkeit eröffnet, lokale Angebote zu unterbreiten.

- **Shopping**
  Mobile Shopping Angebote sind zurzeit noch eher die Ausnahme, der Bereich des Verkaufs von Produkten über mobile Netze beschränkt sich bisher lediglich auf den Ticketverkauf, Auktionen und Reservierungen. Ein Hindernis in diesem Bereich ist die beschränkte Darstellungsfähigkeit mobiler Geräte, da Kunden die Waren vor einem Kauf in der Regel begutachten wollen. Ein großes Potential liegt jedoch in diesem Bereich in dem Vertrieb standardisierter Produkte, die mit wesentlich geringerem Aufwand angeboten werden können.

- **Entertainment**
  Gerade im Bereich mobiler Anwendungen sind Multimedia- und Entertainment Angebote gefragt. Musikstücke, kleine Filmsequenzen und Videospiele können direkt auf das Handy geladen werden. Der heimische PC braucht dabei nicht mehr als Zwischenspeicher genutzt zu werden. Der Einsatz neuer Übertragungstechnologien, die eine höhere Sendeleistung erreichen, wie zum Beispiel UMTS, wird die Verbreitung Multimedialer Angebote vorantreiben.

[7] Siehe [Mat03], Seite 97.
[8] Siehe [Mat03], Seite 98.

- **Ortsabhängige Dienste**
  Dieser Anwendungsbereich umfasst lediglich mobile Geräte, die via GPS[9] oder vergleichbarer Techniken zu orten sind. Mit den Informationen über den Aufenthaltsort eines Benutzers können Suchdienste für Einrichtungen in einer Umgebung angeboten werden, wie zum Beispiel die kürzesten Wege zu dem nächstgelegenen Hotel, einer Bank, einem Parkplatz, einem Einkaufszentrum, etc.

- **Personalisierte Dienste**
  Unter diesem Begriff können alle Dienste zusammengefasst werden, die für stationäre PCs als Web-Portal über das Internet angeboten werden. Auf mobilen Geräten können personalisierte Dienste nicht vereint auf einer mächtigen Plattform angeboten werden. Hier müssen sie einzeln zur Verfügung gestellt werden. Voraussetzung für das Angebot eines personalisierten Dienstes ist, dass ein Anbieter Informationen über den Benutzer zur Verfügung gestellt bekommt. Beispiele für Anwendungen aus diesem Bereich sind unter anderem eine Nachrichtenanzeige aus einer Sparte, ein mobiler E-Mail Abrufservice, Aktienkurse von bestimmten Papieren, etc. Personalisierte Dienste können auch mit ortsgebunden Diensten kombiniert werden.

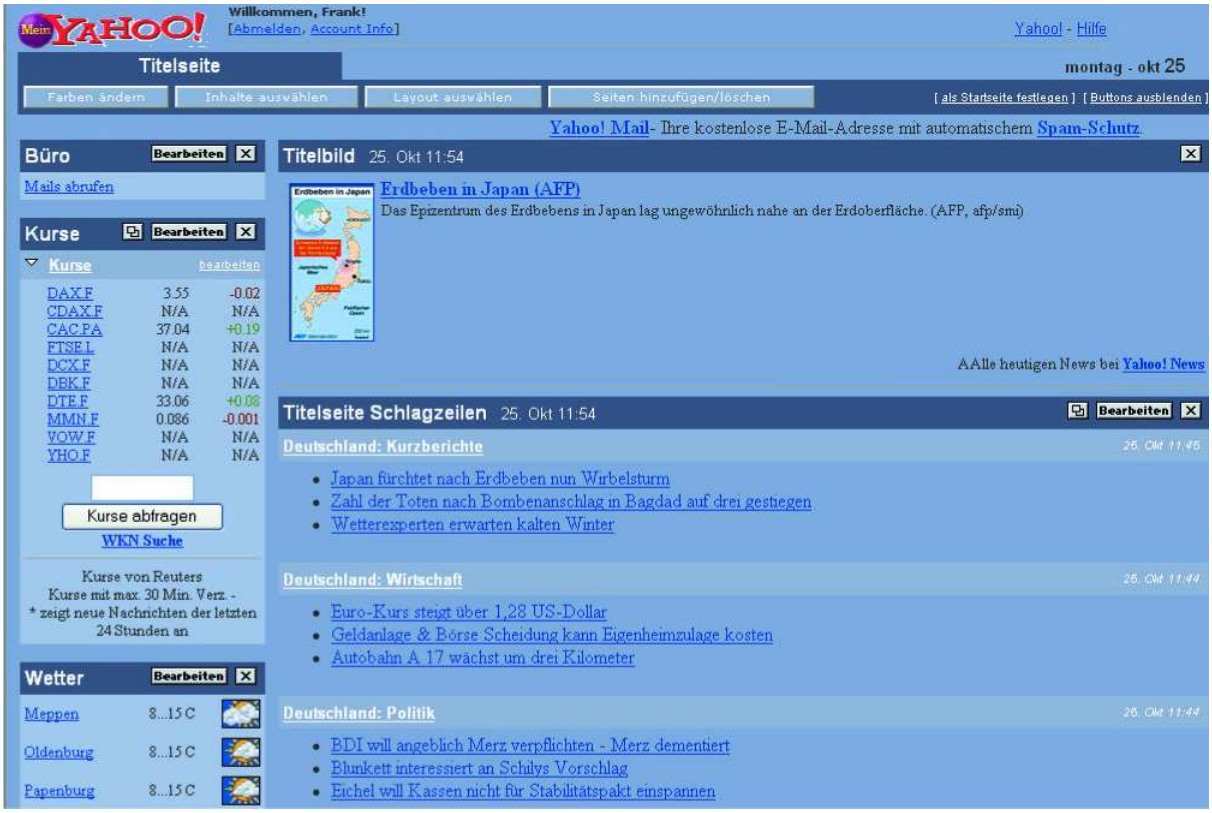


Abbildung 5.2: Personalisierter Web-Dienst

[9] Abkürzung für: „*Global Positioning System*“

5.7 Personalisierte Anwendungen im Bereich des elektronischen Handels

Nach großen Erfolgen im Bereich des elektronischen Handels[10] entdecken viele Firmen den Bereich des mobilen Handels[11] für sich. Die Bedingungen sind ideal. Die Möglichkeit des Handels über das Internet beschränkt sich dadurch nicht mehr nur auf den heimischen PC. Viele Kunden sind inzwischen jederzeit und überall über den Handy-Dienst SMS erreichbar. Auf diesem Weg kann dem Nutzer Werbung unterbreitet und personalisierte Angebote gemacht werden, je nach Interessensgebiet des entsprechenden Kunden. Die Vorgaben, welche Informationen er erhalten möchte, legt ein Benutzer selber fest. Um Anwendungen zu programmieren, die einem Handybesitzer einen personalisierten Dienst auf dessen Gerät zur Verfügung stellen, müssen Programmierer auf besondere Entwicklungsumgebungen zurückgreifen. Die Software muss nämlich so konzipiert werden, dass sie unter den Hardwaretechnischen Voraussetzung, die ein Handy bietet, lauffähig ist. Plattformen, mit denen Entwickler mobile Anwendungen programmieren können, sind beispielsweise das von Qualcomm entwickelte Framework BREW sowie Suns Java 2 Micro Edition (J2ME). Die entsprechenden Endgeräte, auf denen diese Anwendungen lauffähig sind, werden unter anderem von Nokia und Motorola hergestellt.

Mobile Endgeräte haben wichtige Eigenschaften, die sich Firmen zu Nutze machen können, um ihren Kunden persönliche Angebote und Informationen zur Verfügung zu stellen. Aufgrund ihrer Größe kann der Benutzer sie überall mit hinnehmen und ist dadurch jederzeit erreichbar. Die Position eines Handys lässt sich genau bestimmen, und in die Geräte sind Sicherheitsmechanismen integriert, die den mobilen Handel sicherer machen werden als das bisherige Einkaufen über das Internet. Zudem sind die Geräte einfacher zu bedienen als ein PC, schneller betriebsbereit und kostengünstiger in der Anschaffung. Ein Nachteil der Miniaturisierung der Geräte ist die kleine Fläche des Displays eines Handys, auf der nur drei bis vier Zeilen Text dargestellt werden können. Wenn ein Benutzer jedoch mehr als vier Werbenachrichten zugeschickt bekommt, muss er über die kleine Tastatur das Bild „scrollen“, um alle Nachrichten lesen zu können. Das kann aber zu Verärgerungen führen, besonders dann, wenn die Informationen eher allgemein gehalten wurden. Letztendlich kann es passieren, dass ein Benutzer empfangene Nachrichten nicht weiter beachtet und ungelesen von seinem Gerät löscht. Auch die für den Benutzer interessanten Informationen würden in der Masse der generellen Angebote unbeachtet bleiben. Durch den Einsatz personalisierter Werbung kann dieser Effekt verhindert werden. Die Masse der Informationen wird reduziert und dadurch die Aufnahmefähigkeit des Konsumenten gesteigert.

An der „Hong Kong University of Science an Technology“ wurde 2003 eine Studie durchgeführt, in der untersucht wurde, ob das Angebot eines personalisierten Dienstes Benutzer dazu bringen kann, zu einem Dienstanbieter zu wechseln, der diesen bereitstellt. Folgende Thesen wurden zur Untersuchung aufgestellt[12]:

- **H1:** Allgemeine Angebote lassen Konsumenten zu einem Anbieter wechseln, der personalisierte Dienste vertreibt.
- **H2:** Schwierigkeiten beim Herausfiltern nützlicher Informationen aus einer Menge allgemeiner Angebote lassen Konsumenten zu einem Anbieter wechseln, der personalisierte Dienste vertreibt.
- **H3:** Wenn ein Benutzer merkt, dass ein personalisierter Dienst nützlich ist, wird er geneigt sein zu einem Dienstanbieter zu wechseln, der personalisierte Dienste vertreibt.

[10] Synonym für *e-commerce*.
[11] Synonym für *m-commerce*.
[12] Für die weiteren Ausführungen vergleiche [HK03].

- **H4:** Sicherheitsbedenken in Bezug auf Personalisierung halten einen Kunden davon ab zu einem Anbieter zu wechseln, der personalisierte Dienste vertreibt.

Zur Durchführung der Untersuchung wurden 350 Computertechniker und Vollzeitstudenten aus der Abteilung „Computer Science and Information System“ der „Hong Kong University“ befragt. Die Bearbeitungszeit, um den 45 Fragen umfassenden Fragebogen auszufüllen, betrug zwanzig Minuten. Alle Teilnehmer hatten einen Vertrag mit einem Mobilfunkanbieter und Erfahrungen im Umgang mit einem Internetbrowser und personalisierten Anwendungen im Internet, nicht jedoch mit personalisierten Diensten im Zusammenhang mit Mobiltelefonen gemacht. Die Befragung wurde online durchgeführt und die Bögen per E-Mail verschickt.

Die Auswertung der Studie hat gezeigt, dass keine der vier Thesen H1 bis H4 widerlegt wurde. Aus der Bestätigung der ersten These folgt, dass eine zu hohe Anzahl allgemeiner Angebote einen Benutzer dazu veranlassen kann, zu einem Anbieter zu wechseln, der personalisierte Dienste vertreibt. Es bedeutet zugleich, dass sich die Chancen für Firmen, ein Geschäft abzuschließen, nicht dadurch erhöhen lässt, dass man die Kunden mit mehr Werbung anspricht. Gezielt unterbreitete Angebote werden Kunden eher ansprechen als allgemein verbreitete Massenwerbung. Dieser Effekt tritt ebenfalls auf, wenn Konsumenten einen personalisierten Dienst als nützlich und hilfreich ansehen, wie zum Beispiel einen ortsabhängigen Dienst und multimediale Anwendungen. Da auch die These H4 nicht widerlegt wurde, kann man daraus folgern, dass die Sorge um die Sicherheit privater Daten einen Benutzer davon abhalten kann, einen personalisierten Dienst in Anspruch zu nehmen.

Aus der Untersuchung geht aber auch hervor, dass der Nützlichkeit eines personalisierten Dienstes ein höherer Stellenwert zugeschrieben wird als der Angst um das Ausspionieren und Analysieren persönlicher Daten. Können die Anbieter von Mobilfunkverträgen die Vorzüge personalisierter Dienste den Kunden vertrauenswürdig erklären, werden diese eher bereit sein, dem Vertreiber persönliche Daten preiszugeben.

5.8 Zusammenfassung und Ausblick

Personalisierte Anwendungen bekommen einen zunehmend wichtigeren Stellenwert in unserer digitalen, vernetzten Welt. In vielen Geschäftsbereichen sind sie bereits etabliert und kaum noch wegzudenken. Aber auch im privaten Bereich finden sie eine fortschreitende Verbreitung. Grundlagen, die eine Personalisierung erst ermöglichen, sind Technologien, wie das Anlegen von Profilen, sowohl persönlicher als auch allgemeiner Art. Diese basieren auf maschinenlesbaren Formaten wie XML und RDF, die durch Verschlüsselung und digitale Signaturen auch die Sicherheit und den Schutz von persönlichen Daten garantieren können.

Allgemeine Anwendbarkeit erhalten Personalisierte Dienste dadurch, dass sie praktisch von jedem Ort der Welt über das Internet abrufbar sind. Durch Mobilfunktechnologien wie GPRS und UMTS lassen sich auch mobile Geräte an dieses Netz anschließen, die durch den Einsatz dieser Übertragungsstandards aufgrund besserer Transferzeiten immer größer werdende Datenmengen empfangen können. Probleme bezüglich der Darstellung dieser Daten bereiten bestimmte Voraussetzungen mobiler Geräte, wie die Größe des Displays, auf dem Informationen ausgegeben werden sollen. Daher ist es notwendig, alle Kanäle, auf denen Menschen und Maschinen interagieren können, auszunutzen. Für viele Menschen führt dies zu einer Steigerung des Komforts im Umgang mit den Geräten. Personen mit Einschränkungen bezüglich ihrer Wahrnehmungsfähigkeit wird auf diese Art der Zugang zu dieser Technologie überhaupt erst ermöglicht.

Die wichtigsten Bereiche, in denen personalisierte Anwendungen eingesetzt werden, sind Web-Portale,

Finanzgeschäfte, Ticketverkauf, Shopping, Entertainment sowie ortsabhängige und personalisierte Dienste. Gerade der Handel profitiert von der Personalisierung der Anwendungen, denn auf diese Art können den Kunden Angebote bedürfnisgerecht unterbreitet werden. Der Anwender kann die Art der Werbung dadurch bestimmen, dass er Angaben zu seinen Interessen in einem persönlichen Profil ablegt. Allgemeine Werbung empfinden viele Menschen hingegen als belästigend.

Im Hinblick auf die zukünftige Entwicklung personalisierter Anwendungen lässt sich prognostizieren, dass diese eine immer wichtigere Rolle auch im privaten Bereich einnehmen werden. Dies wird sowohl mobile als auch stationäre Geräte betreffen, die miteinander vernetzt sind. In vielen Visionen wird eine Situation dargestellt, in der beispielsweise Kühlschränke feststellen, welche Produkte in ihnen gelagert werden und welche sie aufgrund des Essenswunsches ihres Besitzers automatisch nachbestellen müssen. Sie können zudem warnen, falls das Haltbarkeitsdatum eines Lebensmittels überschritten wurde. Die weiter fortschreitende Entwicklung und Standardisierung grundlegender Technologien wird Anwendungen dieser Art in absehbarer Zeit den Einzug in unseren Alltag ermöglichen.

Kapitel 6

Kontext & User Modeling

abstract Diese Arbeit soll in Vorbereitung auf die Projektgruppe *eXplorer - Kontext-sensitive Umgebungserkundung mit mobiler multimodaler Unterstützung* als Grundlage die Begriffe Kontext und kontextsensitiv zunächst allgemein erklären und gibt dazu einen Überblick darüber, wie sie in vorangegangenen Arbeiten im Bereich des kontext-sensitiven Rechnereinsatzes definiert wurde und welche Ansätze zur Nutzung von Kontextinformationen bereits verfolgt werden. Im Hinblick auf das Thema der Projektgruppe liegt hierbei besonderes Augenmerk auf der Anwendung in mobilen Geräten, insbesondere Navigationssystemen und Tourismusanwendungen. Ein großer und wichtiger Teil von Kontext ist direkt auf den Benutzer selbst bezogen. Diese Informationen werden in Benutzermodellen verwaltet. Aus diesem Grund wird zusätzlich der Begriff des User Modeling oder der Benutzermodellierung zunächst allgemein und anschließend seine Einsatzmöglichkeit im Bereich Tourismus an einem Beispiel sowie die Herausforderung bei der Modellierung visueller Einschränkungen vorgestellt.

6.1 Einleitung

Jeder kennt Navigationssysteme, wie sie im Automobilbereich eingesetzt werden. Seit einigen Jahren helfen sie Autofahrern dabei sich an ihnen unbekannten Orten zurechtzufinden. So berechnen sie Routen von A nach B, warnen vor Staus und Baustellen, bieten dann eine alternative Route an und lotsen einen am Ziel angekommen auf Wunsch noch zum nächsten Parkhaus. Mit Hilfe von Handheld PCs sind solche Systeme auch für den alltäglichen Gebrauch unabhängig vom Verkehrsmittel denkbar und teilweise sogar schon Realität. Hierfür gibt es eine Vielzahl von Aufgaben und Situationen, bei denen Menschen unterstützt werden können. Eine besondere Herausforderung ist dabei die Aufmerksamkeit des Benutzers nicht zu stark zu strapazieren. Nach [ME01] bildet nämlich gerade sie einen Flaschenhals in der Interaktion von Mensch und Maschine. Eine Möglichkeit Ablenkungen des Benutzers zu verringern ist die Berücksichtigung von Kontext. Kontextsensitive Systeme machen Annahmen über die Situation, in der sie eingesetzt werden und passen ihre Ausgaben in Form und Umfang daran an. So werden dem Benutzer im Idealfall nur die Informationen präsentiert, die für seine jeweilige Aufgabe relevant sind. In dieser Arbeit werden daher die Begriffe Kontext und Kontextsensitivität im Kapitel *Was ist Kontext?* zunächst im Abschnitt *Definitionen von Kontext und Kontextsensitivität* anhand von Definitionen und Beschreibungen aus vorangegangenen Arbeiten vorgestellt. Da für unsere Projektgruppe die Einsatzmöglichkeiten von Kontext in mobilen Geräten von besonderem Interesse sind, werden im Abschnitt *Kontextsensitive Anwendungen in mobilen Geräten* Ansätze aus anderen Projekten vorgestellt. Neben der Anpassung an den Kontext des jeweiligen Einsatzes wird auch insbesondere die Anpassung von Systemen an den jeweiligen Nutzer

seit Jahren erfolgreich eingesetzt. Hier wurde der Begriff der Benutzermodellierung (*User Modeling*) geprägt. Im zweiten Kapitel *User Modeling* werden im Abschnitt *User Modeling im Allgemeinen* generelle Aufgaben und Typisierungen von Benutzermodellen angegeben, bevor im Abschnitt *User Modeling im Bereich Tourismus* am Beispiel einer anderen Arbeit die Architektur eines User Modeling Systems im Projekt DeepMap beschrieben wird. Abschließend werden im Kapitel *Ansätze zur Modellierung visueller Einschränkungen* eben diese vorgestellt.

6.2 Was ist Kontext?

Wie schon der Name unschwer erkennen lässt, ist das erklärte Ziel unserer Projektgruppe die Entwicklung eines mobilen multimodalen Systems, welches seine Anwender kontext-sensitiv bei der Umgebungserkundung unterstützt. Eine Voraussetzung dafür ist unter anderem ein grundlegendes Verständnis der Begriffe Kontext und Kontextsensitivität. Kontext ist zunächst ein recht allgemeiner Begriff, zudem einem Synonyme wie Zusammenhang, Umgebung oder Situation einfallen. Außerdem existieren viele fachspezifische Interpretationen und Definitionen von Kontext, die sich von Domäne zu Domäne unterscheiden. Für unser Projekt ist dabei natürlich nur eine spezielle Sicht auf den Begriff von Interesse. Diese beinhaltet im weitesten Sinne Kontext im Bereich der Mensch-Computer-Interaktion (*Human-Computer-Interaction* kurz *HCI*), der mobilen Rechnernutzung (*mobile computing*) und der ubiquitären Rechenumgebungen (*ubiquitous computing*). Selbst eine solche Einschränkung des Blickes auf Kontext führt nicht zu der einen in diesem Bereich allgemein anerkannten Definition von Kontext, sondern zu einer Vielzahl von Definitionen, die zugegebenermaßen immerhin durchaus überlappen. So wird Kontext z.B. grob beschrieben als *Umgebung, in der eine Berechnung stattfindet*[ME01] oder *Menge von Fakten und Umständen, die eine Aktivität oder ein Ereignis umgeben*[AS01a]. Im folgenden Abschnitt wird daher ein Überblick über einige Definitionen von Kontext und Kontextsensitivität geboten.

6.2.1 Definitionen von Kontext und Kontextsensitivität

Viele der ersten Arbeiten zum Thema Kontext definierten diesen über Aufzählungen von Merkmalen oder Faktoren, die sie als Teile von Kontext identifizierten. So bezeichnen Schilit und Theimer Kontext als Ort, Identitäten von Personen und Objekten in naher Umgebung und die Veränderungen, die diese Objekte erfahren. Ähnlich sind auch die Definitionen von Brown und Ryan, die Kontext als Ort, Identitäten von Personen in der Nähe des Nutzers, die Tages- und Jahreszeit sowie Temperatur bzw. als Aufenthaltsort des Nutzers, dessen Umgebung und Identität sowie Zeit definieren [AKD99]. Eine Aufzählung von Dey [AKD99] enthält den emotionalen Zustand des Nutzers, dessen Aufmerksamkeit, Ort und Blickrichtung, Datum und Uhrzeit, Objekte und Personen im Umfeld. All diese Aufzählungen sind insofern interessant als dass sie verdeutlichen, dass Kontext ein zusammenfassender Begriff für viele Faktoren ist und welche das sein können. Über zwei weitere Merkmale und Möglichkeiten ihre Ausprägung zu bestimmen berichten Müller et al [CM01]. Sie untersuchen inwiefern sich Sprache nutzen lässt, um aus ihr Rückschlüsse über Zeitdruck und kognitive Belastung ziehen zu können. Ein großer Nachteil einer Definition anhand einer Aufzählung liegt in den Schwierigkeiten bei der Entscheidung, ob ein bisher nicht aufgelistetes Merkmal auch Teil von Kontext sein kann oder nicht. Aus diesem Grund definieren Dey und Abowd Kontext generell wie folgt:
Kontext ist jegliche Information, die verwendet werden kann die Situation einer Entität zu charakterisieren. Eine Entität kann dabei eine Person, ein Ort oder ein Objekt sein, welches für die Interaktion zwischen einem Benutzer und einer Anwendung als relevant erachtet wird, inklusive des Benutzers und der Anwendung selbst.

Dementsprechend wird Kontextsensitivität folgendermaßen definiert:
Ein System ist kontext-sensitiv wenn es Kontext verwendet, um dem Nutzer relevante Informationen und /oder Dienste bereitzustellen, wobei die Relevanz von der Aufgabe des Nutzers abhängt.
Diese Definitionen legen sich nicht auf einzelne Merkmale fest, sondern überlassen es den Entwicklern selbst, festzulegen welche Merkmale für ihre spezielle Anwendung wichtig sind und liefern die Richtlinien für die Entscheidung gleich mit. Ihre Allgemeinheit macht sie auch in den unterschiedlichsten Anwendungen einsetzbar. Ähnliche generelle Definitionen liefern Schmidt und Gellersen [AS01a] . Sie definieren Kontext als *abstrakte Beschreibung der Situation oder von signifikanten Merkmalen der Situation in der lokalen Umgebung des Benutzers* und ein System mit Kontextbezug als *ein System, das die Fähigkeit hat Aspekte der Umgebung als Kontext zu erfassen und diese für ein kontext-bezogenes Verhalten zu nutzen.*

Die Eigenschaft von Kontext ein abstrakter Begriff zu sein, unter dem verschiedene Merkmale zusammengefasst werden, bietet die Möglichkeit Kontext zu strukturieren. Und auch dafür gibt es unterschiedliche Ansätze. Zum einen wird Kontext in Kategorien unterteilt, zum anderen existieren Modelle, die versuchen Kontext strukturiert zu beschreiben. Schilit teilt Kontext in drei Kategorien ein: *computing context*, *user context* und *physical context*. Computing context ist dabei etwa als Rechenumgebung zu verstehen und beinhaltet Merkmale wie Anschluss zu einem Netzwerk, Kommunikationskosten, Bandbreite und nahe Ressourcen wie Drucker, Bildschirme, und Workstations. User context beinhaltet Informationen über den Benutzer wie dessen Benutzerprofil, Aufenthaltsort, Personen in der Umgebung und aktuelle soziale Situation. In der dritten Kategorie physical context werden physikalische Rahmenbedingungen zusammengefasst z.B.: Helligkeit, Geräuschpegel, Verkehrsbedingungen und Temperatur. In diese Kategorien kann der natürliche und zweifelsohne wichtige Faktor Zeit nicht eingeordnet werden. Deshalb wurden Schilits Kategorien von Chen und Kotz [CK00] um eine weitere Kategorie, dem *time context* erweitert. In dieser Kategorie werden alle Formen von Zeit zusammengefasst. Weiterhin erachten sie eine Aufzeichnung aller Kontextmerkmale über ein Zeitintervall in einer *context history* als nützlich. Mit Hilfe einer solchen history können Vorhersagen über zukünftige Situationen gemacht werden. So könnte man beispielsweise bei der Benutzung eines mobilen Navigationssystems für Fußgänger anhand der letzten Aufenthaltspunkte Annahmen über die aktuelle Blickrichtung machen. Eine weitere Kategorisierung von Kontext stammt von Dey und Abowd [AKD99]. Sie unterscheiden zwischen primärem und sekundärem Kontext. Dem primären Kontext gehören Ort, Identität, Zeit und Aktivität an. Über diese primären Kontextinformationen lassen sich dann ähnlich einer Datenbank Informationen über sekundäre Kontexte erreichen. So reicht es z.B. aus die Identität eines Benutzers zu kennen, um mit dessen Hilfe sekundäre Kontextinformationen wie Telefonnummer, Adresse oder Geburtstag zu bekommen. Voraussetzung dafür ist natürlich, dass man diese Informationen in einem Benutzerprofil hinterlegt hat.
Reichenbacher [Rei03] demonstriert den Aufbau von Kontext an einem allgemeinen Modell zur Benutzung von Kontext. Dabei orientiert auch er sich an den bereits genannten Definitionen und Beschreibungen von Kontext. Den Kern von Kontext nennt er Situation. Diese Situation entspricht der Bindung von Kontext an Ort und Zeit von Gellersen und Schmidt. Um diesen engeren Sinn von Kontext sind vier weitere Teilbereiche von Kontext angesiedelt: Benutzer, Technologie, Aktivitäten und Information.

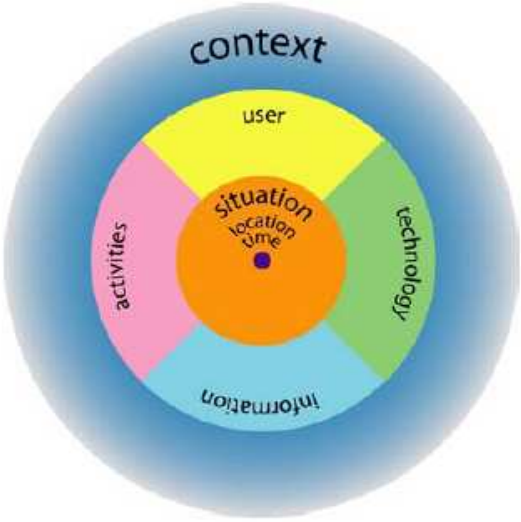


Abbildung 6.1: Kontextmodell von Reichenbacher

Bradley und Dunlop [BD03a] liefern mit ihrem Kontextmodell eine andere Darstellung von Kontext. Dabei unterscheiden sie zwischen der Welt des Benutzers und der Welt der Anwendung, einer kontextuellen Ebene und einer fokalen Ebene und zwischen bedeutenden und nebensächlichen Kontextinformationen. Die kontextuelle Ebene beinhaltet alle äußeren Umstände, die den Benutzer oder die Anwendung bei der Durchführung ihrer fokalen Aktivitäten, also der Aktivitäten, auf die sich Benutzer bzw. Anwendung momentan konzentrieren, beeinflussen. Sie ist ähnlich den zuvor beschriebenen Kategorisierungen von Kontext und den äußeren Bereichen aus Reichenbachers Kontextmodell in vier Dimensionen unterteilt. Dazu gehören hier zeitlicher, sozialer und physischer Kontext, sowie Kontext im Bezug auf die aktuelle Aufgabe. Im Hinblick auf die Ausführung der momentanen Aktionen können manche Kontextinformationen von Bedeutung sein, andere aber nicht. Insofern beinhaltet auch dieses Modell implizit die Kontextkomponente Zeit, da die Aktionen im Fokus von Nutzer und Anwendung über die Zeit variieren und somit auch die unterschiedlichen Kontextinformationen in ihrer Relevanz.

Für unser Projekt, das ja auf Mobilität ausgelegt ist, erscheint der von Wahlster [Wah01] verwendete Begriff des mobilen Kontexts als besonders interessant. Mit mobilem Kontext erweitert er die Kategorie des ortsbezogenen Kontexts um zwei Faktoren, die gerade in mobilen Anwendungen starken Variationen unterworfen sein können. Diese beiden Faktoren sind die Verfügbarkeit und die Qualität von Positionsinformationen. Unter Positionsinformationen werden hier Angaben wie aktuelle Koordinaten, Geschwindigkeit, Beschleunigung, Kopfneigung und Blickrichtung verstanden. Sind einzelne Kontextinformationen nicht oder nur in geringer Qualität vorhanden, so kann darauf unterschiedlich reagiert werden. So kann die Qualität der Informationen entweder durch Inferenz aus anderen Kontextinformationen oder durch direkte Interaktion mit dem Benutzer, also durch Erfragen, verbessert werden. Letzteres ist gerade in mobilen Anwendungen aufgrund einer Beschränkung der Kommunikationskanäle nur bedingt anwendbar. Ist eine Qualitätsverbesserung nicht möglich kann dies mit einer Veränderung der Informationspräsentation kompensiert werden. Bei der Darstellung der eigenen Position auf einer adaptiven Karte könnten ungenauere Positionsinformationen zu einer Vergrößerung der Markierung für die eigene Position führen. Eine entscheidende Frage bei der Entwicklung mobiler kontextsensitiver Anwendungen adressiert den Ort der Ermittlung von Kontextinformationen. Soll der Kontext direkt im Gerät selbst erfasst werden oder soll Kontext indirekt in einer Infrastruktur ermittelt werden, die diesen dann mobilen Geräten zur Verfügung stellt? Beide Möglichkeiten bieten dabei Vor- und Nachteile. Zwar sparen indirekte Messungen Hardwa-

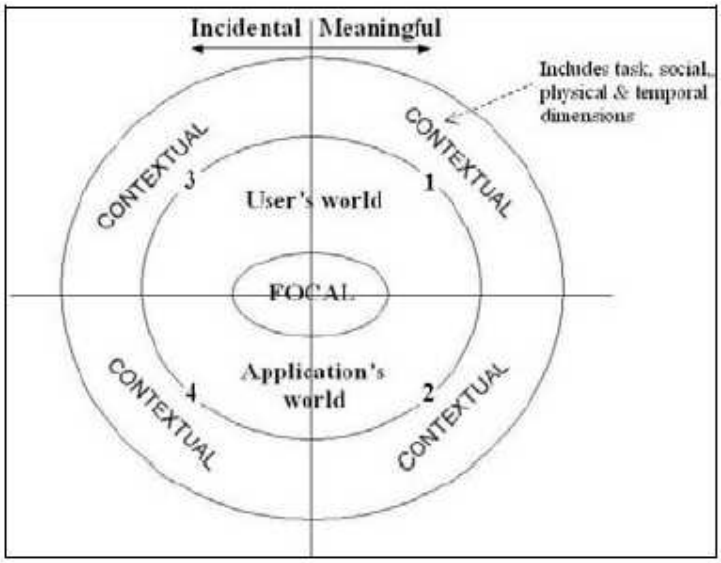


Abbildung 6.2: Kontextmodell von Bradley und Dunlop

reressourcen ein, die in mobilen Geräten wie Handheld PCs in der Regel stark limitiert sind. Zum anderen sind Verfügbarkeit und Qualität von Kontext direkt von der Infrastruktur in der jeweiligen Umgebung abhängig und führen in Regionen ohne entsprechende Infrastruktur zu gravierenden Einschränkungen im Nutzen der Anwendung. Direkte Messungen erfordern hingegen Teile der ohnehin oft knappen Hardwareressourcen, bieten im Gegenzug aber den Zugriff auf Kontextinformationen in Unabhängigkeit von vorhandener Infrastruktur.
Neben den bereits anhand einiger Definitionen beschriebenen verschiedenen Formen von Kontext unterscheiden sich auch kontextsensitive Systeme in ihrem Einsatz der Kontextinformationen. Chen und Kotz unterscheiden hier zwischen *aktiver* und *passiver* Kontextsensitivität. Als passiv kontextsensitiv werden solche Systeme bezeichnet, die Kontextinformationen erfassen und diese dem Benutzer gegebenenfalls auch anzeigen. Ein aktiv kontextsensitives System nutzt die Kontextinformationen darüber hinaus um sein Verhalten an die jeweilige Situation anzupassen. Aktiv kontextsensitive Systeme werden auch oft als *adaptiv* bezeichnet. Weitere Unterschiede identifizieren Schmidt und Gellersen in der Ebene des Kontexteinsatzes. Der Einsatz auf Systemebene ermöglicht kontextsensitives Energie- und Ressourcenmanagement, während er auf der Anwendungsebene zu adaptiven Anwendungen und kontextbasierten Diensten führt. Als dritte Ebene, in der Kontext eingesetzt werden kann, wird die Benutzungsschnittstelle genannt.

6.2.2 Kontext-sensitive Anwendungen in mobilen Geräten

Nach der Einführung in die Begriffe Kontext und Kontextsensitivität werden die Einsatzmöglichkeiten von Kontext an Ansätzen anderer Arbeiten beispielhaft vorgestellt. Zu diesem Thema gibt es bereits viele Anwendungen wie zum Beispiel die Nutzung von Positionsinformationen von Mitarbeiten in einem Bürogebäude, die mit Hilfe des Active Badge Systems gewonnen werden, zur Weiterleitung eingehender Anrufe an ein Telefon in der Nähe des gewünschten Gesprächspartners. Diese Arbeit konzentriert sich allerdings auf Anwendung, die insbesondere die Umgebungserkundung unterstützen. Solche Anwendungen beinhalten in der Regel die Anzeige adaptiver Karten. Dabei kann sich die Adaption z.B. in der Skalierung oder Drehung der Karte oder der Ein- bzw. Ausblendung bestimmter Informationen auf der Karte ausdrücken. Um diesen Anforderungen Rechnung zu tragen, werden Geoinformationen in einem

geeigneten Datenformat wie XML oder GML gespeichert, aus denen sich dann je nach Kontext dynamisch Grafiken generieren lassen. Als Datenformat für diese Grafiken bietet sich SVG an, da SVG-Grafiken frei skalierbar sind, ohne "pixelig" zu werden, sie sich problemlos aus XML oder GML (Geography Markup Language) Informationen erzeugen lassen und zudem noch wenig Speicherplatz benötigen. Reichenbacher [Rei03] beschreibt, wie verschiedene Kontextinformationen in verschiedene Schichten übertragen und wie diese dann durch Übereinanderlegen zur Gesamtkarte integriert werden können. Als erstes Beispiel wird nun das Projekt REAL [Wah01] vorgestellt. REAL ist ein ressourcenadaptierendes mobiles Navigationssystem, das im Gegensatz zu Navigationssystemen, die aus dem Automobilbereich bekannt sind, seine Dienste auf tragbaren Endgeräten zur Verfügung stellt und somit Navigationshilfen für Fußgänger bietet. Eine besondere Schwierigkeit stellt dabei die Berücksichtigung des Benutzerkontextes dar, da dieser zu Fuß viel größeren Änderungen unterliegt als im Auto. Während Autofahrer in der Regel recht ähnliche Wegbeschreibungen für das gleiche Ziel bekommen werden, sollte sich die Wegbeschreibung zum Bahnhof für einen Geschäftsmann in Eile schon von der für einen Touristen unterscheiden, der in dem Bahnhof eine weitere Sehenswürdigkeit in einer fremden Stadt sieht. Neben Informationen zum Aufenthaltsort sollen insbesondere die individuellen Ressourcen des Benutzers oder dessen Kontext in Betracht gezogen werden. So werden dann auch Faktoren wie Geschwindigkeit, Ortskenntnis und Zeitdruck berücksichtigt. Ein weiteres Problem, dem sich REAL stellt, ist seine Verfügbarkeit inner- und außerhalb von Gebäuden, da bisher keine Technologie zur Ortsbestimmung praktisch in beiden Bereichen eingesetzt werden kann. Aus diesem Grund unterteilt sich das System REAL in zwei Subkomponenten IRREAL und ARREAL. Das Outdoor-Navigationssystem ARREAL erhält seine Positionsinformationen über ein GPS-Handgerät und ermittelt daraus aktiv bzw. entsprechend der vorgestellten Bezeichnung von Gellersen direkt die dem Nutzer präsentierte Information. Diese aktive Aufbereitung der Information ist rechenintensiv und beansprucht die Ressourcen der Hardware. Für die Berechnungen dient ARREAL daher als Zentraleinheit ein Subnotebook. Weitere Komponenten sind ein magnetischer Tracker, der neben seiner Funktion als Kompass auch mit zwei Tasten bestückt ist und somit optional als 3D-Zeiger verwendet werden kann, und ein Brillendisplay, das dem Benutzer graphische und textuelle Ausgaben präsentiert. Gemäß der im vorigen Abschnitt beschriebenen Definition von mobilem Kontext enthält dieser Variationen von Verfügbarkeit und Qualität von Orts- und Orientierungsinformationen. Darauf wird mit einem Wechsel zwischen Vogel- und Egoperspektive, und einer Größenänderung der Positionsmarkierung des Benutzers reagiert. Die Darstellung in der Egoperspektive macht verständlicher Weise nur dann Sinn, wenn die vorhandenen Informationen zu Position und Blickrichtung möglichst genau sind. Auch Unterschiede in der Geschwindigkeit des Benutzers haben eine Anpassung des Systems zur Folge. So führt eine Erhöhung der Geschwindigkeit zu einer Anpassung des dargestellten Kartenausschnitts im Sinne eines ZoomOut sowie zu einer Reduzierung der Anzeige zusätzlicher Informationen. Im Gegensatz dazu verwendet IRREAL Infrarotsignale zur Positionsbestimmung, zum Einen, weil der Empfang von GPS-Signalen in Gebäuden nicht möglich ist, zum Anderen, da die Positionsinformationen so passiv bzw. indirekt ermittelt werden können. D.h. der Rechenaufwand für die Aufbereitung der Positionsinformationen wird auf die in Gebäuden angebrachte Infrastruktur abgewälzt, und das mobile Gerät kann seine Ressourcen darauf verwenden, sich dem restlichen Kontext des Nutzers entsprechend zu verhalten. Für diese Anwendungen wurden speziell Infrarotsender entwickelt, die eine Signalübertragung von bis zu 20 Meter, statt üblicher 2 Meter ermöglichen. Besonders interessant ist hier die Art der Übermittlung der Kontextinformationen von der Infrastruktur zum Endgerät und das besondere Format der Daten. Die Daten unterscheiden sich von Sender zu Sender und werden zyklisch gesendet. IRREAL benötigt keinen Rückkanal vom Endgerät zum Sender. Das Besondere an den gesendeten Daten ist, dass diese aus interaktiven Texten und Grafiken in Form eines Präsentationsgraphen bestehen. Eine Interaktion zwischen Benutzer und Sender ist aufgrund des fehlenden Rückkanals nicht möglich. Dafür interagiert er mit der Präsentation. Wichtige-

re Knoten wie z.B. die Wurzel eines Graphen werden öfter gesendet, als andere deren Aufruf weniger wahrscheinlich ist, um deren Verfügbarkeit zu erhöhen. Auf diese Weise lässt sich der Kontextfaktor Geschwindigkeit des Benutzers elegant integrieren. Annahmen über die Geschwindigkeit werden über die Aufenthaltsdauer im Sendebereich eines Senders gemacht. Diese führt dann zur Anpassung des Systems an die Geschwindigkeit, indem sie den Detaillierungsgrad der Anzeige bestimmt. So werden nach längerem Aufenthalt weitere Informationen eingeblendet, die aufgrund ihrer niedrigeren Wichtigkeit als der Wurzelknoten auch seltener und somit in der Regel auch erst später im Hintergrund übertragen werden. So wird nach Betreten eines neuen Sendebereichs beispielsweise zunächst nur ein großer Pfeil angezeigt, der in die Richtung eines zuvor angegebenen Ziels weist. Nach einiger Aufenthaltsdauer im gleichen Sendebereich werden weitere Knoten des Präsentationsgraphen übertragen. Die Verfügbarkeit weiterer Informationen äußert sich dem Benutzer im Erscheinen eines kleinen Hilfeknopfes neben dem Pfeil. Wird dieser ausgewählt, so werden weitere Informationen über die nähere Umgebung angezeigt, in die nach weiterem Warten zusätzlich abrufbare Informationen eingeblendet werden (siehe Abbildung 6.3).

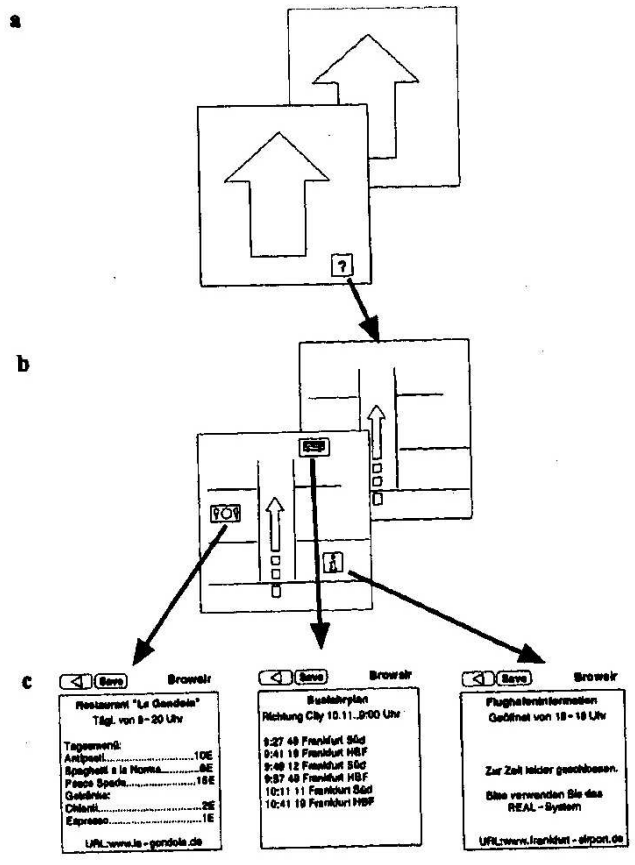


Abbildung 6.3: Abbildung eines IRREAL Präsentationsgraphen

6.3 User Modeling

Wie bereits zuvor dargestellt, beinhaltet die Berücksichtigung von Kontext in großem Maße die Situation des jeweiligen Benutzers. Allerdings hat man sich bei der Entwicklung von Anwendungen schon früh unabhängig von Arbeiten im Bereich Kontext um die Berücksichtigung unterschiedlicher Aspekte des je-

weiligen Benutzers bemüht. Traditionell war Software entweder von aktiver oder passiver Benutzerführung geprägt. *Besonders wünschenswert sind jedoch Kommunikationskonzepte, bei denen sowohl Mensch als auch Computer aktive Rollen übernehmen können.* [Bod92] Voraussetzung für eine interaktive Kooperation von Mensch und Computer ist neben dem bereits genutzten expliziten Kommunikationskanal, über den Eingaben vom Benutzer und Ausgaben vom Computer ausgetauscht werden zusätzlich implizite Vorstellungen bezüglich des Gegenübers. So hat der Benutzer ein mentales Modell vom System, in dem Vorstellungen darüber enthalten sind, wie sich das System wohl verhalten wird, wenn er eine bestimmte Aktion ausführt. Diese Annahmen entsprechen nicht immer dem tatsächlichen Systemverhalten. Ziel ist es, dass sich Systeme wirklich so verhalten, wie es der Benutzer vermutet. Da sich Benutzer im Umgang mit ein und derselben Anwendung beispielsweise aufgrund unterschiedlicher Interessen oder Wissensständen durchaus unterscheiden, wird dem System zur Erfüllung des Ziels ein Benutzermodell oder User Model zur Verfügung gestellt, das es ihm ermöglichen soll, sich der Erwartung des jeweiligen Benutzers entsprechend zu verhalten. Dieses Kapitel beschreibt zunächst im folgenden Abschnitt Benutzermodelle im Allgemeinen und anschließend im Abschnitt User Modeling im Bereich Tourismus Ansätze zum Einsatz von Benutzermodellen in Tourismusanwendungen.

6.3.1 User Modeling im Allgemeinen

Wie bereits einleitend erwähnt, ist das Ziel der Benutzermodellierung ein möglichst genaues Bild vom Benutzer zu bekommen, um darauf basierend den Dialog zwischen Mensch und Computer anzupassen.

Als inhaltliche Teilaufgaben werden *Diagnostizieren*, *Vorhersagen*, *Gestalten*, *Vermitteln*, *Korrigieren*und *Bewerten* genannt. [Bod92] Diagnostizieren bedeutet hier Annahmen über Wissen und Können bzw. Unwissen und Nicht-Können zu machen. Mit Vorhersagen ist die Gewinnung von Informationen bezüglich der Ziele, zukünftiger Aktionen und Reaktionen des Benutzers gemeint. Gestalten bezieht sich auf die Art der Dialogführung bzw. der Präsentationsstrategie. Vermittelnd wirkt ein Benutzermodell, wenn es die Auswahl einzelner Dialogschritte sowie die Angabe von Hilfs- und Zusatzinformationen für den jeweiligen Benutzer beeinflusst. Die Verbesserung und Erweiterung fehlerhafter bzw. unvollständiger Eingaben fallen unter die Teilaufgabe des Korrigierens. Bewerten meint Annahmen über allgemeine Stärken und Schwächen des Benutzers zu machen. Je nach Verwendungszweck der Inhalte eines Benutzermodells lassen sich 3 verschiedene Ziele beschreiben. Dienen die Inhalte der expliziten Konfiguration der Anwendung nach den Wünschen des Benutzers, so spricht man von Benutzerkonventionen. Werden sie stattdessen für die Modellierung der Vorkenntnisse des Benutzers eingesetzt, erhält man ein Modell der Benutzerkompetenz. Sollen die Inhalte den Benutzer beim Erreichen seiner Ziele unterstützen, dienen sie der Beschreibung der Benutzerintentionen.

Benutzermodelle können in ihren Eigenschaften große Unterschiede aufweisen. Eine der ersten Klassifizierungen von Benutzermodellen anhand dieser Merkmale geht auf *Rich* [Ric81] zurück. Sie macht ihre Einteilung anhand dreier Merkmale fest. So unterscheiden sich Benutzermodelle darin, ob sie individuelle Benutzer modellieren oder Benutzer Gruppen oder Stereotypen zuordnen, für die Modelle existieren. Ein weiterer Unterschied liegt in der Art der Erfassung von Benutzerinformationen. Wird das Modell implizit durch Inferenz aus dem Benutzerverhalten mit Informationen gefüllt oder explizit entweder durch den Entwickler oder Benutzer selbst mit Informationen versorgt? Einfacher wäre natürlich die explizite Angabe von Informationen. Allerdings können nicht alle Angaben vom Benutzer gemacht werden und in der Regel will man auch nicht ständig mit Aufforderungen zur Eingabe von Benutzerinformationen belästigt werden, die mit der eigentlichen momentanen Aufgabe nichts direkt zu tun haben. Ein weiteres Merkmal ist der Zeitbezug des Modells, d.h. werden die Informationen langfristig oder kurzfristig gespeichert und verwendet? Mittlerweile wurde diese Klassifizierung um einige Merkmale erweitert. Neben den

Unterscheidungen kanonisch gegenüber individuell, explizit gegenüber implizit und langfristig gegenüber kurzfristig werden nun auch Merkmalsausprägungen wie transparent gegenüber intransparent, dynamisch gegenüber statisch und beschreibend gegenüber vorhersagend zur Klassifizierung von Benutzermodellen herangezogen. Ein Modell ist transparent wenn es die Informationen dem Benutzer zugänglich macht oder sogar eine Bearbeitung der Inhalte zulässt. Im Gegensatz dazu informieren intransparente Modelle den Benutzer nicht über die zu ihm gehörenden Informationen. Ein Benutzermodell heißt statisch wenn sich seine Inhalte nicht während einer Dialogsitzung verändern lassen. Dynamisch ist es hingegen wenn es seine Inhalte zur Laufzeit aktualisiert, egal ob durch das System oder explizit durch den Benutzer. Ein vorhersagendes Modell versucht, die Ziele und Pläne des Benutzers in Erfahrung zu bringen, um ihn bei der Erreichung der Ziele zu unterstützen. Werden die Informationen lediglich zur Anpassung der Darstellung genutzt, ist das Modell hingegen beschreibend.

Die möglichen Inhalte, die erfasst und in Benutzermodellen abgelegt werden können, reichen von "harten" Informationen zu groben Einschätzungen. Mögliche Inhalte von „harten" Informationen zu groben Einschätzungen (siehe Abbildung 4) . Deskriptive Informationen zu Person und Dialoganwendung lassen sich sehr leicht erfassen. Eine Betrachtung und statistische Auswertung dieser Informationen über einen längeren Zeitraum ermöglicht die Gewinnung von Informationen zum Dialogverhalten wie zum Beispiel welche Funktionen wie oft aufgerufen werden, welche Fehler gemacht werden aber auch wie auf angebotene Hilfen und Zusatzinformationen reagiert wird. Aus diesen können wiederum Annahmen über die Präferenzen und Einstellungen des Benutzers gemacht werden. Je mehr Informationen gesammelt werden desto komplexere Schlüsse können über den Benutzer gezogen werden etwa über Systemwissen, Anwendungswissen, Ziele und Pläne sowie kognitive Prozesse. Jedoch nimmt die Sicherheit solcher inferierten groben Einschätzungen stark ab. So können verschiedene beobachtete Inhalte unterschiedliche Gründe haben.

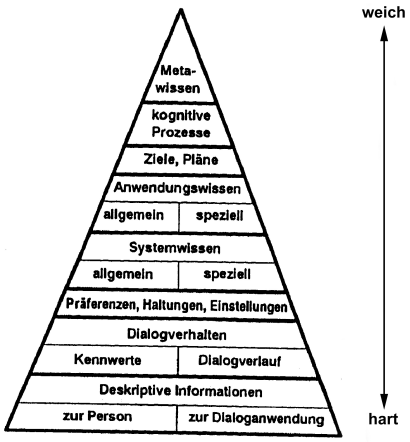


Abbildung 6.4: Mögliche Inhalte eines Benutzermodells

Benutzermodelle werden in User Modeling Systems (UMS) verwaltet. Da solche Systeme in der Regel unabhängig von den Anwendungen sind, die von den Benutzerinformationen profitieren, werden sie auch User Modeling Server genannt. An UMS werden unterschiedliche Anforderungen gestellt. Sie sollten für möglichst viele Anwendungen nutzbar sein, die Möglichkeit haben fehlende Informationen aus existierenden zu inferieren und die dadurch bedingte Unsicherheit einzelner Informationen zu berücksichtigen, die Möglichkeit fehlende Informationen aus Profilen ähnlicher Benutzer zu ergänzen, Benutzerinformationen aus externen Datenquellen wie etwa Datenbanken importieren können, sowie die Privatsphäre der Benutzer wahren. Weitere Anforderungen sind die Fähigkeit zur schnellen Anpassung, Erweiterbarkeit, Lastausgleich, Ausfallstrategien und die Gewährleistung von Konsistenz der Inhalte [Kob00] .

6.3.2 User Modeling im Bereich Tourismus

Um nach der allgemeinen Beschreibung von Benutzermodellen wieder den Bezug zur computergestützten Umgebungserkundung herzustellen, wird in diesem Abschnitt ein Ansatze zum Einsatz von Benutzermodellen in mobilen Tourismus behandelt. Bei dem beschriebenen Projekt handelt es sich um DeepMap[AK02]. Im Rahmen des Projekts DeepMap wurde zum Einsatz der Benutzermodellierung eigens ein User Modeling System (UMS) entwickelt. Hier werden die dabei ermittelten Anforderungen an ein UMS in einer Tourismusanwendung sowie im Anschluss die Architektur des entwickelten UMS vorgestellt.

Tourismus ist von Natur aus stark an persönliche Interessen und Präferenzen gebunden, weshalb sich viele der in diesem Gebiet entwickelten Systeme personalisiert verhalten. Ein Projekt, welches dafür Benutzermodelle einsetzt, ist das Deep Map Projekt des European Media Laboratory in Heidelberg. Das Subprojekt WebGuide erstellt personalisierte Besichtigungsrouten für die Stadt Heidelberg und verbindet dabei geographische Points Of Interest zu Strecken, die für den jeweiligen Benutzer als interessant erachtet werden. Dafür werden folgende Informationen in Betracht gezogen wie etwa Geographische Informationen über Heidelberg, Informationen über Points Of Interest, Informationen über gewählte Verkehrsmittel, Interessen und Präferenzen des jeweiligen Benutzers sowie Einschränkungen durch den Benutzer z.B. in Form von Dauer und Länge einer Strecke.

Vor der Entwicklung des UMS stand eine Analyse der Anforderungen an ein UMS im Bereich Tourismus. Die zentralen Aufgaben decken sich teilweise mit den Beschreibungen von Benutzermodellen im Allgemeinen und allgemeinen Anforderungen an UMS im vorigen Abschnitt. Hierzu zählen die Ermittlung von Interessen und Präferenzen aus dem Dialogverhalten, die Vorhersage von Informationen bezüglich individueller Benutzer aus Informationen ähnlicher Benutzer oder durch die Zuordnung zu Benutzergruppen, Speichern, Aktualisieren und Löschen explizit angegebener Informationen sowie impliziter Annahmen, die Gewährleistung von Konsistenz und Privatsphäre bei der Haltung von Informationen und die Möglichkeit aktuelle Benutzerinformationen autorisierten Anwendungen zur Verfügung zu stellen.

Das in DeepMap entwickelte UMS speichert Benutzerinformationen im Gegensatz zu anderen existierenden Systemen nicht in Wissensrepräsentations- oder Datenbankmanagementsystemen, sondern in einem Verzeichnismanagementsystem oder kurz Verzeichnis. Grob gesehen besteht das UMS aus einer Verzeichniskomponente und den Komponenten zur Benutzermodellierung. Die Verzeichniskomponente besteht aus drei Subsystemen, die (a) für die Repräsentation der gespeicherten Inhalte, (b) die Kommunikation mit externen Clients, denen Benutzerinformationen zur Verfügung gestellt werden und (c) die Vermittlung zwischen den Benutzermodellierungskomponenten zuständig sind (siehe Abbildung 5). Die Aufgaben der Benutzermodellierung übernehmen drei Modellierungskomponenten. Die User Learning Component (ULC), die Mentor Learning Component (MLC) und die Domain Inference Control (DIC). Die ULC lernt aus dem Dialogverhalten die Interessen und Präferenzen der Benutzer und aktualisiert Benutzermodelle. Fehlende Werte werden durch die MLC aus Einträgen ähnlicher Benutzer vorhergesagt. Abstraktere Annahmen über Benutzer werden in der DIC aus Informationen inferiert, die entweder explizit vom Benutzer angegeben bzw. durch ULC oder MLC generiert wurden. Die gespeicherten Inhalte werden in vier verschiedenen Modellen repräsentiert. In User Models werden Informationen zu einzelnen Benutzern wie dessen Interessen abgelegt. Usage Models wird das Dialogverhalten der Benutzer gespeichert und über dessen Identität mit dem zugehörigen Benutzer verknüpft. Das System Model bezieht sich auf Domäne der Anwendung und hat Auswirkungen auf alle User Models. Hier wird festgelegt, welche Informationen überhaupt berücksichtigt werden können. Praktisch bestimmen sie welche Keys der Key-Value-Paare überhaupt in den User Models auftauchen und somit auch mit Values belegt werden können. Service Models beinhalten Beschreibungen von server-internen Ereignissen, an denen User Modeling Komponenten

interessiert sein können. Durch Auswahl von Service Models in anderen Komponenten kann so gesteuert werden über welche Ereignisse diese informiert werden. Das Kommunikationssystem besteht aus drei Schnittstellen zu verschiedenen Kommunikationsstandards. Darunter fallen eine FIPA-Schnittstelle, eine LDAP-Schnittstelle und eine ODBC-Schnittstelle. Die Vermittlung zwischen den einzelnen Benutzermodellierungskomponenten übernimmt ein Scheduler.

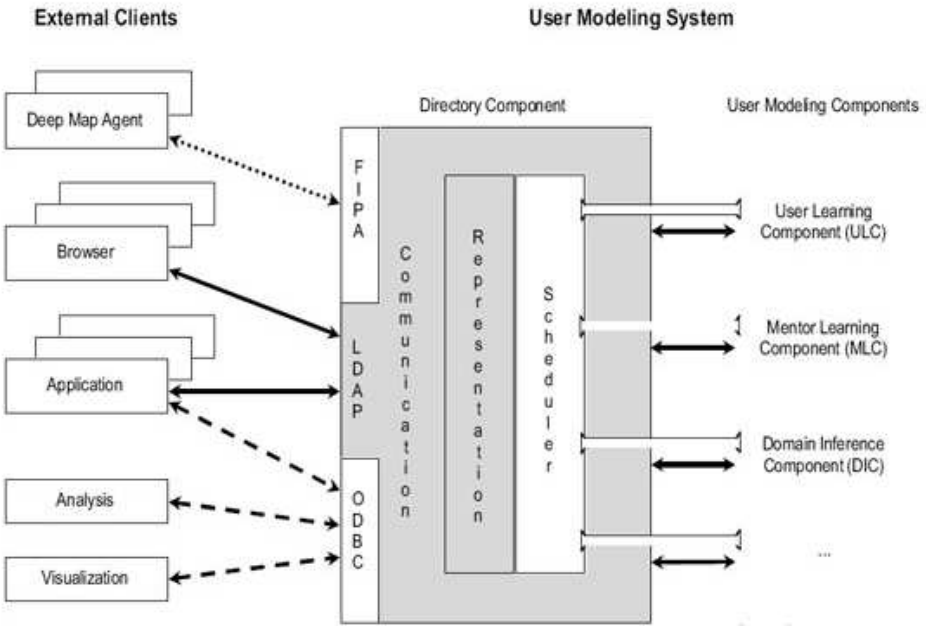


Abbildung 6.5: Architektur des UMS in DeepMap

6.4 Ansätze zur Modellierung visueller Einschränkungen

Heutzutage können Computer Menschen bei einer Vielzahl von Aufgaben unterstützen. Ein Großteil der Informationen, die Computer Menschen dabei liefern, ist visueller Natur. Eine besondere Herausforderung liegt darin, nützliche Informationen auch Menschen mit visuellen Einschränkungen zugänglich zu machen. Bei weniger gravierenden visuellen Einschränkungen, lassen sich visuelle Ausgaben noch wie etwa im AVANTI-Browser in Größe und Kontrast anpassen, um die Einschränkungen zu kompensieren. Zusätzlich lassen sich dort nahezu alle Informationen auch textuell ausgeben, so dass diese über Braille-Tastaturen lesbar oder über Software akustisch zugänglich gemacht werden können. Diese Anpassungen funktionieren zu Hause am eigenen PC oder vielleicht auch an speziellen stationären interaktiven Informationsständen. In mobilen Anwendungen ist ihr Nutzen gerade bei schweren visuellen Einschränkung eher gering. In diesem Abschnitt werden Untersuchungen beschrieben, die sich damit befassen, wie Menschen mit visuellen Einschränkungen mit ihrer Umgebung interagieren, wenn sie zu Fuß unterwegs sind. Bei der Anpassung eines Navigationssystems an blinde oder schwach sehende Benutzer sollten diese Aspekte berücksichtigt werden.

In bekannten Umgebungen kann man sich auch mit visuellen Einschränkungen leicht mit Blindenstock oder Blindenhund orientieren. Allerdings unterstützen diese nur die Orientierung im direkten Umfeld, sind also für die Mikronavigation hilfreich, nicht aber bei der Navigation über weite Entfernungen, der Makronavigation. Um sehbehinderten Menschen in unbekannten oder dynamischen Umgebungen zu helfen, wurden für die Mikronavigation bereits Electronic Travel Aids wie Geräte, die Hindernisse anhand

von Laser oder Ultraschall erkennen, entwickelt. Systeme wie GPS und GIS, die der Makronavigation dienen, haben ein großes Potential auch sehbehinderte Menschen zu unterstützen. Ein Problem liegt dabei in der Frage welche kontextuellen Informationen für Menschen mit visuellen Einschränkungen besonders wichtig sind. In einem Ansatz diese Frage zu klären, baten Bradley und Dunlop [BD04] sehbehinderte und nicht sehbehinderte Menschen in einem Interview unter anderem eine Wegbeschreibung für andere Sehbehinderte anzugeben. Als Ergebnis zeigte sich, dass Sehbehinderte bei Wegbeschreibungen andere Informationen als andere Menschen bevorzugen. Menschen mit visuellen Einschränkungen nutzen demnach im Gegensatz zu Menschen ohne visuelle Einschränkungen vor allem auch kontextuelle Informationen im Bezug auf (a) Sinne wie riechen, hören und fühlen, (b) Bewegungen wie vorbeifahrende Autos oder öffnende Türen und (c) sozialen Kontakt. Die anscheinend offensichtliche Lösung durch Verlagerung der Kommunikation zwischen Mensch und Computer von manueller Eingabe und Visueller Ausgabe hin zu auditiver Ein- und Ausgabe wird durch die starke Auslastung dieses Kanals bei der Orientierung Sehbehinderter in Frage gestellt.

In einer weiteren Untersuchung unterteilen Bradley und Dunlop [BD03b] visuelle Einschränkungen in drei Gruppen: Verlust der peripheren Sicht, Verlust der zentralen Sicht und komplette Blindheit. Auch zwischen diesen Gruppen unterscheiden sich Informationen in ihrer Relevanz. Ein Unterschied, den hierbei alle Gruppen gemein haben, ist die Orientierung drinnen und draußen. Innerhalb von Gebäuden erscheinen manche Informationen interessanter als außerhalb und umgekehrt. Die weitere Arbeit von Bradley und Dunlop sieht ein Experiment vor, in dem Menschen mit visuellen Einschränkungen GPS-unterstützte IPAQs als Navigationshilfe bekommen. Dies soll Aufschluss darüber bringen, inwieweit sich die Navigation mit Hilfe von Informationen, die Menschen mit gleicher visueller Einschränkung bereit gestellt haben, bezüglich Effizienz und Effektivität verbessern lässt.

6.5 Abschließende Anmerkungen

In dieser Arbeit wurden die Begriffe Kontext und Kontextsensitivität im allgemeinen und im Bezug auf ihre Bedeutung für mobile Anwedungen vorgestellt. Als besonderen Aspekt von Kontext wurde der Benutzer selbst herausgestellt. Dazu wurde der Begriff der Benutzermodellierung ebenfalls zunächst im Allgemeinen und anschließend in mobilen Anwendungen bzw. in Tourismusanwedungen dargestellt.

Die angegeben Beispiele zeigen dabei nur einen Ausschnitt der vielen Ansätze, Anwendung kontextsensitiv einzusetzen. Als besondere Herausforderung kontextsensitiven Rechnereinsatzes wurde die Unterstützung visuell eingeschränkter Benutzer beschrieben.

Kapitel 7

Evaluation

Abstract Die Einhaltung von Richtlinien und Kriterien bezüglich der Benutzbarkeit (*usability*) eines Softwaresystems spielen in der heutigen Softwareentwicklung eine wichtige Rolle. Die Bewertung der *usability* eines Softwaresystems ist Aufgabe einer **Evaluation**.
Man unterscheidet zwischen drei Evalutionsformen. Der eher theoretischen formalen Evaluation, der auf Erfahrung beruhenden heuristischen Evaluation und der in der Praxis vielfach eingesetzten empirischen Evaluation.
Während die formale Evaluation eine objektive Bewertung des Softwaresystems vornimmt, spielen bei der heuristischen und empirischen Evaluation auch subjektive Einschätzungen der Gebrauchstauglichkeit des Systems eine Rolle.
Für die Evaluation mobiler und multimodaler Systeme existieren bis heute noch keine allgemein anerkannten Methoden. Erste Vorschläge für solche Methoden sind u.a. das von Beringer vorgestellte *PROMISE* framework.

7.1 Einleitung

Die Entwicklung eines Softwaresystems endet i.d.R. mit der Bewertung des erstellten Produkts. Hierbei wird das System hinsichtlich seiner Gebrauchstauglichkeit, in der Literatur als Usability bezeichnet, überprüft.
Ziel einer solchen Überprüfung ist es, herauszufinden, ob das System den Ansprüchen des späteren Anwenders genügt. Dieser muss seine Aufgaben mit dem neuen System effizient, d.h. mit einem der Aufgabe angemessenem Aufwand und effektiv, also vollständig und korrekt bearbeiten können.
Von heutigen Softwaresystemen erwartet man nicht nur, dass dem Anwender die nötige Funktionalität zur Bearbeitung seiner Aufgaben zur Verfügung gestellt wird, sondern auch, dass sie für die Interaktion mit dem Anwender eine Benutzungsoberfläche bieten, die software-ergonomischen Gesichtspunkten genügt. Die Einhaltung dieser Forderungen zu überprüfen und das Softwaresystem daraufhin zu bewerten, ist Aufgabe einer **Evaluation**.
Ich werde im folgenden eine Einführung in das Thema Evaluation geben. Dabei werde ich zunächst einige Grundbegriffe klären. In Kapitel 7.2 stelle ich einige wichtige Evaluationsformen vor gebe Kriterien an, die bei der Durchführung einer Evaluation generell zu beachten sind. In Kapitel 7.3 gehe ich auf verschiedene Evaluationsmethoden ein und gebe einige Beispiele an, wie diese Methoden in der Praxis eingesetzt werden können. Anschließend gehe ich in Kapitel 7.4 auf die Evaluation mobiler und multimodaler Anwendungen und fasse in Kapitel 7.5 die Ergebnisse dieser Arbeit kurz zusammen.

7.2 Grundlagen

In diesem Kapitel werden einige grundlegende Aspekte der Evaluation von Softwaresystemen angesprochen. Zunächst werde ich neben der Definition des Evaluationsbegriffs selbst auch Angaben bezüglich Aufbau und Ablauf einer Evaluation machen. Danach stelle ich verschiedene Evaluationsformen vor und gebe am Ende dieses Kapitels eine Auflistung von Kriterien und Richtlinien an, die bei der Bewertung eines Softwaresystems zu beachten sind.

7.2.1 Definition des Begriffs Evaluation

Eine ausführliche Definition des Begriffs Evaluation findet sich in [Bro97]. Evaluation wird dort wie folgt beschrieben:

> *Analyse und Bewertung eines Sachverhalts, v.a. als Begleitforschung einer Innovation. In diesem Fall ist E. Effiziens- und Erfolgskontrolle zum Zweck der Überprüfung der Eignung eines in Erprobung befindl. Modells. E. wird auch auf die Planung angewendet, zum Zweck der Beurteilung der Stringenz der Zielvorstellung und der zu deren Verwirklichung beabsichtigten Maßnahmen.*

Nach dieser Definition dienen die Ergebnisse einer Evaluation dem Zweck, einen bestimmten Sachverhalt auf mögliche Fehler und die Einhaltung von Zielvorgaben hin zu überprüfen und Unstimmigkeiten aufzuzeigen. In Bezug auf die Bewertung von interaktiven Computersystemen spricht Preece [Pre94] von Evaluation als:

> *a process through which information about the usability of a system is gathered in order to improve the system or to assess a completed interface*

Hier wird deutlich, dass eine Evaluation nicht um ihrer selbst willen durchgeführt wird, sondern ihre Ergebnisse dazu benutzt werden sollen, das untersuchte System zu verbessern. Weitere Definitionen des Begriffs Evaluation finden sich unter [EVA04].

7.2.2 Aufbau und Ablauf einer Evaluation

Um ein sinnvolle Evaluation durchführen zu können, sind einige Kriterien zu beachten. Jeder Evaluation sollte eine Planungsphase vorausgehen, in welcher der Aufbau, die Durchführung und die Art der Auswertung der erhaltenden Daten festgelegt wird.
Die Planungsphase sollte mit der Formulierung von Kriterien beginnen, nach denen das System evaluiert werden soll. Bei der Bewertung eines Systems muss dabei berücksichtigt werden, dass die Einhaltung einiger Kriterien wichtiger ist, als die anderer Kriterien. So ist die Möglichkeit, in einem Textverarbeitungsprogramm ein Dokument zu speichern sicherlich wichtiger, als die Möglichkeit die Schriftfarbe zu wechseln. In der Bewertung der Einhaltung der Kriterien sollte sich dies wiederspiegeln.
Als nächstes müssen u.a. noch Fragen bezüglich des zeitlichen Ablaufs der Evaluation und der an ihr beteiligten Personen geklärt werden. Die Planung der Evaluationsdurchführung selbst besteht dann aus der Bestimmung der Evaluationsmethoden die angewendet werden sollen. Danach muss noch Klarheit darüber geschaffen werden, wie die erhaltenen Daten ausgewertet werden sollen.
Zum Ablauf einer Evaluation weist Klante [Kla99] auf Baumgartner [Bau97] hin. Dieser schlägt einen vierstufigen Prozess für den Ablauf einer Evaluation vor:

- *Formulierung von Wertkriterien: Kriterien, die das Produkt erfüllen muss um als wertvoll, gut etc. gelten zu können, werden ausgewählt und definiert.*
- *Formulierung von Leistungsstandards: Für die oben eingeführten Kriterien muss jeweils eine Norm gesetzt werden, die das Produkt mindestens erreichen muss (Operationalisierung).*
- *Messung und Vergleich (Analyse): Das Produkt wird unter Anwendung der Kriterien untersucht.*
- *Werturteil (Synthese): Schließlich werden die Ergebnisse zu einem einheitlichen Werturteil verknüpft.*

Nach Klante [Kla99] sollten im folgenden Schritt die Ergebnisse der Evaluation dazu benutzt werden, Systemverbesserungen durchzuführen.

7.2.3 Evaluationsformen

Unter dem Begriff Evaluationsform fasst man die Eigenschaften einer Evaluation zusammen, die u.a. Auskunft über den Detaillierungsgrad und den zeitlichen Ablauf der Evaluation geben. Die folgende Evaluationsformen werden in diesem Abschnitt vorgestellt:

- formative Evaluation
- summative Evaluation
- innere Evaluation
- äußere Evaluation
- Mikroevaluation
- Makroevaluation

Zu den bekanntesten Evaluationsformen zählen die in Abbildung 7.1 dargestellte formative und summative Evaluation.
Die **formative Evaluation** wird während des gesamten Entwicklungsprozesses durchgeführt. Hierbei wird das Produkt in Form von Prototypen nach bestimmten Gesichtspunkten untersucht. Die Auswertung der Ergebnisse führt zu Verbesserungsvorschlägen, deren Umsetzung verbesserte Prototypen zur Folge hat. Diese Form der Evaluation bietet für die Entwickler große Vorteile, da sie während des gesamten Entwicklungsprozesses Rückmeldungen über die software-ergonomische Qualität des Produkts erhalten und so bereits frühzeitig Probleme der Benutzbarkeit des entwickelten Systems erkannt und Verbesserungen durchgeführt werden können. Projektbegleitende Evaluationen erfordern auf der anderen Seite einen hohen Zeitaufwand und führen somit zu erhöhten Kosten, worin ein Nachteil der formativen Evaluation zu sehen ist.
Wird eine Evaluation erst am Ende eines Projekts am fertigen Produkt durchgeführt, spricht man von einer **summativen Evaluation**. Die Auswertung der hier erzielten Ergebnisse kann im Gegensatz zur formativen Evaluation i.d.R. nichts mehr zur Qualitätsverbesserung des Produkts beitragen, da die Implementierungsphase zum Zeitpunkt der Evaluation meist abgeschlossen ist. Der Vorteil einer summativen Evaluation liegt darin, dass nicht nur einzelne Aspekte des Produkts, sondern sowohl das gesamte Produkt als auch der komplette Realisierungsprozess bewertet wird. Die Erkenntnisse, die aus einer summativen

Evaluation gewonnen werden, können dann als Entscheidungshilfen in zukünftigen Projekten Verwendung finden.
Weitere Evaluationsformen sind die innere und äußere Evaluation. An einer **inneren Evaluation** nehmen nur Personen teil, die direkt oder indirekt am Entwicklungsprozess des Systems beteiligt sind. Dadurch fallen neben der Suche nach geeigneten Testpersonen zwar auch zusätzliche Kosten weg, allerdings haben diese Tests den Nachteil, dass das System nicht aus Sicht des Anwenders untersucht wird. Probleme, welche die eigentliche Zielgruppe später mit dem System haben könnte, bleiben so meist unentdeckt.
Eine Alternative stellt die **äußere Evaluation** dar, bei welcher das System entweder von den späteren Anwendern selbst getestet wird oder von Personen, die über ähnliche Kenntnisse und Fähigkeiten verfügen.
Bei der **Mikroevaluation** wird nur ein bestimmter Aspekt des Gesamtsystems betrachtet. Klante [Kla99] weist darauf hin, dass hierzu eventuell eine so genannte *Miniwelt* geschaffen werden muss, in der sich die zu bewertende Komponente unabhängig vom Rest des Systems bedienen lässt. Demgegenüber steht die **Makroevaluation**, bei der das Zusammenspiel der einzelnen Komponenten des Gesamtsystems bewertet werden soll. Bei komplexeren Systemen kann sich das allerdings als schwierig erweisen.
Mit Ausnahme der summativen Evaluation, die pinzipiell nur für die Bewertung des fertigen Systems verwendet wird, können alle anderen hier vorgestellten Evaluationsformen zu jedem Zeitpunkt des Entwicklungsprozesses eingesetzt werden. Bedingung hierfür ist allerdings ein existierender Prototyp des zu bewertenden Systems. Existiert dieser noch nicht als lauffähigen Programm, sondern nur aus einigen Skizzen, ist eine erste Bewertung des Systems zwar möglich, diese kann aber nur von Experten, i.d.R. sind dies die Entwickler selbst, durchgeführt werden.

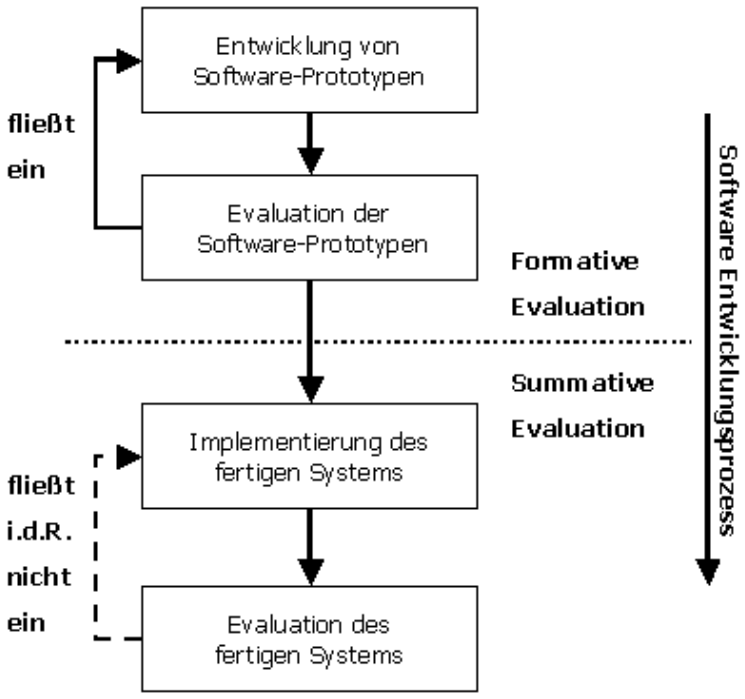


Abbildung 7.1: Formative und summative Evaluation (Quelle: [DE998])

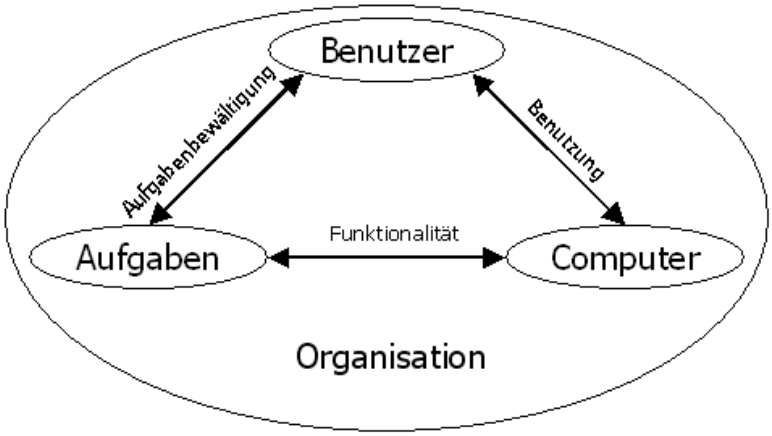


Abbildung 7.2: Einflussfaktoren einer Evaluation (Quelle: [Rei94])

7.2.4 Kriterien und Richtlinien

Bei der Evaluation eines Softwaresystems soll v.a. bewertet werden, inwieweit der Anwender bei der Bewältigung seiner Aufgaben vom System unterstützt wird. Welche Faktoren dabei eine Rolle spielen ist in Abbildung 7.2 dargestellt. Zu beachten ist, dass nicht nur der Anwender oder die Software alleine, sondern das komplette Umfeld, d.h. die zu bearbeitenden Aufgaben und die spätere Arbeitsumgebung, man spricht hierbei von *Ganzheitlichkeit des Systems*, bei der Bewertung eine Rolle spielen.
Für Oppermann und Reiterer [Rei94] stehen bei der software-ergononischen Evaluation die Eigenschaften der Mensch-Computer-Schnittstelle im Vordergrund. Bei der Bewertung dieser Schnittstelle müssen folgende Fragen beantwortet werden:

- Wie gut wird der Benutzer bei der Bewältigung seiner Aufgaben vom Softwaresystem unterstützt?
- Entsteht ein größerer Zeit- und Arbeitsaufwand, wenn der Benutzer seine Aufgaben mit Hilfe des Softwaresystems bewältigt?
- Welchen Lernaufwand muss der Benutzer betreiben, um den Umgang mit dem Softwaresystem zu erlernen?
- Kann der Benutzer das System individuell an seine Bedürfnisse / seinen Arbeitsstil anpassen oder ist ein bestimmtes Vorgehen vom System vorgeschrieben?
- Bietet das Softwaresystem dem Anwender ausreichende Funktionalitäten zur Bewältigung seiner Aufgaben?
- Ist das System flexibel genug um auf neue Aufgabenstellungen reagieren zu können?

Nach welchen Kriterien ein Softwaresystem hinsichtlich der Gestaltung der Benutzungsoberfläche bei einer Evaluation bewertet werden sollte, kann nach Oppermann und Reiterer der ISO 9241 Teil 10 und der DIN 66234 Teil 8 entnommen werden. Beide Normen enthalten Kriterien, die bei der Erstellung einer Benutzungsoberfläche beachtet werden sollten. In der ISO-Norm werden diese Kriterien als *dialogue principles* bezeichnet. Die Kriterien der beiden Normen sind in Tabelle 7.1 aufgelistet und werden im folgenden kurz beschrieben.

- Aufgabenangemessenheit / Suitability for the task: Der Anwender wird vom Dialog darin unterstützt, seine Aufgaben effektiv und effizient zu erledigen.

- Selbstbeschreibungsfähigkeit / Self-descriptiveness: Sämtliche Dialogschritte müssen für den Anwender leicht verständlich sein und ihm muss auf Verlangen Einsatzzweck und Leistungsumfang des Dialogsystems erläutert werden können.
- Steuerbarkeit / Controllability: Von einem steuerbaren Dialog spricht man, wenn der Anwender die Geschwindigkeit des Ablaufs, sowie die Auswahl und Reihenfolge von Arbeitsmitteln oder Art und Umfang von Ein- und Ausgabe beeinflussen kann.
- Erwartungkonformität / Conformity with user expections: Durch Erfahrungen, die der Anwender z.B. durch Schulungen, Nutzung von Online Tutorials oder durch das Studiums des Benutzerhandbuchs mit dem Dialogsystem gemacht hat, entstehen Erwartungen wie ein Dialog in einer bestimmten Situation auszusehen hat. Ein Dialog wird als erwartungskonform bezeichnet, wenn er diesen Erwartungen entspricht.
- Fehlerrobustheit / Error tolerance: Ein Dialog wird als fehlerrobust eingestuft, wenn er in der Lage ist, fehlerhafter Eingaben des Anwenders zu erkennen und auf diese zu reagieren. Dies kann entweder dadurch geschehen, dass ein Fehler vom Dialog selbst oder durch erneute Eingabe des Benutzers behoben wird. Dazu ist es notwendig, dass dem Anwender der Fehler angezeigt wird.
- Suitability of individualisation (Individualisierbarkeit): Dialogsysteme mit der Fähigkeit zur Individualisierung können vom Anwender an seine individuellen Bedürfnisse und Fähigkeiten angepasst werden.
- Suitability for Learning (Erlernbarkeit): Hierbei handelt es sich um die Fähigkeit eines Dialogsystems, den Anwender strukturiert durch den Lernprozess zu führen. Dabei ist es wichtig, dass die einzelnen Schritte des Lernprozesses für den Anwender überschaubar bleiben und auf Konsistenzerhaltung geachtet wird. Dadurch wird die Zeit, die der Anwender zum Erlernen des Systems benötigt, gering gehalten.

ISO-Norm 9241 Teil 10	DIN 66234 Teil 8
Suitability for the task	Aufgabenangemessenheit
Self-descriptivness	Selbstbeschreibungsfähigkeit
Controllability	Steuerbarkeit
Conformity with user expectations	Erwartungskonformität
Error tolerance	Fehlerrobustheit
Suitability of individualization	
Suitability for learning	

Tabelle 7.1: ISO-Norm 9241 Teil 10 und DIN 66234 Teil 8 (Quelle: [Rei94])

Weitere Kriterien, nach denen ein Softwaresystem bewertet werden kann, finden sich nach Oppermann und Reiterer in den Teilen 12-16 der ISO 9241, die sich mit folgenden Anforderungen an eine Benutzungsoberfläche beschäftigen:

- Teil 12 (Presentation of information.): Forderungen bezüglich der Darstellung von Informationen
- Teil 13 (User guidance): Benutzerführung
- Teil 14 (Menu dialogues): Anforderungen an Menüs, wie z.B. Pop-Up oder Pull-Down Menüs
- Teil 15 (Command dialogues): Kriterien für den Einsatz eines Kommandozeilen-basierten Softwaresystems

- Teil 16 (Direct manipulation dialogues): Anforderungen an die direkte Manipulation von Objekten auf dem Bildschirm

Eine ausführliche Beschreibung der ISO 9241 ist unter [ISO04] zu finden.

7.3 Evaluationsmethoden

In diesem Kapitel stelle ich einige Evaluationsmethoden vor. Zunächst gehe ich auf einen eher theoretischen Ansatz, der formalen Evaluation ein. Die Funktionsweise dieser Methode beschreibe ich am so genannten GOMS-Modell. Hiernach gebe ich einen Einblick in die heuristische Evaluation, auch Expertenbegutachtung genannt und komme dann zur empirischen Evaluation, die bei der heutigen Softwareentwicklung eine wichtige Rolle spielt.
Zum Abschluss gehe ich darauf ein, worauf bei der Umsetzung einer kostengünstigen Evaluation zu achten ist.

7.3.1 Formale Evaluation

Die formative Evaluation stellt eine theoretische Vorgehensweise der Bewertung eines interaktiven Softwaresystems dar. Hierzu werden Modelle des Systems und nicht das System selbst analysiert. Eines der bekanntesten Modelle, die für die formative Evaluation benutzt wird, ist das von Card [Car83] vorgestellte **GOMS-Modell**.
Oppermann und Reiterer [Rei94] liefern eine gute Beschreibung dieses Modells, welches aus folgenden Elementen besteht:

- **G**oals (Ziele)
- **O**perators (Operatoren)
- **M**ethods (Methoden)
- **S**election rules (Regeln für die Selektion)

Mittels dieser vier Elemente kann die Bedienbarkeit eines interaktiven Systems bewertet werden.
Durch die **Ziele** werden Zustände beschrieben, die der Anwender erreichen will. Ein solcher Zustand kann z.B. darin bestehen, die Schriftgröße eines Wortes zu verändern. Viele Ziele können wiederum in Teilziele aufgeteilt werden. So muss für obiges Beispiel zunächst das betreffende Wort markiert und anschließend die gewünschte Schriftgröße angegeben werden.
Zu den **Operatoren** zählen die verschiedenen elementaren Interaktionstechniken, die dem Anwender vom System zur Verfügung gestellt werden. Hierzu gehören z.B. Bewegungen, die mit der Maus ausführbar sind oder die Betätigung von Maus oder Tastaturelementen.
Folgen von nacheinander anzuwendenden Operatoren zum Erreichen eines Ziels, werden als **Methoden** bezeichnet. Da oft mehrere Methoden eingesetzt werden können, um ein Ziel zu erreichen, werden zur Auswahl einer Methode **Selektionsregeln** eingesetzt.
Das soeben beschriebene GOMS-Modell dient der Analyse der zu erwartenden Effizienz von geübten Benutzern eines Systems. Oppermann und Reiterer weisen darauf hin, dass für eine solche Analyse die Zeiten festgelegt werden müssen, die für die Anwendung der verschiedenen Bestandteile des Modells zu erwarteten sind. Hierzu muss u.a. untersucht werden, wie viel Zeit der Anwender für die Betätigung einer bestimmten Taste benötigt. Abhängig ist diese Zeit von verschieden Faktoren, wie z.B. der Erfahrung des

Anwenders mit dem System. Solche Untersuchungen werden *Keystroke* Analysen genannt.
Oppermann und Reiterer geben ein Beispiel an, wie anhand der Analyse eines Systems mit dem GOMS-Modell unterschiedliche Varianten der Realisierung einer Aktion miteinander verglichen und quantitative Aussagen darüber getroffen werden können, wie viel besser eine Variante im Vergleich zu einer anderen ist. Das Beispiel kann in [Rei94] eingesehen werden.
Der Nachteil einer formalen Evaluation, wie sie mittels des GOMS-Modells durchgeführt werden kann, liegt in dem hohen zu betreibenden Aufwand und in der begrenzten Aussagekraft der Ergebnisse.

7.3.2 Heuristiche Evaluation

Unter einer *Heuristik* versteht man das methodische Vorgehen zur Gewinnung neuer Erkenntnisse, die durch Erfahrung gewonnen werden. Die Orientierung an Faustregeln während des Lernprozesses ist hierfür eine wichtige Vorraussetzung.
Die **heuristische Evaluation** kann während verschiedener Phasen des Softwareentwicklungsprozesses eingesetzt werden und wird grundsätzlich nur von Experten durchgeführt, weshalb man auch von einer Expertenbegutachtung (Preim in [Pre99b]) spricht.
Man unterscheidet zwischen kalter, warmer und heißer Evaluation. Von einer **kalten Evaluation** spricht man, wenn lediglich eine schriftliche Spezifikation vorliegt. Die **warme Evaluation** wird an ersten Prototypen, die bereits einen gewissen Grad an Funktionalität des späteren Gesamtsystems besitzen vorgenommen. **Heiße Evaluationen** werden am nahezu vollendeten Produkt gegen Ende der Entwicklungsphase durchgeführt. Es stellt sich nun die Frage nach einer möglichen Strategie für die Durchführung einer heuristischen Evaluation. Preim verweist hierzu auf eine von Thimbleby [Thi90] genannte Vorgehensweise.
Hierbei soll sich der Experte bei der Bewertung des Systems stets die Frage stellen, ob ein Computer anhand der Dialoge und Bildschirmausgaben in der Lage wäre zu entscheiden, wie er zu reagieren, resp. welche Schritte er als nächstes durchzuführen hätte. Durch dieses Vorgehen können die Stellen im System identifiziert werden, an denen der nächste Schritt unklar ist, man also raten muss, auf welchen Annahmen und Konventionen das System aufbaut und welche Probleme bei der Benutzung auftreten können. Unter Konvention sind die Voraussetzungen zu verstehen, die an den Anwender gestellt werden um das System benutzen zu können.
Ein weiterer Punkt, der bei der Bewertung eines Systems nach Thimbleby eine Rolle spielt, ist die Bewertung der Informationsaufbereitung. Es soll bewertet werden, ob für den Benutzer die Möglichkeit besteht, die Informationen in einem, seinen individuellen Bedürfnissen angepassten Umfang zu erhalten.
Eine weitere Strategie für die systematische Untersuchung der Benutzungsoberfläche durch Experten sieht Preim in der so genannten *Usability Inspection*. Die Usability Inspection dient der Lokalisierung, Klassifizierung und Priorisierung von Problemen in der Benutzbarkeit einer Benutzungsoberfläche. Schwerwiegende Probleme werden höher priorisiert als andere Probleme. Der Behebung höher priorisierter Probleme wird dann besondere Aufmerksamkeit gewidmet.
Dies ist ein Vorteil gegenüber **empirischen Evaluationsmethoden**, die in Kapitel 7.3.3 vorgestellt werden. Zu den weiteren Aufgaben einer Usability Inspection gehört es, Vorschläge zur Beseitigung der erkannten Probleme zu liefern, sowie eine Abschätzung hinsichtlich der Folgen (Kosten).
Eine Usability Inspection beginnt mit einer Vorbereitungsphase, während der u.a. geklärt werden muss, wie viele Personen an der Evaluation teilnehmen und um wen es sich dabei handelt. Generell sind dies Experten auf dem Gebiet der Mensch-Computer-Interaktion (MCI).
Anschließend werden Schwerpunkte festgelegt, nach denen das System bewertet werden soll, da eine umfassende Untersuchung eines Systems in den meisten Fällen zu aufwendig ist. Die Durchführung einer Usability Inspection kann entweder aus einem einmaligen Durchlauf bestehen, bei dem eine *Checkliste*

abgearbeitet wird oder durch einen so genannten *Walkthrough*. Bei einem *Walkthrough* werden mehrer Testläufe durchgeführt, wobei in jedem Testlauf ein anderer Aspekt untersucht wird.
Die Auswertung einer Usability Inspection besteht aus einer Diskussion der Experten, bei der jeder die von ihm erkannten Probleme vorstellt. Es ist sehr wichtig, dass kein Problem weggelassen und die Projektleitung über sämtliche erkannten Probleme informiert wird. Voraussetzung für den Einsatz einer Usability Inspection ist ein lauffähiger Prototyp. Im Hinblick auf die MCI-Experten, die das System evaluieren ist es wichtig, dass sie nicht zu den Entwicklern des Projekts gehören. Es ist allerdings nützlich wenn man Entwickler aus anderen Projekten zu den Experten hinzuzieht, um diesen die Sichtweise eines MCI-Experten vor Augen zu führen. Dies kann sich auf künftige Entwicklungen positiv auswirken.

7.3.3 Empirische Evaluation

Die **empirische Evaluation** wird eingesetzt, um Erfahrenswerte im Umgang mit der entwickelten Software zu erhalten. Anders als bei der heuristischen Evaluation, setzt die empirische Evaluation meist erst zu einem Zeitpunkt der Entwicklungsphase ein, an welchem eine erste Auswertung und Problembeseitigung erfolgt ist.
Ziel einer empirischen Evaluation ist es, den Gesamteindruck, den der Benutzer vom System hat, zu bewerten. Vor allen bei größeren Systemen wird hierbei oftmals ein Augenmerk auf funktionale Teilbereiche, wie z.B. Piktogramme, Terminologie der Bedienelemente, Verständlichkeit der Systemnachrichten, die Dokumentation, die Installationsprozedur etc. gelegt. Somit kann die empirische Evaluation dazu benutzt werden, zwischen mehreren Design-Varianten zu entscheiden.
Nach Preim [Pre99b] sind folgende Voraussetzungen und Gegenstände für eine empirische Evaluation notwendig:

- Es muss ein Prototyp vorhanden sein
- Ein Evaluationsteam muss benannt werden. Im Idealfall besteht diese Team nicht nur aus Entwicklern des System, damit hinsichtlich der Auswertung der Testergebnisse eine gewisse Neutralität vorhanden ist.
- Eine feste Anzahl von Testpersonen (Probanden) muss benannt und ein Zeitplan organisiert werden. Bei der Planung muss die Suche nach geeigneten Testkandidaten berücksichtigt werden.
- Der Ort, an dem die Evaluation durchgeführt werden soll, muss festgelegt werden. Eine Evaluation kann sowohl im Labor des Entwicklers, als auch im Büro des Anwenders stattfinden.
- Der letzte Punkt betrifft die Klärung der Kriterien, nach denen das System evaluiert werden soll. Testkriterien können der Lernaufwand, die Dauer der Durchführung wichtiger Aufgaben, die Fehlerrate, das Behalten bestimmter Funktionalitäten oder der persönlicher Eindruck der Testperson sein.

Die Kriterien, nach denen ein System bei einer empirischen Evaluation bewertet werden sollte, sind nach Preim in objektive und subjektive Kriterien zu unterteilen. Zu den **objektiven Kriterien** zählen die Bewertung der Effizienz der Interaktion, das Auftreten von Fehlern sowie die Erlernbarkeit des Systems. Die Bewertung der Effizienz soll Aufschluss darüber geben, ob der Anwender seine Aufgaben mit der vom System bereitgestellten Funktionalität in einer vorgegebenen Zeit und mit einem angemessenem Aufwand erledigen kann. Weiterhin ist es den Entwicklern wichtig festzustellen, an welcher Stelle des Systems die Anwender Fehler bei der Bedienung machen, wie sie darauf reagieren (geäußerte Kommentare) und wie schnell sie in der Lage sind, ihre Fehler zu korrigieren.

Durch eine solche Fehleranalyse erhofft man sich Aufschluss darüber, wie lange der spätere Anwender benötigt, um den korrekten Umgang mit dem System zu erlernen und wie oft und welche Hilfsmittel er hierzu studiert.
Die quantitativen Ergebnisse der Bewertung objektiver Kriterien geben sowohl Auskunft über, hinsichtlich ihrer Bedienbarkeit noch zu verbessernder Systemkomponenten, als auch über solche Komponenten, die vom Benutzer nicht wahrgenommen und daher ignoriert werden.
Die oben bereits erwähnten **subjektiven Bewertungskriterien** liefern qualitative Ergebnisse bezüglich der Zufriedenheit des Anwenders mit dem System. Sie dienen dazu, die eventuell schlechten quantitativen Ergebnisse relativieren zu können.
Die Durchführung einer subjektiven Bewertung kann entweder durch eine gezielte Befragung der Testperson (*Interview*) oder durch das Ausfüllen eines speziellen Fragebogens geschehen. Welche Arten der Befragung anwendbar sind, wird weiter unten beschrieben.
Etwas schwieriger gestaltet sich die Bewertung subjektiver Kriterien mittels *Fragebogen*. Shneiderman [Shn98] schlägt hierfür die Verwendung generischer Fragebögen vor. Jede Frage bietet der Testperson neun Antwortmöglichkeiten. Dadurch wird auf der einen Seite genügend Flexibilität für seine Einschätzung des Systems geboten und die Testperson auf der anderen Seite nicht durch zu viele Optionen verwirrt.
Weitere qualitative Aspekten, die nach Preim vom Benutzer bewertet werden sollten, sind die Angemessenheit der verwendeten Terminologie, der bildhaften Komponenten der Benutzungsschnittstelle und eine Einschätzung des Layouts. Des weiteren hält Preim es für sinnvoll, Fragen zu stellen, durch deren Beantwortung die Testpersonen ausdrücken können, ob sie gewisse Dinge als schwierig oder naheliegend empfunden haben.
Dies hängt u.a. von persönlichen Daten wie z.B. Alter, Geschlecht, kognitiven Fähigkeiten, oder Erfahrung mit ähnlichen Systemen ab. Es ist nach Preim daher notwendig, alle relevanten persönlichen Daten der Testpersonen zu ermitteln. Im von Shneiderman vorgestellten generischen Fragebogen werden diese Daten zu Beginn abgefragt.
Wie oben bereits erwähnt wurde, ist neben dem Fragebogen auch ein strukturiertes Interview eine adäquate Möglichkeit, um die subjektiven Eindrücke der Testperson in Erfahrung zu bringen. Die Ergebnisse können dann statistisch ausgewertet werden. Ein Problem sieht Preim in der zeitlichen Verschiebung zwischen Test und Interview. So kann es vorkommen, dass sich die Testpersonen zum Zeitpunkt des Interviews einiger aufgetretener Probleme nicht mehr bewusst sind. Preim gibt in [Pre99b] drei Alternativen/Ergänzungen zum klassischen Interview an:

- Die Tagebuchmethode, bei der Eindrücke, Probleme und Verbesserungsvorschläge zu jedem beliebigen Zeitpunkt mit Datum versehen notiert werden können.
- Gruppeninterviews, die in Form einer moderierte Diskussion abgehalten werden.
- Die Beobachtung des Probanden. Dies kann entweder direkt oder durch Auswertung eines Videomitschnitts geschehen.

Alle diese Methoden ermöglichen eine Analyse des Systems, wie es in der späteren realen Umgebung verwendet wird. Da die Testergebnisse aber oft nicht sehr exakt und nachvollziehbar sind, wird eine empirische Evalaution häufig in so genannten *Usability*-Laboren durchgeführt.
Das sich die Ergebnisse, die eine Evaluation in der realen Umgebung liefert, meist nur schwer oder gar nicht nachzuvollziehen lassen liegt darin, dass die Testperson hier vielen äußeren Einflüsssen ausgesetzt ist. Diese treten in zufälligen Abständen auf und lenken die Testperson von seiner eigentlichen Aufgabe ab.
In einem Usability-Labor ist es möglich, eine *sterile Atmosphäre* zu schaffen, damit die Testperson dem zu

bewertendem System ihre volle Aufmerksamkeit schenken kann. Die Ergebnisse, die so gewonnen werden, lassen jedoch nur Rückschlüsse darauf zu, ob das System generell von der späteren Anwendergruppe zu bedienen ist und sagt nichts darüber aus, wie tauglich es zum Einsatz in einer realen Umgebung ist.
Ein Usability Labor, so wie es von Preim in [Pre99b] beschrieben wird, besteht in der Regel aus drei schalldichten Räumen. In einem Raum sitzt der Proband und versucht eine vordefinierte Aufgabe mit der zu testenden Software zu lösen. Dabei wird er optimaler Weise von einer oder mehreren Kameras gefilmt. Des weiteren können sich Mikrofone im Raum befinden, um eventuelle Äußerungen des Probanden aufzunehmen.
Im zweiten Raum sitzen die Tester, die den Probanden wahlweise durch einen Einwegspiegel oder über Monitore, die mit den Kameras verbunden sind beobachten. Die Durchführung des Tests kann über einen dritten Raum von den Entwicklern oder den Auftraggebern beobachtet werden.
Nützliche Informationen liefert das *laute Denken* der Probanden, das aus Äußerungen über die augenblickliche Tätigkeit bzw. die geplante weitere Vorgehensweise bestehen kann. Weiterhin kann es hilfreich sein, bestimmte Tätigkeiten am Computer wie z.B. Arbeits-/Leerlaufzeiten oder Eingabegeschwindigkeiten zu messen.
Der Ablauf eines Tests im Usability-Labor sieht so aus, dass dem Probanden nach einigen Erläuterungen eine festgelegte Zeit zur Einarbeitung in des System und zur Akklimatisierung gegeben wird. Anschließend bearbeitet der Proband seine Aufgaben, i.d.R. zweimal, damit ein eventueller Lernerfolg gemessen werden kann. Optimaler Weise wird die Beobachtung des Probanden mit dem Ausfüllen eines Fragebogens und einem anschließendes Interview kombiniert.
Die statistische Auswertung der so gewonnenen Daten führt zu verbesserten Prototypen, die ihrerseits wieder untersucht werden Das Ziel besteht darin, nach einer überschaubaren Anzahl an Durchläufen eine Programmversion präsentieren zu können, die sowohl nach subjektiven, als auch nach objektiven Kriterien bewertet als gebrauchstauglich erscheint.
Eine weitere Möglichkeit der Durchführung einer empirischen Evaluation besteht per Systemtest über das Internet. Hiezu werden meist Fragebögen ausgefüllt und die Benutzung des Systems durch den Probanden in so genannten Log-Files gespeichert, die dann zur Auswertung an die Tester geschickt werden. Ein solcher Test ermöglicht bei Einsatz einer Webcam und einer ausreichenden Bandbreite die zusätzliche Auswertung von Gestik und Mimik des Probanden während des Tests. Diese Form der Evaluation hat sich allerdings noch nicht durchgesetzt.
Zur Auswertung der quantitativen und qualitativen Daten ist nach Preim folgendes zu sagen. Die statistische Auswertung quantitative Daten ist leichter als die von Meinungen und Kommentaren. Hierzu werden Mittelwerte und Standardabweichungen ausgerechnet und grafisch dargestellt. Äußerungen und Meinungen hingegen müssen zunächst gesichtet und in Kategorien eingeteilt werden. Diese Ergebnisse müssen dann während der Analyse mit den persönlichen Daten der Probanden in Relation gebracht werden, bevor die Auswertung der Ergebnisse erfolgen kann. Ich werde hier nicht weiter auf das Thema der statistischen Auswertung eingehen und verweise auf [Pre99b].

7.3.4 Discount Usability Engineering

Viele der bisher vorgestellten Evaluationsverfahren erscheinen sehr Zeit- und Kostenintensiv. Dies ist auch eines der Hauptargumente, weshalb viele Unternehmen von einer umfangreichen Gebrauchstauglichkeitsprüfung absehen. Hierbei wird übersehen, dass die Einhaltung von Usability-Kriterien heute ein wichtiges Verkaufsargument ist und im Hinblick auf das immense Angebot an Konkurrenzprodukten unverzichtbar ist.
Auf welche Aspekte zu achten ist, um ein System ausreichend auf seine Gebrauchstauglichkeit zu prüfen

und die Entwicklungskosten dennoch nicht zu sehr ansteigen zu lassen, wird im Folgenden beschrieben und kann in [Pre99b] nachgelesen werden:

- Testziel: Sorgfältige Auswahl der wichtigsten zu testende Aspekte
- Aufwand der Prototyperstellung: Verwendung von schnell zu erstellenden Prototypen und geeigneten Zeichen- und Präsentationsprogrammen
- Testumgebung: Falls keine sorgfältige Analyse der Videoaufzeichnungen durchgeführt werden soll, reicht es aus den Benutzer zu beobachten und seine Handlungen und Äußerungen handschriftlich zu protokollieren.
- Anzahl der Testpersonen: Viele Tests sind bereits mit einer geringen Anzahl von Testpersonen durchzuführen. Das Testergebnis ändert sich dann durch Hinzunahme weiterer Testpersonen nur noch unwesentlich.
- Auswahl der Testpersonen: Testpersonen sollten nicht an der Entwicklung des Systems beteiligt sein und ähnliche Vorkenntnisse wie der spätere Anwender besitzen.

7.4 Evaluation mobiler und multimodaler Anwendungen

Die bisher vorgestellten Evaluationsverfahren haben sich bei der Bewertung von Softwaresystemen bewährt und ihr Einsatz ist in der heutigen Softwareentwicklung unverzichtbar. Durch die Entwicklung mobiler und multimodaler Systeme haben sich nun allerdings die Anforderungen geändert, die an ein Evaluationsverfahren gestellt werden.
Multimodale Systeme bieten, im Gegensatz zu herkömmlichen Systemen, ein breites Spektrum an Interaktionsmöglichkeiten und somit dem Austausch von Informationen zwischen Mensch und System an. Bestand für den Benutzer bislang lediglich die Möglichkeit des Informationsaustauschs über Eingabegeräte wie Maus, Tastatur oder Touchscreen und waren Ausgaben i.d.R. nur über Monitor oder Drucker möglich, sind heutige Systeme bereits technisch in der Lage, sowohl verbale Äußerungen, als auch Gestik und Mimik des Benutzers zu analysieren und darauf zu reagieren. Auch die Möglichkeiten des Systems, Informationen an den Benutzer weiterzuleiten sind vielfältiger geworden. Des weiteren spielt bei der Entwicklung neuer Systeme die Standortunabhängigkeit, kurz Mobilität, eine wichtige Rolle. Dieser Aspekt muss bei einer Evaluation ebenfalls berücksichtigt werden.
Zur Zeit gibt es noch keine allgemein anerkannte Vorgehensweise für die Evaluation mobiler und multimodaler Systeme. Ich werde im folgenden einige Ansätze aufzeigen, wie eine solche Evaluation aussehen könnte und welche Aspekt dabei zu berücksichtigen sind.
Wie die Evaluation eines multimodalen Systems vorgenommen werden kann, wird von Beringer [Ber02] beschrieben. Beringer stellt die Methode *PROMISE* (Procedure for Multimodal Interactive System Evaluation) vor, einer mögliche Erweiterung des *PARADISE* Framework.
PARADISE dient der Evaluation von Dialogsystemen, die auf Spracheingaben reagieren, ist aber nicht geeignet zur Evaluation multimodaler Systeme. Ein wesentliches Problem bei der Evaluation multimodaler Systeme sieht Beringer darin, die unterschiedlichen Komponenten für Input und Output zu bewerten. Multimodale Systeme bieten dem Benutzer mehrere Möglichkeiten des Informationsaustauschs an. So ist die Eingabe von Instruktionen durch den Benutzer, z.B. durch verbale Äußerungen oder bestimmte Gestiken denkbar. Ein Problem stellt dann die Auswertung dieser Instruktionen dar. Prinzipiell gibt es drei Möglichkeiten, wie das System die erhaltenen Informationen verarbeiten kann:

- Es werden nur die Information der Komponente weiterverarbeitet, die ihre empfangenen Daten am schnellsten an das System weiterleitet.
- Es findet ein Informationsabgleich unter den einzelnen Komponenten statt, wobei nur solche Daten weitergeleitet werden, die von allen Komponenten empfangen wurde
- Sämtliche, von den multimodalen Komponenten empfangenen Informationen werden zur weiteren Verarbeitung an das System geschickt und entsprechend ihrer Fehlerwahrscheinlichkeit bewertet, da nicht jede Komponente Informationen mit gleicher Wahrscheinlichkeit fehlerfrei empfängt.

Die letztgenannte Möglichkeit wird von Beringer als die sinnvollste angesehen. Bei der Evaluation eines multimodalen Systems spielt die Gewichtung der einzelnen Komponenten ebenfalls eine wichtige Rolle. Komplexere Komponenten, d.h. Komponenten die hohe Kosten verursachen um Informationen fehlerfrei zu empfangen, sollten höher bewertet werden, als weniger komplexe Komponenten. Des weiteren sollte bewertet werden, wie zufrieden der Benutzer mit der Komponente ist. Diese Zufriedenheit muss mit den Kosten der Komponente in Relation gesetzt werden und in die Gesamtbewertung des Systems mit einfließen.
Für die Evaluation eines mobilen Systems gibt es ebenso, wie für die Evaluation eines multimodalen Systems bisher noch keine allgemein anerkannte Vorgehensweise. Bei der Bewertung eines mobilen Systems ist insbesondere die sich ständig ändernde Umgebung, in der das System benutzt wird, zu berücksichtigen. Die Aufmerksamkeit, die der Benutzer dem System widmet, kann jederzeit durch störende Einflüsse der Umgebung abgelenkt werden. Ein mobiles Systems sollte u.a. also danach bewertet werden, in welchem Umfang sich ändernde Umwelteinflüsse negativ auf die Aufmerksamkeit, die der Benutzer dem System widmet, auswirken.
Die Schwierigkeit besteht hierbei darin, dass es äußere Einflüssen gibt, welche die volle Konzentration des Benutzers verlangen, wie z.B. eine Durchsage am Bahnhof. Andere Einflüsse wiederum können vom Benutzer ignoriert werden. Änderungen der Umwelt des Benutzers haben möglicherweise Einfluss auf seine Wahrnehmung, seine kognitive Verarbeitung und seinen physischen Umgang mit dem System. Bei einer Evaluation sollte bewerten werden, wie das System auf veränderte Umweltbedingungen reagiert und sich den perzeptuellen und kognitiven Fähigkeiten des Benutzers entsprechend anpasst.
Ein solches System ist nur schwer zu Realisieren, da eine Einschätzung der Aufmerksamkeit des Benutzers durch das System stark fehlerbehaftet ist. Es sollte daher dem Benutzer selbst überlassen werden das System bei sich ändernden Umweltbedingungen seinen Bedürfnissen entsprechend anzupassen. Inwieweit dies vom System unterstützt wird, sollte dann im Rahmen einer Evaluation bewertet werden.

7.5 Zusammenfassung

Mit dieser Abeit habe ich einen kurzen Einblick gegeben, was unter dem Begriff Evaluation zu verstehen ist. Ich habe aufgezeigt, wie der generelle Aufbau einer Evaluation aussieht, welche unterschiedlichen Evaluationsformen es gibt und welche Methoden zur Durchführung einer Evaluation angewendet werden können.
Es sollte deutlich geworden sein, wie wichtig es heutzutage bei der Entwicklung eines Softwaresystems ist, dieses vor der Auslieferung auf seine Gebrauchstauglichkeit hin zu überprüfen, um auf dem dem Markt bestehen zu können.
Die in dieser Arbeit vorgestellten Methoden für die Durchführung einer Evaluation unterscheiden sich sowohl in ihrem Aufwand, als auch in den Kosten, die durch sie verursacht werden.
Wie ich aber in Kapitel 7.4 bereits beschrieben habe, gibt es durchaus Möglichkeiten ein Softwaresystem

hinreichend gut auf seine Gebrauchstauglichkeit zu überprüfen, ohne dass dadurch unüberschaubar hohe Kosten entstehen.
Handlungsbedarf besteht noch bei der Entwicklung von Evaluationsmethoden, die eine Bewertung mobiler und multimodaler Systeme hinsichtlich ihrer Gebrauchstauglichkeit ermöglichen.

Kapitel 8

Mobile Anwendungen

abstract Es gibt eine Vielzahl von mobilen Anwendungen in den unterschiedlichsten Anwendungsgebieten. Durch Endgeräte, die einfach zu transpotieren sind, wie z.B. ein Handy oder ein PDA (Personal Digital Assistant), kann man sie an fast jedem erdenklichen Ort nutzen. Die Anwendungen dienen zum Spielen, zur Lokalisierung und damit zu einem ortsbezogenen Informationdienst oder einfach zum Übertragen von Daten. Dieses Dokument soll einen Überblick darüber geben, wie mobile Anwendungen funktionieren, welche Techniken gebraucht werden und wer mobile Anwendungen überhaupt nutzen kann.

8.1 Einleitung

Was sind eigentlich mobile Anwendungen?
Unter mobilen Anwendungen versteht man alles, was man unterwegs mit Hilfe von elektronischen Medien erledigen kann. Hiermit sind sowohl Dinge im privaten Bereich gemeint, wie die Verabredung zum Kinobesuch per SMS, als auch Anwendungen im Beruf, wie z.B. ein Navigationssystem für die Feuerwehr.
Mobile Anwendungen bieten also mobile Lösungen für mobile Tätigkeiten. Mit mobilen Anwendungen sind nicht unbedingt Anwendungen gemeint, die auf mobilen Endgeräten laufen, sondern es sind damit Anwendungen gemeint, die in der Bewegung ausgeführt werden, die an ständig wechselden Einsatzorten stattfinden und deren Aufmerksamkeit in der realen Welt liegt.
Die Nutzung von mobilen Anwendungen in der Zukunft setzt einige Hardwarekomponenten voraus, die entweder schon heute existieren, oder deren Entwicklung in naher Zukunft angekündigt ist, hierzu gehören:

- Wearable Computer
- Ausgabemedien, z.B. monokulare Head-Mounted Displays
- Eingabemedien, z.B. Unterarmtastaturen

Die Nutzung von mobilen Anwendungen ist privat mittlerweile zur festen Gewohnheit geworden. Es kann sich kaum noch jemand vorstellen, nicht immer erreichbar zu sein, so besitzen 80% der Deutschen mittlerweile ein Handy.
Allerdings werden mobile Anwendungen in der Wirtschaft noch nicht viel genutzt. Die Skepsis bei potenziellen Anwendern ist noch zu groß im Hinblick auf
Investitionsrentabilität und Betriebskosten. Ausserdem herrscht auf der technischen Ebene grosse Verunsicherung. „Europa gehört im Bereich der Mobilkommunikation zur Weltspitze, sein Markt für mobile

Anwendungen bleibt jedoch bisher hinter den Erwartungen zurück.“
Um die Möglichkeiten von mobilen Anwendungen auszuschöpfen, müssen bestehende Arbeits- und Geschäftsprozesse auf ihren Gehalt an mobilen Tätigkeiten analysiert und anschließend Lösungen für Einsatz von mobilen Anwendungen gefunden werden.
Die mobilen Anwendungen haben alle eine Gemeinsamkeit: sie sind in eine herkömmliche informationstechnische Infrastruktur eingebunden und es besteht ein
Informations- und Kommunikationsbedürfnis, das mit bestehenden, herkömmlichen Technologien nicht erfüllt werden kann.

8.2 Mobile Anwendungen

8.2.1 Techniken bei der Übertragung

8.2.1.1 Grundlagen und Begriffe

Eines der entscheidenen Merkmale von mobilen Anwendungen ist die Verwendung von mobilen elektronischen Kommunikationstechniken. Diese sollen im Folgenden betrachtet werden.
Als elektronische Kommunikationstechniken bezeichnet man Verfahren, die den Austausch von Signalen auch über eine größere Entfernung ermöglichen. Als mobil bezeichnet man eine Übertragung, die drathlos erfolgt.
Dieses Kapitel beschäftigt sich nun mit den Übertragungstechnologien.
Die wesentlichen Technologien sind:

- Für den Bereich Weitverkehrsnetze (WAN):
  Mobilfunk besonders dann, wenn für mobile Endgeräte vollständige Ortsunabhängigkeit erforderlich ist.

- Für den Bereich lokaler Vernetzung (LAN):
  Wireless LAN besonders dann, wenn mobile Endgeräte sich in einem lokal eingegrenzten Bereich bewegen sollen.

- Für den Bereich persönlicher Vernetzung (PAN):
  Besonders dann, wenn mobile Endgeräte untereinander oder mit Peripheriegeräten verbunden werden sollen (z.B. ein PDA mit einem Drucker)

8.2.1.2 Mobilfunk

Mobilfunk ist eine Form der Telekommunikation, bei der ein Dienstanbieter die Übertragung von Sprache und Daten von und zu mobilen Endgeräten durch ein drahtloses Zugangsnetz auf der Basis elektromagnetischer Wellen ermöglicht. Ein Mobilfunknetz besteht aus dem Mobilvermittlungsnetz (in dem der größte Teil der Übertragung stattfindet) und dem Zugangsnetz. Das Mobilvermittlungsnetz übernimmt die Übertragung zwischen den ortsfesten Einrichtungen des Mobilfunknetzes. Das Zugangsnetz übernimmt die Übertragung zwischen einer Mobilfunkantenne und dem mobilen Endgerät. Der GSM-Standard (siehe 2.1.4) bildet dabei die Grundlage für die Nachfolgestandards GPRS (siehe 2.1.6) und UMTS (siehe 2.1.8).
Zur effizienten Nutzung von Funkresourcen werden Multiplexverfahren eingesetzt, d.h. man bedient sich verschiedener Möglichkeiten das Frequenzspektrum auf unterschiedlichster Art mehrfach zu nutzen. Die

verwendeten Verfahren sind Raummultiplex, Frequenzmultiplex, Zeitmultiplex und Codemultiplex. Dabei kommen häufig Kombinationen dieser Verfahren zum Einsatz.

8.2.1.3 Mobilfunkstandards

Wir befinden uns in der 3. Generation von Mobilfunkstandards. Mobilfunkstandards der 1. Generation verwenden eine analoge Übertragungstechnik, die teils datenfähig ist. Die Mobilfunkstandards der 2. Generation verwenden digitale Netze. Sie sind auf Sprachübertragung optimiert und generell datenfähig. Um jedoch den heutigen Anforderungen zu genügen, war als erster Schritt zur 3. Generation die Fähigkeit zur paketorientierten Datenübertragung erforderlich. Hierfür wurden die Technologien GPRS und EDGE verwendet. Die Mobilfunkstandards der 3. Generation unterstützen vor allem höhere Übertragungsraten und Multimediaanwendungen, sie sind datenoptimiert. Aufgrund der GSM-Dominanz kann von UMTS als wichtigstem 3G-Standard ausgegangen werden. In den Netzen der 4. Generation werden zukünftig verschiedene Funktechnologien integriert zur Verfügung stehen.

8.2.1.4 GSM-Netz

Ein GSM-Netz (Global System for Mobile Communication) stellt drei Arten von Benutzerdiensten zur Verfügung: Supportdienste, Teledienste und Zusatzdienste. Das Netz wird in drei Subsysteme untergliedert: BBS (Base Station Subsystem), NSS (Network and Switching Subsystem), OSS (Operation and Support Subsystem).
Das BBS umfasst das Zugangsnetz zur Anbindung der Mobilfunkteilnehmer an das Netzwerk.
Das NSS umfasst das Mobilvermittlungsnetz zur Vermittlung der Nutzdaten innerhalb des Netzes und zur Bereitstellung der Anbindung an andere Netze.
Das OSS umfasst Systeme für Konfiguration und Wartung des Netzes.

8.2.1.5 Datenübertragung

Der standardmäßige GSM-Datendienst ermöglicht eine Übertragungsrate von maximal 14,4 kBit/s. Eine Softwareerweiterung ermöglicht eine Steigerung der
Übertragungsrate durch Kanalbündelung. Die Datenübertragung kann in Netzen der 2. Generation nur verbindungsorientiert erfolgen, d.h. die Übertragungsstrecke wird exklusiv geschaltet. Um die Netzkapazität dynamisch aufteilen zu können, ist aber eine paketorientierte Datenübertragung notwendig. Dies ermöglicht, dass eine Netzverbindung ständig besteht, Ressourcen aber nur bei Bedarf belegt werden. Dadurch wird die Masse der mobilen Anwendungen erst ermöglicht.

8.2.1.6 GPRS-Netz

Die Aufrüstung des GSM-Standards auf den GPRS (General Paket Radio Service) betrifft nur die Datenübertragung. Die Sprachübertragung bleibt gleich. Bei der Datenübertragung bleibt nur die Übertragung vom mobilen Endgerät bis zur Sendeantenne, bzw. von der Sendeantenne zum mobilen Endgerät gleich. Parallel zu der vorhandenen Domäne wird eine paketvermittelte Domäne aufgebaut. Zur Erhöhung der Datenübertragungsraten verwendet GPRS je nach Übertragungsqualität verschiedene Codierungsverfahren, wodurch pro Zeitschlitz mehr Nutzdaten übertragen werden können.

8.2.1.7 EDGE

EDGE (Enhanced Data Rates for Global Evolution) realisiert sowohl paketvermittlung als auch höhere Bandbreiten. Hierbei werden leistungsvermittelnde Dienste und paketvermittelnde Dienste realisiert, indem ein GPRS-ähnliches Mobilvermittlungsnetz aufgebaut wird. Analog werden verschiedene Codierungsverfahren verwendet und bis zu 8 Zeitschlitze zugewiesen.

8.2.1.8 UMTS-Netz

UMTS (Universal Mobile Telecommunications System) setzt auf ein GRRS-Netz auf und fügt ein vollständig neues Zugangsnetz hinzu, das in einem neuen Frequenzspektrum und unter Verwendung von CDMA Übertragungsraten von bis zu 2 MBit/s erzielt. Die Zellen eines Netzes verwenden prinzipiell dasselbe Frequenzband; ihre Größe ist variabel und von der Nutzerzahl abhängig. Die maximale Übertragungsrate hängt von der Art der Zelle, der Mobilität des Nutzers und der Anzahl der Nutzer in dieser Zelle ab.
Ein typisches Migrationskonzept der 3. Generation besteht in einem Multi Access Radio Network, das eine Kombination aus UMTS in Ballungsräumen, GPRS (oder EDGE) im übrigen Netz und Standard-GSM in Randbereichen vorsieht.

8.2.1.9 Wireless LAN

Als Wireless LAN (WLAN) bezeichnet man eine durch drathlose Technologie realisierte lokale Vernetzung nach den IEEE 802.11 Spezifikationen. Hierbei wird typischerweise eine sternförmige Vernetzung um Zugangsknoten realisiert, alternativ ist auch ein Betriebsmodus vorgesehen, bei dem sich die Endgeräte direkt miteinander vernetzen.
Derzeit ist WLAN im Wesentlichen im LAN-Bereich von Bedeutung, etwa um einen drathlosen Intranetzugang zu ermöglichen. Künftig ist an sogenannten Hot Spots eine Konkurrenzsituation zur Datenanbindung über Mobilfunk möglich.

8.2.1.10 Bluetooth, IrDA

Bluetooth und IrDA sind Industriestandards zur drathlosen Datenübertragung auf PAN-Ebene. Sie dienen der Vernetzung von Endgeräten untereinander oder mit Perepheriegeräten. Während Bluetooth auf Funktechnologie beruht, findet bei IrDA eine Übertragung mittels Infrarotlicht statt. Beide Technologien finden auch in sehr kleinen Endgeräten Platz.

8.2.1.11 SMS/MMS

Der Short Message Service (SMS) ist ein GSM-Dienst zum Versenden von Kurznachrichten bis 140 Byte Länge, die Text und Nutzdaten enthalten können. Es ist mit Hilfe von SMS Cell Broadcast möglich, die Nachrichten an mehrere Empfänger in einer, mehreren oder allen Zellen eines Netzes zu senden.
Multimedia Messaging Service (MMS) ist ein Erweiterung des SMS zum Versenden von Nachrichten mit integrierten Bild-, Video-und Audiodaten, der in GPRS- und UMTS-Netzen bereitgestellt wird.

8.2.1.12 WAP

Das Wireless Application Protocol (WAP) ist ein Standard für die Übertragung und Darstellung von Daten auf mobilen Endgeräten (Abb. 1). Er besteht aus dem Wireless Application Environment mit

der Auszeichnungssprache WML, der Skriptsprache WMLScript, einem Microbrowser, der Programmierschnittstelle Wireless Telephony Applications (WTA) Framework und einem optimierten Protokollstapel für die Übertragung auf der Schnittstelle zwischen Sendeantenne und mobiles Endgerät. Mittels WML werden Webseiten speziell für die Darstellung auf Mobiltelefonen erzeugt und mittels WMLScript einfache Vorgänge (z.B. Berechnungen) automatisiert durchgeführt. Diese Seiten werden dann auf einen Standard-Webserver abgelegt und beim Aufruf zunächst mittels HTTP zu einem WAP-Gateway übertragen. Dort erfolgt eine Protokollversion und die Übertragung durch WSP (WIreless Session Protocol) zum Endgerät.

8.2.2 Architektur

Wenige mobile Anwendungen laufen eigenständig, daher wird auch nicht näher darauf eingegangen. Eine P2P Architektur findet man ebenfalls nur selten. Die meisten Architekturen von mobilen Anwendungen basieren auf einer Client-Server Architektur, deshalb wird hier auf die Client-Server-Architektur eingegangen.

8.2.2.1 Client-Server Architektur

Bei der Client-Server Architektur gibt es zwei Unterschiedliche Prinzipien.Es gibt serverseitige Technikenund es gibt clientseitige Techniken. Realisierte Dienste werden unterschieden in Pull-Dienste, bei denen der Nutzer oder sein Endgerät die Datenübertragung anstoßen, und Push-Dienste, bei denen der Server die Datenübertragung anstößt.

Bei serverseitiger Programmierung sind als Nutzerschnittstelle vor allem IVR (Interactive Voice Responce), Kurznachrichten und HTML für mobile Endgeräte von Bedeutung. Bei clientseitiger Programmierung können je nach Endgerät etwa WMLScript oder Standard-Programmiersprachen und -werkzeuge eingesetzt werden, insbesondere ist hier Java von Bedeutung.

Der Abruf von WWW-Seiten ist ein klassisches Beispiel für einen Pull-Dienst. Das WAP-Modell funktioniert genauso, allerdings ist ein WAP-Gateway mit der Funktionalität eines Proxy-Servers dazwischengeschaltet.

8.2.3 Mobile Endgeräte

8.2.3.1 Grundlagen und Begriffe

Nach den mobilen elektronischen Kommunikationstechniken stehen nun die mobilen Endgeräte im Mittelpunkt der Betrachtung. Unter mobilen Endgeräten versteht man alle diejenigen Geräte, die für den mobilen Einsatz konzipiert sind. Dies beginnt bei Mobiltelefonen, geht über Handheld-Geräten, wie z.B. ein PDA, bis hin zum Laptop. Allerdings gibt es bei einem Laptop Einschränkungen. Der Laptop wurde zwar für einen Einsatz an wechselden Orten optimiert und er kann ohne vorhandene Infrastruktur, d.h. mit eigener Stromversorgung und gegebenenfalls mobiler Stromversorgung. Der Einsatz des Gerätes bleibt aber dem Charakter nach ein stationärer Einsatz. Tablet-PC sind ein Grenzfall, da sie wie ein Arbeitsplatzrechner eingesetzt werden können, aber auch wie ein mobiles Endgerät, z.B. bei einer Inventur. Daher werden sie in diesem Kapitel nicht als eigener Typus mobiler Endgeräte, sondern als Abwandlung kleinerer Handheld-Geräte betrachtet.
In diesem Kapitel werden zuerst die Konzepte des „Ubiquitous Computing“ eingeführt. Schlüsseltechnologien wie immer kleinere, leichtere und leistungsfähigere Prozessoren, Sensoren, Displays und Kommunikationsmodule ermöglichten das Entstehen mobiler Endgeräte in der heutigen Form.
Ein weiterer Abschnitt zeigt die derzeit relevanten Endgerätekategorien auf und differenziert diese nach ihrer Funktionalität und der Möglichkeit zur Benutzerinteraktion. In der Hauptsache handelt es sich hier um Mobiltelefon, Smartphone und PDA.

8.2.3.2 Ubiquitous Computing

Der Begriff leitet sich aus dem englischen Begriff „ubiquitous“ (allgegenwärtig) her. Dies bedeutet, dass Computer zwar allgegenwärtig sind, dennoch immer mehr in den Hintergrund treten, indem sie mit Alltagsgegenständen verschmelzen. Ein Beispiel hierfür ist ein elektronischer Notizblock.
Die Allgegenwärtigkeit ist die charakteristische Eigenschaft von mobilen Endgeräten und hat dabei zwei Seiten: Auf eine Person bezogen ist dies die Portabilität. Beispiele hierfür sind Mobiltelefone, PDAs oder in Kleidungsstücke integrierbare bzw. am Körper tragbare Mikrochips. Auf ein Informationssystem bezogen, ist dies die Tatsache, dass IT überall vorhanden ist.
Realisiert wird dies durch die technologischen Weiterentwicklungen, durch die immer kleiner, leichter und leistungsfähiger werdenden Prozessoren, Sensoren, Displays und der Möglichkeit zur drathlosen Kommunikation.

8.2.3.3 Kategorien mobiler Endgeräte

Im folgenden Abschnitt werden einige Endgerätekategorien aufgezeigt. Es handelt sich hierbei um das Mobiltelefon, das Smartphone und das PDA.
Als Mobiltelefon bezeichnet man ein mobiles Endgerät, das primär auf die Nutzung der Telefonfunktionalität ausgelegt ist. Die neueste Generation von Mobiltelefonen verfügt über die Schnittstellen IVR (Interactive Voice Response), es besteht die Möglichkeit SMS zu empfangen und zu senden und das Mobiltelefon hat die Möglichkeit mittels WAP-Microbrowser (Wireless Application Protocol) zum Internetzugang. Zusätzlich verfügt das Mobiltelefon über die Java 2 Platform Micro Edition (J2ME) Virtual Machine, eine gerätespezifische Implementierung der Programmiersprache Java. So kann man nun komplexe Anwendungen auf das Gerät laden, dort speichern und ausführen, so wird eine weitgehend geräteunabhängige Anwendungsentwicklung ermöglicht. In Java-Anwendungen kann eine integrierte Bezahlfunktionalität verwendet werden.

I-mode ist ein propietärer Standard für die Nutzung spezieller Internetdienste auf mobilen Endgeräten. In seinem Ursprungsland Japan ist er marktbeherrschend, außerhalb von Japan jedoch von geringer Bedeutung.
Ein Personal Digital Assistant (PDA) ist ein Handheld-Computer, dessen Kernfunktionalität als Organizer bezeichnet wird. Er umfasst Terminkalender, Adressbuch, Aufgabenverwaltung und eine Mailfunktionalität. Je nach Betriebssystem kommen noch andere Programme hinzu. Die Standard-Datenanbindung kann mit diesen Geräten auf zwei Arten erfolgen:

- Offline-Synchronisation in Form von Datenabgleich mit einem PC, entweder über eine fest verbundene Docking-Station oder über drathlose PAN-Technologien.
- Direkte Internetverbindung über ein modemfähiges Mobiltelefon oder durch integrierte Mobilfunk-Funktionalität.

Auf jeden Fall besteht die Möglichkeit eine Programmierung in einer höheren Programmiersprache durchzuführen. Besteht eine direkte Internetverbindung, so können auch weitere Dienste wie SMS oder WAP genutzt werden. Das Betriebssystem des PDA ist dem des PC ähnlich. Die marktbeherrschenden Systeme basieren auf den Betriebssystemen Palm OS und Windows CE; beide Systeme haben je nach Art der Anwendung Stärken und Schwächen, die gegeneinander abzuwägen sind.
Den Begriff Smartphone genauer zu definieren ist nicht ganz leicht, da dieser Begriff häufig als Marketinginstrument gebraucht und deshalb irreführend für die verschiedensten Arten von mobilen Endgeräten verwendet wird. Ein Smartphone steht in der Mitte zwischen Mobiltelephone und PDA. Von einem Smartphone spricht man dann, wenn es in erster Linie als Mobiltelefon verwendet wird, aber dennoch über ein Betriebssystem verfügt, dass in wesentlichen Teilen dem eines PDA ähnlich ist.
Damit bilden Smartphones eine eigene Gerätekategorie und sind mit ihren speziellen Eigenschaften ein besonderes Zielgerät für mobile Anwendungen.

8.2.4 Anwendungsgebiete

Es gibt verschiede mobile Anwendungsgebiete. Im engeren Sinne geht es in diesen Anwendungsgebieten immer um die Ortung von Personen oder Objekten, um mit ihnen zu kommunizieren, also Daten zu transferieren oder zu tauschen. Im folgenden Kapitel werden einige dieser Anwendungsgebiete beschrieben.
Ortungen können prizipiell auf drei Arten erfolgen:

- Manuelle Ortseingabe
- Spezialisierte Ortungssysteme (GPS)
- Innerhalb eines drathlosen Kommunikationsnetzes

Manuelle Ortseingabe ist eine Einstiegslösung, bei der der Nutzer beim Dienstaufruf Ortsname oder Ortskennzahl mit angibt.
Spezialisierte Ortungssysteme sind vor allem verschiedene Verfahren für die Ortung innerhalb von Gebäuden, sowie die Satellitenortung mit dem Global Positioning System (GPS). GPS ermöglicht eine sehr genaue Ortung, erfordert aber spezielle Hardware für die Satellitenkommunikation. Hierbei sendet die Hardware ein Signal an verschiedene Satelliten (mindestens drei Satelliten), diese senden die Positionsdaten an einen zentralen Rechner, der dann die genaue Position bestimmt.
Ortung innerhalb bestehender drathloser Kommunikationsnetze umfasst vor allem die Nutzung des Mobilfunks. Als Grundlage ist im Wesentlichen die Zellidentifikation von Bedeutung, d.h. in welcher Zelle sich das Endgerät gerade befindet. Eine genauere Ortung ist sehr langsam und auch aufwändig, sowohl bezüglich der Anforderungen bei der Aufrüstung von Netz und/oder Endgeräten, als auch bezüglich des Ressourcenverbrauchs während der Laufzeit.

8.2.4.1 Logistik

Im Güterverkehr auf der Straße findet die Kommunikation zwischen Fahrer und Unternehmen, insbesondere bei KMUs (klein- und mittelständische Unternehmen), praktisch nur verbal statt. Frachtpapiere müssen immer beim Unternehmen abgeholt werden, dies bedeutet oft unnötige Umwege, Statusberichte finden nur auf Nachfrage statt.
Hier setzt die M-Logistik-Strategie der Bremer Innovations-Agentur ein. Hierfür werden einige Komponenten benötigt:

- Fahrzeugortung über GPS
- Fahrzeugkommunikation über Mobilfunk
- Sendungsidentifikation über Barcode und Lesegerät

- Fahrzeug- und Containeridentifikation über Tags
- Mobiles Endgerät

Die Strategie zielt darauf ab, dass Unkosten für die Unternehmen durch kürzere Wege und geringeren Zeitaufwand minimiert werden. Dies soll mit Hilfe der Komponenten wie folgt geschehen:

- Übertragung von Sendungs-/Frachtbriefdaten an den Fahrer
- Übertragung von Statusdaten/Ereignissen an die Zentrale
- Übertragung von Zusatzinformationen an die Zentrale
- Fahrzeugortung zyklisch oder auf Anfrage

Diese Daten werden in einem zentralen EDV-System festgehalten. Auf dieses EDV-System können der Fahrer, der Disponent und auch der Kunde zugreifen. Sie können die jeweils für sie interessanten Daten zu jeder Zeit abfragen.
Es gibt zwei verschiedene Lösungen:

1. Geeignete Bausteine für das Projekt waren Internet und WAP-fähige Mobiltelefone oder PDAs. Die Datenübertragung erfolgt über GSM oder GPRS. Die Anwendungen laufen auf einem zentralen Server, daher wird keine besondere Anforderung an die Endgeräte gestellt. Hierbei ist ein Ausdruck von Belegen ohne weiteres möglich.
2. Dieses System umfasst ein PDA, bzw. Smartphone und einen mobilen Thermodrucker. Die Software befindet sich in diesem Beispiel auf dem Endgerät. Mit Hilfe des mobilen Druckers kann der Fahrer seine Transportaufträge selber ausdrucken.

8.2.4.2 Mobile Anwendungen im Gesundheitswesen

Ein Anwendungsbereich mit einem hohen Anteil an mobilen Tätigkeiten ist das Gesundheitswesen. Hier finden auch die unterschiedlichsten Arten von mobilen Anwendungen statt, bzw. könnten stattfinden, z.B. in der Notfallmedizin, in der ambulanten Pflege, in der Abwicklung der Visite oder in der Instandhaltung der medizinischen Geräte.
Bereits existierende mobile Anwendungen sind mobile patientenorientierte Notruflösungen: Risikogruppen wie Herzkranke, Bluter, Diabetiker oder Bluthochdruckpatienten werden mit einem speziellen mobilen Endgerät ausgestattet, das ähnlich wie ein Handy zu bedienen ist und zusätzlich mit einem Notrufknopf, einem GPS-Empfänger, sowie mit einem mobilen EKG- oder einem anderen Vitalwert-Messgerät ausgestattet ist.
Zusätzlich existieren aber laut einer Studie noch viele andere Anwendungsgebiete in der Medizin, wo mobilen Anwendungen Verwendung finden könnten. Zusätzlich zu den bereits in Abschnitt erwähnten Beispielen wären das ärztliche Hausbesuche, ortsunabhaägige Zugriff auf Fachinformationen und Nachschlagewerke oder klinische Dokumentation, inkl. Zugriff auf Patientendaten.

8.2.4.3 Touristen-Führer/MobiDenk

Hier existieren verschiedene mobile Anwendungen, diese basieren jedoch alle auf demselben Prinzip. Es sind Client-Server-Architekturen.

1. MobiDenk heißt mobile Denkmäler. Hier wurde eine Datenbank mit wissenswerten Informationen zu Denkmälern angelegt. Steht man nun an einem Denkmal, z.B. in den Herrenhäuser-Gärten in Hannover, kann mit einem PDA Informationen zu diesem Denkmal abrufen. In diesem PDA ist eine GPS-Funktion integriert, d.h. dem Server wird der genaue Standort des Benutzers übermittelt und dieser schickt dann die passenden Daten zu dem Denkmal, an dem der Benutzer gerade steht, auf das PDA.

2. Weitere Anwendungen findet man im Bereich der Stadtführung. Hier gibt es die Möglichkeit sich Informationen über bestimmte, z.B. historische Gebäude geben oder sich zum nächsten Restaurant führen zu lassen. Unterschiede bestehen hier in der Art und Weise wie man sich Informationen zukommen lassen kann. Unterschiedliche Möglichkeiten des Informationsaustausches gibt es aber nicht in einem System, sondern unterschiedliche Systeme übermitteln dem Benutzer unterschiedlich die Informationen. Einige Systeme bieten die Möglichkeit, sich die Informationen auf einem Endgerät anzeigen zu lassen.

3. Bei anderen Systemen ist es so, dass auf dem Endgerät die Informationen gespeichert sind. Dem Endgerät wird übermittelt, wo sich der Benutzer gerade befindet und die Informationen werden dem Benutzer verbal übermittelt.

4. Eine weitere Variante ist, dass der Benutzer kein Endgerät mehr benötigt, sondern einen Sensor für GPS-Ortung bei sich trägt. Die gewünschten Informationen über ein Gebäude werden dem Benutzer über einen Lautsprecher am Gebäude übermittelt.

8.2.4.4 Spiele

Die meisten Spiele benötigen ein Handheld-Endgerät, ein GPS-Gerät und einen zentralen Game-Server. In diesem Kapitel gehe ich näher auf des „paper chase game“ ein. Es gibt noch andere ähnliche Spiele, die hier aber nicht weiter erläutert werden.
Im „paper chase game“ gibt es zunächst einmal eine Person, die das Spiel initiert. Diese Person erstellt eine Karte mit dem Startpunkt, einigen Wegpunkten und den Zielpunkt. Jeder Mitspieler, bzw. jede Gruppe, erhält ein mobiles Endgerät und ein GPS-Gerät zur genauen Ortung der Position. Die Karte mit den Punkten wird vom Server auf das Endgerät überspielt. Die Mitspieler müssen nun anhand der Karte zu den einzelnen Wegpunkten und zum Ziel finden. An jedem Wegpunkt wird dem Mitspieler eine Frage gestellt. Diese wird vom Server, der erkennt anhand der GPS-Ortung an welchem Wegpunkt sich der Mitspieler befindet, übermittelt. Der Mitspieler gibt die Antwort in das Endgerät ein und diese wird dann zum Game-Server übermittelt. Nach Beantwortung der Frage kann der Mitspieler zum nächsten Wegpunkt gehen. Während des ganzen Spiels kann der Mitspieler seine Position auf der Karte auf dem Endgerät sehen. Der Veranstalter, derjenige, der die Karte konzipiert hat, kann über die gesamte Dauer des Spiels den Spielstatus aller Mitspieler sehen.

8.2.5 Anwender

In diesem Kapitel soll kurz betrachtet werden, ob die mobilen Anwendungen multimodal sind. Daraus geht dann hervor, ob sie von jeder Person genutzt werden können oder ob es Einschränkungen gibt.

8.2.5.1 Unterstützung blinder und sehbehinderter Menschen

Die bisher am häufigsten genutzten mobilen Anwendungen können auch blinde oder sehbehinderte Menschen nutzen, nämlich das Telefonieren mit dem Mobiltelefon. Allerdings unterstützt diese Anwendung auch nur das Telefonieren, d.h. SMS werden z.B. nicht verbal ausgegeben.
Es existieren aber andere mobile Anwendungen, die blinde oder sehbehinderte Menschen nutzen können. Zumeist sind dies aber mobile Anwendungen, die direkt auf blinde oder sehbehinderte Menschen zugeschnitten sind, d.h. sie im alltäglichen Leben unterstützen.
Die meisten mobilen Anwendungen können blinde oder sehbehinderte Menschen jedoch nicht nutzen, da die übertragenen Daten nur visuell auf dem Display auf einem Endgerät ausgegeben werden.

8.2.5.2 Gehörlose Menschen

Gehörlose Menschen können theoretisch alle mobilen Anwendungen, außer natürlich das Telefonieren mit dem Mobiltelefon, nutzen, da bei allen mobilen Anwendungen eine visuelle Datenausgabe auf einem Display erfolgt.

Hier kann man sagen, dass es für behinderte Menschen starke Einschränkungen bei der Nutzung von mobilen Anwendungen gibt.

8.3 Zusammenfassung

Es gibt unterschiedliche Arten von mobilen Anwendungen, die in unterschiedlichsten Bereichen eingesetzt werden, bzw. eingesetzt werden sollen. Typische Anwendungsgebiete gibt es also eigentlich nicht. Die Anwendungen beruhen zum größten Teil auf Client-Server Architektur und sind im Grunde genommen kontextunabhängig. Es gibt allerdings Unterschiede in dieser Architektur. Es gibt das Push-Prinzip und das Pull-Prinzip. Die Ortsunabhängigkeit ist theoretisch auch gegeben, allerdings macht es bei einigen Anwendungen keinen Sinn, so kann der mobile Stadtplan von Bamberg nicht in Oldenburg genutzt werden.
All die Anwendungen nutzen zumeist das gleiche Prinzip:
Zunächst wird das mobile Endgerät geortet, um anschließend Daten zum Endgerät zu übertragen, bzw. vom Endgerät zu empfangen, aber es gibt unterschiedliche Arten von Endgeräten. Die wichtigsten sind sicherlich das Mobiltelefon und das PDA. Durch die Fähigkeit zur Ausführung von Programmen in höheren Programmiersprachen, zumeist Java, besteht die Möglichkeit, komplexe Anwendungen auf den Endgeräten auszuführen. Die Interaktionsmöglichkeiten bei den Endgeräten sind zum Einen die Kommunikation per Texte, zusätzlich besteht bei dem Mobiltelefon aber auch die Möglichkeit zur verbalen Kommunikation. Die Interaktionsmöglichkeiten sind also noch sehr eingeschränkt und können verbessert werden.
Die Technik des Datentransfer wird schneller, so dass immer mehr Daten in kurzer Zeit übertragen werden können. Außerdem hält die Computertechnik in alle Lebensbereiche Einzug. Sie wird immer kleiner und fällt einem Menschen im normalen Alltag kaum noch auf.
Viele Anwendungen sind auch nur für Sehende geeignet, hier könnte man sicherlich auch noch einiges verbessern.
In Zukunft werden mobile Anwendungen wohl noch viel mehr in des alltägliche Leben der Menschen Einzug halten. Die Skepsis, die im Moment noch vorhanden ist, wird weichen. Mobile Anwendungen können Zeit sparen, Unkosten einschränken und vielleicht der wichtigste Aspekt: sie können bei Anwendung im Gesundheitswesen Leben retten.

Kapitel 9

Tangible Bits

Abstract Die Interaktion mit Informationssystemen mittels grafischer Benutzeroberflächen ist eine für den Menschen unnatürliche Art zu kommunizieren. Tangible Bits ist eine Vision zur Revolutionierung der Mensch-Computer Interaktion. Ihr Ziel ist es, die Interaktion in eine für den Menschen natürlichere Ebene zu verlagern, indem alltägliche Gegenstände und Oberflächen mit digitalen Informationen verkuppelt werden. Gleichzeitig sollen Umgebungsmedien eingesetzt werden, um digitale Informationen zu übermitteln, ohne die Aufmerksamkeit des Empfängers zu beanspruchen. Dadurch rückt der Mensch wieder in den Mittelpunkt, wo heute der Computer im Zentrum des menschlichen Handelns steht. Diese Seminararbeit beschreibt die wesentlichen Konzepte von Tangible Bits und evaluiert die Einsatzmöglichkeiten innerhalb der Projektgruppe „eXplorer“, in deren Rahmen sie entstanden ist.

9.1 Einleitung

9.1.1 Aufgabenstellung

Die vorliegende Ausarbeitung entstand im Rahmen der Projektgruppe „eXplorer“ mit dem Themenbereich „Kontext-sensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“, die Wintersemester 2004/2005 im Fachbereich Informatik an der Universität Oldenburg angeboten wurde. Thema der Seminararbeit ist eine Vision zur Revolutionierung der Interaktion von Mensch und Computer durch die Kommunikation. Ziel dieser Ausarbeitung ist es, einen Einblick in das dahinter stehende Konzept namens *Tangible Bits* zu geben und eventuell für das Projekt einsetzbare Erkenntnisse, Konzepte und Ideen zu identifizieren und Beispiele zu geben.

9.1.2 Motivation für Tangible Bits

Die Interaktion mit Computersystemen hat sich in den letzten Jahrzehnten zu einer Monokultur entwickelt. Die meisten Computersysteme verlassen sich auf textuelle oder grafikbasierte Darstellung relevanter digitaler Informationen. Die Eingabe von Daten erfolgt meist über universell verwendbare Eingabegeräte, wie z.B. Tastatur oder Maus. Dieses in [Ull02] als WIMP (windows-icon-menu-pointer) bezeichnete Paradigma hat sich in den meisten Fällen als sehr erfolgreich erwiesen. Es wird jedoch wegen der Ungleichmäßigkeit zwischen Ein- und Ausgabe, sowie die für den Menschen unnatürliche Handhabung kritisiert. Während die Ausgabe über Millionen von Pixeln stattfindet, beschränkt sich die Eingabe auf einen einzigen Punkt, z.B. den Cursor bzw. Mauszeiger. Dadurch kann der Nutzer nicht mehrere Aufgaben

parallel erledigen, wie er es z.B. beim Steuern eines Fahrzeugs tut. Die WIMP-Benutzerschnittstellen schöpfen also die Fähigkeiten des Menschen nicht optimal aus.

Tangible Bits ist eine Vision zur Revolutionierung der Interaktion von Mensch und Computer. Ziel ist es, die Interaktion auf eine für den Menschen natürlichere Ebene zu verlagern. Alltägliche Gegenstände sollen zur Ein- und Ausgabe digitaler Informationen erweitert werden und dadurch einen intuitiven und effektiven Zugang zu Informationssystemen ermöglichen. Umgebungsmedien sollen gezielt zur Übertragung von Informationen eingesetzt werden, ohne die Aufmerksamkeit des Nutzers zu beanspruchen.

9.1.3 Übersicht

Abschnitt 9.2 führt in die Interaktion von Mensch und Computer ein. In Abschnitt 9.3 wird das eigentliche Konzept, Beispielumsetzungen, Einsatzgebiete und Stärken vorgestellt. Abschnitt 9.4 evaluiert die Möglichkeiten, Tangible Bits im Rahmen der Projektgruppe einzusetzen. Abschnitt 9.5 fasst die wichtigsten Ergebnisse zusammen und liefert einen Ausblick in die Zukunft von Tangible Bits.

9.2 Human-Computer Interaction (HCI)

Tangible Bits ist eine Vision zur Revolutionierung der *Human-Computer Interaction*. Human-Computer Interaction (HCI) ist ein großes, interdisziplinäres Feld, das sich mit Design, Evaluation und Implementierung interaktiver Rechensysteme zum Gebrauch durch Menschen und den wichtigsten, sie umgebenden Phänomenen beschäftigt. Vom Standpunkt der Informatiker aus steht dabei die Interaktion zwischen einem oder mehreren Menschen mit einem oder mehreren Computersystemen im Vordergrund. Andere Disziplinen wie Psychologie, Soziologie oder Anthropologie unterstützen Design und Evaluation der Benutzerschnittstellen. Durch die unterschiedliche Auslegung von „Interaktion“, „Mensch“ und „Computer“ fallen auch diverse, auf den ersten Blick nicht offensichtliche Themen in den Bereich der HCI. Nach [HBC+96] beschäftigt sich HCI u.a. mit dem :

- **Zusammenspiel** von Mensch und Computer bei der Erledigung von Aufgaben, der
- **Kommunikationsstruktur** zwischen Mensch und Computer, den
- **menschlichen Fähigkeiten** den Computer zu benutzen (Usability, Learnability, Accessibility) sowie der
- **Erstellung der Benutzerschnittstelle** (Design, Engeneering, Implementierung)

Für die Akzeptanz von Informationstechnologie spielt die Benutzerschnittstelle eine entscheidende Rolle. Informationstechnologie erfährt unter anderem Ablehnung durch Berufsgruppen, die in sicherheitskritischen Bereichen, wie z.B. Gefechtsplanung beim Militär oder Start- und Landekoordination am Flughafen arbeiten. Nach [CM04] sind Kritikpunkte an gängigen grafischen Benutzerschnittstellen:

- **Mangelnde Zuverlässigkeit:** Systemausfälle können fatale Auswirkungen haben (z.B. Flugzeugabstürze), wenn entscheidende Informationen nur digital verfügbar sind.
- **Unnatürliche Handhabung:** Die Ein- und Ausgabemodalitäten der Informationssysteme passen nicht zur gewohnten Arbeitsweise der Benutzer, so dass sie nicht so effektiv arbeiten können, wie mit ihren gewohnten Utensilien.

- **Ungenauigkeit:** Durch die Digitalisierung der Daten können wichtige Informationen verloren gehen, wenn z.B. die höchstmögliche Auflösung nicht präzise genug ist.

- **Ablenkung:** Die Konzentration auf den Computer führt zur Ablenkung von der eigentlichen Aufgabe, hemmt also die Produktivität und erschwert die Zusammenarbeit mehrerer Menschen an einem Projekt.

Tangible Bits bietet Lösungen zu diesen Kritikpunkten, indem es sich haptischer Kommunikation, definiert in Abschnitt 9.2.1, bedient. Abschnitt 9.2.2 stellt *Graspable User Interfaces*, ein Art von Schnittstellen zur haptischen Kommunikation vor. Eine spezielle für *Tangible Bits* entscheidende Erweiterung, die so genannten *Tangible User Interfaces* werden in Abschnitt 9.2.3 erläutert.

9.2.1 Haptische Kommunikation

Haptische Geräte/Schnittstellen sind mechanische Geräte, welche die Kommunikation zwischen Mensch und Computer über den Tastsinn ermöglichen. Der Unterschied zu herkömmlichen Geräten, wie z.B. der Maus, ist das Feedback haptischer Schnittstellen. Während man beim Einsatz der Maus lediglich Daten in den Rechner eingibt, erfassen haptische Schnittstellen die Eingabe des Nutzers und bieten gleichzeitig haptische Ausgaben entsprechend der auf dem Bildschirm auftretenden Ereignisse. Zur Zeit sind haptische Schnittstellen deutlich weniger weit verbreitet als visuelle und akustische Schnittstellen [Ber03].

Laut den Autoren von [COJ+02] handelt es sich bei der haptischen Kommunikation um ein sehr mächtiges Kommunikationswerkzeug. Der Tastsinn übermittelt subtile, nonverbale Signale, die trotzdem schnell ins Bewusstsein rücken können. Dies kann helfen, wichtige Informationen hervorzuheben, damit sie nicht übersehen werden.

9.2.2 Graspable User Interfaces

In [FIB95] wird der Begriff *Graspable User Interfaces* eingeführt. Die Autoren definieren ihn wie folgt: *Graspable User Interfaces* erlauben die direkte Kontrolle elektronischer oder virtueller Objekte mittels als Steuerhebel dienenden, physikalischen Artefakten. Die Artefakte sind im Wesentlichen Eingabegeräte, die gezielt mit virtuellen Objekten verkuppelt werden können, um Aktionen auszulösen, wie z.B. das Verändern eines Parameters oder das Starten eines Prozesses. Sie verschmelzen so die virtuelle mit der physikalische Welt. Nimmt man an, dass die Fülle der Möglichkeiten physikalischer Steuerkontrollen von Natur aus reichhaltiger als die der virtueller Welt ist, erlauben Graspable User Interfaces eine wesentlich effektivere Arbeit mit dem Computer als Benutzerschnittstellen nach dem WIMP-Paradigma.

9.2.3 Tangible User Interfaces

In [IU97] wird erstmals der Begriff *Tangible User Interface* eingeführt. Tangible User Interfaces verkuppeln die reale Welt mit digitalen Informationen und machen sie dadurch greifbar. Ihre entscheidende Eigenschaft wird das *Abakus-Prinzip* genannt. Perlen, Stangen und Rahmen des Abakus dienen als manipulierbare physikalische Darstellung numerischer Operationen und Werte. Gleichzeitig ist der aktuelle Zustand der Perlen aber auch die mathematische Ausgabe des Abakus. Er ist damit im Sinne der HCI weder Ein- noch Ausgabegerät. Diese Eigenschaft grenzt Tangible User Interfaces von den Graspable User Interfaces ab. In dieser Arbeit wird der Begriff Tangible User Interface mit Tangible Interface synonym verwendet [UI00].

9.3 Die Vision der Tangible Bits

Dieser Abschnitt erläutert die in [IU97] beschriebene Vision zur Revolutionierung der HCI, den *Tangible Bits*. Tangible Bits sind Projekte, die durch den Einsatz „greifbarer“ Nutzerschnittstellen und Umgebungsmedien versuchen, die Kluft zwischen der realen und der digitalen Welt schließen. Abbildung 9.1 zeigt die Schlüsselkonzepte von Tangible Bits und deren Zusammenspiel. Die Schlüsselkonzepte im Einzelnen sind:

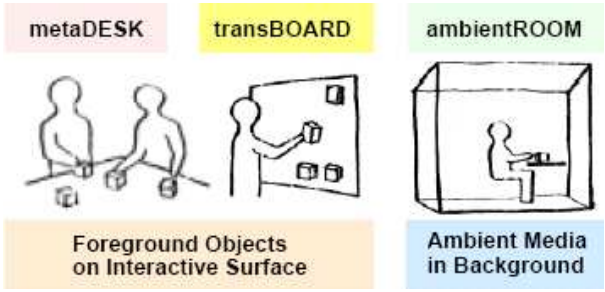


Abbildung 9.1: Forschungsprototypen

1. **Interactive Surfaces** (Interaktive Oberflächen): die Umwandlung von gewöhnlichen Oberflächen, wie z.B. Wänden zu aktiven Schnittstellen zwischen der realen und der digitalen Welt,
2. **Ambient Media** (Umgebungsmedien): die Verwendung von Umgebungsmedien wie Geräuschen, Licht, Luftzug und Wasserfluss um die Sinneswahrnehmung des Menschen anzusprechen, die außerhalb seines aktuellen Gedankenfokus liegen, und
3. **Coupling of Bits and Atoms** (Verkuppeln von Bits und Atomen): das nahtlose Verkuppeln alltäglicher Gebrauchsgegenstände (z.B. Bücher) mit den digitalen Informationen, die sie enthalten bzw. repräsentieren.

Zu jedem dieser Schlüsselkonzepte wird in [IU97] ein Forschungsprototyp präsentiert, die jeweils in 9.3.1 vorgestellt werden. Abschnitt 9.3.2 stellt eine Reihe weiterer Beispiel für Tangible Bits vor. Abschnitt 9.3.3 arbeitet die Stärken und 9.3.4 die wichtigsten Einsatzgebiete von Tangible Bits heraus.

9.3.1 Forschungsprototypen

Die Forschungsprototypen sind der *metaDESK* (9.3.1.1) zur Verkupplung von Bits und Atomen, der *ambientROOM* (9.3.1.2) zu den Umgebungsmedien und das *transBOARD* (9.3.1.3) zu den Interaktiven Oberflächen.

9.3.1.1 Coupling of Bits and Atoms (metaDESK)

Ziel von *Coupling of Bits and Atoms* ist es, die Mensch-Computer Interaktion zurück in die reale Welt zu verlegen, indem Gegenstände für ihre vorgesehene Aufgabe verwendet, aber gleichzeitig mit dem Computer vernetzt werden. *metaDESK* repräsentiert den Versuch die populärsten Elemente einer klassischen grafischen Benutzungsoberfläche, wie z.B. Fenster, Icons oder Menüs durch physikalische Objekte nachzubilden und mit gleichwertiger Funktionalität auszustatten. Wie in Abbildung 9.2 verdeutlicht, bildet metaDESK eine Brücke zwischen der realen und der digitalen Welt, indem er die grafische Benutzeroberfläche greifbar macht bzw. physikalische Objekte mit dem Computer verbindet. Kern des metaDESKs ist ein Tisch, der grafische Inhalte auf seiner Oberfläche durch Projektion von der Rückseite anzeigen kann. Weitere wichtige Elemente sind:

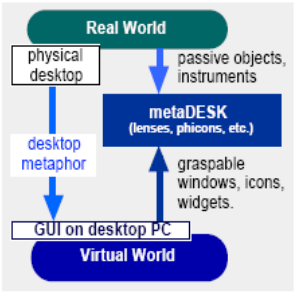


Abbildung 9.2: Design des metaDESK

- *activeLENS*, ein mittels Schwenkarm platzierbarer LCD-Bildschirm.
- *passiveLENS*, einer vom metaDESK gesteuerten, optisch transparenten Linse.
- *phicons*, physikalischen Icons.
- weitere Instrumente, die auf dem metaDESK platziert werden und durch diverse optische, mechanische und elektromagnetische Sensoren mit dem metaDESK kommunizieren können.

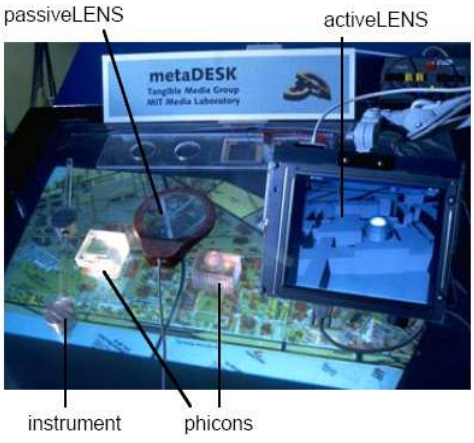


Abbildung 9.3: Tangible Geospace realisiert durch den metaDESK

Abbildung 9.3 zeigt den *Tangible Geospace*, eine prototypische Anwendung des metaDESK. Er erlaubt die Manipulation von zwei- und dreidimensionalen Darstellungen des MIT Media Laboratory Campus. Die zweidimensionale Karte der MIT Campus wird auf die Oberfläche des metaDESKs auf projiziert. Bewegt man die activeLENS über die Karte, stellt sie die darunter liegende Landschaft dreidimensional dar. Durch das Platzieren/Verschieben von Modellen zweier wichtiger durch Phicons repräsentierten Landmarken, dem MIT's Great Dome und dem Media Lab, kann die Darstellung der Karte gesteuert werden. Platziert man z.B. den MIT's Great Dome auf der Oberfläche des metaDESKs, erscheint die Karte der Region um ihn herum, wobei Position und Ausrichtung des Great Domes auf der Karte genau mit denen des Phicons übereinstimmen. Stellt man das zweite Phicon hinzu dreht und skaliert der metaDESK die Darstellung der Karte so, dass beide Phicons direkt über dem Objekt stehen, das sie repräsentieren. Verschiebt man die phicons nun, passt sich die Karte deren neuen Standorten an. Wird gleichzeitig die activeLENS verwendet, passt sich auch die dreidimensionale Darstellung immer entsprechend der Lage der Karte an. Verwendet man zwei Phicons ignoriert der metaDESK die Rotation der Landmarken, da die phicons korrekterweise zwei unterschiedliche Kartenrotationen fordern würden. Das sinnvolle Auflösen solcher Mehrdeutigkeiten ist eines der Kernprobleme des parallelen Arbeiten mit solchen Schnittstellen.

9.3.1.2 Ambient Media (ambientROOM)

In der alltäglichen Interaktionen mit der realen Welt nehmen wir Informationen im wesentlichen über zwei Kanäle auf:

1. Einen Großteil der Informationen, die wir bewusst wahrnehmen, stammt aus dem Zentrum unserer Aufmerksamkeit. Dabei kann es sich z.B. um die Worte, Gesten und die Mimik eines Gesprächspartners handeln, oder aber auch um den Bildschirm eines Computers.

2. Gleichzeitig nehmen wir unbewusst viele weitere Informationen über die Umgebung auf. Wir erkennen z.B. das Wetter anhand der Licht-, Geräusch- und Temperaturverhältnisse oder wir nehmen die Aktivitäten unserer Mitmenschen um uns herum wahr, ohne unsere Aufmerksamkeit darauf zu richten.

Abbildung 9.4: ambientROOM

Abbildung 9.4 zeigt den *ambientROOM*, der auf dem Personal Harbor (TM) der Firma Steelcase[1] basiert und um die Fähigkeit Umgebungsmedien darzustellen erweitert wurde. Als Arbeitsplatz wurde in seinem Inneren ein metaDESK platziert. Der ambientROOM ergänzt die grafiklastigen, im Zentrum der Aufmerksamkeit stehenden Interaktionen des metaDESK durch den Einsatz seiner Umgebungsmedien. Sie dienen als Mittel zur Übermittlung von Informationen über die unbewusste Wahrnehmung des Menschen. Ziel des ambientROOM ist zu erforschen, in wie fern Umgebungsmedien die HCI bereichern können.

In einem Testszenario wurden die Testperson in die Rolle eines Automobil-Verkaufsmanagers versetzt. Ihr Ziel war, dass eine Webseite über ein Automodell möglichst oft durch die virtuellen Interessenten aufgerufen wurde. Die Arbeitsfläche war ein an das Szenario angepasster metaDESK, der in dem ambientROOM aufgestellt wurde, und bot der Testperson die Möglichkeit, ihre Seite zu verwalten. Jeder virtuelle Seitenaufruf wurde der Testperson durch das Abspielen eines Tropfgeräusches über die Lautsprecher des ambientROOM mitgeteilt. Dazu tropfte zu jedem virtuellen Seitenaufruf ein Wassertropfen in einen durchsichtigen, von unten angestrahlten Wassertank, so dass die Wasserbrechung an der Decke sichtbar wurde. Während sich die Testperson auf ihre Aufgabe konzentrierte, nahm sie Veränderungen bei den Besucherzahlen der Seite sofort wahr, ohne dass die Umgebungsmedien ihre ganze Aufmerksamkeit beansprucht hätten. Umgebungsmedien können also zur effektiven Informationsübermittlung verwendet werden und helfen gleichzeitig einer Informationsüberflutung vorzubeugen.

[1] www.steelcase.com

9.3.1.3 Interactive Surfaces (transBOARD)

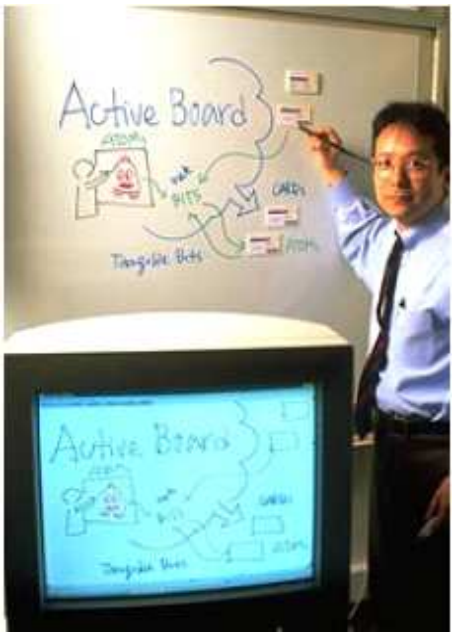


Abbildung 9.5: transBOARD

Das in Abbildung 9.5 dargestellte *transBOARD* dient der Untersuchung des Konzepts der Interaktiven Oberflächen. Es dient als Einwegfilter, indem es Bits aus der realen in die digitale Welt absorbiert. Das transBOARD basiert auf einem Flipchart, dessen Oberfläche von Infrarotscannern abgelesen und in digitale Informationen umgewandelt werden. Dazu unterstützt es den Einsatz von an der Oberfläche haftenden, durch Barcodes eindeutig unterschiedenen Magnetkarten, den so genannten *hyperCARDs*. Im Sinne des Tangible Bits Konzepts handelt es sich dabei um Phicons. Wird eine hyperCARD während einer Zeichensitzung auf die Oberfläche des transBOARDs geheftet, speichert der Server des transBOARDs den aktuellen Beschriftungszustand ab, so dass er durch den Barcode der hyperCARD referenziert werden kann. Dadurch kann ein bestimmter Moment einer Sitzung „mit nach Hause“ genommen werden. Mittels Java[2]-Applet Technologie kann die aktuelle Beschriftung des Flipcharts digital über das Internet ausgelesen werden. Als Beispiel für ein Zusammenspiel des transBOARDs mit dem ambientROOM schlagen die Autoren vor, das Malen auf dem transBOARD durch ein sanftes Kratzen eines Filzstiftes auf einer Wand des ambientROOMs nachzubilden.

9.3.2 Weitere Forschungsprojekte

Dieser Abschnitt stellt einige existierende Tangible Bits vor, anhand deren Forschungseinsätzen die Stärken des Konzepts herausgearbeitet werden. Die Projekte sind nach den Themen Kindererziehung (9.3.2.1), sozialer Interaktion (9.3.2.2) sowie zeit- und sicherheitskritische Anwendungen (9.3.2.3) aufgeteilt.

9.3.2.1 Kindererziehung

Tangible Bits ermöglichen eine wesentlich natürlichere Handhabung von Computern und sind deshalb speziell für Kinder gut geeignet. Folgende Projekte habe den Einsatz von Tangible Bits im Rahmen der Kindererziehung untersucht:

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

Abbildung 9.6: Topobo: Skorpion eines 2.Klässlers

Topobo ist ein Konstruktionssystem für dreidimensionale Modelle, dessen Bauteile kinetische Bewegung aufzeichnen und nachahmen können. Es setzt das Schlüsselkonzept Coupling of Bits and Atoms um. Durch das Zusammenstecken passiver und aktiver Komponenten kann man schnell beweglich Figuren, wie z.B. Tiere (siehe Abb. 9.6) zusammenbasteln. Bewegt man nun die Bausteine des Modells, merken sich diese die Bewegungsabläufe und wiederholen sie immer wieder. Dadurch ist es z.B. möglich einem mit Topobo erstellten Tier das Laufen beizubringen. Das Ziel Topobos ist es, Kindern die Lernfelder Modulare Robotik, Koordination komplexer Systeme und Fortbewegungsfähigkeit nahe zu bringen, die normalerweise erst in höheren Schulstufen gelehrt werden. Die Kinder erlangen so eine besseres Verständnis für die Welt um sie herum. Das Ziel wurde offensichtlich erreicht, da die Kinder während der Untersuchung Topobo erfolgreich einsetzten [RPI04].

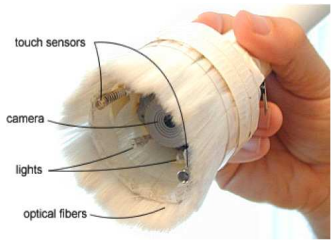


Abbildung 9.7: I/O Brush

I/O Brush ist ein mit dem Computer kommunizierendes Malwerkzeug für Kinder ab vier Jahre. Abbildung 9.7 zeigt den I/O Brush-Pinsel, der eine Kamera sowie Licht- und Tastsensoren besitzt. Damit kann I/O Brush Farben, Texturen und Bewegungen der realen Welt aufnehmen und diese durch Berührung auf ein LCD-Tablet[3] übertragen. I/O Brush setzt damit wie Topobo das Schlüsselkonzept Coupling of Bits and Atoms um. Während Testeinsätzen in Kindergärten versetzte es die Kinder in die Lage komplexe Bilder zu malen und erweckte bei der Suche nach neuen Quellen für Farben und Texturen großes Interesse an ihrer Umgebung in ihnen. Obwohl das Ergebnis digital vorlag, interagierten die Kinder die meiste Zeit mit ihrer Umwelt und ihren Spielkameraden. Die Kinder kamen durch die andere Betrachtungsweise ihrer Umgebung deren grundlegenden, abstrakten Konzepten näher [RMI04].

Abbildung 9.8: Ein Ely geht in den Teleporter

„Ely the Explorer“ ist die Hauptfigur eines interaktiven Spielsystems, das die Schlüsselkonzepte Interactive Surfaces und Coupling of Bits and Atoms umsetzt. In dem Spiel untersuchen die Kinder fremde Kulturen mit Hilfe der so genannten *Elys*, die einerseits als Puppe und andererseits als Computerfigur existieren. Platzieren die Kinder eine Puppe im *Teleporter* (siehe Abb. 9.8), einem verschließbaren Fach im Hauptgerät, erscheint die Figur auf dessen Bildschirm, und die Kinder können ihr bei der Erkundung anderer Kulturen zusehen sowie über eine Touchscreen mit ihr interagieren. Gleichzeitig werden die Kinder dazu animiert, mit einer beiliegenden Digitalkamera Aufnahmen ihrer eigenen Umgebung zu sammeln und in das Gerät einzuspeisen. Das Projekt untersucht das Design von Schnittstellen zur simultanen Kooperation bei der Interaktion von Mensch und Computer. Ein Testeinsatz brachte die Autoren zum Schluss, dass „Ely the Explorer“ eine positiven Beitrag zu existierenden Lernwerkzeugen darstellt. Sie präsentierten aber keine empirisch verwertbaren Ergebnisse. Die Autoren sehen den zukünftigen Einsatz z.B. im therapeutischen Bereich [ALB+04].

[3] http://www.wacom.com/lcdtablets/

9.3.2.2 Soziale Interaktion

Folgende Projekte beschäftigten sich mit den Vorteilen von Tangible Bits bei der menschlichen Interaktion:

Abbildung 9.9: Die TViews Plattform

TViews ist ein Multimediasystem zum Aufzeichnen und Erzählen von Geschichten mit mehreren Handlungssträngen, welches das Schlüsselkonzept Coupling of Bits and Atoms umsetzt. Es besteht aus einer zweidimensionalen Oberfläche (siehe Abb. 9.9), auf die Grafiken projiziert werden können, und bei der mittels Magnetsensoren die Position und Bewegung speziell markierter Objekte verfolgt werden kann. Wird ein Objekt auf der Fläche platziert, wird der damit assoziierte Ausschnitt der Geschichte um die Figur herum projiziert. Ein 10-tägiger Versuch mit Jugendlichen aus Boston brachte den Testpersonen interessante Einblicke über die Sichtweise der Anderen auf gemeinsam erlebte Situationen, während sie parallel an einer Geschichte über ihren jeweiligen Tagesablauf arbeiteten. Das Konzept von TViews kann also eingesetzt werden, um die soziale Kompetenz und das Verständnis für andere zu fördern. Das Tangible Interface von TViews förderte diese Einsichten, da gewöhnliche Eingabegeräte wie Maus und Tastatur das parallele Arbeiten und auch die intuitive Umsetzung kreativer Einfälle eingeschränkt hätten [MD03].

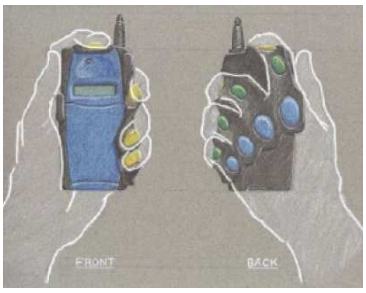


Abbildung 9.10: ComTouch: Audio und Vibration

ComTouch ist ein Set von Funkgeräten (siehe Abb. 9.10), das die akustische Kommunikation um die Übertragung haptischer Signale erweitert, indem der auf ein Gerät ausgeübte Druck auf dem anderen Gerät durch entsprechend starke Vibration ausgedrückt wird. Ziel des Projekts ist, Möglichkeiten zur Bereicherung zwischenmenschlicher Kommunikation mit dem Tastsinn aufzuzeigen. Dieses Projekt ist dem Schlüsselkonzept Ambient Media zuzuordnen, da die Vibrationssignale die Audiosignale lediglich unterstützen und im Idealfall nicht im Fokus der Aufmerksamkeit des Benutzers liegen. Die Entwickler erkannten während Testeinsätzen einen starken Zusammenhang zwischen dem Gesprochenem und den Vibrationssignalen. Selbst Personen, die das Gerät zum ersten mal benutzten, wendeten den neuen Kommunikationskanal erfolgreich an. Sie kommunizierten, indem sie ihre Aussagen durch Vibration betonten oder Signale ihrer Kommunikationspartner nachahmten. Die Frage, wie eine Tastsprache idealerweise aussehen sollte, konnte das Projekt nicht klären. Es weißt jedoch auf die Mächtigkeit der Tastkommunikation hin [COJ+02].

9.3.2.3 Zeit- und sicherheitskritische Anwendungen

RASA ist eine dem in 9.3.1.3 vorgestellten transBOARD ähnliche Interaktive Oberfläche zur Unterstützung der Gefechtsplanung beim Militär. Es erlaubt Militäroffizieren die Gefechtsplanung mittels Papier und Post-it® Notes. Die Post-it® Notes können auf eine Karte geklebt werden, die auf der Interaktiven Oberfläche befestigt ist, und mit Symbolen militärischer Einheiten beschriftet werden. Der mit der Interaktiven Oberfläche vernetzte Computer speichert den aktuellen Zustand des Plans und projiziert eingehende Meldungen über Positionsänderungen von Einheiten auf die Oberfläche. Fällt der Computer aus, wird die Planung anhand der Post-it® Notes und der Papierkarte weiter betrieben. Fährt der Computer wieder hoch, kann die Position anhand der vorhandenen Post-it® Notes aktualisiert werden. Dieses System schützt vor fatalen Folgen durch einen Ausfall des Computers [CM04].

9.3.3 Stärken

Dieser Abschnitt fasst die in den vorigen Abschnitten angedeuteten Stärken und Vorteile von Tangible Bits zusammen. Dies sind:

Verstehen komplexer Zusammenhänge Projekte wie I/O Brush oder Topobo haben gezeigt, dass Menschen komplexe Zusammenhänge durch den Einsatz von Tangible Bits näher gebracht werden können. Deren zugrunde liegende, abstrakte Konzepte werden dank der Tangible Interfaces greifbar und damit leichter verständlich gemacht.

Interaktionsförderung Tangible Bits wie „Ely the Explorer“, TViews oder ComTouch zeigen die Möglichkeiten von Tangible Bits zur Förderung der menschlichen Interaktion auf. Dazu gehören u.a.:

- **Paralleles Arbeiten:** Greifbare Benutzerschnittstellen ermöglichen parallele Eingaben. Dadurch wird auch das Arbeiten mit mehreren Personen an einer Aufgabe möglich. Das Projekt TViews hat gezeigt, dass Menschen so zu sehr interessanten Einsichten über die Wahrnehmung ihrer Mitmenschen gelangen können.
- **Bereicherung der Kommunikation:** Das Projekt ComTouch hat gezeigt, dass die Menschen, die noch nie bewusst haptische Kommunikation eingesetzt haben, intuitiv einem Muster bei deren Einsatz folgen. Es deutet damit das Potential von Tangible Bits zur Bereicherung menschlicher Interaktion via Computer an.
- **Behinderten Förderung:** In [COJ+02] wird darauf hingewiesen, dass die haptische Kommunikation von nicht behinderten wie vom Großteil der behinderten Menschen gleichermaßen genutzt werden kann. Speziell für Menschen mit visuellen oder auditiven Störungen bietet sie eine gemeinsame Interaktionsbasis. Hier liegt ein großes Potential zur Förderung der Kommunikation zwischen Menschen mit visuellen oder auditiven Störungen und anderen Menschen.
- **Vermeidung von Ablenkung:** In [CM04] wird die Eigenschaft grafischer Benutzerschnittstellen kritisiert, die Nutzer von ihrer eigentlichen Aufgabe abzulenken, da deren Bedienung eine für den Menschen ungewohnte und unnatürliche Handlungsweise darstellt und einen nicht unerheblichen Teil seiner Aufmerksamkeit für sich selbst beansprucht. Gleichzeitig lenkt die Sogwirkung, die z.T. von Bildschirmen ausgeht, die Aufmerksamkeit auf den Computer und hemmt damit die interpersonelle Kommunikation. Tangible Bits, die sich an dem natürlichen Umgang des Menschen mit seiner Umwelt orientieren, können helfen, diese Probleme zu lösen.

Steigerung der Bedienungseffizienz Laut [CM04] scheitert die Einführung digitaler Informationssysteme häufig an der Unzulänglichkeit ihrer Benutzerschnittstellen. Effizienzsteigernde Vorteile, die Tangible

Bits gegenüber Informationssystemen mit z.B. grafischen Benutzerschnittstellen bieten, sind:

- **Beibehalten der gewohnten Arbeitsweise:** Das Projekt RASA zeigt, wie eine Arbeitsfläche zu einem Tangible Bit erweitert werden kann, ohne die gewohnte Arbeitsweise zu beeinflussen. Gleichzeitig erhöht die Erweiterung das Potential der Arbeitsfläche beträchtlich.
- **Intuitiveres Verstehen:** Häufig scheitern Nutzer von Computern bei der Umsetzung ihrer Ideen, weil sie seine Bedienung nicht korrekt beherrschen. Tangible User Interfaces verlagern den Umgang mit dem Computer in ein für den Menschen natürlicheres Niveau, so dass dieser seine Erfahrungen aus der physikalischen Welt zum Verstehen der Bedienung verwenden kann.
- **Schnelleres Arbeiten:** Der intuitive Umgang mit dem Computer ermöglicht gleichzeitig schnelleres Arbeiten, da diese Art und Weise mit dem Computer zu interagieren, die natürlichen Fähigkeiten des Menschen besser ausschöpft, z.B. mehrere Eingaben gleichzeitig machen zu können.

Ausfallsicherheit Fällt der durch das Tangible User Interface angesprochene Rechner aus, repräsentiert das Tangible Interface selbst weiter die zuletzt ausgegebenen Daten. Bei dem im Unterabschnitt 9.3.2 vorgestellten Gefechtsplanungssystem RASA bleiben die wichtigsten Informationen zur weiteren Gefechtsplanung auch bei einem Systemausfall erhalten, so dass dieser keine katastrophalen Auswirkungen hat.

Ubiquitous Computing: Mark Weiser schrieb in [Wei91] als Schlusswort:

> *„Most important, ubiquitous computers will help overcome the problem of information overload. There is more information available at our fingertips during a walk in the woods than in any computer system, yet people find a walk among trees relaxing and computers frustrating. Machines that fit the human environment, instead of forcing humans to enter theirs, will make using a computer as refreshing as taking a walk in the woods.“*

Tangible Bits sind allesamt Projekte, welche die von Mark Weiser als kommende Revolution der Informationstechnologie bezeichnete Vision des *Ubiquitous Computings* der Umsetzung ein Stück näherbringen, da sie den Menschen einen natürlicheren Umgang mit Computern ermöglichen.

9.3.4 Einsatzgebiete

Die Stärken von Tangible Bits rücken mehr und mehr ins Bewusstsein der Entwickler von Informationssystemen. [UI01] listet folgende Gebiete auf, in denen sie zunehmend Anwendung finden:

- **Information storage, retrieval, and manipulation:** Die größte Klasse von Anwendungen mit Tangible Interfaces stellt die Assozation digitaler Daten mit physikalischen Artefakten dar. Beispiel dafür sind die in [UI99a] vorgestellte Tangible Bits namens mediaBlocks. Dabei handelt es sich um kleine, hölzerne Blöcke die als Phicons zu Speicherung und Transport von Onlinemedien dienen. Ein mediaBlock wird zur physikalischen Repräsentation des in ihn gespeicherten Mediums. Diese Systeme bergen ein großes Potential zur Verwendung in vernetzten, intelligenten Haushaltsgeräten.
- **Information visualization:** Tangible Bits bieten das Potential reichhaltiger multimodaler Ein- und Ausgabe von Daten. Das in [UI99b] präsentierte URP[4] bietet einen guten Einblick in die Vorteile des Einsatzes von Tangible Bits in diesem Gebiet. Es erlaubt die Planung von Städten, Plätzen, usw. mittels Phicons. Es bietet Städteplanern damit den Komfort des intuitiven und dreidimensionalen Arbeitens und nutzt gleichzeitig das volle Potential der computer-gestützten Städteplanung aus.

[4] Urban Resource Planning

- **Modeling and simulation:** Viele räumliche Schnittstellen und die Klasse der konstruktiven Schnittstellen, repräsentiert durch Projekte wie URP bzw. Topobo zeigen die Stärken des Einsatz von Tangible Bits, da sie die abstrakten Konzepte der Simulation greifbar und damit besser verständlich machen.
- **Systems management, configuration, and control:** Tangible Bits sind ideal um komplexe Systeme zu kontrollieren, konfigurieren und steuern. Beispiel dafür sind die mediaBlocks, die als Steuerung für den ebenfalls in [UI99a] vorgestellten mediaSequencer dienen, indem die in den mediaBlocks enthaltenen Medien vom mediaSequencer zu einer Präsentation zusammengefügt werden.
- **Education, entertainment, and programming systems:** Viele der in 9.3.2 vorgestellten Projekte zeigen die Stärken von Tangible Bits in Erziehung (Ely, Topobo, I/O Brush), Unterhaltung (TViews) und Programmierung (TViews) auf. Bei der Erziehung und der Programmierung liegt die besondere Stärke von Tangible Bits in der Fähigkeit, abstrakte Konzepte greifbar zu machen. Gleichzeitig fördert der hohe Unterhaltungswert die Faszination an der Sache bzw. die pädagogische Wirkung.

9.4 Einsatz im Kontext der Projektgruppe

Der folgende Abschnitt erläutert, in welcher Art und Weise sich Erkenntnisse und Konzepte aus dem Bereich Tangible Bits mit der Aufgabe der Projektgruppe verknüpfen lassen. Zunächst wird dazu die Bedeutung des Arbeitstitels der Projektgruppe: **„Kontext-sensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“** erklärt. Dieser Arbeitstitel legt die Gruppe nicht auf ein bestimmtes Projekt fest, begrenzt aber den Handlungsspielraum durch folgende Aspekte:

- **Kontext-sensitive Umgebungserkundung** bedeutet, dass sich das Projekt mit der Wahrnehmung des aktuellen Kontexts, in dem eine Zielplattform eingesetzt wird, und der angemessenen Reaktion darauf beschäftigen soll.
- **Mobile Unterstützung** meint in diesem Fall eine Implementierung des Projekts auf mobilen Geräten wie PDAs oder Mobiltelefonen.
- **Multimodale Unterstützung** fordert die Unterstützung mehrerer Kommunikationsmodi sowohl zur Ein- als auch zur Ausgabe von Daten.

9.4.1 Einschränkungen durch die Zielplattformen

Dieser Abschnitt erläutert die Grenzen mobiler Plattformen, da die zu realisierende Software auf diesen eingesetzt werden soll. Die Vorgabe, mobile Geräte zu unterstützen, begrenzt die Möglichkeiten bei der Implementierung und damit auch den Einsatz von aus dem Tangible Bits Konzept gewonnenen Erkenntnissen. Limitiert werden die Möglichkeiten durch die Hardware (9.4.1.1) und durch das Betriebssystem/die Laufzeitumgebung (9.4.1.2) bzw. den darauf verfügbaren Funktionalitäten.

9.4.1.1 Hardware

Die Erstellung eigener Hardware ist wegen den fehlenden Kenntnissen, der knappen Zeit quasi ausgeschlossen. Die Projektgruppe ist also gezwungen auf bestehende Hardware zur mobilen Unterstützung von Software zurückzugreifen. Bereitgestellt werden Mobiltelefone und PDAs, die explizit keine Tangible User Interfaces zur Interaktion anbieten. Heutzutage gängige Fähigkeiten und deren Bedeutung sind:

- **Vibration:** Standard in den Mobiltelefonen der aktuellen Generation. Ermöglicht den Einsatz haptischer Kommunikation im Sinne des Schlüsselkonzepts Ambient Media oder kann zur Bereicherung der Kommunikation im Allgemeinen eingesetzt werden (siehe ComTouch, 9.3.2).
- **Audio:** Die Palette der Audiowiedergabe reicht je nach Gerät von monophonen Klangerzeugern über polyphone Klangsynthese bis hin zum Abspielen digital vorliegender Klänge. Je nach dem wie fortgeschritten diese ist, kann man sie sehr effektiv als Umgebungsmedium im Sinne des Schlüsselkonzepts Ambient Media verwenden.
- **Datenübertragung:** Aktuelle mobile Geräte ermöglichen Datenübertragung u.a. über GSM, UMTS, DVB-H, Infrarot, Bluetooth, u.v.m.. Die Datenübertragung ist in zweiter Instanz für Tangible Bits wichtig, weil sie die Verfügbarkeit von zu repräsentierenden Daten erhöht bzw. erst möglich macht.
- **Ortbestimmung:** Manche mobile Geräte ermöglichen das Feststellen der eigenen Position mittels GPS. Will man das mobile Gerät als Phicon einsetzen, kann seine aktuelle Position eine entscheidende Bedeutung für seine Handlungsweise haben. Wegen der groben Auflösung von GPS, unterstützt es jedoch nicht jedes Konzept, das auf Ortsbestimmung beruht.
- **Touchscreen:** Einige mobile Geräte verfügen über Bildschirme mit Drucksensoren. Diese ermöglichen die Verwendung des Geräts als Interaktive Oberfläche, entsprechend dem Schlüsselkonzept von Tangible Bits.

9.4.1.2 Betriebssystem/Laufzeitumgebung

Als Betriebssystem/Laufzeitumgebung für die zu erstellende Software kommen wegen ihrer hohen Verbreitung vor allem das Symbian OS[5] oder die Java 2 Micro Edition[6] in Frage:

- **Symbian OS:** Das in C++ geschriebene Symbian OS ist eines der am weitest verbreiteten Betriebssysteme auf mobilen Endgeräten wie Mobiltelefonen oder PDAs. Es erlaubt den vollen Zugriff auf alle Funktionen eines mobilen Gerätes, die darauf lauffähigen Programme sind jedoch i.d.R. nicht plattformunabhängig.
- **Java 2 Micro Edition (J2ME):** Die meisten mobilen Endgeräte unterstützen die J2ME. Die Java Virtual Machine für die J2ME basiert zumeist auf dem Symbian OS. Programme, die für die J2ME geschrieben sind, können wesentlich plattformunabhängiger eingesetzt werden. Das Zielgerät muss lediglich gleich- oder höherwertige Versionen der verwendeten Bibliotheken wie z.B. MIDP[7] oder CLDC[8] unterstützen.

Verwandte Projekte der Abteilung, in der diese Projektgruppe angeboten wird, wie z.B das in [KKB04] beschriebene AccesSights, verwendeten die dort verfügbare Niccimon-Plattform. Sie stellt eine Sammlung von Programmbibliotheken für mobile Endgeräte dar. Die Projektgruppe sollte ihre Einsatzmöglichkeiten im Rahmen des Projekts evaluieren.

Die Entscheidung für eine Zielplattform beeinflusst die Erreichbarkeit von Spezialfähigkeiten der mobilen Geräte. Die älteren Versionen der J2ME Bibliotheken (MIDP 1.0 und CLDC 1.0) ermöglichen von den in 12.3.3.1 genannten Fähigkeiten lediglich die Datenübertragung via Internet anzusteuern. Um Schlüsselkonzepte von Tangible Bits in diesem Projekt umsetzen zu können, ist ein besserer Zugriff auf die in 12.3.3.1 genannten Fähigkeiten nötig.

[5] http://www.symbian.com/
[6] http://java.sun.com/j2me/
[7] Mobile Information Device Profile
[8] Connected Limited Device Configuration

9.4.2 Tangible Bits im Rahmen der Aufgabe

Dieser Abschnitt enthält Anregungen, in welcher Art mobile Geräte zur Umsetzung der Schüsselkonzepte von Tangible Bits eingesetzt werden können.

- **Coupling of Bits and Atoms:** Die Umsetzung des Schlüsselkonzept Verkupplung von Bits und Atomen ist wegen der feststehenden Gestalt der mobilen Endgeräte nur beschränkt möglich. Denkbar wäre z.B. das mobile Endgerät analog zu der in 9.3.1.1 beschriebenen activeLENS zu verwenden, indem es eine alternative Darstellung zu dem Ort liefert, an dem es sich das Gerät gerade befindet. Problematisch ist dabei die Bereitstellung der korrekten Informationen zu den jeweiligen Orten. Eine andere Idee wäre, das mobile Gerät wie die in 9.3.4 vorgestellten mediaBlocks zum Speichern und Transport von Daten zu verwenden. Wird das Gerät in eine bestimmte Position gebracht, z.B. in die Nähe eine Infrarotschnittstelle, überträgt es automatisch die Daten in den eigenen Festspeicher und gibt sie, wenn sie das nächste mal auf eine solche Infrarotschnittstelle trifft, die Daten an diese weiter.

- **Ambient Media:** Der Einsatz von Umgebungsmedien unterstützt die Umsetzung der Aspekte Multimodalität und kontext-sensitive Umgebungserkundung. Wird die resultierende Anwendung in einem Kontext benutzt, der es dem Nutzer nicht erlaubt seine Aufmerksamkeit auf das Display des Gerätes zu richten, kann sie wichtige Informationen durch den Einsatz von Umgebungsmedien übermitteln. Das Medium ist dabei so zu wählen, dass es die gewünschte Information erfolgreich zum gewünschten Empfänger transportiert, dieser es aber im Idealfall nur unterbewusst wahrnimmt. Damit das Übertragungsmedium möglichst ideal gewählt werden kann, sollte die Palette der verfügbaren Übertragungsmedien des mobilen Gerätes voll ausgeschöpft werden. So wird gleichzeitig der Anforderungsaspekt der multimodalen Unterstützung erfüllt.

- **Interactive Surfaces:** Verfügt ein mobiles Endgerät über Drucksensoren, wie z.B. in einem Touchscreen enthalten, kann es sich durch den Einsatz entsprechender Software zu einer Interaktiven Oberfläche erweitern. So ist ein Touchscreen, auf dem man mittels Stift z.B. Malen oder Schreiben kann, im Sinne der Schlüsselkonzepte von Tangible Bits eine Interaktive Oberfläche.

All diesen Ideen ist gemein, dass sie eine natürlichere und intuitivere Bedienung der aus dem Projekt resultierenden Software ermöglichen würden. Damit erhöht sich der Kreis der Personen, die die Software entsprechend ihres Zwecks effektiv einsetzen können.

9.5 Zusammenfassung

Für Menschen mit wenig Erfahrung im Umgang mit Computern stellt die korrekte Interaktion mit dem Rechner häufig die erste und meist auch entmutigende Hürde dar. Sie scheitern daran, dass ihre Vorstellungskraft, die sie aus der realen Welt mitbringen, nur schwer auf die digitale Welt angewendet werden kann.

Tangible Bits überbrücken die Lücke zwischen der physikalischen und der digitalen Welt durch den Einsatz von Umgebungsmedien und greifbaren Objekten als Benutzerschnittstellen, in Form der Schlüsselkonzepte Interactive Surfaces, Ambient Media und Coupling of Bits and Atoms. Umgesetzt wurden diese Konzepte in den Forschungsprototypen transBOARD, einem interaktiven Whiteboard, ambientROOM, einem Raum mit der Fähigkeit Umgebungsmedien einzusetzen, und metaDESK, einer physikalischen Umsetzung klassischer grafischer Benutzeroberflächen. Ihre Stärken gegenüber grafischen Benutzeroberflächen offenbaren Tangible Bits u.a. bei der Vermittlung komplexer Zusammenhänge, der Interaktionsförderung,

der Steigerung der Bedienungseffizienz und der höheren Ausfallsicherheit. Eingesetzt werden Tangible Bits vermehrt in den Bereichen Informationsspeichung und -darstellung, Modellierung und Simulation, Systemverwaltung und -steuerung, sowie Lehre, Programmierung und Unterhaltung.

Weiterhin evaluierte diese Ausarbeitung die Einsatzmöglichkeiten von Tangible Bits innerhalb der Projektgruppe „eXplorer“, in deren Rahmen sie entstanden ist. Die Möglichkeiten werden einerseits durch die einzusetzende Hardware (PDAs, Mobiltelefone) andererseits durch die darauf verfügbaren Betriebssysteme/Laufzeitumgebungen (Symbian OS, Java 2 Micro Edition) eingeschränkt. Die Ausarbeitung offeriert zu jedem der Schlüsselkonzepte von Tangible Bits mögliche Umsetzungen, die innerhalb dieser Einschränkungen realisierbar sind. Bei einem mobilen Endgerät mit Touchscreen kann dieser leicht zur Interaktiven Oberfläche erweitert werden. Umgebungsmedien können zur angemessenen Reaktion in bestimmten Situationen eingesetzt werden. Wird das mobile Gerät als „greifbares Fenster“ eingesetzt, agiert es damit entsprechend des Schlüsselkonzepts Coupling of Bits and Atoms.

Tangible User Interfaces, die eine Bedienung von Computern analog zur Verwendung eines gleichwertigen Geräts in der realen Welt ermöglichen, helfen unbedarften Nutzer sich schneller in die Bedienung einzufinden, da die alte Verhaltensmuster aus der realen Welt auf die Interaktion mit dem Rechner übertragen können. Infolgedessen sinkt die Berührungsangst mit dem Medium Computer. Man kann die digitalen Ausgaben besser anhand der eigenen Erfahrungen in der realen Welt klassifizieren. Menschen ohne tieferes Verständnis für die Arbeitsweise eine Computers begreifen die laufenden Vorgänge besser. Auch Menschen mit Affinität zu Computern könnten durch die Nutzung von Tangible Bits eine deutliche Steigerung ihrer Arbeitseffizienz erfahren. Langfristig gesehen könnten Tangible Bits ein Baustein zur Umsetzung der von Mark Weiser in [Wei98] beschriebenen dritten Ära des Computerzeitalters, des Ubiquitous Computing sein.

Glossar

Human-Computer Interaction: Interdisziplinäres Feld, das sich mit Design, Evaluation und Implementierung interaktiver Rechensysteme zum Gebrauch durch Menschen und den wichtigsten, sie umgebenden Phänomenen beschäftigt.

Graspable User Interface: Benutzerschnittstelle eines Computers, die die direkte Kontrolle elektronischer oder virtueller Objekte mittels physikalischen Artefakten erlaubt.

Tangible User Interface: Spezialfall von Graspable User Interfaces, bei denen die gleichen Objekte als Ein- und Ausgabegeräte dienen, also dem Abakus-Prinzip gehorchen. Schließt z.T. auch den Gebrauch von Umgebungsmedien ein.

Abakus-Prinzip: Das Zusammenfallen von Steuerung, Darstellung und deren zugrunde liegenden Assoziationen.

Tangible Bits: Name einer von Hiroshi Ishii und Brygg Ullmer formulierten Vision zur Revolutionierung der HCI mit dem Ziel, eine Brücke zwischen der physikalischen und der virtuellen Welt herzustellen.

Interactive Surfaces: Schlüsselkonzept von Tangible Bits; Umwandlung von gewöhnlichen Oberflächen, wie z.B. Wänden zu aktiven Schnittstellen zwischen der realen und der digitalen Welt; Forschungsprototyp transBOARD.

Ambient Media: Schlüsselkonzept von Tangible Bits; Verwendung von Umgebungsmedien wie Geräuschen, Licht, Luftzug und Wasserfluss um die Sinneswahrnehmung des Menschen anzusprechen, die außerhalb seines aktuellen Gedankenfokus liegen; Forschungsprototyp ambientROOM.

Coupling of Bits and Atoms: Schlüsselkonzept von Tangible Bits; nahtloses Verkuppeln alltäglicher Gebrauchsgegenstände (z.B. Bücher) mit den digitalen Informationen, die sie enthalten bzw. repräsentieren; Forschungsprototyp: metaDESK

Phicon: Zusammengesetzt aus den Wörtern „physical“ und „icon“. Stellt ein physikalisches Piktogramm/Symbol analog zu Icons in klassischen graphischen Benutzeroberflächen dar.

Ubiquitous Computing: Vision der dritten Ära des Computerzeitalters, bei der der Mensch wieder im Vordergrund des Denkens steht, und die Computer im Hintergrund auf den Menschen zuarbeiten.

Multimodalität: Die Verwendung mehrerer Kommunikationskanäle, z.B. visuelle und haptische Kommunikation

MIDP: Mobile Information Device Profile; zusammen mit CLDC die Java-Laufzeitumgebung für mobile Geräte.

CLDC: Connected Limited Device Configuration; stellt Methoden zur Kommunikation z.B. mit dem Internet bereit.

Teil II

Ergebnisdokumente der Softwareentwicklung

Kapitel 10

Projektplan

10.1 Einleitung

Dieses Dokument stellt den Projektplan der Projektgruppe *eXplorer* dar. Ziel des Projektplans ist die zeitliche Planung des Projekts, und die Koordination des Ressourceneinsatz. Dazu werden wichtige Gruppen von Teilaufgaben identifiziert und in Arbeitspaketen zusammengefasst. Jedes Arbeitspaket definiert, welche Meilensteine und/oder Ergebnisdokumente zu welchem Zeitpunkt fertiggestellt sein sollen. Zusätzlich beschreibt der Projektplan die Organisationsstruktur der Projektgruppe und stellt die wichtigsten einzusetzenden Arbeitswerkzeuge vor.

Der Projektplan gliedert sich in folgende Abschnitte:

- **Abschnitt 10.1**: dieser Abschnitt
- **Abschnitt 10.2**: erläutert die Struktur, nach der die Projektgruppe intern organsiert ist.
- **Abschnitt 10.3**: stellt die wichtigsten Werkzeuge vor, die in der Projektgruppe zum Einsatz kamen.
- **Abschnitt 10.4**: listet die Arbeitspakete auf, die vor der Entwicklungs des Softwareprodukts durchgeführt wurden.
- **Abschnitt 10.5**: beschreibt die Vorgehensmodelle, die bei der Planung der Softwareentwicklung eine Rolle spielten.
- **Abschnitt 10.6**: beschreibt die Arbeitspaketen der Softwareentwicklung.

10.2 Organisationsstruktur der Projektgruppe

Dieser Abschnitt beschreibt die Struktur, nach der die Projektgruppe organisiert ist. Die Projektgruppe wird durch die Verteilung bestimmter Verantwortungsbereiche auf ihre Mitglieder organisiert. Arbeitspaket 0 listet alle Teilaufgaben auf, die diese Verantwortungsbereiche beschreiben. Bei den folgenden Teilaufhaben handelt es sich um diejenigen, die von einer Person für die Dauer des gesamten Projekts übernommen werden:

T 0.1 Projektsitzungen Die Projektgruppe trifft sich zweimal wöchentlich für jeweils zwei Stunden um die Arbeitsschritte zu koordinieren, Entscheidungen zu treffen und den Fortlauf des Projekts zu planen. Diese Teilaufgabe trifft auf alle Gruppenmitglieder zu.

T 0.2 Projektmanagement Die Norm DIN 69901 definiert entsprechend Projektmanagement als die „Gesamtheit von Führungsaufgaben, -organisation, -techniken und -mitteln für die Abwicklung eines Projektes“. Neben dem rein technischen Können sind die sozialen Fähigkeiten des Projektmanagements sehr wichtig für den Projekterfolg. Gutes Projektmanagement zeichnet sich dadurch aus, dass Problemsituationen mit möglichst wenig Reibungsverlusten gemeistert werden. Alle Mitglieder der Projektgruppe nehmen am Projektmanagement teil. Dazu managen die jeweils Verantwortlichen ihr Arbeitspaket, während der Projektmanager sich mit ihnen abstimmt, den groben Überblick über den Stand des Projekts behält und den Projektplan entsprechend wartet. Das Ergebnisdokument *Projektplan* dieser Teilaufgabe liegt mit diesem Dokument vor.

T 0.3 Sitzungsmoderation Der Sitzungsmoderator macht sich Gedanken zum Ablauf der Sitzungen, erstellt entsprechende Tagesordnungen und moderiert die halbwöchentlichen Sitzungen.

T 0.4 Qualitätssicherung Der Beauftragte zur Qualitätssicherung wacht über den Qualitätsstandard der Software. Er sollte dabei einen möglichst untechnischen Blickwinkel einnehmen, weil die meisten Abstriche in der Qualität aus Problemen mit der Technologie resultieren werden, und er so möglichst unbeeinflusst davon bleibt. Weiterhin ist es die Aufgabe des Qualitätsmanagers, die von der Gruppe geschriebenen Dokumente zu evaluieren, um Fehler jeglicher Art zu berichtigen. Während der Implementierung plant der Beauftragte zusammen mit dem Qualitätsmanager die Integrationstests und den Abschlusstest und führt diese durch.

T 0.5 Webauftritt Der Webadministrator erstellt und pflegt die Webseite, auf der die Gruppe ihr Projekt der Öffentlichkeit präsentiert.

T 0.6 Dokumentation Der Dokumentenwart erstellt Vorlagen für Ergebnisdokumente und sorgt für deren passende Aufbewarung. Er kann außerdem Richtlinien für alle weiteren Dokumente erstellen, damit die Gruppe ein einheitliches Erscheinungsbild bekommt.

T 0.7 Evaluation Der Qualitätsmanager plant die Evaluierung der Software und führt diese durch.

T 0.8 Eventmanagement Der Eventmanager plant und koordiniert Events, wie Kneipentreffs oder Sportveranstaltungen.

T 0.9 TeX Der Beauftragte für LaTeX ist Ansprechpartner bei Problemen im Umgang mit der Satzsprache LaTeX, mit der die Ergebnisdokumente der Projektgruppe erstellt werden sollen.

T 0.10 UML Der entsprechende Beauftragte ist Ansprechpartner bei Problemen mit UML, mit der die Architektur des System beschrieben werden soll.

Geschätzer Zeitaufwand

Aufgabe	Beauftragter	Zeitaufwand
Projektsitzungen	alle	4 Std./Woche
Projektmanagement	MP	2 Std./Woche
Sitzungsmoderation	SK	2 Std./Woche
Qualitätssicherung	OL	1 Std./Woche
Webauftritt	FJ	1 Std./Woche
Dokumentwart	AB	1 Std./Woche
Evaluation	JP	1 Std./Woche
Eventmanagement	SA	1 Std./Woche
TeX	DN	1 Std./Woche

10.3 Werkzeugeinsatz

Dieser Abschnitt beschreibt die Hard- und Software Werkzeuge, die bei der Erstellung der Dokumente und der Software zum Einsatz kommen.

10.3.1 Dokumentation

MiKTeX MiKTeX ist eine LaTeX-Distribution für Microsoft Windows Betriebssysteme. Die meisten Programme der Distribution liegen als Kommandozeilenversionen vor, daher ist zum Schreiben des LaTeX-Quelltextes ein Texteditor notwendig. Ein Editor, der für MiKTeX entworfen wurde und häufig parallel genutzt wird, ist TeXnicCenter.

TeXnicCenter TeXnicCenter ist ein Texteditor, der speziell für die Zusammenarbeit mit der MiKTeX-Distribution entworfen wurde. Der Texteditor markiert die LaTeX-Syntax farbig und kann über Icons oder Menü LaTeX-Befehl einfügen. Er bietet den Vorteil, dass man die erstellten Dokumente durch einen einfachen Knopfdruck als PDF-, DVI- oder PS-Datei ausgeben kann.

Visio 2002/2003 Visio ist ein weit verbreitetes Illustrationsprogramm der Firma Microsoft. Visio dient dazu, verschiedene Abläufe visuell am PC darzustellen. Hierzu kann man sich verschiedenen Vorlagen mit passenden Werkzeugen und Symbolen für Flussdiagramme, Geschäftsprozesse und Diagrammen bedienen und diese per Drag and Drop auf ein Dokument ziehen. Das eigentlich kostenpflichtige Programm kann durch Mitglieder des Departments für Informatik umsonst über die Microsoft Developer Network - Academic Alliance bezogen werden.

10.3.2 Webauftritt

Apache Web-Server Der Apache HTTP Server ist ein Produkt der Apache Software Foundation und der meistverbreitete Webserver im Internet. Neben Unix und Linux läuft Apache auch auf Win32, NetWare sowie einer Vielzahl weiterer Betriebssysteme. Der Apache bietet die Möglichkeit, mittels serverseitiger Skriptsprachen Webseiten dynamisch zu erstellen. Häufig verwendete Skriptsprachen sind PHP oder Perl. Diese sind kein Bestandteil des Webservers, sondern müssen ebenfalls entweder als Module eingebunden werden oder über die CGI-Schnittstelle angesprochen werden.

10.3.3 Versionsmanagement

CVS Concurrent Versions System (CVS) bezeichnet ein Programm zur Versionsverwaltung von Dateien, hauptsächlich Software-Quellcode. CVS ist ein reines Kommandozeilen-Programm, aber es wurde für alle gängigen Betriebssysteme mindestens eine graphische Oberfläche für CVS entwickelt, zum Beispiel WinCVS für Windows. CVS vereinfacht die Verwaltung von Quellcode dadurch, dass es alle Dateien eines Software-Projektes an einer zentralen Stelle, einem so genannten Repository, speichert. Dabei können jederzeit einzelne Dateien verändert werden, es bleiben jedoch alle früheren Versionen erhalten, einsehbar und wiederherstellbar, auch können die Unterschiede zwischen bestimmen Versionen verglichen werden. Somit hilft CVS dabei, einen Überblick über die einzelnen Versionen der Dateien und die dazugehörigen Kommentare zu behalten und kann insbesondere verwendet werden, um bei größeren Projekten die Arbeit der verschiedenen Entwickler eines Projektes zu koordinieren.

10.3.4 Softwareentwicklung

Visual Studio 2003 .NET Visual Studio .NET ist eine integrierte Entwicklungsumgebung (IDE) von Microsoft für Windows. Visual Studio .NET ermöglicht dem Programmierer, Programme für Windows (insbesondere für das .NET Framework), das Web und Mobile Geräte zu schreiben. Visual Studio .NET zeichnet sich seit seinen ersten Versionen vor allem durch die Integration verschiedener Programmiersprachen in eine Entwicklungsumgebung aus. Bekannteste Beispiele sind Visual Basic und Visual C++. Visual Studio 2003 .NET kann wie Visio 2002 durch Mitglieder des Departments für Informatik umsonst über die Microsoft Developer Network - Academic Alliance bezogen werden.

Pocket PC SDK 2003 Dieses Software Developement Kit (SDK) erlaubt die Erstellung von Anwendungen für den Pocket PC 2003 in den Programmiersprachen C-Sharp, Visual Basic .NET, sowie emBedded Visual C++ 4.0 mit Service Pack 3.

10.3.5 Entwicklungshardware

Pocket PC iPAQ 5450 iPAQ ist eine Produktlinie von Personal Digital Assistants (PDAs), die ursprünglich von Compaq hergestellt wurden. Seit der Übernahme von Compaq durch Hewlett-Packard werden sie unter dem HP Label verkauft. Die Hardware wird freundlicherweise von Universität und OFFIS für die Dauer des Projekts zur Verfügung gestellt.

Als Betriebssystem kommt Pocket PC von Microsoft zum Einsatz, das ähnlich zu den bekannten Windows Betriebssystemen von Desktop PCs zu benutzen ist. Die Bedienung erfolgt über einen Touchscreen, oft benötigte Anwendungen werden auf separate Knöpfe gelegt (z.B. Kalender, Adressen, etc). Die Eingabe von Text erfolgt über eine Bildschirmtastatur auf dem Touschscreen oder über Handschrifterkennung. Durch Aufsteckmodule (sog. Jackets) können diese PDAs um zusätzliche Funktionen erweitert werden.

FALCOM NAVI GPS Empfänger FALCOM NAVI ist ein mit einer Bluetooth Schnittstelle ausgerüsteter GPS Empfänger. Die empfangenen GPS Daten werden mittels des NMEA Protokolls[1] über die Bluetooth Schnittstelle übertragen.

[1] http://www.kh-gps.de/nmea-faq.htm

10.3.6 Qualitätssicherung

Mantis Mantis ist ein web-basiertes Bugtracking System. Es wurde in PHP geschrieben und benötigt eine MySQL Datenbank sowie einen Web-Server, um eingesetzt werden zu können. Der Zugriff erfolgt mittels Browser. Das Bugtracking System dient dazu, Bugs, d.h. Fehler in der Software, an einer zentralen Stelle zu sammeln und zu verwalten. Fehler durchlaufen verschiedene Bearbeitungszustände, wie z.B. *anerkannt, behoben, geschlossen.* Zu jedem Zustand wird ein Kommentar vorgehalten, so dass der Fehler sowie seine Lösung nachvollzogen werden kann.

NUnit NUnit ist ein Software-Framework zum Unit-Testing für alle .NET Sprachen. Ursprünglich portiert von JUnit ist die 2005 aktuelle Version 2.2 komplett in C-Sharp neu geschrieben und unterstützt .NET spezifisch. NUnit basiert wie alle Unit-Testing-Frameworks auf XUnit.

10.4 Arbeitpakete Vorarbeiten

In diesem Abschnitt befinden sich die Arbeitspakete, die die Einarbeitung in das Thema und die Suche nach einem konkreten Projekt umfassen. Er gliedert sich in folgende Teile:

Arbeitpaket A - Seminarphase enthält die Teilaufgaben der Seminarphase.

Arbeitpaket B - Entwicklung konkurrierender Produktszenarien beschreibt die Teilaufgaben zur Entwicklung konkurrierender Produktszenarien.

Arbeitpaket C - Produktskizze umfasst die Teilaufgaben zur Anfertigung der Produktskizze.

10.4.1 Arbeitspaket A - Seminarphase

Arbeitspaket Nummer	A	**Startdatum**	01.09.2004
Personenmonate	18	**Ende**	06.11.2004

Ziele

Die Seminarphase dient der Projektgruppe dazu, in die wissenschaftliche Thematik des Projekts einzusteigen. Durch die Beschäftigung mit den Themen aus dem vorgegebenen wissenschaftlichen Bereich, sammeln die Teilnehme Spezialkenntnisse. Diese werden dazu verwendet, um die Idee für ein konkretes Projekt innerhalb des vorgegebenen Themenbereichs zu entwickeln. Jedes Gruppenmitglied verfasst in diesem Arbeitspaket eine Seminararbeit zu einem bestimmten Thema und präsentiert sie der Gruppe an einem gemeinsamem Seminarwochenende. Die Teilaufgaben werden sequentiell durchgeführt.

Beschreibung der Teilaufgaben

T A.1 Aufbereitung eines wissenschaftlichen Themas Jedes Gruppenmitglied wählt ein Thema aus den von den Betreuern vorgeschlagenen Themen aus bzw. schlägt ein eigenes Thema aus dem vorgegebenen Bereich vor. Der Teilnehmer führt eine eigenständige Recherche zu dem gewählten Thema durch und erstellt dazu eine Seminarausarbeitung.

T A.2 Präsentation des Spezialthemas Die Gruppenteilnehmer arbeiten zu ihren Ausarbeitungen Vorträge aus und präsentieren sie während einens gemeinsamen Seminarwochenendes.

Ergebnisdokumente/Meilensteine

E A.1: Aufbereitetes Thema in Form einer 15-20 seitigen Ausarbeitung.

E A.2: Folien zur Präsentation des Themas

M A.3: Seminarwochenende auf Norderney 6./7.11.2004

10.4.2 Arbeitspaket B - Szenariofindung

Arbeitspaket Nummer	B	**Startdatum**	07.11.2004
Personenmonate	11	**Ende**	12.12.2004

Ziele

Ziel von Arbeitspaket B ist die Formulierung mehrerer Szenarien, in denen die beteiligten Akteure durch den Einsatz von Kontext-sensitiver, mobiler und multimodaler Software unterstützt werden könnten. Anhand dieser Szenarien sollen Softwareprodukte skizziert werden. In einer Abstimmung entscheidet sich die Gruppe für eines der Szenarien, welches im weiteren Verlauf der Projektgruppe analysiert, entworfen und implementiert werden soll. Die Teilaufgaben werden sequentiell durchgeführt.

Eingabe

Die Eingabe für dieses Arbeitspaket ist die aus der Seminarphase (Arbeitspaket A) gewonnene Expertise.

Beschreibung der Teilaufgaben

T B.1 Entwicklung mehrerer Szenarien In dieser Phase sollen Szenarien entwickelt werden, in denen der Einsatz von Kontext-sensitiver, mobiler und multimodaler Software sinnvoll erscheint. Dabei handelt es sich um die kreativste Phase des gesamten Projekts. Technische Aspekt sollten bei den Überlegungen keine Rolle spielen.

T B.2 Auswahl eines Szenarios Die in B.1 entwickelten Szenarien treten in dieser Phase gegeneinander an, mit dem Ziel, möglichst viele Gruppenteilnehmer von sich zu überzeugen. Am Ende mehrerer Präsentationen und Kritikphasen entscheidet sich die Gruppen für ein Szenario, aus dem ein Softwareprodukt entwickelt werden soll.

Ergebnisdokumente/Meilensteine

- **E B.1** Anwendungsszenarien
- **E B.2** Produktskizzen
- **M B.3** Festlegung auf ein zu entwickelndes Produkt

10.4.3 Arbeitspaket C - Produktskizzierung

Arbeitspaket Nummer	C	**Startdatum**	12.11.2004
Personenmonate	04	**Ende**	06.01.2005

Ziele

In Arbeitspaket C verfeinert die Projektgruppe das in Arbeitspaket B ausgewählte Szenario zu einer Produktskizze. Die Teilaufgaben werden sequentiell durchgeführt. *Anmerkung: Die Gruppe entschied sich für das Szenario „Agentenspiel“. Die folgenden Arbeitspakete sind zum Teil entsprechend angepasst worden.*

Eingabe

Eingabe sind das von der Gruppe am Ende von Arbeitspaket B zur Verfeinerung ausgewählt Szenario und das fachliche Wissen aus Arbeitspaket A.

Beschreibung der Teilaufgaben

T C.1 Erstellen der Produktskizze In dieser Teilaufgabe erstellt die Gruppe anhand des in Arbeitpaket B ausgewählten Szenarios eine Produktskizze. Dabei wird vor allem Wert auf die Definition der primären Use-Cases gelegt, da sie in der folgenden Teilaufgabe als Basis für einen ersten Test dienen.

T C.2 Validierung des skizzierten Produkts In dieser Teilaufgabe wird das Produkt einem virtuellen Test unterzogen. Dazu erstellt die Gruppe Skizzen der wichtigsten Ansichten auf Papier. Die Betreuer spielen anhand dieser Skizzen die Bedienung der Software durch. Die gewonnenen Erfahrungen fließen wiederum in die Produktskizze ein.

Ergebnisdokumente/Meilensteine

- **E C.1** Testergebnisse in Form von Evaluationsbögen
- **E C.2** Der Verlauf des Tests wird ausserdem als Video dokumentiert, welches ebenfalls als Ergebnisdokument gewertet wird und zur späteren Analyse des Testverlaufs herangezogen werden kann.
- **M 1.1** Mit dem Abschluss dieser Teilaufgabe hat sich die Projektgruppe auf ein Produkt festgelegt. Die folgenden Phasen dienen nun dazu, die genauen Anforderungen zu analysieren, das Produkt zu entwerfen und es zu implementieren.

10.5 Vorgehensmodelle der Projektphase

Die aus den Vorarbeiten (Abschnitt 10.4) hervorgegangene Produktskizze, soll nun über die verbleibende Zeit des Projekts zu einem Produkt weiterentwickelt werden. Dieser Abschnitt beschreibt die Vorgehensmodelle, die zur Umsetzung der Produktskizze und zu Planung des weiteren Projektverlaufs verwendet werden.

10.5.1 Vorgehensmodell der Softwareentwicklung

Die Entwicklung der Produktsoftware aus der Produktskizze orientiert sich am Wasserfallmodell. Das Wasserfallmodell, erstmals vorgestellt 1970 in [Roy70], ist das traditionelle Vorgehensmodell der Softwareentwicklung. Im diesem Modell hat jede Phase wohldefinierte Start- und Endpunkte mit eindeutig

definierten Ergebnissen. Erweiterungen des Modells erlauben, in die vorangehenden Phasen zurück zu springen, um eventuelle Fehler und Probleme auszumerzen. In seiner ursprünglichen Version besteht es aus fünf Phasen:

1. **Anforderungsanalyse und -spezifikation** (en. Requirement analysis and specification)
   Ziel der Anforderungsanalyse ist die schriftliche Fixierung der Anforderungen an das Softwaresystem. Das resutlierende Dokument wird Anforderungsdefinition genannt und dient als Vertrag zwischen Kunde und Entwicklern.

2. **Systemdesign und -spezifikation** (en. System design and specification)
   In der Phase Systemdesign und -spezifikation wird die Umsetzung der während der Anforderungsdefinition fixierten Anforderungen geplant und dokumentiert. Das resultierende Dokument findet man z.B. unter der Bezeichnung Entwurf.

3. **Programmierung und Modultests** (en. Coding and module testing)
   In dieser Phase wird der Entwurf zu der eigentlichen Software umgesetzt. Gleichzeitig werden in dieser Phase die einzelnen Funktionseinheiten (Module) getestet.

4. **Integration und Systemtests** (en. Integration and system testing)
   In dieser Phase werden die implementierten und getesteten Funktionseinheiten zum Gesamtsystem zusammengesetzt und das Zusammenspiel getestet.

5. **Auslieferung, Einsatz und Wartung** (en. Delivery and maintenance)
   Hat die Software die Tests bestanden, kann sie an den Kunden ausgeliefert werden. Dies bedeutet für Software jedoch selten das Ende des Projekts, da Software meist regelmäßgie Wartung erfordert.

Abschnitt 10.6 beschreibt die Umsetzung des Wasserfallmodells in Form von Arbeitspaketen. Jede Phase wird genau einem Arbeitspaket zugeordnet. Zusätzlich gibt es zwei Arbeitspakete zwischen Anforderungsanalyse und Systemdesign, in der die Machbarkeit des Projekt überprüft wird. Das Einfügen dieser Arbeitspakete ist sinnvoll, da sich das Projekt im Rahmen von wenig etablierten und risikobehafteten Technologien bewegt.

10.5.2 Besonderheiten der Implementierungsphase

Um den aus den wenig etablierten Technologien resultierenden Risiken zu begegnen, werden spezielle Methoden zur Sicherung der Softwarequalität angewendet. Es werden Methoden adaptiert, die u.a. im Extreme Programming (siehe [Bec99]) Anwendung finden. Dazu gehören:

- **Testgetriebenes Programmieren**: Jede Zeile Code wird durch einen Testfall motiviert, der zunächst fehlschlägt. Die Implementierung hat das Ziel, die durch den Testfall festgesetzten Anforderungen zu erfüllen. Die Unit Tests werden gesammelt, gepflegt und nach jedem Kompilieren ausgeführt.

- **Kurze Iterationen**: Die Entwicklung erfolgt in Perioden von ein bis drei Wochen. Am Ende jeder Iteration steht ein funktionsfähiges, getestetes System mit neuer Funktionalität.

- **Pair-Programming**: Die Programmierer arbeiten stets zu zweit am Code und diskutieren während der Entwicklung intensiv über Entwurfsalternativen. Dadurch erfährt der Code ein kontinuiierliches Review.

- **Refactoring**: Das Design des Systems wird ständig in kleinen, funktionserhaltenden Schritten verbessert. Finden zwei Programmierer Code-Teile, die schwer verständlich sind oder unnötig kompliziert erscheinen, verbessern und vereinfachen sie den Code. Die Unit-Tests sichern zu, dass das System nach den Änderungen weiterhin lauffähig ist.
- **Fortlaufende Integration**: Das System wird mehrmals täglich neu gebaut. Der entwickelte Code wird regelmäßig in die Versionsverwaltung eingecheckt und ins bestehende System integriert. Die Unit Tests müssen zur erfolgreichen Integration alle bestehen.
- **Einfachheit/Simplicity**: Jeder Testfall wird auf die einfachst denkbare Weise erfüllt. Es wird keine unnötig komplexe Funktionalität programmiert, die momentan nicht gefordert ist. Als Affirmation dient das YAGNI[2]-Prinzip.
- **Gemeinsame Verantwortlichkeit**: Der gesamte Code gehört dem Team. Jedes Paar soll jede Möglichkeit zur Codeverbesserung jederzeit wahrnehmen. Das ist kein Recht sondern eine Pflicht.

Sobald die grobe Struktur des Entwurfs in Code umgesetzt ist, wird anwendungsfall-orientiert implementiert, d.h. die Teilaufgaben der Implementierung orientieren sich nicht an der Umsetzung einzelner Module, sonder beinhalten die Umsetzung eines Anwendungsfalls über alle beteiligten Module hinweg.

Abbildung 10.1 auf Seite 160 zeigt den aktualisierten Implementierungsplan. Jede Ellipse steht für einen komplexen Anwendungsfall. Je höher der Anwendungsfall platziert ist, um so höher ist seine Priorität. Anwendungsfälle, deren Elipsen durchgezogene Linien haben, fallen unter die Kategorie der „must-haves“, d.h. diese Anwendungsfälle müssen auf jeden Fall implementiert werden. Anwendungsfälle, deren Elipsen gestrichelte Linien haben, sind „should-haves“, sollten also nach Möglichkeit umgesetzt werden.

[2] Abkürzung für: „You ain't gonna need it“

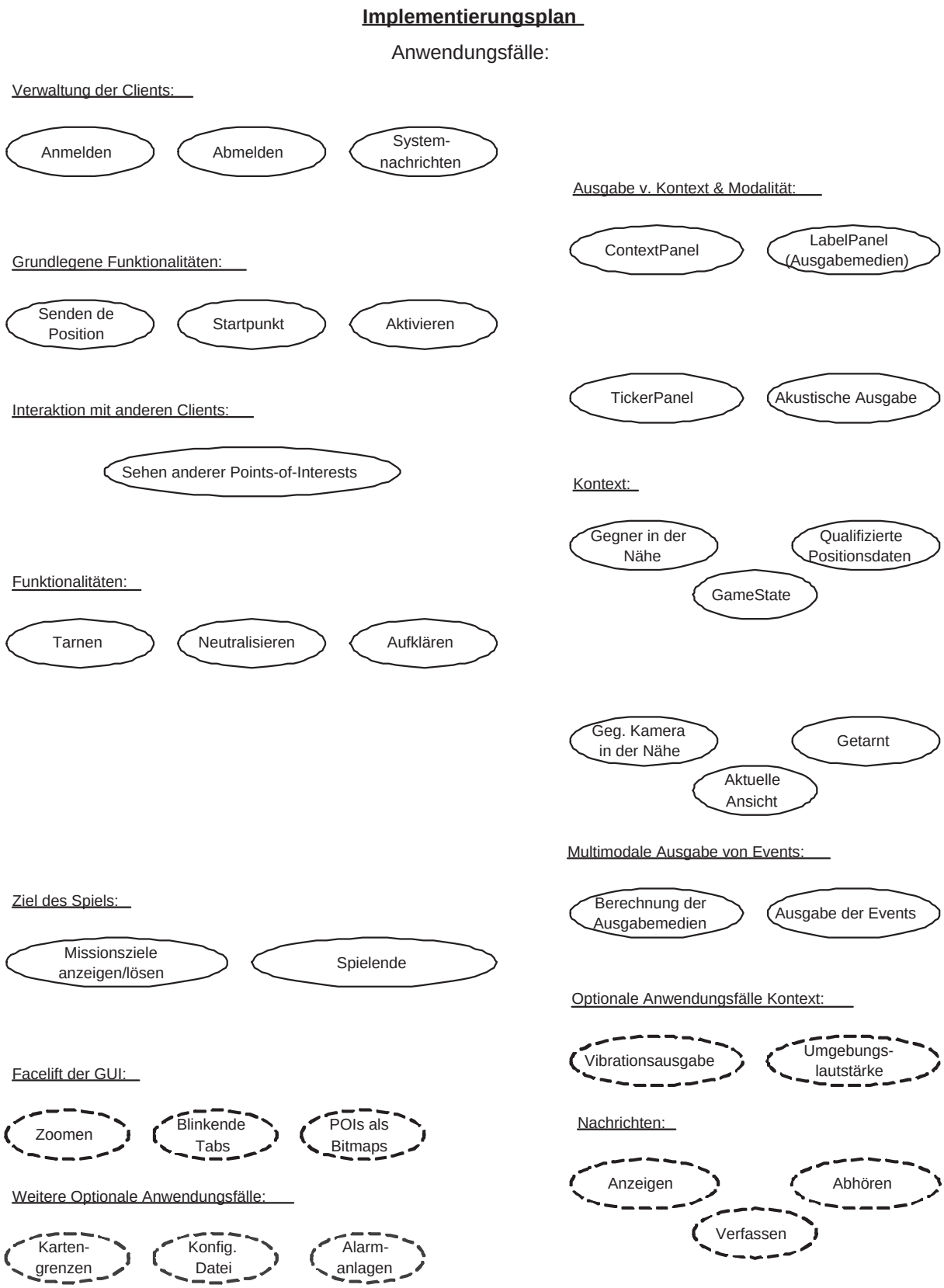


Abbildung 10.1: Implementierungsplan

10.6 Arbeitpakete des Projekts

Dieser Abschnitt enthält alle Arbeitspakete, die Teilaufgaben zur Umsetzung des skizzierten Produkts enthalten. Das Projekt ist in folgende Arbeitspakete gegliedert:

Arbeitpaket 0 - Jahresaufgaben enthält die Aufgaben, die die Gruppenteilnehmer für die Dauer des gesamten Projekts übernehmen.

Arbeitpaket 1 - Anforderungsdefinition enthält Teilaufgaben zur Erstellung der Anforderungsdefinition.

Arbeitpaket 2 - Fallstudie beschreibt die Teilaufgaben der Fallstudie.

Arbeitpaket 3 - Technologiestudie beschreibt die Teilaufgaben der Technologiestudie.

Arbeitpaket 4 - Entwurf enthält die Teilaufgaben, die den Entwurf der Software umfassen.

Arbeitpaket 5 - Qualitätsmanagement beschreibt die Teilaufgaben, die beim Qualitätsmanagement der entwickelten Artefakte anfallen.

Arbeitpaket 6 - Implementierung beschreibt die Teilaufgaben, in die sich die Implentierung gliedert.

Arbeitpaket 7 - Projektabschluss beschreibt die Teilaufgaben, die den Abschluss des Projekts umfassen.

10.6.1 Arbeitspaket 1 - Anforderungsdefinition

Arbeitspaket Nummer	1	**Startdatum**	06.01.2005
Personenmonate	03	**Deadline**	17.01.2005

Ziele

Ziel des 1. Arbeitspakets ist die Formulierung der Anforderungen des Agentenspiels. Sie setzen den verbindlichen Rahmen für die folgenden Entwicklungsschritte.

Eingabe

Die Eingaben für dieses Arbeitspaket umfasst die in den Seminararbeiten (D A.1) gewonnene Expertise der Projektgruppenteilnehmer, sowie die Ergebnisse der Produktskizzen (D B.1) und der Vorüberlegungen für das Agentenspiel (D C.1).

Beschreibung der Teilaufgaben

Die einzelnen Tasks beinhalten die Bearbeitung der Teilaspekte der Anforderungsdefinition.

T 1.1 Ausgangssituation und Zielsetzung Diese Teilaufgabe beschreibt das Ziel der Projektgruppe und umreißt den Inhalt des zu erstellenden Spiels.

T 1.2 Systemeinsatz und Systemumgebung Diese Teilaufgabe beschreibt die äußeren Bedingungen, in der die Software eingesetzt wird. Wichtigste Aspekte sind die speziellen Anforderungen an den mobilen Einsatz der Software sowie der zu erwartenden Benutzergruppe.

T 1.3 Funktionale Anforderungen Teilaufgabe 1.3 definiert die funktionalen Anforderungen an das resultierende System. Diese setzen sich zusammen aus den Anwendungsfällen, dem geforderten Verhalten in Fehlerfällen und der Analyse der zu verwalteten Daten.

T 1.4 Nichtfunktionale Anforderungen Teilaufgabe T 1.4 arbeitet alle weiteren Anforderungen heraus, die keine Funktionalitäten sondern reine Eigenschaften des Systems darstellen. Die wichtigsten nichtfunktionalen Anforderungen an das resultierende System sind Multimodalität und kontextsensitives Verhalten.

T 1.5 Benutzungsschnittstelle In dieser Teilaufgabe wird das Basislayout der Benutzungsschnittstelle festgelegt. Die wichtigsten Bedienelemente werden skizziert und ihre Funktion beschrieben. Dabei ist die Usability ein wichtiger zu beachtender Aspekt.

T 1.6 Dokumentationsanforderungen Teilaufgabe T 1.6 benennt alle Dokumente, die im Laufe der Produktentwicklung entstehen.

Ergebnisdokumente/Meilensteine

- **E 1.1** Anforderungsdefinition, Fertigstellung bis 17.01.2005

10.6.2 Arbeitspaket 2 - Fallstudie

Arbeitspaket Nummer	2	**Startdatum**	17.01.2005
Personenmonate	14	**Deadline**	31.03.2005

Ziele

Ziel des 2. Arbeitspaketes ist die Erarbeitung einer Fallstudie, durch die eine erste Struktur der zu implementierenden Software (Agentenspiel) beschrieben wird. Im Laufe dieser Entwicklung werden Programmmodule identifiziert, die als Ideensammlung und Vorlage für das Arbeitspaket 4 (Entwurf) verwendet werden sollen. Gute Ideen und Konzepte sollen dabei in die Arbeiten zum Entwurf mit einfließen, die Ergebnisse dieses Arbeitsschrittes sind jedoch für das Arbeitspaket 4 (siehe Kapitel 10.6.4 auf Seite 165) nicht bindend. Nach der Erarbeitung der Struktur wird überprüft, ob dieses erste Konzept zur Erfüllung der nichtfunktionalen Anforderungen an das Agentenspiel geeignet ist.
Aus den Überlegungen zum Konzept für den Entwurf des Agentenspiels werden die Programmmodule herausgefiltert, die als Framework für die Entwicklung Kontext-sensitiver, umgebungserkundender, multimodaler und mobiler Software verwendet werden können.

Eingabe

Die Eingaben für dieses Arbeitspaket sind die durch die Anforderungsdefinition festgelegten funktionalen und nichtfunktionalen Anforderungen an das Agentenspiel.

Beschreibung der Teilaufgaben

Die einzelnen Tasks beinhalten die Bearbeitung der Teilaspekte der Fallstudie.

T 2.1 Basiskonzept In dieser Teilaufgabe sollen die grundlegenen Module des Agentenspiels identifiziert werden, die zur Umsetzung der Anforderungen benötigt werden. Es soll umgangssprachlich beschrieben werden, welche Aufgaben sie erfüllen. Die Projektgruppe arbeitet in Kleingruppen verschiedene Vorschläge aus, anhand derer dann ein möglichst optimales Basiskonzept erarbeitet werden soll.

T 2.2 Programmmodule Diese Teilaufgabe verfeinert die in T 2.1 gewonnenen Erkenntnisse. Hier sollen die Funktionalitäten der in T 2.1 identifizierten Module abstrakt beschrieben werden. Jedes Modul wird einem oder mehreren Mitgliedern der Gruppe (je nach Anzahl der Module) zur Bearbeitung zugewiesen. Die Ergebnisse helfen bei der Auswahl geeigneter Technologien zur Realisierung des Agentenspiels.

T 2.3 Funktionalitäten d. Frameworks Hier sollen einige Gruppenmitglieder die Funktionalitäten des Agentenspiels identifizieren, die für andere Kontext-sensitive, mobile und multimodale Anwendungen interessant sein könnten. T 2.3 abstrahiert das in T 2.1 erstellte Basiskonzept. Ergebnis soll eine Liste der Funktionalitäten sein, die das Framework anbieten soll.

T 2.4 Datenhaltung In T 2.4 sollen die von den einzelnen Modulen verwalteten Daten genau analysiert werden. Dies soll schwerpunktmäßig in der ersten Woche geschehen, da laut Ian Sommerville die Daten während der Programmkonzipierung weniger Fluktuationen unterworfen sind, als die Funktionalitäten. T 2.4 ist wichtig, um später eine klare Schnittstellendefinition zu erzielen und damit Fehler und Unstimmigkeiten während des Entwurfs und der Implementierung zu vermeiden.

T 2.5 Umsetzung der Spielidee T 2.5 überprüft und verfeinert die Ergebnisse von T 2.2. T 2.5 soll zusichern, dass sie Spielidee mit der erarbeiteten Fallstudie umgesetzt werden kann, und dass keine für das Spiel benötigten Daten/Funktionalitäten vergessen wurden.

T 2.6 Umgebungsmodell, Nutzermodell, Kontextmodell In dieser Teilaufgabe werden - ähnlich wie in T 2.5 die funktionalen Anforderungen - die nicht-funktionalen Anforderungen überprüft. Die Leitfragen dabei lauten:

- „In welcher Umgebung wird gespielt und wie beeinflusst sie das Spiel?“
- „Welche Menschen benutzten das System und wie sollte sich das auf die Interaktion mit der Anwendung auswirken?“
- „Welche besonderen Anforderungen werden durch verschiedene Spielsituationen an den Spieler und die Software gestellt?“

T 2.7 Interaktionsdesign In dieser Teilaufgabe wird ermittelt, welche Kanäle dem Benutzer zur Interaktion mit der Software angeboten werden soll. Daraufhin wird überprüft, ob die Architektur die Erstellung dieser Benutzerschnittstellen zulässt.

T 2.8 Informationsdesign In dieser Teilaufgabe wird ermittelt, welche Kanäle dem Benutzer der Software zum Austauschen von Informationen angeboten werden soll. Daraufhin wird überprüft, ob die Architektur die Erstellung dieser Informationskanäle zulässt.

Ergebnisdokumente/Meilensteine

- **E 2.1** Die Fallstudie

10.6.3 Arbeitspaket 3 - Technologiestudie

Arbeitspaket Nummer	3	**Startdatum**	01.02.2005
Personenmonate	09	**Deadline**	31.03.2005

Ziele

Ziel des 3. Arbeitspakets ist die Identifizierung von Technologien die zur Realisierung des Agentenspiels verwendet werden sollen. Relevante Technologien müssen gesichtet, diese anhand eines Kriterienkatalogs bewertet und der Gruppe präsentiert werden.

Eingabe

Keine explizit vorgesehenen Eingaben.

Beschreibung der Teilaufgaben

T 3.1 Sichtung vorhandener Technologien Diese Teilaufgabe umfasst die Sichtung vorhandener Technologien, die sich zur Umsetzung des Basiskonzepts eignen könnten.

T 3.2 Kriterienkatalog In T 3.2 soll ein Kriterienkatalog erstellt werden, anhand dessen sich die Gruppe bei der Abschätzung relevanter Technologien orientieren soll.

T 3.3 Abschätzung relevanter Technologien In T 3.3 werden die in T 3.1 gesichteten Technologien auf ihre Relevanz bezüglich des Projekts diskutiert. Ziel ist es die Technologien zu finden, die mittels Technikreferaten ausgearbeitet werden sollen.

T 3.4 Ausarbeitung von Technikreferaten In dieser Teilaufgabe untersucht die Gruppe die in T 3.3 ausgewählten Technologien, dokumentiert diese und bereitet die Ergebnisse für eine Präsentation vor der Gruppe auf. Zur jeder Technologie soll, wenn möglich, ein Prototyp erstellt werden, um die Machbarkeit zu überprüfen.

T 3.5 Präsentation der Technikreferate Die in T 3.4 erstellten Technikreferate sollen der Gruppe präsentiert werden. Die Präsentation soll dabei die Dauer von 20 Minuten nicht überschreiten. Zu jeder Präsentation werden zusätzlich 10 Minuten für Diskussion und Umbau eingeplant. Der Präsentierende soll dabei die Technologien bewerten und eine Empfehlung aussprechen.

T 3.6 Auswahl einzusetzender Technologien Anhand der in T 3.5 gewonnen Erkenntnisse entscheidet die Gruppe mit welchen Technologien das Agentenspiel implementiert werden soll.

Ergebnisdokumente/Meilensteine

- **E 3.1** Kriterienkatalog bis 23.02.2005
- **E 3.2** Aus T 3.4 resultierende Technikreferate, Stichtag ist 16.03.2005
- **M 3.3** Festlegung einzusetzender Technologien bis 31.03.2005

10.6.4 Arbeitspaket 4 - Entwurf

Arbeitspaket Nummer	4	**Startdatum**	01.04.2005
Personenmonate	14	**Deadline**	31.05.2005

Ziele

Im 4. Arbeitspaket erstellt die Gruppe den Feinentwurf zum Agentenspiel. Der Entwurf soll zur Koordination der gemeinsamen Implementierung beitragen und diese erheblich vereinfachen.

Eingabe

Die Ergebnisse der Arbeitspakete 1 bis 3 fließen in den Entwurf ein.

Beschreibung der Teilaufgaben

T 4.1 Design Pattern Zur Bearbeitung dieser Aufgabe erhält jedes Gruppenmitglied mehrere Design Pattern zugeteilt, die der Gruppe kurz zusammengefasst präsentiert und erläutert werden sollen. Dazu werden die Erläuterungen anhand einer Vorlage zu einer großen Powerpointpräsentation zusammengefasst. Durch das Auseinandersetzen mit Design Pattern sollen die Teilnehmer qualifiziert werden, eine bessere Architektur zu entwerfen und später weniger Zeit für eventuelles Re-Engeneering aufwenden zu müssen.

T 4.2 Fachliches Modell In dieser Teilaufgabe identifiziert die Gruppe die Programmzustände des Agentenspiels und gibt einen Überblick darüber, in wiefern sich die Zustände gegenseitig beeinflussen. Außerdem soll ein Überblick über das Spiel, den Spielablauf inklusive Spielregeln und die Interaktionsmöglichkeiten der Spieler mit dem Spiel gegeben werden. Ergebnis ist ein Regelwerk, anhand dessen die Programmlogik des Agentenspiels entworfen und verifiziert werden kann.

T 4.3 Architekturentwurf In T4.3 soll die Architektur des Agentenspiels entworfen werden. Zunächst sollen dazu die Einflussfaktoren auf die Architektur idenifiziert und analysiert werden. Darauf soll eine Analyse der größten Risiken und die Beschreibung der Lösungsstrategien folgen. Außerdem sollen die Umgebung, in der das System laufen wird, die einzelnen Architekturbausteine und die Interaktion zwischen diesen identifiziert werden.

T 4.4 Entwurf der Nutzungsschnittstelle In dieser Teilaufgabe wird die Nutzungsschnittstelle für das Agentenspiel entworfen. Diese muss eine multimodale Nutzerinteraktion erlauben und sich bei der Auswahl der Modalitäten für Ein- und Ausgabe von Informationen am jeweiligen Kontext des Nutzers orientieren. Als Grundlage für die Identifizierung des Kontexts dienen hierbei die in T4.2 identifizierten Programmzustände.

T 4.5 Feinentwurf des Agentenspiels In dieser Teilaufgabe erstellt die Gruppe den Feinentwurf für das Agentenspiel. Dazu werden die in T4.3 identifizierten Architekturbausteine verfeinert und alle öffentlichen und geschützten Attribute und Methoden dieser Bausteine dokumentiert. Ergebnis dieser Teilaufgabe sind Klassendiagramme sowie textuelle Beschreibungen zu jedem Modul und seinen Hilfsklassen.

T 4.6 Re-Engineering Werden während der Implementierung große Änderungen der Architektur oder der Klassenstruktur nötig, wird deren Re-Engineering dieser Teilaufgabe zugeordnet. Im Idealfall wird diese Teilaufhabe jedoch nicht wahrgenommen werden müssen.

Ergebnisdokumente/Meilensteine

- **E 4.1** Design Pattern Folien, bis 07.04.2005
- **E 4.2** Entwurf des Agentenspiels, bis 31.05.2005

10.6.5 Arbeitspaket 5 - Qualitätsmanagement

Arbeitspaket Nummer	5	**Startdatum**	01.04.2005
Personenmonate	9	**Ende**	30.09.2005

Ziele

Dem Qualitätsmanagement (QM) fällt bei der Durchführung von Softwareprojekten eine entscheidende Rolle zu. Neben der Überprüfung von Dokumenten auf Korrektheit und Konsistenz, sowie der Validierung und Verifikation der erstellten Software, nimmt das QM heutzutage ebenfalls Einfluss auf das Projektmanagement.
Das Ziel dieses Arbeitspakets besteht darin, die Projektgruppe frühzeitig auf Schwachstellen der verwendeten Hard- und Software aufmerksam zu machen, Werkzeuge und Vorgehensweisen zur Validierung des erstellten Quellcodes vorzustellen, sowie die Testphase nach Abschluss der Implementierungsphase zu planen.

Eingabe

Als Eingaben dieses Arbeitspaketes dienen die Ergebnisse der Technologiestudien.

Beschreibung der Teilaufgaben

T 5.1 Machbarkeitsstudie (Hardware) Das primäre Ziel dieser Teilaufgabe besteht darin, die Tauglichkeit der zur Verfügung stehenden Hardware, in Hinblick auf die Realisierung der Funktionalitäten des Agentenspiels zu untersuchen. Dadurch soll frühzeitig festgestellt werden, welche Funktionalitäten eventuell nicht oder nur eingeschränkt umsetzbar sein werden. Stehen mehrere Hardware-Lösungen zur Verfügung, besteht ein weiteres Ziel dieser Teilaufgabe darin, die Vor- und Nachteile dieser Hardware-Lösungen einander gegenüberzustellen, um dadurch die geeignetste identifizieren zu können.

T 5.2 Machbarkeitsstudie (Software) In der Technologiestudie (Arbeitspaket 3) werden unterschiedliche Technologien hinsichtlich ihrer Verwendungsmöglichkeit für das Agentenspiel untersucht. Das

Ziel dieser Studien besteht darin, die Vor- und Nachteile der unterschiedlichen Technologien gegeneinander abzuwägen und so, die für die Umsetzung des Agentenspiels geeignetsten Technologien zu identifizieren. Innerhalb dieser Teilaufgabe werden die Ergebnisse der Technologiestudien in vertiefenden Machbarkeitsstudien näher untersucht, wobei das Hauptaugenmerk auf Seiten der Software liegt. Durch die Erstellung von Prototypen soll hierbei getestet werden, ob und mit welchem Aufwand die untersuchten Technologien für die Umsetzung des Agentenspiels genutzt werden können. Darüber hinaus bietet die Erstellung von Prototypen den Teilnehmern der Projektgruppe die Möglichkeit, sich bereits vor der eigentlichen Implementierungsphase mit den verschiedenen Entwicklungswerkzeugen vertraut zu machen.

T 5.3 Modultests Innerhalb dieser Teilaufgabe werden Techniken vorgestellt, die von der Projektgruppe während der Implementierungsphase zur Durchführung von Modultest (auch bekannt als *Unit*-Tests oder Komponententests) eingesetzt werden sollen. Hierzu werden das NUnit-Framework vorgestellt und die Funktionsweise des so genannten Logging in C# erläutert.

T 5.4 Integrationstests Das Ziel dieser Teilaufgabe besteht darin, das Zusammenspiel der einzelnen Softwaremodule zu testen. Als Orientierungshilfe für die Erstellung von Integrationstests dienen hierbei die im Implementierungsplan auf Seite 160 dargestellten Anwendungsfälle. Die Reihenfolge, in der die einzelnen Integrationstest durchzuführen sind, wird im so genannten Testplan festgehalten. Als Vorlage für diesen Testplan und den Aufbau der Testprotokolle der einzelnen Integrationstests dient der IEEE Standard 829-1998.

T 5.5 Abschlusstest Innerhalb dieser Teilaufgabe werden komplette Spielabläufe durchlaufen. Die Spielabläufe setzen sich dabei aus einer Reihe von Integrationstests zusammen und müssen detailliert geplant werden. Das Ziel dieser Teilaufgabe besteht zum einen darin, mögliches Fehlverhalten bei längerer Spieldauer zu identifizieren, sowie die *Spielbarkeit* des Agentenspiels generell zu bewerten.

Ergebnisdokumente/Meilensteine

- **E 5.1** Machbarkeitsstudien (Hardware), Stichtag 12.05.05
- **E 5.2** Machbarkeitsstudien (Software), Stichtag 12.05.05
- **E 5.4** Protokolle der Integrationstests, Stichtag 18.09.05
- **E 5.5** Protokolle der Abschlusstests, Stichtag 30.09.05

10.6.6 Arbeitspaket 6 - Implementierung

Arbeitspaket Nummer	6	**Startdatum**	01.05.2005
Personenmonate	27	**Ende**	18.09.2005

Ziele

In der Implementierung werden die Ergebnisse der Entwurfsphase in maschinenverständlichen Code umgesetzt. Für bessere Lesbarkeit werden zunächst Coding Conventions aufgestellt, die eine einheitliche Struktur und Schreibweise im gesamten Programmcode gewährleisten sollen. Das wichtigste Ziel ist, nach Ende der Implementierung auch tatsächlich ein lauffähiges und möglichst stabiles Produkt vorliegen zu haben. Aus diesem Grund werden die einzelnen im Entwurf konzipierten Module und Anwendungsfälle

priorisiert und erst mit der Implementierung weniger wichtiger Programmteile begonnen, wenn die wichtigeren Funktionalitäten fertiggestellt sind. Neben der Erstellung des Programmcodes spielt auch dessen Dokumentation eine wichtige Rolle. Diese erfolgt selbstverständlich in Form von Kommentaren im Code, darüber hinaus aber auch in der Beschreibung weiterer Besonderheiten wie Konfigurationsmöglichkeiten, Funktionsweisen ausgewählter Programmteile, sowie vorgenommenen Optimierungsmaßnahmen.

Eingabe

Als Eingabe der Implementierung dienen vorwiegend die Ergebnisse des Entwurfs. Allerdings wirken sich auch die Erkenntnisse aus den Technologie- und insbesondere der Machbarkeitsstudien auf die Implementierung aus.

Beschreibung der Teilaufgaben

T 6.1 Coding Conventions Ziel dieser Teilaufgabe ist die Verständlichkeit und Wiederverwendbarkeit des im Anschluss zu erstellenden Programmcodes zu gewährleisten. Im Einzelnen wird hier einheitlich festgelegt, wie die Paket- und Klassenstruktur umzusetzen ist, welche Ordnung innerhalb einzelner Klassen vorgesehen ist, in welcher Form Kommentare eingefügt werden sollen und welche Richtlinien zur Namensgebung einzuhalten sind.

T 6.2 Modulpriorisierung In diesem Bereich werden die einzelnen Anwendungsfälle den Prioritäten *must-haves*, *should-haves* und *nice-to-haves* zugeordnet, um sie anschließend in der Reihenfolge dieser Einteilung zu implementieren. So soll sichergestellt werden, dass immer eine möglichst stabile Version vorliegt und nicht gegen Projektende ein Produkt mit zwar sehr vielen aber unausgereiften Features vorliegt.

T 6.3 Implementierung In der Implementierungsphase wird der Programmcode geschrieben. Hierbei wird zunächst aus den Ergebnissen des Entwurfs ein Klassengerüst angelegt. Dieses wird dann unter Einhaltung der Prioritäten mit den Funktionalitäten der einzelnen Anwendungsfälle gefüllt.

Ergebnisdokumente/Meilensteine

- **E 6.1** Coding Conventions, Stichtag 15.05.05
- **E 6.1** Modulpriorisierung, Stichtag 22.05.05
- **E 6.3** Dokumentation der Implementierung (Konfiguration, Besondere Algorithmen, Optmierungsmaßnahmen), Stichtag 30.09.05

10.6.7 Arbeitspaket 7 - Projektabschluss

Arbeitspaket Nummer	7	**Startdatum**	01.08.2005
Personenmonate	02 + X	**Ende**	30.09.2005

Ziele

Der Projektabschluss ist die letzte Phase des Projektes. Er beinhaltet die Übergabe der Projektergebnisse, woraus eine Abnahme resultiert.

Eingabe

Die Eingabe in dieses Arbeitspaket sind die Ergebnisse der vorhergehenden Arbeitspakete.

Beschreibung der Teilaufgaben

T 7.1 Abschlußanalyse, Erfahrungssicherung und Abschlußsitzung Folgende Themen gehören zu einer Projektabschlußsitzung:

1. Rückschau
   - Was war gut?
   - Was war weniger gut?
   - Welche Ziele wurden erreicht/nicht erreicht?
2. Anerkennung und Kritik
3. Erfahrungssicherung für künftige Projekte
   - Was kann aus dem Projektverlauf gelernt werden?
   - Welche Maßnahmen werden konkret getroffen, um Fehler nicht zu wiederholen?
4. Information über den Projektabschluß
   - Wer bekommt den Abschlußbericht?
   - Wer wird nur kurz über den Projektabschluß informiert?
5. Abschlußfeier

Außerdem empfiehlt es sich, ein Projekt mit einer systematischer Sicherung der gewonnenen Erfahrungen abzuschließen. Das Sammeln entsprechender Daten ist die Basis für das Bilden von Kennzahlen sowie den Aufbau eines Kennzahlensystems. Das Einrichten von Erfahrungsdatenbanken ist dabei besonders geeignet zur Erfahrungssicherung, weil hiermit die Erkenntnisse aus unterschiedlichen Entwicklungsbereichen über einen längeren Zeitraum in eine gemeinsame Datenbasis zusammengeführt werden. Das Sammeln von Erfahrungsdaten stellt außerdem eine wichtige Voraussetzung für das Kalibrieren von Aufwandsschätzverfahren dar. Ohne eine konsequente Erfahrungssicherung ist ein wirkungsvolles Wissensmanagement nicht möglich.

T 7.2 Abschlußbericht Folgender Aufbau empfiehlt sich für den Projektabschlußbericht:

- Projektauftrag/Projektziel
- Planungsunterlagen vor Projektfreigabe
- Ist-Unterlagen bei Projektende
- Abschlußanalyse: Gegenüberstellung von ursprünglichen und im Projektverlauf aktualisierten Plangrößen mit den Ergebnissen bei Projektende bezüglich
  - Termine, Aufwände, Kosten
  - inhaltlicher Zielerreichung.
- Dank an alle Beteiligten für deren Mitwirkung am Projekt.
- Ansprechpartner zur weiteren Betreuung des Projekts

T 7.3 Präsentation Präsentation des Projekts auf dem PG-Tag im Januar und am 13.10. vor den Betreuern.

T 7.4 Projektauflösung Letzter Schritt in der Projektabschlussphase und damit im gesamten Projektablauf ist die Projektauflösung.

Ergebnisdokumente/Meilensteine

- **E 7.1**
  - Erfahrungsberichte 13.09.05
  - Projektanalyse, Stichtag 28.09.05
  - Projektabschlußsitzung, Stichtag 29.09.05
- **E 7.2**
  - Webseite, Stichtag 15.09.05
  - Abschlußbericht, Stichtag 20.09.05
  - Reviewphase, Stichtag 27.09.05
- **E 7.3**
  - Präsentation (Betreuer), Stichtag 13.10.05
  - Präsentation, PG-Tag, Januar 2006
- **E 7.4** Projektauflösung, Stichtag 13.10.05

Kapitel 11

Anforderungsdefinition

11.1 Einleitung

Dieses Kapitel stellt die Anforderungen an das im Rahmen der Projektgruppe eXplorer entwickelte Spiel *NABB* dar. Im folgenden wird die Systemumgebung beschrieben und alle nichtfunktionalen und funktionalen Anforderungen aufgeführt. Die funktionalen Anforderungen sind in Anwendungsfälle unterteilt, die in ihrer Summe das Verhalten der Anwendung festlegen.

Das Kapitel gliedert sich in folgende Abschnitte:

Abschnitt 11.1: Liefert einen kurzen Überblick des Inhalts dieses Kapitels.

Abschnitt 11.2: Beschreibt Ausgangssituation und Zielsetzung, mit Motivation und Beschreibung des Spielinhalts.

Abschnitt 11.3: Definiert den Systemeinsatz und die Systemumgebung.

Abschnitt 11.4: Enthält die funktionalen Anforderungen, gegliedert in Anwendungsfälle und die Datenhaltung.

Abschnitt 11.5: Enthält die nichtfunktionalen Anforderungen bezüglich Kontextsensitivität, Multimodalität und Fehlertoleranz der Anwendung.

Abschnitt 11.6: Beschreibt die Anforderungen an die Benutzungsschnittstelle.

Abschnitt 11.7: Enthält die Dokumentationsanforderungen für Dokumente für Anwender und Entwickler.

11.2 Ausgangssituation und Zielsetzung

Im Rahmen der Projektgruppe „eXplorer“ soll ein Spiel für mehrere Spieler für mobile Geräte entwickelt werden. Die Grundidee des Spiels wird im Abschnitt „Spielinhalt“ erläutert. Zentral verwaltet wird das Spielgeschehen von einem Server, der die Aktionen der Teilnehmer koordiniert und auswertet. Jeder Spieler benötigt ein mobiles Endgerät (Client), das mit dem Server via Mobilfunk in Verbindung steht.

Die Umwelt in der sich die Spieler bewegen, soll in das Spielgeschehen integriert werden. Ausserdem stellt die Mobilität der Spieler besondere Anforderungen an die Bedienbarkeit der Software, da die Steue-

rung einen flüssigen Spielablauf möglichst wenig stören soll. Die Erstellung neuer und die Umgestaltung bestehender Spielszenarien soll unabhängig von einer Änderung des Programmcodes möglich sein.

11.2.1 Spielinhalt

Zur Durchführung des Agentenspiels werden die Teilnehmer in zwei Gruppen unterteilt, die gegeneinander antreten. Das Spielfeld ist ein durch ein Szenario festgelegtes Gelände, in dem sich die Spieler frei bewegen können. Es gibt Missionsziele, die durch das Szenario festgelegt sind. Eine Gruppe (das *Infiltrator*-Team) hat die Aufgabe diese Ziele zu erfüllen, das andere Team (*Defender*-Team) muss diese Gruppe an der Erfüllung der Ziele hindern, um das Spiel zu gewinnen. Jeder Spieler kann während des Spiels Aktionen (Tarnen, Aufklären, Abhören, Neutralisieren) ausführen, durch die er Einfluss auf den Spielverlauf ausübt. Die Stärke einer jeweiligen Aktion kann ein Spieler zu Spielbeginn festlegen, indem er dieser Funktion Eigenschaftspunkte zuteilt. Während des Spielverlaufs kann sich die Eigenschaft einer Aktion verändern, in dem ein Spieler virtuelle Objekte („Goodies“) auf dem Spielfeld findet und aufnimmt. Die Spieler eines Teams können ihr Vorgehen durch das Versenden von Nachrichten untereinander koordinieren.

11.2.2 Motivation

Das Ziel unserer Produktentwicklung ist die Erstellung einer Software, die von Personen zur Gestaltung ihrer Freizeit genutzt wird. Die Durchführung des Spiels soll in erster Linie Spaß bereiten. Die Teilnehmer können sich in einem festgelegten Gebiet bewegen und miteinander kommunizieren, um gemeinsam eine bestimmte Aufgabe zu erfüllen. Im Vordergrund steht dabei ein spielerisches Kräftemessen, bei dem es auf Fähigkeiten wie Koordination, Kooperation, Schnelligkeit und Geschicklichkeit ankommt.

Die Motivation zur Durchführung dieses Projektes besteht darin ein Spiel zu entwickeln, das auf der einen Seite rechnerbasiert verwaltet und gesteuert, auf der anderen Seite aber von den Spielern in der realen Welt gespielt wird. Der kontinuierliche und schnelle Spielverlauf stellt eine große Herausforderung dar, weil die Bedienung der Software so konzipiert werden muss, dass ein Spieler mit seinem mobilen Endgerät interagiert, ohne wesentlich vom eigentlichen Spielgeschehen abgelenkt zu werden. Außerdem sollen die Möglichkeiten mobiler Kommunikation und Positionsbestimmung anhand einer konkreten Anwendung erprobt werden.

11.3 Systemeinsatz und Systemumgebung

Die Steuerungssoftware des Spiels ist in zwei Teile aufgeteilt, der Client läuft auf den mobilen Geräten der Spieler, die Koordination des Spielgeschehens übernimmt ein Spielserver. Dieser soll auf einem stationären PC mit Internetanbindung betrieben werden.

Die Nutzung des Spiels setzt die Positionsbestimmung von Mitspielern voraus und arbeitet somit mit vorhandener Infrastruktur im Bereich der Positionsbestimmung (z.B. GPS, Wireless Lan etc.) zusammen. Ausserdem werden während des Spiels zwischen den mobilen Endgeräten bzw. zwischen mobilem Endgerät und Spielserver Daten übertragen. Auch hier soll vorhandene Infrastruktur zur drahtlosen Datenübertragung (z.B. GPRS, Bluetooth etc.) zum Einsatz kommen.

Der primäre Nutzer ist der Spieler, für den die Anwendung eine Freizeitaktivität darstellt. Ihm soll eine simple aber multimedial reichhaltige Interaktion geboten werden. Einige dieser Spieler können Administratorrechte erhalten, die ihnen die Möglichkeit geben, den Spielserver zu verwalten und so z.B. Spieleinstellungen vorzugeben und zu verändern.

11.4 Funktionale Anforderungen

Die funktionalen Anforderungen sind in zwei Bereich unterteilt.

In dem Kapitel *Anwendungsfälle* (11.4.1) werden alle Funktionen der Software strukturiert aufgeführt. Dabei beschreibt jeder Anwendungsfall eine Aktion im Spielverlauf. In der Summe ergeben die Anwendungsfälle den Funktionsumfang, den die Spielsoftware abdecken muss.

Das Kapitel *Datenhaltung* (11.4.2) beschreibt die von der Software verwalteten Daten und wie diese strukturiert und verwaltet werden.

11.4.1 Anwendungsfälle

Neben der textuellen Beschreibung werden die Anwendungsfälle auch grafisch verdeutlicht. Abbildung 11.4.1 zeigt die Anwendungsfälle, die mit dem unmittelbaren Spiel zusammenhängen. Abbildung 11.4.1 zeigt die Anwendungsfälle, die mit der vorausgehenden Charaktergestaltung in Beziehung stehen und Abbildung 11.4.1 fasst die Anwendungsfälle zusammen, die zur Auswahl eines Spiels gehören. Es folge die Auflistung aller Anwendungsfälle:

Anwendungsfall
: Systemnachricht

Akteure
: Server, Spieler

Vorbedingungen
: Keine

Beschreibung
: Fällt dem Server ein Fehlverhalten eines Spielers auf, so wird dieser gewarnt sein Fehlverhalten zu unterlassen.

Parameter, Variablen
: 1. Eindeutiger Bezeichner des Empfängers
: 2. Nachricht selbst

Nachbedingung
: Spieler ist über Fehlverhalten aufgeklärt

Anwendungsfall
: Bestrafung eines Spielers

Akteure
: Server, Spieler

Vorbedingungen
: Spieler wurde bereits über Fehlverhalten informiert

Beschreibung
: Stellt ein Spieler trotz Warnung sein Fehlverhalten nicht ein, so bekommt er eine Bestrafung.

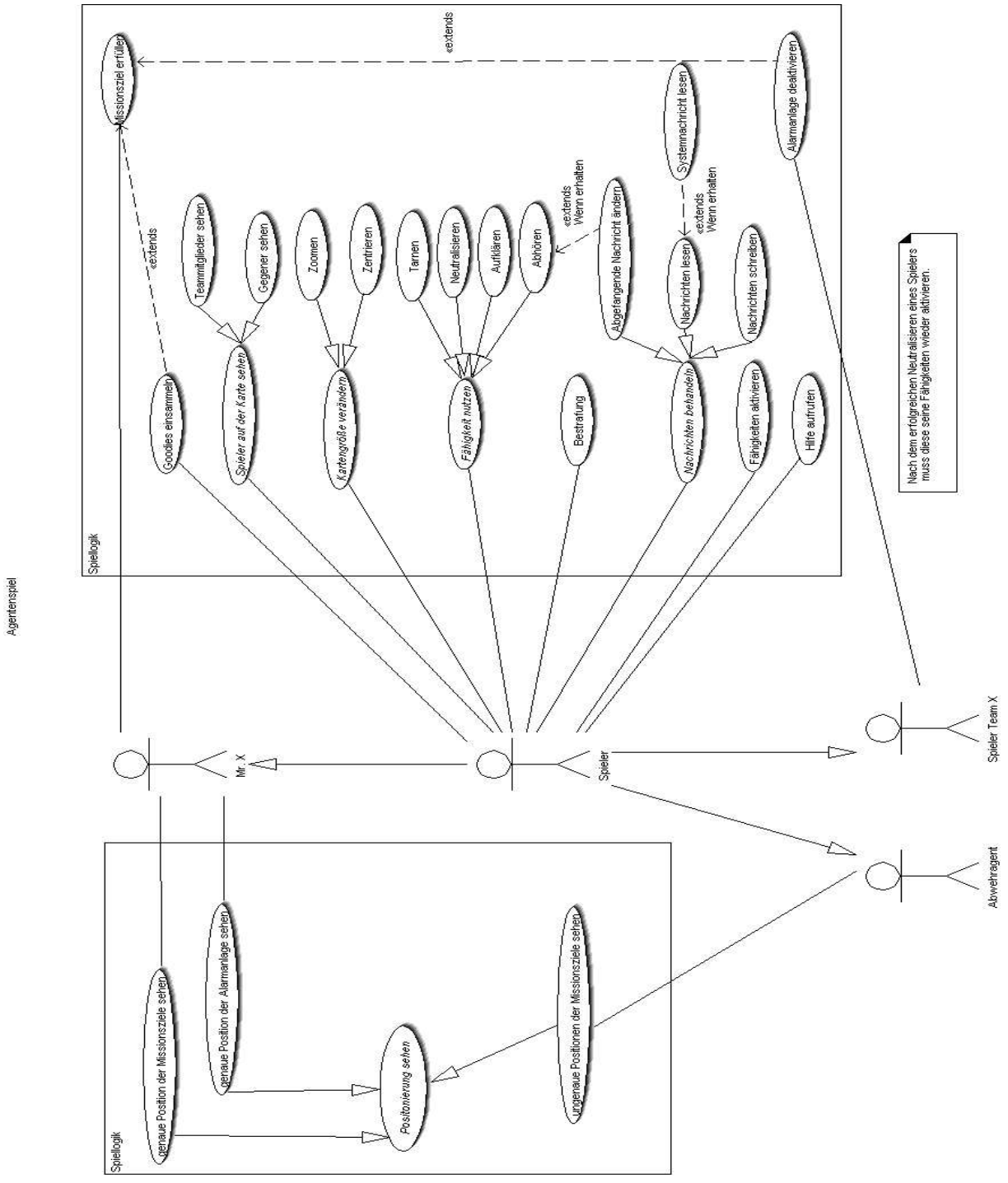


Abbildung 11.1: Anwendungsfälle zum Agentenspiel

Parameter, Variablen

1. Eindeutiger Bezeichner des Spielers
2. Die Strafe

Nachbedingung

Die Strafe wurde ausgesprochen

Anwendungsfall

Nachricht schreiben

Akteure

Spieler

Vorbedingungen

Keine

Beschreibung

Ein Spieler kann einem Mitspieler eine Nachricht in Form von Text, Sprache oder Grafiken senden.

Parameter, Variablen

1. Eindeutiger Bezeichner des Empfängers
2. Eindeutiger Bezeichner des Senders
3. Nachricht selbst

Nachbedingung

Empfänger hat Nachricht vom Sender erhalten

Anwendungsfall

Nachricht lesen

Akteure

Spieler

Vorbedingungen

Keine

Beschreibung

Empfangene Nachrichten werden in einer Liste gespeichert. Einzelne Nachrichten können aus dieser Liste ausgewählt und dann gelesen, weitergeleitet oder gelöscht werden.

Parameter, Variablen

1. Nachricht selbst

Nachbedingung

Nachricht wurde bearbeitet

Anwendungsfall
Nachricht abhören
Akteure
Spieler
Vorbedingungen
Spieler nicht im Tarnmodus, ausreichend Abhörpunkte verfügbar
Beschreibung
Ein Spieler kann über die Abhörfunktion versuchen auch solche Nachrichten zu empfangen, die nicht an ihn adressiert sind. Der Erfolg eines Abhörversuchs hängt von der Anzahl der verfügbaren Fähigkeitspunkte der Eigenschaft Abhören ab. Weiterhin spielt die Abhörfähigkeit des Senders in die Erfolgsbestimmung rein (Hacker sind schwerer zu hacken). Abhören ist nicht räumlich, sondern nur zeitlich begrenzt, d.h. es können zwar alle Nachrichten unabhängig vom Absendeort abgehört werden, allerdings verbraucht ein Abhörversuch Abhörpunkte, die sich bei Inaktivität wieder regenerieren.
Parameter, Variablen
1. Abhörfähigkeit des Abhörers
2. Abhörfähigkeit des Senders
3. Bei Erfolg die Nachricht
Nachbedingung
Bei Erfolg wurde Nachricht gelesen

Anwendungsfall
Teammitglied auf Karte sehen
Akteure
Spieler
Vorbedingungen
Spieler nicht im Tarnmodus
Beschreibung
Auf der Übersichtskarte werden die Positionen aller Mitspieler, die nicht gerade getarnt sind, angezeigt. Neutralisierte Mitspieler sind dabei von aktiven Mitspielern etwa durch ihre Farbgebung unterscheidbar.
Parameter, Variablen
1. Koordinaten aller Spieler
2. Teamzugehörigkeit aller Spieler
3. Status aller Spieler (Tarnung)
Nachbedingung

Nicht getarnte Teammitglieder sind zu sehen

Anwendungsfall

Gegner auf Karte sehen

Akteure

Spieler

Vorbedingungen

Gegnerischer Spieler in Aufklärungsradius einer Aufklärungskamera oder im Sichtradius des Akteurs und nicht getarnt

Beschreibung

Gegnerische Spieler sind auf der Karte nur zu sehen, wenn sie sich im Sichtradius des Akteurs oder innerhalb des Aufklärungsradius einer selbst gesetzten Aufklärungskamera befinden. Der Sichtradius ist von den Eigenschaften des jeweiligen Spielers abhängig, ebenso wie der Aufklärungsradius einer Kamera von den Aufklärungsfähigkeiten des Spielers, der die Kamera aufgestellt hat, abhängt (siehe Aufklären).

Parameter, Variablen

1. Koordinaten aller Spieler
2. Teamzugehörigkeit aller Spieler
3. Status aller Spieler
4. Koordinaten und Radien eigener Aufklärungskameras

Nachbedingung

Gegner wird auf der Karte gesehen

Anwendungsfall

Tarnen

Akteure

Spieler

Vorbedingungen

Ausreichend Tarnpunkte verfügbar

Beschreibung

Durch die Funktion Tarnen wird der Akteur für alle anderen Spieler auf der Karte unsichtbar, auch wenn er den Sichtradius eines Gegners betritt (Ausnahme siehe Aufklären). Die Dauer der Tarnung hängt dabei von den verfügbaren Tarnpunkten ab, d.h. Tarnen verbraucht Tarnpunkte. Allerdings regenerieren sich diese wieder. Im Tarnmodus kann der Akteur selbst auf seiner Karte niemanden mehr sehen.

Parameter, Variablen

1. Eindeutiger Bezeichner des Tarnenden

2. Tarnfähigkeit

Nachbedingung

Spieler ist getarnt

Anwendungsfall

Aufklären

Akteure

Spieler

Vorbedingungen

Ausreichend Aufklärungspunkte vorhanden

Beschreibung

Die Funktion Aufklären erlaubt das Setzen von Aufklärungskameras innerhalb eines Aufklärungsradius des Akteurs. Dieser Aufklärungsradius hängt von den Aufklärungspunkten des Akteurs ab. Wurde eine Kamera gesetzt, verfügt diese über einen Sichtradius und eine Einsatzdauer, die ebenfalls beide von der Aufklärungsfähigkeit des Akteurs abhängen. Betritt ein gegnerischer Spieler den Sichtradius einer aktiven Aufklärungskamera, so wird dieser, auch wenn er getarnt ist, für den Spieler, der die Kamera gesetzt hat, auf dessen Karte sichtbar. Ist die Einsatzdauer einer Kamera abgelaufen, wird diese inaktiv. Die Aufklärungspunkte füllen sich automatisch wieder auf. Bei erfolgreicher Aufklärung wird der Aufklärer multimodal darüber informiert.

Parameter, Variablen

1. Koordinaten der Aufklärungskamera
2. Koordinaten der gegnerischen Spieler im Aufklärungsradius
3. Aufklärungspunkte des Aufklärers

Nachbedingung

Gegnerische Spieler im Aufklärungsradius gegebenenfalls enttarnt

---

Anwendungsfall

Neutralisieren

Akteure

Spieler

Vorbedingungen

Ausreichend Neutralisierungspunkte verfügbar

Beschreibung

Mit der Neutralisationsfunktion können gegnerische Spieler neutralisiert werden. Der Akteur setzt dazu innerhalb seines Wirkungsradius das Zentrum eines Neutralisationskreises, welcher dann solange anwächst, bis

entweder der Knopf zum Setzen losgelassen wird oder der maximale Neutralisationsradius erreicht ist. Im Anschluss wird die Neutralisationsfunktion sofort ausgelöst. Beide Radien ergeben sich aus den Neutralisationspunkten des Akteurs. Der Erfolg eines Neutralisationsversuchs hängt von der Größe des Neutralisationsradius ab. Je größer der Neutralisierungsradius, desto schwächer die Attacke und umgekehrt. Befinden sich mehrere Gegner innerhalb des Neutralisationsradius, wird gleichzeitig für jeden Gegner ein Neutralisationsversuch gestartet und dessen Erfolg separat ermittelt. Wurde ein Gegner erfolgreich neutralisiert, so verliert er all seine Fähigkeitspunkte und wird für seine Mitspieler als neutralisiert auf der Karte angezeigt. Alle Agenten werden gewarnt, wenn in ihrer Nähe ein Neutralisationskreis wächst.

Parameter, Variablen

1. Koordinaten aller Spieler
2. Koordinaten des Aktionsradius des Akteurs
3. Neutralisierungspunkte des Akteurs
4. Teamzugehörigkeit der Spieler im Aktionsradius

Nachbedingung

Alle gegnerischen Spieler im Aktionsradius neutralisiert
Neutralisierte Spieler werden für ihr Team als neutralisiert angezeigt

---

Anwendungsfall

Fähigkeiten aktivieren

Akteure

Spieler

Vorbedingungen

Fähigkeiten deaktiviert, Spieler an seinem Startpunkt

Beschreibung

Sind die Fähigkeiten eines Spielers entweder zu Beginn des Spiels oder durch eine Neutralisation deaktiviert, können diese durch Aufsuchen des Startpunkts aktiviert werden. Der Startpunkt wird für Spieler mit deaktivierten Fähigkeiten auf der Karte angezeigt. Nach erfolgreicher Aktivierung der Fähigkeiten wird der Akteur für seine Mitspieler auf der Karte als aktiv dargestellt.

Parameter, Variablen

1. Koordinaten des Spielers
2. Koordinaten des Startpunktes
3. Fähigkeitspunkte des Spielers

Nachbedingung

Der Spieler besitzt seine Fähigkeiten

Anwendungsfall
Alarmanlage deaktivieren
Akteure
Teammitglied von Mr. X
Vorbedingungen
Position der Alarmanlage erreicht
Beschreibung
Die Missionsziele für Mr. X sind alarmgesichert. Wird der Alarm ausgelöst, erfahren die Abwehragenten, wo Mr. X momentan zuschlägt. Um dies zu verhindern, kann die Alarmanlage deaktiviert werden. Dazu muss der Akteur eine Aufgabe z.B. in Form eines Rätsels lösen oder eine Frage nach Eigenschaftspunkten eines Mitspielers korrekt beantworten.
Parameter, Variablen
1. Koordinaten der Alarmanlage
2. Koordinaten der Abwehragenten
3. Das Rätsel
4. Die Lösung
5. Eingabe Abwehragent

Nachbedingung
Ist die Alarmanlage deaktiviert?

Anwendungsfall
Missionsziel erfüllen
Akteure
Mr. X
Vorbedingungen
Position des Missionsziels erreicht
Beschreibung
Zu Beginn stehen mehrere Missionsziele zur Auswahl. Die Reihenfolge, in der sie erfüllt werden müssen, ist frei wählbar. Befindet sich Mr. X an der Position eines Missionsziels, so wird ihm eine Aufgabe gestellt. Kann er diese lösen, so gilt das Missionsziel als erfüllt.
Parameter, Variablen
1. Koordinaten des Missionszieles
2. Koordinaten Mr. X
3. Zu lösende Aufgabe
4. Lösung der Aufgabe
5. Eingabe Mr. X

Nachbedingung
Hat Mr. X das Missionsziel erfüllt?

Anwendungsfall
Genaue Position der Missionsziele sehen
Akteure
Mr. X
Vorbedingungen
Keine
Beschreibung
Mr. X ist der einzige Spieler, dem die genaue Position der Missionsziele auf der Karte angezeigt werden. Diese sind permanent sichtbar und bedürfen keiner Aktion zu deren Anzeige.
Parameter, Variablen
1. Koordinaten der Missionsziele
Nachbedingung
Orte der Missionsziele für Mr. X sichtbar

Anwendungsfall
Genaue Position der Alarmanlage sehen
Akteure
Mr. X
Vorbedingungen
Keine
Beschreibung
Mr. X ist der einzige Spieler, dem die genaue Position der Alarmanlagen zu den Missionszielen auf der Karte angezeigt werden. Diese sind permanent sichtbar und bedürfen keiner Aktion zu deren Anzeige.
Parameter, Variablen
1. Koordinaten Alarmanlage
Nachbedingung
Ort der Alarmanlage ist für Mr. X sichtbar

Anwendungsfall
Ungenaue Position der Alarmanlage sehen

Akteure
Abwehragent
Vorbedingungen
Keine
Beschreibung
Die Abwehragenten wissen nicht genau wo Mr. X zuschlagen wird. Allerdings haben sie Informationen darüber, in welchen Gebieten Angriffe erwartet werden. Diese Regionen werden den Abwehragenten auf der Karte sichtbar gemacht. Die Anzeige der Zielregionen bedarf keiner Aktion. Jedoch können die Zielregionen durch Zusatzinformationen, die über Goodies erhalten werden können, weiter eingegrenzt werden.
Parameter, Variablen
1. Koordinaten der Missionsziele
2. Koordinaten der Gebiete der Missionsziele
Nachbedingung
Abwehragenten können die Gebiete auf der Karte sehen, wo Mr. X Missionen zu erfüllen hat

---

Anwendungsfall
Goodies sammeln
Akteure
Spieler
Vorbedingungen
Position des Goodies erreicht
Beschreibung
Ab und zu erscheinen auf der Karte Goodies. Begibt man sich an die Position eines solchen Goodies, können diese eingesammelt werden. Mögliche Inhalte der Goodies sind z.B. temporäre Zusatzpunkte für einzelne Eigenschaften oder für Agenten des Team X und Mr. X Hinweise zur Lösung von Aufgaben zur Erfüllung von Missionszielen oder speziell für Abwehragenten genauere Informationen zu den Zielregionen. Nach dem Einsammeln verschwinden die Goodies von der Karte.
Parameter, Variablen
1. Koordinaten der Goodies
2. Koordinaten der Spieler
3. Eigenschaft des Goodies
Nachbedingung
Die Eigenschaft des Goodies wirkt sich auf den Spieler aus

Anwendungsfall

Kontextsensitive Hilfe

Akteure

Server, Spieler

Vorbedingungen

Keine

Beschreibung

Wenn ein Spieler Hilfe benötigt, wird ihm zu der Aktion, die er gerade durchführen möchte, die entsprechende Hilfe bei Bedarf angezeigt.

Parameter, Variablen

1. Aktion, die Spieler durchführen möchte

Nachbedingung

Dem Spieler wird der Hilfe-Text angezeigt

Anwendungsfall

Serverauswahl

Akteure

Server, Spieler

Vorbedingungen

Spieler befindet sich in näherer Umgebung zum Spielfeld

Beschreibung

Möchte eine Person einem Spiel beitreten, so kann sich die Person die Server in der näheren Umgebung anzeigen lassen und sehen wann und wo das nächste Spiel beginnt. Anschliessend kann sich die Person bei einem anmelden.

Parameter, Variablen

1. Koordinaten des Spielers
2. Koordinaten des Spielfeldes

Nachbedingung

Spieler kann einem Spiel beitreten

Anwendungsfall

Karte zentrieren

Akteure

Spieler

Vorbedingungen

Keine

Beschreibung

Ist auf dem mobilen Endgerät des Spielers die Spielkarte zu sehen, so wird durch antippen einer Stelle auf der Karte diese zentriert und herangezoomt.

Parameter, Variablen

1. Koordinaten des Punktes auf der Karte

Nachbedingung

Der Spieler sieht eine vergrößerte Form der Karte an der Stelle wo er die Karte angetippt hat

---

Anwendungsfall

Karte zoomen

Akteure

Spieler

Vorbedingungen

Keine

Beschreibung

Ein Spieler kann die Ansicht der Karte an seiner Position vergrößern oder verkleinern.

Parameter, Variablen

1. Koordinaten des Spielers

Nachbedingung

Die Kartengröße hat sich verändert

Anwendungsfall

Server erstellen

Akteure

Spieler

Vorbedingungen

Der Spieler hat die notwendigen Rechte

Beschreibung

Eine Person kann einen Spielserver erstellen. Hierzu sucht sie sich eine vorhandene Karte aus, bzw. stellt eine neue Karte bereit. Weiterhin kann er die Anzahl der Spieler, die am Spiel teilnehmen können, die Dauer des Spiels und die Anzahl der Missionsziele festlegen.

Parameter, Variablen

1. Verfügbare Karten

Nachbedingung

Ein neuer Spielserver ist erstellt

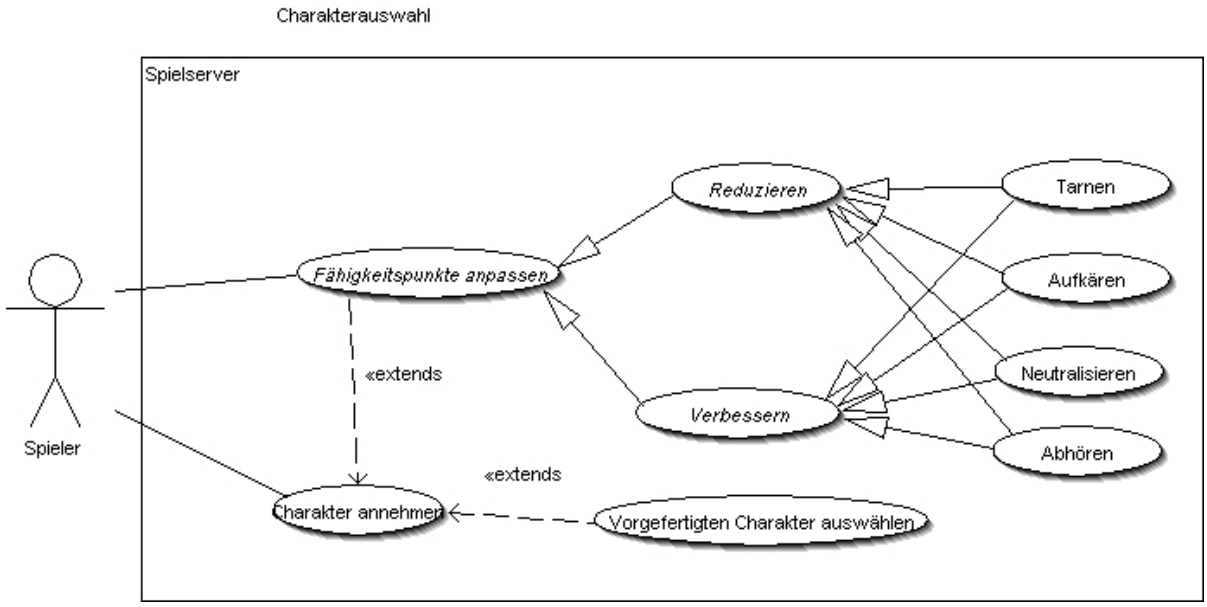


Abbildung 11.2: Anwendungsfälle zur Charakterauswahl

Anwendungsfall

Fähigkeitspunkte anpassen

Akteure

Spieler

Vorbedingungen

Spieler ist einem Spiel beigetreten

Beschreibung

Nachdem ein Spieler einem Spiel beigetreten ist, kann er seine Fähigkeiten ändern. Hierfür stehen eine bestimmte Anzahl von Punkten zur Verfügung. Diese Punkte kann der Spieler auf die vier Fähigkeiten Tarnen, Neutralisieren, Aufklären und Abhören verteilen. Jedoch muss jede Fähigkeit mindestens einen, höchstens aber vier Punkte haben.

Parameter, Variablen

1. Eindeutiger Bezeichner des Spielers
2. Fähigkeiten des Spielers

Nachbedingung

Der Spieler hat seine Fähigkeitspunkte verteilt

Anwendungsfall

Vorgefertigten Charakter auswählen

Akteure

Spieler

Vorbedingungen

Spieler ist einem Spiel beigetreten

Beschreibung

Wenn ein Spieler einem Spiel beigetreten ist, kann er einen vorgefertigten Charakter auswählen und braucht keine Fähigkeitspunkte zu verteilen.

Parameter, Variablen

1. Vorgefertigter Charakter
2. Eindeutiger Bezeichner des Spielers

Nachbedingung

Der Spieler hat einen vorgefertigten Charakter ausgewählt

Anwendungsfall

Charakter annehmen

Akteure

Spieler

Vorbedingungen

Der Spieler hat seine Fähigkeitspunkte verteilt

Beschreibung

Wenn der Spieler seinen Charakter erstellt hat, bzw. einen Charakter ausgewählt hat, muss er diese Aktion noch einmal bestätigen.

Parameter, Variablen

keine

Nachbedingung

Spieler hat seine Charaktererstellung abgeschlossen

Anwendungsfall

Verfügbare Spiele finden

Akteure

Server, Spieler

Vorbedingungen

keine

Beschreibung

Wenn ein Spieler einem Spiel beitreten möchte, kann er sich auf seinem mobilen Endgerät alle in der Nähe zur Verfügung stehenden Spiele anzeigen lassen.

Parameter, Variablen

1. Verfügbare Spiele

Nachbedingung

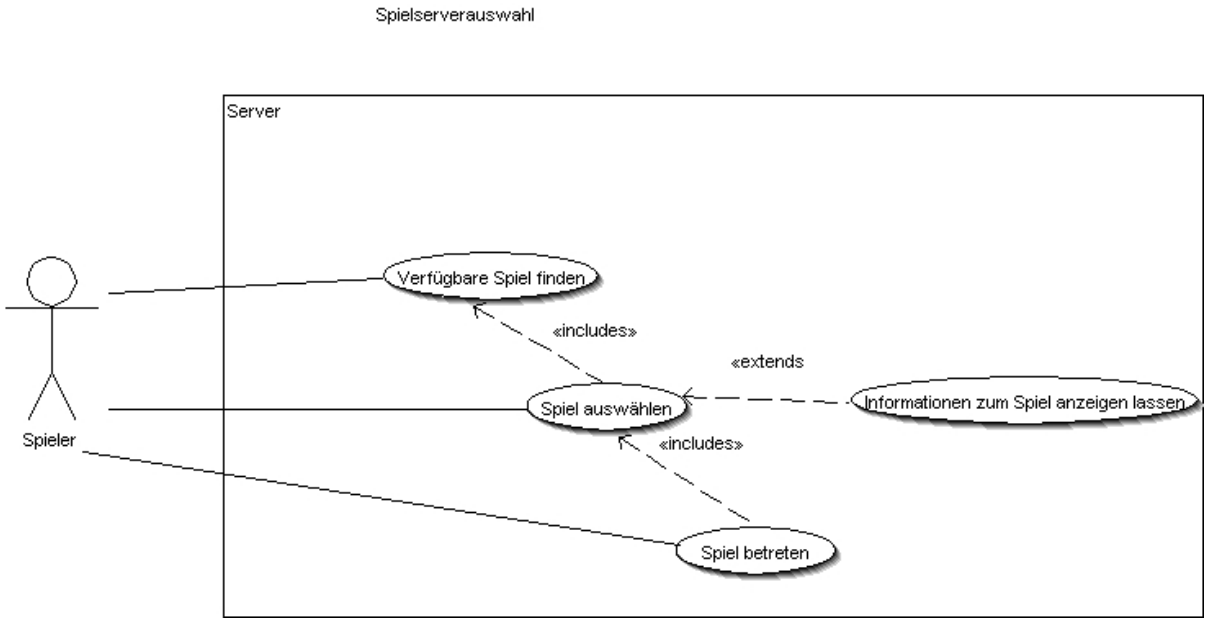


Abbildung 11.3: Anwendungsfälle zur Spielauswahl

Alle in der Nähe (räumlich) stattfindenden Spiele werden angezeigt

---

Anwendungsfall
Spiel beitreten
Akteure
Spieler
Vorbedingungen
Spieler hat ein Spiel ausgewählt
Beschreibung
Ein Spieler kann aus einer Liste ein Spiel auswählen, dem er beitreten möchte.
Parameter, Variablen
1. Ausgewähltes Spiel
Nachbedingung
Spieler ist einem Spiel beigetreten

Anwendungsfall
Spiel auswählen
Akteure

Spieler

Vorbedingungen

Verfügbare Spiele werden auf dem mobilen Endgerät angezeigt

Beschreibung

Ein Spieler kann aus einer Liste ein Spiel auswählen, über das er nähere Informationen haben möchte und kann sich diese anzeigen lassen.

Parameter, Variablen

1. Verfügbare Spiele
2. Informationen über verfügbare Spiele

Nachbedingung

Dem Spieler werden Informationen zu einem bestimmten Spiel angezeigt

---

11.4.2 Datenhaltung

11.4.2.1 Spielserver

Der Spielserver hält die Liste aller verfügbaren Spiele bereit. Außerdem werden vom Server die Szenarien verwaltet. Zu einem Szenario gehören folgende Daten:

- Karte
- Bild- und Klangdateien und weitere Ressourcen
- Eine Datei zur Beschreibung des kompletten Szenarios (umfasst das Auftauchen von Goodies, die Auswahl (Zusammenstellung) von Missionszielen, das Eintreten von Ereignissen, usw.)

11.4.2.2 Speicherung der Daten und Anforderung an die Datenübertragung

Um das Datenvolumen der Übertragung möglichst gering zu halten, wird die Karte bei Spielstart nur übertragen, wenn sie zu Beginn noch nicht auf den Clients vorhanden ist. Das gleiche gilt für in dem Szenario verwendete Klang und Bilddaten, bzw. größere Objekte. Ziel ist es, lediglich die minimal erforderlichen Daten zwischen den Geräten zu übertragen. Generell sollen größere Objekte durch Referenzen verwaltet werden.

Server	Client
Karte	Karte (bei Spielstart übertragen)
Liste der Spielobjekte	Liste der sichtbaren Spielobjekte
Liste der Profile	Profilidentifikation und Liste der Profile
Spielzeit/Spieldauer	Nachrichten (gesendet und empfangen)

- Karte: Bei der Datenstruktur der Karte handelt es sich um mehrere Bilddateien, die vom Client angezeigt werden.

- Spielobjekt: Oberklasse für alle anzuzeigenden Objekte auf der Karte. Enthält Koordinaten und eine Bildreferenz.
  - Spieler: Enthält eine Identifikationsnummer für sein Profil.
  - Goodie: Enthalten Informationen über den Effekt und die Position auf dem Spielfeld.
  - Sonstige Interaktionsobjekte (Missionsziele und Optionale Ziele): enthält
    - die zu erledigende Aufgabe
    - ggf. weitere Daten, wie z.B. Bilder
    - die Position auf dem Spielfeld
- Profil: enthält
  - eine Identifikationsnummer
  - die Merkmalsausprägungen der vier Eigenschaften
  - den Namen des Spielers
  - den aktuellen Status des Spielers
- Nachricht: enthält
  - einen Text
  - ggf. einen Anhang, der aus Bild- und/oder Klangdateien bestehen kann
- Status: enthält
  - die Teamzugehörigkeit (Mr. X, Abwehragent, Helfer von Mr. X)
  - Informationen über den Spielzustand (neutralisiert, ausgeschlossen, ausgetreten, usw.)

11.5 Nichtfunktionale Anforderungen

Dieser Abschnitt beschreibt die nichtfunktionalen Anforderungen an die Software. Dabei handelt es sich um Anforderungen an qualitative Merkmale der Software. Dies sind im folgenden:

- **Kontextsensitivität**: Die Besonderheit an dem Spiel ist die Möglichkeit, die reale Umgebung des Spielers in das Spiel mit einzubeziehen. In den Spielszenarien werden Ereignisse auf Orte in der Umwelt bezogen sein. Desweiteren ist die Entfernung zwischen Spielern bei vielen Aktionen entscheidend, die der Spieler taktisch berücksichtigen muss. So Entsteht ein Bezug zwischen Umwelt und einzelnen Spielsituationen, die den Spielverlauf nachhaltig prägen.
- **Multimodalität**: Da vom Spieler ein hoher Grad an Mobilität während des Spielablaufs erwartet wird, sollen möglichst viele Ein- bzw. Ausgabekanäle genutzt werden, um die Bedienung der mobilen Geräte zu erleichtern. Dies hängt stark von den technischen Möglichkeiten der Endgeräte ab. Ein akustischer Ausgabekanal zum Signalisieren von Spielereignissen sollte aber auf jeden Fall zur Verfügung stehen und genutzt werden.
- **Fehlertoleranz**: Da das Spiel von Techniken zur mobilen Kommunikation und Positionsbestimmung abhängt, soll Rücksicht auf deren Fehleranfälligkeit genommen werden. Ein kurzzeitiger Abbruch der Kommunikation zwischen Server und einem Client sollte zum Beispiel nicht zum Abbruch des Spiels führen oder automatisch den Ausschluss des Spielers zur Folge haben. Spieler erhalten bei regelwidrigem Verhalten Warnungen und bei Fehlern Fehlernachrichten vom Server, um einen möglichst reibungslosen Spielablauf zu ermöglichen.

11.6 Anforderungen an die Benutzungsschnittstelle

Der Interaktion zwischen Benutzer und mobiler Anwendung sind aufgrund des konstruktionsbedingten eingeschränkten Design mobiler Endgeräte enge Grenzen gesetzt. Hierzu zählen u.a. eine begrenzte Bildschirmgröße und fehlende Eingabemöglichkeiten. Um die Benutzbarkeit zu fördern, soll die Interaktion mit dem Spiel multimodal erfolgen. Folgende Ziele sollen dabei erreicht werden:

- Die Benutzungsschnittstelle soll den Benutzer durch ihr Design ansprechen.
- Dem Benutzer soll das Spielen mit der Anwendung erleichtert werden.
- Die Benutzungsschnittstelle soll während der Bewegung bedienbar sein.
- Die Benutzungsschnittstelle soll die Art der Ein- und Ausgabemittel der Umgebung anpassen, sich somit kontextsensitiv verhalten.

Zur Umsetzung dieser Ziele auf mobilen Endgeräten eignen sich folgende Mittel:

Optische Mittel

Zu den optischen Mitteln zählen blinkende Symbole, Bilder oder farbliche Abstimmung der grafischen Oberfläche. Die Bedienung soll hauptsächlich über eine grafische Oberfläche erfolgen, die in drei Komponenten gegliedert ist:

- Statusleite
- Inhaltsfenster
- Aktionsleiste

Die Statusleiste informiert über den Zustand des Spiels. Symbole in der Statusleiste informieren den Spieler zu jeder Zeit über die Eigenschaften der Netzwerkverbindungen, wie z.B. die Qualität das GPS oder WLAN Signals. Das Inhaltsfenster ist zur Darstellung des Spielgeschehens vorgesehen. Die Aktionsleiste wird für die Bedienung des Spiels benutzt.

Akustische Mittel

Zu den akustischen Mitteln zählen Warntöne oder eine Sprachausgabe. Dabei sollen die akustischen Mittel den Informationsgehalt der optischen Mittel verstärken und die Aufmerksamkeit des Benutzer auf diese lenken.

Sensorische Mittel

Wenn möglich, sollen zusätzlich zu den akustischen Mitteln, Vibrationen des Gerätes den Informationsgehalt optischer Mittel verstärken. Denkbar ist auch, dass in bestimmten Situationen die sensorischen Mittel, die akustischen Mittel ersetzen.

Die Interaktion des Benutzers mit der Anwendung über das mobile Endgerät, soll durch Knöpfe, den Stift, eine aufklappbare Falttastatur und andere Eingabegeräte erfolgen. Falls realisierbar, wird eine Spracheingabe und eine Eingabe über Gesten in Erwägung gezogen.

11.7 Dokumentationsanforderungen

11.7.1 Anwenderorientierte Dokumentation

Die Dokumentation für den Benutzer besteht aus einer Online-Hilfe in deutscher Sprache im HTML-Format. Eine auf Papier gedruckte Dokumentation (Handbuch) ist nicht vorgesehen, die Online-Hilfe ist jedoch ausdruckbar. Zahlreiche Illustrationen erleichtern dem Benutzer das Verstehen der Online-Hilfe. Über einen Hilfebutton kann der Benutzer in jeder Spielansicht die Online-Hilfe aufrufen. Die Hilfe ist kontextsensitiv, d.h. wird die Hilfe aus einer bestimmten Ansicht aufgerufen, startet sie automatisch mit der Beschreibung dieses Abschnitts und der dort verfügbaren Funktionen.

11.7.2 Entwicklerorientierte Dokumentation

Die Dokumentation der Entwicklung soll in elektronischer Form erfolgen. Alle im Projektplan vorgesehenen Ergebnisdokumente sollen mit LaTeXverfasst werden. Ein technisches Handbuch soll als HTML-Dokument erstellt werden, der Sourcecode wird entsprechend den Richtlinien und Konventionen der gewählten Programmiersprache dokumentiert. Die für das Projekt wichtigsten Dokumente werden zu einem Abschlussbericht zusammengefasst.

Kapitel 12

Technologiestudie

12.1 Einleitung

Das folgende Kapitel enthält einen Überblick über die von der Projektgruppe durchgeführten Technologiestudien. Aufgabe der Technologiestudien ist es, verfügbare Technologien zu identifizieren, diese zu sichten und auf ihre Eignung zur Realisierung des Agentenspiels zu testen. Hierfür wird ein allgemeingültiger Kriterienkatalog erstellt, anhand dessen die Eignung der Technologie im Hinblick auf die Entwicklung des Agentenspiels überprüft wird.
Das Kapitel gliedert sich in folgende Abschnitte:

Abschnitt 12.2: Kriterienkatalog, anhand dessen eine Bewertung der einzelen Technologien erfolgt.

Abschnitt 12.3: Technololgiestudie zum Thema „Zielplattform“.

Abschnitt 12.4: Technololgiestudie zum Thema „Mobile Ein- und Ausgabegeräte“.

Abschnitt 12.5: Technololgiestudie zum Thema „Datenübertragung“.

Abschnitt 12.6: Technololgiestudie zum Thema „Positionsbestimmung“.

Abschnitt 12.7: Technololgiestudie zum Thema „Kontext“.

Abschnitt 12.8: Fazit der Technololgiestudien.

12.2 Kriterienkatalog

12.2.1 K.O. Kriterien

Verfügbarkeit
Die Technologie muss für die Entwicklung der PG zur Verfügung stehen.

Zeitaufwand
Die Technologie muss in angemessenem Zeitrahmen verwendbar sein. Der Einarbeitungsaufwand darf die Möglichkeiten der PG nicht überschreiten.

Spielbarkeit
Spielbarkeit ist sowohl Soll als auch K.O. Kriterium. Es gelten die selben Unterkriterien. Schränkt eine Technologie in einem oder mehreren der Unterkriterien das Spiel soweit ein, dass die Durchführung praktisch nicht mehr möglich ist, scheidet diese aus.

12.2.2 Soll Kriterien

Offene Standards
Es sollen möglichst offene Standards eingesetzt werden.

Plattformunabhängigkeit
Bei der Auswahl von Technologien sollte die Anzahl möglicher Plattformen nicht unnötig eingeschränkt werden.

Niedrige Kosten im Einsatz
Die Durchführung des Spiels sollte möglichst wenig Kosten verursachen.

Niedriger Aufwand im Einsatz
Zur Durchführung sollte möglichst wenig Aufwand betrieben werden.

Kommerzielle Nutzung
Eine kommerzielle Nutzung sollte nicht eingeschränkt werden.

Spielbarkeit
Ein hoher Grad an Spielbarkeit sollte angestrebt werden. Dazu gehören folgende Unterprunkte:

- Niedrige Latenzzeit
- Genauigkeit der Positionsbestimmung
- Kontinuität der Datenübertragung
- mobiler Einsatz
- Multimodalität
- Wetterunabhängigkeit

12.3 Zielplattformen

12.3.1 Einleitung

Dieses Paper entstand im Rahmen der Projektgruppe „eXplorer“, die 2004/2005 an der Carl von Ossietzky Universität Oldenburg angeboten wurde. Die Aufgabenstellung des Projekts ist die Entwicklung von Software mit Augenmerk auf „kontext-sensitivier Umgebungserkundung mit mobiler, multimodaler Unterstützung“. Konkret entschlossen wir uns zur Entwicklung eines Spiels, bei dem zwei Teams, ausgerüstet mit mobilen Endgeräten gegeneinander antreten. Ein Team bekommt dabei eine Reihe von zu erfüllenden Missionszielen, während das andere Team deren Erfüllung zu verhindern versucht. Mit den mobilen Endgeräten können die Positionen der anderen Spieler festgestellt werden und Spielaktionen ausgeführt werden. Ein Server koordiniert das Spiel und dient als Schiedsrichter.

Wir untersuchen in diesem Paper das *.NET Compact Framework*, *SuperWaba* und *Java 2 Micro Edition* und arbeiten ihre Vor- und Nachteile heraus. Dabei orientieren wir uns an dem von der Gruppe aufgestellten Kriterienkatalog und dessen Gewichtungen. Unsere Ergebnisse dient der Projektgruppe als Grundlage, um sich für eine Zielplattform zu entscheiden.

Zunächst erläutern wir in Abschnitt 12.3.2 die notwendigen Fachbegriffe, sowie die Kriterien, nach der wir unsere Untersuchung gestaltet haben. Abschnitt 12.3.3 stellt die wichtigsten Hardwaretypen und Betriebssysteme. Die Abschnitte 12.3.4, 12.3.5 und 12.3.6 stellen die Zielplattformen *.NET Compact Framework*, *SuperWaba* und *Java 2 Micro Edition* vor, evaluieren sie anhand des Kriterienkatalogs und Beschreiben die Implementierung eines „Hello World“ Programms. In Abschnitt 12.3.7 haben wir die Ergebnisse der Evaluierung zur besseren Übersicht in einer Bewertungsmatrix zusammengefasst.

12.3.2 Grundlagen

Der Begriff *Zielplattform* steht in diesem Dokument für die Kombination aus Programmiersprache, Bibliotheken sowie Entwicklungs- und Testumgebung mit denen die von uns geplante Anwendung für mobile Endgeräte realisiert werden kann. Davon getrennt zu betrachten sind Hardware und Betriebssystem, die wir unter dem Begriff *System* zusammenfassen.

12.3.2.1 Rahmenbedingungen (Kriterienkatalog)

In diesem Abschnitt präsentieren wir die Kriterien, anhand derer wir die möglichen Zielplattformen untersucht haben. Als Ausgangspunkt haben wir die im gemeinsamen Kriterienkatalog aufgestellten Richtlinien genutzt und im Kontext der Zielplattformen neu gewichtet bzw. konkretisiert.

Damit die Zielplattform in die nährer Untersuchung einbezogen wird, muss sie folgende Richtlinien erfüllen:

- **Verfügbarkeit**: Die Zielplattform muss der Projektgruppe zur Verfügung stehen.
- **Zeitaufwand**: Der Einarbeitungsaufwand darf die zeitlichen Möglichkeiten der Projektgruppe nicht überschreiten.
- **Spielbarkeit**: Die Zielplattform muss Technologien unterstützen, die eine ausreichend genaue Positionsbestimmung und angemessen schnelle Datenübertragung ermöglicht. Außerdem muss die Plattform mehrere Ein- und Ausgabekanäle unterstützen, um multimodale Interaktion zu ermöglichen.

Eine weitere Bewertung findet anhand folgender Kriterien statt:

- **Offene Standards**: Es sollen möglichst offene Standards eingesetzt werden.
- **Geräteunabhängigkeit**: Die Zielplattform soll auf möglichst vielen Geräten existieren bzw. nachrüstbar sein.
- **Niedriger Aufwand im Einsatz**: Zur Installation und Ausführung des Spiels sollte möglichst wenig Aufwand betrieben werden.
- **Kommerzielle Nutzung**: Eine kommerzielle Nutzung sollte nicht eingeschränkt werden.
- **Spielbarkeit**: Hier ist zu klären, wie gut die Zielplattform die obigen unter „Spielbarkeit“ erwähnten Richtlinien erfüllt. Außerdem bewerten wir die zu erwartende Perfomanz im Vergleich zu den anderen Zielplattformen.

12.3.3 Technologielandschaft

Dieser Abschnitt bietet einen Überblick über die aktuelle Technologielandschaft im Bereich der frei programmierbaren mobilen Endgeräte. In 12.3.3.1 stellen wie die wichtigsten Hardwaretypen vor und in 12.3.3.2 präsentieren wir die offenen mobilen Betriebssysteme.

12.3.3.1 Mobile Hardware

Es gibt mobile Hardware in vielen verschiedenen und ständig neuen Formen. Die Grenzen zwischen den Geräteklassen verwischen immer mehr (z.B. PDAs mit Telefoniermöglichkeit, oder Smartphones mit Prozessorleistungen jenseits von 100 MHz) und es kommen immer neue Geräteklassen auf den Markt, so

dass eine Katalogisierung schwierig ist und ihre Gültigkeit schnell verlieren würde. Darum orientieren wir uns zunächst daran, welche Hardware uns vom OFFIS[1] aus überhaupt leihweise zur Verfügung stehen würde. Das OFFIS ist im Besitz mehrerer Personal Digital Assistants und einiger Smartphones, die der Projektgruppe leihweise zur Verfügung gestellt werden könnten.

Smartphones Bei dem Begriff „Smartphone“ handelt es sich um ein Schlagwort für vergleichsweise leitungsfähige Mobiltelefone. Es gibt keine eindeutige Definition des Begriffs Smartphone. Laut Nokia besitzt ein Smartphone die folgenden Merkmale:

- Smartphones basieren auf einem kommerziellen, offenen Betriebssystem, das Drittanbietern ermöglicht, innovative Anwendungen für das jeweilige Gerät zu entwickeln
- Es gibt eine große Auswahl an Anwendungen von Drittanbietern
- Smartphones haben ein hochwertiges Display und eine grafische Benutzeroberfläche
- Sie verfügen über einen großen internen und einen erweiterbaren externen Speicher für Nutzerdaten und Anwendungen
- Sie bieten umfassende Verbindungsmöglichkeiten - sowohl lokal (USB, Bluetooth, WLAN, IrDA) als auch über Weitverkehrsnetze (GSM, WCDMA, CDMA)
- Die Daten lassen sich mühelos zwischen Smartphone und PC übertragen oder synchronisieren

Im folgenden werden die von den Herstellern zur Verfügung gestellten Informationen zu den im OFFIS verfügbaren Smartphones präsentiert:

[1] www.offis.de

Nokia 9210 Communicator *The Nokia 9210 Communicator combines advanced messaging and PDA features in a GSM device. The foldout QWERTY keyboard, large screen, e-mail support, and PersonalJavattechnology make it well-suited to enterprise applications.*[2]

Abbildung 12.1: Nokia 9210

- **Betriebssystem**: Symbian OS 6.0
- **Display**: Auflösung 640x200 bei 12-Bit Farbtiefe (4096 Farben)
- **Multimedia Unterstützung**: WAV, Video/audio Streaming
- **Java Technologie**: Personal Java, JavaPhone API
- **Datenübertragung (WAN)**: Infrarot, Serielles Datenkabel
- **Datenübertragung (PAN)**: HSCSD

Nokia 6600 *The Nokia 6600 imaging phone features advanced messaging capabilities and the ability to capture and play back video. Security features, e-mail, and Bluetooth connectivity make the device well-suited to enterprise applications.*[3]

Abbildung 12.2: Nokia 6600

- **Betriebssystem**: Symbian OS 7.0s
- **Display**: Auflösung 176x208 bei 16-Bit Farbtiefe (65536 Farben)
- **Multimedia Unterstützung**: 3GPP formats (H.263), MPEG-4, RealVideo, MIDI tones (poly 24), True Tones (WB-AMR)
- **Java Technologie**: CLDC 1.0, MIDP 2.0, Nokia UI API, Wireless Messaging API (JSR-120), Mobile Media API (JSR-135), Bluetooth API (JSR-82 No OBEX)
- **Datenübertragung (WAN)**: Infrarot, Serielles Datenkabel
- **Datenübertragung (PAN)**: GSM, GPRS

[2] http://www.forum.nokia.com/main/0,,016-2088,00.html?model=9210
[3] http://www.forum.nokia.com/main/0,,016-2209,00.html?model=6600

Sony-Ericsson P800 *The Sony Ericsson P800/P802 is a multimedia smartphone for worldwide communications, and with inbuilt camera. It is a phone that mixes fun and entertaining features with timesaving and efficient applications giving the users of PDAs, Communicators and other handheld devices a new, exciting choice. It has a wide range of features and functionality supporting imaging, messaging, gaming, music and connectivity.*[4]

Abbildung 12.3: Sony-Ericsson P800

- **Betriebssystem**: Symbian OS v.7.0
- **Display**: Auflösung 208x320 bei einer Farbtiefe von 12-Bit
- **Multimedia Unterstützung**: MIDI tones (poly 24)
- **Java Technologie**: CLDC 1.0, MIDP 1.0
- **Datenübertragung (WAN)**: Bluetooth und evtl. weitere
- **Datenübertragung (PAN)**: DSM, GPRS

[4] http://developer.sonyericsson.com/site/global/products/phones/p800/p800.jsp

Sony-Ericsson P910 *The Sony Ericsson P910 is the next generation of high-end smartphones. The P910 is a highly sophisticated smartphone, combining PDA, phone, still and video camera, e-mail and web browser, all in one stylish, mobile package. The P910 has the convenience of a Qwerty keyboard inside the flip and a large, 262 k-color touch screen.*[5]

- **Betriebssystem**: Symbian OS 6.0
- **Display**: Touchscreen mit einer Auflösung von 208x320
- **Multimedia Unterstützung**: 3GPP (H.263), MP4, MIDI (40), AMR, MP3, AU, WAV
- **Java Technologie**: CLDC 1.0, MIDP 2.0, Bluetooth (JSR-82), WMA (JSR-120), MMAPI (JSR-135), JTWI R1 (JSR-185), Personal Java with JNI
- **Datenübertragung (WAN)**: Infrarot, Bluetooth, RS-232 + USB
- **Datenübertragung (PAN)**: HSCSD

Abbildung 12.4: Sony-Ericsson P910

[5] http://developer.sonyericsson.com/site/global/products/phones/p910/p_p910.jsp

12.3.3.2 Betriebsysteme für mobile Endgeräte

Jedes mobile Endgerät verfügt über ein Betriebssystem. Die ersten Betriebssysteme waren speziell für ein Gerät entwickelt und hatten meist keine spezielle Benennung. Die Hersteller erkannten jedoch den Vorteil von geräteübergreifenden Betriebssystemen: die Entwicklung von auf verschiedenen Geräten einsetzbarer Software. Es konkurieren maßgeblich vier offene, geräteübergreifende Betriebssysteme miteinander, nämlich SymbianOS, Windows CE, Embedded Linux und PalmOS

SymbianOS SymbianOS ist die Weiterentwicklung des 1993 vom Psion eingeführten Betriebssystems EPOC, das auf den von Psion hergestellten Handhelds eingesetzt wurde. Im Frühjahr 1998 schlossen sich PSION Ericsson, Nokia und Motorola für die Weiterentwicklung zusammen. Ihr Ziel war es, ein neues Betriebssystem für zukünftige Generationen von Mobiltelefonen (speziell Smartphones) zu entwickeln. Panasonic schloß sich 1999 und Siemens und Sony 2002 dem Konsortium an. Zur Zeit ist Symbian OS das führende Betriebssystem für Smartphones.

Windows CE .NET Windows CE (Compact Edition) .NET ist ein komponentbasiertes, modulares Betriebssystem für „embedded“ Systeme. Hardwarehersteller können spezielle Konfigurationen für ihre Zielplatform zusammenstellen und mit eigenen Modulen kombinieren. Über einen Plattform-Assistenten haben Entwickler die Wahl zwischen einer Vielzahl von definierten Geräteentwürfen (z.B. PDA, Handy, Smartphone, Settop-Box, Web Pad). Windows CE .NET wurde von Grund auf neu entwickelt, ist also keine Portierung oder Anpassung anderer Betriebssyteme der Microsoft Familie.

Windows Mobile 2003 Windows Mobile 2003 basiert auf dem Kernel von Windows CE.NET 4.2 und ist speziell für PDAs und Smartphones gedacht. Das Vorgänger-OS Pocket PC 2002 war noch auf Windows CE 3.0 abgestimmt. Die Marke *Windows Mobile* ist Microsofts Versuch, ein einheitliches Paket für das PDA und Smartphone Marktsegment zu schaffen. Windows Mobile enthält außer dem reinen Betriebssystem einige Anwendungen wie Pocket Word oder Pocket Excel.

Embedded Mobile Linux Es existieren diverse Portierungen des „original“ Linux x86 Kernels für embedded Umgebungen. Von besonderem Interesse sind dabei jene, die für einen ARM Prozessor und die entsprechende Architektur ausgelegt sind, da diese der Hardware der meisten PDAs, Handys, Smartphones etc entsprechen. Diese Portierungen werden von unterschiedlichen kommerziellen Organisationen oder Open Source Projekten entwickelt und gepflegt, es gibt also nicht *das* Mobile Linux Betriebssystem sondern viele unterschiedliche Varianten mit eigenen Schwerpunkten und Ausprägungen. Beispiele sind *uClinux*[6], *Real Time Linux Foundation, Inc.*[7], *lynuxworks*[8], *Wind River*[9] und *MonstaVista*[10].

Palm OS Seit 1996 existiert eine Serie mobiler Betriebssysteme der Firma *PalmSource*[11] für Handys und PDAs auf Basis der ARM Architektur unter dem Namen *Palm OS*. Die neuste Version heißt*Palm OS Garnet*. Diverse Hersteller statten Geräte mit Palm OS Betriebssystemen aus, insbesondere der Firma *Palm*. Für registrierte Entwickler bietet *PalmSource* Software Development Kits für die Entwicklung von mobilen Anwendungen. *Palm OS* unterstützt nur Anwendungen auf Basis der eigenen APIs in C++, nicht aber Java oder .NET.

[6] uClinux: *http://www.uclinux.org*
[7] Real Time Linux Foundation, Inc.: *http://www.realtimelinuxfoundation.org*
[8] lynuxworks: *http://www.lynuxworks.com*
[9] Wind River: *http://www.windriver.com*
[10] MonstaVista: *http://www.mvista.com*
[11] *http://www.palmsource.com*

12.3.4 .NET Compact Framework

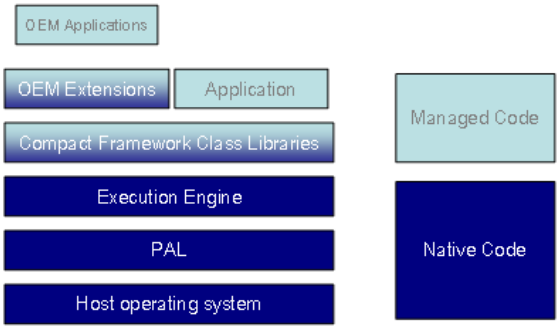


Abbildung 12.5: Namespaces supported in .NET Framework and .NET Compact Framework

Das .NET Compact Framework ist eine verschlankte Version des .NET Frameworks, deren Einsatzgebiete laut dem Entwickler Mircosoft im Bereich Personal Digital Assistants, Mobiltelefonen und Set-Top Boxen liegt. Wie bei Java erzeugt ein .NET Compiler Zwischencode, die so genannte Microsoft Intermediate Language (MSIL), die dann von einer Laufzeitumgebung, der Common Language Runtime (CLR) verarbeitet wird. Im Gegensatz zu JAVA, wo der Bytecode außschliesslich interpretiert wird, findet bei .NET auf manchen Plattformen eine Just-In-Time (JIT) Kompilierung der MSIL zu nativem Code statt. Da hierbei die Besonderheiten der aktuellen Architektur bekannt ist (z.B. Anzahl der Prozessorregister), kann der Kompiler deren Ressourcen effektiver ausnutzen als jeder andere Kompiler.

12.3.4.1 Bewertung

Das Microsoft .NET Compact Framework wird in die engere Auswahl einbezogen, da alle K.O.-Kriterien als erfüllt anzusehen sind.

- **Verfügbarkeit**: Mit der Studentenlizenz von Microsoft steht der Projektgruppe *Visual Studio .NET* kostenlos als IDE zur Entwicklung von nicht-kommerziellen Anwendungen mit .NET zur Verfügung. Das *.NET Compact Framework* und weitere Entwicklungstools (wie Emulatoren für unterschiedliche mobile Geräte) sind ebenfalls frei verfügbar. Desweiteren gibt es Open Source Projekte die genutzt werden können, wie NDoc[12], einem Dokumentationswerkzeug oder *OpenNETCF*[13], einer API mit Erweiterungen zm .NET Compact Framework.

- **Zeitaufwand**: Die Sprache der Wahl für .NET sollte C# sein. Sie wurde speziell für das .NET Framework entwickelt. Da die Struktur von C# der von Java sehr ähnelt, ist ein geringer Einarbeitungsaufwand in die Sprache für die Projektgruppe zu erwarten. Zu den Bibliotheken und Eigenschaften vom .NET Compact Framework finden sich im Internet viele Tutorials, Diskussionsforen, Artikel, etc., die eine Einarbeitung erleichtern sollten. Hier sei als Einstiegspunkt die Internetseite des Microsoft Developer Networks genannt.[14] Dazu steht der Projektgruppe mit Visual Studio .NET ein Leistungsstarkes IDE für die Entwicklung für .NET zur Verfügung.

- **Spielbarkeit**:
  - **Performanz**: Applikation werden als eine Art Metafile für .NET Compact geschrieben. Diese wird dann nach dem Herunterladen auf dem mobilen Endgerät kompiliert. Dabei fungiert die Runtime-Umgebung als Compiler und generiert dabei nicht nur schnellen Code, sondern bindet hardwarespezifische Features in den Code ein. Microsoft zufolge ist dieses Verfahren schneller als Java, auch wenn es nicht die Performance von nativem Code erreicht.

[12] *http://ndoc.sourceforge.net*
[13] *http://www.opennetcf.org*
[14] http://msdn.microsoft.com/smartclient/understanding/netcf/

  - **Positionsbestimmung**: Eine GPS Maus[15] kann via Bluetooth angesteuert werden. Die Verbindung übernimmt dabei das Betriebssystem. In .NET können dann die Daten der GPS Maus über einen Port abgefragt werden.
  - **Datenübertragung**: Datenübertragung per GSM/GRPS, UMTS oder WLAN muss von dem eingesetzten Betriebssystem zu Verfügung gestellt werden. Die Eigenschaften des Übertragungsmediums sind dabei für Anwendungen transparent, sie können diese als Internetverbindung nutzen. Die Datenübertragung ist also vom Betriebssystem abhängig.
  - **Multimodalität**: Mit dem .NET Compact Framework können Video und Audiodateien wiedergegeben und aufgenommen werden, die Ansteuerung des Vibrationsalarms ist ebenfalls möglich.

- **Offene Standards**: Laut Microsoft, ist es erklärtes Ziel mit dem .NET Frameworks und den Betriebssystemen für den mobilen Einsatz Marktführer im Bereich mobile Computing zu werden. Dabei soll das .NET Compact Framework selbst zum Standard werden.[16] Der Einsatz von offenen Standards ist kaum zu erwarten.

- **Geräteunabhängigkeit**: Ziel von Microsoft ist es, mit dem .NET Compact Framework eine hardwareunabhängige Plattform zu schaffen. So soll etwa bei neuer Hardware nicht die Anwendung an sich geändert werden müssen, sondern nur eine Anpassung der entsprechenden Library erfoderlich sein. Analog zum Windows-PC wird hierzu ein Satz von APIs (etwa Sound, Grafik etc.) als Subset von .NET definiert, auf den die Programmierer zugreifen. Microsoft stellt jedoch lediglich generische, gerätespezifische Bibliotheken zur Verfügung. In erster Linie werden die Hersteller in die Pflicht genommen ähnlich wie beim Treiber-Support für Windows.

- **Niedriger Aufwand im Einsatz**: Unter Windows Mobile 2003 sollten Anwendungen des .NET Compact Frameworks sofort einsetzbar sein. Ist der Einsatz von nativen Bibliotheken nötig, müssen je nach eingesetztem Gerät Vorkehrungen getroffen werden. Der Benutzer sollte aber keinen speziellen Aufwand für den Einsatz betreiben müssen.

- **Kommerzielle Nutzung**: Mit dem Erwerb der Eintwicklungsumgebung Visual Studio .NET, also nicht nur der Studentenversion, ist eine kommerzielle Nutzung ohne Probleme möglich.

12.3.4.2 Testbeschreibung

Da die verfügbare Hardware bereits mit Windows Mobile 2003 ausgestattet ist, empfiehlt es sich, das .NET Compact Framework als Entwicklungsplattform zu nutzen. Der Einstieg in die Entwicklung erweist sich als recht einfach, wie die kurze Beispielanwendung zeigt. In der Entwicklungsumgebung Visual Studio .NET wird ein neues Projekt für die entsprechende mobile Hardware angelegt. (Der Quellcode unserer HelloWorld Anwendung ist im Abschnitt 12.3.8 abgedruckt.) Mit dem Form-Desinger kann die Oberfläche gestaltet werden, in diesem Fall wurden ein Label mit der aufschlussreichen Botschaft „Hallo mobile Welt!“ und ein Button zum Beenden der Anwendung eingefügt. Der Form-Designer übernimmt das Anlegen von Initialisierungs-Code entsprechend der vom Entwickler angegebenen Eigenschaften. Es muss nur noch die Action-Methode für den Button per Hand mit Leben gefüllt werden. Danach lässt sich die Anwendung kompilieren und ein Emulator zum Debuggen auswählen. Ist ein mobiles Gerät angeschlossen, kann die Anwendung auch direkt dorthin übertragen und gestartet werden.

[15] Peripheriegerät, das GPS-Positionsdaten empfangen kann.
[16] ComputerWoche: http://www.computerwoche.de/index.cfm?pageid=255&artid=36656&type=detail&category=160

12.3.5 SuperWaba

SuperWaba[17] ist eine open-source Plattform zur Entwicklung von Anwendungen für PDAs auf der Basis von Palm OS, Windows CE/ Pocket PC / .NET und für Smartphones mit den Betriebssystemen Windows Mobile und Symbian 7. SuperWaba basiert auf Java, stellt aber eine eigene Virtual Machine zur Verfügung. Außerdem gehört zu dem SuperWaba SDK ein eigenes class-Dateiformat und eine Sammlung von Basis-Klassen (API). Wegen der Ähnlichkeit zu Suns Java-Standard können gängige Java Entwicklungswerkzeuge und Umgebungen (wie Eclipse oder J-Boss) für die Entwicklung mit SuperWaba genutzt werden.

Abbildung 12.6: SuperWaba Laufzeitumgebung

Die SuperWaba API unterstützt unter anderem:

- Exceptions und Threads
- Sammlung von UI-Elementen
- Datentypen double und long
- Endgeräte mit verschiedenen Auflösungen sowie Graustufen
- sockets, serial/IR und Bluetooth

12.3.5.1 Bewertung

Die SuperWaba Plattform erfüllt eigentlich alle K.O.-Kriterien, fällt aber (bis jetzt) beim Test durch. Trotzdem soll die Bewertung dokumentiert werden:

- **Verfügbarkeit**: SuperWaba ist in zwei Lizenzen verfügbar. Für Nichtabonnenten des professionellen SuperWaba-Pakets steht das SDK (Virtual Machine, APIs Dokumentation) unter der GNU General Public License[18] zur Verfügung. Dies bedeutet, dass die Quellen aller Software, die mit dem SDK entwickelt wird, der Öffentlichkeit verfügbar gemacht werden müssen.

  Mit dem professionellen Paket gilt die GNU Lesser General Public License[19], welche einen kommerziellen Vertreib der Software erlaubt. Das professionelle Paket kostet 495,00 $.

- **Zeitaufwand**: Da die zur SuperWaba Plattform gehörende Sprache sehr dicht an Java angelehnt ist, ergeben sich dieselben Vorteile für die Mitglieder der Projektgruppe wie bei der Java 2 Micro Edition. Jedoch ist laut Dokumentation ein höherer Aufwand beim Einrichten der Entwicklungsumgebung zu erwarten und der Entwicklungsprozess gestaltet sich komplizierter, da SuperWaba keine eigenen Entwicklungstools bietet. (Ein eigenes Eclipse-Plugin soll 2005 entwickelt werden.)

- **Spielbarkeit**:

  - **Performanz**: Laut Angaben des Herstellers[20], erreicht SuperWaba etwa ein Drittel der Geschwindigkeit in der Ausführung von nativ für die Hardware entwickelte und kompilierte C - Programme, soll aber schneller als andere Java Plattformen sein.

[17] http://www.superwaba.com/
[18] http://www.gnu.org/copyleft/gpl.html
[19] http://www.gnu.org/copyleft/lesser.html
[20] http://superwaba.sourceforge.net/cgi-bin/twiki/view/Main/SuperWaba

  - **Positionsbestimmung**: SupaWaba bietet einen eigenen Artikel, um den Einstieg mit GPS zu erleichtern. Der Gebrauch von GPS in Java (und SuperWaba) hängt stark von eingesetzter Hardware, Betriebssystem und Treibern ab, da Java mit diesen über eine native Schnittstelle kommunizieren muss. Die SuperWaba API bietet jedoch Klassen zur Aufbereitung von GPS Daten (z.B. ein Positions-Objekt), liegen die Daten erstmal vor.
  - **Datenübertragung**: Über sockets unterstützt die SuperWaba API TCP/IP, die Infrarot Schnittstelle, den seriellen Port und Bluetooth.
  - **Multimodalität**: SuperWaba ist spezialisiert auf die Eingeschaften von PDAs, also auf Oberflächen mit unterschiedlichen Größen und den Gebrauch von Graustufen statt Farbe. Andere Modalitäten werden nicht besonders beachtet.

- **Offene Standards**: Da SuperWaba von einer offenen Gemeinschaft unter der GNU General Public License entwickelt wird, kommen ausschließlich offene Standards zum Einsatz.
- **Geräteunabhängigkeit**: Nach der Installation der SuperWaba Virtual Machine sollten alle SuperWaba Anwendungen funktionieren, es sei denn sie greifen auf native oder gerätespezifische Funktionen zurück.
- **Niedriger Aufwand im Einsatz**: Vor einer Nutzung von Software für die SuperWaba Plattform muss die SuperWaba Virtual Machine auf dem mobilen Endgerät installiert werden. Dieser Vorgang ist nicht trivial (siehe Test) und erfordert genau Kenntnis der Hardware.
- **Kommerzielle Nutzung**: Eine kommerzielle Nutzung wäre nur mit einer professionellen Lizenz sinnvoll, welche 495,00 $ kostet.

12.3.5.2 Testergebnisse

Um Anwendungen auf einem PDA laufen lassen zu können, muss die SuperWaba Virtual Machine installiert werden. Dazu ist eine Auswahl und Kombination von unterschiedlichen Dateien nötig, abhängig von eingesetzter Hardware und Betriebssystem des PDAs. Trotz Test vieler Kombinationen, war es uns nicht möglich, die SuperWaba VM auf dem hp iPAQ zum laufen zu bekommen.

Testumgebungen für SuperWaba gibt es einige, zum Beispiel ein Applet auf dem die SuperWaba VM simuliert wird. Jedoch geht der Einsatz nicht so schnell wie bei den anderen getesteten Plattformen. Um Anwendungen zu starten müssen diese nicht nur kompiliert werden, sondern mit bestimmten Werkzeugen konvertiert und auf das Endgerät übertragen werden.

Bei der Evaluation fiel negativ auf, dass Teile der Dokumentation und bestimmte Funktionen von SuperWaba nur bei Erwerb der professionellen Lizenz zur Verfügung stehen.

12.3.6 Java 2 Micro Edition

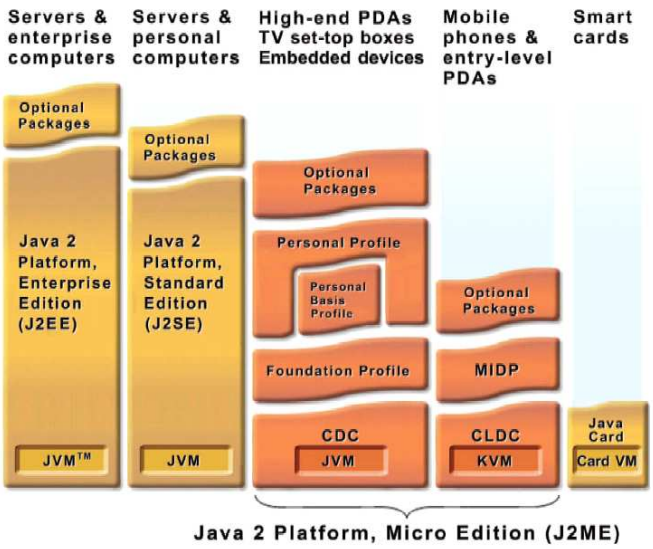


Abbildung 12.7: Einordnung der Micro Edition in die Java 2 Plattformen

Die Micro Edition[21] der Java 2 Platform bietet eine Anwendungsumgebung, die sich speziell an den Bedürfnissen ressourcenbeschränkter Hardware orientiert, wie z.B. Mobiltelefone, Pagers, Personal Digital Assistants, Set-Top Boxes, Fahrzeugtelemetrie, usw... . Die Architektur (siehe Abb.12.3.6) beinhaltet eine Vielzahl so genannter *Configurations*, *Profiles* und *Optional Packages*, aus denen die Entwickler die geeignet scheinenden Module zu einer vollständigen Laufzeitumgebung zusammensetzen können. Die Zusammensetzung entscheidet darüber, auf welchen Geräten die Software später laufen wird.

- **Configuration**: Eine Configuration besteht aus einer Virtual Machine und einem minimalen Satz von Klassenbibliotheken, welche die Basisfunktionalitäten für eine bestimmte Auswahl an Geräten mit gemeinsamen Charakteristiken bieten. Zur Zeit gibt es die *Connected Limited Device Configuration* (CLDC) und die *Connected Device Configuration* (CDC).

- **Profile**: Um eine vollständige Laufzeitumgebung bieten zu können, muss die Configuration mit einem Profile kombiniert werden. Dabei handelt es sich um ein Set von High-Level APIs, die z.B. den Life-Cycle der Anwendung, das User Interface und die Zugriffsmethoden auf gerätespezifische Eigenschaften definiert. Die meisten Smartphones und java-fähigen Handys kombinieren CLDC mit dem *Mobile Information Device Profile* (MIDP). Auf PDAs findet man eher die CDC und das *Personal Profile*.

- **Optional Package**: Die J2ME Plattform kann durch optionale Pakete erweitert werden. Zur Zeit gibt es viele Standard-APIs für existierende und aufkommende Technologien, wie z.B. Wireless Messaging, Multimedia, Bluetooth und Positionsbestimmung. Vorteil des modularen Aufbaus ist, dass auch mobile Endgeräte mit geringen Funktionalitäten die J2ME-Plattform anbieten können, und nicht zwingend bestimmte Hardware (wie z.B. Bluetooth) zu Verfügung stellen müssen.

SUN stellt eine Liste der wichtigsten Configurations, Profiles und Optional Packages unter http://java.sun.com/j2me/ bereit. Viele Hersteller bieten noch zusätzliche optionale Pakete an. Damit kann die Funktionalität ihrer Geräte besser ausgereizt werden, was aber zu Lasten der Portabilität geschieht.

[21] http://java.sun.com/j2me/index.jsp

12.3.6.1 Bewertung

Die Java 2 Micro Edition wird in die nähere Auswahl einbezogen, da sie alle K.O.-Kriterien erfüllt: Sie ist sowohl für uns als Entwickler als auch auf vielen mobilen Endgeräten verfügbar. Die Einarbeitungszeit ist überschaubar, da wir die Sprache JAVA bereits beherrschen. Es gibt bereits viele Spiele für J2ME, so dass die Plattform uns erlauben sollte, ein gut spielbares Produkt zu entwickeln. Die für unser Projekt besonderen Funktionalitäten werden allesamt in optionale Paketen bereitgestellt. Es folgt die Bewertung anhand der weiteren Richtlinien:

- **Verfügbarkeit**: Alle Entwicklungertools für die Java 2 Micro Edition sind frei verfügbar. Geräte auf denen die J2ME läuft, sind weit verbreitet. Die Verwertungsrechte der mit J2ME entwickelten Software werden nicht eingeschränkt. Weiterhin gibt es sehr Leistungsstarke und ebenfalls kostenlose IDEs und PlugIns zur Entwicklung von Software für J2ME.

- **Zeitaufwand**: Jeder Teilnehmer der Projektgruppe hat bereits den Programmierkurs Java und das Softwareprojekt hinter sich und ist deshalb zumindest mit den grundlegenen Prinzipien der Sprache vertraut. Im Internet finden sich sehr viele gute Tutorials, Diskussionsforen, Artikel, ..., die eine Einarbeitung in die Besonderheiten der Micro Edition erleichtern. Hier seien besonders die Internetseiten von SUN[22] und Nokia[23] empfohlen. Dazu kann mit Leistungsstarken IDEs wie Eclipse oder IDEA programmiert werden, die durch ihre vielen Automatisierungen die Implementierungsarbeit extrem beschleunigen.

- **Spielbarkeit**:

  - **Performanz**: die Java 2 Micro Edition ist speziell für mobile Geräte mit begrenzten Ressourcen entwickelt worden. Das heißt zwar einerseits, dass die Micro Edition nach Gesichtspunkten der Performanz optimiert wurde, bedeutet aber zugleich, dass das Gerät, auf dem eine für J2ME geschriebene Software läuft, mit großer Wahrscheinlichkeit eine eher geringe Rechenleistung besitzt. Entwickeln wir mit der J2ME müssen wir speziell darauf achten, damit das Spiel performant ist.
  - **Datenübertragung**: Datenübertragung per GSM/GRPS oder UMTS ist bereits in der Configuration einer J2ME Laufzeitumgebung integriert. Die Bluetoothschnittstelle mancher Smartphones kann komfortabel über die optionale Bluetooth-API (JSR-082) angesteuert werden. WLAN wird nur von den wenigsten Smartphones unterstützt, deshalb gibt es höchstens herstellereigene APIs dafür, die dann aber an bestimmte Geräte gebunden sind.
  - **Positionsbestimmung**: Eine GPS Maus[24] kann via Bluetooth angesteuert werden. Unter den offiziellen Optional Packages existiert auch die Location-API (JSR-179), die Ortsbestimmung mittels GPS oder E-OTD ermöglicht, aber die Details vor dem Entwickler verbirgt. Es wird zu diesem Zeitpunkt jedoch kaum mobile Endgeräte geben, die diese API unterstützen.
  - **Multimodalität**: Mit der auf Smartphones verbreiteten Mobile Media API (JSR-135) können Video und Audiodateien wiedergegeben und aufgenommen werden. Die Ansteuerung des Vibrationsalarms geschieht jedoch durch herstellerspezifische APIs, falls überhaupt möglich. Weiterhin stellen viele Smartphones neuester Generation integrierte Kameras zur Verfügung, die zur multimodalen Eingabe genutzt werden könnten, wenn der Hersteller APIs zu deren Ansteuerung bereitstellt.

[22] http://java.sun.com/j2me/index.jsp
[23] http://forum.nokia.com
[24] Peripheriegerät, das GPS-Positionsdaten empfangen kann.

- **Offene Standards**: Die Java 2 Micro Edition ist ein offener Standard. Ebenso sind die meisten Entwicklertools frei erhältlich. Nokia bietet z.B. ein kostenloses PlugIn für die Entwicklungsumgebung Eclipse an, mit dem Applikationen für J2ME und speziell auch die Nokia-eigenen Erweiterungen geschrieben werden können.

- **Geräteunabhängigkeit**: Die Abhängigkeiten von bestimmten Geräten sind bei der J2ME durch die Entwickler steuerbar. Je höher die Version der eingesetzten Configurations und Profiles ist, und je mehr optionale und herstellereigene Pakete verwendet werden, um so kleiner wird die Anzahl der Geräte, auf der die Software läuft. Hersteller wie Nokia sind jedoch bemüht, durch die Einführung so genannter *Series*, einheitliche Plattformen mit einem einheitlichen Satz an optionalen Packages zu bieten. Insgesamt handelt es sich bei J2ME jedoch um die Plattform, die auf den meisten mobilen Endgeräte im Massenmarkt zu finden ist.

- **Niedriger Aufwand im Einsatz**: Die meisten mobilen Endgeräte mit J2ME können Java-Applikationen über Datenkabel oder über GRPS installieren. Im Einsatz muss dann lediglich die GPS Maus aktiviert und erreichbar sein. Weitere spezielle Vorbereitungen zum Starten eines Programms sind nicht nötig.

- **Kommerzielle Nutzung**: Zur kommerziellen Nutzung scheint J2ME am attraktivsten, da nahezu alle Handys neuster Generation eine J2ME Laufzeitumgebung bieten. Bluetooth und GPRS bzw. UMTS werden in nicht allzuferner Zukunft Standard in den meisten Handys sein, wie es in jüngster Zeit die integrierten Kameras geworden sind. Längerfristig gesehen, könnte auch GPS in einen Teil der Geräte integiert werden. Weiterhin ist die gute Geräteunabhängigkeit von J2ME und abzusehende weite Verbreitung von Smartphones einer kommerziellen Verwertung der Software sehr förderlich.

12.3.6.2 Testbeschreibung

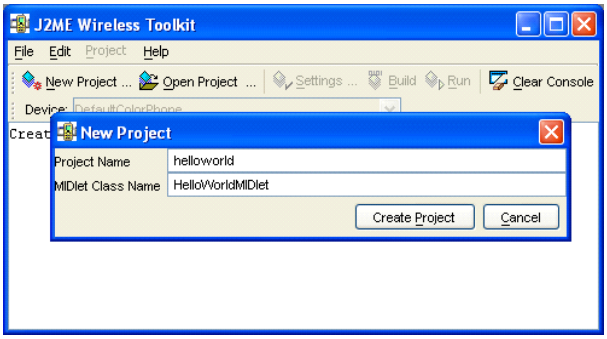


Abbildung 12.8: KToolbar des Wireless Toolkits 2.2

Um ein Gefühl für den Umgang Java 2 Micro Edition zu bekommen, wurde eine HelloWorld Anwendung geschrieben. Wir luden das von Sun angebotene J2ME Wireless Toolkit 2.2[25] als primäres SDK herunter. Wir starteten das zugehörige Programm KToolbar und legten ein neues Projekt an. Als J2ME Konfiguration wählten wir die Configuration CDLC 1.1 und das Profile MIDP 2.0 aus. Dazu integrierten wir die optionalen Pakete JSR-082 (Bluetooth) und JSR-135 (Mobile Media API). Das Paket JSR-179 (Location API) stand leider nicht zur Auswahl. Da JSR-179 von der Firma Nokia vorgeschlagen wurde, könnte jedoch es sein, dass die von Nokia angebotenen SDKs diese API unterstützen.

[25] http://java.sun.com/products/j2mewtoolkit/index.html

Abbildung 12.9: J2ME Emulator des Wireless Toolkits 2.2

Um komfortabel arbeiten zu können, wollten wir Eclipse als Quelltexteditor verwenden. Wir legten in Eclipse ein neues Projekt an. Als Verzeichnis für den Sourcecode des Projekts gaben wir den Pfad an, den die KToolbar als Source-Verzeichnis für das Projekt angelegt hatte. Um die J2ME spezifischen APIs verwenden zu können, importierten wir die Java-Archive `cldcapi11.jar`, `midpapi20.jar`, `mmapi.jar` und `jsr082.jar` aus dem lib-Verzeichnis des Wireless Toolkits in das Projekt. So erkannte Eclipse die J2ME spezifischen Klassen und Befehle und setzte automatisch die richtigen Imports. Auf den Webseiten von Nokia fanden wir Dokumente, die Tutorials zur Erstellung einer HelloWorld Applikation boten. J2ME-Anwendungen werden analog zu Applets und Servlets von einer Basisklasse abgeleitet, in der Methoden zur Steuerung des Lebenszyklus definiert sind. Diese Basisklasse heißt MIDlet, darum werden J2ME-Applikationen auch MIDlets genannt. Der Quellcode unseres HelloWorld MIDlets ist im Abschnitt 12.3.8 abgedruckt. Wir konnten den Quelltext jedoch nicht wie gewohnt mit Eclipse kompilieren, sondern musste dazu die KToolbar verwenden, um ein J2ME kompatibles Programm zu erzeugen. Aus der KToolbar heraus riefen wir den Emulator des Wireless Toolkits auf und starteten die Applikation. Abbildung 12.9 zeigt den Emulator und die Ausgabe der HelloWorld MIDlets.

12.3.7 Vergleichsmatrix

	.NET Compact Framework	SuperWaba	Java 2 Micro Edition
Verfügbarkeit			
Entwicklertools	++	+	++
Hardware	+	?	o
Verbreitung	o	—	++
Einarbeitungsaufwand			
Installation der Tools	++	o	+
Erlernen der Sprache	o	+	+
Dokumentation	+	+	++
Spielbarkeit			
Performanz	+	-	-
Positionsbestimmung	nur nativ	integriert via Bluetooth	integriert via Bluetooth
Datenübertragung	Bluetooth, IrDA, Sockets (WLAN, Internet)	Bluetooth-API, IrDA, Internet via Sockets	Bluetooth-API, Internet via Sockets
Multimodalität	+	o	+
Offene Standards	o	++	o
Geräteunabhängigkeit	+	+	o
Niedriger Einsatzaufwand	++	-	++
Kommerzielle Nutzung	—	-	++

12.3.8 „Hello World“ Listings

Listing 12.1: Hello World .NET Compact Framework

```
1  using System;
2  using System.Drawing;
3  using System.Collections;
4  using System.Windows.Forms;
5  using System.Data;
6
7  namespace MobileHalloWelt
8  {
9     /// <summary>
10    /// Zusammenfassung fuer Form1.
11    /// </summary>
12    public class Form1 : System.Windows.Forms.Form
13    {
14       private System.Windows.Forms.Label halloWeltL;
15       private System.Windows.Forms.Button endB;
16       private System.Windows.Forms.MainMenu mainMenu1;
17
18       public Form1()
19       {
20          // Erforderlich fuer die Windows Form Designerunterstuetzung
21          InitializeComponent();
22       }
23       /// <summary>
24       /// Verwendete Ressourcen bereinigen.
25       /// </summary>
26       protected override void Dispose( bool disposing )
27       {
28          base.Dispose( disposing );
29       }
30       #region Vom Windows Form-Designer generierter Code
31       /// <summary>
32       /// Erforderliche Methode fuer die Designerunterstuetzung.
33       /// Inhalt der Methode darf nicht mit dem Code Editor geaendert werden.
34       /// </summary>
35       private void InitializeComponent()
36       {
37          this.mainMenu1 = new System.Windows.Forms.MainMenu();
38          this.halloWeltL = new System.Windows.Forms.Label();
39          this.endB = new System.Windows.Forms.Button();
40          //
41          // halloWeltL
42          //
43          this.halloWeltL.Font = new System.Drawing.Font(
44                 "Palatino␣Linotype", 14.25F, System.Drawing.FontStyle.Italic);
45          this.halloWeltL.Location = new System.Drawing.Point(24, 40);
46          this.halloWeltL.Size = new System.Drawing.Size(152, 24);
47          this.halloWeltL.Text = "Hallo␣mobile␣Welt!";
48          //
49          // endB
50          //
51          this.endB.Location = new System.Drawing.Point(24, 192);
52          this.endB.Size = new System.Drawing.Size(96, 48);
53          this.endB.Text = "Ende␣(bitte)";
54          this.endB.Click += new System.EventHandler(this.endB_Click);
55          //
```

```
56          // Form1
57          //
58          this.Controls.Add(this.endB);
59          this.Controls.Add(this.halloWeltL);
60          this.Menu = this.mainMenu1;
61          this.Text = "HelloWorldForm";
62
63      }
64      #endregion
65
66      static void Main()
67      {
68          Application.Run(new Form1());
69      }
70
71      private void endB_Click(object sender, System.EventArgs e)
72      {
73          this.Close();
74      }
75  }
76 }
```

Listing 12.2: Hello World J2ME

```
77 import javax.microedition.lcdui.*;
78 import javax.microedition.midlet.MIDlet;
79 import javax.microedition.midlet.MIDletStateChangeException;
80
81 /**
82  * HelloWorld MIDlet, das anhand des Dokuments "Brief Introduction
83  * to MIDP Programming Version 1.0" der Firma Nokia.
84  */
85 public class HelloJ2ME extends MIDlet {
86
87     // Signalisiert dem MIDlet in den Zustand "Active" zu wechseln
88     protected void startApp() throws MIDletStateChangeException {
89         Displayable current = Display.getDisplay(this).getCurrent();
90         if (current == null) {
91             HelloScreen helloScreen = new HelloScreen(this, "Hello␣World.");
92             Display.getDisplay(this).setCurrent(helloScreen);
93         }
94     }
95
96     // Signalisiert dem MIDlet in den Zustand "Paused" zu wechseln
97     protected void pauseApp() {}
98
99     // Signalisiert dem MIDlet in den Zustand "Destroyed" zu wechseln
100    protected void destroyApp(boolean b)
101            throws MIDletStateChangeException {}
102
103    // Beendet die Anwendung
104    protected void exitRequested() throws MIDletStateChangeException {
105        destroyApp(false);
106        notifyDestroyed();
107    }
108
109    // Textbox, die den Text "Hello World" anzeigt
110    class HelloScreen extends TextBox implements CommandListener {
111        private final HelloJ2ME midlet;
```

```
112         private final Command exitCommand;
113 
114         // erzeugt eine neue Instanz der Textbox und registiert einen
115         // Commandlistener mit dem die Anwendung beendet werden kann
116         HelloScreen(HelloJ2ME midlet, String string) {
117             super("HelloWorldMIDlet", string, 256, 0);
118             this.midlet = midlet;
119             exitCommand = new Command("Exit", Command.EXIT, 1);
120             addCommand(exitCommand);
121             setCommandListener(this);
122         }
123 
124         // wird aufgerufen, wenn die mit dem Command assoziierte
125         // Taste gedrueckt wurde. Das MIDlet wird beendet.
126         public void commandAction(Command c, Displayable d) {
127             if (c == exitCommand) {
128                 try {
129                     midlet.exitRequested();
130                 } catch (MIDletStateChangeException e) {
131                 }
132             }
133         }
134     }
135 }
```

12.4 Mobile Ein- und Ausgabegeräte

Innerhalb dieses Kapitels werden verschiedene mobile Endgeräte, die für die Umsetzung des Agentenspiels in Frage kommen, vorgestellt. Hauptaugenmerk liegt hierbei in der Identifikation der verschiedenen Ein- und Ausgabemodalitäten. Zunächst werden die mobilen Endgeräte vorgestellt, die der Projektgruppe vom OFFIS zur Verfügung gestellt werden können und auf denen die spätere Anwendung installiert werden soll. Neben den bekannten PDAs (Personal Digital Assistents) kommen hierfür auch Smartphones in Frage. Im Anschluss daran wird kurz auf RFID (Radio Frequency Identification) eingegangen und verschiedene Möglichkeiten aufgezeigt, diese Technologie mittels eines mobilen Endgerätes zu nutzen. Den Abschluss dieses Kapitels bildet die Vorstellung weiterer mobiler Technologien, die zwar theoretisch für den Einsatz im Agentenspiel geeignet wären, jedoch nicht vom OFFIS für die Projektgruppe zur Verfügung gestellt werden können.

12.4.1 PDA

Ein PDA (Personal Digital Assistant) ist im Grunde ein kleiner tragbarer Computer mit einem schnell startendem Betriebssystem. Heutige PDAs bieten eine Vielzahl an Anwendungsmöglichkeiten. So können sie als Adressbuch, Terminplaner und zur Nutzung vieler Büroanwendungen, wie z.B. Text- oder Tabellenkalkulationsprogrammen genutzt werden. Über eine Internetverbindung, z.B. realisiert über ein WLAN-Modul, ist des weiteren der Zugang zum Internet möglich, um z.B. E-Mails zu versenden oder zu empfangen. Darüber hinaus sind heutige PDAs für den Einsatz multimedialer Anwendungen ausgelegt, wodurch sich das Spektrum möglicher Anwendungsbereiche dieser mobilen Endgeräte stark erweitert. In diesem Abschnitt werden die vom OFFIS für die Projektgruppe zur Verfügung gestellten PDAs un-

Abbildung 12.10: Hewlett-Packard iPAQ 5450

tersucht. Hierbei handelt es sich um Geräte vom Typ iPAQ 5450 der Firma Hewlett-Packard (siehe Abb.12.10 auf S.214). Hauptaugenmerk liegt hierbei in der Untersuchung der zur Verfügung stehenden visuellen, haptischen und akustischen Ein- und Ausgabemöglichkeiten.

visuelle Eingabemöglichkeiten Für visuelle Eingaben steht ein Lichtsensor zur Verfügung. Dieser dient dazu, die aktuellen Lichtverhältnisse zu messen, damit bei Bedarf die Hintergrundbeleuchtung des Displays zugeschaltet werden kann.

visuelle Ausgabemöglichkeiten Visuelle Ausgaben erfolgen über das Display des PDA. Dabei handelt es sich um ein transflektives 3,8 Zoll TFT-Display. Dieses erlaubt eine Auflösung von 320x240 Pixel

bei 65.536 Farben. Ausgaben auf dem Display können sowohl im Hoch- als auch im Querformat erfolgen. Wie bereits bei der Beschreibung des Lichtsensors erwähnt, verfügt das Display über eine Hintergrundbeleuchtung, die bei schlechten Lichtverhältnissen aktiviert wird.

auditive Eingabemöglichkeiten Für die auditive Eingabe steht ein Mikofon zur Verfügung. Dies ermöglicht u.a. den Einsatz des PDAs als Diktiergerät.

auditive Ausgabemöglichkeiten Die auditive Ausgabe kann auf zwei Arten erfolgen. Entweder bedient man sich des integrierten Mikrofons des iPAQ 5450 oder man schließt einen externen Lautsprecher, i.d.R. wird dies ein entsprechender Kopfhörer sein, über die 3,5 mm Stereokopfhörerbuchse an den PDA an.

haptische Eingabemöglichkeiten Eine haptische Eingabe kann auf mehrere Arten erfolgen. Der PDA verfügt über ein berührungsempfindliches Display, einen so genannten Touchscreen. Hier können per Berührung, z.B. durch Finger oder Zeigestift, Eingaben gemacht werden. Darüber hinaus können die vier Funktionstasten und eine 5-Wege Navigationstaste für die Ansteuerung verschiedener Funktionen des PDA genutzt werden. Diese Tasten befinden sich unterhalb des Displays. Eine Besonderheit des iPAQ 5450 besteht in dem biometrischen Fingerabdruckscanner. Dieser dient der Authentifikation des Benutzers und kann zusätzlich zu Benutzerkennung und Passwort genutzt werden. Eine weitere haptische Eingabemöglichkeit besteht in der Regulierung der Lautstärke durch einen entsprechenden Regler.

haptische Ausgabemöglichkeiten Für die haptische Ausgabe steht ein Vibrationsalarm zur Verfügung. Der iPAQ 5450 verfügt über eine Reihe von Schnittstellen, die im Folgenden kurz vorgestellt werden.

- Bluetooth Schnittstelle
- Infrarot (Serial Infrared (SIR): bis zu 115.200 bps)
- WLAN Schnittstelle (IEEE 802.11b)
- Steckplatz, kompatibel mit MMC- (Multi Media Card) und SD-Karten
- serielle Schnittstelle

Weitere Informationen zu diesem PDA können über die Internetseiten von Hewlett-Packard[26] eingesehen werden.
Zum Abschluss wird nun noch einiges Zubehör des iPAQ 5450 vorgestellt. Um GPS Signale empfangen und auswerten zu können, muss der iPAQ mit einem externen GPS-Empfänger ausgestattet werden. Am OFFIS stehen hierfür drei GPS-Mäuse der Marke Falcom NAVI-1 (siehe Abb.12.11 auf S.216) zur Verfügung. Die GPS Mäuse können über die Bluetooth Schnittstelle mit dem iPAQ verbunden werden. Darüber hinaus stehen für den iPAQ 5450 sogenannte Jackets zur Verfügung. Diese erweitern den PDA um PCMCIA (Personal Computer Memory Card International Association) Steckplätze, über die u.a. RFID-Reader angeschlossen werden können. Abb. 12.12 auf S.216 zeigt ein solches Jacket. Vom OFFIS können für die Projektgruppe zwei dieser Geräte zur Verfügung gestellt werden.

[26] http://www.hp.com/

Abbildung 12.11: Falcom NAVI-1

Abbildung 12.12: Hewlett-Packard iPAQ PC Card Expansion Pack Plus

12.4.2 Smartphones

Smartphones oder MDAs (Mobile Digital Assistants) vereinen den Leistungsumfang eines PDA mit dem eines Mobiltelefons. So können diese Geräte einerseits wie herkömmliche Mobiltelefone zum Telefonieren oder Versenden von Kurznachrichten verwendet werden und bieten andererseits die Möglichkeit, Anwendungen wie z.B. Texteditoren oder Tabellenkalkulationsprogramme zu nutzen. Vom OFFIS könner der Projektgruppe vier verscheidene Smartphones zur Verfügung gestellt werden:

- Nokia N-Gage
- Nokia N-Gage QD
- Sony Ericsson P910
- T-Mobil SDA

Diese Smartphones werden im folgenden Abschnitt vorgestellt. Wie bei der Vorstellung der PDAs liegt auch hier wieder das Hauptaugenmerk auf den visuellen, auditiven und haptischen Ein- und Ausgabemöglichkeiten.

12.4.2.1 Nokia N-Gage

Das Nokia N-Gage (siehe Abb.12.13 auf S.217) stellt eine Mischung aus Spielkonsole und Mobiltelefon dar. Es bietet die Möglichkeit über eine Internetverbindung mit mehreren Mitspielern vernetzt zu spielen, E-Mails zu versenden, Internetseiten über einen XHTML-Browser anzuzeigen oder Audio- und Videodateien abzuspielen. Da es sich um ein Smartphone handelt, kann dieses Gerät auch als Mobiltelefon genutzt werden. Zusätzlich ist im Nokia N-Gage ein UKW-Stereo-Radio integriert. Die visuellen, auditiven und haptischen Ein- und Ausgabemöglichkeiten dieses Smartphones werden in diesem Abschnitt nun kurz vorgestellt.

Abbildung 12.13: Nokia N-Gage

visuelle Eingabemöglichkeiten Das Nokia N-Gage verfügt über keinerlei visueller Eingabemöglichkeiten.

visuelle Ausgabemöglichkeiten Die visuelle Ausgabe erfolgt über ein TFT-Farbdisplay mit 4.096 Farben und einer Auflösung von 176x208 Pixel. Das Display verfügt über eine Hintergrundbeleuchtung.

auditive Eingabemöglichkeiten Das Nokia N-Gage verfügt über ein integriertes Mikrofon, über welches Tonaufzeichnungen vorgenommen werden können.

auditive Ausgabemöglichkeiten Die auditive Ausgabe kann entweder über den integrierten Lautsprecher oder über einen Kopfhörer erfolgen, der über den Stereokopfhöreranschluss an das Smartphone angeschlossen werden kann.

haptische Eingabemöglichkeiten Für die haptische Eingabe stehen beim Nokia N-Gage auf der Vorderseite neben dem normalen Wählfeld eines Mobiltelefons eine Reihe weiterer Tasten zur Verfügung. Zwei Tasten sind für die Aufnahme, bzw. zum Beenden eines Telefongesprächs vorgesehen. Für die Navigation steht eine 5-Wege Navigationstaste zur Verfügung. Drei weitere Tasten dienen der Ansteuerung von Menü, Radio und MP3-Player. Darüber hinaus existieren auf der Vorderseite des Gerätes noch vier weitere Tasten, die für die Spielsteuerung vorgesehen sind.

haptische Ausgabemöglichkeiten Die haptische Ausgabe kann über einen Vibrationsalarm erfolgen. Das Nokia N-Gage verfügt über eine Reihe von Schnittstellen, die im Folgenden kurz vorgestellt werden.

- Bluetooth Schnittstelle (erfüllt Bluetooth Spezifikation 1.1): Reichweite bis zu 10 Meter
- GPRS (Klasse 6)
- HSCSD (High Speed Circuit Switched Data)
- MMC (Multi Media Card) Steckplatz
- USB 1.1 Schnittstelle

Weitere Informationen zu diesem Smartphone können über die entsprechenden Internetseiten von Nokia[27] eingesehen werden.

[27] http://www.n-gage.com/de-DE/gamedeck/ngage/

12.4.2.2 Nokia N-Gage QD

Mit dem N-Gage QD (siehe Abb.12.14 auf S.218) stellt Nokia den Nachfolger des N-Gage vor. Da das Vorgängermodell nicht den erhofften Erfolg brachte, man jedoch weiterhin ein *Spielehandy* anbieten wollte, entschied sich Nokia dafür, das N-Gage zu überarbeiten. Das Ergebnis war das N-Gage QD. Bei diesem Smartphone wurde u.a. auf den Einbau eines Radios und eines MP3-Players verzichtet. Darüber hinaus erfolgt die Audioausgabe über die Kopfhörer nur noch in Mono. Über welche visuellen, auditiven und haptischen Ein- und Ausgabemöglichkeiten dieses Smartphone verfügt, wird nun kurz vorgestellt.

Abbildung 12.14: Nokia N-Gage QD

visuelle Eingabemöglichkeiten Das Nokia N-Gage QD verfügt über keinerlei visueller Eingabemöglichkeiten.

visuelle Ausgabemöglichkeiten Die visuelle Ausgabe erfolgt über ein TFT-Farbdisplay mit 4.096 Farben und einer Auflösung von 176x208 Pixel. Das Display verfügt über eine Hintergrundbeleuchtung.

auditive Eingabemöglichkeiten Das Nokia N-Gage QD verfügt über ein integriertes Mikrofon, über welches Tonaufzeichnungen vorgenommen werden können.

auditive Ausgabemöglichkeiten Die auditive Ausgabe kann entweder über den integrierten Lautsprecher oder über einen Kopfhörer erfolgen, der über den Monokopfhöreranschluss an des Smartphone angeschlossen werden kann.

haptische Eingabemöglichkeiten Für die haptische Eingabe stehen beim Nokia N-Gage QD auf der Vorderseite neben dem normalen Wählfeld eines Mobiltelefons eine Reihe weiterer Tasten zur Verfügung. Zwei Tasten sind für die Aufnahme, bzw. zum Beenden eines Telefongesprächs vorgesehen. Für die Navigation steht eine 4-Wege Naviagtionstaste zur Verfügung. Zwei weitere Tasten dienen der Ansteuerung des Menüs oder zur Spielsteuerung.

haptische Ausgabemöglichkeiten Die haptische Ausgabe kann über einen Vibrationsalarm erfolgen. Das Nokia N-Gage QD verfügt über eine Reihe von Schnittstellen, die im Folgenden kurz vorgestellt werden.

- Bluetooth Schnittstelle (erfüllt Bluetooth Spezifikation 1.1): Reichweite bis zu 10 Meter
- GPRS (Klasse 6)
- HSCSD (High Speed Circuit Switched Data)

- MMC (Multi Media Card) Steckplatz

Weitere Informationen zu diesem Smartphone können über die entsprechenden Internetseiten von Nokia[28] eingesehen werden.

12.4.2.3 Sony Ericsson P910

Sony Ericsson bietet mit dem P910 (siehe Abb.12.15 auf S.219) ein Smartphone der neuesten Generation an. Vom OFFIS kann derzeit ein solches Gerät zeitweise für die Projektgruppe zur Verfügung gestellt werden. Angaben zu verwendetem Betriebssystem und unterstützten JAVA-Technologien wurden bereits in Kapitel 12.3.3 auf Seite 196 gemacht. In diesem Abschnitt sollen nun die visuellen, auditiven und haptischen Ein- und Ausgabemöglichkeiten dieses Gerätes näher untersucht werden.

Abbildung 12.15: Sony Ericsson P910

visuelle Eingabemöglichkeiten Das Sony Ericsson P910 verfügt über eine integrierte Kamera zur Aufnahme von Bildern und Videos, jeweils mit einer Farbtiefe von 24-Bit (16 Mio. Farben). Bilder werden im JPEG/JFIF-Format abgespeichert, wobei der Benutzer zwischen drei Qualitätsstufen wählen kann:

- 640x480 Pixel (VGA)
- 320x240 Pixel (QVGA)
- 160x120 Pixel (QQVGA)

Videos werden im Format 3GPP/MPEG4 mit AMR Sound mit einer Auflösung von 176x144 Pixel aufgezeichnet.

visuelle Ausgabemöglichkeiten Die visuelle Ausgabe erfolgt über ein transflektives TFT-Farbdisplay mit 18-Bit (262.144 Farben), das sowohl im Hoch- als auch im Querformat genutzt werden kann. Bei geschlossener Tastaturklappe beträgt das Sichtfeld des Displays 40x28 mm, was einer Auflösung von 208x144 Pixel entspricht. Durch Öffnen der Tastaturklappe vergrößert sich das Sichtfeld des Displays auf 40x61 mm. Das entspricht einer Auflöung von 208x320 Pixel. Das Display verfügt zusätzlich über eine Hintergrundbeleuchtung deren Stärke variierbar ist.

auditive Eingabemöglichkeiten Das Sony Ericsson P910 verfügt über ein integriertes Mikrofon über welches Tonaufzeichnungen vorgenommen werden können.

[28] http://www.n-gage.com/de-DE/gamedeck/ngage_qd/

auditive Ausgabemöglichkeiten Die auditive Ausgabe kann entweder über den integrierten Lautsprecher oder über einen Kopfhörer erfolgen, der über den Stereokopfhöreranschluss an das Smartphone angeschlossen werden kann.

haptische Eingabemöglichkeiten Für die haptische Eingabe stehen beim Sony Ericsson P910 mehrere Möglichkeiten zur Verfügung. Das transflektive Display kann, genau wie bei einem PDA, über einen Zeigestift oder per Fingerdruck genutzt werden. Darüber hinaus steht auf der Vorderseite der Tastaturklappe eine normale Mobiltelefontastatur, sowie auf der Rückseite eine *QWERTZ*-Tastatur zur Verfügung.

haptische Ausgabemöglichkeiten Die haptische Ausgabe kann über einen Vibrationsalarm erfolgen. Das Sony Ericsson P910 verfügt über eine Reihe von Schnittstellen, die im Folgenden kurz vorgestellt werden.

- Bluetooth Schnittstelle (erfüllt Bluetooth Spezifikation 1.1): Reichweite bis zu 10 Meter
- Infrarot (Serial Infrared (SIR): bis zu 115.200 bps)
- GPRS
- HSCSD (High Speed Circuit Switched Data)
- Slot für Sony Memory Stick: Duo (bis zu 128 MB werden unterstützt), PRO Duo (bis zu 1 GB werden unterstützt)
- USB-Anschluss
- serielle Schnittstelle

Weitere Informationen zu diesem Smartphone können über die Internetseiten von Sony Ericsson[29] eingesehen werden.

12.4.2.4 T-Mobile SDA

T-Mobile bietet mit dem SDA (siehe Abb.12.16 auf S.221) ein leistungsstarkes Smartphone an. Das OFFIS kann der Projektgruppe zeitweise ein solches Gerät zur Verfügung stellen. In diesem Abschnitt werden nun die wichtigsten Leistungsmerkmale dieses Smartphones beschrieben, wobei das Hauptaugenmerk wieder auf den visuellen, auditiven und haptischen Ein- und Ausgabemöglichkeiten liegt.

visuelle Eingabemöglichkeiten Das T-Mobile SDA verfügt über eine integrierte Kamera zur Aufnahme von Bildern und Videos, jeweils mit einer Farbtiefe von 24-Bit (16 Mio. Farben). Bilder werden im JPEG/JFIF-Format abgespeichert, wobei der Benutzer zwischen drei Qualitätsstufen wählen kann:

- 640x480 Pixel (VGA)
- 320x240 Pixel (QVGA)
- 160x120 Pixel (QQVGA)

Videos werden im Format 3GPP/MPEG4 mit AMR Sound, H.263 oder Motion-JPEG AVI mit einer Auflösung von 176x144 Pixel oder 128x96 Pixel aufgezeichnet.

[29] http://www.sonyericcson.com/

Abbildung 12.16: T-Mobile SDA

visuelle Ausgabemöglichkeiten Die visuelle Ausgabe erfolgt über ein transflektives TFT-Farbdisplay mit 65.536 Farben. Das Display hat eine Größe von 2,2 Zoll und bietet eine Auflösung von 176x220 Pixel an.

auditive Eingabemöglichkeiten Das T-Mobile SDA verfügt über ein integriertes Mikrofon, über welches Tonaufzeichnungen vorgenommen werden können.

auditive Ausgabemöglichkeiten Die auditive Ausgabe kann entweder über den integrierten Lautsprecher oder über einen Kopfhörer erfolgen, der über den Stereokopfhöreranschluss an das Smartphone angeschlossen werden kann.

haptische Eingabemöglichkeiten Für die haptische Eingabe stehen beim T-Mobile SDK mehere Möglichkeiten zur Verfügung. Neben der normalen Mobiltelefontastatur stehen eine 5-Wege Navigationstaste und acht weitere Tasten zur Verfügung. Hierzu gehören zwei Tasten für die Aufnahme, resp. Beendigung eines Telefongesprächs, eine Taste für die direkte Verbindung zur T-Mobile Internetseite, eine Taste für den Schnellzugriff auf SMS-, MMS- und E-Mail-Dienste sowie vier Tasten für die Menüsteuerung.

haptische Ausgabemöglichkeiten Die haptische Ausgabe kann über einen Vibrationsalarm erfolgen. Das T-Mobile SDA verfügt über eine Reihe von Schnittstellen, die im Folgenden kurz vorgestellt werden.

- Bluetooth Schnittstelle (Klasse 2): Reichweite bis zu 20 Meter
- Infrarot (Serial Infrared (SIR): bis zu 115.200 bps)
- GPRS
- SD/MMC Steckplatz
- 5-Pol-Mini-USB-Schnittstelle

Weitere Informationen zu diesem Smartphone können über die Internetseiten von T-Mobile[30] eingesehen werden.

[30] http://www.t-mobile.de/

12.4.3 RFID

Die RFID (Radio Frequency Identification) Technik bietet die Möglichkeit, Daten per Radiowellen zwischen einem Datenträger und einem mobilen Endgerät auszutauschen. RFID wird als Oberbegriff für die komplette technische Infrastruktur verwendet. Sie umfasst:

- den Transponder (auch RFID-Etikett, -Chip, -Tag, -Label, Funketikett oder-chip genannt)
- die Sende-Empfangs-Einheit (auch Reader genannt)
- die Integration mit Servern, Diensten und sonstigen Systemen wie z.B. Kassensystemen oder Warenwirtschaftssystemen

Die Daten werden auf den RFID-Transpondern gespeichert. Das Auslesen bzw. Schreiben der Informationen wird, wie bereits erwähnt wurde, per Radiowellen vorgenommen. Bei niedrigen Frequenzen geschieht dies induktiv über ein Nahfeld, bei höheren Frequenzen über ein elektromagnetisches Fernfeld. Die Entfernung, über die ein RFID-Transponder ausgelesen werden kann, schwankt aufgrund der Ausführung (aktiv/passiv), benutztem Frequenzband, Sendestärke und Umwelteinflüssen zwischen wenigen Zentimetern bis derzeit max. 1.000 Metern (per Abhöreinrichtungen laut der Studie „Risiken und Chancen des Einsatzes von RFID-Systemen“ vom Bundesamt für Sicherheit in der Informationstechnik 2004).
Um die RFID Technologie mit einem mobilen Endgerät, wie beispielsweise einem PDA nutzen zu können, muss dieses zunächst mit einem RFID-Reader ausgestattet werden. Hierzu gibt es RFID-Reader als SD- oder CF-Karten. Der Vorteil der SD-Karten ist, dass diese direkt in den entsprechenden Slot des am OFFIS vorhandenen iPAQ 5450 gesteckt werden können. Für die Verwendung einer CF-Karte muss der iPAQ zunächst mit einem Jacket ausgestattet werden, wie es weiter oben bereits beschrieben wurde. RFID Transponder sind in unterschiedlichsten Ausführungen erhältlich. Bei den am OFFIS zur Verfügung

Abbildung 12.17: RFID Transponder

stehenden RFID Transpondern (siehe Abb.12.17 auf S.222) unterscheidet man zwischen den vier Serien:

- Unique
- Titan
- Hitag1
- Hitag2

Alle Transponder besitzen eine einmalige Laser-programmierte Seriennummer. Bei der Titan, Hitag1- und Hitag2 Serie steht zusätzlich ein Lese-/Schreibspeicher zur Verfügung. Die wichtigsten Eigenschaften der einzelnen Serien werden hier kurz aufgelistet.

Serie Unique

- kontaktloser, nur lesbarer, Identifikationsbaustein
- 64 Bit Laser-programmierter Baustein-Fixcode, davon 40 Bit Seriennummer
- Spannungsversorgung des Transponders über das elektromagnetische Feld der Lesestation
- Datenübertragung durch Amplitudenmodulation
- 125 kHz Trägerfrequenz

Lese/Schreib Transponder Serie Titan

- 1 kBit EEPROM mit einer Speicherorganisation von 32 Worten mit je 32 Bits
- 928 Bits frei verfügbar
- 32 Bit Fixcode Seriennummer
- 32 Bit Bausteinidentifikationsnummer
- 32 Bit Passwort
- definierbarer lesegeschützter Bereich
- definierbarer Bereich, welcher bei vorhandener Spannung automatisch gesendet wird
- 125 kHz Trägerfrequenz
- Datenübertragung durch Amplitudenmodulation

Lese/Schreib Transponder Serie Hitag1

- 2 kBit EEPROM mit einer Speicherorganisation von 16 Blöcken je 4 Seiten zu 4 Bytes
- 1792 Bits frei verfügbar
- 32 Bit Fixcode Seriennummer
- 2 fixe Crypto-Sicherheitsblöcke und weitere 4 Speicherblöcke die entweder frei zugänglich oder als Sicherheitsbereich einstellbar sind. Hier können Daten verschlüsselt abgelegt werden und der Zugriff erfordert eine gegenseitige Authentifizierung.
- 6 Blöcke selektierbar als R/W oder OTP
- 125 kHz Trägerfrequenz
- Datenübertragung durch Amplitudenmodulation

Lese/Schreib Transponder Serie Hitag2

- 256 Bit EEPROM mit einer Speicherorganisation von 8 seiten mit je 4 Bytes
- 128 Bits frei verfügbar
- 32 Bit Fixcode Seriennummer
- Crypto oder Passwort Modus
- 48 Bit Crypto-Key für Verschlüsselung
- 32 Bit Passwort
- definierbarer lesegeschützter Bereich
- definierbarer Speicherbereich, welcher bei vorhandener Spannung automatisch gesendet wird
- 125 kHz Trägerfrequenz
- Datenübertragung durch Amplitudenmodulation

Wie dieser Auflistung zu entnehmen ist, arbeiten alle verfügbaren RFID-Transponder auf einer Frequenz von 125 kHz, die verfügbaren RFID-Reader jedoch auf einer Frequenz von 13,56 MHz. Aufgrund dieser Inkompatibilität kann die RFID-Technologie derzeit nicht eingesetzt werden.

12.4.4 weitere mobile Technologien

Zum Abschluss dieses Kapitels werden nun einige mobile Technologien vorgestellt, deren Einsatz innerhalb der Projektgruppe zwar denkbar wäre, aber aus verschiedenen Gründen derzeit nicht realisierbar ist. Zu den hier vorgestellten Technologien zählen neben dem *Push-To-Talk* von T-Mobile auch das von Siemens entwickelte *Digital Graffiti*.

PTT - Push-To-Talk Mit seinem Push-to-Talk-Dienst bietet T-Mobile eine Walkie-Talkie Funktion für das Mobiltelefon an. Der Unterschied zu der Nutzung eines normalen Walkie-Talkie besteht dabei in der unbegrenzten Reichweite. Voraussetzung für die Nutzung dieses Dienstes ist eine T-Mobile Karte, sowie ein PTT fähiges Handy. Eine Liste kompatibler Mobiltelefone ist auf den entsprechenden Seiten von T-Mobile[31] zu finden.
Eine PTT-Nachricht darf eine maximale Länge von 30 Sekunden haben und kann sowohl an Einzelpersonen, als auch an Personengruppen gesendet werden. Voraussetzung für den Empfang ist wiederum ein PTT-fähiges Mobiltelefon und eine T-Mobile Karte. Zu den weiteren Funktionen von PTT zählen u.a. die Statusabfrage, um zu sehen, welche Mitglieder einer bestimmten Gruppe derzeit empfangsbereit sind. Nähere Informationen zu PTT sind über die entsprechenden Internetseiten von T-Mobile[32] zu erhalten.

Digital Graffiti In einer Pressemitteilung vom 02. Februar 2005 machte Siemens darauf aufmerksam, in näherer Zukunft den so genannten Digital-Grafitti-Dienst auf den Markt bringen zu wollen. Dieser Dienst bietet Mobiltelefonbenutzern die Möglichkeit, Nachrichten in Form von virtuellen Haftnotizen an einem beliebigen Ort für bestimmte Personen zu hinterlassen.
Laut Siemens soll es mittels des Digital-Graffiti-Dienstes möglich sein, eine Nachricht nicht nur einem

[31] http://www.t-mobile.de/business/handys/pushtotalk/
[32] http://www.t-mobile.de/push-to-talk/

Empfänger, sondern auch einem bestimmten Ort zuzuordnen. Hierzu wird die vom Absender verfasste Nachricht, wobei es sich sowohl um eine Text- als auch um eine Bildnachricht handeln kann, zusammen mit den Koordinaten, von denen aus die Nachricht versendet wurde, beim Mobilfunkanbieter gespeichert. Dieser leitet die Nachricht erst dann zum Empfänger weiter, wenn dieser die angegebenen Koordinaten der Nachricht erreicht. Erst dann wird die Nachricht, also das digitale Graffiti, für ihn sichtbar.
Der Einsatz von Digital Graffiti innerhalb der Projektgruppe ist derzeit nicht möglich. Allerdings könnte man einen ähnlichen Dienst selbst implementieren. Der Text der hier erwähnten Pressemitteilung kann auf den Internetseiten von Siemens[33] nachgelesen werden.

[33] http://w4.siemens.de/ct/de/news/2004_2005/2005_02_02_digital_graffiti.pdf

12.5 Datenübertragung

12.5.1 Einleitung

In den folgenden Abschnitten werden verschiedene Technologien zur Übertragung von Daten vorgestellt. Dabei werden die Übertragungstechnologien Infrarot, Bluetooth, WLAN, GPRS und UMTS genauer betrachtet. Einleitend wird in jedem Abschnitt jeweils eine kurze Übersicht über wichtige Charakteristika jeder Technologie gegeben. Daraufhin wird jeder Übertragungsstandard nach festgelegten Kriterien untersucht, um die Technologien vergleichen zu können. Unter diese Kriterien fallen „Übertragungsrate“, „Reichweite“, „Verbreitung“, „Spielbarkeit“, „Zusätzliche Hardware / Infrastruktur“, „Kosten im Einsatz“, „Latenzzeit“, „Kontinuität der Datenübertragung“, „Mobiler Einsatz“ und „Wetterabhängigkeit“. Das abschließende Fazit enthält eine Einschätzung, welche dieser Technologien den Anforderungen zur Entwicklung des Spiels genügen.

12.5.2 Infrarot

Der Infrarot-Standard wurde 1994 von der IrDA (Infrared Data Association) als Alternative zum seriellen Port entwickelt. Dies zeigt sich vor allem in der begrenzten Reichweite einer Infrarot-Übertragung von nur einem Meter. In dem Standard ist eine Übertragungsgeschwindigkeit von bis zu 4 MBit/s vorgesehen, die reale Übertragungsgeschwindigkeit liegt jedoch eher bei 155 KBit/s. Als Medium zur Datenübertragung dienen Lichtwellen in dem Spektrum zwischen 850 und 900 nm. Für die Kommunikation zwischen Infrarot-Geräten kann sowohl ein verbindungsloses als auch eine verbindungsorientiertes Verfahren eingesetzt werden.

Für einen Entwickler erhöht sich der Implementierungsaufwand bei einem Einsatz dieser Übertragungstechnik aufgrund einer Vielzahl an verwendeten Protokollen.

12.5.2.1 Bewertung

Übertragungsrate: 155 KBit/s

Reichweite: 1 Meter (Sendekegel: 30°)

Verbreitung: In den meisten mobilen Geräten sind Infrarot-Module bereits integriert

Spielbarkeit: Aufgrund der geringen Reichweite nicht für den Spieleinsatz geeignet

Zusätzliche Hardware / Infrastruktur Keine zusätzliche Harware (bei integriertem Infrarot-Chip) erforderlich / Keine zusätzliche Infrastruktur erforderlich

Kosten im Einsatz: Es entstehen keine laufenden Kosten

Latenzzeit: (Keine Angabe möglich)

Kontinuität der Datenübertragung Kann abreißen

Mobiler Einsatz: Aufgrund der geringen Reichweite nicht für den mobilen Einsatz geeignet

Wetterabhängigkeit: Infrarot ist störungsanfällig gegenüber Sonnenlichteinstrahlung

12.5.3 Bluetooth

Bluetooth ist ein Funkübertragungsstandard, der für die Vernetzung von persönlichen Geräten konzipiert wurde. Für die Übertragung von Daten benötigen die Geräte keinen direkten Sichtkontakt. Bluetooth-Module lassen sich in drei Klassen einteilen, deren Sendeleistung maßgeblich für die Übertragungsreichweite ist. Geräte der Klasse 3 senden mit einer Leistung von 1 mW und erreichen dadurch eine Weite von 10 Metern, Geräte der Klasse 2 senden mit einer Leistung von 2 mW und können Daten über eine Distanz von 20 Metern übertragen und Geräte der Klasse 1 senden mit einer Leistung von 100 mW und übertragen Daten an Geräte, die sich in bis zu 100 Metern Entfernung befinden. Aufgrund der Leistungsaufnahme der Module werden in mobilen Geräten in der Regel Module der Klasse 3 eingesetzt.

Bis zu acht bluetoothfähige Geräte sind in der Lage kleine ad-hoc Netzwerke zu bilden (Piconet), in denen ein Gerät als Master die Kommunikation zwischen den Geräten leitet. Überlagern sich die Wirkungsbereiche mindestens zweier Piconets bildet sich ein Scatternet, wenn mindestens ein Gerät Anschluss an beide Netze hat.

Bluetooth-Geräte senden Daten auf dem 2,4835 GHz ISM-Band, das weltweit für jedermann frei nutzbar ist, auf dem unter anderem WLAN Daten überträgt. Um Kollisionen zu vermeiden, nutzt Bluetooth das Frequenzsprungverfahren, wobei bis zu 1600 mal pro Sekunde die Frequenz gewechselt wird. Dadurch ist dieser Übertragungsstandard weniger anfällig gegenüber Störungen als beispielsweise WLAN.

Die Übertragungsgeschwindigkeit von Bluetooth beträgt bis zu 721 KBit/s, durch einen hohen Overhead der eingesetzten Protokolle geht aber ein Großteil der theoretisch verfügbaren Bandbreite verloren.

12.5.3.1 Bewertung

Übertragungsrate: Bis zu 721 KBit/s.

Reichweite: Die Reichweite ist abhängig von der Klasse des Bluetooth-Moduls und kann zwischen 10 (Klasse 3) und 100 Metern (Klasse 1) betragen

Verbreitung: In den meisten mobilen Geräten sind Bluetooth-Module bereits integriert

Spielbarkeit: Da in mobilen Geräten aufgrund der geringeren Leistungsaufnahme in der Regel Bluetooth-Module der Klasse 3 eingesetzt werden, reicht die geringe Übertragungsreichweite von 10 Metern nicht aus, um auf einem Spielfeld (Beispiel Uni-Campus Haarentor: ca. 400 000 Quadratmeter) eine flächendeckende Datenübertragung zu gewährleisten. Für die Erkennung von anderen Geräten in der unmittelbaren Umgebung ist Bluetooth dagegen gut geeignet.

Zusätzliche Hardware / Infrastruktur Keine zusätzliche Harware (bei integriertem Bluetooth-Modul) erforderlich / Keine zusätzliche Infrastruktur erforderlich

Kosten im Einsatz: Es fallen keine laufenden Kosten an

Latenzzeit: 60-140 ms

Kontinuität der Datenübertragung Kann abreißen

Mobiler Einsatz: Aufgrund der geringen Reichweite nicht für den mobilen Einsatz geeignet

Wetterabhängigkeit: Unabhängig gegenüber Wettereinflüssen

12.5.4 WLAN

Der Begriff Wireless LAN oder auch WLAN wird häufig auf zwei verschiedene Arten verwendet. Zum einen als Sammelbegriff für drahtlose lokale Netzwerke, zum anderen als Bezeichnung für alle Netzwerke des Standards IEEE 802.11. In diesem Abschnitt wird WLAN als Bezeichnung für Netzwerke des Standards IEEE 802.11 verwendet. Der seit 1997 existierende Standard IEEE 802.11 (WLAN) ist der wohl am weitesten verbreitete Standard für drahtlose lokale Netze. Der Standard garantiert eine Mindestbandbreite von einem MBit/s, die optional auf zwei MBit/s erweiterbar ist.

IEEE 802.11b
Die Erweiterung 802.11b bietet eine maximale Bandbreite von 11 MBit/s. IEEE 802.11b sendet im 2,4 GHz Frequenzband mit einer maximalen Sendeleistung von 100 mW. 802.11b ist momentan die WLAN-Variante mit der größten Verbreitung.

IEEE 802.11a
Die Erweiterung 802.11a bietet eine maximale Bandbreite von 54 MBit/s. 802.11a sendet im 5 GHz Frequenzband mit einer maximale Sendeleistung von 200 mW. Die Verbreitung von Geräten, die IEEE 802.11a unterstützen, ist in Deutschland gering. Außerdem ist sie Nutzung dieser Norm auf den Einsatz im Indoorbereich in Deutschland beschränkt.

IEEE 802.11g
Die Erweiterung 802.11g bietet eine maximale Bandbreite von 54 MBit/s. 802.11g verwendet wie die Variante 11b das 2,4 GHz Frequenzband mit einer maximalen Sendeleistung von 100 mW. Zudem ist 802.11g abwärtskompatibel zu 11b, auch wenn im Kompatibilitätsmodus die Leistungsfähigkeit der 802.11g Clients verringert wird. Die Reichweite eines WLAN-Clients ist abhängig von der eingesetzten Sendeleistung, Polarisation, verwendeter Antennen und besonders stark von der Einsatzumgebung. Im Allgemeinen liegt die Reichweite in Gebäuden zwischen 30 bis 60 Metern und zwischen 200 bis 300 Metern im Freien.

Topologisch unterscheidet man zwischen drei verschiedenen Betriebsmodi von WLAN:

- Im **Infrastruktur-Modus** erfolgt die Anbindung an des Netzwerk über eine Basissattion, den so genannten Accesspoint. Der Accesspoint besitzt dabei sowohl eine drahtlose Anbindung gemäß IEEE 802.11 als auch eine drahtgebundene Anbindung und dient als Vermittler zwischen beiden Netzen.
- Im **Ad-hoc-Modus** gibt es, im Gegensatz zum Infrastruktur-Modus, keine fest Basisstation. Eine Anbindung an ein festes Netzwerk steht nicht zur Verfügung und es können nur Stationen miteinander kommunizieren, die sich in Kommunikationsreichweite befinden.
- Vom **Distribution-Systems** spricht man, wenn mindestens zwei miteinander verbundene Funkzellen betrieben werden. Dabei ist die Art der Vernetzung der einzelnen Funkzellen frei wählbar. So ist es möglich über ein 802.x Netz mehrere Funkzellen transparent miteinander zu vernetzen.

Die lokale Anordnung wird hierdurch nicht eingeschränkt. Zellen können komplett separiert, zur Erhöhung der Netzkapazität gänzlich kongruent sein oder nur teilweise überlappen um die räumliche Verfügbarkeit zu erhöhen.

Abbildung 12.18: WLAN-Abdeckung des Campusgeländes der „Carl von Ossietzky Universität Oldenburg“

12.5.4.1 Bewertung

Übertragungsrate: Die theoretische Übertragungsrate liegt bei IEEE 802.11b bei 5,5-11 Mbit/s und bei 802.11a/g bei 6-54 Mbit/s. In der Praxis liegen die Übertragungsraten aber deutlich darunter, 5 Mbit/s netto bei 802.11b und 32 Mbit/s netto bei 802.11a/g. Die Tatsache, dass der iPAQ im Test ca. ein Achtel dieses Werts erreicht, kann darauf zurückgeführt werden, dass der PDA generell keinen höheren Datendurchsatz unterstützt.

Reichweite: Die Reichweite liegt prinzipiell bei 30 bis 60 Metern in Gebäuden und 200 bis 300 Metern im Freien. Bei mobilen Endgeräten wie zum Beispiel PDAs, kann die Reichweite aber auch deutlich darunter liegen. Der Grund ist eine verringerte Sendeleistung, um Energie auf den mobilen Endgeräten zu sparen.

Verbreitung: In neueren Notebooks ist ein WLAN-Modul heute Standard. In mobilen Endgeräten, wie zum Beispiel PDAs, nur in den höherwertigen Geräten.

Spielbarkeit: Aufgrund der begrenzten maximalen Reichweite der WLAN-Übertragung von bis zu 300 Metern ist eine fächendeckende Übertragung von Daten auf Spielfeldern mit einer Ausdehnung von mehreren hundert Metern und größer nicht gewährleistet. Im Unterschied zur Bluetooth-Übertragung können die Geräte in der Regel nicht ohne größeren Aufwand gekoppelt werden (Einwahl bei Accesspoints), so dass WLAN auch für die Erkennung von Geräten in unmittelbarer Reichweite nur in speziellen Fällen geeignet scheint.

Zusätzliche Hardware / Infrastruktur Wenn ein WLAN-Client vorhanden ist, dann wird keine zusätzliche Hardware benötigt. Die nötige Infrastruktur ist bedingt vorhanden (siehe Abb. 12.18 auf S. 229) oder im Ad-Hoc Modus nicht nötig.

Kosten im Einsatz: Abgesehen von den Kosten für die Hardware fallen keine laufenden Kosten an. Eine Ausnahme würde der Einsatz von kommerziellen Hot-Spots bedeuten, für deren Benutzung Kosten anfallen würden.

Latenzzeit: Die Latenzzeit liegt, unabhängig vom eingesetzten Standard, zwischen 2 bis 20 ms

Kontinuität der Datenübertragung Reißt selten ab

Mobiler Einsatz: Im Rahmen einer Funkzelle (Accesspoint) ist der mobile Einsatz im Rahmen des Agentenspiels problemlos möglich. Soll sich der Einsatz über mehrere Funkzellen erstrecken, ist ein Distributions-System nötig oder eine andere Art des Zugriffsmanagements.

Wetterabhängigkeit: Der WLAN-Standard wird vom Wetter nicht beeinflusst

12.5.5 GPRS (General Packet Radio Service)

Innerhalb der Weiterentwicklung des GSM Netzes wurde der GPRS-Dienst ermöglicht. GPRS bietet neben dem kanalvermittelnden Datendienst von GSM einen paketorientierten Dienst. Dabei ist GPRS für Bereiche von Anwendungen vorgesehen, die regelmäßig kleinere Datenmengen übertragen. GPRS erlaubt den Zugang in verschiedene existierende Netze, beispielsweise zu Netzwerken, die auf IP oder X.25 basieren. GPRS erlaubt eine theoretische Übertragungsrate von bis zu 171.2 kbit/s. Diese wird aber nur erreicht, wenn die Empfangsqualität optimal ist und alle 8 zur Verfügung stehenden Funkkanäle gebündelt werden. Die praktische Übertragungsrate hängt aber davon ab, wie hoch das Datenaufkommen aller Funkteilnehmer einer Zelle ist, da sie sich die gesamte Bandbreite teilen müssen.

GPRS-Geräte werden anhand zweier Klassifizierungen unterschieden. Die erste Klasse gibt an, in wieweit Endgeräte gleichzeichtig Sprach- und Datenverbindungen aufrechterhalten können. Die zweite Klasse gibt die Multislot-Klassen an (siehe Tabelle 12.1 auf Seite 231). Die Multislot-Klassen geben an, in wieweit Endgeräte in der Lage sind, mehrere Funkkanäle gleichzeitig zu nutzen. Dabei stehen für jeden Funkkanal theoretisch 21,4 kBit/s zur Verfügung, ein realistischer Wert ist aber eher 13,4 kBit/s. Damit hat ein Endgerät mit Multislot-Klasse 10 eine realistische Übertragungsrate von 53,6 kBit/s in Empfangsrichtung und 26,8 kBit/s in Senderichtung.

- **Klasse A:** Gleichzeitige leitungsvermittelte (GSM) und paketvermittelte (GPRS) Übertragungen sind möglich.

- **Klasse B:** Gleichzeitige leitungsvermittelte (GSM) und paketvermittelte (GPRS) Übertragungen sind nicht möglich. Bei bestehender GPRS Übertragung werden ankommende GSM-Anrufe nur gemeldet.
- **Klasse C:** Leitungsvermittelte (GSM) und paketvermittelte (GPRS) Übertragungen müssen manuell aktiviert werden, eine gleichzeitige Nutzung ist nicht möglich.

Klasse	Empfangskanäle maximal	Sendekanäle maximal	Kanäle gesamt maximal
1	1	1	2
2	2	1	3
3	2	2	3
4	3	1	4
5	4	2	4
6	3	2	4
7	3	3	4
8	4	1	5
9	3	2	5
10	4	2	5
11	4	3	5
12	4	4	5

Tabelle 12.1: Multislot-Klassen

12.5.5.1 Bewertung

Übertragungsrate: Die Übertragungsrate ist abhängig vom eingesetzten Gerät und Auslastung der Funkzelle. Als realistischer Wert wird 13,4 kBit/s angenommen. Steht kein GPRS zur Verfügung, kann auf Basis von GSM eine Übertragungsrate von 9,6 kbit/s erreicht werden.

Reichweite: Es gibt grundsätzlich keine Beschränkung in der Reichweite. Das ganze Bundesgebiet ist mit GSM und GPRS Technik erschlossen

Verbreitung: Heute beherrschen nahezu alle Handys GSM und GPRS

Spielbarkeit: Aufgrund der flächendeckenden Versorgung scheint die GPRS-Technologie als gut geeignet für die Umsetzung des Spiels.

Zusätzliche Hardware / Infrastruktur Die Infrastruktur ist vorhanden, als zusätliche Hardware wird ein Handy mit GSM/GPRS Modem benötigt

Kosten im Einsatz: Zuzüglich zu den Kosten der Hardware, muss die übertragene Datenmenge extra bezahlt werden. Die Preise liegen zurzeit bei ungefähr 5 Euro für 2 MB Datenvolumen.

Latenzzeit: Die Latenzzeit liegt um die 1150 ms (Durchschnittswert, vgl. Abb. 12.19 auf Seite 233)

Kontinuität der Datenübertragung Die Datenübertragung ist äußerst stabil und reißt sehr selten im Betrieb ab

Mobiler Einsatz: Ein mobiler Einsatz ist uneingeschränkt möglich

Wetterabhängigkeit: Der GSM/GPRS wird vom Wetter nicht beeinflusst

12.5.6 UMTS (Universal Mobile Telecommunications System)

Ähnlich GPRS handelt es sich bei UMTS (Universal Mobile Telecommunications System) um eine Weiterentwicklung von GSM. Hiermit entsteht weltweit erstmalig ein einheitliches Mobilfunknetz. Mit UMTS werden Datendurchsätze von bis zu 2 Mbit/s erwartet. Obwohl sich Funkwellen mit Lichtgeschwindigkeit ausbreiten, kann UMTS die höchste Transferrate nicht mehr garantieren, wenn sich der Empfänger mit mehr als 10 Metern pro Sekunde bewegt. Die oben genannten 2 Mbit/s können daher nur als theoretischer Wert verstanden werden, die nur in einer drahtlosen Übertragung im Nahbereich erreicht werden kann, etwa im Bereich einer Pikozelle. In einer Pikozelle ist die Anbindung entweder stationär, oder der Empfänger darf sich nicht mit mehr als 10 Metern pro Sekunde fortbewegen. Folgende Datenraten sind möglich:

- **Makroebene:** Makrozellen haben eine Größe von einigen 10 Kilometern, bei einer Übertragungsrate von mindestens 144 KBit/s, bei einer maximalen Bewegungsgeschwindigkeit des Endgerätes von 500 Km/h
- **Mikroebene:** Mikrozellen haben eine Größe von bis zu einigen Kilometern, bei einer Übertragungsrate von 384 KBit/s, bei einer Bewegungsgeschwindigkeit von maximal 120 Km/h
- **Pikoebene**: Pikozellen haben eine Größe von 50 Metern, bei einer Übertragungsrate von 2MBit/s, bei maximal 10 Km/h. UMTS ist damit bis zu 30 mal schneller als ISDN und bis zu 200 mal schneller als heutige GSM-Handys

12.5.6.1 Bewertung

Übertragungsrate: Die garantierte Übertragungsrate ist abhängig von der Geschwindigkeit und des Anbieters. Theoretische sind 2 Mbit/s möglich

Reichweite: Es gibt grundsätzlich keine Beschränkung in der Reichweite. Zurzeit steht in allen größeren deutschen Städten UMTS zur Verfügung.

Verbreitung: UMTS-Handys sind noch selten, die Verbreitung wird aber voraussichtlich in diesem Jahr stark steigen

Spielbarkeit: UMTS scheint aufgrund der hohen Übertragungsraten bei gleichzeitig hoher Mobilität ideal für einen Einsatz in dem Spiel geeignet. Momentan ist aber eine flächendeckende Versorgung von UMTS-Verbindungen nicht gewährleistet, sondern befindet sich im Aufbau.

Zusätzliche Hardware / Infrastruktur Die Infrastruktur ist in Oldenburg vorhanden, als zusätliche Hardware wird ein UMTS-Handy benötigt

Kosten im Einsatz: Zuzüglich zu den Kosten der Hardware, muss die übertragene Datenmenge extra bezahlt werden. Die Preise liegen zurzeit bei ungefähr 5 Euro für 2 MB Datenvolumen.

Latenzzeit: Die Latenzzeit liegt um die 500 ms (Durchschnittswert, vgl. Abb. 12.19 auf Seite 233)

Kontinuität der Datenübertragung Die Datenübertragung ist äußerst stabil und reißt sehr selten im Betrieb ab

Mobiler Einsatz: Ein mobiler Einsatz ist, innerhalb der mit UMTS erschlossenen Gebiete, uneingeschränkt möglich

Wetterabhängigkeit: UMTS wird vom Wetter nicht beeinflusst

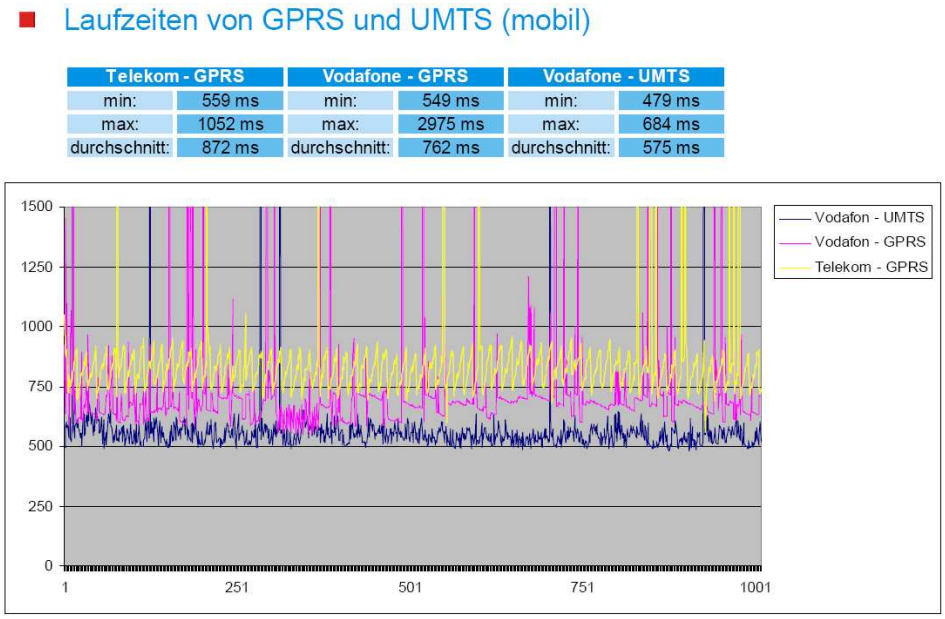


Abbildung 12.19: Laufzeiten von GPRS und UMTS

12.5.7 Fazit

Aufgrund der geringen Reichweite bei der Datenübertragung erscheinen Infrarot und Bluetooth für einen Einsatz bei der Entwicklung des Spiels nicht geeignet. WLAN, GPRS und UMTS erfüllen hingegen die wichtigsten Kriterien. Die Vorzüge dieser Technologien in Bezug auf Übertragungsgeschwindigkeit im Verhältnis zur Übertragungsreichweite wird durch die Abbildung 12.20 auf Seite 234, in Bezug auf Übertragungsgeschwindigkeit im Verhältnis zur Bewegungsgeschwindigkeit durch die Abbildung 12.21 auf Seite 234 verdeutlicht.

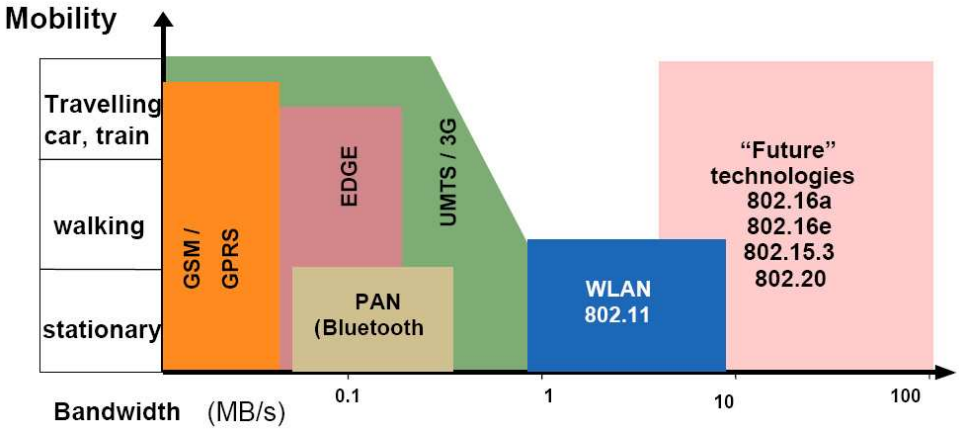


Abbildung 12.20: Übertragungsgeschwindigkeit im Verhältnis zur Bewegungsgeschwindigkeit

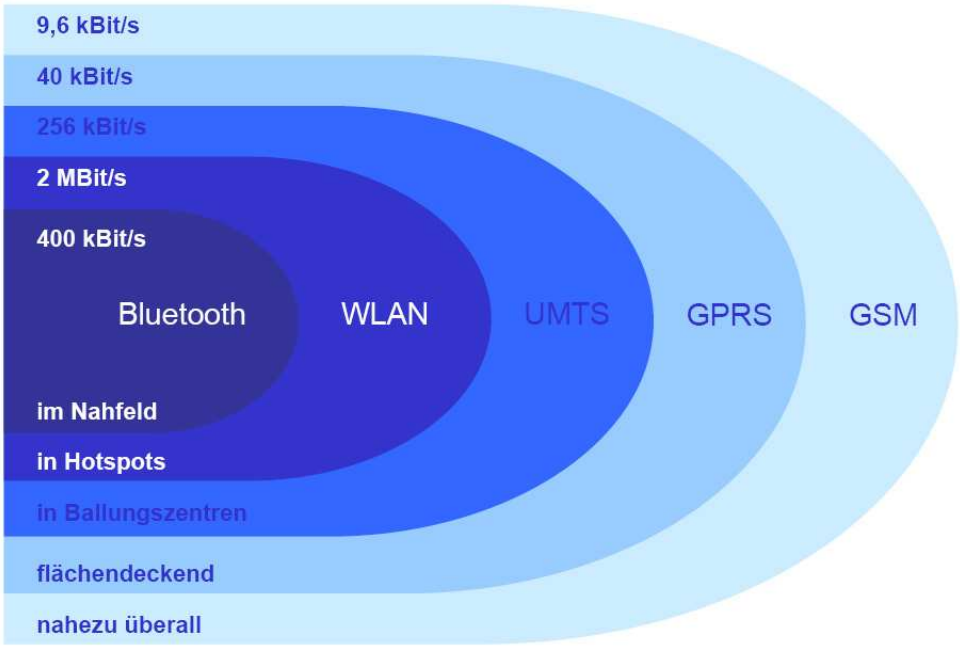


Abbildung 12.21: Übertragungsgeschwindigkeit im Verhältnis zur Reichweite

12.6 Positionsbestimmung

12.6.1 Einleitung

In den folgenden Kapiteln möchten wir den Teilnehmern der Projektgruppe einen Überblick über die momentan existierenden Methoden zur Positionsbestimmung geben.
Wir verzichten auf allgemeine Erklärungen zum Thema Positionsbestimmung wie z.B. Triangulierung oder Lateration, da uns dies keinen wirklichen Nutzen für die Projektgruppe einbringen würde.
Bei jedem vorgestellten System werden wir jedoch auch die Methode der Positionsbestimmung erläutern, so dass die Teilnehmer der Projektgruppe einen groben Überblick über die Funktionsweise der einzelnen Systeme erhalten.

12.6.2 Infrarot-Systeme

12.6.2.1 Active Badge

Das Active Badge System wurde in den Jahren 1989-1992 im Olivetti & Oracle Research Laboratory in Cambridge entwickelt. Grundlage dieses Systems ist ein Netzwerk von Infrarotsensoren, das mit einem so genannten Location Server verbunden ist. In den existierenden Anwendungen sind die Sensoren so verteilt, dass meist ein Sensor pro Raum installiert wurde.
Am zu lokalisierenden Objekt befindet sich ein Infrarottransmitter, die so genannte Badge, welche alle 10 Sekunden für jeweils 0.1 Sekunden ein Infrarotsignal aussendet, in dem ihre ID kodiert ist. Dieses Signal wird dann von einem (oder mehreren) Sensor(en) empfangen und in Verbindung mit der eigenen Identität an den Location Server weitergeleitet. Dieser sammelt die erhaltenen Daten und ermittelt aus diesen und anhand der Kenntnis über die Standorte der Sensoren die Position der Badge.

Diese Art der Positionsbestimmung ist im Allgemeinen jedoch nur raumgenau, da z.B. auf die Messung von Signallaufzeiten verzichtet wird und dieses System auch nur für die raumgenaue Ortung von Einrichtungsgegenständen konzipiert wurde.

Die Positionsbestimmung auf Basis des Active Badge Systems kommt für den Einsatz in unserer Projektgruppe nicht in Frage, da Infrarotlicht mit dem Sonnenlicht interferiert und somit für den Einsatz im Freien nur bedingt geeignet ist. Außerdem reicht die Genauigkeit dieses Systems für einen Einsatz in unserer Projektgruppe leider nicht aus.

12.6.2.2 WIPS

Das Wireless Indoor Positioning System ist quasi die Umkehrung des Active Bat Systems.

Beim WIPS sind die Infrarotsender (Bake) fest installiert (meist 1 Sender pro Raum) und über WLAN mit einem Location Server verbunden und der Empfänger, die so genannte Badge, befindet sich am zu lokalisierenden Objekt. Die Sender senden periodisch ihre Benutzerkennung aus, die dann von der Badge empfangen wird. Die Badge leitet die empfangenen Signale über WLAN an den Location Server weiter, der anhand der empfangenen Daten und dem Wissen über die Standorte der Sender die Position der Badge bestimmt. Anschließend sendet der Location Server die Position der Badge an diese zurück.

Diese Art der Positionsbestimmung ist im Allgemeinen jedoch nur raumgenau, was den Einsatz in unserer Projektgruppe nicht empfehlenswert machen würde. Außerdem interferiert Infrarotlicht mit dem Sonnenlicht und somit ist dieses System für den Einsatz im Freien nur bedingt geeignet.

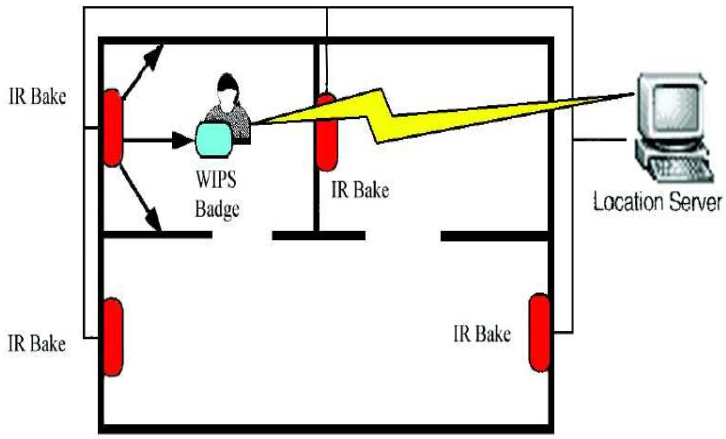


Abbildung 12.22: Funktionsweise des WIPS

12.6.3 Ultraschall-Systeme

12.6.3.1 Active Bat

Das Active Bat System ist eine Weiterentwicklung des Active Badge System durch das AT&T. Grundlage dieses Systems ist ein gitterförmiges Netzwerk von Deckensensoren, das mit einem so genannten Location Server verbunden ist. Diese Sensoren sind so im Raum verteilt, dass 1 Sensor pro Quadratmeter vorhanden ist. Am zu lokalisierenden Objekt befindet sich ein Ultraschalltransmitter, die so genannte Bat, welche über Funk mit dem Location Server verbunden ist. Jede Bat besitzt eine einzigartige ID, um die Kommunikation zwischen Bat und Location Server zu ermöglichen.

Soll die Position einer Bat bestimmt werden, sendet der Location Server einen Funkimpuls (1) an die jeweilige Bat und gleichzeitig ein Resetsignal an das Sensorennetzwerk. Aufgrund dieses Requests sendet die Bat ein Ultraschallsignal aus (2), welches von den Sensoren des Netzwerkes empfangen wird. Aus der Differenz zwischen Erhalten des Resets und Empfangen des Ultraschallsignals errechnen die einzelnen Sensoren ihren jeweiligen Abstand zur Bat und geben diesen an den Location Server weiter (3).
Anhand der verschiedenen Abstände und der Kenntnis über die Positionen der einzelnen Sensoren ist der Location Server in der Lage, die Position der Bat in 95% der Fälle auf 9 cm genau zu bestimmen.

Die Positionsbestimmung auf Basis des Active Bat Systems kommt für den Einsatz in unserer Projektgruppe nicht in Frage, da dieses System bisher nur als Forschungsprojekt realisiert wurde. Außerdem entsteht bei der Realisierung ein erheblicher Infrastrukturaufwand (1 Sensor pro m^2), der dieses System zu kostenintensiv für den Einsatz in unserer Projektgruppe werden ließe.

12.6.3.2 Cricket

Das Cricketsystem wurde von einer Forschungsgruppe des Massachusetts Institute of Technology (MIT) in den Jahren 2000-2004 entwickelt bzw. weiterentwickelt und ist quasi des Umkehrung des Active Bat Systems. Beim Cricket System sind nämlich die Sender fest installiert (1 Sender pro $2m^2$) und der Empfänger, der so genannte Listener, befindet sich am zu lokalisierenden Objekt.
Die Bestimmung der Position eines mobilen Endgerätes wird im Cricket System folgendermaßen realisiert: Die fest installierten Sender senden periodisch gleichzeitig sowohl ein Ultraschall- als auch ein Funksignal aus. Diese Signale werden von dem Listener empfangen, der aus der Zeitdifferenz der jeweiligen Signale seine Position berechnet.

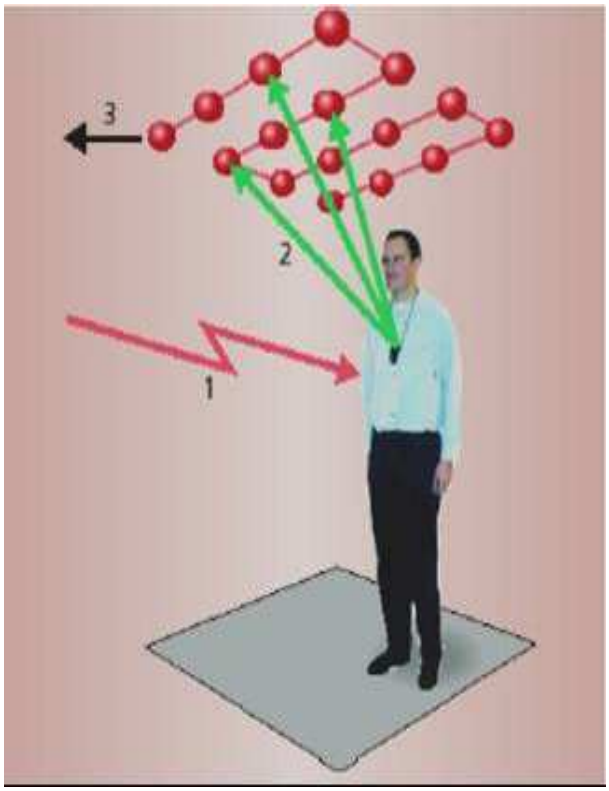


Abbildung 12.23: Funktionsweise des Active Bat Systems

Die Genauigkeit dieses Systems liegt bei ca. 3 cm.

Die Positionsbestimmung auf Basis des Cricket Systems kommt für den Einsatz in unserer Projektgruppe nicht in Frage, da dieses System für den Einsatz innerhalb von Gebäuden konzipiert wurde. Außerdem entsteht bei der Realisierung ein erheblicher Infrastrukturaufwand (1 Sensor pro 2 m^2), der dieses System zu kostenintensiv für den Einsatz in unserer Projektgruppe werden ließe.

12.6.4 Funk-Systeme

12.6.4.1 SpotON

Das SpotON System wurde im Jahr 2000 zum ersten Mal von der Universität Washington vorgestellt. Grundlage dieses Systems ist ein Netzwerk von Funksensoren, das mit einem Location Server verbunden ist.

Am zu lokalisierenden Objekt befindet sich ein Funksender, die so genannte Badge, die, wann immer ihre Position bestimmt werden soll, ein Funksignal aussendet. Dieses Signal wird dann von den installierten Sensoren empfangen und jeder Sensor misst die jeweilige Signalstärke des Funksignals. Diese Signalstärken werden dann über das Netzwerk an den Location Server weitergegeben, welcher anhand der Signalstärken und des Wissens über die Positionen der Sensoren die Position der Badge bestimmen kann. Die Genauigkeit dieses Systems liegt bei bis zu 3 Metern, ist im Allgemeinen jedoch ungenauer.

Die Positionsbestimmung auf Basis des SpotON Systems kommt für den Einsatz in unserer Projektgruppe nicht in Frage, da es sich bei diesem System bis jetzt nur um ein Forschungsprojekt handelt, das zusätzlich für den Einsatz innerhalb von Gebäuden konzipiert wurde.

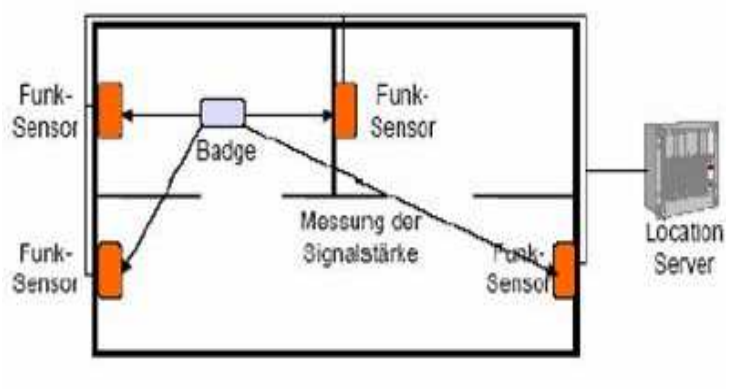


Abbildung 12.24: Funktionsweise von SpotON

12.6.4.2 WLAN

Auch die Positionsbestimmung über WLAN macht sich die Eigenschaft von Access Points zu Nutze, die Signalstärke einer Funkverbindung messen zu können. Bevor jedoch die eigentliche Positionsbestimmung durchgeführt werden kann, muss einiges an Vorarbeit geleistet werden. Das gesammte Areal, auf dem man später die Positionsbestimmung durchführen möchte, muss abgelaufen und eine „Landkarte“ von Signalstärken erstellt werden. Dazu muss an jedem Punkt, der später zur Positionsbestimmung herangezogen werden soll, die Signalstärke zu jeder erreichbaren Basisstation ermittelt werden. Diese Signalstärken werden dann zusammen mit der jeweiligen Position der Messung in einer Datenbank o.ä. gespeichert.

Soll nun die Position eines mit einer WLAN-Karte ausgerüsteten Gerätes bestimmt werden, so wird zunächst die Signalstärke der Funkverbindung zu jedem erreichbaren Access Point bestimmt. Danach werden die Signalstärken mit den Einträgen aus der Datenbank verglichen und der ähnlichste Eintrag als Position gewählt. Dabei kann diese Wahl z.B. auf Basis des Euklidischen Distanzmaßes oder anderen Verfahren beruhen.
Die Genauigkeit dieses Verfahrens liegt, je nach Infrastruktur, bei bis zu einem Meter.

Da für eine einigermaßen genaue Positionsbestimmung mindestens 3 Access Points dauerhaft erreichbar sein müssen, kann dieses Verfahren nicht innerhalb der Projektgruppe eingesetzt werden. Die Infrastruktur für eine Positionsbestimmung mittels WLAN ist schlichtweg nicht vorhanden. (siehe WLAN-Abdeckung des Unigeländes in Abb.12.18 auf S.229)

12.6.5 Bluetooth

Bluetooth ist ein Funkübertragungsstandard, der für die Vernetzung von persönlichen Geräten konzipiert wurde. Bluetooth-Module lassen sich in drei Klassen einteilen, deren Übertragungsreichweite zwischen 10 (Klasse 3) und 100 (Klasse 1) Metern variiert. In den meisten mobilen Geräten werden Module der Klasse 3 eingesetzt, so dass diese Geräte lediglich einen Reichweite von 10 Metern aufweisen.
Diese Reichweite ist jedoch für den Einsatz innerhalb der Projektgruppe zu gering. Es besteht zudem die Möglichkeit, dass die Bluetooth-Signale von anderen Sendern, die im selben Frequenzbereich senden, gestört werden können.

12.6.6 GPS

GPS (Global Positioning System) basiert auf Satelliten, die ständig mit einem eigenen Code Signale ausstrahlen, aus deren Signallaufzeit GPS-Empfänger ihre Position bestimmen können. Theoretisch reichen dazu die Signale von drei Satelliten, da daraus die genaue Position und Höhe des Empfängers bestimmt werden kann. In der Praxis haben aber die meisten GPS-Empfänger keine Uhr, die genau genug ist, um daraus die Laufzeiten korrekt berechnen zu können. Deshalb wird meist das Signal eines vierten Satelliten benötigt.

Mit den GPS-Signalen lässt sich aber nicht nur die Position, sondern auch die Geschwindigkeit des Empfängers bestimmen. Durch die relative Bewegung des Empfängers zu den Satelliten lässt sich diese Geschwindigkeit berechnen.

Damit ein GPS-Empfänger immer zu mindestens vier Satelliten Kontakt hat, werden insgesamt mindestens 24 Satelliten eingesetzt, die die Erde jeden Sternentag zweimal in einer Höhe von 20.200 km umkreisen. Jeweils mindestens vier Satelliten bewegen sich dabei auf jeweils einer der 6 Bahnebenen, die 55° gegen die Äquatorebene geneigt sind und gegeneinander um jeweils 60° verdreht sind.

GPS wurde 1995 offiziell in Betrieb genommen. Es gibt die zwei Dienstklassen SPS (Standard Positioning Service) und PPS (Precise Positioning Service). PPS dient ausschließlich der militärischen Nutzung. SPS wurde anfangs auf eine Genauigkeit von 100 Meter ausgelegt. Durch Abschaltung eines Störsenders im Jahre 2000 wurde die Genauigkeit auf weniger als 10 Meter in 95 Prozent der Fälle verbessert. Eine Erhöhung der Genauigkeit (0,5 - 5 m) kann durch Einsatz von Differential GPS (DGPS) erreicht werden.

Bei dem Verfahren DGPS gibt es einen Empfänger, dessen Position bestimmt werden soll (Rover) und mindestens einen weiteren Empfänger, dessen Position bekannt ist (GPS-Basisstation). Eine Basisstation kann diverse Informationen über die Ursachen ermitteln, warum die mittels GPS bestimmte Position fehlerhaft ist, da deren Position bekannt ist. Mit diesen Informationen (Korrekturdaten) von einer Basisstation kann ein Rover seine Genauigkeit erhöhen. Die erreichbare Genauigkeit ist u.a. vom Abstand zwischen Rover und Basisstation abhängig.

12.6.7 API's

In diesem Abschnitt sollen momentan existierende API's zum Thema Positionsbestimmung untersucht werden, inwiefern sich diese für den Einsatz innerhalb der Projektgruppe eignen.

12.6.7.1 Java Location API

Die Java Location API ist ein optionales Paket für die Java 2 Micro Edition[34]. Sie erlaubt einen standardisierten Zugriff auf Ortsinformationen und Orientierung eines mobilen Endgerätes. So können Positionsinformationen z.B. einmalig abgerufen werden, die Java Location API ermöglicht aber auch periodische Updates der Positionsinformationen.
Unterstützt werden alle gängigen Ortungsmethoden (GPS, E-OTD, usw.), die vorhandene Hardware entscheidet jedoch letztendlich, welche Ortungsmethoden verwendet werden können.
Die Java Location API ist zur Zeit leider noch nicht auf vielen mobilen Endgeräten integriert, momentan wird sie lediglich vom Motorola i860 und Nokias 3. Generation der 60er Serie unterstützt.

[34] die genaue Spezifikation ist unter http://jcp.org/en/jsr/detail?id=179 zu finden

12.6.7.2 Andere Java API's

Eine andere API, die den standardisierten Zugriff auf Positionsinformationen erlaubt ist die *Chaeron GPS Library*. Diese unterstützt zur Zeit die beiden Protokolle NMEA und Garmin und ist unter der GNU General Public License (GPL) lizensiert. Die *Chaeron GPS Library* ist auf mobilen Endgeräten lauffähig mit der *Jump* und der *SuperWaba* Virtual Machine.
Ein besonderes Feature der *Chaeron GPS Library* ist das Speichern der Daten im GPSml Format, einem XML-basierten Austauschformat, das es anderen Anwendungen erlaubt die gespeicherten Daten weiterzuverarbeiten.

Manche Mobilfunkhersteller bieten eigene Implementierungen zum Zugriff auf GPS-Positionsdaten an. So bietet Motorola z.B. eine GPS API, die auf manchen Handys bzw. Smartphones verfügbar ist. Eine Übersicht über die Handys bzw. Smartphones, die diese API unterstützen sucht man im Internet jedoch vergebens.

12.6.7.3 .NET

Auch für die Entwicklung von ortsbasierten Anwendungen unter .NET gibt es eine Vielzahl von Toolkits, API's und Frameworks, jedoch sind diese alle kostenpflichtig. Die Preise für eine Nutzung liegen hierbei zwischen $50 und $1000, was eine Benutzung innerhalb der Projektgruppe schon ausschließt.
Allerdings gibt es im Internet eine Menge guter Tutorials, die die Grundlagen des Zugriffs auf GPS-Informationen mittels .NET erläutern[35], so dass die Gewinnung von Positionsinformationen über GPS auch von der Projektgruppe selber implementiert werden kann.

12.6.8 Fazit

Wie gesehen gibt es viele unterschiedliche Möglichkeiten, um Positionsinformationen zu ermitteln. Leider weisen viele dieser Positionsbestimmungssysteme teilweise erhebliche Probleme auf, die einen Einsatz innerhalb der Projektgruppe ausschließen. So sind viele der vorgestellten Systeme bisher lediglich als Prototypen realisiert oder erfordern einen hohen Infrastrukturaufwand, dass sie für den Einsatz in der Projektgruppe nicht in Frage kommen. Einzig **GPS** bietet keinerlei gravierende Probleme, die gegen einen Einsatz sprechen könnten. Die Infrastruktur ist vorhanden und kann kostenlos genutzt werden und die Kosten für die benötigten Empfänger sind überschaubar bzw. die Empfänger stehen der Projektgruppe kostenlos zur Verfügung. Auch die Genauigkeit der Positionsbestimmung ist für die Zwecke der Projektgruppe absolut ausrechend.

Somit bietet sich die Nutzung von **GPS** als optimalste Lösung an, um innerhalb unserer Projektgruppe die Positionsbestimmung zu übernehmen.

[35] siehe z.B. http://www.codeguru.com/vb/mobile/pocketpc/article.php/c8079/

12.7 Kontext

In diesem Abschnitt werden die Ergebnisse der Technologiestudie zum Thema Kontext prägnant zusammengefasst. Ziel dieser Studie war es, ein gemeinsames Verständnis für Kontext innerhalb der Projektgruppe zu entwickeln. Dies war insbesondere notwendig, da die Berücksichtigung von Kontext einen entscheidenden Aspekt der Aufgabenstellung dieser Projektgruppe ausmacht und lange keine konkrete Vorstellung von Kontext in der Projektgruppe existierte. Im Folgenden wird sich zunächst mit der Frage, was Kontext überhaupt ausmacht, auseinander gesetzt und im Anschluss festgehalten, inwiefern dieser im Agentenspiel in ein Kontextmodell einfließen kann. Abschließend werden die Erkenntnisse der Studie in einem Fazit zusammengefasst und in Zusammenhang mit der Entwicklung des Kontextverständnisses der Projektgruppe über die gesamte Projektdauer gebracht.

12.7.1 Kontextübersicht

Neben dem eigentlichen Kontext werden drei weitere grundlegende Begriffe zum Thema vorgestellt. Dabei handelt es sich um Kontextsensitivität, Kontextkategorien und Mobiler Kontext. Unter Kontext werden alle Informationen verstanden, die für die Interaktion von Benutzer und Anwendung relevant sind. Dazu gehören zum Beispiel: Orte, Personen und Objekte in der Umgebung, Zeit, Temperatur und Aufmerksamkeit des Benutzers. Kontextsensitivität spaltet sich je nach Verwendung des erfassten Kontexts in passive und aktive Kontextsensitivität auf. Von passiver Kontextsensitivität spricht man, wenn der Kontext lediglich erfasst und dem Benutzer eventuell dargestellt wird. Wenn darüber hinaus aufgrund des erfassten Kontexts Anpassungen etwa auf System-, Anwendungs- oder Darstellungsebene erfolgen, wird dies als aktive Kontextsensitivität bezeichnet. Der zu erfassende Kontext unterteilt sich wiederum in drei Kategorien, den technischen, benutzerbezogenen und den physikalischen Kontext. Technischer Kontext beinhaltet unter anderem technische Ressourcen und die Rechenumgebung. Unter benutzerbezogenen Kontext fallen Aspekte wie Benutzerprofil und eventuell der Aufenthaltsort des Benutzers. Als physikalischer Kontext werden Umgebungseigenschaften wie Helligkeit, Temperatur und Geräuschpegel verstanden. Den für das Agentenspiel wohl interessanten Mobilen Kontext betont die Merkmale Positionsinformationen sowie deren Qualität und Verfügbarkeit. Auf etwaige ungenaue Positionsinformationen könnte beispielsweise durch Inferenz aus anderen Kontextinformationen, Interaktion mit dem Benutzer oder aber durch Anpassung der Positionsdarstellung reagiert werden.

12.7.2 Kontextmodell

In das Agentenspiel fließt der Kontext über nahezu alle Spielfunktionen ein. Dazu gehören etwa Neutralisieren, Abhören, Tarnen, usw. Innerhalb des Agentenspiels wurden folgende mögliche Kontextkategorien mit dazugehörigen Merkmalen identifiziert:

- **Spiel**
  Falls es mehrere Spielserver bzw. Spiele auf einem Server gibt, benötigt jedes Spiel eine Identifikation und einen Zeitstempel. Für jedes Spiel müsste ein Szenario (mit Karte) und eine Liste von Mitspielern vorliegen.

- **Benutzer**
  Zu jedem Benutzer wird ein Profil mit seinen Fähigkeiten, seinem Spielmodus (Tarnen, Abhören) und welches Fenster sich aktuell in seiner Ansicht befindet zusätzlich zu personenbezogenen Daten wie dem Namen und Vergleichbarem gespeichert.

- **Technische Ressourcen**
  Dazu gehören Ein- & Ausgabemöglichkeiten, verfügbare Netzanbindung, Server und Szenarien.

- **Physikalische Bedingungen**
  Angedacht sind die Berücksichtigung von Geschwindigkeit, Helligkeit und dem Geräuschpegel.

- **Mobiler Kontext**
  Der Mobile Kontext beinhaltet die eigene Position, sowie deren Verfügbarkeit und Qualität, die Nähe zu Personen, Goodies, Kameras und Missionszielen.

Mögliche Quellen für die Kontextinformationen können dabei die Logik des Servers und des Clients, sowie die Sensorik des Clients und die Benutzer selbst sein.

Auf dem Client ist ein Kontextmanager vorgesehen, der die eingehenden Kontextinformation aus den Kontextquellen zunächst in geeigneten Datenstrukturen speichert, um diese möglichst im Sinne aktiver Kontextsensitivität zu verwenden. Mögliche Anfragen an den Kontextmanager wären etwa nach der momentanen Bewegungsgeschwindigkeit oder der Netzanbindung. Graphisch ist diese Systematik in Abbildung 12.25 auf Seite 242 dargestellt.

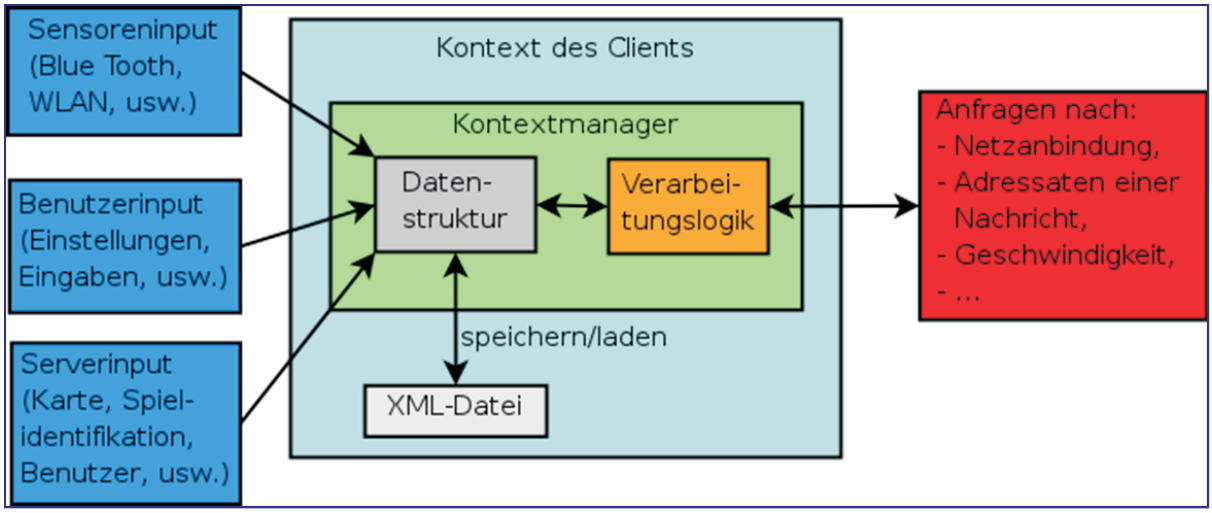


Abbildung 12.25: Mögliches Kontextmodell

Möglicher zu berechnender Kontext auf dem Client könnte die Wahl des Kommunikationskanals zum Server, mögliche Adressaten einer Nachricht, die Geschwindigkeit ohne dafür einen eigenen Sensor zu benutzen und die Qualität der Positionsinformationen sein.

12.7.3 Fazit

Allgemein können jegliche Informationen, die für das Zusammenspiel von Benutzer und Anwendung relevant sind, als potentieller Kontext angesehen werden. Insbesondere kann festgehalten werden, dass die bereits in Abschnitt 12.7.2 aufgeführten Kontextkategorien Spiel, Benutzer, Technische Ressourcen, Physikalische Bedingungen und Mobiler Kontext für die Verwendung im Agentenspiel in Frage kommen. Inwieweit der Kontextmanager Kontextinformationen nur verwaltet oder aus ihnen neue Informationen gewinnt, ist im Entwurf endgültig zu klären. Eine ausführliche Ausarbeitung zum Thema Kontext und dessen Verwendung ist in der Seminararbeit von Michael Onken zu finden.

Im Verlauf des Projekts haben sich die hier aufgeführten Kontext-Konzepte weiter entwickelt. Aufgrund dessen sind die obigen Ausführungen zum Kontext als eine frühe Version anzusehen. Mehr und aktuelleres in Bezug auf das Agentenspiel zum Thema Kontext ist in den Abschnitten B.6 und 13.3 zu finden.

12.8 Entscheidung

Nach ausführlicher Diskussion in der Projektgruppe wurden folgende Technologien zur Realisierung des Agentenspiels ausgewählt. Als Zielplattform fiel die Entscheidung auf das .NET Compact Framework. Zum einen erfüllt das .NET Compact Framework alle K.O. Kriterien, außerdem eignet es sich besonders für die Entwicklung für Windows Mobile 2003 Geräte, wie die vom Offis zur Verfügung gestellten IPAQs 5450, für die sich die Gruppe ebenfalls entschied. Die Datenübertragung soll mittels GPRS erfolgen. Obwohl sich WLAN besser geeignet hätte, führte die Tatsache, dass WLAN am Campus der Universität Oldenburg nicht flächendeckend zur Verfügung steht, zur Entscheidung GPRS als Datenübertragungstechnologie einzusetzen. GPRS ist flächendeckend verfügbar und die notwendigen Geräte, in diesem Fall Handys mit Softmodems, können von den Mitgliedern der Gruppe gestellt werden. Die Positionsbestimmung wird über GPS erfolgen. GPS ist als einzige unter den untersuchten Technologien ausgereift und ohne gravierende Probleme einsetzbar. Eine Entscheidung bezüglich des im Agentenspiel eingesetzten Kontexts wurde in die Entwurfsphase verschoben.

Kapitel 13

Entwurf

13.1 Einleitung

Dieses Dokument stellt den konzeptionellen Entwurf des Spiels NABB (*Not Another Boring Brettspiel*) dar und wurde im Rahmen der Projektgruppe eXplorer "Kontext-sensitive Umgebungserkundung mit mobiler multimodaler Unterstützung" erstellt.

Mobile Anwendungen zur Navigation in realen Umgebungen bekommen eine immer größere Bedeutung. Diese Anwendungen bieten im Allgemeinen eine Basisfunktionalität zur Darstellung der aktuellen Positionen eines Benutzers auf einer Karte, zur Eingabe eines Zielpunktes und zum Berechnen der kürzesten Route zu diesem Zielpunkt. Verwendet werden dazu geographische Informationssysteme in Verbindung mit GPS-Informationen. Navigationssysteme in Autos sind hierfür ein Beispiel. Die steigende Verbreitung dieser Kommunikationstechnologien ermöglicht die Umsetzung neuartiger Spielkonzepte. Ziel dieses Projekts ist die Entwicklung eines Spiels, das die reale Welt mit virtuellen Inhalten verbindet. Die Spieler kommunizieren untereinander und interagieren mit dem Spiel mittels mobiler Endgeräte. Eine Besonderheit der Anwendung ist die Auswahl der Interaktionsmodalitäten anhand des Nutzerkontextes.

Dieses Dokument gliedert sich in folgende Abschnitte:

Abschnitt 13.1: Enthält eine Einleitung in dieses Dokument.

Abschnitt 13.2: Enthält das Fachliche Modell des Spiels. Es wird ein Überblick über das Spiel gegeben, die Regeln und die Interaktionsmöglichkeiten der Spieler mit dem Spiel erläutert bzw. grafisch dargestellt.

Abschnitt 13.3: Enthält eine Einführung in die Erfassung und Modellierung von Kontextinformationen und dessen Umsetzung innerhalb des Agentenspiels.

Abschnitt 13.4: Beschreibt den Architekturentwurf. Es werden die Einflussfaktoren auf die Architektur identifiziert und analysiert, eine Risikoanalyse durchgeführt und die Architektur aus Kontext-, Struktur-, Verhaltens- und Abbildungssicht beschrieben.

Abschnitt 13.5: Beschreibt den Entwurf der Nutzungsschnittstelle des Agentenspiels.

Abschnitt 13.6: Enthält die Modellierung des Agentenspiels. Die Entwürfe der verschiedenen Klassen sind gegliedert in Server, Client und gemeinsam genutzte Klassen, im Folgenden als Common bezeichnet. Diese enthalten neben der textuellen Beschreibung der Klassen auch Klassendiagrammen, welche die Interaktion zwischen den einzelnen Klassen, bzw. Modulen darstellen.

13.2 Fachliches Modell

In diesem Abschnitt wird mit Hilfe eines fachlichen Modells das Agentenspiel NABB[1] beschrieben. Im fachlichen Modell werden die für die Darstellung von NABB relevanten Zustände erläutert, die das System einnehmen kann. Außerdem wird ein Überblick darüber gegeben, inwiefern sich diese Zustände gegenseitig beeinflussen und es werden die Spielregeln anhand des Spielablaufs erklärt.

Das Regelwerk des Agentenspiels beruht auf einer Reihe von Zustandsvariablen, aus denen im Prinzip alle möglichen Kombinationen gebildet werden müssten. Da viele Zustandsvariablen jedoch voneinander unabhängig sind, wird die Darstellung der Zustandsdiagramme optimiert, indem das Regelwerk durch nebenläufige Zustandsdiagramme modelliert und nach Abhängigkeiten der einzelnen Diagramme untereinander untersucht wird.
Die Darstellung der Diagramme erfolgt in der Notation von UML 1.1.

13.2.1 Spielidee

Es gibt die beiden Teams der *Defender* und der *Infiltratoren*, auf die die Spieler gleichmäßig verteilt werden. Im Team der *Infiltratoren* nimmt ein Spieler die Rolle eines besonderen Agenten, des *Commanders*, ein. Nur der *Commander* kennt das Missionsziel seines Teams und muss die Vorgehensweise seiner Mitspieler koordinieren, um das Missionsziel zu erfüllen und somit das Spiel gewinnen zu können. Im Gegenzug versuchen die *Defender*, die *Infiltratoren* an der Erfüllung ihrer Aufgaben zu hindern.

13.2.2 Spielbeginn

Vor dem eigentlichen Spielbeginn stellt sich jeder Mitspieler ein Spielerprofil nach seinen Vorlieben zusammen. Dies erfordert zum einen die Festlegung eines Spielernamens, unter dem im späteren Spielverlauf z.B. Nachrichten verschickt werden können. Zum anderen kann der Spieler Eigenschaftspunkte auf die 4 Grundfähigkeiten *Abhören*, *Aufklären*, *Neutralisieren* und *Tarnen* verteilen. Dabei stehen ihm insgesamt 9 Eigenschaftspunkte zur Verfügung, die beliebig auf die Fähigkeiten verteilt werden können. Hierbei ist jedoch zu beachten, dass auf eine Fähigkeit mindestens 1 Punkt verteilt werden muss und maximal 4 Punkte verteilt werden können. Grundsätzlich gilt: Je mehr Punkte auf eine Fähigkeit verteilt wurden, desto erfolgreicher kann diese im späteren Spielverlauf eingesetzt werden. Eine detailliertere Beschreibung der 4 Grundfähigkeiten erfolgt in Kapitel 13.2.3.3.

Hat der Spieler sein Spielerprofil angelegt, so kann er an einem Spiel teilnehmen. Wenn er einem Spiel beigetreten ist, wird er über seine Teamzugehörigkeit informiert. Der erste Spieler, der dem Spiel als Mitglied der *Infiltratoren* beitritt, wird zum *Commander* ernannt. Dann muss jeder Spieler seinen ihm zugewiesenen Startpunkt aufsuchen, damit seine Fähigkeiten aktiviert werden und er das Spielgeschehen aktiv beeinflussen kann.

Spielzustand Der Spielzustand (Abb. 13.1) beschreibt den Zustand eines Spielers während eines Spiels. Ein Spieler befindet sich zu Spielbeginn, oder wenn er neutralisiert wurde, im Zustand *Passiv*. Er kann somit das Spielgeschehen nicht aktiv beeinflussen. Dies ist nur im Zustand *Aktiv* möglich. Ist das Spiel beendet, wechseln alle Spieler in den Zustand *Spielende*.

- **Passiv** (initial): Ein Spieler wechselt in den Zustand *Passiv*, wenn er dem Spiel beitritt oder neutralisiert wurde. In diesem Zustand kann der Spieler die Karte erkunden. Außer seinem Startpunkt

[1] NABB steht für Not Another Boring Brettspiel

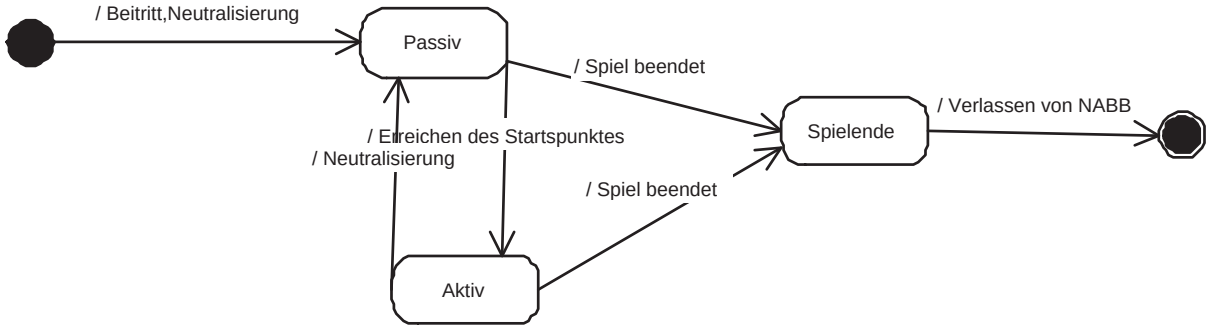


Abbildung 13.1: Spielzustand

wird ihm jedoch kein POI (Point-Of-Interest) angezeigt. Erreicht er diesen, bevor das Spiel zu Ende ist, wechselt er in den Zustand *Aktiv*.

- **Aktiv**: Im Zustand *Aktiv* nimmt ein Spieler aktiv am Spielgeschehen teil. Ihm stehen sämtliche Funktionalitäten zur Verfügung. So kann ein Spieler z.B. Nachrichten anzeigen und verfassen und mit in der Nähe befindlichen POIs interagieren. Mögliche Ausnahmen zur Ausführung der Funktionalitäten werden in den anderen Zustandsdiagrammen definiert.

- **Spielende**: Ein Spieler wechselt in den Zustand *Spielende*, wenn das Spiel beendet ist. Er kann dabei wie im Zustand *Passiv* nicht mehr auf seine Funktionalitäten zugreifen und es gibt auch keinen Startpunkt mehr.

13.2.3 Spielverlauf und Regeln

Allen Spielern ist es gestattet, sich frei auf dem Spielfeld zu bewegen. In der Regel sieht jeder Spieler die Positionen seiner Teammitglieder auf der Übersichtskarte, solange diese oder er selber nicht getarnt sind. Spieler des gegnerischen Teams werden nur dann auf der Übersichtskarte angezeigt, wenn diese sich im Aufklärungsradius des Spielers oder einer vom Spieler gesetzten Aufklärungskamera befinden. Ist ein Spieler getarnt, so kann er weder Teammitglieder noch Spieler des gegnerischen Teams auf der Übersichtskarte sehen. Er kann allerdings seinerseits auch nicht von Teammitgliedern und gegnerischen Spielern auf deren Übersichtskarte erkannt werden, so lange er sich nicht im Sichtradius einer Aufklärungskamera befindet. Sofern ein Spieler nicht neutralisiert ist, kann er seinen Teammitgliedern jederzeit eine Nachricht zuzusenden. Diese Nachricht kann entweder an einzelne Teammitglieder oder an das gesamte Team verschickt werden.

Hauptinteraktion Der Zustandsautomat in Abb. 13.2 modelliert, welche Hauptinteraktion für einen Spieler gerade im Vordergrund steht. Dieser Zustand ist vor allem vom aktuellen Spielzustand abhängig. So kann der Spieler z.B. nur im Spielzustand *Aktiv* alle Interaktionsmöglichkeiten ausschöpfen. In den anderen Spielzuständen steht ihm nur eine eingeschränkte Umgebungserkundung zur Verfügung.

- **Passive Umgebungserkundung** (initial): Ein Spieler wechselt in diesen Interaktionszustand, wenn er sich im Spielzustand *Passiv* befindet. Die Anwendung vermittelt ihm dabei geographische Informationen über das Gebiet, in dem er sich aufhält, damit er durch seine Umgebung navigieren kann. Dazu wird die Lage des Startpunkts dargestellt, den der Spieler erreichen muss, um in den Zustand *Aktiv* zu wechseln.

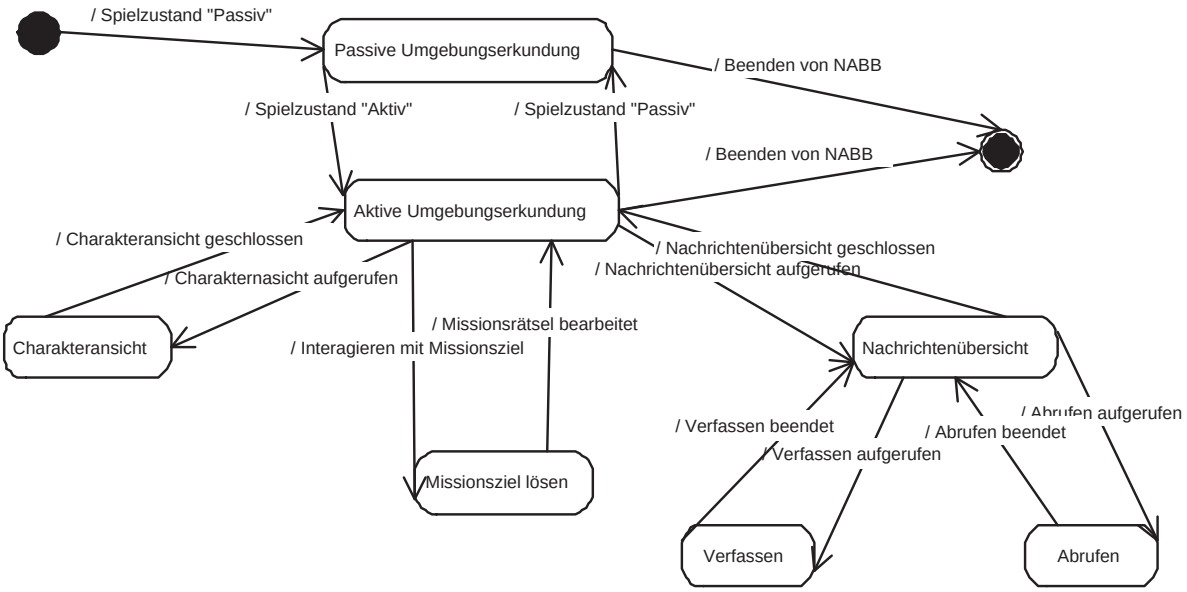


Abbildung 13.2: Hauptinteraktion

- **Aktive Umgebungserkundung**: Der Zustand *Aktive Umgebungserkundung* ist das Gegenstück zur *Passive Umgebungserkundung*, wenn ein Spieler sich im Spielzustand *Aktiv* befindet. Der Startpunkt wird nicht mehr dargestellt. Dafür erhält der Spieler Informationen über sich in der Nähe befindliche POIs. Aus diesem Zustand kann ein Spieler auch die folgenden Interaktionszustände erreichen.

- **Charakterübersicht**: Die Charakterübersicht vermittelt einem Spieler Informationen über seinen Spielcharakter, z.B. wie viele Fähigkeitspunkte der Spieler welcher Fähigkeit zugewiesen hat.

- **Nachrichtenübersicht**: In dem Interaktionszustand *Nachrichtenübersicht* stellt die Anwendung einem Spieler eine Übersicht der bereits empfangenen Nachrichten zur Verfügung. Aus diesem Zustand kann der Spieler in zwei weitere Interaktionszustände wechseln:

  - **Nachricht verfassen**: Ein Spieler kann neue Nachrichten verfassen und diese an Mitspieler aus seinem eigenen Team senden.
  - **Nachricht abrufen**: Im Zustand *Nachricht abrufen* ruft ein Spieler eine Nachricht aus der Übersicht ab, um deren Inhalt zu lesen.

- **Missionsziel lösen**: In diesem Zustand befindet sich der *Commander*, wenn er sich in der Nähe eines Missionsziels befindet. Andere Spieler können sich nicht in diesem Zustand befinden.

13.2.3.1 Nachrichten

Jeder Spieler kann, sofern er aktiv ist, Nachrichten schreiben, um seinen Teammitgliedern Informationen zukommen zu lassen. Diese Nachrichten können sowohl an einzelne Spieler des eigenen Teams als auch an das komplette eigene Team gesendet werden.

Ungelesene Post Der Zustandsautomat in Abb. 13.3 zeigt an, ob ungelesene Nachrichten für den Spieler vorhanden sind.

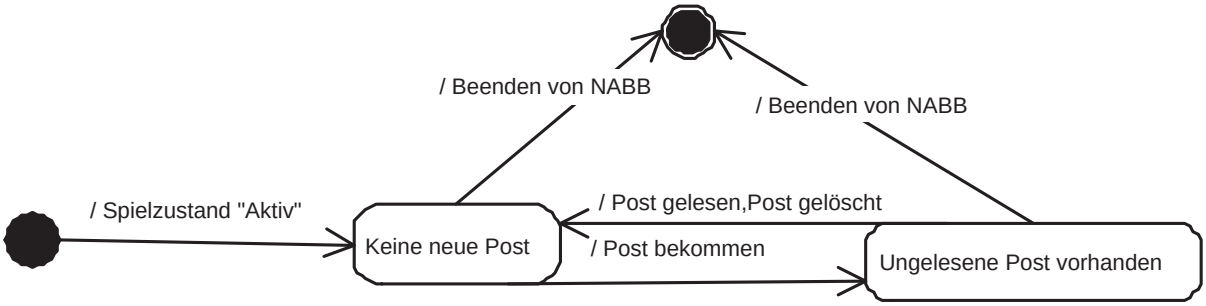


Abbildung 13.3: Ungelesene Post

- **Keine neue Post**: Der Spieler hat alle Nachrichten gelesen.
- **Ungelesene Post vorhanden**: Der Spieler hat eine oder mehrere Nachrichten empfangen und noch nicht gelesen.

13.2.3.2 Points-of-Interest

In diesem Abschnitt werden Zustände im Bezug auf die Nähe eines Spielers zu bestimmten Points-of-Interests (POIs) beschrieben. Befindet sich ein Spieler in der Nähe eines POI, so kann dies den Einsatz neuer Funktionen ermöglichen. Für die einzelnen POIs *Mitspieler*, *gegnerischer Spieler*, *Missionsziel*, *Goodie* und *gegnerische Kamera* wurden jeweils unabhängig voneinander zwei Zustände definiert. Diese geben an, ob sich der Spieler in der für das Spiel relevanten Nähe eines POIs befindet. Die Zustände gehen jeweils von *POI nah* zu *kein POI* über und umgekehrt.

Mitspieler Wie bereits beschrieben, wird die Nähe zu Mitspielern über zwei Zustände angegeben (Abb. 13.4). Im Zustand *Mitspieler nah* befindet sich ein Mitspieler in der näheren Umgebung. Eine neue Funktionalität wie z.B. eine direkte Kommunikation über Bluetooth zwischen den beiden Mitspielern wäre möglich. Die Verfügbarkeit einer solchen besonderen Funktion wird dem Spieler mitgeteilt. Ist ein Spieler weiter von seinen Mitspielern entfernt, so befindet er sich im Zustand *kein Mitspieler*.

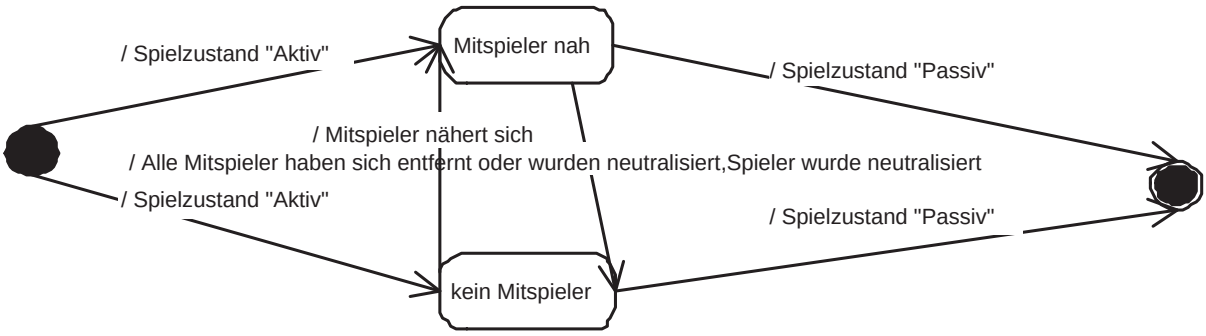


Abbildung 13.4: Nähe zu Mitspieler

Gegnerischer Spieler Befindet sich ein gegnerischer Spieler in der Nähe, so befindet sich der Spieler im Zustand *gegnerischer Spieler nah* (Abb. 13.5). In diesem Fall wird der Spieler darüber informiert, um sich auf die besondere Situation einstellen zu können. Diese Situation birgt beispielsweise die Möglichkeit den Gegner zu neutralisieren, aber auch die Gefahr, selbst neutralisiert oder zumindest entdeckt zu werden.

Ist kein gegnerischer Spieler in der Nähe, so ist keine erhöhte Aufmerksamkeit erforderlich. Der Spieler befindet sich dann im Zustand *kein gegnerischer Spieler*.

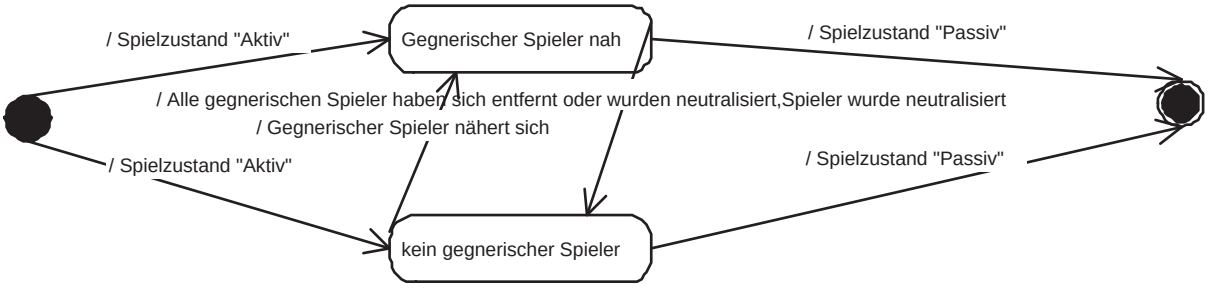


Abbildung 13.5: Nähe zu gegnerischem Spieler

Missionsziel Die Hauptaufgabe des *Commanders* liegt darin, Missionen zu erfüllen. Diese Missionen können z.B. aus dem virtuellen Diebstahl von Daten oder aus dem „Knacken“ eines virtuellen Safes bestehen. Wenn der *Commander* den Ort der Mission erreicht hat, wird ihm eine Aufgabe gestellt, für die er drei Lösungsversuche hat. Schafft er dies nicht, so gilt die Mission als nicht erfüllt und das Team der *Defender* hat das Spiel gewonnen. Wenn ein *Infiltrator* ein Goodie aufgenommen hat, so kann dieses Hinweise zur Lösung des Missionsziels beinhalten.

In der Nähe von Missionszielen bzw. im Zustand *Missionsziel nah* (Abb. 13.6) kann der Commander versuchen, das Missionsziel zu lösen. Dabei wirken sich die Missionsziele nur auf den Commander selbst aus, d.h. ein anderer Spieler außer dem Commander kann sich nicht im Zustand *Missionsziel nah* befinden. Im Zustand *Missionsziel fern* ist keine Interaktion mit dem Missionsziel möglich.

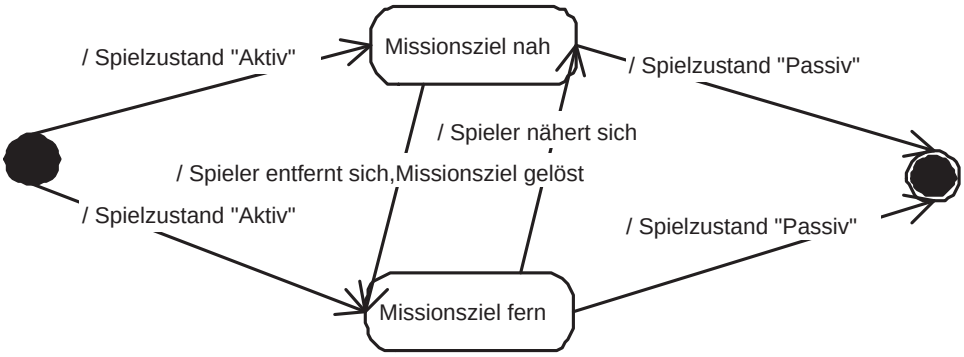


Abbildung 13.6: Missionsziel

Goodie Ein weiterer Teil des Spiels sind spezielle Objekte, so genannte Goodies, die sich auf dem Spielfeld befinden und von den Spielern aufgespürt werden müssen. Diese können kleine Bonifikationen in Form von temporären Upgrades auf bestimmte Fähigkeiten oder Informationen für einen Spieler enthalten, die zur Erfüllung der Missionsziele dienen. Um einen solchen Bonus erhalten zu können, muss der Spieler zunächst ein Goodie einsammeln. Dazu muss er sich in die Nähe des auf der Karte angezeigten Goodies bewegen. Erreicht ein Spieler ein Goodie, so wechselt er in den Zustand *Goodie nah*. Nur in diesem Zustand kann das Goodie eingesammelt werden. Die Verfügbarkeit dieser Funktion wird einem Spieler beim Wechsel in diesen Zustand mitgeteilt. Verlässt ein Spieler die Nähe des Goodies, etwa durch

Einsammeln und der dadurch bedingten Entfernung des Goodies vom Spielfeld, so wechselt er in den Zustand *kein Goodie*.

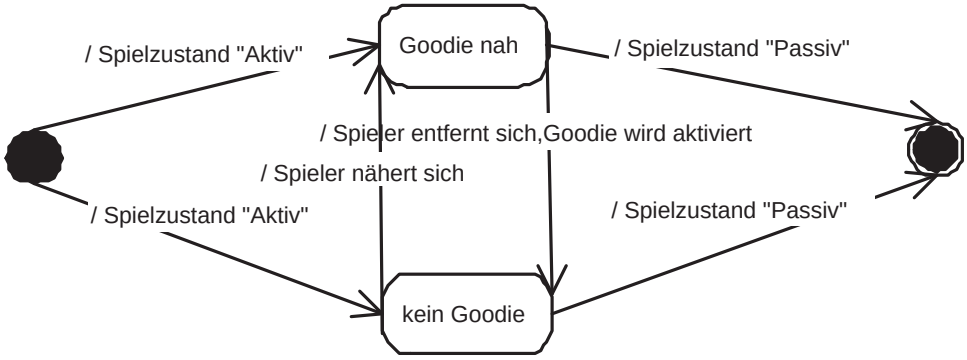


Abbildung 13.7: Goodie

13.2.3.3 Fähigkeiten

Jeder Spieler kann im Spiel die 4 Grundfähigkeiten *Abhören*, *Aufklären*, *Neutralisieren* und *Tarnen* ausführen. Die Funktionsweise und Wirkung der jeweiligen Fähigkeit wird im Folgenden beschrieben.

Aufklären Der Zustandsautomat in Abb. 13.8 zeigt die möglichen Zustände der Funktion *Aufklären* an. Diese Fähigkeit erlaubt das Setzen von Aufklärungskameras auf dem Spielfeld, wobei eine Kamera an der momentanen Position des Spielers abgelegt wird. Jede Kamera hat ihrerseits einen Sichtradius, in dem alle POIs sichtbar werden. So können selbst getarnte Spieler aufgespürt werden, sobald diese sich in den Sichtradius einer Aufklärungskamera bewegen. Der Sichtradius, sowie die Anzahl der zur Verfügung stehenden Kameras hängen von den vor Spielbeginn verteilten Fähigkeitspunkten auf diese Funktion ab. Bei einem Fähigkeitspunkt beträgt der Radius 20 Meter, bei 2 bzw. 3 Fähigkeitspunkten je 24 Meter und bei 4 Fähigkeitspunkten 28 Meter. Die Anzahl der zur Verfügung stehenden Kameras, entspricht den Fähigkeitspunkten der Funktion *Aufklären*.
Platzierte Kameras können jederzeit vom Spieler wieder eingesammelt werden, indem er sich zu deren Position begibt und auf das Kamerasymbol klickt. Gegnerische Kameras können auf diese Weise deaktiviert werden.

- **Aufklären nicht verfügbar** (initial): Ein Spieler wechselt in den Zustand *Aufklären nicht verfügbar*, wenn er dem Spiel beitritt oder neutralisiert wurde. Die Funktion *Aufklären* kann in diesem Zustand nicht eingesetzt werden. Nimmt ein Spieler wieder aktiv am Spielgeschehen teil, kann er direkt in den Zustand *Aufklären bereit* wechseln, wenn er noch Kameras vorrätig hat. Hat er bereits alle Kameras gesetzt, wechselt er aus diesem Zustand in den Zustand *Keine Kameras vorrätig*.
- **Keine Kameras vorrätig**: In dem Zustand *Keine Kameras vorrätig* befindet sich ein Spieler, wenn er alle seine Aufklärungskameras auf dem Spielfeld positioniert hat. Aus diesem Zustand kann in den Zustand **Aufklären bereit** gewechselt werden, wenn der Spieler eine seiner Kameras aufgenommen hat oder eine seiner Kameras von einem Gegner deaktiviert wurde.
- **Aufklären bereit**: Wenn ein Spieler Kameras zur Verfügung hat, befindet er sich im Zustand **Aufklären bereit**. Hat der Spieler seine letzte Kamera gesetzt, wechselt er in den Zustand **Keine Kameras vorrätig**.

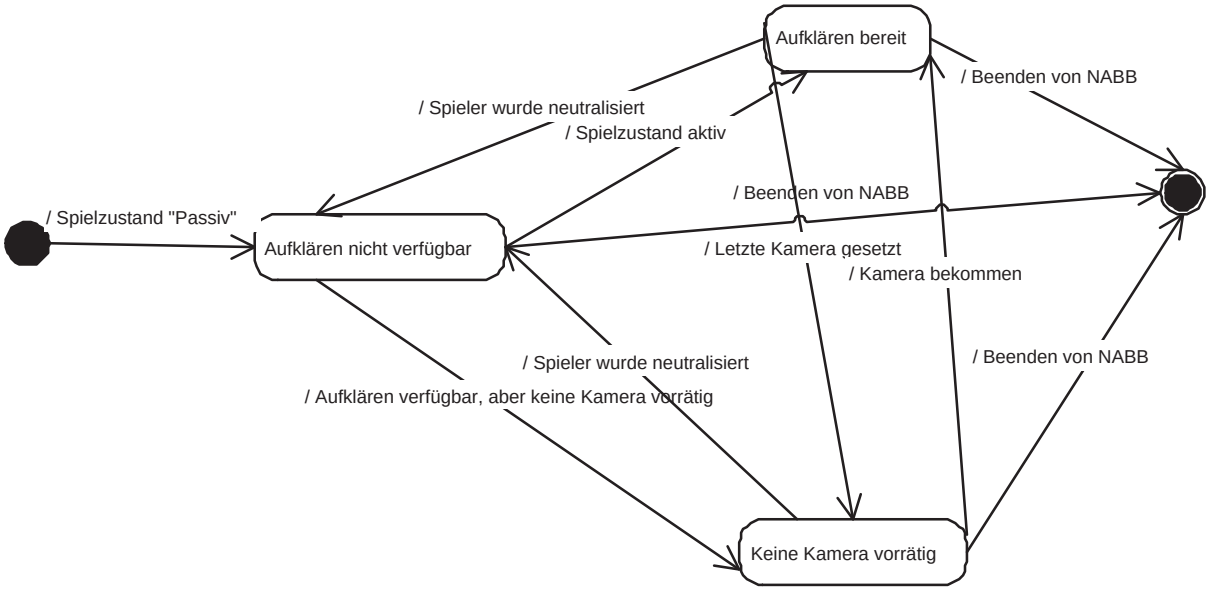


Abbildung 13.8: Aufklären

Neutralisieren Der Zustandsautomat in Abb. 13.9 zeigt die möglichen Zustände der Funktion *Neutralisieren* an. *Neutralisieren* kann in einem bestimmten Radius um den aktuellen Standort des Spielers eingesetzt werden. Dieser Radius beträgt bei einem Fähigkeitspunkt 20 Meter, bei 2 bzw. 3 Fähigkeitspunkten je 24 Meter und bei 4 Fähigkeitspunkten 28 Meter. Die Dauer des Neutralisierungsvorgangs beträgt 5 Sekunden. Alle Spieler, die sich nach diesen 5 Sekunden innerhalb des Neutralisierungsradius befinden, werden neutralisiert. Einem Versuch der Neutralisation kann man sich nur entziehen, indem man sich vor Ablauf der Neutralisationszeit aus dem Neutralisationsradius entfernt.

- **Neutralisieren nicht verfügbar** (initial): Ein Spieler wechselt in den Zustand *Neutralisieren nicht verfügbar*, wenn er dem Spiel beitritt oder neutralisiert wurde. Aus diesem Zustand kann der Spieler lediglich in den Zustand *Neutralisieren lädt auf* wechseln, nachdem er seine Fähigkeiten aktiviert hat.
- **Neutralisieren lädt auf**: In dem Zustand *Neutralisieren lädt auf* befindet sich ein Spieler, wenn der Spielzustand einen Neutralisationsvorgang erlauben würde, die Vorbereitungszeit (2 Minuten) für den Einsatz der Funktion aber noch nicht abgelaufen ist. Nach Ablauf der Vorbereitungszeit wechselt er in den Zustand *Neutralisieren bereit*. Wird ein Spieler in diesem Zustand neutralisiert, wechselt er wieder in den Initialzustand *Neutralisieren nicht verfügbar*.
- **Neutralisieren bereit**: Ein Spieler wechselt in den Zustand *Neutralisieren bereit*, sobald ihm die Funktion *Neutralisieren* zur Verfügung steht. Versucht ein Spieler einen Gegner zu neutralisieren, wechselt er in den Zustand *Neutralisieren aktiviert*. Wird ein Spieler neutralisiert, wechselt er in den Zustand *Neutralisieren nicht verfügbar*.
- **Neutralisieren aktiviert**: In den Zustand *Neutralisieren aktiviert* wechselt ein Spieler, wenn er die Funktion *Neutralisieren* eingesetzt hat. Dieser Zustand wird verlassen, wenn die Dauer des Neutralisationsvorgangs abgelaufen ist oder er von einem Gegner neutralisiert wurde.

Tarnen Der Zustandsautomat in Abb. 13.10 zeigt die möglichen Zustände der Funktion *Tarnen* an. Ein Spieler kann sich während des Spielverlaufs tarnen und ist dann für seine Mitspieler und Gegner nur noch

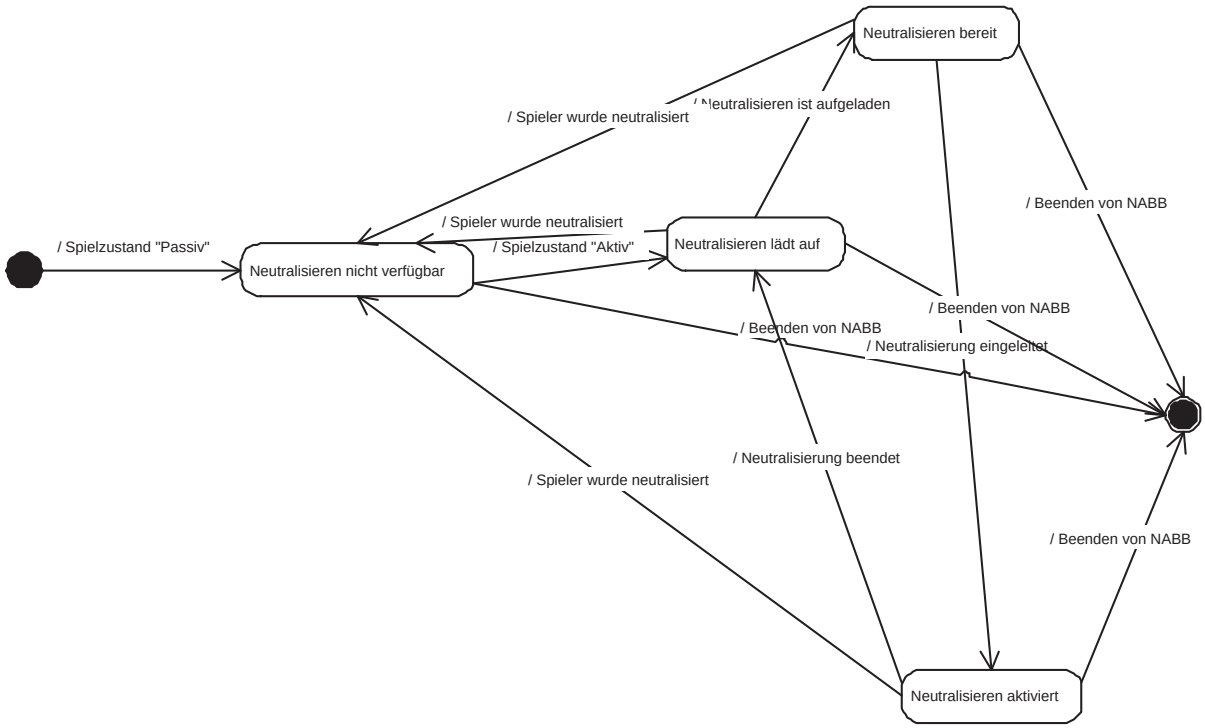


Abbildung 13.9: Neutralisieren

auf der Übersichtskarte sichtbar, wenn er sich im Sichtradius einer Aufklärungskamera befindet. Er kann allerdings auf seiner Karte auch keine *Points Of Interest* mehr sehen. Darüber hinaus ist der getarnte Spieler Der Tarnmodus steht, in Abhängigkeit der zu Spielbeginn auf die Fähigkeit *Tarnen* verteilten Punkte, für eine bestimmte Zeit zur Verfügung. Bei einem Fähigkeitspunkt kann sich ein Spieler für 50 Sekunden tarnen, bei 2 Punkten für 70, bei 3 Punkten für 80 und bei 4 verteilten Fähigkeitspunkten für 100 Sekunden.

- **Tarnen nicht verfügbar** (initial): Ein Spieler wechselt in den Zustand *Tarnen nicht verfügbar*, wenn er dem Spiel beitritt oder neutralisiert wurde. Aus diesem Zustand kann der Spieler lediglich in den Zustand *Tarnen lädt auf* wechseln, nachdem er seine Fähigkeiten aktiviert hat.

- **Tarnen lädt auf**: In dem Zustand *Tarnen lädt auf* befindet sich ein Spieler, wenn der Spielzustand es erlaubt, dass der Spieler sich tarnen kann, die Vorbereitungszeit (2 Minuten) für den Einsatz der Funktion aber noch nicht abgelaufen ist. Nach Ablauf der Vorbereitungszeit wechselt er in den Zustand *Tarnen bereit*. Wird ein Spieler in diesem Zustand neutralisiert, wechselt er wieder in den Initialzustand *Tarnen nicht verfügbar*.

- **Tarnen bereit**: Ein Spieler wechselt in den Zustand *Tarnen bereit*, sobald ihm die Funktion *Tarnen* zur Verfügung steht. Aktiviert er die Tarnfunktion, wechselt er in den Zustand *Getarnt*; wird er neutralisiert, in den Zustand *Tarnen nicht verfügbar*.

- **Getarnt**: In den Zustand *Getarnt* wechselt ein Spieler, der die Funktion *Tarnen* eingesetzt hat. Diesen Zustand verlässt er erst, wenn er die Funktion deaktiviert hat, die Zeit, in der er sich tarnen kann abgelaufen ist, oder er von einem Gegner neutralisiert wurde.

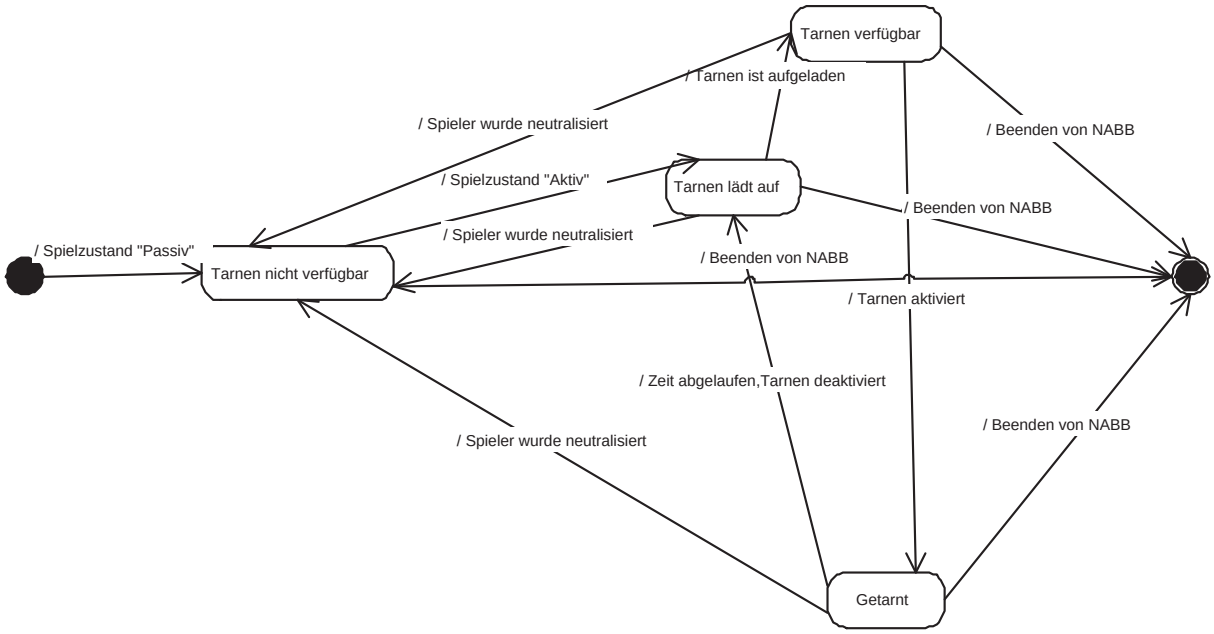


Abbildung 13.10: Tarnen

Abhören Der Zustandsautomat in Abb. 13.11 zeigt die möglichen Zustände der Funktion *Abhören* an. Wenn die Funktion *Abhören* aktiviert wird, können Nachrichten des gegnerischen Teams abgehört werden. Der Abhörmodus steht, in Abhängigkeit der zu Spielbeginn auf die Fähigkeit *Abhören* verteilten Punkte, für eine bestimmte Zeit zur Verfügung. Bei einem Fähigkeitspunkt kann ein Spieler für 50 Sekunden abhören, bei 2 Punkten für 70, bei 3 Punkten für 80 und bei 4 verteilten Fähigkeitspunkten für 100 Sekunden.

- **Abhören nicht verfügbar** (initial): Ein Spieler wechselt in den Zustand *Abhören nicht verfügbar*, wenn er dem Spiel beitritt oder neutralisiert wurde. Aus diesem Zustand kann ein Spieler in den Zustand *Abhören lädt auf* wechseln, nachdem er seine Fähigkeiten aktiviert hat.

- **Abhören lädt auf**: In dem Zustand *Abhören lädt auf* befindet sich ein Spieler, wenn der Spielzustand es erlaubt, dass der Spieler Nachrichten abhören kann, die Vorbereitungszeit (2 Minuten) für den Einsatz der Funktion aber noch nicht abgelaufen ist. Nach Ablauf der Vorbereitungszeit wechselt er in den Zustand *Abhören bereit*. Wird ein Spieler in diesem Zustand neutralisiert, wechselt er wieder in den Initialzustand *Abhören nicht verfügbar*.

- **Abhören bereit**: Ein Spieler wechselt in den Zustand *Abhören bereit*, sobald die Funktion *Abhören* zur Verfügung steht. Aktiviert er die Funktion *Abhören*, wechselt er in den Zustand *Abhören aktiviert*, wird er neutralisiert in den Zustand *Abhören nicht verfügbar*.

- **Abhören aktiviert**: In diesen Zustand wechselt ein Spieler, der die Funktion *Abhören* eingesetzt hat. Diesen Zustand verlässt er erst, wenn er die Funktion deaktiviert hat, die Zeit, in der er Nachrichten abhören kann, abgelaufen ist, oder er von einem Gegner neutralisiert wurde.

Gegnerische Kamera Bei der Bewegung über das Spielfeld kann es passieren, dass ein Spieler in die Nähe einer gegnerischen Aufklärungskamera gerät. Befindet er sich im Sichtradius einer solchen Kamera, so wird er für den Besitzer der Kamera sichtbar. Um dem Risiko entdeckt zu werden entgehen zu können,

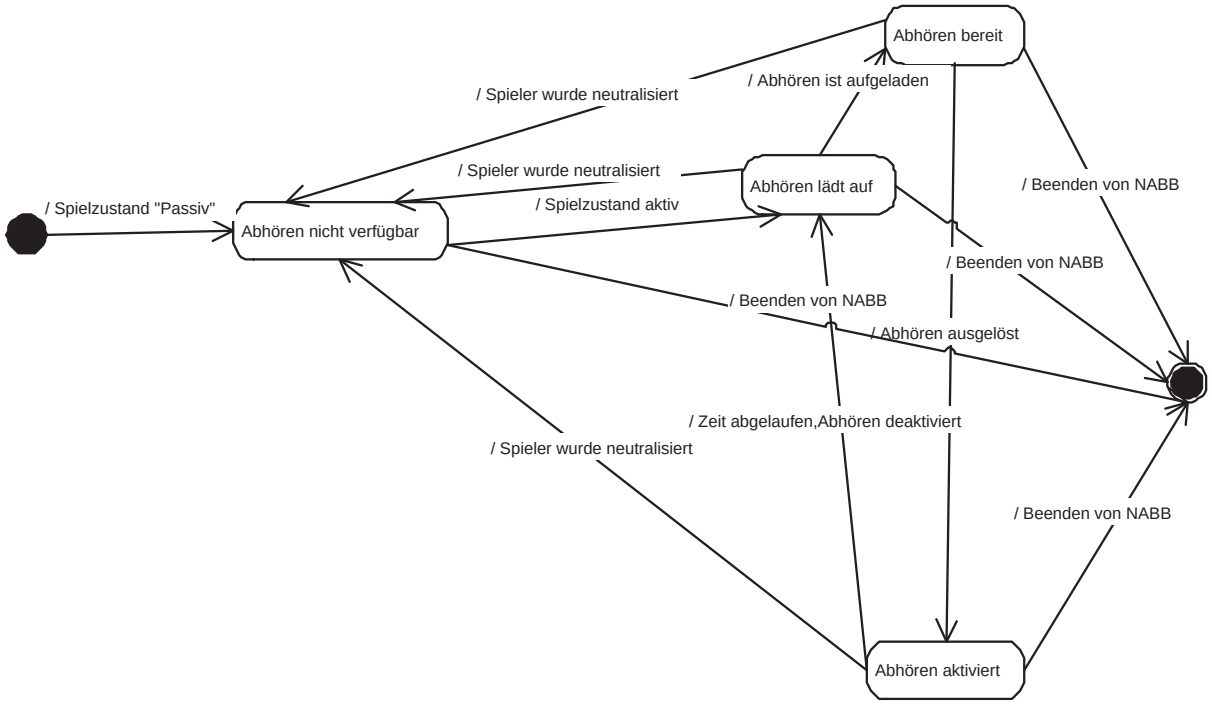


Abbildung 13.11: Abhören

wird ein Spieler über die Anwesenheit einer gegnerischen Kamera informiert. Er befindet sich dann im Zustand *gegnerische Kamera nah* (Abb. 13.12). Ist keine gegnerische Kamera in der Nähe, ist ein Spieler im Zustand *keine gegnerische Kamera*. Eine gesetzte Kamera kann vom gegnerischen Team, wie oben beschrieben, deaktiviert werden.

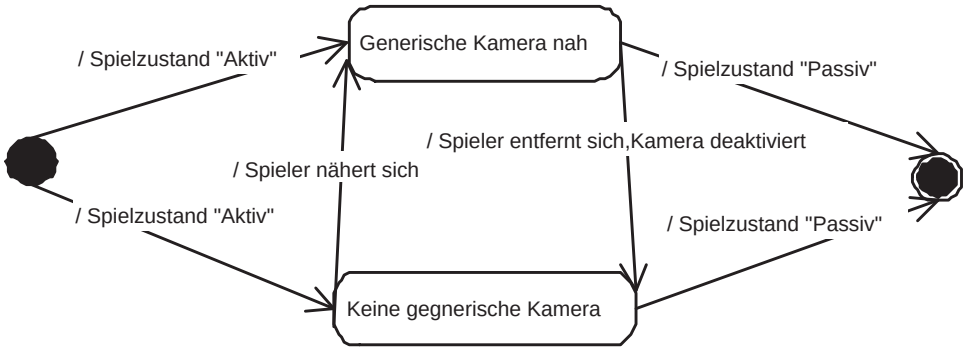


Abbildung 13.12: Gegnerische Kamera

13.2.3.4 Spielende

Das Spielende ist erreicht, wenn der *Commander* neutralisiert wurde oder die Lösung zu einem Missionsziels vom *Commander* dreimal falsch eingegeben wurde. In diesen Fällen hat das Team der *Defender* gewonnen. Wenn alle Missionsziele erfüllt wurden ist das Team der *Infiltratoren* der Sieger. Der zugehörige Zustandsautomat ist in Abb. 13.1 auf S.247 zu sehen.

13.2.4 Datenmodell

Das Datenmodell stellt gespeicherte Daten, ihre innere Struktur und ihre Beziehungen untereinander dar. Die Beschreibung der Daten erfolgt durch eine formale Darstellung in Form von Diagrammen und zusätzlicher textueller Beschreibung.
In den folgenden Diagrammen wird die Datenstruktur des Agentenspiels beschrieben. Um die Übersichtlichkeit zu gewährleisten, ist diese formale Darstellung in mehrere Diagramme aufgeteilt.

13.2.4.1 Das Szenario

Grundlage eines jeden NABB-Spiels ist das Szenario. Durch dieses wird die Umgebung, in der gespielt wird, sowie die Koordinaten der statischen Points of Interest bestimmt. Das Laden des Szenarios geschieht aus einer XML-Datei, die einem bestimmten Schema folgen muss, um vom Spiel erkannt zu werden.

Im Folgenden soll dieses XML-Schema erklärt werden und es soll dargestellt werden, auf welche Art die Daten aus der XML-Szenariodatei im Spiel verwendet werden.

XML-Schema Zur Definition des Aufbaus einer NABB-Szenariodatei wird XML-Schema verwendet, da dieses unter anderem, im Gegensatz zu DTDs, die Definition von standardisierten, atomaren Datentypen für Elemente und Attribute ermöglicht und außerdem komplexe Typen für Elementinhalte unterstützt.

Jedes Szenario enthält genau ein *Map*-Tag, welches die Karte repräsentiert, auf der das Spiel stattfindet. In den *StartingPoints* werden die Startpunkte der *Defender*, der *Infiltratoren* sowie ein Default-Startpunkt gespeichert.
Danach folgt mindestens ein Missionsziel (*missionTarget*) innerhalb der *missionTargets*-Umgebung. Es können jedoch innerhalb eines Szenarios auch mehrere Missionsziele existieren.
Ebenso können innerhalb des Szenarios mehrere Goodies vorhanden sein, deren Existenz wird jedoch innerhalb des XML-Schemas nicht vorausgesetzt.
Innerhalb des Schema wird zusätzlich eine *systemConfiguration* verlangt. Diese dient dazu, den Aufklärungsradius und den Neutralisationsradius in Abhängigkeit von der Größe der Karte modular zu bestimmen.
Jedes der eben beschriebenen Tags weist seinerseits noch Attribute auf, die im Zusammenhang der Abbildung 13.13 auf S.259 näher beschrieben werden.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xs:schema targetNamespace="NABB" elementFormDefault="qualified" xmlns:xs="http://www.w3.
       org/2001/XMLSchema" xmlns="NABB">
3    <xs:element name="scenario">
4      <xs:complexType>
5        <xs:sequence>
6          <xs:element name="map">
7            <xs:complexType>
8              <xs:sequence>
9                <xs:element name="filePath" type="xs:string"/>
10               <xs:element name="upperLeftPoint" type="position"/>
11               <xs:element name="lowerRightPoint" type="position"/>
12               <xs:element name="zoomedMap" type="xs:string" minOccurs="0" maxOccurs="
                     unbounded"/>
13             </xs:sequence>
14           </xs:complexType>
```

```
15        </xs:element>
16        <xs:element name="startingPoints">
17          <xs:complexType>
18            <xs:sequence>
19              <xs:element name="defaultStartPoint" type="position"/>
20              <xs:element name="defenderStartPoint" type="position"/>
21              <xs:element name="infiltratorStartPoint" type="position"/>
22            </xs:sequence>
23          </xs:complexType>
24        </xs:element>
25        <xs:element name="missionTargets">
26          <xs:complexType>
27            <xs:sequence>
28              <xs:element name="missionTarget" maxOccurs="unbounded">
29                <xs:complexType>
30                  <xs:complexContent>
31                    <xs:extension base="importantAreaWithTask">
32                      <xs:attribute name="missionId" type="xs:positiveInteger" use="
                            required"/>
33                      <xs:attribute name="type" type="MissionTargetTypes" use="required"
                            />
34                    </xs:extension>
35                  </xs:complexContent>
36                </xs:complexType>
37              </xs:element>
38            </xs:sequence>
39          </xs:complexType>
40        </xs:element>
41        <xs:element name="goodies">
42          <xs:complexType>
43            <xs:sequence>
44              <xs:element name="goodie" maxOccurs="unbounded">
45                <xs:complexType>
46                  <xs:complexContent>
47                    <xs:extension base="importantArea">
48                      <xs:attribute name="goodieId" type="xs:positiveInteger" use="
                            required"/>
49                      <xs:attribute name="type" type="GoodieTypes" use="required"/>
50                    </xs:extension>
51                  </xs:complexContent>
52                </xs:complexType>
53              </xs:element>
54            </xs:sequence>
55          </xs:complexType>
56        </xs:element>
57        <xs:element name="systemConfiguration">
58          <xs:complexType>
59            <xs:sequence>
60              <xs:element name="visionRadiusFactor" type="xs:decimal"/>
61            </xs:sequence>
62          </xs:complexType>
63        </xs:element>
64      </xs:sequence>
65      <xs:attribute name="name" type="xs:string" use="required"/>
66    </xs:complexType>
67    <!-- ********************************keys****************************** -->
68    <xs:key name="missionId">
69      <xs:selector xpath=".//missionTarget"/>
```

```
70      <xs:field xpath="@missionId"/>
71    </xs:key>
72    <xs:key name="goodieId">
73      <xs:selector xpath=".//goodie"/>
74      <xs:field xpath="@goodieId"/>
75    </xs:key>
76    <xs:key name="taskId">
77      <xs:selector xpath=".//task"/>
78      <xs:field xpath="@taskId"/>
79    </xs:key>
80    <!-- ******************************key refs****************************  -->
81    <xs:keyref name="refMissionTargetId" refer="missionId">
82      <xs:selector xpath=".//alarmSystem"/>
83      <xs:field xpath="@refMissionTargetId"/>
84    </xs:keyref>
85  </xs:element>
86  <!-- ****************************complexType***************************** -->
87  <xs:complexType name="importantArea">
88    <xs:sequence>
89      <xs:element name="position" type="positionImpl"/>
90      <xs:element name="visibleRadius" type="xs:decimal"/>
91      <xs:element name="description" type="xs:string" default="No␣description"/>
92    </xs:sequence>
93  </xs:complexType>
94  <xs:complexType name="importantAreaWithTask">
95    <xs:complexContent>
96      <xs:extension base="importantArea">
97        <xs:sequence>
98          <xs:element name="task">
99            <xs:complexType>
100             <xs:attribute name="taskId" type="xs:positiveInteger" use="required"/>
101             <xs:attribute name="type" type="TaskTypes" use="required"/>
102           </xs:complexType>
103         </xs:element>
104       </xs:sequence>
105     </xs:extension>
106   </xs:complexContent>
107 </xs:complexType>
108 <xs:complexType name="position">
109   <xs:sequence>
110     <xs:element name="position" type="positionImpl"/>
111   </xs:sequence>
112 </xs:complexType>
113 <xs:complexType name="positionImpl">
114   <xs:sequence>
115     <xs:element name="latitude" type="xs:decimal"/>
116     <xs:element name="longitude" type="xs:decimal"/>
117     <xs:element name="altitude" type="xs:decimal"/>
118   </xs:sequence>
119 </xs:complexType>
120 <!-- -****************************simpleType****************************** -->
121 <xs:simpleType name="MissionTargetTypes">
122   <xs:restriction base="xs:string">
123     <xs:whiteSpace value="collapse"/>
124     <xs:enumeration value="DataTheftMission"/>
125   </xs:restriction>
126 </xs:simpleType>
127 <xs:simpleType name="TaskTypes">
```

```
128     <xs:restriction base="xs:string">
129       <xs:whiteSpace value="collapse"/>
130       <xs:enumeration value="BinaryCode"/>
131     </xs:restriction>
132   </xs:simpleType>
133   <xs:simpleType name="GoodieTypes">
134     <xs:restriction base="xs:string">
135       <xs:whiteSpace value="collapse"/>
136       <xs:enumeration value="1"/>
137     </xs:restriction>
138   </xs:simpleType>
139 </xs:schema>
```

Listing 13.1: XML-Schema zur Szenariodatei

Im folgenden Diagramm sind die Klassen abgebildet, die dafür sorgen, dass die Daten aus der XML-Szenariodatei ins Spiel einfließen.

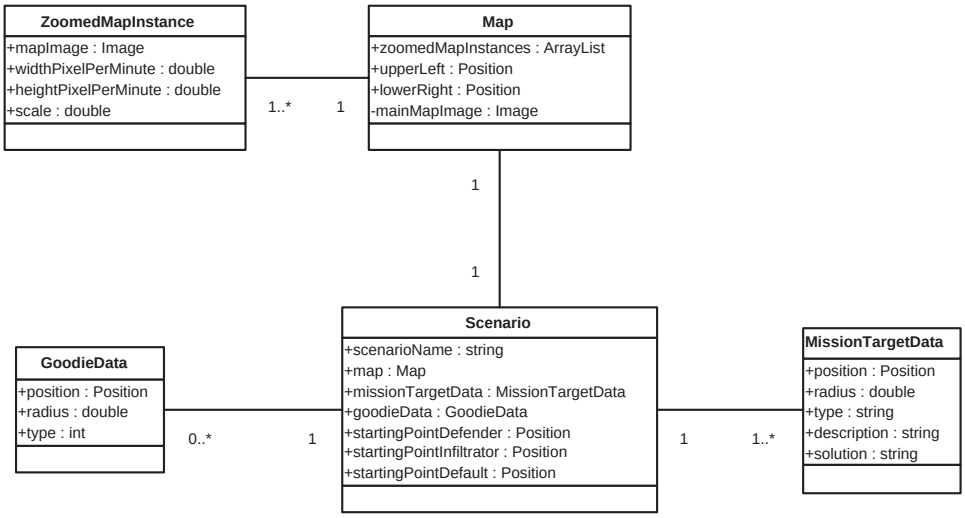


Abbildung 13.13: Das Szenario und seine assoziierten Klassen

Scenario Die Klasse *Scenario* sorgt für das Auslesen der Daten aus der XML Datei und die Speicherung dieser Daten innerhalb des Spiels. Dabei werden die einzelnen Daten entweder in Variablen gespeichert (z.B. der Szenarioname) oder es werden Objekte instanziiert, die wiederum die notwendigen Daten enthalten (z.B. das Map-Objekt, GoodieData, MissionTargetData und die verschiedenen Startpunkte).

Map In dieser Klasse werden die Informationen zur Karte des Szenarios gespeichert. Diese umfassen zum einen die Bilddatei der Karte und die GPS-Positionen der linken oberen und der rechten unteren Ecke der Karte und zum anderen eine Liste von verfügbaren Zoomstufen für die Karte.

ZoomedMapInstance Diese Klasse speichert alle notwendigen Daten zu einer Zoomstufe einer Karte. Dazu gehören die Bilddatei der Karte, der Zoomfaktor, das Verhältnis von Höhenpixeln zu GPS-Minuten sowie das Verhältnis von Breitenpixeln zu GPS-Minuten der Zoomstufe.

GoodieData Die Klasse *GoodieData* speichert die Informationen zu einem Goodie. Diese umfassen die Position des Goodies, seinen Typ sowie den Aktivierungsradius innerhalb dessen man mit dem Goodie interagieren kann.

MissionTargetData In dieser Klasse werden die notwendigen Informationen zu einem Missionsziel gespeichert. Dazu gehören die Position des Missionsziels, sein Typ, der Interaktionsradius, eine Beschreibung des Missionsziels sowie der Text, der zum Lösen des Missionsziels angegeben werden muss.

13.2.4.2 Die Spielobjekte

Im folgenden Diagramm sind die auf dem Spielfeld sichtbaren Spielobjekte, sowie die damit verbundenen Klassen *Position* und *QualifiedPosition* dargestellt. Außerdem sind die Klassen *UserProfile* und *ClientState* dargestellt, die zur Speicherung des aktuellen Status eines Spielers dienen.

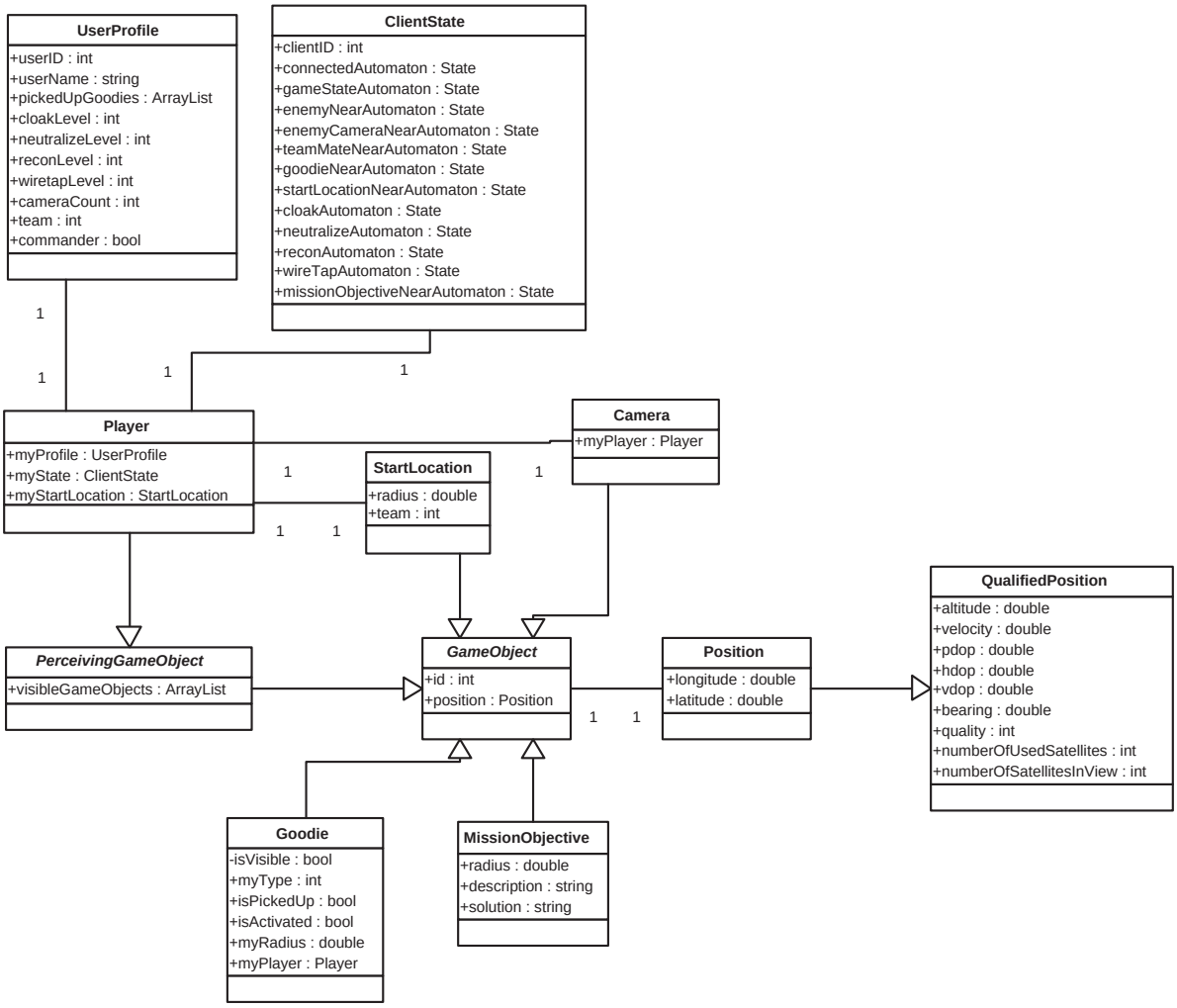


Abbildung 13.14: Übersicht über die Spielobjekte und assoziierte Klassen

GameObject Die Klasse *GameObject* bildet die abstrakte Grundlage für alle Spielobjekte. Sie speichert die Attribute, die allen Spielobjekten gemein sind: eine eindeutige ID und eine Position.

Camera Diese Klasse repräsentiert eine Aufklärungskamera, welche von den Spielern gesetzt werden kann. Zusätzlich zu den von der Klasse *GameObject* geerbten Attributen wird der Spieler, der die Kamera gesetzt hat, gespeichert.

Goodie In der Klasse *Goodie* werden der Typ des Goodies und sein Interaktionsradius gespeichert. Außerdem wird vorgehalten, ob das Goodie gerade sichtbar ist, ob es bereits aktiviert wurde und ob es von einem Spieler aufgenommen wurde. Ist letzteres der Fall, so wird auch der zugehörige Spieler gespeichert.

StartLocation Diese Klasse repräsentiert den Startpunkt eines Spielers, den er aufsuchen muss, um aktiv am Spielgeschehen teilnehmen zu können. Sie speichert den Interaktionsradius des Startpunkts und die Information, zu welchem Team dieser Startpunkt gehört.

PerceivingGameObject Ein *PerceivingGameObject* hat die Fähigkeit, andere GameObjects wahrzunehmen und speichert deswegen eine Liste der im Moment sichtbaren *GameObjects*

Player Die Klasse *Player* repräsentiert einen Spieler des Agentenspiels. Hier werden zusätzlich zu den von der Klasse *PerceivingGameObject* geerbten Attributen, die Startposition des Spielers, sein Profil sowie sein *ClientState* gespeichert.

UserProfile Diese Klasse stellt ein Nutzerprofil eines Spielers dar und ist damit eng mit der Klasse *Player* verknüpft. Hier werden die Client-Identifikation, der Spielername, die Teamzugehörigkeit des Spielers und seine Rolle (Defender, einfacher Infiltrator oder Commander) gespeichert. Außerdem bietet die Klasse Informationen darüber, welches Level der Spieler in den jeweiligen Fähigkeiten besitzt, wieviele Aufklärungskameras er noch setzen kann und welche Goodies er gerade aufgenommen hat.

ClientState In der Klasse *ClientState* wird die Client-Identifikation sowie die aktuellen Zustände der in Kapitel 13.2.3 beschriebenen Automaten gespeichert.

Position Die Klasse *Position* repräsentiert eine Position auf dem Erdball. Diese wird durch die geographische Breite (latitude) und die geographische Länge (longitude) bestimmt.

QualifiedPosition Diese Klasse stellt eine Erweiterung zur normalen *Position* dar. Sie speichert zusätzlich noch die Höhenangabe der Position, die Geschwindigkeit und Bewegungsrichtung des Spielers und einige Werte zur Genauigkeit der Positionsbestimmung.

13.3 Erfassung und Modellierung von Kontextinformationen

Es existieren diverse Modelle für die Erfassung und Modellierung von Kontextinformationen (siehe Seminararbeit *Kontext & User Modeling* auf Seite 104). Dabei wird der Kontext einer Person häufig in unterschiedliche Kategorien unterteilt, wie von Bradley und Dunlop [BD03a]. Es wird zwischen der Welt des Benutzers und der Welt der Anwendung, einer kontextuellen Ebene und einer fokalen Ebene und zwischen bedeutenden und nebensächlichen Kontextinformationen unterschieden. Die kontextuelle Ebene beinhaltet alle äußeren Umstände, die den Benutzer oder die Anwendung bei der Durchführung ihrer fokalen Aktivitäten, also der Aktivitäten, auf die sich Benutzer bzw. Anwendung momentan konzentrieren,

beeinflussen. Sie ist in vier Dimensionen unterteilt. Dazu gehören hier zeitlicher, sozialer und physischer Kontext, sowie Kontext im Bezug auf die aktuelle Aufgabe. Im Hinblick auf die Ausführung der momentanen Aktionen können manche Kontextinformationen von Bedeutung sein, andere aber nicht. Insofern beinhaltet dieses Modell implizit die Kontextkomponente Zeit, da die Aktionen im Fokus von Nutzer und Anwendung über die Zeit variieren und somit auch die unterschiedlichen Kontextinformationen in ihrer Relevanz.

Die kontextuelle Ebene Den Rahmen für den Spielverlauf setzen einzelnen Aufgaben, die von den Spielern der jeweiligen Teams kooperativ und in Konkurrenz zu Spielern des anderen Teams gelöst werden sollen. Die Zustände des Spielverlaufs sind als Zustandsautomaten modelliert. (Siehe *Spielverlauf und Regeln*, Abschnitt 13.2.3 auf Seite 247.) Der Spieler muss sich ständig informieren können, in welchem Zustand sich das Spiel gerade befindet, da auf Basis dieses Kontextwissens Entscheidungen gefällt werden, wie die kurzfristigen und langfristigen Ziele am Besten zu erreichen sind. Die Zustandsautomaten spiegeln sich in der Nutzungsschnittstelle des Clients wieder, die mit Hilfe unterschiedlicher Modalitäten Auskunft über eintretende Ereignisse und den gegenwärtigen Spielzustand liefert. (Siehe *Nutzungsschnittstelle*, Kapitel 13.5 auf Seite 291.)

Fokale Aktivitäten Die fokalen Aktivitäten des Benutzers ergeben sich aus den einzelnen Spielsituationen, wobei der physische Kontext und der Kontext in Bezug auf die aktuelle Aufgabe von besonderer Bedeutung sind. Ein wichtiger Aspekt der Spielidee ist die Kooperation von Spielern im selben Team. Der Spieler soll also ständig über die Position und Aktivitäten des eigenen Teams informiert werden. Auch bei gegnerischen Spielern spielt die physische Nähe eine wichtige Rolle, da zum Beispiel deren Positionen nur in einem bestimmten Radius angezeigt werden oder auch Spielaktionen wie Neutralisieren nur nahe an einem anderen Spieler wirksam sind. Außerdem enthält das Spiel virtuelle Objekte, die über Positionen auf dem physischen Spielfeld verfügen, an denen dem Spieler besondere Aktionen zur Verfügung stehen. Dabei spielt wieder die physische Nähe der Spieler zu diesen Positionen eine wichtige Rolle.

Mobiler Kontext Eine Modellierung von solchen positionsabhängigen Informationen wird häufig als mobiler Kontext bezeichnet. Eine Definition des mobilen Kontexts nach Wahlster [Wah01] erweitert die Kategorie des ortsbezogenen Kontexts um zwei Faktoren, die Verfügbarkeit und die Qualität von Positionsinformationen. Unter Positionsinformationen werden hier Angaben wie aktuelle Koordinaten, Geschwindigkeit, Beschleunigung, Kopfneigung und Blickrichtung verstanden.

Es ist im Rahmen der Projektgruppe mit den gegebenen technischen Mitteln unmöglich, Angaben wie Kopfneigung und Blickrichtung des Benutzers zu erfassen. Mit den verfügbaren GPS-Daten lässt sich jedoch eine physische, dreidimensionale Position des Spielers errechnen, sowie eine grobe Abschätzung der Geschwindigkeit und der Bewegungsrichtung. (Die Höhe des Spielers über dem Meeresspiegels steht zwar ebenfalls zur Verfügung, jedoch findet sie keine Anwendung im Spiel, da signifikante Höhenunterschiede von Spielern nur in Gebäuden erreicht werden; dort ist jedoch die Erfassung eines zuverlässigen GPS-Signals unmöglich.) Diese Positionsinformationen werden vom Spiel erfasst und haben zum einen den erwähnten Einfluss auf die Spielsituation, zum anderen werden dem Spieler Informationen über unterschiedliche Modalitäten je nach mobilem Kontext zur Verfügung gestellt. (siehe *Multimodalität*, Abschnitt 13.5.1 auf Seite 291.) Dabei spielt die berechnete Geschwindigkeit eine Rolle. Läuft der Spieler, überschreitet also die errechnete Geschwindigkeit einen Grenzwert, werden seine Möglichkeiten auf visuelle Informationen zu reagieren als eingeschränkt eingestuft. Auf diesen Kontext reagiert die Nutzungsschnittstelle in dem wichtige Informationen verstärkt akustisch ausgegeben werden.

Das GPS-Signal enthält auch Informationen über die Qualität der Positionsangaben. Diese werden vom System erfasst und dem Spieler angezeigt, damit dieser sein Verhalten anpassen kann. Eine gewisse Ungenauigkeit in den Positionsdaten gehört jedoch zu den Reizen eines *outdoor*-Spiels und kann sogar von den Spielern als taktisches Element genutzt werden.

Der Kontext des Benutzers wird in der Spiellogik vom Kontextmanager erfasst. Dieses Modul überwacht die anderen Module und bietet eine Schnittstelle für den Zugriff auf den aktuellen Kontext (für weitere Informationen siehe Modul *ContextManager*, Abschnitt 13.4.4.5 auf Seite 277).

13.4 Architekturentwurf

Dieser Abschnitt beschreibt den Entwurf der Softwarearchitektur von NABB. Da es sich bei der Softwarearchitektur um eine noch junge Disziplin handelt, existiert noch keine allgemein anerkannte Definition. Nach [PBG04] bildet die Softwarearchitektur die Brücke zwischen Anforderungsanalyse und Feinentwurf. Sie beschreibt die Strukturen des Softwaresystems auf den obersten Abstraktionsebenen, welche die Architekturbausteine, deren extern sichtbaren Eigenschaften sowie die Beziehungen und Interaktionen zwischen diesen umfassen. Ziel der Softwarearchitektur ist die bessere Beherrschung der Komplexität eines Softwaresystems, so dass die Entwicklungszeit verkürzt, die Wartbarkeit erleichtert und die Qualität der Software verbessert wird.

Dieser Architekturentwurf folgt den in [PBG04] empfohlenen Entwicklungsschritten. Zunächst werden in Abschnitt 13.4.1 die Einflussfaktoren auf die Architektur identifiziert und analysiert. Darauf folgt in Abschnitt 13.4.2 die Analyse der größten Risiken und die Beschreibung der Lösungsstrategien. Die folgenden Abschnitte beschreiben die Architektur aus verschiedenen Sichten: Abschnitt 13.4.3 beschreibt die Kontextsicht, d.h. die Umgebung, in der das System laufen wird. Abschnitt 13.4.4 zeigt die Struktursicht des Systems. Hierzu werden die einzelnen Architekturbausteine beschrieben und die Entscheidungsprozesse, die zur Verwendung bestimmter Architekturstile geführt haben, beschrieben. Abschnitt 13.4.5 enthält die auf der Struktursicht aufbauende Verhaltenssicht und beschreibt die Interaktion zwischen den identifizierten Architekturbausteinen. Abschnitt 13.4.6 beschreibt die Abbildungssicht auf das System, indem es die Realisierung von Client und Server aus den verwendeten Artefakten beschreibt.

13.4.1 Einflussfaktoren

Dieser Abschnitt identifiziert und analysiert die wesentlichen Faktoren, die Einfluss auf die Entwicklung der Softwarearchitektur haben. Die Einflussfaktoren sind in drei Klassen unterteilt: die organisatorischen (siehe Abschnitt 13.4.1.1), die technologischen (siehe Abschnitt 13.4.1.2) und die produktspezifischen Faktoren (siehe Abschnitt 13.4.1.3). Abschnitt 13.4.1.4 stellt die Einflussfaktoren heraus, die den größten Einfluss auf das Design der Architektur haben.

13.4.1.1 Organisatorische Einflussfaktoren

Die organisatorischen Einflussfaktoren haben ihren Ursprung in der Struktur des Unternehmens, in dem das Projekt entwickelt wird. Sie schränken den Softwarearchitekten meist sehr stark in seinen Design-Entscheidungen ein und lassen sich nur selten beeinflussen. Im Falle dieses Projekts wird ein Softwareprodukt im Rahmen einer universitären Projektgruppe erstellt, wodurch sich deutlich andere Einflüsse ergeben, als bei einem Unternehmensprojekt. Nachfolgend werden folgende organisatorische Einflussfaktoren betrachtet: Management, Mitarbeiter, Prozess und Entwicklungsumgebung, Entwicklungszeitplan und Entwicklungsbudget.

O1 Management

> **O1.1 Entwicklungsziel**: Das Projekt zielt nicht darauf ab, ein kommerzielles Produkt zu erzeugen, sondern die Machbarkeit wissenschaftlicher Aspekte zu testen und einen möglichst großen Lerneffekt zu erzielen. Dadurch verändern sich die Gewichtungen der anderen Einflussfaktoren und die Dokumentation der wissenschaftlichen Aspekte tritt in den Vordergrund. So sind in diesem Zusammenhang innovative Aspekte wie Multimodalität und Kontextsensitivität höher als Performanz zu bewerten.

O1.2 Entwickeln vs. Kaufen: Während in Unternehmensprojekten abgewogen werden muss, ob Komponenten selbst entwickelt bzw. gekauft werden sollen, stellt sich diese Frage in einer Projektgruppe aufgrund der mangelnden finanziellen Ressourcen nicht.

O1.3 Zeitplan vs. Funktionalität: Eine Projektgruppe dauert für gewöhnlich genau 12 Monate. Deshalb hat die Einhaltung des Zeitplans eine höhere Priorität als die Umsetzung aller gewünschten Funktionalitäten.

O1.4 Organisationstruktur: Das Projektmanagement übernimmt die Gruppe selbst. Die Betreuer nehmen die Rolle der Auftraggeber ein.

O2 Mitarbeiter

O2.1 Fähigkeiten/Interessen: Die Fähigkeiten und Interessen der Mitarbeiter bzw. Teilnehmer sind zum Teil sehr unterschiedlich. Die Teilnehmer haben meist nur wenig Erfahrung in der Softwareentwicklung. Dadurch wird es besonders wichtig, komplexe Aufgaben in überschaubare Teilaufgaben mit wohldefinierten Schnittstellen herunterzubrechen.

O2.1 Motivation/Anreize/Ziele: Motivation und Anreiz liegen für die Teilnehmer der Projektgruppe in der Möglichkeit, im Rahmen der Projektgruppe Erfahrungen in der Durchführung eines umfangreichen Projekts zu sammeln. Dabei ist in erster Linie das Erlernen des Erkennens und Lösens von auftretenden Problemen und Schwierigkeiten während der verschiedenen Phasen des Projekts ein wesentliches Ziel der Gruppenarbeit.

O3 Prozess und Entwicklungsumgebung

O3.1 Wenige Vorgaben: Dank des universitären Umfelds werden zur Verwendung von Entwicklungsprozessen, Entwicklungswerkzeugen und Plattformen nahezu keine Vorgaben gemacht. Das heißt aber zugleich, dass nur wenig Erfahrung mit den Prozessen, Werkzeugen und Plattformen vorhanden ist. Dadurch erhöht sich die Gefahr, dass Risiken einer möglichen Fehlentwicklung zu lange übersehen werden.

O3.2 Keine kostenpflichtigen Entwicklungswerkzeuge: Entwicklungswerkzeuge sind je nach Zielplattform kostenintensive Produkte, die für eine Projektgruppe nicht angeschafft werden können. Das führt eventuell dazu, dass auf weniger ausgereifte, aber dafür verfügbare Produkte zurückgegriffen werden muss, wodurch die Effizienz der Entwicklung beeinträchtigt wird.

O4 Entwicklungszeitplan

O4.1 Strikte Zeitschranken: Die Dauer einer Projektgruppe beträgt immer genau 12 Monate. Durch die Projektgruppenordnung wird vorgeschrieben, dass jeder Teilnehmer 20 Stunden Zeit pro Woche in das Projekt investieren soll. Die Planung beschränkt sich also darauf, festzustellen, welche Funktionalität in der vorgegebenen Zeit mit den gegebenen Ressourcen umgesetzt werden soll.

O5 Entwicklungsbudget

O5.1 Begrenzte Mittel: Einer Projektgruppe stehen Räumlichkeiten, Arbeitsplätze und Zeit als Budget zur Verfügung. Finanzielle Mittel zur Anschaffung neuer Hardware/Software oder zur Weiterbildung der Teilnehmer können nur in Ausnahmefälle bereitgestellt werden.

13.4.1.2 Technologische Einflussfaktoren

Technologische Einflussfaktoren resultieren aus Vorgaben bezüglich Hard- und Software sowie einzuhaltenden Standards. Sie schränken das Design der Softwarearchitektur ein. Es gilt besonders die Aufgrund des technischen Fortschritts zukünftig zu erwartenden Änderungen zu berücksichtigen. Nachfolgend werden die Einflussfaktoren Hardware und Softwaretechnologien betrachtet.

T1 Hardware

T1.1 Mobile Endgeräte: Die Anforderung „kontextsensitive Umbebungserkundung“ ist nur durch den Einsatz mobiler Informationssysteme erfüllbar. Aufgrund der begrenzten finanziellen Mittel müssen die an der Universität vorhandenen Geräte benutzt werden. Dabei handelt es sich um HP iPaq PocketPC-Modelle der Serie 5450. Eine nahezu flächendeckende Internetverbindung kann via GPRS über als Modem verwendete Mobiletelefone oder spezielle Chipsätze ermöglicht werden. Übertragungstechnologien wie WLAN und Bluetooth können ebenfalls zur Kommunikation zwischen den mobilen Endgeräten eingesetzt werden. Sie gewährleisten keine flächendeckende Datenübertragung, zeichnen sich aber gegenüber GPRS durch andere Qualitätsmerkmale wie einer höheren Übertragungsgeschwindigkeit bei WLAN aus.

T1.2 GPS Empfänger: Um die Umgebung erkunden zu können, ist die Bestimmung der Position eines mobilen Endgeräts nötig. Es existieren verschiedene Techniken um ein Gerät zu lokalisieren von denen die meisten auf die Ortung des Gerätes innerhalb eines begrenzten Raumes beschränkt sind. Aufgrund der Anforderung, ein Gerät außerhalb von Gebäuden und in einem weitläufigen Areal lokalisieren zu können, wird auf das GPS-System zurückgegriffen. Für den Empfang der GPS-Koordinaten werden die an der Universität verfügbaren Empfänger eingesetzt (Falcom NAVI-1).

T2 Softwaretechnologien

T2.1 Betriebssystem: Als Betriebssystem ist auf der Hardware Windows Mobile 2003 vorinstalliert. Von der Verwendung anderer Betriebssysteme wurde uns abgeraten, da bei deren Installation die Hardware Schaden nehmen könne.

T2.2 Zielplattform: Als Zielplattform wurde das .NET Compact Framework von Microsoft zur Realisierung der Software für die mobilen Endgeräte gewählt.

T2.3 Programmiersprache: Es wird C# als Programmiersprache verwendet, da die Ähnlichkeit zu der bereits allen Teilnehmern bekannt Sprache Java sehr groß ist.

13.4.1.3 Produktfaktoren

Die Produktfaktoren spiegeln die funktionalen und nicht-funktionalen Anforderungen an die Software wider. Diese haben den größten Einfluss auf das Design der Softwarearchitektur. Besonderes Augenmerk sollte dabei auf die Qualitätsattribute der nicht-funktionalen Anforderungen gelegt werden, da diese meist vernachlässigt werden. Zusammen mit den funktionalen Anforderungen werden sie im Folgenden betrachtet.

P1 Funktionale Anforderungen

P1.1 kontextsensitive Umgebungserkundung: Die Software soll kontextsensitiv auf ihre Umwelt reagieren. Dazu wird eine Vielzahl von Sensoren aller Art benötigt. Die eingesetzten Sensoren bestimmen den Grad der Kontextsensitivität. Stehen neue Sensoren zur Verfügung, soll deren Anbindung möglichst einfach realisierbar sein.

P1.2 Multimodale Ausgaben: Die Architektur soll multimodale Ausgaben ermöglichen. Dabei sollen die Ausgabegeräte nicht a priori festgelegt sein. Die multimodale Ausgabe soll daher gekapselt werden und möglichst gut um die Unterstützung neuer Ausgabemedien erweiterbar sein.

P1.3 Multimodale Eingaben: Analog zur *multimodalen Ausgaben* verhält es sich mit den Eingabemöglichkeiten.

P2 Nicht-funktionale Anforderungen

P2.1 Leistung: Die Software wird zur Unterstützung eines Spiels entwickelt, das unter Echtzeitbedingungen durchgeführt wird. Aus diesem Grund müssen die Reaktionszeiten der Software möglichst gering sein, da lange Reaktionszeiten entgegen der natürlichen Erwartungshaltung des Menschen stehen und somit den Spielspaß hemmen.

P2.2 Zuverlässigkeit/Fehlerbehandlung: Da die Software keine sicherheitskritischen Systeme überwacht oder steuert, sind Zuverlässigkeit und Fehlerhandling keine primäreren Einflussfaktoren.

P2.3 Wartbarkeit: Da der Zeitraum einer Projektgruppe auf 12 Monate begrenzt ist, wird eine eventuelle Wartung der Software nicht von der Projektgruppe selbst durchgeführt. Es wird daher darauf geachtet, dass die grundlegenden Konzepte der Architektur gut dokumentiert sind.

13.4.1.4 Zusammenfassung der wesentlichen Einflussfaktoren

Bei dem Entwurf der Struktur spielen folgende Einflussfaktoren eine entscheidende Rolle:

- **Überschaubare Teilaufgaben**: Während der Realisierung der Software sind bis zu 9 Personen an der Implementierung beteiligt. Darum ist es wichtig, möglichst voneinander unabhängige, in sich geschlossene Architekturbausteine mit wohldefinierten Schnittstellen zu bilden, so dass paralleles Arbeiten ermöglicht bzw. erleichtert wird.
- **Austauschbarkeit und Erweiterbarkeit**: Die Umsetzung der Aspekte kontextsensitive Umgebungserkundung und Multimodalität erfordern den intensiven Einsatz externer Hardware in Form verschiedener Sensoren. Dabei soll berücksichtigt werden, dass nach Fertigstellung des Projekts neue Hardware zur besseren Realisierung dieser Aspekte zur Verfügung stehen könnte. Ebenso soll berücksichtigt werden, dass speziell die Hardware im Bereich der mobilen Endgeräte einem rasanten technologischen Wandel unterliegen. Darum müssen speziell die Komponenten zur Ansteuerung dieser Hardware austauschbar und erweiterbar sein.
- **Verständlichkeit und Wartbarkeit**: Da einige Teilnehmer der Projektgruppe nur wenig Erfahrung in der Softwareentwicklung mitbringen, muss besonders darauf geachtet werden, dass die Architektur gut verständlich ist. Dies wirkt sich zugleich positiv auf die Wartbarkeit der Software aus.

13.4.2 Risikoanalyse und Lösungsstrategien

13.4.2.1 Einführung neuer Technologien

Da das Softwaresystem mit Technologien realisiert werden soll, mit deren Umgang die Gruppenmitglieder wenig oder gar keine Erfahrung besitzen, könnte die Qualität der Software unter der fehlenden Expertise leiden.

Beeinflussende Faktoren

O2.1: Die Teilnehmer haben wenig Erfahrung mit Projektentwicklungen.

O4.1: Die strikte Projektdauer erfordert eine Berücksichtigung der Einarbeitungszeit in neue Technologien im Zeitplan.

T1: Die Teilnehmer haben keine Erfahrung mit der einzusetzenden Hardware.

T2: Die Teilnehmer haben keine Erfahrung mit der einzusetzenden Software, insbesondere mit der verwendeten Programmiersprache C#.

Lösung Die Teilnehmer müssen frühzeitig Erfahrung im Umgang mit den Technologien sammeln.

Strategie Machbarkeitsstudien für Absicherung und Erfahrung
Um die Architekturrisiken mit den neuen Technologien abzusichern, werden Machbarkeitsstudien durchgeführt. Diese werden von den zukünftigen Entwicklern für das .NET Framework programmiert, so dass diese zugleich Erfahrung in der Programmierung mit dem .NET Framework sammeln können. Die Ergebnisse werden im Rahmen der Qualitätssicherung dokumentiert.

Strategie *Pair-Programming*
Während der Implementierung wird das *Pair-Programming-Prinzip* angewendet. Durch das Programmieren zu zweit wird ein höherer Lerneffekt erzielt. Zudem wird die geschriebene Software kontinuierlich einem Review durch den nicht-programmierenden Partner unterzogen, was zu einer besseren Qualität der Software führt.

Strategie Testgetriebenes Programmieren
Um die Qualität der Software zu gewährleisten wird testgetriebenes Programmieren eingesetzt.

13.4.2.2 Unzureichende Technologien

Wichtig zur Umsetzung von NABB sind eine möglichst genaue Ortsbestimmung der Spieler und eine Datenübertragung zwischen Client und Server mit möglichst geringer Latenzzeit. Die verfügbaren Technologien sind aber möglicherweise nicht ausgereift genug, um diese Ansprüche zu erfüllen.

Beeinflussende Faktoren

T1.1: Es stehen nur die von der Universität gestellten Geräte zur Verfügung

O4.1, O5.1: Es sind weder Zeit noch Mittel vorhanden, um alternative Technologien zu entwickeln/-kaufen.

Lösung Aus unzureichenden Technologien resultierende Probleme müssen frühzeitig aufgedeckt werden, so dass die Anforderungen korrigiert werden können.

Strategie Technikreferate
Die risikobelasteten Technologien werden in Technikreferaten untersucht. Dabei soll frühzeitig festgestellt werden, welche Alternativen sich bieten, und welche die besten nicht-funktionalen Eigenschaften bezüglich der Anforderungen besitzen.

Strategie Machbarkeitsstudien/Prototyping
Es werden Machbarkeitsstudien und Prototypen zum Testen der risikobelasteten Technologien durchgeführt, um deren Laufzeitverhalten frühzeitig festzustellen.

Strategie Anpassung des Spiels
Die Regeln von NABB werden an eventuelle technische Einschränkung angepasst. Beispiel hierfür ist das Neutralisieren eines Mitspielers. Die wahrnehmbare Latenzzeit der Datenübertragung wird als Wartezeit in den Spielablauf integriert.

13.4.3 Kontextsicht

Die Kontextsicht beschreibt die Einbettung des zu entwickelnden Systems in die Umgebung. Das System wird dabei von außen im Sinne einer Blackbox betrachtet. In dieser Sicht werden Systemgrenzen identifiziert und die nach außen sichtbaren Schnittstellen festgelegt [PBG04].

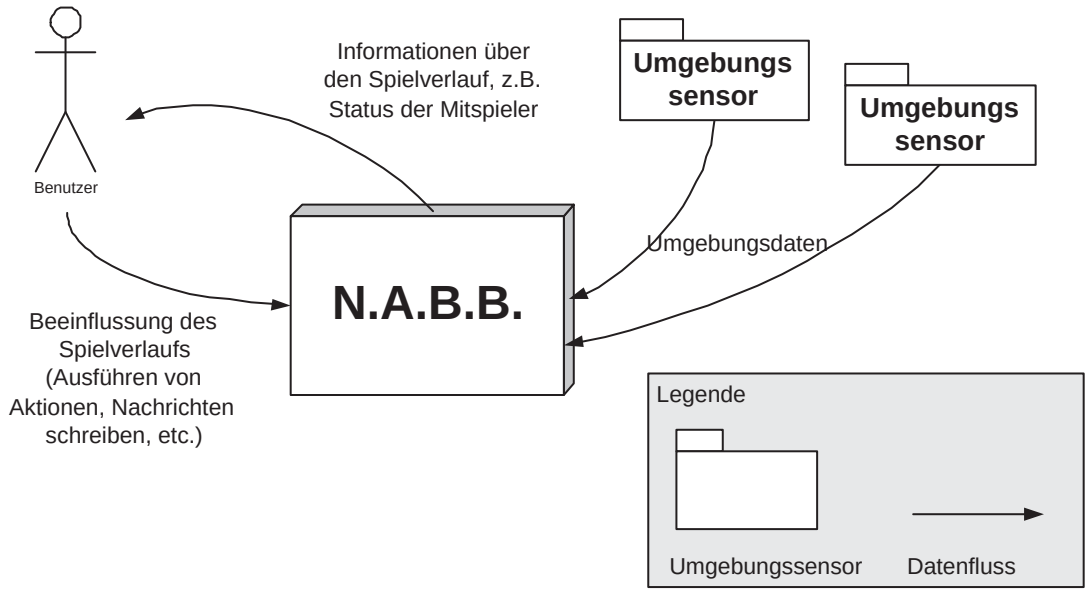


Abbildung 13.15: Kontextsicht

Abbildung 13.15 zeigt die graphische Darstellung der Kontextsicht. Die externen Elemente, also diejenigen, mit denen das System interagiert, sind der Benutzer und die Umgebungssensoren. Der Nutzer beeinflusst das Spiel direkt, indem er Aktionen ausführt, Nachrichten schreibt, usw. .... Das System informiert den Nutzer über den aktuellen Stand des Spiels, z.B. den Status der Mitspieler. Die Schnittstelle zwischen Mensch und System soll multimodal arbeiten. Vorgesehen ist die multimodale Darstellung von Spielereignissen und wichtigen Spielzuständen.

Die angeschlossenen Sensoren dienen zur Erkundung der Umgebung und liefern damit eine Basis zur Einschätzung des Kontexts eines Spielers. Sie beobachten Veränderungen in der Umgebung des Nutzers und leiten die gewonnen Daten zur Auswertung an das System weiter. Sie können damit indirekt als Eingabemedium verwendet werden, indem der Spieler sich in eine Umgebung begibt, die über die Umgebungssensoren die gewünschte Reaktion hervorruft.

Primäres Beispiel für einen Umgebungssensor ist die GPS-Maus, ein externes Gerät zur Ortsbestimmung auf Basis von GPS-Satelliten. Der Spieler kann indirekt Eingaben über die GPS-Maus tätigen, indem er sich an bestimmte, für das Spiel relevante Orte (z.B. Missionsziele) begibt. Weitere Beispiele für Umgebungssensoren sind, Bestimmung der Umgebungslautstärke mittels Mikrophon, Gestenerkennung mittels Kamera und Bestimmung der Umgebungshelligkeit mittels Lichtsensor.

13.4.4 Struktursicht

In dieser Sicht werden die grundlegenden Architekturbausteine des Systems und ihre Beziehung aus statischer Perspektive dargestellt. Die Zerlegung wird je nach Komplexität hierarchisch fortgeführt. Die Struktursicht beschreibt den Aufbau der wesentlichen Architekturbausteine mit ihren Schnittstellen und legt die Verantwortlichkeiten sowie die möglichen Wechselwirkungen fest. Die Sicht ist eng mit der Verhaltenssicht (13.4.5) gekoppelt, da dort die zur Laufzeit beobachtbaren Wechselwirkungen exemplarisch beschrieben werden. Die Struktursicht bildet den Rahmen für die Betrachtung des Innenlebens eines Systems und den Ausgangspunkt für die Beschreibung der dynamischen Abläufe. Für diese Sicht sind die Architekturbausteine, die Schnittstellen und die Kommunikationsbeziehungen relevant [PBG04].

13.4.4.1 Architekturstil der Systemarchitektur

Das Design des Spiels, erfordert eine Realisierung durch ein verteiltes System, da mehrere Spieler an verschiedenen Orten daran teilnehmen können. Die Architekturstile zur Realisierung verteilter Systeme, die zur Diskussion standen, sind: *Client-Server* und *Peer-to-Peer*.

- *Client-Server*: Der Server bildet die koordinierende Instanz. Alle Kommunikation der Systems läuft über den Server. Jeder Spieler wird durch einen Client repräsentiert.
  - *Thin-Client*: Beim Client-Server Stil mit Thin-Client übernimmt der Client lediglich die Darstellung, während die Berechnungen komplett auf dem Server stattfinden (Beispiel: Browser). Das Systemmanagement ist in diesem Fall sehr simpel, da es nahezu komplett auf dem Server stattfindet. Der Nachteil ist, dass Server und Netzwerkanbindung ausreichend Kapazität benötigen, um die gewünschte Anzahl an Clients bedienen zu können.
  - *Fat-Client*: Beim Fat-Client ist ein Teil der Logik auf den Client ausgelagert. Die Komplexität des Systemmanagements nimmt dabei zu, dafür sinken aber die Anforderung an die Kapazität des Servers und der Netzwerkanbindung.
- *Peer-to-Peer*: Beim Architekturstil Peer-to-Peer wird die Koordinationsfunktion auf die Clients - die in diesem Fall dann Peers genannt werden - ausgelagert. Die Kommunikation kann direkt zwischen den Peers stattfinden. Wie beim Client-Server Stil wird jeder Spieler durch einen Client/Peer repräsentiert.
  - *Dezentrales Peer-to-Peer*: Beim dezentralen Peer-to-Peer sind alle Peers gleichberechtigt und verantwortlich für die Koordination des Netzwerks.
  - *Assistiertes Peer-to-Peer*: Beim assistierten Peer-to-Peer existieren eine oder mehrere Instanzen, die die Verwaltung des Netzwerks übernehmen. Bei diesen Instanzen kann es sich um so genannte ausgezeichnete Peers oder auch um dedizierte Server handeln. Die Unterscheidung zum Client-Server Stil liegt hier darin, dass die Peers trotzdem das Spiel koordinieren.

Die Realisierung des Systems im Peer-to-Peer Stil wurde verworfen, da mit den mobilen Endgeräten eine direkte Netzwerkverbindung nur über kürzere Distanz (ungefähr 10 m) mittels Bluetooth hergestellt werden kann und dies nicht ausreicht, um die Anforderungen des Spiels zu erfüllen. Andere Netzwerkverbindungen wie GPRS und UMTS nutzen automatisch Internetserver, so dass direkte Verbindungen zwischen zwei Geräten nur über Relais möglich gewesen wäre, was zu deutlich Performanzverlusten geführt hätte. Weiterhin ist bei den mobilen Endgeräten davon auszugehen, dass die verfügbaren Ressourcen wie Rechenzeit und Speicher knapp sind, so dass sich die Auslagerung der Koordinationsfunktionen auf die Clients negativ auf die Performanz ausgewirkt hätte.

Die Realisierung des Systems im Client-Server Stil mit Thin-Client wurde ebenfalls verworfen, da die Netzwerkverbindung zwischen den mobilen Endgeräten und den Servern unzuverlässig und unperformant sind. Deshalb wird das System im Client-Server Stil mit Fat-Client mit dem Ziel realisiert, möglichst viel Netzwerkverkehr einzusparen. Abbildung 13.16 zeigt die schematische Struktur des gewählten Systemarchitekturstils.

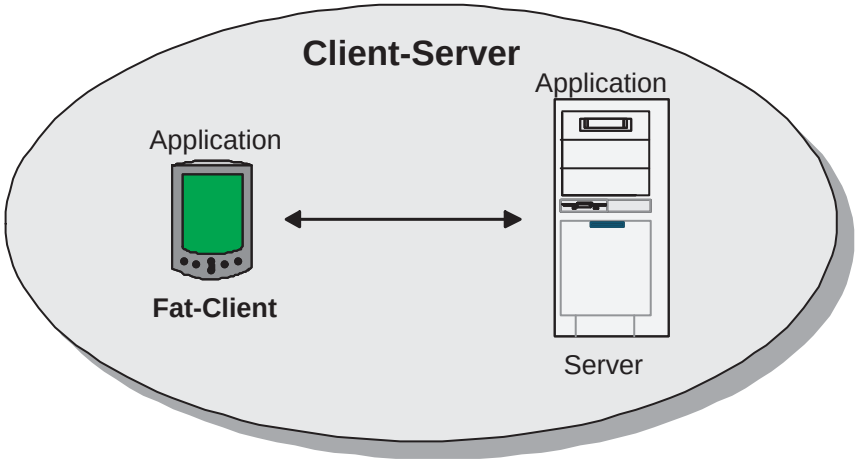


Abbildung 13.16: Systemarchitektur Client-Server mit Fat-Client

13.4.4.2 Architekturstil der Softwarearchitektur

Zur Realisierung der Softwarearchitektur standen die Architekturstile *Objektorientierung* und *Implicit Invocation* zur Diskussion.

- *Objektorientierung*: Bei dem Stil *Objektorientierung* werden Architekturen aus kooperierenden Objekten zusammengesetzt. Objekte kapseln dabei Zustandsvariablen und Verhalten. Die Zusammenarbeit zwischen Objekten geschieht durch Zusendung von Nachrichten bzw. durch den Aufruf von Funktionen auf den Objekten. Die Auswirkungen dieses Architekturstils auf die Qualitätsattribute sind vielfältig; je nach Anwendung des Stils sind positive und negative Folgen für die einzelnen Attribute wie Leistung, Wartbarkeit und Zuverlässigkeit denkbar.

- *Implicit Invocation / Ereignisbasierte Steuerung*: Beim Stil der *Objektorientierung* muss dem Sender die Identität des empfangenen Objekts bekannt sein. Will man diese Kopplung vermeiden, so bietet sich der Stil *Implicit Invocation* an. Hier versenden Bausteine Ereignisse, die von ihnen unbekannten Bausteinen empfangen und bearbeitet werden. Dieser Stil wird daher auch *ereignisbasierte Steuerung* genannt. Der Vorteil dieses Stils ist, dass er erlaubt, die einzelnen Bausteine voneinander zu entkoppeln, was positive Auswirkung auf Flexibilität und Wartbarkeit hat. Der Nachteil ist, dass für die Implementierung der impliziten Aufrufe über Ereignisse eine zusätzliche Maschinerie als Kommunkationssystem nötig ist. Dieser Overhead beeinflusst die Leistung des Systems negativ. Zusätzlich leidet die Übersichtlichkeit des Systems.

Die negative Beeinflussung der Leistung durch den Overhead beim *Implicit Invocation* Architekturstil und die schlechtere Übersichtlichkeit, gewichtete die Projektgruppe stärker als die bessere Flexibilität und Wartbarkeit, so dass zur Realisierung von Client und Server der Architekturstil *Objektorientierung* eingesetzt wird.

13.4.4.3 Generischer Aufbau eines Architekturbausteins

Die Umsetzung des Architekturstils *Objektorientierung* erfolgt, indem die Funktionalitäten von Client und Server durch gleichberechtigte Bausteine realisiert werden. Abbildung 13.17 verdeutlicht den Aufbau eines Architekturbausteins, im Folgenden *Modul* genannt.

Ein Modul besteht aus einer Klasse, die den Namen des Moduls trägt (`ConcreteModule`). Dazu existieren eventuell weitere Hilfsklassen (`Helper1` / `Helper2`), wobei das Fassaden-Muster (siehe [GHJV94]) angewendet wird, so dass die Modulklasse die Hilfsklassen versteckt. Die Kommunikation zwischen den Modulen erfolgt einerseits durch das Aufrufen von Methoden auf der Modulklasse. Andererseits können sich Module als Observer bei anderen Modulen registrieren, wenn sie die entsprechende Observer Schnittstelle (`ConcreteModuleObserver`) implementieren (siehe [GHJV94], Observer Muster).

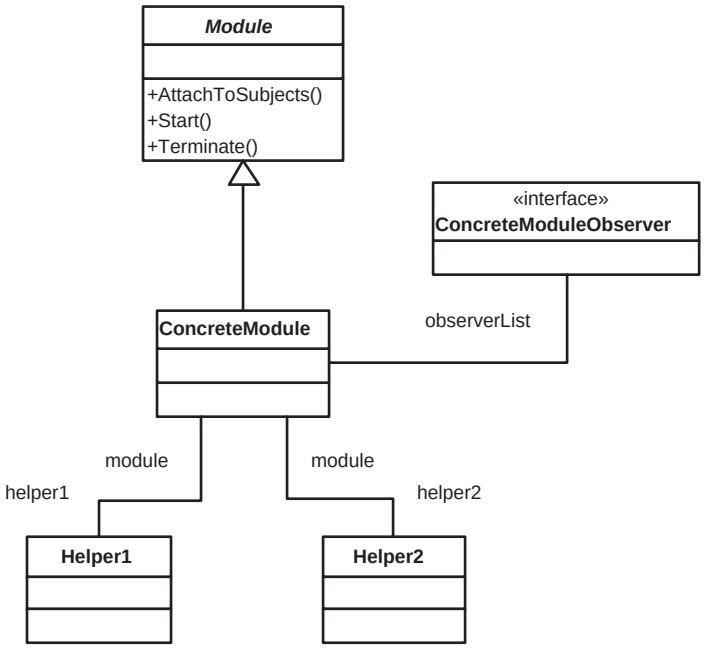


Abbildung 13.17: Generischer Aufbau eines Architekturbausteins

Die öffentlichen Klassen aller Module sind auf Clientseite von der Klasse „ClientModule“, auf Serverseite von der Klasse „ServerModule“ abgeleitet. Über deren Methoden steuert die Applikation den Lebenszyklus der Module. Im Einzelnen sind das:

`AttachToSubjects()`: Durch den Aufruf dieser Methode wird sichergestellt, dass das Modul zu Beginn seiner Lebenszeit als Observer auf andere Module registriert wird. Wenn diese Methode aufgerufen wird, sind alle über die Applikation erreichbaren Module bereits initialisiert.

`Start()`: Nach der Registrierung wird die Methode „Start()“ eines Moduls aufgerufen. Soll ein Modul in einem eigenen Thread laufen, dann muss es die Methode „Start()“ überschreiben und das Erstellen eines Threads an dieser Stelle realisieren.

`Terminate()`: Diese Methode kann überschrieben werden, um vor dem Beenden belegte Ressourcen wieder freizugeben. Dabei kann es sich beispielsweise um das Beenden eines gestarteten Threads handeln.

13.4.4.4 Architekturbausteine des Servers

Abbildung 13.18 zeigt die Architekturbausteine, aus denen der Server zusammengesetzt wird. Bei der Darstellung der Architektur wurde mit dem UML-Standard gebrochen, um die Kommunikationsbeziehungen zwischen den Modulen adäquat darstellen zu können. Die Bedeutung der einzelnen Symbole wird im folgenden erläutert:

Die aus UML entnommenen Komponenten stellen die Module dar. Die Beschriftung enthält den Namen des Moduls. Die Observer-Beziehungen der Module untereinander werden durch die gestrichelten Pfeile dargestellt, wobei der Pfeil immer von dem Subjekt zum Observer führt. Der Pfeil zeigt die Datenflussrichtung an, er führt also von dem Objekt weg, in dem die Änderungen observiert werden und zeigt auf das Objekt, das über die Änderung informiert wird. Die Pfeile mit den durchgezogenen Linien zeigen an, dass ein Modul auf die öffentliche Schnittstelle eines anderen Moduls zugreift. Der Pfeil führt dabei von dem Modul fort, das die Methoden aufruft und zeigt auf das Modul, dessen Methoden aufgerufen werden. Zeigt ein Pfeil in beide Richtungen, greifen die Module gegenseitig auf die öffentlichen Schnittstellen des jeweils anderen zu. Die Beschriftung an den Pfeilen skizziert die Daten und Aktionen, die durch diese Kommunkationsbeziehung transportiert bzw. ausgeführt werden.

Die Module sind den „Schichten“ `Logic` bzw. `Communication` zugeordnet. Das Wort Schichten steht deshalb in Anführungszeichen, weil es sich nicht um Schichten nach den Architekturstilen *N-Tier* oder *Layer* handelt. Die Einteilung geschah nur, um die Übersichtlichkeit des Systems zu verbessern.

In der Beschreibung der Module tauchen die Begriffe *Ereignis*, *Nachricht* und *Postnachricht* auf. Bei *Postnachrichten* handelt es sich um eMail-ähnliche Mitteilungen, die von Spielern verfasst und an andere Spieler gesendet werden. *Nachricht* und *Ereignis* sprechen von Mitteilungen, die zwischen Architekturbausteinen ausgetauscht werden, also von Bausteinen erzeugt, um bestimmte Operationen anzustoßen. Die Begriffe werden zum Teil synonym verwendet. Aus Sicht der Softwarearchitektur wäre Nachricht der korrekte Begriff dafür. Da *Nachrichten* aber häufig spielbezogene Ereignisse transportieren, werden sie auch zum Teil *Ereignisse* genannt.

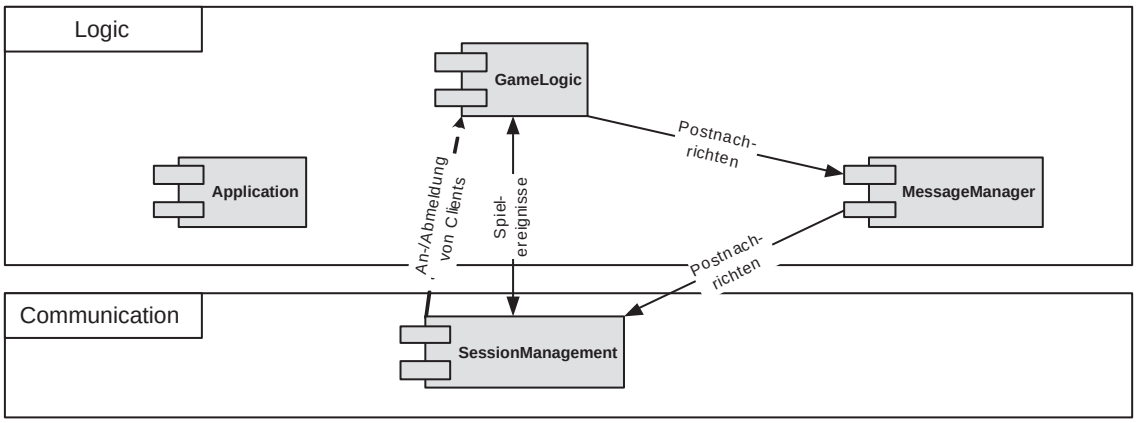


Abbildung 13.18: Architekturbausteine des Servers

Es folgt die Beschreibung der Module, ihrer Schnittstellen und ihrer Kommunikationsbeziehungen. Die Beschreibungen sind alphabetisch sortiert:

GameLogic Dieses Modul implementiert die Spiellogik. Es verwaltet alle Spiel Objekte, wie Spieler, Missionsziele, Startpunkte, etc. . Es reagiert auf von den Clients eingehende Spielereignisse, indem es daraus die neuen Zustände der Spielobjekte berechnet und ggf. die Zustandsänderungen an Clients wei-

terleitet. Das Modul hält außerdem die Szenariodaten vor, die aus der in Abschnitt 13.2.4.1 beschriebenen XML-Datei ausgelesen werden.

Schnittstellen: Über die öffentliche Schnittstelle können die teilnehmenden Spieler verwaltet werden, der Einsatz von Fähigkeiten gesteuert werden, Missionsziele gelöst werden, usw.

Kommunikationsbeziehungen: Dieses Modul leitet Spielereignisse an die Netzwerkschicht und Postnachrichten an den MessageManager weiter.

Risikoanalyse: Die Verwendung von XML-Dateien zur Speicherung der Szenariodaten ist risikobelastet, da die XML-APIs im .NET Compact Framework nicht vollständig implementiert sind.

MessageManager Dieses Modul nimmt eingehende Postnachrichten auf, und leitet sie an die angegebenen Clients weiter. Dabei wird darauf geachtet, dass nur Spieler die Nachrichten empfangen, die dies laut Regelwerk auch dürfen.

Schnittstellen: Die öffentliche Schnittstelle erlaubt es, Postnachrichten weiterleiten zu lassen. Über die zugehörige Observerschnittstelle können andere Module das Eingehen von Nachrichten verfolgen.

Kommunikationsbeziehungen: Der MessageManager erhält die weiterzuleitenden Postnachrichten durch das GameLogic Modul. Es reicht die Nachrichten direkt an das Kommunikationsmodul weiter.

NetworkManager Dieses Modul verwaltet die Verbindungen zwischen Clients und Servern mittels Sessionmanagement und organisiert die Serialisierung der Nachrichten, die zwischen Client und Server ausgetauscht werden.

Schnittstellen: Die öffentliche Schnittstelle nimmt die zu versendenden Nachrichten auf. Über die Observerschnittstelle wird mitgeteilt, wenn ein Client eine Verbindung hergestellt oder beendet hat.

Kommunikationsbeziehungen: Das GameLogic Modul nimmt die eingehenden Nachrichten auf und verarbeitet sie. Aus dieser Verarbeitung entstehen dann eventuell wieder Nachrichten, die über den NetworkManager versendet werden sollen. Das GameLogic Modul implementiert außerdem die Observerschnittstelle und reagiert so auf das Spiel betretende bzw. verlassende Spieler.

Risikoanalyse: Die Technologie zum Herstellen von Internetverbindungen zwischen mobilem Endgerät und einem Server ist mit Risiko behaftet, weil die Technologie auf sehr komplexen Vorgängen basiert. Eine Machbarkeitsstudie sollte versuchen, Risiken frühzeitig aufzuspüren.

ServerApplication Dieses Modul verwaltet die weiteren Module des Servers und setzt dadurch die Anwendung zusammen. Es hält die Referenzen auf alle Module, initialisiert sie beim Start der Anwendung und steuert ihren Lebenszyklus. ServerApplication besitzt außerdem eine Warteschlange für eingehende Ereignisse, die dann durch den Thread des Moduls abgearbeitet werden, indem die für das Ereignis korrekten Methoden auf den anderen Modulen aufgerufen werden.

Schnittstellen: Über die Schnittstelle des Moduls können alle Module sowie die Ereigniswarteschlange referenziert werden.

Kommunikationsbeziehungen: Während das Modul die wartenden Ereignisse abarbeitet, greift es auf die öffentlichen Schnittstellen aller anderen Module zu.

Anmerkung: In der graphischen Übersicht (Abb. 13.18) wurden die Kommunikationsbeziehungen dieses Moduls aus Gründen der Übersichtlichkeit weggelassen, da es Kommunikationsbeziehungen zu jedem Modul besitzt.

13.4.4.5 Architekturbausteine des Clients

Abbildung 13.19 zeigt die Architekturbausteine, aus denen der Client zusammengesetzt wird. Die Darstellung ist nach dem gleichen Schema aufgebaut, wie in Abschnitt 13.4.4.4. Es folgt die Beschreibung der Module, ihrer Schnittstellen und ihrer Kommunikationsbeziehungen. Die Beschreibungen sind alphabetisch sortiert:

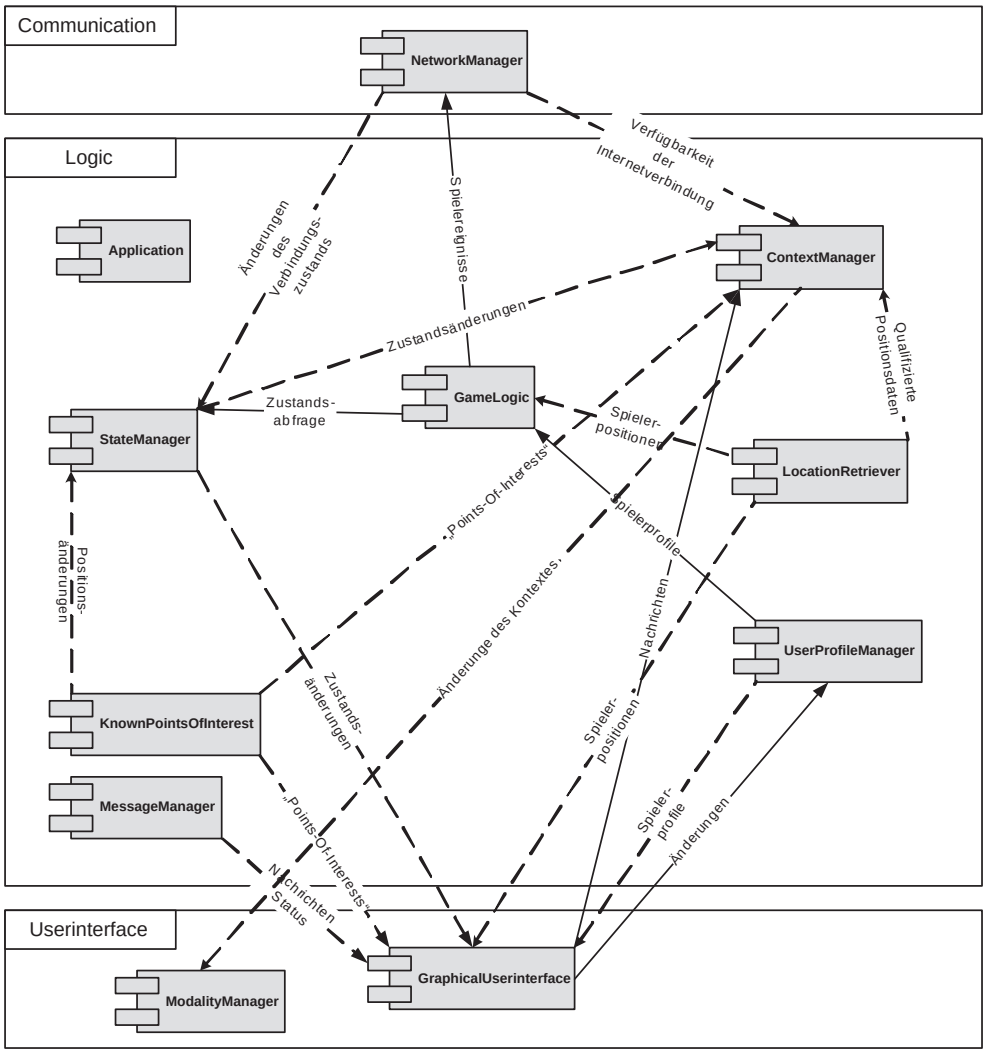


Abbildung 13.19: Architekturbausteine des Clients

Application Dieses Modul ist das Hauptmodul des Clients, das alle anderen Module erzeugt und koordiniert. Es speichert die Daten, mit denen die Anwendung initialisiert werden soll. Zusätzlich arbeitet es die Ereignisse ab und verteilt sie auf die entsprechenden Module.

Schnittstellen: Über die Schnittstelle des Moduls können alle Module sowie die Ereigniswarteschlange referenziert werden.

Kommunikationsbeziehungen: Während das Modul die wartenden Ereignisse abarbeitet, greift es auf die öffentlichen Schnittstellen aller anderen Module zu.

ContextManager Dieses Modul überwacht andere Module, um den Kontext des Spielers zu bestimmten. Der Kontext wird in zwei Varianten vorgehalten: in der qualifizierten Form werden möglichst genaue Daten zum Kontext des Spielers gesammelt, wie z.B. seine Bewegungsgeschwindigkeit und -richtung. Die andere Form des Kontexts stellt eine interpretierte Variante des qualifizierten Kontexts dar. Dieser berechnete/interpretierte Kontext reduziert den Kontext der Bewegungsgeschwindigkeit auf den Zustand „laufend/nicht laufend“.

Schnittstellen: Über die öffentliche Schnittstelle des Moduls können sowohl der qualifizierte als auch der berechnete Kontext referenziert werden. Über die Observerschnittstelle können Änderungen an beiden Formen des Kontexts verfolgt werden.

Kommunikationsbeziehungen: Der ModalityManager überwacht Änderungen am Kontext, um die Überprüfung der korrekten Modalitäten zur Ausgabe von Ereignisse und Zuständen immer direkt bei einer Änderung des Kontexts neu durchführen zu können. Ebenso überwacht das GraphicalUserInterface den Kontext, um ihn einerseits zu visualisieren, und andererseits, um die Anzeige an den Kontext anpassen zu können. Der ContextManager selbst berechnet den Kontext, indem er sich als Observer bei folgenden Modulen registriert:

- `LocationRetriever`: um die Bewegungsgeschwindigkeit und damit den Grad der Ablenkung berechnen zu können
- `StateManager`: da die Programmzustände direkt den Kontext des Spielers beeinflussen
- `NetworkManager`: da die Verfügbarkeit der Internetverbindung Teil des Kontexts ist
- `GraphicalUserInterface`: da die aktuelle Ansicht ebenfalls Teil des Kontexts ist

GameLogic Dieses Modul implementiert einen Teil der Spielregeln und erlaubt die Gültigkeit von Aktionen zu prüfen, ohne den Server anzusprechen. Das dient in erster Linie der Reduktion des Netzwerkverkehrs.

Schnittstellen: Über die öffentliche Schnittstelle können alle Spielaktionen ausgeführt werden, wie z.B. das Lösen von Missionsziele, Aktivieren der Tarnung, usw.

Kommunikationsbeziehungen: Die GameLogic ist Observer auf dem LocationRetriever. Ändert sich die aktuelle Position des Spielers, übernimmt die GameLogic das Versenden eines entsprechenden Spielereignisses.

GraphicalUserinterface Dieses Modul kapselt den Aufbau der grafischen Nutzungsschnittstelle. Deren Aufbau wird detailliert in Abschnitt 13.5 erläutert. Es werden mehrere Anzeige- und Steuerelemente, sowie eine Hauptansicht definiert. Die Hauptansicht kann verschiedene Inhalte, wie z.B. die Karte oder den aktuellen Kontext anzeigen.

Schnittstellen: Die Nutzungsschnittstelle greift über eine Fassade (siehe Fassade Entwurfsmuster[2]) auf die Module der „Logik-Schicht“ zu. Gleichzeitig bezieht sie alle darzustellenden Informationen

[2] http://www.dofactory.com/Patterns/PatternFacade.aspx

über die Observerschnittstellen anderer Module. Dies entspricht der Trennung von Daten und ihrer Darstellung bzw. Bearbeitung nach dem Model-View-Controler Muster. Die Dynamik dieses Musters wird in Abschnitt 13.4.5.4 detaillierter erläutert.

Kommunikationsbeziehungen: Das Modul ist als Observer registriert bei:

- `LocationRetriever`: um die aktuelle Position des Spielers anzeigen zu können
- `StateManager`: um die aktuelle Position der sichtbaren Points-of-Interest anzeigen zu können
- `MessageManager`: um die eingehenden Nachrichten anzeigen zu können
- `StateManager`: um Zustände, wie „Gegner in der Nähe“ oder „Verbindung unterbrochen“ anzeigen zu können
- `UserProfileManager`: um die aktuellen Spielerprofile anzeigen zu können

KnownPointsOfInterest Dieses Modul speichert die aktuell für den Client sichtbaren Points-of-Interest.

Schnittstellen: Über die öffentliche Schnittstelle können Points-of-Interest verwaltet werden. Über die Observerschnittstelle, können andere Module das Auftauchen, das Verschwinden, sowie Änderungen von Points-of-Interest überwachen.

Kommunikationsbeziehungen: Änderungen an den Points-of-Interest werden über die Observerschnittstelle an den ContextManager zur Validierung des Kontexts, zur Nutzungsschnittstelle zwecks graphischer Darstellung und zum StateManager zur Neuberechnung der Zustände übergeben.

LocationRetriever Dieses Modul bestimmt die Position des Clients. Es kapselt die verwendete Positionsbestimmungstechnologie.

Schnittstellen: Über die Schnittstelle des Moduls können Daten über die aktuelle Position und Bewegungsgeschwindigkeit des Clients abgefragt werden. Mittels der Observerschnittstelle können sich Module über Änderungen dieser Daten informieren lassen.

Kommunikationsbeziehungen: Der LocationRetriever greift nicht auf andere Module zu und implementiert keine Observerschnittstellen.

Risikoanalyse: Der Einsatz der Positionsbestimmungs-Technologie birgt einige Risiken. Es müssen die passenden Bibliotheken gefunden, eingebunden, und ihr Zusammenspiel mit dem System untersucht werden.

MessageManager Dieses Modul verwaltet die ein- und ausgehenden Post-Nachrichten des Nutzer. Es definiert dazu ein Nachrichtenobjekt, welches alle relevanten Informationen einer Nachricht enthält.

Schnittstellen: Über die Schnittstelle des Moduls können die empfangen und gesendeten Nachrichten referenziert werden. Außerdem können empfangene Nachrichten hinzugefügt, und alte Nachrichten entfernt werden. Über die Observerschnittstelle können sich andere Module über eingehende oder gesendete Nachrichten informieren lassen.

Kommunikationsbeziehungen: Die graphische Nutzungsschnittstelle ist Observer auf dem MessageManager, um den Nutzer über ungelesene Nachrichten informieren zu können. Sie nutzt außerdem die öffentliche Schnittstelle, um die Nachrichten anzeigen und darstellen zu können.

ModalityManager Dieses Modul gibt auftretende Ereignisse und Zustände aus. Dabei orientiert sich das Modul am aktuellen Kontext und den Präferenzen des Nutzers, um die passenden Darstellungsmodalitäten für die Ereignisse/Zuständen auszuwählen.

Schnittstellen: Über die öffentliche Schnittstelle können Ereignisse und Zustände in der dem Kontext angemessenen und den Präferenzen der Nutzers entsprechenden Modalitäten ausgegeben werden.

Kommunikationsbeziehungen: Der ModalityManager überwacht den ContextManager und die UserInterfaceConfiguration, um die bei Änderungen zur Ausgabe von Ereignissen und Zuständen zu verwendenden Modalitäten neu zu berechnen.

NetworkManager Dieses Modul verwaltet die Verbindungen zwischen Clients und Servern und organisiert die Serialisierung der Nachrichten, die zwischen Client und Server ausgetauscht werden.

Schnittstellen: Über die öffentliche Schnittstelle werden die zu versendenden Nachrichten aufgenommen. Über die Observerschnittstelle kann der Status der Verbindung zum Server überwacht werden.

Kommunikationsbeziehungen: Die eingehenden Nachrichten werden in der ClientApplication gesammelt und von dort aus weiterverarbeitet. StateManager, KontextManager und die GameLogic überwachten mittels der Observerschnittstelle den Status der Verbindung zum Server.

Risikoanalyse: Die Technologie zum Herstellen von Internetverbindungen zwischen mobilem Endgerät und einem Server ist mit Risiko behaftet, weil die Technologie auf sehr komplexen Vorgängen basiert. Eine Machbarkeitsstudie sollte versuchen, Risiken frühzeitig aufzuspüren.

StateManager Dieses Modul speichert und aktualisiert die in Abschnitt 13.2.3 definierten Programmzustände der Spieler. Über eingehende Ereignisse werden Zustandswechsel herbeigeführt. Das Modul dient auf Client-Seite vor allem zur Berechnung des Kontexts und der Darstellung der grafischen Nutzungsschnittstelle, wird aber auch zur Berechnung der Gültigkeit von Spielaktionen benötigt.

Schnittstellen: Über die öffentliche Schnittstelle können die Programmzustände der Spieler referenziert werden. Des Weiteren können auftretende Events übergeben werden, aus denen der StateManager eventuelle Änderungen der Programmzustände berechnet. Über die Observerschnittstelle können die Änderungen der Zustände verfolgt werden.

Kommunikationsbeziehungen: Die Application übergibt die aufgetretenen Events an das Modul weiter, um die Programmzustände der Spieler immer auf dem neusten Stand zu halten. Die GameLogic fragt die Programmzustände der Spieler ab, um spielrelevante Entscheidungen zu treffen. Der StateManager implementiert die PointsOfInterestObserver-Schnittstelle, da auch Positionsänderungen zu Veränderungen von Programmzuständen führen können.

UserProfileManager Dieses Modul speichert und verwaltet die Profile der am Spiel teilnehmenden Spieler. Die Profile beinhalten alle Daten, wie Name des Spielers, Verteilung der Fähigkeitspunkte und aufgenommene virtuelle Gegenstände.

Schnittstellen: Über die öffentliche Schnittstelle können Spielerprofile angelegt, verändert oder gelöscht werden. Die Observerschnittstelle teilt den registrierten Observern diese Änderungen mit.

Kommunikationsbeziehungen: Das GameLogic Modul benutzt den UserProfileManager, um die Profildaten der Spieler zu verwalten.

13.4.5 Verhaltenssicht

Die Verhaltenssicht geht auf die dynamischen Aspekte des Innenlebens der beschriebenen Bausteine ein. Sie beschreibt unter anderem, wie eine Aktion oder Teilaktion mit Hilfe der in der Struktursicht beschriebenen Bausteine realisiert wird. Für diese Sicht sind die Architekturbausteine, deren Interaktionen und eventuelle Zustandsbeschreibungen relevant [PBG04].

Die dynamischen Aspekte der Architektur sind in den folgenden Unterabschnitten nach Themengebiet gruppiert. Die Sequenzdiagramme, die die zeitlichen Abläufe der Anwendung darstellen, legen den Fokus auf die Kommunikation *zwischen* den Bausteinen. Von der Kommunikation innerhalb der Anwendung wird abstrahiert. Ist ein Modul in einem Sequenzdiagramm mit einem grauen Kasten hinterlegt, existiert eine Verfeinerung des Ablaufs. Das verfeinernde Diagramm ist entweder Teil der Verhaltenssicht, oder es befindet sich im Feinentwurf-Abschnitt. Die genaue Seitenzahl und die Nummer der Abbildung sind dann in der Beschreibung vermerkt. In manchen Diagrammen wird die Kommunikation zwischen Modulen aus Client und Server beschrieben. Client- und Serverseite werden dann durch eine beschriftete Markierung voneinander getrennt. Werden Nachrichten über das Netzwerk versendet, wird das durch gestrichelte Datenflusspfeile gekennzeichnet.

13.4.5.1 Initialisierung und Lifecycle-Steuerung der Anwendung

Dieser Abschnitt visualisiert und beschreibt die dynamischen Aspekte von der Initialisierung und der Lifecycle-Steuerung der Anwendung. Abbildung 13.20 (S.281) zeigt den exemplarischen Ablauf der Initialisierung der Klasse ClientApplication/ServerApplication, die die Hauptklassen von Client und Server repräsentieren.

Wird eine neue Instanz von Client-/ServerApplication erzeugt, initialisiert sie zunächst alle Module. Sind alle Module instanziiert, registrieren sie sich zu ihren Subjekten, d.h. den Modulen, bei denen sie als Observer auf Änderungen achten sollen. Danach kann die Anwendung gestartet werden.

Abbildung 13.21 (S.281) visualisiert den Startvorgang der Anwendung. Dabei werden die eventuell vorhandenen internen Threads der Module und der Application gestartet.

Der Aufruf von `Start()` auf der Client-/ServerApplication wird an die in `Client/-ServerModule` definierte Methode `Start()` aller Module weitergereicht. Nachdem alle Module gestartet wurden, wird die Methode `Run()` der Client-/ServerApplication in einem eigenen Thread gestartet. Diese beginnt mit der zyklischen Abarbeitung aller in der Warteschlange eingehenden Events.

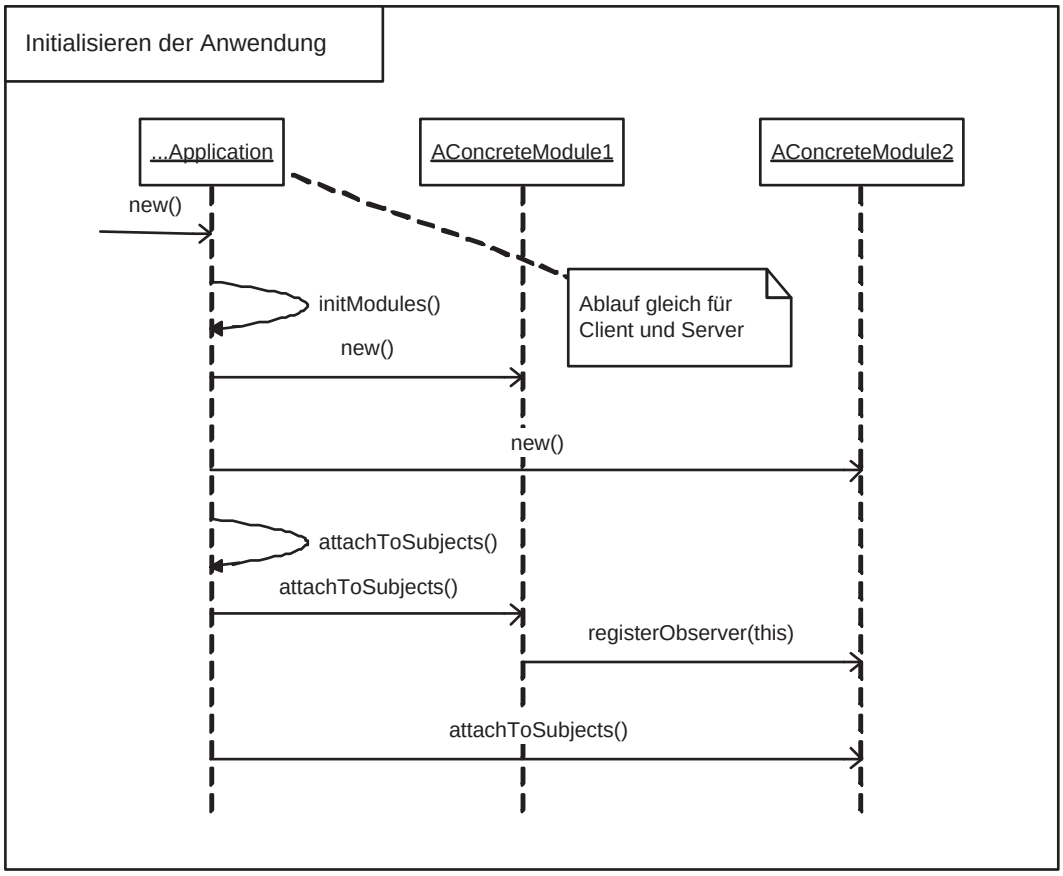


Abbildung 13.20: Exemplarisches Initialisieren der Anwendung

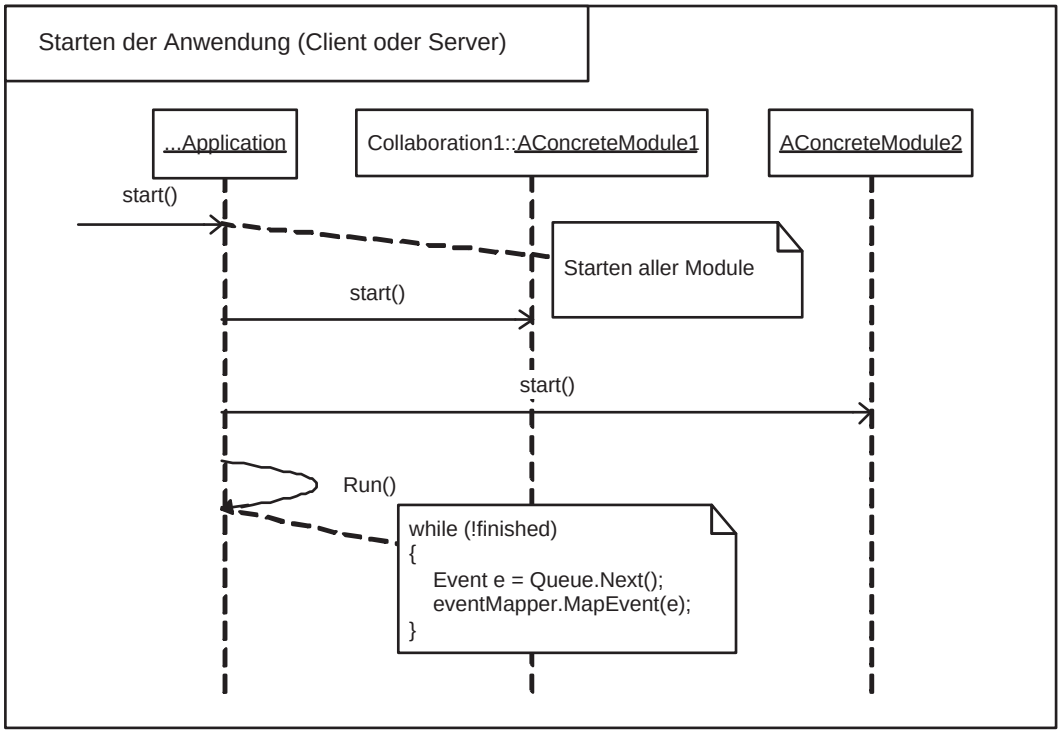


Abbildung 13.21: Exemplarisches Starten der Anwendung

13.4.5.2 Ablauf der Anmeldung

Dieser Abschnitt fast die Abläufe bezüglich der Anmeldung eines Clients beim Server zusammen. Abbildung 13.22 (S.282) zeigt die Übertragung der Anmeldedaten vom Client zum Server.

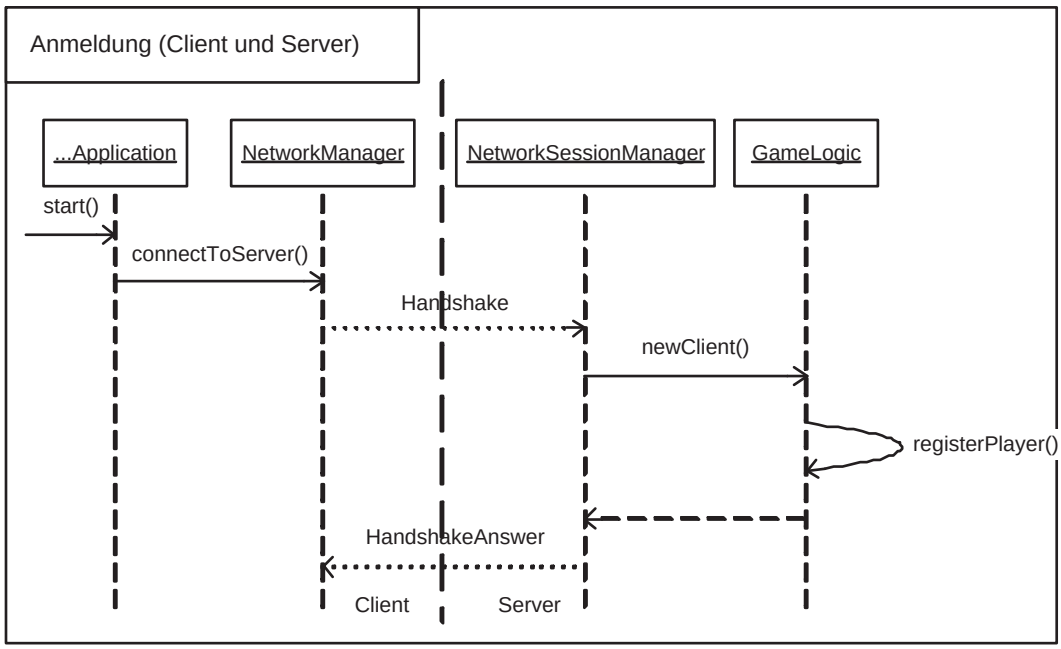


Abbildung 13.22: Ablauf der Anmeldung

Wird die Anwendung gestartet, weist sie den NetworkManager an, sich mit dem Server zu verbinden. Zu diesem Zeitpunkt muss auf Client-Seite bereits das vollständige Spielerprofil, allerdings ohne die Spieler-ID, erstellt worden sein. Das Spielerprofil wird dann während des Handshakes an den Server verschickt und dort der GameLogic übergeben. Dort wird die Anmeldung durchgeführt und als Antwort die gültige Spieler-ID bzw. eine Fehlermeldung zurückgegeben. Der Handshake wird mit der Antwortnachricht des Servers abgeschlossen.

13.4.5.3 Übertragung von Positionsänderungen

Dieser Abschnitt beschreibt den Ablauf der Übertragung einer Positionsänderung vom Client zum Server und innerhalb der beiden Systeme. In Abbildung 13.23 (S.283) ist die Übertragung einer neu bestimmten Position zum Server visualisiert.

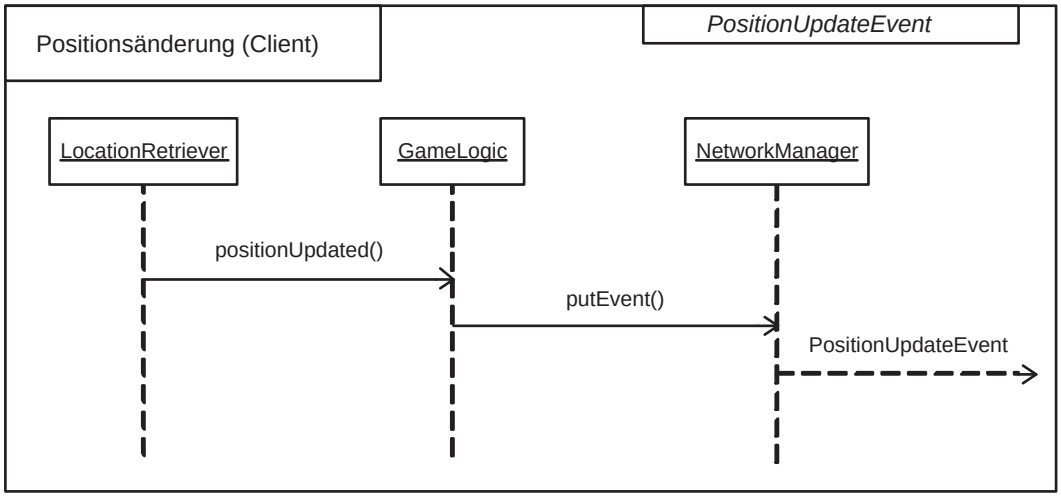


Abbildung 13.23: Übertragung der Positionsänderung im Client

Stellt der LocationRetriever eine Änderung der Position fest, teilt sie dies allen registrierten Observern über die Observerschnittstelle mit. In Abbildung 13.23 wurden die anderen Observer aus Gründen der Übersichtlichkeit ausgeblendet. Der relevante Observer ist in diesem Fall die GameLogic. Diese generiert ein PositionUpdateEvent und stellt es in die Warteschlange der Netzwerkschicht, so dass es an den Server verschickt wird.

Abbildung 13.24 (S.284) zeigt, wie der Server auf das eingehende PositionUpdateEvent, dass die geänderte Position transportiert, reagiert. Es setzt in dem Moment an, wenn das in Abbildung 13.23 (S.283) generierte PositionUpdateEvent vom NetworkManager deserialisiert wurde.

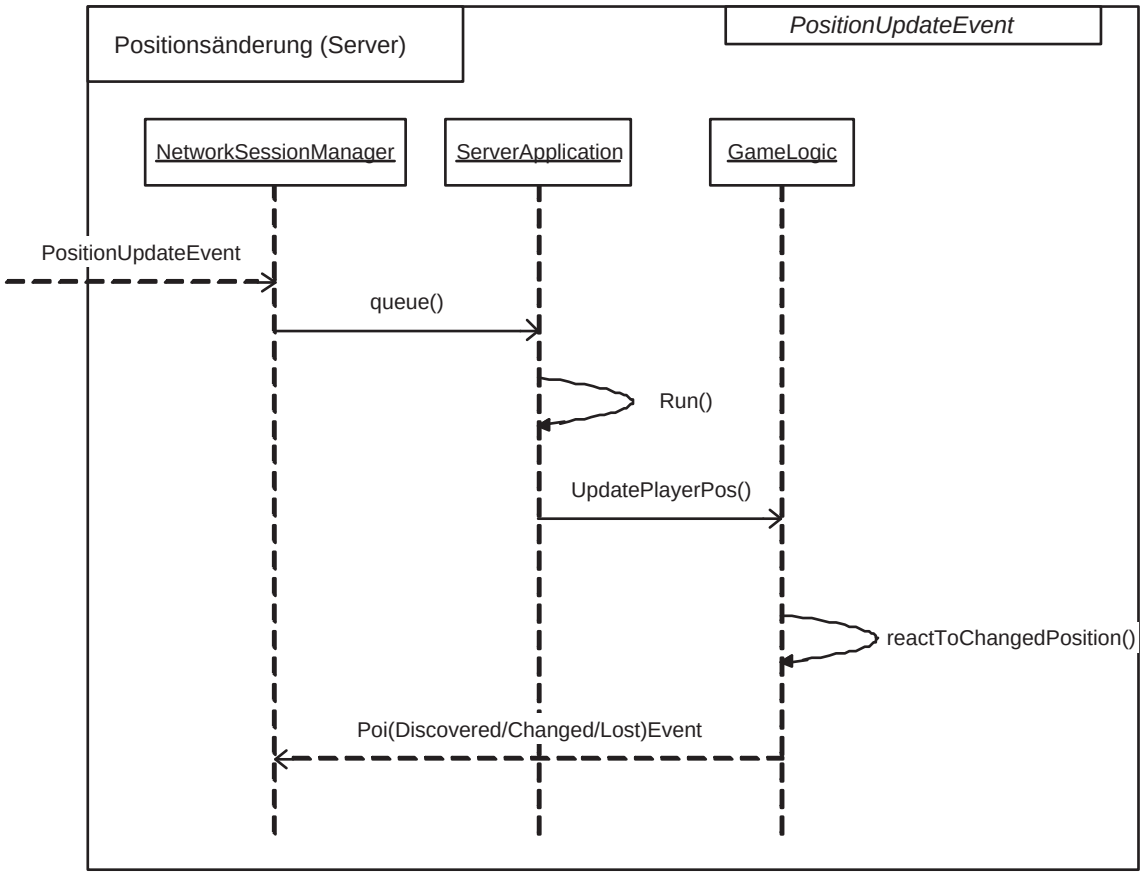


Abbildung 13.24: Übertragung der Positionsänderung im Server

Zunächst wird das PositionUpdateEvent in die Warteschlange der ServerApplication gelegt. In dem Thread der ServerApplication, wird es dann in der Methode `Run()` aus der Warteschlange ausgelesen und daraus ein Methodenaufruf auf die GameLogic abgeleitet. Die GameLogic gibt die Positionsänderung an das PointsOfInterest Modul weiter. Dort wird geprüft, ob der Spieler sich in der Nähe anderer Points-of-Interest befindet, oder durch diese Positionsänderung in deren Nähe geraten ist, bzw. sie verlassen hat. Diese Annäherungen werden an die GameLogic weitergegeben, so dass dort unter Berücksichtigung der Spielregeln entschieden werden kann, ob z.B. einem Spieler die Position eines gegnerischen Spielers anzeigt werden soll. Soll einem Spieler die Position (bzw. deren Änderung) eines Points-of-Interest angezeigt werden, generiert die GameLogic entsprechende Events und stellt sie in die Warteschlange für ausgehende Ereignisse des Servers.

13.4.5.4 Verhalten der Grafischen Nutzungsschnittstelle

Der Aufbau der Nutzungsschnittstelle orientiert sich am Model-View-Controller (MVC) Muster (siehe Abb. 13.25) aufgebaut. MVC ist ein bekanntes und erfolgreiches Muster, das zur Trennung von Fachobjekten und deren Darstellung eingesetzt wird. Es handelt sich dabei um eine Verallgemeinerung des Observer Musters.

Die Nutzungsschnittstelle setzt das Model-View-Controller Muster nur zum Teil um, da Control und View in der Standardbibliothek des .NET Compact Framework für graphische Nutzungsschnittstellen nicht getrennt sind. Beispiel dafür ist der Button, der sowohl eine graphische Repräsentation darstellt, aber auch Nutzereingaben zulässt.

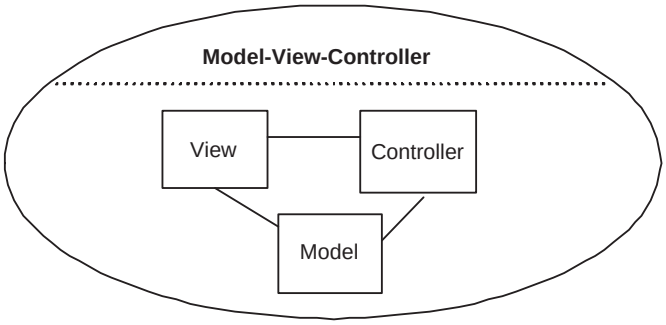


Abbildung 13.25: Model-View-Controller Muster

Die Nutzungsschnittstelle (View) des Clients stellt nur Daten (Model) dar, die über die Observerschnittstellen anderer Module empfangen wurden. Alle Eingaben (Controls) werden ausschliesslich über eine Fassade an die Module der „Logic-Schicht“ weitergegeben. Die daraus resultierende Entkopplung von Nutzungsschnittstelle und Daten/Logik macht sie sehr flexibel gegenüber Änderungen an der Darstellung. Anhand der Beschreibung vom Einsatz einer Funktionalität (13.4.5.5) kann man die Umsetzung des Model-View-Controller Musters an einem Beispiel nachvollziehen. Deshalb kann in diesem Abschnitt auf ein Beispiel verzichtet werden.

13.4.5.5 Einsetzen einer Funktionalität am Beispiel von Tarnen

Dieser Abschnitt erläutert, wie sich die Architektur beim Einsatz einer Spielfunktionalität verhält. Als Beispielfunktionalität wurde *Tarnen* ausgesucht, da diese Funktionalität keine Besonderheiten aufweist, und das Verhalten so am ehesten auf die anderen Funktionalitäten übertragen werden kann. Abbildung 13.26 (S.286) zeigt, wie die Eingabe des Spielers auf Seite des Clients verarbeitet wird.

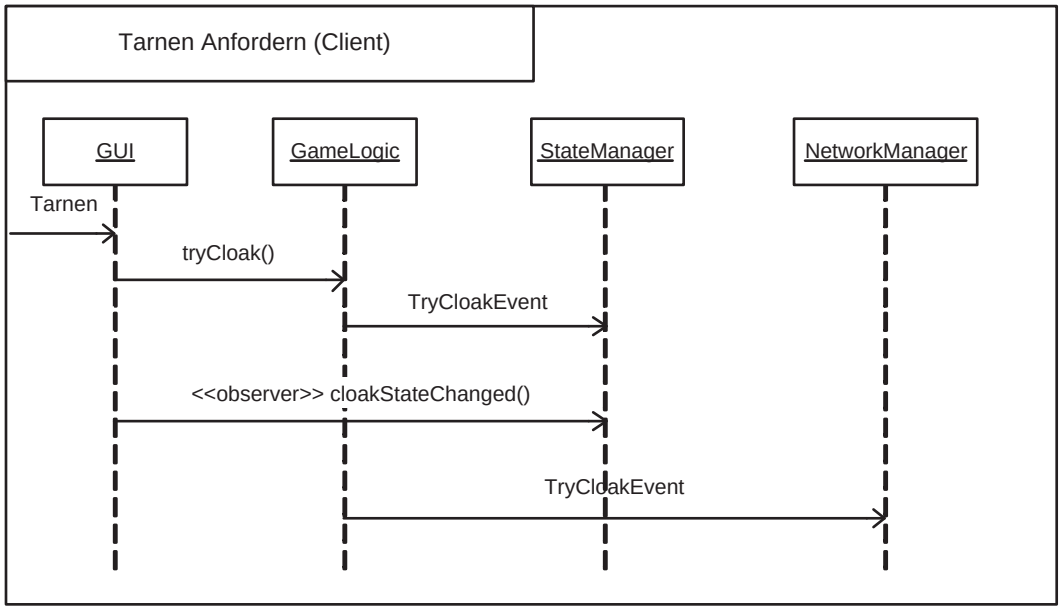


Abbildung 13.26: Client fordert an, sich zu tarnen

Aktiviert der Nutzer ein Eingabeelement, das mit der Funktionalität *Tarnen* in Verbindung steht, empfängt die GUI ein entsprechendes Ereignis. Sie leitet die Anfrage des Nutzers an die GameLogic weiter. Dort kann nun bereits überprüft werden, ob der Spieler sich nach Kenntnis der GameLogic überhaupt tarnen darf. Dieses Beispiel geht davon aus, dass die GameLogic keinen Grund erkennt, warum sich der Spieler nicht tarnen dürfte und erzeugt ein TryCloakEvent. Dieses wird an den StateManager und den NetworkManager weitergeleitet. Der StateManager merkt sich, dass der Spieler einen Versuch unternommen hat, sich zu tarnen, indem er einen Zustandswechsel der Tarnfunktionalität herbeiführt. Via Observer teilt er dies der GUI mit, so dass sie weitere Eingaben durch Eingabeelemente, die mit der Funktionalität *Tarnen* in Verbindung stehen, ignorieren kann. Der NetworkManager versendet das TryCloakEvent an den Server.

Abbildung 13.27 (S.287) zeigt, wie das TryCloakEvent auf Seite des Servers verarbeitet wird.

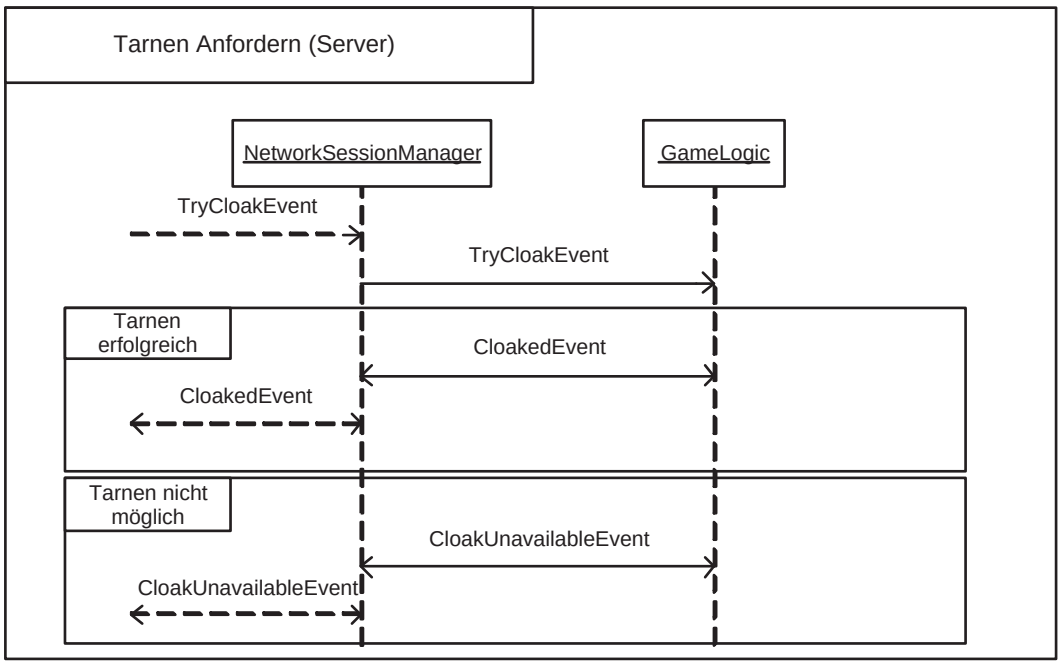


Abbildung 13.27: Server bearbeitet Anforderung, sich zu tarnen

Zunächst wird das TryCloakEvent an die GameLogic weitergereicht. Dort wird überprüft, ob der Spieler sich überhaupt tarnen darf. Dazu wird auch auf den StateManager zurückgegriffen, da dieser den Zustand der Tarnfunktionalität speichert. Darf der Spieler sich tarnen, generiert die GameLogic ein CloakedEvent und sendet es via NetworkManager an den Client zurück. Darf sich der Spieler nicht tarnen, wird stattdessen ein CloakUnavailableEvent erzeugt. Beide Events werden außerdem an den StateManager weitergereicht, damit dieser den korrekten Zustand für die Tarnfunktionalität einnehmen kann.

Abbildung 13.28 (S.288) zeigt, wie das CloakedEvent bzw. das CloakUnavailableEvent auf Seite des Clients verarbeitet wird.

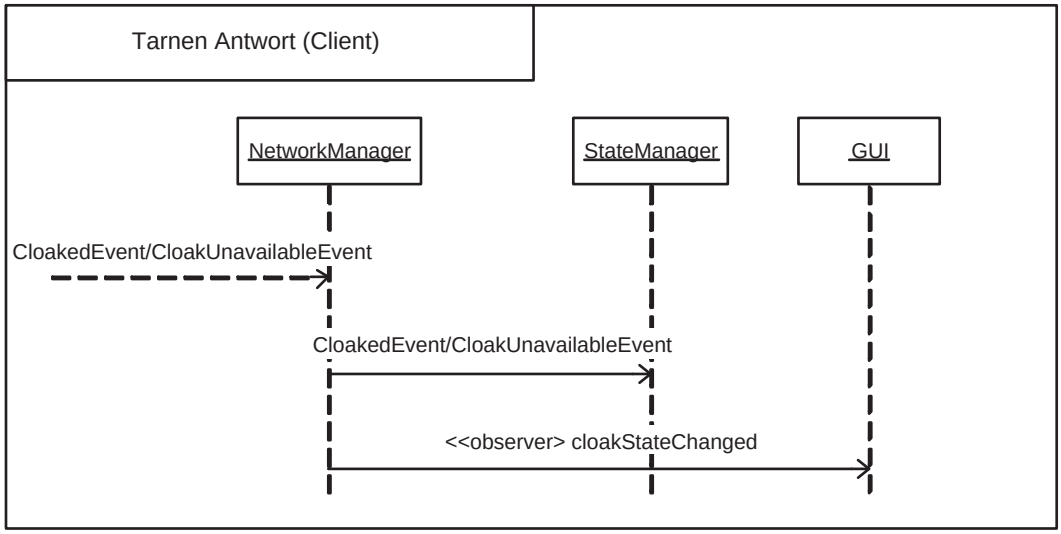


Abbildung 13.28: Verarbeitung der Antwort des Servers

Das CloakedEvent bzw. das CloakUnavailableEvent wird an den StateManager weitergereicht, der darauf hin den korrekten Zustand für die Tarnfunktionalität einnimmt. Über die Observerschnittstelle wird die Zustandsänderung zurück an die GUI geleitet und kann dort dem Nutzer präsentiert werden.

13.4.5.6 Zusammenspiel Kontext/Ausgabemodalitäten

Dieser Abschnitt erläutert, wie Änderungen des Kontexts sich auf die Wahl der Ausgabemodalitäten zur Präsentation von Ereignissen und Zuständen auswirken. Abbildung 13.29 (S.289) zeigt, wie sich in die Änderung der Bewegungsgeschwindigkeit des Spielers, die Teil des berücksichtigten Kontexts ist, auf die Wahl der Ausgabemodalitäten auswirkt.

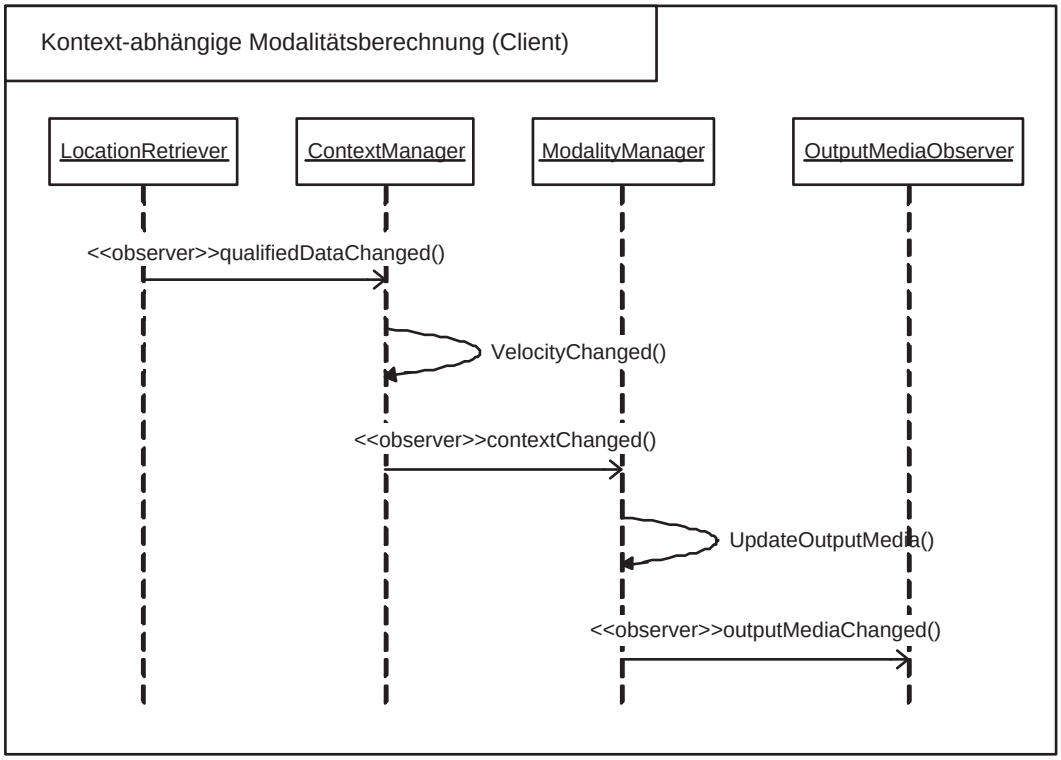


Abbildung 13.29: Zusammenspiel Kontext/Ausgabemodalitäten

Das `LocationRetriever` Modul hat eine Änderung der Bewegungsgeschwindigkeit festgestellt. Diese ist Teil der qualifizierten Positionsdaten. Darum wird die Methode `qualifiedDataChanged()` aller `LocationObserver` aufgerufen. Der `ContextManager` implementiert den `LocationObserver` und reagiert nun auf die Änderung der Bewegungsgeschwindigkeit. Er stellt nun z.B. fest, dass sich der Kontext des Spielers von *stehend* nach *bewegend* geändert hat und informiert die `ContextObserver` implementierenden Module darüber, via der Methode `contextChanged()`. Der `ModalityManager` wird als entsprechender Observer über die Änderung des Kontexts informiert und startet nun die Berechnung der für den Kontext angemessenen Ausgabemodalitäten neu. In diesem Beispiel könnte er nun z.B. feststellen, dass der Spieler sich bewegt und sich folglich nicht mehr so gut auf visuelle Ausgaben konzentrieren kann. Der `ModalityManager` würde nun verstärkt akustische und haptische Signale zur Ausgabe von Ereignissen und Zuständen einsetzen.

13.4.6 Abbildungssicht

Diese Sicht beschreibt sowohl die technischen Abbildungen, etwa die Zuordnung von Artefakten wie z.B. einer DLL oder eine Enterprise Java Bean (EJB) zu Architekturbausteinen, als auch die nichttechnischen Aspekte, wie die Aufteilung von Arbeitspaketen auf die Teammitglieder. Artefakte stehen dabei für die physikalische Realisierung der Architekturbausteine. Bei dieser Sicht werden organisatorische Aspekte der Software in den Vordergrund gestellt. Für diese Sicht sind die Ausführungseinheiten (z.B. Rechner, Steuergeräte, etc.), die Artefakte (z.B. DLL, EJB, etc.) und die Pakete relevant [PBG04].

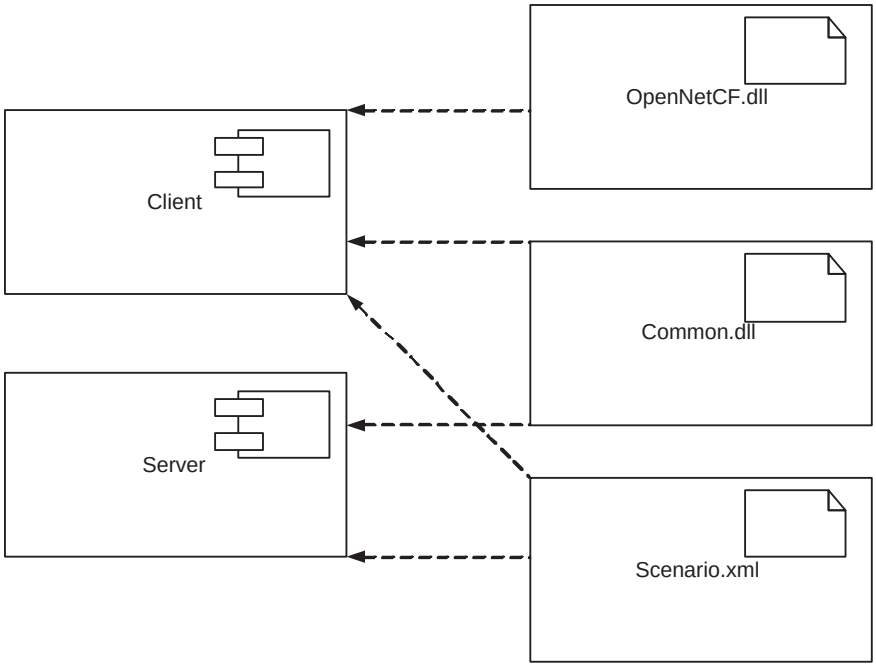


Abbildung 13.30: Realisierung von Client und Server durch Artefakte

Abbildung 13.30 zeigt, welche Artefakte, in diesem Fall Bibliotheken und Konfigurationsdateien, zum Erzeugen von Client und Server verwendet wurden. *OpenNetCF.dll* steht für die Bibliotheken des OpenNet Compact Frameworks, einer freien Bibliothek von Drittanbietern. *Common.dll* enthält alle von Client und Server gemeinsam verwendeten Datenstrukturen und Funktionen. *Scenario.xml* enthält die Beschreibung des Szenarios im XML-Format.

13.5 Nutzungsschnittstelle

In diesem Kapitel wird der Entwurf der Nutzungsschnittstelle von NABB vorgestellt. Diese muss eine multimodale Nutzerinteraktion erlauben und sich bei der Auswahl der Modalitäten für Ein- und Ausgabe von Informationen am jeweiligen Kontext des Nutzers orientieren. Als Grundlage für die Identifizierung des Kontexts dienen hierbei die bereits vorgestellten Programmzustände (vgl. Kapitel 13.2 auf Seite 246).

13.5.1 Multimodalität

Im Bereich der Mensch-Computer-Interaktion (Human-Comuputer-Interaction (HCI)) spielt die Berücksichtigung der Multimodalität eine entscheidende Rolle. Speziell bei der Entwicklung von Nutzungsschnittstellen wird heute angestrebt, die Interaktion zwischen Mensch und Maschine über mehrere Wahrnehmungskanäle zu realisieren. In diesem Abschnitt wird beschrieben, welche dieser Wahrnehmungskanäle bei der Entwicklung der multimodalen Nutzungsschnittstelle für NABB Berücksichtigung finden sollen.
Der Mensch nimmt seine Umwelt über seine Sinne wahr. Zu diesen zählen der visuelle (sehen), der auditive (hören), der olfaktorische (riechen), der gustatorische (schmecken) und der haptische Sinn. Eine Nutzungsschnittstelle, die den Informationsaustausch zwischen Mensch und Maschine über mehr als einen dieser Sinne ermöglicht, wird als multimodal bezeichnet.
Im Anwendungskontext von NABB ist die multimodale Kommunikation auf die von den mobilen Endgeräten ansprechbaren Ein- und Ausgabekanäle beschränkt. Nach eingehender Untersuchung der zur Verfügung stehenden mobilen Endgeräte (siehe Kapitel 12.4 auf Seite 214) stehen somit noch der visuelle, der akustische und der haptische Kanal zur Verfügung. Eingaben können akustisch und haptisch und Ausgaben visuell, akustisch und haptisch erfolgen. Aus diesem Grund werden der gustatorische und der olfaktorische Kanal in den folgenden Betrachtungen außer acht gelassen.
Eine Aufgabe der multimodalen Nutzungsschnittstelle von NABB besteht darin, den Spieler über den aktuellen Spielzustand zu informieren. Überlegungen, wie dies in den unterschiedlichen Modalitäten zu realisieren ist, werden weiter unten beschrieben (siehe 13.5.2 auf Seite 292). Darüber hinaus stellt sich die Frage, wie viele verschiedene Zustände ein Mensch in den verschiedenen Modalitäten unterscheiden kann. Im Folgenden werden die zur Verfügung stehenden Modalitäten hinsichtlich dieses Kriteriums untersucht.

- **visueller Kanal**: Der Mensch ist in der Lage visuelle Informationen selektiv wahrzunehmen. Darum können theoretisch beliebig viele Teilaspekte eines Zustands gleichzeitig vom System dargestellt werden. Es ist jedoch nicht garantiert, dass der Benutzer sie auch wahrnimmt. Der Sehsinn ist bei den meisten Menschen der am weitesten entwickelte Sinn und wird deshalb bei der Mensch-Computer-Interaktion primär zur Übermittlung von Informationen benutzt.

- **akustischer Kanal**: Bei gleichzeitig erklingenden akustischen Signalen kann der Mensch nur eine begrenzte Anzahl auseinanderhalten bzw. identifizieren. Das liegt daran, dass es dem Menschen schwieriger fällt, sich speziell auf einzelne Klänge zu konzentrieren, da der Hörsinn oft weniger trainiert ist, als der Sehsinn. Spezielle Gruppen wie z.B. sehbehinderte Menschen können aber eventuell genügend trainiert sein, um deutlich mehr Zustände unterscheiden zu können. Ein weiteres Problem ist, dass man akustische Signale nicht physikalisch ausblenden kann, wie man es mit Teilen einer grafischen Anzeige durch Verändern des Sichtfeldes tun kann. Es muss also darauf geachtet werden, dass der akustische Kanal nicht überladen wird.

- **haptischer Kanal**: Das Potential des Tastsinns liegt darin, dass er im Gegensatz zum Seh- oder Hörsinn nicht über einen „Punkt“ wie die Augen oder die Ohren, sondern über die gesamte Körperoberfläche des Menschen wahrgenommen wird. Informationen können also nicht nur durch Stärke,

Richtung und Art der Kontaktoberfläche einer Berührung, sondern auch durch die Stelle auf der *Benutzeroberfläche* unterschieden werden. Im Gegensatz zu Akustik und Optik gibt es bei der Haptik keine Sprache. Es gibt auch viel weniger allgemeingültige Symbole, was die gezielte Übermittlung von Informationen schwieriger macht. Haptische Signale kann der Mensch wie akustische Signale nicht physikalisch ausblenden.

Bei der Darstellung von Zuständen über den visuellen, akustischen oder haptischen Kanal, sollten die Gestaltgesetze beachtet werden, um Fehlinterpretationen der dargestellten Informationen vorzubeugen. Die Gestaltungsgesetze betrachten zwar vornehmlich visuell aufgenommene Informationen, sind aber laut [Hed02] nicht ausschließlich auf diese beschränkt. Zu den Gestaltungsgesetzen zählen:

- **Gesetz der Nähe**: nahe beieinander liegende Gegenstände werden als zusammengehörig empfunden
- **Gesetz der Gleichartigkeit**: Elemente mit gleichen oder ähnlichen Eigenschaften (Farbe, Form) werden als Gruppe wahrgenommen
- **Gesetz der guten Gestalt bzw. Fortsetzung**: schneidende Linien werden bevorzugt in Weiterführung ihres ursprünglichen Verlaufs gesehen
- **Gesetz der Geschlossenheit:** geschlossene Linienzüge werden als strukturelle Einheiten gesehen
- **Gesetz des gemeinsamen Schicksals:** gemeinsam, gleichartige Veränderungen erfahrende Elemente werden als Gruppe aufgefasst

13.5.2 Multimodale Ausgabe

In diesem Abschnitt wird beschrieben, auf welche Arte und Weise die multimodale Ausgabe von Informationen auf dem NABB-Client erfolgen soll. Dabei wird zwischen der Ausgabe von Zuständen und Ereignissen unterschieden, wie sie bereits im Kapitel 13.2 auf Seite 246 beschrieben wurde.
Da Zustandswechsel nicht innerhalb festgelegter Zeitintervalle auftreten, sondern nur durch Ereignisse hervorgerufen werden können, erfolgt die Ausgabe eines Zustands so lange, wie sich der Client in diesem befindet. Ein Ereignis hingegen tritt nur kurzzeitig auf und wird daher auch lediglich zum Zeitpunkt seines Auftretens durch eine Ausgabe signalisiert.
Die einzelnen Zustände und Ereignisse werden gemäß ihrer Spielrelevanz priorisiert. Auf diese Weise kann sichergestellt werden, dass sowohl Zustände als auch Ereignisse höherer Priorität stets vor Zuständen, resp. Ereignissen niedrigerer Priorität ausgegeben werden. Darüber hinaus wird die Ausgabe von Ereignissen jederzeit der Ausgabe von Zuständen vorgezogen.

13.5.2.1 Multimodale Darstellung von Zustände

In der Beschreibung des *fachlichen Modells* (siehe 13.2 auf Seite 246) wurden einzelne Zustände von NABB bereits identifiziert. In diesem Abschnitt wird beschrieben, durch welche Ausgaben dem Spieler die aktuellen Zustände repräsentiert werden sollen. Hierzu stehen verschiedene Ausgabemöglichkeiten zur Verfügung, deren Verwendung im Folgenden beschrieben wird.
Aus der Beschreibung des *fachlichen Modells* geht hervor, dass sich der aktuelle Spielzustand von NABB zu jedem Zeitpunkt aus einer Reihe aktiver Zustände zusammensetzt. Eine Ausgabe aller dieser Zustände würde die Aufnahmefähigkeit eines Spielers allerdings weit überschreiten.

Um dem entgegen zu wirken, wird für einige Zustandsautomaten (vgl. 13.2 auf Seite 246) ein Standardzustand festgelegt, von dessen Aktivität der Spieler so lange ausgehen kann, bis er explizit auf eine Abweichung des Zustands von diesem Standardwert hingewiesen wird. So wird bspw. im Automat *Spielzustand* der Zustand *Aktiv* als Standardzustand angenommen und dies dem Benutzer nicht explizit über Symbole, Geräusche oder Vibrationen, sondern implizit über ein Ausbleiben solcher Ausgaben signalisiert. Ändert sich der Zustand in *Passiv* oder *Spielende*, so wird dies explizit dargestellt.
Da die gleichzeitige Darstellung mehrerer Zustände oder Ereignisse über den gleichen Ausgabekanal von Seiten der Hardware nur schwer realisierbar und die so übermittelten Informationen vom Spieler kaum zu verarbeiten sind, werden den einzelnen Zuständen und Ereignissen Prioritäten zugeordnet. Unterschieden wird zwischen den vier Prioritätsstufen „0“, „+“, „++“ und „+++“, wobei „0“ die niedrigste und „+++“ die höchste Priorität repräsentiert. Im Falle, dass zwei Zustände zum gleichen Zeitpunkt durch die gleiche Ausgabemodalität dargestellt werden sollen, entscheidet die Anwendung zugunsten des Zustands mit der höheren Priorität. Analog wird bei Ereignissen verfahren (vgl. 13.5.2.2 auf Seite 298).
Die Darstellung eines Zustands kann gleichzeitig visuell, akustisch und haptisch erfolgen. Es hat sich allerdings gezeigt, dass dies bei Spielern zu einer Reizüberflutung führen kann. Aus diesem Grund hat der Spieler in der Kontextansicht (vgl. 13.5.4.5 auf Seite 307) die Möglichkeit, die zu verwendenden Ausgabemodalitäten selbst festzulegen.
Eine Auflistung möglicher Ausgaben für die einzelnen Zustände ist in Tabelle 13.1 auf Seite 294 und in Tabelle 13.2 auf Seite 295 zu sehen. Während Tabelle 13.1 hauptsächlich Zustände beschreibt, die den Umgebungskontext des Spielers betreffen, geht es in Tabelle 13.2 um Zustände, die den Status der Funktionen Tarnen, Neutralisieren, Abhören und Aufklären beschreiben. In beiden Tabellen werden mögliche Ausgaben in den Modalitäten *Visuell*, *Akustisch* und *Haptisch* vorgeschlagen. Die visuelle Ausgabe gliedert sich in die Bereiche *Icon/Symbol*, *Anzeige* und *Textnachricht*. In der Spalte Icon/Symbol sind die visuellen Ausgaben aufgeführt, die dem Spieler unabhängig von der derzeitigen Ansicht der Nutzungsschnittstelle angezeigt werden. Die Spalte Anzeige enthält visuelle Ausgaben, die dem Spieler eventuell nur in bestimmten Ansichten der Nutzungsschnittstelle zur Verfügung stehen. In der Spalte Textnachricht stehen textuelle Ausgaben, die dem Spieler über den Nachrichtenticker oder über die Nachrichtenansicht angezeigt werden.
Bei der akustischen Ausgabe wird zwischen Signal und Sprache und bei der haptischen Ausgabe zwischen Vibrationsart und Vibrationscode unterschieden. Die Art der Vibration unterscheidet sich hierbei hinsichtlich der Intensität und der Frequenz. Die Wiedergabe höher priorisierter Zustände unterscheidet sich von der niedriger priorisierter Zustände, in der eher schubweisen Vibration und der Verwendung einer höheren Frequenz. Beim Vibrationscode handelt es sich um eine Art Morsecode. Ein Punkt repräsentiert eine kurze und ein Strich eine längere Vibration.

		Visuell			*Akustisch*		*Haptisch*	
Zustand	**Priorität**	**Icon/Symbol**	**Anzeige**	**Textnachricht**	**Signal**	**Sprache**	**Vibration**	**Vibrationscode**
Mitspieler nah	+	MAP-Tab hervorheben	Mitspielersymbol auf Karte hervorheben	Mitspieler in der Nähe, Interaktion möglich	ICQ-Klopfen	Mitspieler nah!	leicht, niedrige Frequenz, konstant	„. “
Gegenspieler nah	++	„Enemy-Symbol“ leuchtet	Symbol für Gegenspieler auf Karte einblenden	Achtung: Gegnerischer Spieler in der Nähe!	Warnton high	Achtung: Gegner!	mittel, niedrige Frequenz, schubweise	„-“
Goodie nah	+	MAP-Tab hervorheben	Goodie-Symbol auf Karte einblenden	Goodie erreicht!	„Interaktion möglich“-Ping	Goodie erreicht!	leicht, niedrige Frequenz, konstant	„.“
Missionsziel nah	+	MENU-Tab hervorheben	Symbol für Missionsziel auf Karte hervorheben	Missionspunkt erreicht!	„Interaktion möglich“-Ping	Missionspunkt erreicht!	leicht, niedrige Frequenz, konstant	„.“
eigene Kamera nah	+	MAP-Tab hervorheben	Kamerasymbol auf Karte hervorheben	Eigene Kamera erreicht!	„Interaktion möglich“-Ping	Eigene Kamera erreicht!	leicht, niedrige Frequenz, konstant	„.“
Gegenspieler in Aufklärungsradius der eigenen Kamera	++	MAP-Tab hervorheben	Gegenspielersymbol wird auf der Karte angezeigt	Gegenspieler entdeckt!	„U-Boot Sonar“ oder „Alien-Scanner“	Gegenspieler entdeckt!	mittel, niedrige Frequenz, schubweise	„-„
gegnerische Kamera nah	++	MAP-Tab hervorheben	Kamerasymbol auf Karte einblenden	Gegnerische Kamera entdeckt!	„Surren“ einer Kamera	Gegnerische Kamera entdeckt!	leicht, niedrige Frequenz, konstant	„.“
Spielfeldgrenze nah	++	MAP-Tab hervorheben	Spielfeldgrenze auf Karte hervorheben	Spielfeldgrenze nah!	Warnton low	Spielfeldgrenze nah!	mittel, hohe Frequenz, konstant	„- -“
Spielfeld verlassen	+++	MAP-Tab blinkt	Spielfeldgrenze auf Karte hervorheben	Sie haben das Spielfeld verlassen!	Warnton medium	Sie haben das Spielfeld verlassen!	mittel, hohe Frequenz, schubweise	„. - -“
Offline	+++	Fähigkeitsanzeigen verblassen, Verbindungsstatus-Symbol blinkt	sämtliche Symbole, außer eigener Position verschwinden von der Karte	Achtung: Keine Verbindung zum Server!	Warnton medium	Achtung, keine Verbindung zum Server	mittel, hohe Frequenz, schubweise	„. . .“
neue Post	+	MAIL-Tab hervorheben	neue Nachricht in Nachrichtenliste hervorheben	Neue Nachricht empfangen!	SMS-Signal	Sie haben Post!	leicht, niedrige Frequenz, 2x kurz hintereinander	„. .“

Tabelle 13.1: Multimodale Darstellung von Zuständen (Teil 1)

Zustand	Priorität	*Visuell* Icon/Symbol	Anzeige	Textnachricht	*Akustisch* Signal	Sprache	*Haptisch* Vibration	Vibrationscode
Tarnmodus nicht verfügbar	+	Füllstandsanzeige hervorheben	Füllstandsanzeige nicht voll	Tarnen nicht verfügbar!	„Düh“	Tarnen nicht verfügbar!	leicht, niedrige Frequenz, konstant	„. -“
Tarnmodus bereit	0	Füllstandsanzeige voll						
Tarnmodus aktiviert	+	Füllstandsanzeige nimmt ab	eigenes Symbol wird transparent auf der Karte dargestellt					
Tarnmodus lädt auf	0	Füllstandsanzeige nimmt zu						
Aufklärungskamera nicht verfügbar	+	Füllstandsanzeige hervorheben	Füllstandsanzeige leer	Aufklären nicht verfügbar!	„Düh“	Aufklären nicht verfügbar!	leicht, niedrige Frequenz, konstant	„. -“
Aufklärungskamera gesetzt	+	Füllstandsanzeige zeigt Anzahl verfügbarer Kameras an						
Aufklärungsmodus bereit	0	Füllstandsanzeige zeigt Anzahl verfügbarer Kameras an						
Abhörmodus nicht verfügbar	+	Füllstandsanzeige hervorheben	Füllstandsanzeige nicht voll	Abhören nicht verfügbar!	„Düh“	Abhören nicht verfügbar!	leicht, niedrige Frequenz, konstant	„. -“
Abhörmodus aktiviert	+	Füllstandsanzeige nimmt ab						
Abhörmodus bereit	0	Füllstandsanzeige voll						
Abhörmodus lädt auf	0	Füllstandsanzeige nimmt zu						
Neutralisationsmodus nicht verfügbar	+	Füllstandsanzeige hervorheben	Füllstandsanzeige nicht voll	Neutralisieren nicht verfügbar!	„Düh“	Neutralisieren nicht verfügbar!	leicht, niedrige Frequenz, konstant	„. -“
Neutralisationsmodus bereit	0	Füllstandsanzeige voll						
Neutralisationsmodus aktiviert	+	Füllstandsanzeige läuft ab						
Neutralisationsmodus lädt auf	0	Füllstandsanzeige nimmt zu						

Tabelle 13.2: Multimodale Darstellung von Zuständen (Teil 2)

Für einige Zustände ist lediglich die Ausgabe über ein Icon/Symbol vorgesehen, da weitere Ausgaben dieser Zustände den Spieler möglicherweise von wichtigeren Zustandsausgaben ablenken würden. Wie bereits beschrieben wurde, führt eine gleichzeitige Ausgabe aller Zustände über sämtliche Ausgabekanäle zu einer Reizüberflutung auf Seiten des Spielers. Daher ist für die Zustandsausgabe eine Standardeinstellung vorgesehen. Diese ist in Tabelle 13.3 auf Seite 297 zu sehen.
In der oberen Hälfte der Tabelle sind die Standardeinstellungen der wichtigsten Zustände aus den Tabellen 13.1 und 13.1 aufgeführt. Die untere Hälfte enthält Standardeinstellungen weiterer Zustände, die ebenfalls als spielrelevant angesehen werden. Zu den berücksichtigten Ausgabemodalitäten zählen:

- Icons/Symbole (**I**)
- Anzeigen (**A**)
- Textnachrichten (**T**)
- akustische Signale (**AS**)
- Sprachausgaben (**SA**)
- Vibration (**V**)

Die Wahl der zu verwendenden Ausgabemodalitäten ist von der augenblicklichen Ansicht der Nutzungsschnittstelle abhängig. Hierzu zählen die Kartenansicht, die Kontextansicht, die Menüansicht und die Nachrichtenansicht. Bei der Kartenansicht wird zusätzlich zwischen dem Spielzustand des Spielers (aktiv/passiv) unterschieden.

Zustand	Passive Umgebungserkundung	Aktive Umgebungserkundung	Kontextansicht	Menüansicht	Nachrichtenansicht
Mitspieler nah	-	AS,A	AS,V	AS,V	AS,V
Gegenspieler nah	-	AS,A,(V)	AS,V	AS,V	AS,V
Goodie nah	-	AS,I	AS	AS	AS
Missionsziel nah	-	AS,I	-	-	-
eigene Kamera nah	-	AS,I	AS,I	AS,I	AS,I
Gegner in Aufklärungsradius der eigenen Kamera	-	AS,V	AS,V	AS,V	AS,V
gegnerische Kamera nah	-	AS,I	AS	AS	AS
Offline	AS,V	AS,V	AS,V	AS,V	AS,V
neue Post	-	AS,I	AS,I	AS,I	AS,I
Spielfeldgrenze nah	-	A	SA	SA	SA
Spielfeld verlassen	SA	A	SA	SA	SA
Tarnen nicht verfügbar	-	I	I	I	I
Aufklären nicht verfügbar	-	I	-	-	-
Abhören nicht verfügbar	-	I	-	-	-
Neutralisieren nicht verfügbar	-	I	I	I	I
GPS nicht verfügbar	AS,V	AS,V	-	-	-
Startpunkt nah	AS,V	-	-	-	-
Startpunkt erreicht	A	-	-	-	-
Spielende	SA	V,T	SA,T	SA,T	SA,T
eigene Position	A	A	-	-	-

Tabelle 13.3: Standardeinstellung für multimodale Darstellung von Zuständen

13.5.2.2 Multimodale Darstellung von Ereignissen

In diesem Abschnitt werden die multimodalen Ausgabemöglichkeiten von Ereignissen, die während der Ausführung von NABB auftreten können beschrieben.
Wie bereits im letzten Abschnitt beschrieben wurde, treten Ereignisse nur kurzzeitig auf, was bei der Festlegung der möglichen Darstellung eines Ereignisses zu berücksichtigen ist. Die Ausgabe eines Ereignisses sollte daher nur einmal, bei dessen Eintritt erfolgen.
Analog zu Zuständen, müssen auch Ereignisse priorisiert werden, damit bei gleichzeitigem Eintreten mehrerer Ereignisse festgelegt werden kann, in welcher Reihenfolge die Ereignisse abzuarbeiten sind. Zu diesem Zweck wird, wie auch bei der Priorisierung von Zuständen, zwischen den vier Prioritätsstufen „0“, „+“, „++“ und „+++“ unterschieden, wobei „0“ die niedrigste und „+++“ die höchste Prioritätsstufe repräsentiert. Die Ausgabe eines Ereignisses ist dabei stets der Ausgabe eines Zustands vorzuziehen.
Die Darstellung eines Ereignisses kann visuell, haptisch und akustisch erfolgen (siehe Tabelle 13.4 auf Seite 299). Dabei wird bei der visuellen Ausgabe zwischen der Darstellung von Icons/Symbolen und Textnachrichten unterschieden. In der Spalte Icon/Symbol werden Ausgaben beschrieben, die dem Spieler, unabhängig von der aktuellen Ansicht der Nutzungsschnittstelle angezeigt werden können. Die Spalte Textnachrichten enthält textuelle Ausgaben, die dem Spieler über den Nachrichtenticker oder die Nachrichtenansicht angezeigt werden.
Die auditive Ausgabe eines Ereignisses kann als Signal und als Sprachausgabe erfolgen, während die haptische Ausgabe mittels Vibration erfolgt. Die Art der Vibration unterscheidet sich hierbei hinsichtlich der Intensität und der Frequenz. Die Wiedergabe höher priorisierter Ereignisse unterscheidet sich von der niedriger priorisierter Ereignisse, in der eher schubweisen Vibration und der Verwendung einer höheren Frequenz. Beim Vibrationscode handelt es sich um eine Art Morsecode. Ein Punkt repräsentiert eine kurze und ein Strich eine längere Vibration.
Die Nutzung mehrerer Ausgabekanäle zur Darstellung eintretender Ereignisse kann sehr schnell zu einer Reizüberflutung des Spielers führen. Aus diesem Grund soll der Darstellungsumfang einzelner Ereignisse standardmäßig eingeschränkt werden. Die Standardeinstellungen für die multimodale Darstellung von Ereignissen ist in Tabelle 13.5 auf Seite 300 aufgelistet. Zu den berücksichtigten Ausgabemodalitäten zählen:

- Icons/Symbole (**I**)
- Textnachrichten (**T**)
- akustische Signale (**AS**)
- Sprachausgaben (**SA**)
- Vibration (**V**)

Die Wahl der zu verwendenden Ausgabemodalitäten ist von der augenblicklichen Ansicht der Nutzungsschnittstelle abhängig. Hierzu zählen die Kartenansicht, die Kontextansicht, die Menüansicht und die Nachrichtenansicht. Bei der Kartenansicht wird zusätzlich zwischen dem Spielzustand des Spielers (aktiv/passiv) unterschieden.

		Visuell		*Akustisch*		*Haptisch*	
Ereignis	**Priorität**	**Icon/Symbol**	**Textnachricht**	**Signal**	**Sprache**	**Vibration**	**Vibrationscode**
Startpunkt nah	++	Fähigkeitsanzeige und Karte werden hervorgehoben	Sie haben den Startpunkt erreicht!	Computer hochfahren	Fähigkeiten werden aktiviert!	mittel, hohe Frequenz, schubweise	„.“
Goodie auf Karte	++	MAP-Tab hervorheben	Goodie entdeckt!	Ping	Goodie entdeckt!	leicht, niedrige Frequenz, konstant	„.“
Neutralisiert	+++	Fähigkeitsanzeige und Karte verblassen	Sie wurden neutralisiert!	Computer herunterfahren	Sie wurden neutralisiert!	mittel, hohe Frequenz, schubweise	„. . -“
Missionsziel erfüllt	++	Karte blinkt	Sie haben die Mission erfüllt!	Yippie	Mission erfüllt!	mittel, hohe Frequenz, schubweise	„- - -“
Missionsziel nicht erfüllt	++	Rotes Kreuz auf dem Bildschirm	Mission gescheitert!	verlorene Wette	Mission gescheitert!	mittel, hohe Frequenz, schubweise	„. . -“
Rundenende	+++	Fähigkeitspunkte und Karte verblassen	Spiel vorbei, Gewinner ist ...!	Musikthema	Spiel vorbei, Gewinner ist ...!	mittel, hohe Frequenz, schubweise	„. . -“
Rundenanfang	+++	Fähigkeitsanzeige und Karte werden hervorgehoben	Begeben Sie sich zu Ihrem Startpunkt!	Musikthema	Begeben Sie sich zu Ihrem Startpunkt!	mittel, hohe Frequenz, schubweise	„- - -“
neue Post	+	MAIL-Tab wird hervorgehoben	Sie haben Post!	SMS-Signal	Sie haben Post!	leicht, niedrige Frequenz, konstant	„. .“
Fähigkeit nicht verfügbar	+	Füllstandsanzeige hervorheben	Die Fähigkeit ist nicht verfügbar!	„Düh“	Fähigkeit ist nicht verfügbar!	leicht, niedrige Frequenz, konstant	„. -“
Kamera gesetzt	+	Füllstandsanzeige hervorheben	Kamera gesetzt!	„Schnappschuss“	Kamera gesetzt!	leicht, niedrige Frequenz, konstant	„- .“
Kamera eingesammelt	+	Füllstandsanzeige hervorheben	Kamera aufgenommen!	kurzer Ton	Kamera aufgenommen!	leicht, niedrige Frequenz, konstant	„. -“
Eigene Kamera deaktiviert	+	Füllstandsanzeige hervorheben	Eigene Kamera zerstört!	Geschepper	Eigene Kamera zerstört!	leicht, niedrige Frequenz, konstant	„. -“
gegnerische Kamera deaktiviert	+	Grünes Häkchen erscheint	Gegnerische Kamera zerstört!	Geschepper	Gegnerische Kamera zerstört!	leicht, niedrige Frequenz, konstant	„- .“
Tarnmodus aktiviert	+	Füllstandsanzeige hervorheben	Tarnmodus aktiviert!	leiser Wind	Tarnmodus aktiviert!	leicht, niedrige Frequenz, konstant	„- .“
Tarnmodus endet	+	Füllstandsanzeige hervorheben	Tarnmodus deaktiviert!	kurzer Ton	Tarnmodus deaktiviert!	leicht, niedrige Frequenz, konstant	„. -“
Abhören aktiviert	+	Füllstandsanzeige hervorheben	Abhören aktiviert!	„Hä“	Abhören aktiviert!	leicht, niedrige Frequenz, konstant	„- .“
Abhören endet	+	Füllstandsanzeige hervorheben	Abhören deaktiviert!	kurzer Ton	Abhören deaktiviert!	leicht, niedrige Frequenz, konstant	„. -“
Neutralisieren aktiviert	+	Füllstandsanzeige hervorheben	Neutralisieren aktiviert!	‘ssst“	Neutralisieren aktiviert!	leicht, niedrige Frequenz, konstant	„- .“
Neutralisieren endet	+	Füllstandsanzeige hervorheben	Neutralisieren deaktiviert!	kurzer Ton	Neutralisieren deaktiviert!	leicht, niedrige Frequenz, konstant	„. -“

Tabelle 13.4: Multimodale Darstellung von Ereignissen

Ereignis	Passive Umgebungserkundung	Aktive Umgebungserkundung	Charakteransicht	Missionsziel lösen	Nachricht verfassen	Nachricht abrufen
Startpunkt nah	AS,V	-	-	-	-	-
Goodie auf Karte	-	AS,V	AS,V	AS,V	AS,V	AS,V
Neutralisiert	-	AS,V,SA,T	AS,V,SA	AS,V,SA	AS,V,SA	AS,V,SA
Missionsziel erfüllt	-	AS,V,SA,T	-	SA,V	-	-
Missionsziel nicht erfüllt	-	AS,V,SA,T	-	SA,V	-	-
Rundenende	AS,T	AS,V,SA,T	AS,V,SA	AS,V,SA	AS,V,SA	AS,V,SA
Rundenanfang	AS,T	AS,V,SA,T	-	AS,V,SA	-	-
neue Post	-	AS,V	AS	AS	AS	AS
Fähigkeit nicht verfügbar	-	AS,V	-	-	-	-
Kamera gesetzt	-	AS	-	-	-	-
Kamera eingesammelt	-	AS	-	-	-	-
Eigene Kamera deaktiviert	-	AS,V	AS,V	AS	AS,V	AS,V
gegnerische Kamera deaktiviert	-	AS	-	-	-	-
Tarnmodus aktiviert	-	AS	-	-	-	-
Tarnmodus endet	-	AS,V	AS,V	AS	AS,V	AS,V
Abhören aktiviert	-	AS	-	-	-	-
Abhören endet	-	AS,V	AS,V	AS	AS,V	AS,V
Neutralisieren endet	-	AS,V	AS,V	AS	AS,V	AS,V
Neutralisieren aktiviert	-	AS	-	-	-	-

Tabelle 13.5: Standardeinstellung für multimodale Darstellung von Ereignissen

13.5.3 Eingabemöglichkeiten

Im folgenden Abschnitt werden die Eingabemöglichkeiten vorgestellt, die einem Spieler von NABB zur Verfügung stehen. Dabei werden lediglich haptische Eingaben betrachtet, da der Einsatz anderer Eingabmodalitäten zwar angedacht, ein Einsatz zum jetzigen Zeitpunkt aber nicht vorgesehen ist. So wurde bspw. die Verwendung einer Sprachsteuerung aufgrund des hohen zu betreibenden Aufwands, der den Rahmen einer Projektgruppe gesprengt hätte, fallen gelassen.
Die möglichen Benutzereingaben, wie sie in Tabelle 13.6 auf Seite 302 aufgelistet sind, werden bezüglich der Anforderungen an ihre Verfügbarkeit verschiedenen Prioritätsklassen zugeordnet. So ist es notwendig, dass zeitkritische Eingaben, wie z.B. die Aktivierung des Tarnmodus, ohne größeren Aufwand vom Spieler vorgenommen werden können. Entsprechende Buttons sind daher so in die Nutzungsschnittstelle zu integrieren, dass der Spieler sie jederzeit ansteuern kann.
Neben den Benutzereingaben und deren Klassifizierung werden in einer dritten Spalte der Tabelle 13.6 die verschiedenen haptischen Eingabemöglichkeiten aufgeführt.

Benutzereingabe	**Priorität**	**ausführen durch Anklicken ...**
Neutralisieren aktivieren	hoch	... des Neutralisations-Button im Display oder am mobilen Gerät
Abhören aktivieren	niedrig	... des WireTap-Button im Display
Abhören deaktivieren	niedrig	... des WireTap-Button im Display
Aufklärungskamera setzen	mittel	... des Recon-Button im Display
Aufklärungskamera aufnehmen	mittel	... des Kamera-Symbols auf der Karte
gegnerische Kamera deaktivieren	hoch	... des Kamera-Symbols auf der Karte
Tarnen aktivieren	hoch	.... des Cloak-Button im Display oder am Gerät
Tarnen deaktivieren	mittel	... des Cloak-Button im Display oder am Gerät
Kartenausschnitt verschieben	mittel	... der Richtungstasten des Steuerkreuzes am Gerät
Kartenausschnitt vergrößern	mittel	... des ZoomIn-Button am Gerät oder auf der Karte
Kartenausschnitt verkleinern	mittel	... des ZoomOut-Button am Gerät auf der Karte
Karte auf eigene Position zentrieren	hoch	... des Steuerkreuzes am Gerät
Goodie aufsammeln	mittel	... des Goodie-Symbols auf der Karte
Goodie ablegen	mittel	... des Drop-Button in der Menüansicht
Goodie aus Inventar auswählen	niedrig	... des Goodie-Eintrags in der Menüansicht
Goodie einsetzen	mittel	... des Activate-Button in der Menüansicht
Wechsel zur Menüansicht	mittel	... des Menu-Tab
Wechsel zur Kartenansicht	hoch	... des Map-Tab
Wechsel zur Kontextansicht	hoch	... des Cont-Tab
Wechsel zur Nachrichtenansicht	mittel	... des Mail-Tab
Nachrichten aus Liste empfangener Nachrichten auswählen/anzeigen	niedrig	... der jeweiligen Nachricht in der Nachrichtenansicht
ausgewählte Nachricht löschen	niedrig	... des Delete-Button in der Nachrichtenansicht
Nachricht verfassen	niedrig	... des Compose-Tab in der Nachrichtenansicht
Empfänger für Nachricht angeben / auswählen	niedrig	... der Empfänger in der Liste der Mitspieler in der Nachrichtenansicht
Nachricht versenden	niedrig	... des Send-Button in Nachrichtenansicht
Missionsziel lösen	mittel	... des Solve-Button in der Menuansicht

Tabelle 13.6: Eingabemöglichkeiten

13.5.4 Grafische Darstellung der Nutzungsschnittstelle

Im folgenden Abschnitt wird der generelle Aufbau der Nutzungsschnittstelle von NABB vorgestellt. Zunächst wird ein Überblick der verwendeten peripheren Ein- und Ausgabemöglichkeiten eines PDAs gegeben. Daran anschließend werden die verschiedenen Ansichten von NABB kurz vorgestellt. Die den Beschreibungen beigefügten Grafiken dienen dazu, eine Übersicht der für die Interaktion zwischen Nutzer und Anwendung verwendeten Elemente zu geben. Anordnung und Aussehen einzelner Elemente kann während der Implementierungsphase noch variabel gestaltet werden.

13.5.4.1 Periphere Ein- und Ausgabemöglichkeiten

Aktuelle PDAs verfügen über eine Reihe visueller, haptischer und akustischer Ein- und Ausgabemöglichkeiten (vgl. 12.4.1 auf Seite 214). Die rudimentäre PDA-Darstellung in Abbildung 13.31 auf Seite 303 zeigt, auf welche Art und Weise die peripheren Ein- und Ausgabemöglichkeiten eines PDAs für NABB genutzt werden sollen.

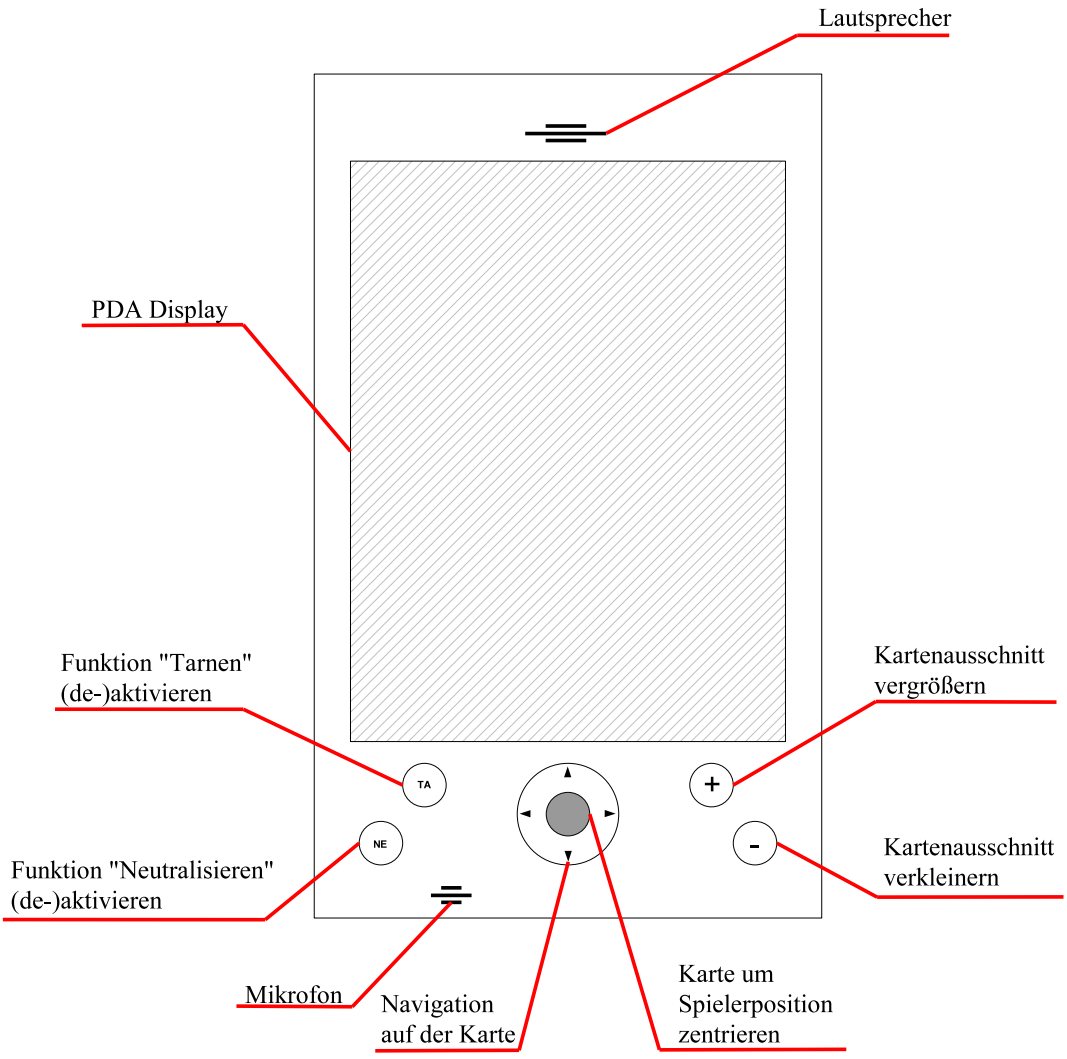


Abbildung 13.31: PDA Ansicht

Die visuelle Darstellung der Karte erfolgt über das PDA-Display. Aufgrund der geringen Größe dieses Displays, kann nur ein begrenzter Ausschnitt der Karte dargestellt werden. Um eine bessere Orientierung auf der Karte gewährleisten zu können, ist daher eine Zoom-Funktion für die Kartenansicht vorgesehen. Diese Funktion kann über die beiden rechten Funktionstasten des PDAs angesteuert werden. Neben der Zoom-Funktion, wird dem Spieler die Möglichkeit eingeräumt, frei auf der Karte zu navigieren, d.h. die Kartenansicht beliebig zu verschieben. Über das 5-Wege Steuerkreuz kann die Kartenansicht nach links, rechts, oben oder unten verschoben werden. Durch Anklicken des Steuerkreuzes wird die Kartenansicht wieder um die aktuelle Position des Nutzers zentriert.
Neben der Navigation auf der Karte, spielt auch die Ansteuerung der Funktionen Tarnen, Neutralisieren, Abhören und Aufklären eine wichtige Rolle. Je nach Spielsituation kann insbesondere der schnelle Zugriff auf die Funktionen Tarnen und Neutralisieren spielentscheidend sein. Aus diesem Grund werden die beiden linken Funktionstasten des PDA mit diesen Funktionen belegt.
Für die akustische Ausgabe, sowohl von einfachen Signalen als auch von Sprachnachrichten wird der Lautsprecher des PDA genutzt. Äquivalent hierzu wird das Mikrofon des PDA zur Aufnahme von Geräuschen oder Sprachnachrichten verwendet.

13.5.4.2 Standardansicht

Die Standardansicht von NABB ist in Abbildung 13.32 auf Seite 305 dargestellt. Diese enthält neben einem Bereich zur Anzeige der Karte noch eine Reihe von Statusanzeigen und Kontrollelementen, auf die im Folgenden näher eingegangen wird.
Die Statusleiste dient u.a. dazu, den Spieler über den Status des GPS-Signals, als auch über gegnerische Spieler oder gegnerische Kameras in der Nähe zu informieren. Darüber hinaus gibt die Statusleiste die, in Abhängigkeit vom augenblicklichen Kontext, zur Verfügung stehenden Ein- und Ausgabemodalitäten wieder. Hierzu gehören:

- Audiosignale, wie bspw. Warntöne (**A**udio)
- Visuelle Ausgaben (**V**oice)
- Haptische Ausgaben, wie bspw. Vibrationsalarm (**H**aptic)
- Empfang/Versand von Nachrichten (**M**ailBox)
- Anzeige des Nachrichtenticker (**T**icker)

Über eine Tabulator-Leiste kann der Spieler zwischen den unterschiedlichen Ansichten von NABB wechseln. Zu diesen zählen die Kartenansicht (MAP), die Kontextansicht (CONT), die Menüansicht (MENU) und die Nachrichtenansicht (MAIL). Ein *blinkender* Tab signalisiert dem Spieler, dass innerhalb der zugehörigen Ansicht aktuelle Informationen verfügbar sind.
Der Nachrichtenticker dient dazu, den Spieler über spielrelevante Ereignisse zu informieren. Die Elemente der Kontrollleiste ermöglichen den Zugriff auf die Funktionen Tarnen (CLOAK), Neutralisieren (NEUTRALIZE), Aufklären (RECON) und Abhören (WIRETAP). Der gegenwärtige Status dieser Fähigkeiten kann ebenfalls über die Kontrollleiste eingesehen werden.

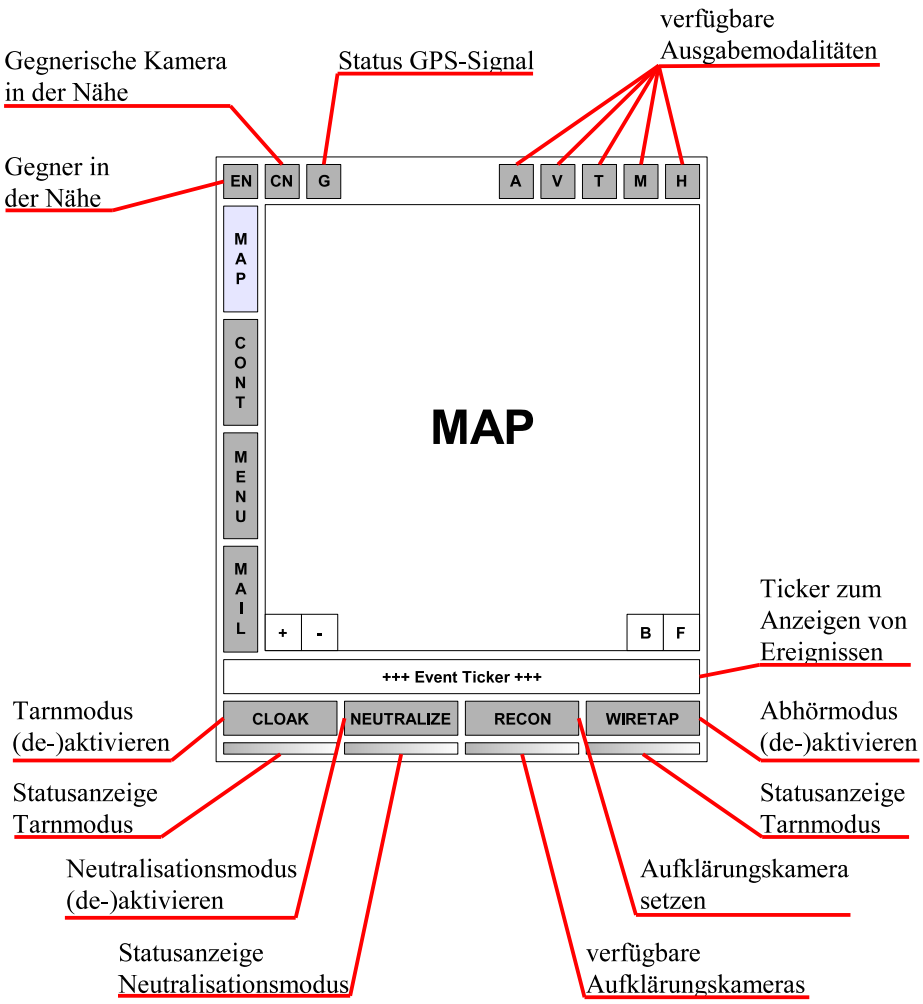


Abbildung 13.32: Standardansicht

13.5.4.3 Setup-Ansicht

Die *Setup*-Ansicht (siehe 13.33 auf Seite 306) wird zu Programmstart aufgerufen. Der Spieler muss an dieser Stelle zunächst einige Einstellungen vornehmen. Zu den notwendigen Angaben zählen:

- Spielszenario
- IP-Adresse und Portnummer des Servers
- Spielername
- Verteilung der Fähigkeitspunkte
- (De-)Aktivierung von GPS

Sind alle Angaben vom Spieler korrekt gemacht worden, kann er das Spiel durch Anklicken des *OK*-Buttons starten. Die Anwendung kann jederzeit durch Anklicken des *Close*-Buttons geschlossen werden.

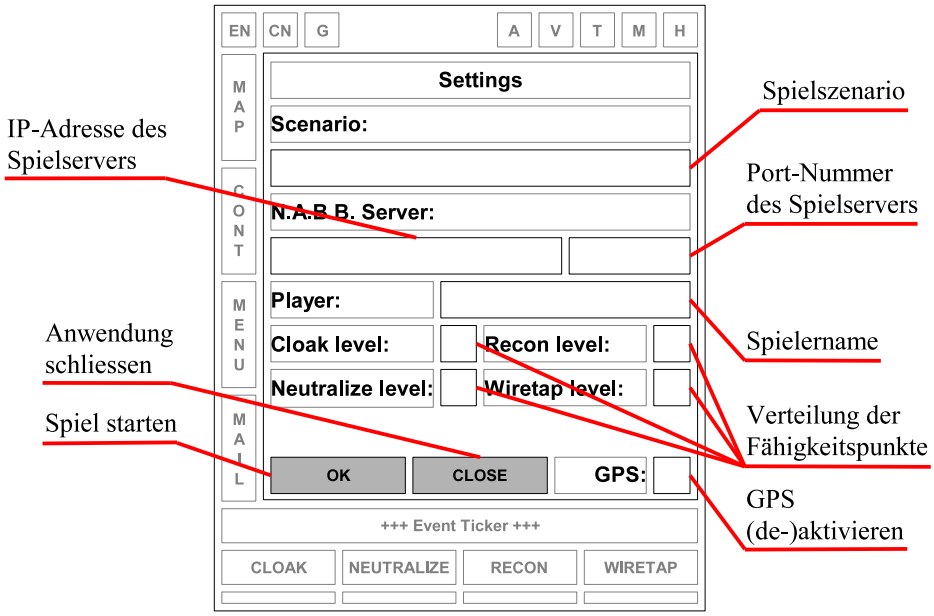


Abbildung 13.33: *Setup*-Ansicht

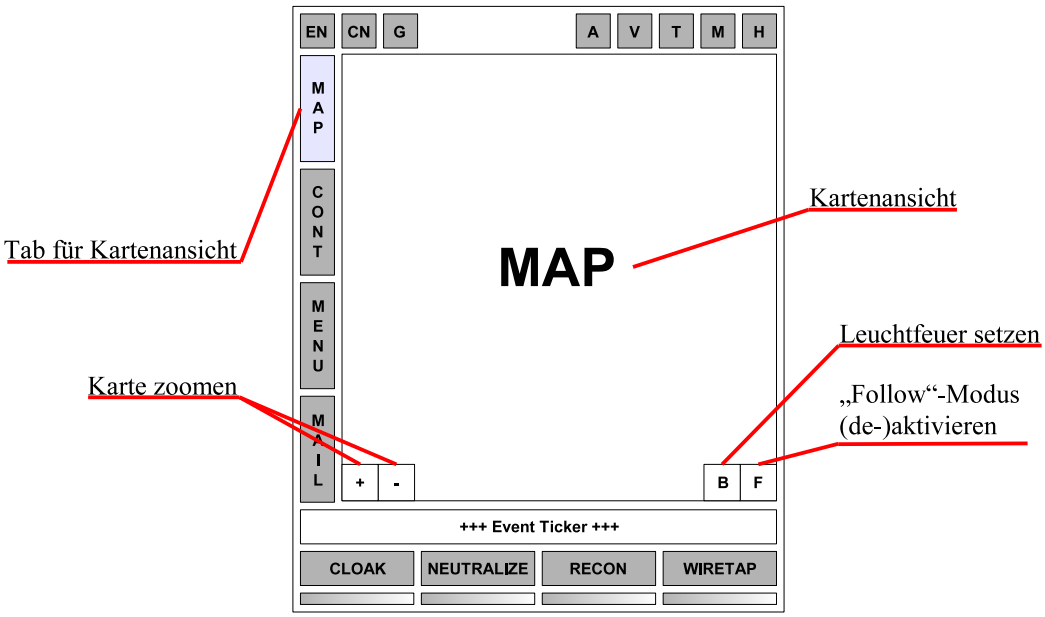


Abbildung 13.34: Kartenansicht

13.5.4.4 Kartenansicht

Die Darstellung der Karte ist in Abbildung 13.34 auf Seite 306 zu sehen. Durch Anklicken des Kartentabs (MAP) gelangt der Spieler in diese Ansicht.
Auf der Karte werden dem Spieler alle relevanten Spielinformationen angezeigt. Neben der eigenen Position werden dem Spieler auch die Positionen seiner Mitspieler und seiner abgelegten Kameras angezeigt. Gegnerische Spieler, gegnerische Kameras und *Goodies* werden für den Spieler sichtbar, sobald sich diese Objekte in seinem Sichtradius befinden.
Ein *blinkender* Kartentab signalisiert dem Spieler, dass neue Umgebungsinformationen auf der Karte verfügbar sind.

13.5.4.5 Kontextansicht

Die Kontextansicht gliedert sich in die zwei Bereiche *General* und *Preferences*. Durch Anklicken des Kontexttabs (CONT) wechselt der Spieler in die Ansicht *General* (siehe 13.35 auf Seite 13.35). Hier wird der derzeitige Kontext des Spielers angezeigt. Zu den angezeigten Kontextdaten zählen:

- Verbindungsstatus
- augenblickliche Geschwindigkeit
- Bewegungsrichtung des Spielers
- Zustand der visuellen Wahrnehmung
- Zustand der auditiven Wahrnehmung
- Zustand der haptischen Wahrnehmung

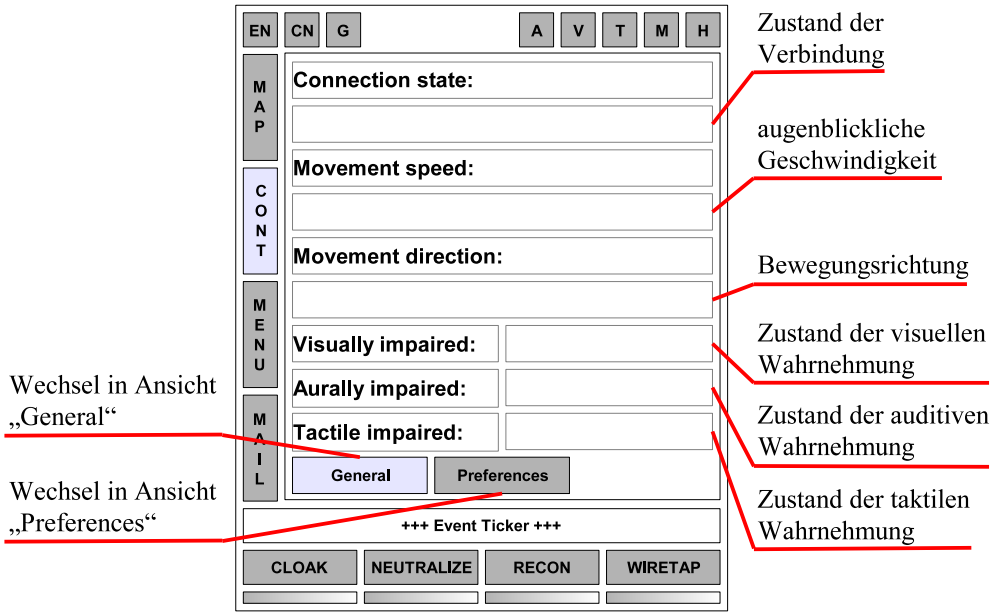


Abbildung 13.35: Ausgabe des augenblicklichen Kontexts

Durch Anklicken des *Preferences*-Buttons gelangt man in die *Preferences*-Ansicht, die in Abbildung 13.36 auf Seite 308) dargestellt ist. Hier kann der Spieler die von NABB angebotenen Ausgabemodalitäten (visuell, akustisch, haptisch) de- oder aktivieren.

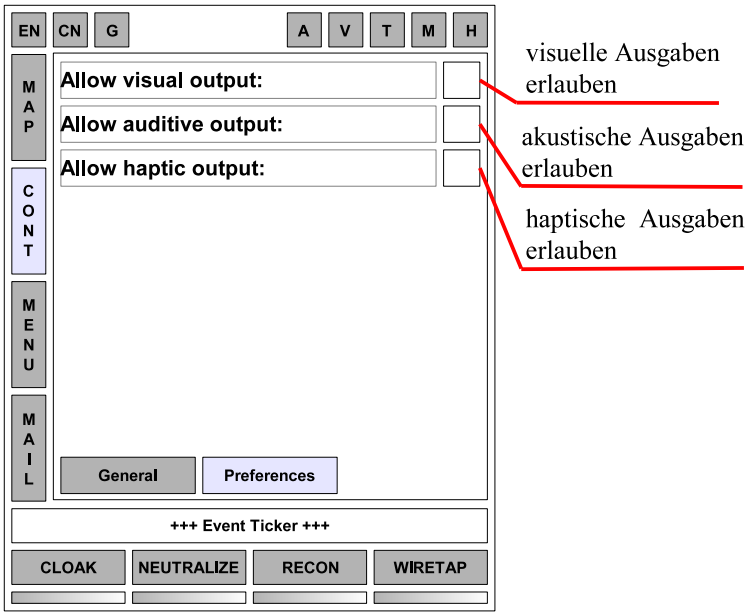


Abbildung 13.36: Bevorzugte Ausgabemodalitäten

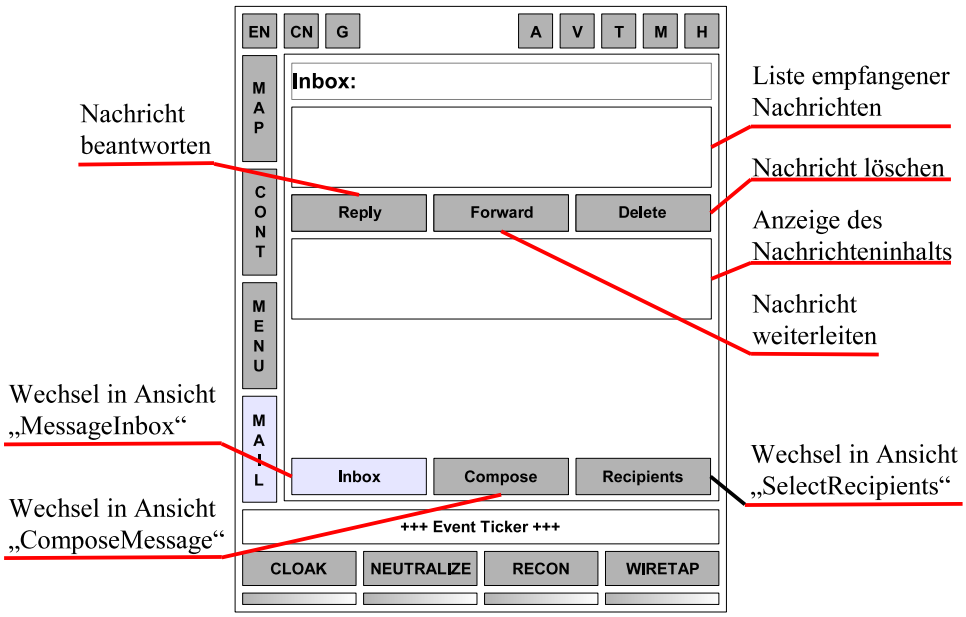


Abbildung 13.37: Nachrichteneingang

13.5.4.6 Nachrichtenansicht

Der Austausch von Informationen mit Mitspielern erfolgt in NABB über das Versenden und Empfangen von Nachrichten. Durch Anklicken des Nachrichtentabs (MAIL) gelangt man in die Ansicht *Nachrichteneingang* (siehe Abbildung 13.37 auf Seite 308). Der Empfang einer neuen Nachricht wird dem Spieler durch einen *blinkenden* Nachrichtentab signalisiert.
In dieser Ansicht werden dem Spieler alle empfangenen Nachrichten angezeigt. Durch Anklicken einer empfangenen Nachricht kann sich der Spieler den Nachrichteninhalt anzeigen lassen, dem Absender eine Antwort zuschicken (Reply), die Nachricht an weitere Teammitglieder weiterleiten (Forward) oder die Nachricht löschen (Delete).
Zum Verfassen einer Nachricht muss der Spieler die Ansicht wechseln. Hierzu muss er auf den *Compose*-Button klicken.

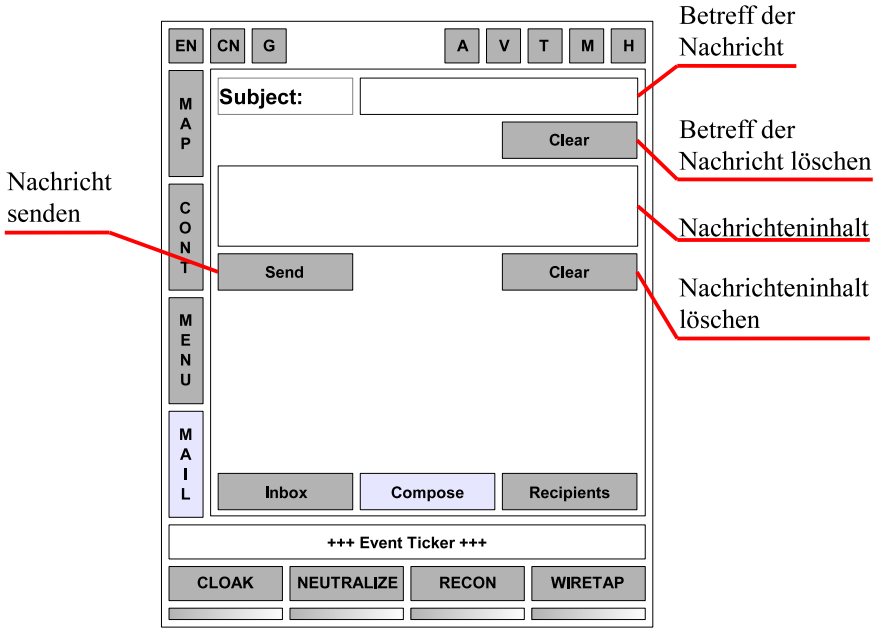


Abbildung 13.38: Nachricht verfassen

In der Ansicht zum Verfassen einer Nachricht (siehe Abbildung 13.38 auf Seite 309) kann der Spieler sowohl einen Betreff, als auch einen Nachrichtentext eingeben. Die Eingabe von Betreff und Nachrichtentext erfolgt nach der Auswahl des jeweiligen Textfeldes über ein virtuelles Tastaturfeld. Beide Einträge können über den zugehörigen *Clear*-Button wieder gelöscht werden.
Eine Nachricht die aus einem Betreff und mindestens einen Empfänger besteht, kann durch Anklicken des *Send*-Button versandt werden. Die Angabe der Empfänger der Nachricht erfolgt in einer weiteren Ansicht (siehe 13.39 auf Seite 310), in die der Spieler durch Anklicken des *Recipients*-Button wechselt.
In dieser Ansicht können Empfänger ausgewählt und durch Anklicken des *Add*-Button der Liste der Empfänger hinzugefügt werden. Darüber hinaus ist es möglich, bestimmte Empfänger aus dieser Liste auszuwählen und durch Anklicken des *Remove*-Button wieder aus der Empfängerliste zu entfernen.

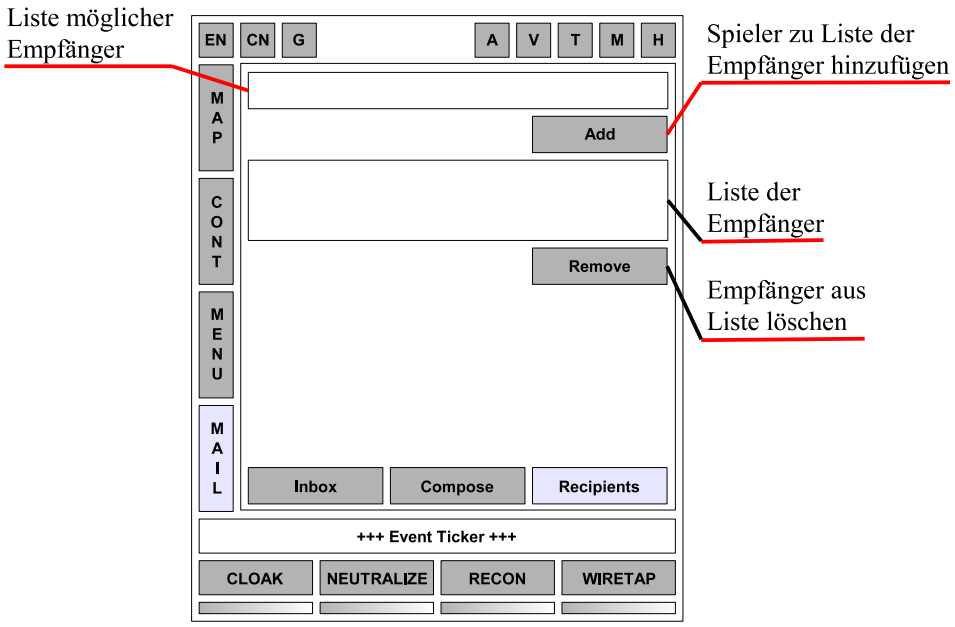


Abbildung 13.39: Empfänger auswählen

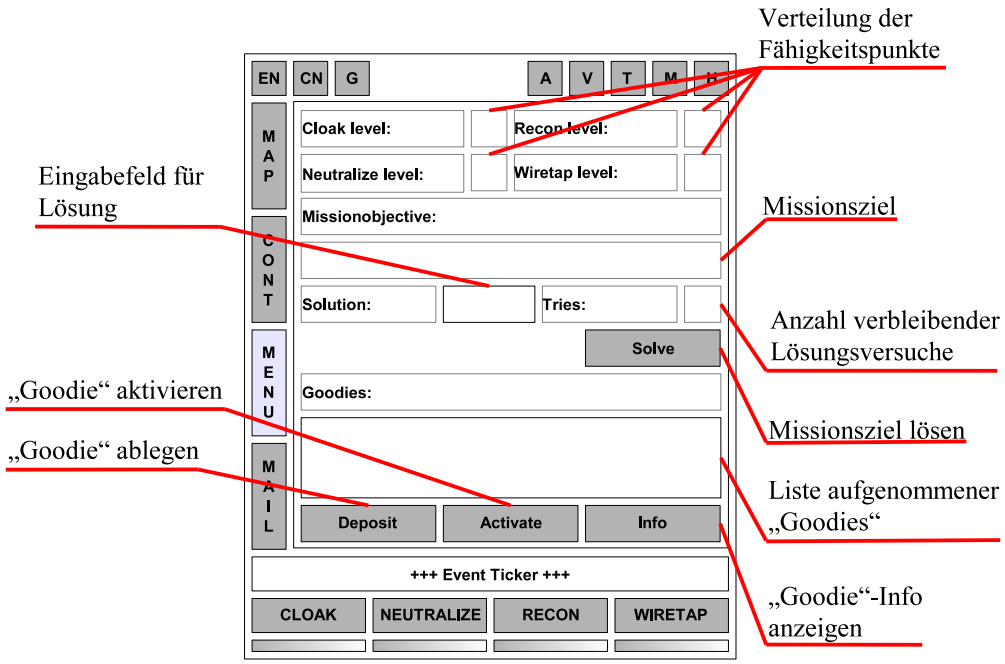


Abbildung 13.40: Menüansicht

13.5.4.7 Menüansicht

In die Menüansicht (siehe Abbildung 13.40 auf Seite 310) wechselt der Spieler durch Anklicken des Menütabs (MENU). In dieser Ansicht wird dem Spieler u.a. die zu Spielbeginn gewählte Verteilung seiner Fähigkeitspunkte angezeigt.
Die Menüansicht enthält weiterhin eine Liste mit allen aufgenommen Goodies, die durch Anklicken des *Drop*-Button an der augenblicklichen Position des Spielers abgelegt werden. Handelt es sich beim Spieler um den *Commander*, lassen sich die Goodies über den *Activate*-Button zusätzlich noch aktivieren.
Erreicht der *Commander* die Position des Missionsziels, werden in seiner Menüansicht zusätzliche Anzeigen eingeblendet. Zu diesen gehören ein Feld zur Eingabe der Lösung, eine Anzeige der verbleibenden Lösungsversuche, sowie ein Button, der angeklickt werden muss, wenn der Commander eine Lösung eingetragen hat.

13.6 Detaillierte Klassenbeschreibungen

Die folgenden Kapitel enthalten die detaillierten Beschreibungen zu den Klassen des Agentenspiels.
Die Entwürfe der verschiedenen Klassen sind gegliedert in Server, Client und gemeinsam genutzte Klassen, im Folgenden als Common bezeichnet. Diese enthalten neben der textuellen Beschreibung der Klassen auch Klassendiagrammen, welche die Interaktion zwischen den einzelnen Klassen, bzw. Modulen darstellen.

Kapitel 14

Implementierung

Dieser Abschnitt dokumentiert die Arbeit der Projektgruppe während der Implementierungdphase. Dazu gehören die festgelegten Coding Conventions. Ein weiterer Schritt vor dem Beginn des Programmierens ist die Einteilung der identifizierten Klassen aus dem Entwurf in Prioritätsklassen. Auf diese Weise soll sichergestellt werden, dass die essentiellen Klassen rechtzeitig fertiggestellt werden und dass, falls die eine oder andere Klasse aus Zeitdruck nicht optimal umgesetzt werden kann, es sich dabei nicht um eine Klasse handelt, die für den Spielverlauf unbedingt notwendig ist. Auf welche Weise die Konfiguration der Anwendung erfolgen kann, wird im gleichnamigen Teilabschnitt Konfiguration beschrieben. Anschließend wird noch auf Besonderheiten der Implementierung eingegangen, die während der Entwurfsphase noch nicht abzusehen waren. Zum Abschluss erfolgt eine Vorstellung der vorgenommen Optimierungsmaßnahmen.

14.1 Coding Conventions

In diesem Abschnitt werden die Coding Conventions unserer Wahl festgehalten. Dazu wird zunächst die Paket- und Klassenstruktur und der Aufbau der Klassen beschrieben. Im Anschluss sind einige Richtlinien zur Namensgebung und Kommentaren im Quelltext zu finden. Die Coding Conventions entsprechen zum größten Teil den Richtlinien von Microsoft zu C#[1], mit einigen, im Weiteren beschriebenen Ausnahmen. Die Einhaltung der Coding Conventions soll die Programmierarbeit erleichtern und die spätere Wiederverwendbarkeit erhöhen.

14.1.1 Paket- und Klassenstruktur

Um die Wartbarkeit der Pakete und Klassen zu erhöhen, steht jede öffentliche Klasse in einer eigenen Datei. Alle Klassen eines Namensraums werden in einem Verzeichnis zusammengefasst. Die Verzeichnisstruktur des Projekts entspricht der Gliederung der Namensräume, jeder Namensraum hat ein eigenes Verzeichnis mit den Kind-Namensräumen als Unterverzeichnisse.

14.1.2 Aufbau der Klassen

Die Klassen werden in mehrere Abschnitte unterteilt, die durch `#region` Präprozessor-Befehle und eine Überschrift als Kommentar gekennzeichnet sind. Ein Beispiel ist:

[1] Microsoft Richtlinien:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpgenref/html/cpconNamingGuidelines.asp

```
#region Eigenschaften

// ============================================================================
// Eigenschaften
// ============================================================================


#endregion
```

Falls es nicht leer sind folgende Abschnitte vorhanden:

- Konstanten,
- Attribute,
- Eigenschaften,
- Konstruktoren und Initialisierungsmethoden,
- und Klassenmethoden.

Sollten mehrere Methoden zu einem weiteren Aufgabenbereich gehören, können auch diese in einem eigenen Abschnitt zusammengefasst werden.

Eigenschaften Zu beachten ist, dass get- und set-Methoden von Eigenschaften nicht etwa unter Klassenmethoden, sondern wie auch in der Microsoftempfehlung nachzulesen, direkt im Bereich Eigenschaften mit angegeben werden, wie folgendes Beispiel für eine Eigenschaft namens Colour veranschaulichen soll.

```
public int Colour
{
    get {// Hier Code einfügen}
    set {// Hier Code einfügen}
}
```

14.1.3 Kommentare

Allgemein soll, wie auch schon im Entwurf, der eigentliche Code in englischer Sprache, Kommentare zum besseren Verständnis allerdings in deutsch verfasst werden. Jede Klasse enthält einen einleitenden `<summary>` Kommentarblock, in dem die Funktion der Klasse beschrieben wird. Danach folgen Angaben über die Autoren und das letzte Änderungsdatum:

```
/// <summary>
/// Zusammenfassung für ExampleClass.
/// </summary>
/// <author>SK, SA</author>
/// <date>06.06.2005</date>
```

Alle öffentlichen Methoden und Attribute werden nach dem Kommentierungsstandards von Microsoft für .NET und die automatische Generierung von Dokumentation vorgenommen.

14.1.4 Richtlinien zur Namensgebung

Für eine einheitliche Struktur sind folgende Richtlinien einzuhalten:

- **Groß- und Kleinschreibung**
  Für die Groß- und Kleinschreibung werden drei Bezeichnungsstyle verwendet: *Pascal Case*, *Camel Case* und *Upper Case*. Im Pascal Case werden die Anfangsbuchstaben jedes neuen Wortes (inklusive des ersten) innerhalb eines Bezeichners groß und alle anderen Buchstaben klein geschrieben. Beispiel: `BackColour`. Eingesetzt wird dieser Stil bei Namen für:

  – Klassen
  – Schnittstellen
  – Eigenschaften
  – Methoden

  Der Camel Case unterscheidet sich davon nur durch Verwendung eines kleinen Buchstabens am Anfang des Bezeichners. Beispiel: `backColour`. Dieser Stil wird für Bezeichner von Attributen und Parametern verwendet. Im Upper Case werden nur Grossbuchstaben verwendet. Zum Einsatz kommt dieser allerdings nur bei Bezeichnern mit weniger als drei Buchstaben. Beispiel: `ID`. Eine Besonderheit bei ID ist, dass sie auch innerhalb eines zusammengesetzten Bezeichners groß geschrieben wird wie etwa in `ClientID`. Insbesondere fallen Abkürzungen wie `Nabb` oder `Poi` nicht darunter.

- **Benennung von privaten Attributen**
  Abweichend von den Microsoft Vorgaben werden private Attribute nicht mit einem „_“ begonnen, sondern ebenfalls im *Pascal Case* gehalten, also `myName` statt `_myName`.

- **Benennung von Interfaces**
  Interfaces werden nicht mit einem `I` begonnen wie sonst nach den .NET Coding Conventions üblich. Das Interface Serializable heißt also `Serializable` und nicht `ISerializable`.

14.2 Priorisierung der Implementierung

Um die Implementierung priorisieren zu können, werden die durch Anwendungsfälle beschriebenen Programmieraufgaben einer von drei Prioritätsklassen zugeordnet:

- **must-haves** umfasst alle Anwendungsfälle, die auf jeden Fall in der finalen Version der Anwendung umgesetzt sein müssen und fehlerfrei laufen sollen. Hierzu gehören alle Anwendungsfälle, die dafür notwendig sind, um NABB spielen zu können, sowie alle Anwendungsfälle, die zu den Aspekten „kontext-sensitiv“ und „multimodal“ der Aufgabenstellung gehören. Der Implementierungsplan in Abschnitt 10.1, Seite 160 listet diese Anwendungsfälle auf, erkenntlich an der durchgezogenen Linie der Ellipse.

- **should-haves** sind alle Anwendungsfälle, die nach Möglichkeit umgsetzt werden sollen. Bei der Planung geht man davon aus, das ein großer Teil dieser Anwendungsfälle tatsächlich umgesetzt wird. Zu den *should-haves* gehören alle Anwendungsfälle, die gewünschte Erweiterungen des Spiels bzw. der Kontext-sensitivität und der multimodalen Interaktion umfassen. Im Implementierungsplan sind diese Anwendungsfälle durch die gestrichelten Linien bei den Ellipsen gekennzeichnet.

- **nice-to-haves** bezeichnet die Anwendungsfälle, die nur implementiert werden, wenn die *must-haves* und *should-haves* implementiert wurden. Darunter fallen vor allem sehr kreative, riskante Ideen, die nicht unbedingt in der Anwendung enthalten sein müssen, deren Machbarkeit man aber gerne testen möchte. Sie werden nicht durch den Implementierungsplan erfasst.

Die *nice-to-haves* werden so implementiert, dass sie per Präprozessor wahlweise in die Anwendung integriert bzw. herausgelassen werden können. So kann man hinterher von der Anwendung zwei Versionen erzeugen, von denen die eine nur die *must-haves* und *should-haves* implementiert, während die andere auch die *nice-to-haves* anbietet. Die erste Version soll mittels Integrationstests und Bugfixes möglichst stabil gemacht werden. Sie wird im Feinentwurf mit Klassendiagrammen und Kommentaren detailiiert beschrieben. Die zweite Version implementiert zusätzlich die *nice-to-haves* und soll nach Möglichkeit bei der Abnahme präsentiert werden. Stellt sich heraus, dass die *nice-to-haves* die Anwendung zu instabil machen, wird lediglich diese erste Version präsentiert. So kann man während der Phase der Integrationstests gefahrlos an der Umsetzung weiterer Ideen und Anwendungsfälle experimentieren, ohne den Erfolg des Projekts zu gefährden. Wie die *nice-to-haves* ein- bzw. ausgeschaltet werden, wird in Abschnitt 14.3.2 erläutert.

14.3 Konfiguration

In diesem Abschnitt wird beschrieben, wie die Konfiguration eines Spiels vorgenommen werden kann. Neben der Beschreibung der verwendeten Konfigurationsdatei *NabbConfig.txt* werden hier verschiedenen Konfigurationsmöglichkeiten der Kompilierung vorgestellt. Darüber hinaus orientiert sich jedes Spiel bei der Initialisierung an einem bereitgestellten Szenario. Szenarioinformationen werden in XML-Dateien gespeichert und müssen sowohl dem Server wie auch dem Client zur Verfügung stehen. Zu Beginn eines Spiels kann auf beiden Seiten das gewünschte Szenario ausgewählt werden, wobei darauf geachtet wird, dass beide identisch sind. Wie die Szenariodateien im Einzelnen aufgebaut sind und welche Informationen sie enthalten, wird ausführlich im Abschnitt 13.2.4 auf Seite 256 beschrieben.

14.3.1 NabbConfig

Das Agentenspiel NABB kann durch die, nach dem ersten Starten des Programms vorhandene, *NabbConfig.txt*-Datei konfiguriert werden. Diese befindet sich im Verzeichnis des Programms. Diese Konfiguration ermöglicht das Übernehmen der bisherigen Setup-Daten und das Setzen von Startparametern, ohne das eine Neukompilierung nötig ist. Nachfolgend sind die konfigurierbaren Einstellungsmöglichkeiten aufgelistet.

- **Server**:Die IP und der Port des Servers, zu dem sich NABB verbinden soll.
- **Scenario**: Der vollständig Pfad zu der Szenariodatei.
- **Neutralize**: Die zu letzt eingestellten Punkte auf „Neutralize“.
- **Recon**: Die zu letzt eingestellten Punkte auf „Recon “.
- **Wiretab**: Die zu letzt eingestellten Punkte auf „Wiretab “.
- **Cloak**: Die zu letzt eingestellten Punkte auf „Cloak “.
- **Connect**: Gibt an, ob sich zum Server verbunden werden soll. Ist der Wert „true“, dann wird ein Verbindungsversuch unternommen.

- **GPSMouse**: Beeinflusst, ob eine echte GPS-Mouse verwendet wird, oder ob der `MapCenterLocationRetriever` zum Einsatz kommt.
- **GPSMousePort**: Der Port, den die GPS-Mouse verwendet. Hat nur eine Bedeutung, wenn der Eintrag für **GPSMouse** den Werte „true“ hat.
- **VibrationLed**: Gibt an, ob das mobile Endgerät vibrieren kann. Mit einer „5“ wird die Vibrationsfähigkeit aktiviert.

Initial oder wenn ein Eintrag komplett fehlt, werden Standardwerte angenommen. Die Konfiguration ist robust gegen fehlende Einträge, die Reihenfolge der Einträge ist nicht relevant.

14.3.2 Bedingte Kompilierung

Dieser Abschnitt beschreibt, wie das Spiel bei der Kompilierung konfiguriert werden kann. Visual Studio ermöglicht es, verschiedene Konfigurationen zur Kompilierung anzulegen. Im Falle von NABB wurde für die Implementierung die Voreinstellung *Debug* angepasst und für den Release mehrere Varianten der Voreinstellung *Release* angelegt:

- **Debug** dient zur Entwicklung von NABB. Aus Geschwindigkeitsgründen bei der Kompilierung wurde dabei auf die Generierung der XML-Dokumentation und der Optimierung des Codes verzichtet. Dafür wurden die Unit-Tests mit in die Zieldateien eingebunden, so dass NABB regelmäßig getestet werden konnte. Zusätzlich wird auch die eigens geschriebene Logging-API mit eingebunden, so dass anhand der Log-Datei die Fehlersuche erleichtert wird.
- **Release** optimiert die Zieldateien für den Einsatz als Produkt. Die Kompilierung dauert länger, weil der Code optimiert wird, dafür ist der erzeugte Code schlanker und performanter. Die Unit-Tests werden nicht mit eingebunden um weiteren Platz zu sparen. Es existieren folgende Varianten der *Release* Kompilierung:
  - **Release BASIC** bindet lediglich die *must-haves* und die *should-haves* ein und stellt damit die minimalste und performanteste Variante von NABB dar.
  - **Release BASIC with Doc** generiert zusätzlich bei der Kompilierung die XML Dokumentation des Quellcodes.
  - **Release BASIC with Logging** bindet die eigens erstellte Logging-API mit ein, für den Fall, dass Fehler in der *Release BASIC* Kompilierung auftreten.
  - **Release NICE-TO-HAVES** integriert neben den *must-haves* und *should-haves* auch die *nice-to-haves* in die Zieldatei. Diese Variante ist damit eventuell weniger stabil als die *Release BASIC* Variante.
  - **Release NICE-TO-HAVES with Logging** fügt der Zieldatei zusätzlich die Logging-API hinzu, um eine eventuell nötige Fehlersuche zu unterstützen.

Jede Konfiguration unterscheidet sich u.a. von anderen Konfigurationen durch die Wahl der Konstanten für bedingte Kompilierung. Die Kompilierung des Quellcodes von NABB kann über folgende Konstanten beeinflusst werden:

- **DEBUG** ist die voreingestellte Konstante der *Debug* Konfiguration. Bei NABB entscheidet sie, ob die Unit-Tests mit in die Zieldatei mit eingebunden werden.

- **RELEASE** ist die voreingestellte Konstante der *Release* Konfiguration. Sie wird in NABB nicht verwendet.

- **LOG** steuert, ob die Infrastruktur zum Loggen von Ereignissen mit in die Zieldatei eingebunden werden soll, oder nicht. Da das Logger stark zu Lasten der Performanz geht, sollte er nur in der Entwicklungsphase und zur Fehlersucht eingebunden werden.

- **NICE2HAVES** gibt an, ob Funktionalitäten, die unter die Kategorie „Nice-to-haves“ fallen, in die Zieldatei eingebunden werden solle, oder nicht. So kann man gleichzeitig die „Must-haves“ und „Should-haves“ intensiv Testen, und ohne Risiko neue Features als „Nice-to-haves“ implementieren. Sollte es später mit den „Nice-to-haves“ Probleme geben, können sie für das Release des Produkts abgeschaltet werden.

14.4 Besonderheiten der Implementierung

In diesem Abschnitt werden Besonderheiten der Implementierung beschrieben. Besonderheiten sind sie deswegen, weil sich ihre Notwendigkeit erst während der Implementierung herausgestellt hat und sie nicht bereits während der Entwurfsphase vorgesehen waren. Da sie, obwohl nicht im Entwurf beschrieben, dennoch wichtiger Bestandteil der Anwendung sind, werden sie im Folgenden vorgestellt. Darunter fallen die Abschnitte der Logging API und der threadsicheren Benutzungsschnittstelle, deren Notwendigkeit erst während der Implementierung erkannt wurde. Abschließend wird die Positionbestimmung mittels GPS erläutert.

14.4.1 Logging API

Beim Ausführen der Anwendung soll es möglich sein, aus dem Code heraus Nachrichten mitzuloggen. Zumeist bedeutet das, kurze Textnachrichten auszugeben bzw. zu speichern, die Aufschluss über den aktuellen Zustand des Programmverlaufs geben. So kann z.B. bei der Suche nach einem Fehler der Verlauf des Programms nachvollzogen werden. Die im folgendem vorgestellte Logging API ist von der Projektgruppe selbst entwickelt worden und ist sowohl im .NET Framework als auch im .NET Compact Framework einsetzbar.

Die naive Methode, den Anwendungsverlauf zu loggen, ist das Schreiben von Nachrichten auf die Console mittels `Console.WriteLine(string)`. Die typischen Zielgeräte des .NET Compact Frameworks besitzen jedoch keine Console, so dass die Aufrufe von `Console.WriteLine(string)` ignoriert werden.

14.4.1.1 Klasse `Debugger`

Die Logging API wird durch die Klasse `Debugger` bereitgestellt. Sie bietet eine Reihe von öffentlichen, statischen Methoden, denen die zu loggenden Nachrichten und die aufrufende Klasse als Parameter übergeben werden können. Folgende Methoden repräsentieren die verschiedenen Typen von Log-Nachrichten:

- **InfoMessages** sind Nachrichten, die am ehesten im Programmcode verweilen. Der Debugger bietet zur Ausgabe dieser Nachrichten die Methode `Debugger.LogInfoMessage(object, string)` an. Sie soll immer dann verwendet werden, wenn der auszugebende Text einen Zustand beschreibt, der auch vom Laien verstanden werden kann, z.B. „Spieler Maria hat sich getarnt“. Diese Nachrichten sollen nur selten vorkommen und beschreiben dann den Aufruf ganzer Anwendungsfälle.

- **Exception** sind aufgetretende Ausnahmen (Klassen, die vom Typ `System.Exception` abgeleitet sind). Sie werden über die Methode `Debugger.LogException(object, Exception)` ausgegeben.
- **DebugMessage** sind Nachrichten, die den Entwicklern zum Nachvollziehen des Programmverlaufs dienen. Hier können z.B. Zustände von Variablen und Methodenaufrufe ausgegeben werden. Der Debugger bietet dazu die Methoden `Debugger.Log(object, string)` und `Debugger.Log(string)` an.

14.4.1.2 Schnittstelle `LogConsumer`

Die eigentliche Ausgabe der Nachrichten übernimmt der Debugger jedoch nicht selbst. Dazu müssen zunächst Klassen registriert werden, die die Schnittstelle `LogConsumer` implementieren. Die Anwendung stellt bereits folgende Implementierungen bereit:

- `AbstractLogger`: implementiert die Methoden der Schnittstelle und leitet die Aufrufe eine abstrakte Methoden mit der Signatur `Log(string)` weiter. So müssen die konkreten Implementierung der `LogConsumer` nur noch die Ausgaben einer Zeichenkette implementieren, während die administrativen Aufgaben bereits abgehandelt wurden.
- `ConsoleLogger`: implementiert den `AbstractLogger` und schreibt die auszugebenden Nachrichten auf die Console. Dieser `LogConsumer` wird vorzugsweise auf dem Server eingesetzt.
- `FileLogger`: implementiert den `AbstractLogger` und schreibt die auszugebenden Nachrichten in eine Textdatei. Dieser `LogConsumer` ist das Mittel der Wahl zum Loggen von Nachrichten auf dem Client.
- `LogControl`: implementiert die Schnittstelle `LogConsumer` direkt und ist ein Element der grafischen Nutzungsschnittstelle des Servers. Die Log-Nachrichten werden in ein scrollbaren Textfeld geschrieben.

Zu jedem `LogConsumer` kann man explizit das Log-Level einstellen, ab dem Nachrichten ausgegeben werden:

- Log-Level **Nothing**: alle Log-Nachrichten werden verworfen
- Log-Level **InfoMessages**: es werden nur InfoMessages ausgegeben
- Log-Level **Exceptions**: es werden zusätzlich noch Ausnahmen ausgegeben
- Log-Level **All**: alle Nachrichten werden ausgegeben, inklusive den DebugMessages

14.4.1.3 Schnittstelle `LogFilter`

Bevor eine Nachricht jedoch überhaupt an die registrierten `LogConsumer` weitergeleitet wird, wird sie zunächst durch den `LogFilter` gefiltert. Dabei handelt es sich um eine Implementierung der Schnittstelle `LogFilter`. Sie bietet Methoden an, um zu einer Nachricht einen Wahrheitswert berechnen zu lassen. Nur wenn die Methode denn Wert `true` zurückliefert, reicht der `Debugger` sie an die `LogConsumer` zur Ausgabe weiter.

Ob eine Nachricht ausgegeben werden soll, kann anhand zwei Eingaben bestimmt werden:

- Der `LogFilter` kann so eingestellt werden, dass Nachrichten mit einem zu niedrigen Log-Level komplett ignoriert werden. Dies geschieht auch, wenn ein `LogConsumer` eine großzügigeres Log-Level eingestellt hat.
- Die zweite Eingabe ist die Nachricht selbst. So kann z.B. nach bestimmten Schlüsselwörtern gesucht werden, anhand derer entschieden wird, ob eine Nachricht ausgegeben werden soll, oder nicht.

Konkrete Implementierungen der Schnittstelle `LogFilter` stellt die Anwendung folgende bereit:

- `AbstractLogFilter`: ist eine Teilimplementierung der Schnittstelle `LogFilter`, bei der man bereits das minimale Log-Level der auszugebenden Nachrichten einstellen kann. Sie enthält weiterhin keine Logik.
- `DefaultLogFilter`: ist die Standardimplementierung des `AbstractLogFilter`. Diese Klasse hält eine Liste von Schlüsselwörtern vor, anhand derer Nachrichten abgelehnt werden. Enthält eine Nachricht ein Wort, dass in der Schlüsselwortliste auftaucht, wird es nicht ausgegeben. So kann man z.B. sehr einfach die Log-Ausgaben bestimmter Module unterdrücken, indem man den Namen des Moduls in die Liste aufnimmt.
- `AdditiveLogFilter`: ist die zum `DefaultLogFilter` gegensätzlich Implementierung des `AbstractLogFilter`. Sie enthält ebenfalls eine Liste von Schlüsselwörtern. Nachrichten werden aber nur ausgegeben, wenn sie mindestens ein Wort aus der List enthalten. So kann man z.B. speziell die Log-Ausgaben eines bestimmten Moduls überwachen.

14.4.2 Threadsichere Benutzungsschnittstelle

Während der Hauptimplementierungsphase ist folgender Fehler aufgetreten: nach dem Starten der Clientapplikation reagierte die graphische Benutzerschnittstelle nach einiger Zeit nicht mehr auf Eingaben des Benutzers. Die Clientapplikation ist trotzdem teilweise weitergelaufen, bspw. sind verschiedene Zeitgeber, die das Aufladen der Funktionalitäten steuern, noch aktiv gewesen. Der Zeitpunkt des Einfrierens der Benutzschnittstelle ist nicht deterministisch gewesen. Vier Aspekte sind dabei aufgefallen:

- Die Benutzschnittstelle ist immer eingefroren.
- Je mehr Spielaktivität zu verzeichnen gewesen ist, desto schneller ist der Fehler aufgetreten.
- Die Aktivität der CPU ist nicht erhöht gewesen.
- Der Speicherverbrauch während der Spielaktivität ist im vertretbaren Rahmen geblieben und nicht angestiegen.

Dieses Fehlerbild ist typisch für einen nicht deterministischen Deadlock. Der Deadlock entsteht mit einer gewissen Wahrscheinlichkeit. Nach Recherche im Internet ist das Problem deutlich geworden. Das .NET-Framework erlaubt nur dem Thread die Kommunikation mit Steuerelementen, in dem sie erstellt worden sind. Das ist immer der Hauptthread der Applikation. In der Clientapplikation ist nun folgendes passiert: Zwei Threads haben Veränderungen der graphische Benutzerschnittstelle herbeigeführt. Zum einen der Hauptthread durch Interaktion des Benutzers mit den Steuerelementen, bspw. durch Anklicken eines Buttons oder der Karte. Dies ist unproblematisch, weil die Behandlung von Ereignissen, ausgelöst durch Steuerelemente, immer über den Hauptthread abgearbeitet wird. Der zweite Fall aber kann einen Deadlock auslösen. Die graphische Benutzerschnittstelle wird auch durch eingehende `Events` vom Server durch den `EventMapper`-Thread geändert. Dieses passiert unter anderem bei der Ausführung einer

Funktionalität oder bei der Interaktion mit Objekten auf der Karte. Wenn jetzt der Hauptthread die Ereignisbehandlung eines Steuerelemente noch nicht abgeschlossen hat und der `EventMapper` ebenfalls ein Steuerelementereignis auslöst, so entsteht im ungünstigen Fall ein Deadlock. Da die Behandlung von Steuerelementereignissen standardmäßig blockierend abläuft, warten die beiden Threads jetzt aufeinander. S. Senthil Kumar erläutert dieses Problem und dessen Hintergründe detailliert unter [Kum05].

Die Lösung des Problems besteht darin, die Aufrufe des `EventMapper`-Threads in die Ereigniswarteschlange des Hauptthreads einzureihen und nicht auf ihre Abarbeitung zu warten. Für das .NET-Framework funktioniert das folgendermaßen:

```
1  private delegate void RefreshPanelDelegate(Panel panel);
2
3  private static void SafeRefreshPanel(Panel panel)
4  {
5     panel.Refresh();
6  }
7
8  /// <summary>
9  /// Wird aufgerufen, wenn sich der Zustand eines Spielobjekts geaendert hat.
10 /// </summary>
11 ///<param name="gameObject">das betroffene Spielobjekt</param>
12 public void GameObjectStateChanged(GameObject gameObject)
13 {
14    if (mapPanel.InvokeRequired)
15    {
16       Object[] parameters = {mapPanel};
17       mapPanel.Invoke(new RefreshPanelDelegate(SafeRefreshPanel), parameters);
18    }
19    else
20    {
21       mapPanel.Refresh();
22    }
23 }
```

Listing 14.1: C# und Invoke auf dem Server

Der Lösungsweg, wie in Listing 14.1 exemplarisch gezeigt, ist unter anderem unter [Unbnt] und [Kum05] beschrieben. Zentraler Punkt ist der Methode `Invoke` der `Control`-Klasse. Diese sorgt dafür, dass die durch das *delegate* gekapselte Methode, die dann das Ereignis für das Steuerelement (`Control`) auslöst, im korrekten Thread, dem Hauptthread, aufgerufen wird. Mehr zu *delegate*s ist unter [Han02] im Abschnitt „Delegates & Events“ ab Seite 347 zu finden. Es wird mit der Eigenschaft `InvokeRequired` der `Control`-Klasse überprüft, ob es erlaubt ist, in dem gerade aktiven Thread die Ereignisbehandlung für das `Control` durchzuführen, oder ob `Invoke` benutzt werden muss. Auf der der Serverapplikation wird vorsorglich dieses Vorgehen genutzt, um auch dort ein einfrieren der Karte zu verhindern.

Für die Clientapplikation eignet sich dieses Vorgehen allerdings nur in deutlich abgewandelter Form, da dort das .NET CF-Framework verwendet wird. Dieses unterstützt weder die Eigenschaft `InvokeRequired` der `Control`-Klasse, noch die Verwendung eigener *delegates* für die `Control.Invoke`-Methode. Im einzelnen bedeutet dies, dass `Control.Invoke` aus Sicherheitsgründen immer benutzt werden muss, da nicht bekannt ist, ob es umgangen werden kann und weiterhin kann nur das *delegate* `System.EventHandler` verwendet werden. Um diese Probleme allgemeingültig zu lösen, ist die Klasse `InvokeHelper` mit ihren diversen internen Wrapperklassen eingeführt worden.

Der `InvokeHelper` besteht aus einer Ansammlung statischer Methoden, welche die verschiedenen, für die Clientapplikation notwendigen Interaktionen mit `Control`s vornehmen. Dazu gehören das Auslesen und Setzen der `Text`-Eigenschaft und `Enabled`-Eigenschaft, das Setzen der Hintergrundfarbe, usw. Die Funktionsweise des `InvokeHelper`s und der Wrapperklassen wird anhand von Listing 14.2 erläutert. Hier wird ein Ausschnitt das `InvokeHelper`s gezeigt, der analog zu Listing 14.1 die `Refresh`-Methode eines `Control`s aufruft. In diesem Fall allerdings kompatibel zu dem .NET CF-Framework.

```
1  namespace Common.Logic.Userinterface.Gui
2  {
3     /// <summary>
4     /// Die Klasse InvokeHelper stellt statische Hilfsmethoden zur threadsicheren
          Veraenderung
5     /// von Gui-Komponenten zur Verfuegung.
6     /// </summary>
7     /// <author>Stefan Andressen</author>
8     /// <version>1.02</version>
9     /// <date>2005-09-07</date>
10    public class InvokeHelper
11    {
12       #regionÖ ffentliche statische Hilfsmethoden
13
14       /// <summary>
15       /// Ruft threadsicher Refresh() auf dem Control auf.
16       /// </summary>
17       /// <param name="control">das neu zu zeichnende Control</param>
18       public static void RefeshControl(Control control)
19       {
20          RefreshWrapper wrapper = new RefreshWrapper(control);
21          control.Invoke(new EventHandler(wrapper.Refresh));
22          Application.DoEvents();
23       }
24
25       #endregion
26
27       #region Private Hilfsklassen
28
29       private class RefreshWrapper
30       {
31          private Control control;
32
33          public RefreshWrapper(Control control)
34          {
35             this.control = control;
36          }
37
38          public void Refresh(object o, EventArgs e)
39          {
40             control.Refresh();
41          }
42       }
43
44       #endregion
45    }
46 }
```

Listing 14.2: Ausschnitt des `InvokeHelper`s mit Wrapperklasse

Die Methode `InvokeHelper.RefeshControl` bekommt als Parameter das `Control` übergeben. Es wird eine Instanz der Klasse `RefreshWrapper` mit diesem `Control` erzeugt. Der `RefreshWrapper` hat eine Methode `Refresh`, deren Signatur der durch das *delegate* `System.EventHandler` vorgegebenen entspricht. Mit einer Instanz des `System.EventHandler`, die als Parameter die `Refresh`-Methode des `RefreshWrapper`s erhält, wird die `Invoke`-Methode des `Control`s aufgerufen. Dadurch ist sichergestellt, dass die `Refresh`-

Methode vom `RefreshWrapper` im Hauptthread aufgerufen wird. Die `RefreshWrapper.Refresh`-Methode tut ihrerseits nichts weiter, als `Refresh` für das `Control` aufzurufen. Abschließend wird noch `Application.DoEvents` aufgerufen, was sicherstellt, dass die `Control.Refresh`-Methode auch wirklich jetzt aufgerufen wird.

Mit dieser Vorgehensweise ist es gelungen, die graphische Benutzerschnittstelle der Clientapplikation und der Serverapplikation threadsicher zu gestalten. Auf dem Client sind durch den `InvokeHelper` leichte Performanzeinbußen zu verzeichnen, die aber während des Spiels nicht negativ auffallen. Die Performanzeinbußen sind nicht zu vermeiden, will man eine Korrektheit des Ablaufs der Clientapplikation gewährleisten. Der `InvokeHelper` zeichnet sicher weiterhin durch seine einfache Benutzbarkeit aus. Andere Klassen müssen sich nicht um Interna der *delegates* und `Control.Invoke` kümmern, sondern rufen nur die statischen Methoden des `InvokeHelper`s auf. Ein negativer Beigeschmack verbleibt aufgrund der mangelhaften Dokumentation des .NET CF-Framework. Es ist nicht in der offiziellen .NET-Dokumentation erwähnt[2], dass im .NET CF-Framework keine eigenen *delegate*s verwendet werden können, sondern nur *delegates* vom Typ `System.EventHandler`. Diese Tatsache hat zu einigen ratlosen Stunden geführt. Es bleibt zu hoffen, dass sich die Dokumentation des .NET-Frameworks mit der Zeit verbessert. In der Version 2.0 des .NET-Frameworks werden sogar Exceptions geworfen, wenn mit `Control`s aus dem falschen Thread heraus interagiert wird. Wäre dies schon im .NET-Frameworks 1.1 der Fall gewesen, dann hätte man den Grund für die Deadlocks nicht lange suchen müssen.

14.4.3 Positionbestimmung mittels GPS

Die Bestimmung der Positionen einzelner Clients erfolgt in NABB mittels GPS. Das eingesetzte Protokoll ist das Datenprotokoll der National Marine Electronics Association (NMEA) [3]. Das NMEA-Protokoll verwendet den ASCII-Zeichensatz , wobei Carriage Return und Line Feed ebenfalls übertragen werden. Die Übertragung erfolgt in Form von *Sätzen* mit einer Datenrate von 4800 Baud. Jeder Satz fängt mit einem „$“, gefolgt von einer zweistelligen *Talker-ID* und einer dreistelligen *Sentence-ID* an. Der eigentliche Inhalt besteht aus kommaseparierten Datenfeldern, die am Ende mit einer optionalen Checksumme und einem obligatorischen CR/LF abgeschlossenen werden. Inklusive „$“ und CR/LF darf ein Satz maximal 82 Zeichen enthalten. Sind bestimmte Daten nicht vorhanden, werden die entsprechenden Datenfelder weggelassen, die separierenden Kommas aber trotzdem gesendet. Welche Sätze gesendet werden, hängt vom Gerätehersteller und vom verwendeten Chipsatz ab. NABB unterstütz NMEA in der Version 2.2 und kann folgende Sätze verarbeiten: GGA, GLL, GSA, GSV, RMC und VTG. Proprietäre Sätze, die bei vielen Hersteller neben den Standardsätzen Verwendung finden, werden nicht unterstützt.
Die Funktionsweise des `GPSLocationRetriever` ist denkbar einfach. Werden über einen virtuellen COM-Port Daten empfangen, werden diese zuerst in einen String konvertiert. Der String wird in Teilstrings unterteilt, wobei jeder GPS-String durch das Carriage Return abgeschlossen ist. Ist ein GPS-String als ganzer String identifiziert, erkennbar durch $ als Start- und ein Carriage Return als Abschluss-Symbol, wird mit dem Parsen der Informationen innerhalb des Strings begonnen. Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ’,’) und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind dabei durch den Standard fest vorgegeben.

- Ein RMC-Satz (Startsequenz $GPRMC) enthält u.a. die Angaben (geographische) Länge, Breite und die momentane Bewegungsrichtung.

[2] Die .NET CF-Dokumentation ist in der .NET-Dokumentation enthalten.
[3] Weitere Informationen unter www.nmea.org.

- Ein GLL -Satz (Startsequenz $GPGLL) enthält u.a. die Angaben (geographische) Länge und Breite.
- Ein VTG -Satz (Startsequenz $GPVTG) enthält u.a. die Angaben über die aktuelle Geschwindigkeit.
- Ein GGA -Satz (Startsequenz $GPGGA) enthält u.a. die Angaben (geographische) Länge, Breite und Höhe, sowie Informationen über die Qualität der empfangenen GPS-Signale, die Anzahl der Satelliten, von denen Signale empfangen wurden und die horizontale Genauigkeit.
- Ein GSA -Satz (Startsequenz $GPGSA) enthält u.a. die Angaben über die Genauigkeit und die horizontale sowie die vertikale Genauigkeit der GPS-Informationen.
- Ein GSV -Satz (Startsequenz $GPGSV) enthält u.a. die Angaben über die Anzahl der Satelliten, die momentan in Sichtweite des GPS-Empfängers liegen. (Es werden nicht zwangsläufig auch von allen Satelliten Signale empfangen und zur Positionsbestimmung herangezogen).

14.5 Optimierungsmaßnahmen

Dieser Abschnitt führt die wichtigsten Optimierungsmaßnahmen an, die NABB während der Implementierung und der Integrationstests erfahren hat.

14.5.1 Optimierung der Kartendarstellung

In frühen Versionen von NABB wurde die Karte immer dann neugezeichnet, wenn die GPS-Maus die Position des Spielers bestimmt hatte. Da dies sehr häufig geschah und das Zeichnen der Karte eine sehr zeitintensive Operation ist, stauten sich die Anfragen zum Neuzeichnen der Karte. Die Folge war, dass die eigene Position auf der Karte mit deutlicher und immer größer werdenden Verzögerung gezeichnet wurden.

Die Optimierungsmaßnahmen setzen an zwei Stellen an. Zum einen wird die Karte nur noch neu gezeichnet, wenn sich die eigene Position tatsächlich geändert hat. Dazu wird nur noch in Intervallen von einer Sekunde überprüft, ob sich die eigene Position überhaupt verändert hat. Die Karte wird also höchstens jede Sekunde neu gezeichnet.

Zum anderen wurde das Zeichnen der Karte selbst optimiert. Zuvor wurde bei jeder Operation die gesamte Karte gezeichnet, obwohl sich zumeist Teile der Karte außerhalb des sichtbaren Anzeigebereiches befinden. In der optimierten Version wird nur noch der sichtbare Bereich der Karte gezeichnet.

Kapitel 15

Qualitätsmanagement

15.1 Einleitung

Dem Qualitätsmanagement (QM) fällt bei der Durchführung von Softwareprojekten eine entscheidende Rolle zu. Neben der Überprüfung von Dokumenten auf Korrektheit und Konsistenz, sowie der Validierung und Verifikation der erstellten Software, nimmt das QM heutzutage ebenfalls Einfluss auf das Projektmanagement. Die Planung eines Softwareprojekts kann von Seiten des QM durch Vorgaben bezüglich zu verwendender Hard- und Software, der Festlegung einzuhaltender Normen und Standards, sowie einer Aufwandsabschätzung der durchzuführenden Testphasen unterstützt werden. Durch eine konsistente Einhaltung der Vorgaben des QM lassen sich nach Fertigstellung der Software verlässliche Aussagen bezüglich der Softwarequalität treffen.
Das folgende Kapitel bietet einen Überblick der Durchführung des QM der Projektgruppe „eXplorer“. Es gliedert sich dabei in folgende Abschnitte:

Abschnitt 15.1: Liefert einen kurzen Überblick des Inhalts dieses Kapitels.

Abschnitt 15.2: Durchführung von Verbindungstests, der zur Verfügung stehenden mobilen Endgeräte (Bluetooth, GPS, GPRS).

Abschnitt 15.3: Erstellung von Softwareprototypen, zur Ansteuerung verschiedener Funktionen der mobilen Endgeräte.

Abschnitt 15.4: Einführung in die testgetriebene Programmierung (NUnit), sowie in die Erstellung von Testausgaben (Logging).

Abschnitt 15.5: Planung, Durchführung und Protokollierung von Tests einzelner Anwendungsfälle.

Abschnitt 15.6: Planung, Durchführung und Protokollierung kompletter Spielszenarien.

15.2 Machbarkeitsstudie (Hardware)

Das primäre Ziel dieser Studie besteht darin, die Tauglichkeit der zur Verfügung stehenden Hardware, in Hinblick auf die Realisierung der Funktionalitäten des Agentenspiels zu untersuchen. Dadurch soll frühzeitig festgestellt werden, welche Funktionalitäten eventuell nicht oder nur eingeschränkt umsetzbar sein werden. Stehen mehrere Hardware-Lösungen zur Verfügung, besteht ein weiteres Ziel dieser Studie darin, die Vor- und Nachteile dieser Hardware-Lösungen einander gegenüberzustellen, um dadurch die geeignetste Lösung identifizieren zu können.

15.2.1 Bluetooth

Bluetooth ist ein freies, nicht lizenzpflichtiges Funkverfahren, um elektronische Geräte kabellos miteinander zu verbinden. Bluetooth nutzt das 2,4-GHz-ISM-Band, um Sprache, Daten und Bilder über kurze Entfernungen zu übertragen.
Bluetooth setzt auf so genannte drahtlose Ad-hoc-Pikonetze. Pikonetze sind lokale Netze, die üblicherweise eine geringe Ausdehnung haben und keine Infrastruktur benötigen. Jedes Gerät kann simultan mit bis zu sieben anderen Geräten innerhalb eines Pikonetzes kommunizieren. Das Ziel dieses Tests besteht darin, die Kompatibilität der für die Projektgruppe vpm OFFIS[1] zur Verfügung gestellten Hardware zu untersuchen und dessen Funktionsumfang zu testen.

15.2.1.1 Beschreibung des Tests

Alle Tests wurden mit der, von den Geräten zur Verfügung gestellten Softwareschnittstelle der Bluetoothfunktion durchgeführt.
Um eine Bluetoothverbindung unter den Geräten zu ermöglichen, müssen die einzelnen Geräte das „Suchen und Finden“ von Geräten und Diensten, sowie das „Koppeln“ von Geräten erlauben. Die Einstellungen sind grundsätzlich über einen Bluetoothmanager zu erledigen. Die genaue Durchführung ist vom Hersteller des jeweiligen Geräts abhängig.
Mit den eingesetzten Geräten wurde ein Pikonetz aufgebaut und mittels des *OBEX File Transfer*-Protokolls Dateien unter den Geräten ausgetauscht. Über das *SerialPort*-Protokoll wurde eine Verbindung zu einer GPS-Maus aufgebaut und mit zwei PDAs und einer GPS-Maus wurde eine Mehrfachverbindungen getestet.

Eingesetzte Geräte:

- PDA Dell Axim X30 Wireless-PAN/LAN
- PDA Hewlett Packard iPAQ 5450
- Mobile Telefon Siemens S65
- Mobile Telefon Sony Ericsson T610
- GPS-Maus Falcom NAVI-1

15.2.1.2 Fazit

Es traten keine nennenswerten Probleme beim Einsatz von Bluetooth auf. Eine sichere Verbindung der Geräte untereinander war bis zu einer Distanz von 10 Metern möglich. Der Datendurchsatz war abhängig

[1] Oldenburger Forschungs- und Entwicklungsinstitut für Informatikwerkzeuge und -systeme

von den eingesetzten Geräten und den Umgebungsbedingungen, lag aber in dem von Bluetooth zu erwartendem Rahmen von 500 k/bits. Einzig die Tatsache, dass die GPS-Maus nur eine Verbindung gleichzeitig halten kann, muss beachtet werden. Die Bluetooth-Technologie ist somit aus Hardwaresicht für die Projektgruppe vollständig einsetzbar.

15.2.2 GPS

Das Global Positioning System (GPS) ist ein satellitengestütztes Navigationssystem zur weltweiten Positionsbestimmung. GPS wurde vom US-Verteidigungsministerium entwickelt und wird von diesem auch betrieben. GPS ist seit April 1995 zivil nutzbar. Im Mai 2000 wurde die künstliche Ungenauigkeit bei allen Satelliten abgeschaltet, so dass die Messgenauigkeit nun in mindestens 90% der Messungen besser als 10 m ist.

15.2.2.1 Beschreibung des Tests

Mittels einer Bluetoothverbindung wurde eine GPS-Maus mit einem PDA gekoppelt. Für die Anzeige der GPS-Daten wurde das Programm *GPSInfo* der Firma *Navpoint* verwendet.

Eingesetzte Geräte:

- PDA Hewlett Packard iPAQ 5450
- GPS Maus Falcom NAVI-1

15.2.2.2 Fazit

Eine Verbindung konnte ohne Problem aufgebaut werden und die GPS-Maus wurde erfolgreich mit dem PDA gekoppelt. Die GPS-Technologie ist somit aus Hardwaresicht für die Projektgruppe vollständig einsetzbar. Eine Überprüfung der Genauigkeit der von der GPS-Maus gelieferten Positionsdaten war nicht möglich.

15.2.3 GPRS

Beim General Packet Radio Service (GPRS) dient zur paketorientierten Datenübertragung und stellt eine Erweiterung des GSM-Mobilfunk-Standards dar. GPRS ermöglicht bei Bündelung aller 8 möglichen GSM-Zeitschlitze eines Kanals eine theoretische Übertragungsrate von 171,2 kBit/s. In der Praxis ist die Anzahl der nutzbaren Zeitschlitze jedoch durch die Fähigkeit des eingesetzten Mobilgerätes (multislot capability) und der Netze begrenzt.

15.2.3.1 Beschreibung des Tests

Über eine Bluetoothverbindung wurde ein Handy mit einem PDA gekoppelt. Anschließend wurde eine Verbindung mit dem Internet mittels eines im Handy integrierten Softmodem und dem *Dial-up Networking*-Protokoll hergestellt und der Zugriff auf einen beliebigen Server über den Internet Explorer des PDA getestet.

Eingesetzte Geräte:

- PDA Hewlett Packard iPAQ 5450
- Mobile Telefon Siemens S65

15.2.3.2 Fazit

Eine Verbindung konnte aufgebaut werden und das Handy wurde erfolgreich mit dem PDA gekoppelt. Ebenso verlief die Verbindung mit dem Internet, über das im Handy integrierte Softmodem, ohne Probleme. Die zu Testzwecken ausgewählter Internetadresse *www.heise.de* konnte erreicht und die zugehörige Internetseite ohne Probleme heruntergeladen werden. Die GPRS-Technologie ist somit aus Hardwaresicht für die Projektgruppe vollständig einsetzbar.

15.3 Machbarkeitsstudie (Software)

Im Kapitel 12 (Technologiestudie) auf Seite 193 wurden unterschiedliche Technologien hinsichtlich ihrer Verwendungsmöglichkeit für das Agentenspiel untersucht. Das Ziel dieser Studien bestand darin, die Vor- und Nachteile der unterschiedlichen Technologien gegeneinander abzuwägen und so, die für die Umsetzung des Agentenspiels geeignetsten Technologien zu identifizieren.
Im folgenden Abschnitt werden die Ergebnisse der Technologiestudien aus Kapitel 12 in vertiefenden Machbarkeitsstudien näher untersucht, wobei das Hauptaugenmerk auf Seiten der Software liegt. Durch die Erstellung von Prototypen soll hierbei getestet werden, ob und mit welchem Aufwand die untersuchten Technologien für die Umsetzung des Agentenspiels genutzt werden können.

15.3.1 Native Programmierung

Viele der Geräteschnittstellen eines Pocket PCs sind nicht aus .NET heraus ansteuerbar. Die .NET Plattform bietet jedoch die Möglichkeit, nativen Code auszuführen. Man kann dabei auf die vorhandenen Bibliotheken des Betriebssystems zurückgreifen, aber auch Bibliotheken von Drittanbietern in das Projekt integrieren. Das schließt natürlich auch selbst erstellte, native Bibliotheken mit ein.
Im Fachjargon von .NET wird zwischen *managed* und *unmanaged* Code unterschieden. Bei *managed* Code handelt es sich um die *Microsoft Intermediate Language (MSIL)*, die von Visual Studio .NET beim Kompilieren der Hochsprachen (C#, VB, C++, J#) erzeugt wird. Sie wird von der *Common Language Runtime (CRL)*[2] interpretiert. *Unmanaged* Code ist dagegen ausführbare Maschinensprache, die (z.B. aus C++ Code) direkt für eine bestimmte Architektur erzeugt wird.

15.3.1.1 Beschreibung des Tests

Native Funktionen des Betriebssystems aufrufen Um native Funktionen aus bestehenden DLLs aufrufen zu können, muss eine Klasse `System.Runtime.InteropServices` importieren. Die Signaturen der nativen Funktionen müssen im .NET Code angegeben sein. In C# identifiziert das Schlüsselwort `extern` die Signatur einer nativen Methode. Über das Attribut `[DllImport(`“Name der DLL“`)]` muss der Name der DLL angegeben werden, welche die Funktion implementiert. Die durch die Signatur identifizierte, native Methode kann dann, wie alle anderen Funktionen auch, aus dem Code heraus aufgerufen werden. Prototyp 15.1 zeigt den zugehörigen Prototyp.

Native DLLs in ein .NET Projekt integrieren Ist eine DLL nicht Teil des Betriebssystems oder nicht anderweitig auf dem Zielgerät vorhanden, kann sie in ein bestehendes Visual Studio .NET Projekt eingebunden werden. Dazu muss man die Funktion „neues Element hinzufügen“ aufrufen, und die gewünschte DLL auswählen (siehe Abb.15.1). Die DLL wird nun beim Kompilieren des Projekts mit in die Zieldatei eingebunden.

Native DLLs selbst erstellen und verwalten Um nativen Code für Windows CE zu erzeugen, kann man auf die Entwicklungsumgebung eMbedded Visual C++ 4.0 (eVC) von Microsoft zurückgreifen. Dort kann man z.B. ein DLL-Projekt für die gewünschte Zielarchitektur anlegen, die DLL implementieren und anschließend in ein .NET Projekt importieren. Dieser Prozess kann jedoch sehr umständlich sein, weil die DLL für jede Architektur (z.B. WCE Emulator oder WCE x86) neu übersetzt und importiert werden muss. Ein Tutorial[3] von Adrian Stanley erläutert, wie man die Verwaltung des Quellcodes aus Visual

[2] http://www.microsoft.com/germany/msdn/library/net/DieNETCommonLanguageRuntime.mspx
[3] http://www.codeproject.com/netcf/MakeDemo.asp

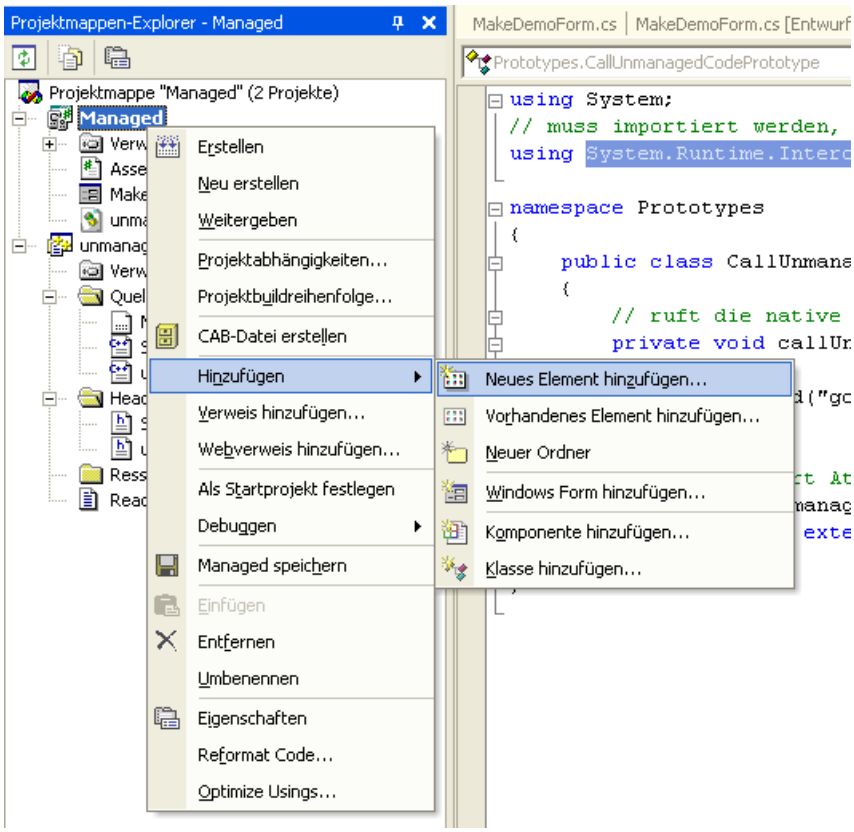


Abbildung 15.1: Hinzufügen einer DLL zu einem VS .NET Projekt

Studio .NET heraus gestalten kann und gleichzeitig die Übersetzung des eVC-Quellcodes von der in VS .NET angegebenen Zielarchitektur abhängig macht.

15.3.1.2 Fazit

Da die Funktionalität des .NET Compact Frameworks sehr begrenzt ist, wird die Projektgruppe vorhandene native Methoden nutzen müssen, um die in der Anforderungsdefinition geforderte Funktionalität zu erfüllen. Der Einsatz von nativem Code stellt jedoch kein Problem dar.

15.3.1.3 Prototypen

```
1  using System;
2  // muss importiert werden, damit native Funktionen aufgerufen werden öknnen
3  using System.Runtime.InteropServices;
4
5  namespace Prototypes
6  {
7     public class CallUnmanagedCodePrototype
8     {
9        // ruft die native Funktion auf
10       private void callUnmanagedFunction()
11       {
12          fnUnmanaged("go native!");
13       }
14
```

```
15        // das DllImport Attribut sagt, in welcher DLL die Funktion implementiert ist
16        [DllImport("Unmanaged.dll")]
17        private static extern int fnUnmanaged(string message);
18    }
19 }
```

Listing 15.1: Aufrufen nativer Funktionen

15.3.2 Kommunikation zwischen Java und C#

Ein Ergebnis der Technologiestudie (siehe Kapitel 12 auf Seite 193) bestand darin, dass die Software des Clients in der Sprache C# und die des Servers in der Sprache Java implementiert werden sollte.
Die Kommunikation zwischen dem Server und dem Client besteht in dem Serialisieren und Versenden von Objekten. Dabei werden die primitiven Datentypen eines Objekts von der sendenden Station in einen Datenstrom geschrieben und auf der Gegenseite von der empfangenden Station wieder ausgelesen. Um Probleme bei der Kommunikation zwischen diesen beiden Sprachen vor Beginn der Implementierungsphase zu erkennen, wurde ein einfacher Server in Java und ein einfacher Client in C# implementiert.

15.3.2.1 Beschreibung des Tests

Zu Beginn dieses Tests (vgl. hierzu Listing 15.2 (Server) auf Seite 334 und Listing 15.3 (Client) auf Seite 337) wird eine Verbindung zwischen dem Server und dem Client aufgebaut. Die Kommunikation zwischen den Stationen wird über Sockets realisiert. Auf Serverseite wird ein Serversocket initiiert, dem der Port bekannt ist, auf dem der Client das Objekt senden wird (in diesem Beispiel auf Port 5024). Der Client erzeugt einen Socket, dem zusätzlich zu dem Port, auf dem die Kommunikation stattfindet, die IP-Adresse des Servers mitgeteilt wird. Über diesen Socket sendet der Client ein *MyObject*-Objekt an den Server. Dieses Objekt enthält als primitive Datentypen einen Integer-, einen Double- und einen String-Wert. Um das Objekt zu verschicken, werden diese Datentypen in einen Datenstrom geschrieben. Der Server wartet auf das Eintreffen eines Datenstroms in einem eigenen *Thread*. Wird der Datenstrom empfangen, so wird die deserialize()-Methode aufgerufen, in welcher der Integer-, der Double- und der String-Wert wieder ausgelesen werden.

15.3.2.2 Fazit

Die Ergebnisse der Tests zeigen, dass die Kommunikation zwischen einem Server (in diesem Beispiel in Java geschrieben) und einem Client (in diesem Beispiel in C# geschrieben) zwar möglich, jedoch nur mit zusätzlichem Aufwand zu realisieren ist. Die unterschiedliche Kodierung von Zahlenwerten (Little- / Big-Endian) sowie eine Java-spezifische UTF8-Kodierung von Zeichenketten (C# verwendet eine standardisierte UTF8-Kodierung) erzwingt eine Kodierung bzw. Dekodierung empfangener Bytefolgen, sowohl auf der Server-, als auch auf Client-Seite. Dies stellt eine potentielle Fehlerquelle und einen zusätzlichen Implementierungsaufwand dar. Der Serialisierungsprozess, also das Schreiben und Lesen von primitiven Datentypen in bzw. aus einem Datenstrom, müsste neu geschrieben werden. Von der Umsetzung der Kommunikation zwischen einem Server und einem Client, bei dem eine Station in Java und die andere Station in C# implementiert wird, ist deshalb abzuraten.

15.3.2.3 Prototypen

```
1  /* Erstellt am 12.05.2005
2   */
3  package serializationTest;
4
5  import java.io.DataInputStream;
6  import java.io.IOException;
7  import java.io.UnsupportedEncodingException;
8  import java.net.ServerSocket;
9  import java.net.Socket;
10
11 /**
12  * @author Stefan Andressen, Frank Jellinghaus
13  */
14 public class SerializationTest {
15
16    public static void main(String[] args) {
17       Deserializer des = new Deserializer();
18       System.out.println("Horche␣jetzt␣auf␣Port␣"+des.getPort()+":␣");
19       des.start();
20    }
21 }
22
23 class MyObject {
24
25    private int myInt;
26    private double myDouble;
27    private String myString;
28
29    public MyObject() {
30       this.myInt = 0;
31       this.myDouble = 0.0;
32       this.myString = "";
33    }
34
35    public MyObject(int i, double d, String s) {
36       this.myInt = i;
37       this.myDouble = d;
38       this.myString = s;
39    }
40
41    /* Diese Methode bekommt einen DataInputStreamü bergeben,
42     * aus dem sie einen Int- und einen Double-Wert sowie einen
43     * String ausliest. Aufgrund der unterschiedlichen Kodierung
44     * der Zahlenwerte in Java (Big Endian) und C-Sharp (Little Endian)
45     * sowie der String-Kodierung (Java nutzt eine modifizierte
46     * UTF-8-Kodierung) ümssen noch Methoden zur Umkodierung aufge-
47     * rufen werden.
48     */
49    public void deserialize(DataInputStream in) {
50       try {
51          this.myInt = this.swapInt(in.readInt());
52          this.myDouble = this.swapDouble(in.readDouble());
53          this.myString = this.encodeString(in, "UnicodeBig");
54       } catch (IOException e) {
55          e.printStackTrace();
56       }
```

```
57    }
58
59    public String toString() {
60       return "MyInt: "+this.myInt
61             +"\nMyDouble: "+this.myDouble
62             +"\nMyString: "+this.myString+"\n";
63    }
64
65    /* Die swapXXX-Methoden sind üfr die Umkodierung der Zahlenwerte
66     * zustaendig, die encodeString-Methode üfr die Umkodierung einer
67     * Zeichenkette.
68     */
69    public short swapShort (short value) {
70      int b1 = value & 0xff;
71      int b2 = (value >> 8) & 0xff;
72
73      return (short) (b1 << 8 | b2 << 0);
74    }
75
76    public int swapInt (int value) {
77      int b1 = (value >>  0) & 0xff;
78      int b2 = (value >>  8) & 0xff;
79      int b3 = (value >> 16) & 0xff;
80      int b4 = (value >> 24) & 0xff;
81
82      return b1 << 24 | b2 << 16 | b3 << 8 | b4 << 0;
83    }
84
85    public long swapLong (long value) {
86      long b1 = (value >>  0) & 0xff;
87      long b2 = (value >>  8) & 0xff;
88      long b3 = (value >> 16) & 0xff;
89      long b4 = (value >> 24) & 0xff;
90      long b5 = (value >> 32) & 0xff;
91      long b6 = (value >> 40) & 0xff;
92      long b7 = (value >> 48) & 0xff;
93      long b8 = (value >> 56) & 0xff;
94
95      return b1 << 56 | b2 << 48 | b3 << 40 | b4 << 32 |
96           b5 << 24 | b6 << 16 | b7 <<  8 | b8 <<  0;
97    }
98
99    public float swapFloat (float value) {
100     int intValue = Float.floatToIntBits (value);
101     intValue = swapInt (intValue);
102     return Float.intBitsToFloat (intValue);
103   }
104
105   public double swapDouble (double value) {
106     long longValue = Double.doubleToLongBits (value);
107     longValue = swapLong (longValue);
108     return Double.longBitsToDouble (longValue);
109   }
110
111   public String encodeString(DataInputStream in,
112      String encoding) {
113
114      byte[] byteArray = null;
```

```
115         try {
116            int length = in.readInt();
117             byteArray = new byte[length];
118            in.readFully(byteArray);
119         } catch (IOException e) {
120            e.printStackTrace();
121         }
122         try {
123            return new String(byteArray, encoding);
124         } catch (UnsupportedEncodingException e1) {
125            e1.printStackTrace();
126         }
127         return null;
128      }
129   }
130
131   class Deserializer extends Thread {
132
133      private ServerSocket mySocket;
134      private int myPort;
135
136      private MyObject myObject;
137
138      public Deserializer() {
139         this.myPort = 5024;
140         this.myObject = new MyObject();
141         try {
142            this.mySocket = new ServerSocket(myPort);
143         } catch (IOException e) {
144            e.printStackTrace();
145         }
146      }
147
148      public int getPort() {
149         return this.myPort;
150      }
151
152      /* Beim Starten des Deserializer-Threads in der
153       * main()-Methode wird die run()-Methode aufgerufen.
154       */
155      public void run() {
156         Socket socket = null;
157         DataInputStream in = null;
158
159         try {
160            /* Das Server-Socket-Objekt wartet auf eine
161             * eingehende Verbindung, liest den InputStream
162             * der Socket-Verbindung ein und ruft die
163             * deserialize()-Methode des myObject-Objekts auf.
164             */
165            socket = mySocket.accept();
166
167            in = new DataInputStream(socket.getInputStream());
168
169            this.myObject.deserialize(in);
170
171            System.out.println(myObject);
172
```

```
173         } catch (IOException e) {
174             e.printStackTrace();
175         }
176     }
177 }
```

Listing 15.2: Server in Java

```
1  using System;
2  using System.IO;
3  using System.Text;
4  using System.Net;
5  using System.Net.Sockets;
6
7
8  namespace SerializedTest
9  {
10     class MyObject
11     {
12         private int m_integerTest;
13         private double m_doubleTest;
14         private String m_stringTest;
15
16         public MyObject(int integerTest, double doubleTest, String stringTest)
17         {
18             m_integerTest  = integerTest;
19             m_doubleTest   = doubleTest;
20             m_stringTest   = stringTest;
21         }
22
23         public void serialize(BinaryWriter writer)
24         {
25             Console.WriteLine("Beginne␣serialize");
26             writer.Write(m_integerTest); //int32 (little endian)
27             writer.Write(m_doubleTest);   //double64 (little endian)
28             byte[] dataArray = Encoding.BigEndianUnicode.GetBytes(m_stringTest); //bytes in
                   BigEndianUnicode
29             int length = IPAddress.HostToNetworkOrder(dataArray.Length);
30             writer.Write(length); //int32  (big endian)
31             writer.Write(dataArray, 0, dataArray.Length);  //char Data  (big endian)
32             Console.WriteLine("Beende␣serialize");
33         }
34
35         public override String ToString()
36         {
37             StringBuilder valueTest = new StringBuilder();
38             valueTest.Append("IntegerTest:␣");
39             valueTest.Append(m_integerTest);
40             valueTest.Append("\n");
41             valueTest.Append("DoubleTest:␣");
42             valueTest.Append(m_doubleTest);
43             valueTest.Append("\n");
44             valueTest.Append("StringTest:␣");
45             valueTest.Append(m_stringTest);
46             valueTest.Append("\n");
47             return valueTest.ToString();
48         }
```

```
49     }
50
51     class Serializer
52     {
53        BinaryWriter writer;
54        const String ipAddress = "192.168.0.29";
55        const int port = 5024;
56
57        public Serializer()
58        {
59           Console.WriteLine("Serializer");
60           TcpClient socket = new TcpClient(ipAddress, port);
61           NetworkStream networkStream = socket.GetStream();
62           writer = new BinaryWriter(networkStream, new UnicodeEncoding());
63        }
64
65        ~Serializer()
66        {
67           writer.Close(); //Destruktor
68        }
69
70        public void serialize(MyObject obj)
71        {
72           Console.WriteLine("Serialisiere:␣");
73           obj.ToString();
74           obj.serialize(writer);
75        }
76     }
77
78     class MainClass
79     {
80        static void Main(string[] args)
81        {
82           Console.WriteLine("Beginne␣Uebertragung!");
83           Serializer serializer = new Serializer();
84           MyObject obj = new MyObject(1, 3.4, "Hallo␣Welt!");
85           serializer.serialize(obj);
86           Console.Read(); //Warte auf Tastenanschlag
87        }
88     }
89  }
```

Listing 15.3: Client in C#

15.3.3 Bluetooth

Bluetooth ist ein Industriestandard für die drahtlose (Funk-)Vernetzung von Geräten über kurze Distanz. Bluetooth kann dabei sowohl in mobilen Kleingeräten, wie Mobiltelefonen und PDAs, als auch in Computern und Peripheriegeräten zum Einsatz kommen. Innerhalb dieses Softwaretests soll überprüft werden, ob und wie sich Bluetooth über das .NET Framework und das OpenNETCF ansteuern lässt.

15.3.3.1 Beschreibung des Tests

Mittels Bluetooth soll es den Clients ermöglicht werden, untereinander Daten auszutauschen und in kleineren Gruppen Netzwerke zu bilden.

Folgende Funktionen sollen mit dem Bluetoothmodul getestet werden:

- An- und Ausschalten des Moduls
- Test einer Verbindung mit einem unbekannten Client
- Test einer Verbindung mit einem bekannten Client
- Aufbau eines Ad-hoc-Netzwerks
- Aufbau eines Pikonetzes

15.3.3.2 Fazit

Die Tests konnten auf den, der Projektgruppe zur Verfügung gestellten Geräten (HP iPAQ 5450) nicht durchgeführt werden. Grund dafür ist der auf diesen Geräten eingesetzte Bluetooth Stack des Herstellers Widcomm/Broadcom. Sowohl das .NET Framework als auch das OpenNETCF unterstützen nur den Bluetooth Stack der Firma Microsoft. Einzig und allein das An- und Ausschalten des Bluetoothmoduls ist auch mit dem Widcomm/Broadcom Stack möglich. Hierfür muss eine von HP zur Verfügung gestellte API eingesetzt werden. Die volle Funktionalität des Widcomm/Broadcom Stack ist nur über kostenpflichtige *third-party*-APIs erreichbar und kommt somit für die Projektgruppe nicht in Frage.

15.3.3.3 Prototyp

```
1  [DllImport("iPAQUtil.dll", SetLastError=true)]
2
3  public static extern bool iPAQSetBlueToothRadio(ref int lpdwValue);
4
5
6  /* Der Aufruf erfolgt dann mit : */
7
8  int val = 0;
9  iPAQSetBlueToothRadio(ref val);
10
11 /* val kann dabei 0 zu deaktivieren oder 1 zum aktivieren von bluetooth sein. */
```

Listing 15.4: (De-)aktivieren des Blutoohmoduls

15.3.4 GPS

Es existiert eine Vielzahl von Dokumentationen im Internet, die den Empfang von GPS-Signalen und die Umwandlung der Rohdaten in der Sprache C# beschreiben. Zu diesen Dokumentationen ist in den meisten Fällen auch der Quellcode der zugehörigen Programme verfügbar. Dieser ist in der Regel jedoch sehr umfangreich. Die wesentlichen Schritte, die zur Realisierung einer GPS-Verbindung notwendig sind, sollen daher in dem folgenden Abschnitt näher erläutert werden.

15.3.4.1 Beschreibung des Tests

Der GPS-Empfänger soll aus C# heraus über eine Bluetooth-Verbindung angesteuert werden können. Die Kommunikation zwischen Empfänger und Rechner erfolgt über einen (virtuellen) seriellen Port. Um einen Port über Bluetooth erstellen zu können, muss das OpenNETCF[4] auf dem Rechner installiert sein. Der Namensraum *Serial*[5] des OpenNETCF muss importiert werden, um einen Bluetooth-Port erzeugen zu können. Dabei müssen folgende Parameter beachtet werden:

- ein DetailedPortSettings-Objekt wird mit new HandshakeNone(); erzeugt.
- dem Port-Objekt werden zwei Parameter übergeben: der erste Parameter enthält die Port-Nummer der Bluetooth-Verbindung als String-Repräsentation (bsp.: „COM4“) und der zweite Parameter enthält das zuvor initialisierte DetailedPortSettings-Objekt.

Für dieses Port-Objekt müssen noch folgende Einstellungen vorgenommen werden:

- RThreshold: gibt die Anzahl der empfangenen Bytes an, nach der ein Event erzeugt wird
- InputLen: gibt die Anzahl der Bytes an, die nach dem Auftreten eines Events aus dem „Input“-Puffer gelesen werden
- SThreshold: gibt die Anzahl der Bytes an, die zusammen übertragen werden
- DataReceived: registriert einen EventListener auf dem Com-Port und ruft die als Parameter spezifizierte Methode auf, sobald ein Event auftritt

Nach dem Setzen dieser Einstellungen wird mit dem Aufruf der Methode „port.open();“ die Bluetooth-Verbindung zwischen Rechner und GPS-Empfänger hergestellt und geöffnet. Erhält der GPS-Empfänger korrekte Signale (die Initialisierungszeit kann stark variieren, je nachdem ob der Empfänger eine ungestörte Sichtverbindung zum Himmel hat, ob Gebäudewände o.Ä. dazwischen liegen, etc.) erzeugt dieser nach einer bestimmten Anzahl empfangener Bytes (die Anzahl wird über die Einstellung RThreshold vorgenommen) ein Event. Durch das Eintreten dieses Events wird die Methode port_DataReceived aufgerufen, in der die Bytes vom Com-Port ausgelesen werden. Danach müssen noch folgende Schritte ausgeführt werden:

1. Umwandeln der Bytes in einen String (GetString()-Methode)
2. Aufteilen des Strings in Teilstrings. Das Ende eines NMEA[6]-Strings wird durch ein Zeilenschaltungszeichen signalisiert.

[4] Verfügbar unter http://www.opennetcf.org/
[5] OpenNETCF.IO.Serial
[6] National Marine Electronics Association

3. Danach müssen die Teilstrings zu kompletten NMEA-Strings zusammengesetzt werden (die NMEA-Strings haben nicht alle die gleiche Länge, deswegen muss überprüft werden, an welcher Stelle sie beginnen und an welcher sie enden).

4. Aufrufen der Methoden, die aus den NMEA-Strings die wichtigen Informationen extrahieren können (die Informationen stehen in einem NMEA-String an einer fest definierten Stelle, die Stellen durch Kommata getrennt)

15.3.4.2 Fazit

Einzige Voraussetzung für die Ansteuerung eines GPS-Empfängers aus C# heraus ist die Installation des OpenNETCF. Das Extrahieren der Informationen ist in wenigen Schritten ausführbar. Für die Empfangsrate der Strings sind die initialen Einstellungen des Com-Ports verantwortlich. Die geeignetsten Einstellungen gilt es durch Ausprobieren herauszufinden.

15.3.4.3 Prototyp

```
1  using System;
2  using System.IO;
3  using OpenNETCF.IO.Serial;
4
5  using System.Collections;
6  using System.Data;
7  using System.Text;
8
9  namespace SerialTest
10 {
11    /* Dieser Prototyp ist eine Konsolenanwendung, die eine Bluetooth-Verbindung mit einem
12     * GPS-äEmpfnger herstellt, die empfangenen Daten weiterverarbeitet und sie
13     * letztednlich auf der Konsole ausgibt.
14     */
15    class GPS
16    {
17       // Umrechnungskonstante (die Geschwindigkeit wird in Knoten geliefert)
18       const double KNOTS_TO_KMH = 1.852;
19
20       private static String nmeaSentence = "";
21       private static String nmeaTemp;
22       private static String[] temp;
23       private static Port comport;
24
25       [STAThread]
26       static void Main(string[] args)
27       {
28          // Die Verbindung muss nicht per Handshake initiiert werden.
29          DetailedPortSettings ps = new HandshakeNone();
30
31             // Eingabe der Portnummer
32          String port;
33
34          System.Console.Write("\nWelcher␣Port␣soll␣ögeffnet␣werden?␣Port:␣");
35          String porteingabe = System.Console.ReadLine();
36          System.Console.WriteLine("");
37
38          port = "COM"+porteingabe+":";
```

```
39
40          // Initialisierung des Ports
41          comport = new Port(port, ps);
42
43          // Allgemeine Port-Einstellungen:
44          // nach 82 Byte wird ein Event erzeugt.
45          comport.RThreshold = 82;
46          // 82 byte werden beim Aufruf von Input
47          // aus dem Puffer gelesen.
48          comport.InputLen = 82;
49          comport.SThreshold = 82;
50          // EventListener registrieren.
51          comport.DataReceived +=new Port.CommEvent(SerialTest.Class1.port_DataReceived);
52
53          comport.Open(); // Portö ffnen
54
55          // Bricht das Empfangen der
56          // Daten von der GPS-Maus ab ...
57          String eingabe = System.Console.ReadLine();
58          comport.Close(); // ... und ßschliet den Port.
59      }
60
61      public static void port_DataReceived()
62      {
63          // äAbhngig von den Port-Einstellungen muss eine Anzahl an Bytes gelesen werden.
64          byte[] inputData = new byte[82];
65
66          // Daten lesen.
67          inputData = comport.Input;
68
69          // Umwandlung der Bytes in Strings.
70          Encoding enc = Encoding.ASCII;
71          nmeaTemp = enc.GetString(inputData, 0, inputData.Length);
72
73          // Trennen der Strings in Teilstrings. Jeder GPS-String wird durch das
74          // Zeilenschaltungszeichen abgeschlossen.
75          temp = nmeaTemp.Split('\n');
76
77          // Die Teilstrings ümssen zu ävollstndigen GPS-Strings üzusammengefgt werden.
78          for(int i=0; i<temp.Length; i++)
79          {
80              if(temp[i].StartsWith("$"))
81              {
82                  nmeaSentence = temp[i];
83              }
84              else
85              {
86                  nmeaSentence += temp[i];
87              }
88
89              // Wird ein GPS-String als ganzer String identifiziert, dann kann er
90              // durch den Aufruf der folgenden Methoden weiterverarbeitet werden.
91              if(nmeaSentence.EndsWith("\r"))
92              {
93                  if(nmeaSentence.StartsWith("$GPRMC"))
94                  {
95                      string[] position = Class1.ParseGPRMC(nmeaSentence);
96                      System.Console.WriteLine("\nGPRMC:");
```

```
97                  System.Console.WriteLine("======");
98                  System.Console.WriteLine("äLnge:␣"+position[0]);
99                  System.Console.WriteLine("Breite:␣"+position[1]+"\n");
100                 System.Console.WriteLine("Geschwindigkeit:␣"+position[2]+"\n");
101             }
102
103             if(nmeaSentence.StartsWith("$GPGGA"))
104             {
105                 string[] position = Class1.ParseGPGGA(nmeaSentence);
106                 System.Console.WriteLine("\nGPGGA:");
107                 System.Console.WriteLine("======");
108                 System.Console.WriteLine("äLnge:␣"+position[0]);
109                 System.Console.WriteLine("Breite:␣"+position[1]+"\n");
110                 System.Console.WriteLine("öHhe:␣"+position[2]+"\n");
111             }
112         }
113     }
114 }
115
116 /* In den beiden folgenden Methoden werden die Informationen der GPS-Strings
117  * (äabhngig von der Startsequenz, in diesen äFllen $GPRMC / $GPGGA) in
118  * Teilstrings unterteilt (Trennzeichen innerhalb der String ist ',') und in
119  * ein String-Array geschrieben.
120  * Die Positionen der Informationen innerhalb eines GPS-Strings sind fest
121  * definiert, ist eine Information innerhalb des Strings nicht üverfgbar wird
122  * dies durch zwei aufeinanderfolgende Kommata signalisiert. Dadurch wird die
123  * Reihenfolge der Informationen im Array nicht verschoben, so dass relevante
124  * Informationen direkt durch das Auslesen einer bestimmten Stelle des Arrays
125  * erhalten werden öknnen.
126  */
127
128 /* Der $GPGGA-String äenthlt u.a. die Angaben (geographische) äLnge, Breite
129  * und öHhe.
130  */
131 public static string[] ParseGPGGA(String nmeaSentence)
132 {
133
134     string[] Words = GetWords(nmeaSentence);
135
136     string[] position = new String[3];
137
138     if(Words[2] != "" & Words[3] != "" & Words[4] != "" & Words[5] != ""
139         & Words[9] != "" & Words[10] != "")
140     {
141         string Latitude = Words[2].Substring(0, 2) + "°";
142         Latitude = Latitude + Words[2].Substring(2) + "\"";
143         Latitude = Latitude + Words[3];
144
145         string Longitude = Words[4].Substring(0, 3) + "°";
146         Longitude = Longitude + Words[4].Substring(3) + "\"";
147         Longitude = Longitude + Words[5];
148
149             string Altitude = Words[9] + "␣" + Words[10];
150
151         position[0] = Latitude;
152         position[1] = Longitude;
153         position[2] = Altitude;
154     }
```

```
155
156         return position;
157      }
158
159      /* Der $GPRMC-String äenthlt u.a. die Angaben (geographische) äLnge, Breite
160       * und die Geschwindigkeit.
161       */
162      public static String[] ParseGPRMC(String nmeaSentence)
163      {
164         string[] Words = GetWords(nmeaSentence);
165
166         string[] position = new String[3];
167
168         if (Words[3] != "" & Words[4] != "" & Words[5] != "" & Words[6] != ""
169            & Words[7] != "")
170         {
171
172            string Latitude = Words[3].Substring(0, 2) + "°";
173            Latitude = Latitude + Words[3].Substring(2) + "\"";
174            Latitude = Latitude + Words[4];
175
176            string Longitude = Words[5].Substring(0, 3) + "°";
177            Longitude = Longitude + Words[5].Substring(3) + "\"";
178            Longitude = Longitude + Words[6];
179
180            double temp = Double.Parse(Words[7].Replace(".", ","))
181               * KNOTS_TO_KMH;
182
183            string Speed = temp + "␣Km/h";
184
185            position[0] = Latitude;
186            position[1] = Longitude;
187            position[2] = Speed;
188         }
189
190         return position;
191      }
192
193      /* Eine Hilfsmethode, die einen String in Teilstrings zerlegt und diese
194       * in Form eines String-Arrays üzurckliefert.
195       */
196      public static String[] GetWords(String sentence)
197      {
198         return sentence.Split(',');
199      }
200   }
201 }
```

Listing 15.5: Prototyp für den Aufbau einer GPS-Verbindung über einen Bluetooth-Port

15.3.5 Internetverbindung

Die verfügbaren PDAs (HP iPAQ 5450) enthalten ein WLAN-Modul mit dem eine Verbindung ins Internet möglich ist. Jedoch reicht eine reine WLAN-Verbindung für die Zwecke der Projektgruppe nicht aus, da die Verfügbarkeit von WLAN-Hotspots nicht ausreichend gesichert ist. Alternativ ist eine Einwahl zu einem Internetprovider mittels eines GPRS-fähigen Handys möglich. Dieses kann als Modem genutzt und über Bluetooth mit dem PDA verbunden werden.

15.3.5.1 Beschreibung des Tests

Der Test der Internetverbindung ist in zwei Teilaufgaben gegliedert. Im ersten Teil wird die Verbindung vom PDA in das Internet über ein Handy (das als Modem fungiert) getestet. Der zweite Teil beschreibt die Verwendung von Sockets, um mit dem .NET Compact Framework Daten mittels einer bestehenden Verbindung an einen Server zu verschicken.

Einwahl über ein Handy Als erstes müssen die Bluetoothschnittstellen der beiden Geräte (Handy und PDA) aktiviert werden. Das Handy kann dann auf dem PDA über den Bluetooth Verbindungsmanager von Windows CE als bekanntes Gerät eingetragen und eine Modemverbindung hergestellt werden. Die Einwahl in das Internet erfolgt über einen Provider, dessen Telefonnummer in den Einstellungen der Verbindung angegeben wird. Im Test wurde der InternetService von Vodafone gewählt. Nach der Einwahl stehen die gängigen Internetdienste zur Verfügung, z.B. können Internetseiten wie `www.google.de` im Internet Explorer von Windows CE angezeigt werden.

Sockets im .NET Compact Framework Dieser Test fand unter Zuhilfenahme einer Beispielapplikation[7] des *msdn Smart Client Developer Center* statt. Die Beispielapplikation schickt eine handschriftliche Signatur vom PDA an einen Server, welcher die Unterschrift prüft und das Ergebnis an den PDA zurückschickt. Seitens des PDAs erfolgt die Verbindung über asynchrone Sockets um die Steuerbarkeit der Anwendung während des Datentransfers nicht zu beeinträchtigen. Die Verbindungslogik ist in einer eigenen Klasse `ClientSocket` gekapselt. Der Quellcode ist unter dem Punkt „Prototyp“ zu finden.

15.3.5.2 Fazit

Sowohl die Verbindung zu einem Internet Service Provider (ISP) als auch die Kommunikation über Sockets ist problemlos möglich. Es entstehen jedoch je nach Wahl des ISPs Verbindungskosten. Im Falle des InternetService von Vodafone fallen tagsüber Gebühren von 0,19 Euro pro Minute an.

15.3.5.3 Prototyp

```
1 /*
2    NotifyCommand enum
3    --
4    List of socket notification commands.
5
6    ClientSocket class
7    --
8    Connects and sends data over TCP sockets. Uses async sockets so a delegate
9    method is called when the socket command completes. Raises an Notify event
```

[7] http://msdn.microsoft.com/smartclient/codesamples/default.aspx#cf

```
10      when socket operations complete and passes any associated data with the event.
11  */
12
13  using System;
14  using System.Net;
15  using System.Net.Sockets;
16  using System.Threading;
17  using System.IO;
18  using System.Text;
19  using System.Configuration;
20  using System.Collections;
21  using System.Diagnostics;
22  using Common;
23
24  namespace PocketSignature
25  {
26      /// <summary>
27      /// Socket notification commands.
28      /// </summary>
29      public enum NotifyCommand
30      {
31          Connected,
32          SentData,
33          ReceivedData,
34          Error
35      }
36
37      /// <summary>
38      /// Use async sockets to connect, send and receive data over TCP sockets.
39      /// </summary>
40      public class ClientSocket
41      {
42          // notification event
43          public delegate void NotifyEventHandler(NotifyCommand command, object data);
44          public event NotifyEventHandler Notify;
45
46          // socket that exchanges information with server
47          Socket _socket;
48
49          // holds information sent back from server
50          byte[] _readBuffer = new byte[1];
51
52          // used to synchronize the shutdown process, terminate
53          // any pending async calls before Disconnect returns
54          ManualResetEvent _asyncEvent = new ManualResetEvent(true);
55          bool _disconnecting = false;
56
57          // async callbacks
58          AsyncCallback _connectCallback, _sendCallback, _receiveCallback;
59
60          /// <summary>
61          /// Returns true if the socket is connected to the server. The property
62          /// Socket.Connected does not always indicate if the socket is currently
63          /// connected, this polls the socket to determine the latest connection state.
64          /// </summary>
65          public bool Connected
66          {
67              get
```

```
68          {
69              // return right away if have not created socket
70              if (_socket == null)
71                  return false;
72
73              // the socket is not connected if the Connected property is false
74              if (!_socket.Connected)
75                  return false;
76
77              // there is no guarantee that the socket is connected even if the
78              // Connected property is true
79              try
80              {
81                  // poll for error to see if socket is connected
82                  return !_socket.Poll(1, SelectMode.SelectError);
83              }
84              catch
85              {
86                  return false;
87              }
88          }
89      }
90
91      public ClientSocket()
92      {
93          // hookup async callbacks
94          _connectCallback = new AsyncCallback(ConnectCallback);
95          _sendCallback = new AsyncCallback(SendCallback);
96          _receiveCallback = new AsyncCallback(ReceiveCallback);
97      }
98
99      /// <summary>
100     /// Connect to the specified address and port number.
101     /// </summary>
102     public void Connect(string address, int port)
103     {
104         // make sure disconnected
105         Disconnect();
106
107         // connect to the server
108         IPAddress ipAddress = IPAddress.Parse(address);
109         IPEndPoint endPoint = new IPEndPoint(ipAddress, port);
110         _socket = new Socket(AddressFamily.InterNetwork,
111             SocketType.Stream, ProtocolType.Tcp);
112
113         _asyncEvent.Reset();
114         _socket.BeginConnect(endPoint, _connectCallback, null);
115     }
116
117     /// <summary>
118     /// Disconnect from the server.
119     /// </summary>
120     public void Disconnect()
121     {
122         // return right away if have not created socket
123         if (_socket == null)
124             return;
125
```

```
126             // set this flag so we don't raise any error notification
127             // events when disconnecting
128             _disconnecting = true;
129 
130             try
131             {
132                 // first, shutdown the socket
133                 _socket.Shutdown(SocketShutdown.Both);
134             }
135             catch {}
136 
137             try
138             {
139                 // next, close the socket which terminates any pending
140                 // async operations
141                 _socket.Close();
142 
143                 // wait for any async operations to complete
144                 _asyncEvent.WaitOne();
145             }
146             catch {}
147 
148             _disconnecting = false;
149         }
150 
151         /// <summary>
152         /// Send data to the server.
153         /// </summary>
154         public void Send(byte[] data)
155         {
156             // send the data
157             _socket.BeginSend(data, 0, data.Length,
158                 SocketFlags.None, null, null);
159 
160             // send the terminator
161             _asyncEvent.Reset();
162             _socket.BeginSend(Network.TerminatorBytes, 0,
163                 Network.TerminatorBytes.Length,
164                 SocketFlags.None, _sendCallback, true);
165         }
166 
167         /// <summary>
168         /// Read data from server.
169         /// </summary>
170         public void Receive()
171         {
172             _asyncEvent.Reset();
173             _socket.BeginReceive(_readBuffer, 0, _readBuffer.Length,
174                 SocketFlags.None, _receiveCallback, null);
175         }
176 
177         /// <summary>
178         /// Raise notification event.
179         /// </summary>
180         private void RaiseNotifyEvent(NotifyCommand command, object data)
181         {
182             // the async operation has completed
183             _asyncEvent.Set();
```

```
184
185             // don't raise notification events when disconnecting
186             if ((this.Notify != null) && !_disconnecting)
187                 Notify(command, data);
188         }
189
190         //
191         // async callbacks
192         //
193         private void ConnectCallback(IAsyncResult ar)
194         {
195             try
196             {
197                 // pass connection status with event
198                 _socket.EndConnect(ar);
199                 RaiseNotifyEvent(NotifyCommand.Connected, this.Connected);
200             }
201             catch (Exception ex)
202             {
203                 RaiseNotifyEvent(NotifyCommand.Error, ex.Message);
204             }
205         }
206
207         private void SendCallback(IAsyncResult ar)
208         {
209             try
210             {
211                 _socket.EndSend(ar);
212                 RaiseNotifyEvent(NotifyCommand.SentData, null);
213             }
214             catch (Exception ex)
215             {
216                 RaiseNotifyEvent(NotifyCommand.Error, ex.Message);
217             }
218         }
219
220         private void ReceiveCallback(IAsyncResult ar)
221         {
222             try
223             {
224                 _socket.EndReceive(ar);
225
226                 // the server sends an acknowledgment back
227                 // that is stored in the read buffer
228                 RaiseNotifyEvent(NotifyCommand.ReceivedData,
229                     _readBuffer[0] == 1 ? true : false);
230             }
231             catch (Exception ex)
232             {
233                 RaiseNotifyEvent(NotifyCommand.Error, ex.Message);
234             }
235         }
236     }
237 }
```

Listing 15.6: Die Klasse `ClientSocket`

15.3.6 Kartendarstellung

Dieser Abschnitt beschäftigt sich mit der Darstellung der Umgebung auf dem PDA. Zu der Umgebung gehören zum Einen die Karte und zum Anderen die Points of Interest. In dieser Machbarkeitsstudie wurden existierende Kartendarstellungsverfahren, sowie allgemeine Probleme im Umgang mit der Nutzungsschnittstelle des zur Verfügung stehenden PDA-Typs (HP iPAQ 5450) untersucht.
Für die Darstellung der Karte bieten sich folgende Möglichkeiten an:

- **Vektordaten**: Karten im Vektorformat haben einige Vorteile. Die Datenmenge ist oft geringer, als bei Rasterdaten. Vektorgrafiken lassen sich darüber hinaus stufenlos skalieren. Problematisch ist hierbei jedoch die Weiterverarbeitung auf dem PDA, da dieser nur mit Bitmap-Grafiken zusammenarbeitet.

- **Bitmap**: Karten, die als Bitmaps vorliegen, können problemlos von den APIs des PDAs weiterverarbeitet werden. Nachteilig ist hier jedoch, dass man die Fähigkeit verliert, stufenlos zu skalieren.

- **Akustisch/Haptisch**: Neben der visuellen wäre zusätzlich auch eine akustische oder haptische Darstellung der Karte denkbar. Beide Möglichkeiten wurden in der Projektgruppe diskutiert. Die Idee einer akustischen Darstellung der Karte wurde aufgrund einer möglichen Überladung des akustischen Kanals verworfen. Der akustische Kanal ist mit der Darstellung von Ereignissen und Zuständen bereits ausgelastet. Die Möglichkeit einer haptischen Darstellung der Karte, bspw. durch Ansteuerung des Vibrationsalarms wurde aus dem gleichen Grund zunächst nicht weiter verfolgt.

Zur Darstellung der Points of Interest wurde überlegt, die in der Nutzungsschnittstellen-API des PDAs vorhandenen `Button`-Objekte zu verwenden. So könnte der Nutzer mit den Points of Interest interagieren.

15.3.6.1 Beschreibung des Tests

Nutzungsschnittstelle Allgemein Zunächst wurde getestet, ob die Nutzungsschnittstelle mit der API des PDAs, so wie im Entwurf geplant, umgesetzt werden kann. Hierzu wurde ein GUI-Prototyp erstellt, in dem mehrere komplexe Panels (ControlPanel, etc.) realisiert wurden. Hierbei traten keine nennenswerten Probleme auf.

Darstellung der Karte durch eine Bitmap In einem weiteren Prototyp wurde die Möglichkeit getestet, eine Bitmap als Karte zu verwenden. Dieser Prototyp ermöglicht die Anzeige einer Bitmap, die größer ist als das Display des PDAs. Über vier Knöpfe für die einzelnen Richtungen lässt sich die angezeigte Bitmap nach Belieben verschieben. Zusätzlich ist es gelungen, eine Zentrierung der angezeigten Bitmap auf eine bestimmte Stelle des Bildes zu ermöglichen. Hierzu muss auf die zu zentrierende Position der Bitmap geklickt werden.

Button-Objekte als Points of Interests Darüber hinaus wurde getestet, ob Buttons zur Darstellung von Points of Interests geeignet sind. In dem Prototyp wurden extrem kleine Buttons verwendet, die keine Beschriftung hatten, sich aber durch unterschiedliche Färbung voneinander unterschieden. Außerdem wurde getestet, ob die Position der Buttons zur Laufzeit geändert werden kann.

15.3.6.2 Fazit

Die Tests verliefen erfolgreich. Eine Darstellung der Karte durch eine Bitmap ist somit problemlos möglich.

15.3.7 Lichtsensor

Obwohl ein Großteil der heute eingesetzten PDAs über transflektive Displays verfügt, sind diese dennoch anfällig für direkte Lichteinstrahlung. Es ist daher wünschenswert, die Helligkeit des Displays dem Umgebungslicht anzupassen, um somit eine optimale Darstellung zu garantieren.
Alle der uns zur Verfügung stehenden PDAs des Typs HP iPAQ 5450 haben einen eingebauten Lichtsensor. Es soll getestet werden ob es möglich ist, über diesen Lichtsensor das Umgebungslicht zu messen.

Fazit

Es ist nicht möglich den Lichtsensor für die Zwecke der Projektgruppe einzusetzen. Der Lichtsensor scheint nur direkt mit der Beleuchtungseinheit und dem Display zu kommunizieren. Eine Abfrage der Lichtwerte ist somit nicht möglich. Allein der Zustand der Automatik, ob eingeschaltet oder ausgeschaltet, kann vom Betriebssystem abgefragt und gesetzt werden. Diese Funktion kann aber nur über das iPAQ SDK erreicht werden, dessen Benutzung kostenpflichtig ist.

15.3.8 Abspielen von Samples

Bei Samples handelt es sich um digital aufgezeichnete Wellenformen, die akustische Signale beliebiger Art speichern können (Sprache, Musik, Naturgeräusche, etc.). Das .NET Compact Framework unterstützt die Wiedergabe von Samples nicht, so dass man auf native Bibliotheken zurückgreifen muss.

15.3.8.1 Beschreibung des Tests

WCE_PlaySound Prototyp 15.7 verwendet die native Windows CE Funktion `WCE_PlaySound` aus der Bibliothek `CoreDll.DLL` um PCM Wave files[8] aus einer Datei sowie aus dem Hauptspeicher abzuspielen. Nachteil dieser Methode ist, dass `WCE_PlaySound` lediglich ein Sample zu einem Zeitpunkt abspielen kann.

OpenNet.Multimedia.Audio.Player Prototyp 15.8 verwendet die Klasse Player aus dem OpenNet SmartDeviceFramework[9] um Wave PCM Dateien abzuspielen. Jede `Player`-Klasse kann ein Sample zu einer Zeit abspielen, aber es können mehrere Player gleichzeitig Samples abspielen.

15.3.8.2 Fazit

Für die Projektgruppe empfiehlt sich der Einsatz des `OpenNet.Multimedia.Audio.Player`s, da dieser im Gegensatz zu der nativen Methode `WCE_PlaySound` mehrere Samples gleichzeitig abspielen kann. Das ist im Hinblick auf die Realisierung der Nutzungsschnittstelle sehr wichtig.

[8] http://www.sonicspot.com/guide/wavefiles.html
[9] http://www.opennetcf.org/

15.3.8.3 Prototypen

```
using System;
using System.IO;
using System.Runtime.InteropServices;

namespace SoundTest
{
   public class Samples
   {
      private enum Flags
      {
         SND_SYNC = 0x0000, /* play synchronously (default) */
         SND_ASYNC = 0x0001, /* play asynchronously */
         SND_NODEFAULT = 0x0002, /* silence (!default) if sound not found */
         SND_MEMORY = 0x0004, /* pszSound points to a memory file */
         SND_LOOP = 0x0008, /* loop the sound until next sndPlaySound */
         SND_NOSTOP = 0x0010, /* don't stop any currently playing sound */
         SND_NOWAIT = 0x00002000, /* don't wait if the driver is busy */
         SND_ALIAS = 0x00010000, /* name is a registry alias */
         SND_ALIAS_ID = 0x00110000, /* alias is a predefined ID */
         SND_FILENAME = 0x00020000, /* name is file name */
         SND_RESOURCE = 0x00040004 /* name is resource name or atom */
      }

      [DllImport("CoreDll.DLL", EntryPoint="PlaySound", SetLastError=true)]
      private static extern int WCE_PlaySound(string szSound, IntPtr hMod, int flags);

      [DllImport("CoreDll.DLL", EntryPoint="PlaySound", SetLastError=true)]
      private static extern int WCE_PlaySoundBytes(byte[] szSound, IntPtr hMod, int flags
          );


      public void Play(string sampleFile)
      {
         WCE_PlaySound(sampleFile, IntPtr.Zero, (int) (Flags.SND_ASYNC | Flags.
             SND_FILENAME));
      }

      public void Play(Stream stream)
      {
         byte[] sample = new byte[stream.Length];
         stream.Read(sample, 0, (int) stream.Length);
         WCE_PlaySoundBytes(sample, IntPtr.Zero, (int) (Flags.SND_ASYNC | Flags.
             SND_MEMORY));
      }
   }
}
```

Listing 15.7: Abspielen von Samples via WCE_PlaySound

```
1  using System.IO;
2  using OpenNETCF.Multimedia.Audio;
3
4  namespace OpenNetSound
5  {
6     public class OpenNetSoundPrototype
7     {
8        public void play(string soundFile)
9        {
10          Player audioPlayer = new Player();
11          FileStream soundStream = File.OpenRead(@"\windows\alarm1.wav");
12          audioPlayer.Play(soundStream);
13       }
14    }
15 }
```

Listing 15.8: Abspielen von Samples via OpenNet Audio Player

15.3.9 Klangerzeugung

Bibliotheken zur Klangerzeugung sind nicht Bestandteil des .NET Compact Frameworks. Sounds mit einfachen Wellenformen, wie z.B. Sinus, Rechteck, Dreieck oder Sägezahn könnte man allerdings auch selbst berechnen, als Samples speichern und dann als solche abspielen.

Fazit

Der momentane Entwurf der Nutzungsschnittstelle sieht den Einsatz von durch Klangerzeugern produzierten Tönen nicht vor. Gleichzeitig ist die Realisierung eines Klangerzeugers sehr aufwändig und kann daher nicht für die Projektgruppe empfohlen werden.

15.3.10 Sprachausgabe/Spracherkennung

Unter den Stichwörtern *text-to-speech (TTS)* bzw. *speech-recognition (SR)* findet man im Internet Informationen zu Sprachausgabe und Spracherkennung. Microsoft stellt für das .NET Compact Framework die Speech API (SAPI), eine Schnittstelle zur Einbindung von TTS und SR Software von Drittanbietern zur Verfügung. Die zugehörige DLL ist jedoch auf Windows Mobile standardmäßig nicht vorhanden, sondern wird von den TTS-/SR-Anwendungen mitgeliefert.

15.3.10.1 Beschreibung der Recherche

Existierende Bibliotheken In einem Chat[10] vom 15.01.2004 behauptet ein Mitarbeiter von Microsoft, dass es zur Zeit nur sehr teure, kostenpflichtige Implementierungen der SAPI gäbe. Die einzige auffindbare, freie Bibliothek, die in ihrer Beschreibung Windows CE als Zielplattform angibt, heißt flite[11]. Zur Zeit liegt jedoch nur der Quellcode in der Sprache C vor, d.h. eine Anpassung an das .NET Compact Framework könnte extrem zeitaufwändig sein.

[10] http://msdn.microsoft.com/chats/transcripts/mobileembedded/embedded_011504.aspx
[11] http://www.speech.cs.cmu.edu/flite/index.html

Eigenrealisierung Unter http://www.mperfect.net/ttSpeech/ befindet sich ein Artikel über den Versuch des Autors, eine simple .NET Compact Framework Bibliothek zur Synthese von Sprache in Eigenregie zu erstellen. Er geht dabei auf alle Schritte der Implementierung ein, veröffentlicht jedoch keinen Quellcode. Es bleibt jedoch die Option, die Schritte nachzuvollziehen, und eine eigene Bibliothek zur Sprachsynthese zu schreiben. In einem anderen Artikel mit analogem Aufbau, geht der Autor auch auf die Erkennung von Sprache mittels Software ein. Wichtig zu erwähnen ist auch, dass die von dem Autor implementierten Prototypen einige Minuten zum initialisieren brauchten und sehr viel Speicherplatz belegten.

15.3.10.2 Fazit

Sowohl das Anpassen existierender Bibliotheken, als auch die Eigenentwicklung von TTA und SR sind nicht trivial. Beide Vorgänge sind extrem Ressourcen-intensiv. Die Projektgruppe sollte daher zunächst genau prüfen, ob sie die dafür nötige Zeit in die Entwicklung eines Sprachsynthese- und Spracherkennungsmoduls investieren will.

15.3.11 Mikrofon

Innerhalb dieser Machbarkeitsstudie soll überprüft werden, ob und wie sich das Mikrofon eines PDA über das .NET Framework und das OpenNETCF ansprechen und steuern lässt.

15.3.11.1 Beschreibung des Tests

Der folgende Prototyp verwendet die Klasse *Recorder* aus dem OpenNet SmartDeviceFramework[12] um Wave PCM Dateien aufzunehmen. Um diese Klasse nutzen zu können, muss das OpenNETCF auf dem Rechner installiert sein. Der Namensraum *OpenNETCF.Multimedia.Audio* des OpenNETCF muss importiert werden, um ein Recorder-Objekt erzeugen zu können. Zusätzlich muss der Namensraum *System.IO* importiert werden, um Zugriff auf den benötigten FileStream zu haben.
Zunächst müssen sowohl ein Recorder, als auch ein FileStream initialisiert werden. Um die Aufnahmen am Mikrofon zu starten, muss die Methode `recordFor()` mit dem FileStream und der Aufnahmelänge als Parameter im Recorder-Objekt aufgerufen werden.

15.3.11.2 Fazit

Der Einsatz des `OpenNETCF.Multimedia.Audio.Recorder` ist sehr einfach und daher für die Projektgruppe zu empfehlen.

15.3.11.3 Prototyp

```
1 using System.IO;
2 using OpenNETCF.Multimedia.Audio;
3
4 namespace OpenNetMikrofon
5 {
6    public class OpenNetMikrofonPrototype
7    {
8       public void record()
```

[12] http://www.opennetcf.org/

```
9       {
10          Recorder recorder = new Recorder();
11          FileStream fileStream = File.OpenWrite(@"\aufnahme.wav");
12          recorder.RecordFor(fileStream, 8);
13      }
14   }
15 }
```

Listing 15.9: Ansteuerung des Mikrofons via OpenNet Audio Recorder

15.3.12 Hardwarebuttons

Einige Hardwarebuttons des PDAs sollen zur Ansteuerung bestimmter Aktionen im Agentenspiel benutzt werden können. Eine direkte Ansteuerung dieser Buttons ist allerdings nicht möglich. Aus diesem Grund hat Microsoft die GamesAPI entwickelt. Diese hat die Hauptaufgabe, schnell Grafiken zu erzeugen. Darüber hinaus kann über die GamesAPI festgelegt werden, dass Eingaben über bestimmte Hardwarebuttons direkt an die Applikation und nicht, wie sonst üblich, an die Shell weitergeleitet werden.

15.3.12.1 Beschreibung des Tests

Die GamesAPI liegt für alle gängigen Prozessoren als DLL vor. Um diese im .NET Framework nutzen zu können, muss ein Wrapper geschrieben werden, der den Ablauf einzelner Funktionen ermöglicht. Zum Testen der GamesAPI wurde ein Beispiel[13] von Microsoft verwendet.

15.3.12.2 Fazit

Die Funktionen der GamesAPI können genutzt werden. Hier gibt es lediglich die Einschränkung, dass noch ein Wrapper geschrieben werden muss. Es existieren bereits Wrapper[14] im Internet, die allerdings kostenpflichtig sind.

15.3.12.3 Prototyp

```
1  using System;
2  using System.Runtime.InteropServices;
3
4  namespace GXGraphicsLibrary
5  {
6     /// <summary>
7     /// Full encapsulation of GAPI graphics functions and constants.
8     /// This entire class is only visible internally to the library.
9     /// </summary>
10    internal class GAPI
11    {
12       /// <summary>
13       /// Failure return value of GAPI functions.
14       /// </summary>
15       public const int GX_FAIL = 0;
16
```

[13] http://msdn.microsoft.com/smartclient/codesamples/default.aspx?pull=/library/en-us/dnnetcomp/html/wrapgapi1.asp
[14] http://www.gapidraw.com/

```
17          /// <summary>
18          /// Success return value of GAPI functions.
19          /// </summary>
20          public const int GX_SUCCESS = 1;
21
22          /// <summary>
23          /// Sets the display to fullscreen mode - the only mode
24          /// our library will support.  Used in GXOpenDisplay.
25          /// </summary>
26          public const uint GX_FULLSCREEN = 0x01;
27
28          /// <summary>
29          /// Specifies if the display is in landscape mode in GXDisplayProperties.ffFormat.
30          /// </summary>
31          public const uint kfLandscape = 0x8;
32
33          /// <summary>
34          /// Specifies if the display pixels are paletted in GXDisplayProperties.ffFormat.
35          /// </summary>
36          public const uint kfPalette = 0x10;
37
38          /// <summary>
39          /// Specifies if the display pixels are direct color in GXDisplayProperties.
                ffFormat.
40          /// </summary>
41          public const uint kfDirect = 0x20;
42
43          /// <summary>
44          /// Specifies if the display pixels are 15 bit 555 format in GXDisplayProperties.
                ffFormat.
45          /// </summary>
46          public const uint kfDirect5550 = 0x40;
47
48          /// <summary>
49          /// Specifies if the display pixels are 16 bit 565 format in GXDisplayProperties.
                ffFormat.
50          /// </summary>
51          public const uint kfDirect565 = 0x80;
52
53          /// <summary>
54          /// Specifies if the display pixels are 24 bit 888 format in GXDisplayProperties.
                ffFormat.
55          /// </summary>
56          public const uint kfDirect888 = 0x100;
57
58          /// <summary>
59          /// Specifies if the display pixels are 12 bit 444 format in GXDisplayProperties.
                ffFormat.
60          /// </summary>
61          public const uint kfDirect444 = 0x200;
62
63          /// <summary>
64          /// Specifies if the display pixels are inverted in GXDisplayProperties.ffFormat.
65          /// </summary>
66          public const uint kfDirectInverted = 0x400;
67
68          /// <summary>
69          /// Defines the display properties of the device in GXDisplayProperties.ffFormat.
```

```
70      /// </summary>
71      public class GXDisplayProperties
72      {
73          public uint cxWidth = 0;      // Screen width
74          public uint cyHeight = 0;     // Screen height
75          public int cbxPitch = 0;      // Byte offset to next pixel in X
76          public int cbyPitch = 0;      // Byte offset to next pixel in Y
77          public int cBPP = 0;       // Bits per pixel
78          public uint ffFormat = 0;     // Pixel format
79      }
80
81      /// <summary>
82      /// Starts a draw cycle to the display device. Called at the
83      /// start of each frame.
84      /// </summary>
85      /// <returns>A pointer to display memory pixel 0,0</returns>
86      [DllImport("gx.dll", EntryPoint="#1")]
87      extern public static IntPtr GxBeginDraw();
88
89      /// <summary>
90      /// End a draw cycle.  Call at the end of every draw frame.
91      /// </summary>
92      /// <returns>GX_FAIL or GX_SUCCESS</returns>
93      [DllImport("gx.dll", EntryPoint="#4")]
94      extern public static int GXEndDraw();
95
96      /// <summary>
97      /// Open the display for drawing.  Called only once.
98      /// </summary>
99      /// <param name="hWnd">Handle to the parent window</param>
100     /// <param name="dwFlags">Flags specifying display properties</param>
101     /// <returns>Non-zero if successful</returns>
102     [DllImport("gx.dll", EntryPoint="#8")]
103     extern public static int GXOpenDisplay(IntPtr hWnd, uint dwFlags);
104
105     /// <summary>
106     /// Close the display.  Called only once when application
107     /// has finished using GAPI.
108     /// </summary>
109     /// <returns>GX_FAIL or GX_SUCCESS</returns>
110     [DllImport("gx.dll", EntryPoint="#2")]
111     extern public static int GXCloseDisplay();
112
113     /// <summary>
114     /// Specifies if the display memory is a DRAM buffer.
115     /// </summary>
116     /// <returns>0 if false, non-zero otherwise</returns>
117     [DllImport("gx.dll", EntryPoint="#7")]
118     extern public static int GXIsDisplayDRAMBuffer();
119
120     /// <summary>
121     /// Resume drawing from a suspend.
122     /// </summary>
123     /// <returns>Always GX_SUCCESS</returns>
124     [DllImport("gx.dll", EntryPoint="#10")]
125     extern public static int GXResume();
126
127     /// <summary>
```

```
128         /// Set the graphics viewport.  This does not inherently clip.
129         /// </summary>
130         /// <param name="dwTop">Top pixel row</param>
131         /// <param name="dwHeight">Height of viewport</param>
132         /// <param name="dwReserved1">Unused</param>
133         /// <param name="dwReserved2">Unused</param>
134         /// <returns>GX_FAIL or GX_SUCCESS</returns>
135         [DllImport("gx.dll", EntryPoint="#11")]
136         extern public static int GXSetViewport(uint dwTop, uint dwHeight, uint dwReserved1
                , uint dwReserved2);
137
138         /// <summary>
139         /// Suspend drawing.  Used when application is minimized.
140         /// </summary>
141         /// <returns>Always GX_SUCCESS</returns>
142         [DllImport("gx.dll", EntryPoint="#12")]
143         extern public static int GXSuspend();
144
145         /// <summary>
146         /// Fills an instance of GXDisplayProperties.
147         /// </summary>
148         /// <param name="displayProps">Class instance of GXDisplayProperties</param>
149         [DllImport("gapinet.dll")]
150         extern public static void GXGetDisplayProperties(GXDisplayProperties displayProps);
151     }
152 }
```

Listing 15.10: GamesAPI

15.3.13 Touchscreen

Der Touchscreen dient zur haptischen Steuerung des Agentenspiels. Die Benutzung des Touchscreens ist problemlos möglich. Die Eingaben mit dem Zeigestift oder dem Finger auf dem Touchscreen werden wie Mauseingaben behandelt.

15.3.13.1 Beschreibung des Tests

Der Test erfolgt mittels einer über das vom MSDN Developer Center zur Verfügung gestellte Applikation [15]. Diese Applikation ermöglicht es einem Benutzer, auf dem PDA zu zeichnen. Dafür wurde ein spezielles Control-Widget realisiert, um die Zeichnung zu erkennen. Prototyp 15.11 enthält den Quellcode der Applikation.

15.3.13.2 Fazit

Der Touchscreen ist problemlos einsetzbar.

15.3.13.3 Prototyp

```
1  /*
2     SignatureControl class
3     --
4     Collects and displays a signature. The signature is made up of
5     a list of line segments. Each segment contains an array of points
6     (x and y coordinates).
7
8     Draws to a memory bitmap to prevent flickering.
9
10    Raises the SignatureUpdate event when a new segment is added to
11    the signature.
12 */
13
14 using System;
15 using System.Windows.Forms;
16 using System.Drawing;
17 using System.Collections;
18 using Common;
19
20 namespace PocketSignature
21 {
22    /// <summary>
23    /// Custom control that collects and displays a signature.
24    /// </summary>
25    public class SignatureControl : Control
26    {
27       // gdi objects
28       Bitmap _bmp;
29       Graphics _graphics;
30       Pen _pen = new Pen(Color.Black);
31
32       // list of line segments
33       ArrayList _lines = new ArrayList();
34
```

[15] http://msdn.microsoft.com/smartclient/codesamples/default.aspx?pull=/library/en-us/dnnetcomp/html/ppcsignatureapp.asp

```
35      // the current line segment
36      ArrayList _points = new ArrayList();
37      Point _lastPoint = new Point(0,0);
38
39      // if drawing signature or not
40      bool _collectPoints = false;
41
42      // notify parent that line segment was updated
43      public event EventHandler SignatureUpdate;
44
45
46      /// <summary>
47      /// List of signature line segments.
48      /// </summary>
49      public ArrayList Lines
50      {
51         get { return _lines; }
52      }
53
54      /// <summary>
55      /// Return the signature flattened to a stream of bytes.
56      /// </summary>
57      public byte[] SignatureBits
58      {
59         get { return SignatureData.GetBytes(this.Width, this.Height, _lines); }
60      }
61
62      public SignatureControl()
63      {
64      }
65
66      protected override void OnPaint(PaintEventArgs e)
67      {
68         // blit the memory bitmap to the screen
69         // we draw on the memory bitmap on mousemove so there
70         // is nothing else to draw at this time (the memory
71         // bitmap already contains all of the lines)
72         CreateGdiObjects();
73         e.Graphics.DrawImage(_bmp, 0, 0);
74      }
75
76      protected override void OnPaintBackground(PaintEventArgs e)
77      {
78         // don't pass to base since we paint everything, avoid flashing
79      }
80
81      protected override void OnMouseDown(MouseEventArgs e)
82      {
83         base.OnMouseDown(e);
84
85         // process if currently drawing signature
86         if (!_collectPoints)
87         {
88            // start collecting points
89            _collectPoints = true;
90
91            // use current mouse click as the first point
92            _lastPoint.X = e.X;
```

```
93              _lastPoint.Y = e.Y;
94
95              // this is the first point in the list
96              _points.Clear();
97              _points.Add(_lastPoint);
98          }
99      }
100
101     protected override void OnMouseUp(MouseEventArgs e)
102     {
103         base.OnMouseUp(e);
104
105         // process if drawing signature
106         if (_collectPoints)
107         {
108             // stop collecting points
109             _collectPoints = false;
110
111             // add current line to list of segments
112             Point[] points = new Point[_points.Count];
113             for (int i=0; i < _points.Count; i++)
114             {
115                 Point pt = (Point)_points[i];
116                 points[i].X = pt.X;
117                 points[i].Y = pt.Y;
118             }
119
120             _lines.Add(points);
121
122             // start over with a new line
123             _points.Clear();
124
125             // notify container a new segment was added
126             RaiseSignatureUpdateEvent();
127         }
128     }
129
130     protected override void OnMouseMove(MouseEventArgs e)
131     {
132         base.OnMouseMove(e);
133
134         // process if drawing signature
135         if (_collectPoints)
136         {
137             // add point to current line segment
138             _points.Add(new Point(e.X, e.Y));
139
140             // draw the new segment on the memory bitmap
141             _graphics.DrawLine(_pen, _lastPoint.X, _lastPoint.Y, e.X, e.Y);
142
143             // update the current position
144             _lastPoint.X = e.X;
145             _lastPoint.Y = e.Y;
146
147             // display the updated bitmap
148             Invalidate();
149         }
150     }
```

```
151
152         /// <summary>
153         /// Clear the signature.
154         /// </summary>
155         public void Clear()
156         {
157             _lines.Clear();
158             InitMemoryBitmap();
159             Invalidate();
160         }
161
162         /// <summary>
163         /// Create any GDI objects required to draw signature.
164         /// </summary>
165         private void CreateGdiObjects()
166         {
167             // only create if don't have one or the size changed
168             if (_bmp == null || _bmp.Width != this.Width || _bmp.Height != this.Height)
169             {
170                 // memory bitmap to draw on
171                 InitMemoryBitmap();
172             }
173         }
174
175         /// <summary>
176         /// Create a memory bitmap that is used to draw the signature.
177         /// </summary>
178         private void InitMemoryBitmap()
179         {
180             // load the background image
181             _bmp = Global.LoadImage("sign␣here.png");
182
183             // get graphics object now to make drawing during mousemove faster
184             _graphics = Graphics.FromImage(_bmp);
185         }
186
187         /// <summary>
188         /// Notify container that a line segment has been added.
189         /// </summary>
190         private void RaiseSignatureUpdateEvent()
191         {
192             if (this.SignatureUpdate != null)
193                 SignatureUpdate(this, EventArgs.Empty);
194         }
195     }
196 }
```

Listing 15.11: Ansteuerung des Touchscreens

15.3.14 Vibration

Bei dieser Machbarkeitsstudie soll geprüft werden, inwieweit sich der Vibrationsalarm eines Pocket PCs aus dem .NET Compact Framework ansteuern lässt. Der Vibrationsalarm soll zur Umsetzung der multimodalen Präsentation von Informationen eingesetzt werden.

15.3.14.1 Beschreibung des Tests

AYGShell API Extensions Die Ansteuerung der Funktionalität „Vibration“ bis Windows CE 4.2 ist über die AYGShell API Extensions[16] möglich. Seit Windows CE 4.2 sind die OEMs dafür verantwortlich, dass die API zur Verfügung gestellt wird.

Virtuelle LED Der Vibrationsalarm kann über den gleichen Mechanismus bedient werden, mit dem auch die LEDs im Gehäuse des Pocket PCs angesteuert werden können. Er stellt somit eine virtuelle LED dar. Im Namensraum `OpenNETCF.Notification` des OpenNet Compact Frameworks[17], können diese LEDs mittels der Klasse `LED` angesteuert werden. Prototyp 15.12 zeigt, wie der Vibrationsalarm mit Hilfe dieser Klasse ein- und ausgeschaltet werden kann.

15.3.14.2 Fazit

Die Ansteuerung des Vibrationsalarms ist möglich, man kann dabei aber nur auf die Dauer der Vibration Einfluss nehmen. Frequenz und Stärke der Vibration können nicht geändert werden.

15.3.14.3 Prototyp

```
1  // Dieser Prototyp zeigt, wie der Vibrationsalarm des Pocket PC iPAQ 5450 aus .NET CF
2  // Programmen angesteuert werden kann.
3
4  using OpenNETCF.Notification;
5
6  namespace Vibration
7  {
8     public class Vibration
9     {
10       // Auf dem iPAQ 5450 steuert man den Vibrationsalarm um die virtuelle LED Nr. 5
11       private const int VIRTUAL_VIBRATION_LED_NR = 5;
12
13       // Mit dieser Klasse öknnen die LEDs des iPAQs angesteuert werden.
14       // Auch steuerbar ist z.B. die Bluetooth-Status LED (blaue LED).
15       private static Led led = new Led();
16
17       // schaltet die virtuelle LED - und damit den Vibrationsalarm - ein
18       public static void Start()
19       {
20          Vibration.led.SetLedStatus(VIRTUAL_VIBRATION_LED_NR, Led.LedState.On);
21       }
22
23       // schaltet die virtuelle LED - und damit den Vibrationsalarm - aus
24       public static void Stop()
25       {
26          Vibration.led.SetLedStatus(VIRTUAL_VIBRATION_LED_NR, Led.LedState.Off);
27       }
28    }
29 }
```

Listing 15.12: Ansteuern des Vibrationsalarms

[16] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wceui40/html/_ceconAYGShellAPIExtensions.asp
[17] www.opennetcf.org

15.3.15 RFID

Um Informationen an beliebigen Orten hinterlegen zu können, wurde überlegt, RFID Tags als ein Art digitale *PostIts* einzusetzen. Durch die folgende Machbarkeitsstudie soll untersucht werden, wie die Daten, die über einen RFID-Reader von einem RFID-Tag empfangen werden, innerhalb des PDA verarbeitet werden können. Insbesondere soll untersucht werden, auf welche Art und Weise, eine Applikation auf diese Daten zugreifen kann.

15.3.15.1 Beschreibung des Tests

Um die Informationen der RFID-Tags auslesen zu können, müssen die vom OFFIS zur Verfügung gestellten PDAs (HP iPAQ 5450) mit RFID-Readern ausgestattet werden. Hierzu gibt es zwei Möglichkeiten. Entweder man verwendet einen RFID-Reader in Form einer SD-Karte, die in den entsprechenden Slot des PDAs gesteckt wird oder man verwendet einen RFID-Reader in Form einer CF-Karte. Hierzu muss der PDA zunächst mit einem Jacket ausgestattet werden, welches einen entsprechenden Slot zur Verfügung stellt (vgl. Kapitel 12.4.1 auf Seite 214).
Innerhalb dieser Machbarkeitsstudie soll getestet werden, wie die Ansteuerung der RFID-Reader aus dem .NET Framework heraus realisiert werden kann. Wichtig ist u.a., eventuell benötigte Zusatzpakete zu identifizieren.
Eine Recherche im Internet nach einer Möglichkeit, die RFID-Reader direkt aus dem .NET-Framework heraus ansteuern zu können, brachte keine Ergebnisse. Von einigen Firmen, wie z.B. *AppForge*[18], werden Anwendungen angeboten, mit deren Hilfe die Ansteuerung der RFID-Reader programmiert werden kann. *AppForge* bietet hierzu die Anwendung *Crossfire* an. Diese kann in Microsoft .NET integriert werden und unterstützt dann durch spezielle Module und Bibliotheken den Zugriff auf die Daten, die von den RFID-Tags gelesen werden. *Crossfire* ist eine kostenpflichtige Anwendung, kann aber als 30 Tage Testversion bezogen werden.

15.3.15.2 Fazit

Ein Zugriff auf die Daten, die vom RFID Reader gelesen werden, ist derzeit für die Projektgruppe nicht möglich. Des weiteren kann die Übertragung von Daten zwischen RFID-Tag und RFID-Reader des PDAs derzeit nicht getestet werden, da die zur Verfügung stehenden mobilen Geräte inkompatibel sind. Dies liegt daran, dass die RFID-Tags und die RFID-Reader auf unterschiedlichen Frequenzen arbeiten.

[18] http://www.appforge.com/

15.4 Modultests

Modultests, auch bekannt als *Unit*-Tests oder Komponententests, dienen der Validierung der Korrektheit von Modulen einer Software. Unter einem Softwaremodul versteht man hierbei z.B. eine Methode oder eine einzelne Klasse. Bei der Softwareentwicklung spielen Modultests eine tragende Rolle. Die Durchführung von Modultests stellt sicher, dass alle Module einer Software einen bestimmten Fehlerfreiheitsgrad aufweisen. Um dies zu gewährleisten, ist es allerdings notwendig, nach jeder Änderung eines Moduls, alle Testfälle erneut zu durchlaufen, um die Fehlerfreiheit zu überprüfen.
Modultests können entweder nach Fertigstellung des zu testenden Moduls oder parallel während der Implementierung durchgeführt werden. Letzterer Ansatz wird beim testgetriebenen Programmieren (*Test-First*-Programmieren) angewandt. Die Modultests werden hierbei parallel zum eigentlichen Quellcode erstellt und gepflegt. Dies ermöglicht es dem Programmierer, ungewollte Nebeneffekte oder Fehler, die durch seine Änderungen verursacht wurden, sofort nachvollziehen zu können. Für die parallele Durchführung von Implementierung und Modultests stehen je nach verwendeter Programmiersprache eine Reihe von Testing-Frameworks zur Verfügung. Das von der Projektgruppe für Modultests in C# verwendete NUnit-Framework wird im Abschnitt 15.4.1 näher vorgestellt.
Für das Auffinden von Fehlern ist es darüber hinaus oftmals hilfreich, an bestimmten Stellen des Quellcodes Testausgaben zu erzwingen. Wie das zu diesem Zweck eingesetzte Logging in C# funktioniert, wird in Abschnitt 15.4.2 erläutert.

15.4.1 NUnit

Dieser Abschnitt beschreibt das Testing-Framework NUnit[19] und seinen Einsatz während der Implementierungsphase. Zunächst wird die Motivation für den Einsatz von Modultests erläutert. Danach folgen Installationshinweise und schliesslich eine kurze Einführung in die wichtigsten Elemente von NUnit.

15.4.1.1 Motivation

Die so genannten Modultests werden eingesetzt, um die korrekte Arbeitsweise einzelner Methoden zu validieren. Die Tests laufen automatisch ab, sparen dem Programmierer also eine Menge Zeit. Modultests sind besonders nützlich, wenn größere Änderungen am Code vorgenommen werden. Anhand der Tests kann man dann feststellen, ob die Änderungen durch unbeachtete Seiteneffekte zu Fehlern in der Software geführt haben. Damit eine spürbare Absicherung durch die Tests entsteht, müssen jedoch sehr konsequent alle Methoden und Funktionalitäten getestet werden.
Man unterscheidet bei Tests zwischen lokalen Modultests, die eine einzelne Klasse oder Methode kontrollieren, und globale Akzeptanz-Tests, mit denen ein von außen sichtbares Verhalten überprüft wird. Letztere modellieren die Anforderungen an bestimmte Module oder das ganze System, so dass der Programmierer validieren kann, ob eine Anforderung umgesetzt wurde, oder nicht.
Beim so genannten *testgetriebenen Programmieren*, welches in der Projektgruppe eingesetzt werden soll, wird zunächst überlegt, welche Funktionalität implementiert werden soll. Danach werden Testfälle geschrieben, welche die Anforderungen validieren. Da die Funktionalität an sich noch nicht implementiert wurde, schlagen die Tests zunächst fehl. Erst dann beginnen die Programmierer die Funktionalität zu implementieren. Anhand der Testfälle können sie feststellen, wann sie alle Anforderungen erfüllt haben.

[19] http://sourceforge.net/projects/nunit

15.4.1.2 Installation

Unter `http://sourceforge.net/projects/nunit` kann der *Installer* für Windows heruntergeladen werden. Bei der Installation muss lediglich der Installationspfad angegeben werden. Danach ist NUnit als *Assembly*-Verweis in Visual Studio Projekten verfügbar.

15.4.1.3 Primärer Einsatz von NUnit

Listing 15.13 zeigt eine Beispielklasse, anhand derer ein Test mit NUnit demonstiert werden soll. Es handelt sich dabei um eine simple Modellierung eines Bankkontos. Es bietet die Funktionalitäten, Geld einzuzahlen, Geld abzuheben und Geld auf ein anderes Konto zu überweisen.

```
1  namespace bank
2  {
3     public class Account
4     {
5        private float balance;
6
7        public void Deposit(float amount)
8        {
9           balance+=amount;
10       }
11
12       public void Withdraw(float amount)
13       {
14          balance-=amount;
15       }
16
17       public void TransferFunds(Account destination, float amount)
18       {
19       }
20
21       public float Balance
22       {
23          get{ return balance; }
24       }
25    }
26 }
```

Listing 15.13: NUnit: zu testende Beispielklasse

Will man die Klasse nun mit NUnit testen, legt man dazu eine Testklasse an. Es wird empfohlen, für die Dauer des Projekts, eine gleichnamige Klasse, um die Endung `Test` erweitert, anzulegen. Listing 15.14 zeigt die erste Version der Testklasse. Die Klasse muss mit dem Attribut `[TestFixture]` gekennzeichnet sein, damit NUnit sie als Testklasse erkennen kann. Der Test wird in einer oder mehreren beliebige benannten Methoden angelegt, die mit dem Attribut `[Test]` markiert werden müssen.
In der `[Test]` Methode von Listing 15.14 werden zwei `Accounts` angelegt, auf die einmal 150 und einmal 200 Werteinheiten eingezahlt werden. Dann werden von dem Konto mit 200 Einheiten, 100 Einheiten auf das andere Konto überwiesen. Über die NUnit-Methode `Assert.AreEqual` wird nun getestet, ob die Konten nun die erwarteten Bestände von 250 und 100 Werteinheiten aufweisen. Wenn bei dem Testlauf die beiden Parameter der Methode `Assert.AreEqual` ungleich sind, gilt der NUnit-Test als nicht bestanden.

```
1  using NUnit.Framework;
2
3  namespace bank
4  {
5     [TestFixture]
6     public class AccountTest
7     {
8        [Test]
9        public void TransferFunds()
10       {
11          Account source = new Account();
12          source.Deposit(200.00F);
13          Account destination = new Account();
14          destination.Deposit(150.00F);
15
16          source.TransferFunds(destination, 100.00F);
17          Assert.AreEqual(250.00F, destination.Balance);
18          Assert.AreEqual(100.00F, source.Balance);
19       }
20    }
21 }
```

Listing 15.14: NUnit: Beispiel-Testfall

Als nächstes soll beschrieben werden, wie man einen Test in ein Visual Studio Projekt integriert. Zunächst legt man dazu ein Projekt für eine Konsolenanwendung in C# an und fügt die Klassen `Account` (Listing 15.13) und `AccountTest` (Listing 15.14) hinzu. Damit das Projekt kompiliert werden kann, muss der Verweis `nunit.framework` dem Projekt hinzugefügt werden. Jetzt kann die *.exe*-Datei zu dem Projekt erzeugt werden.
Um den Test zu durchlaufen, startet man nun über das Startmenü das Programm *NUnit-Gui*. Über File → Open öffnen man den Datei-Browser und wählt die erstellte *.exe*-Datei aus. NUnit-Gui zeigt nun in einem Baum den Namensraum `bank`, darunter die Klasse `AccountTest`, und darin die Testmethode `TransferFunds` an. Über den Button „run“ startet man nun das automatische Abarbeiten aller vorhandenen Testfälle - in diesem Fall des Testfalls `TransferFunds`. Der Test schlägt fehl, da die Methode `Account.TransferFunds(Account, float)` leer ist, der Transfer also nicht stattfindet.
Nun müssten sich die Programmierer daran setzten, die Methode so zu implementieren, dass die Tests nicht mehr fehlschlagen. Erst wenn alle Tests laufen, wird der Code ins CVS eingecheckt. So wird sichergestellt, dass alle sich im CVS befindlichen Programme stets einen bestimmten Fehlerfreiheitsgrad aufweisen.

15.4.1.4 Validierung des Exceptionhandlings mit NUnit

Dieser Abschnitt beschreibt, wie man überprüfen kann, ob Methoden die erwarteten Exceptions zu bestimmten Eingaben werfen. Wir verwenden dazu wieder das Bankkonto aus dem vorigen Abschnitt. Listing 15.15 zeigt die nach dem NUnit Test korrigierte `Account`-Klasse. Die Methode zum Tätigen von Überweisungen wurde korrekt implementiert, so dass der Test positiv ausfällt. Zusätzlich wurde der Klasse das Attribut `minimumBalance` eingefügt, das die minimale Anzahl an Einheiten pro Transfer angibt. Werden bei einem Transfer weniger Einheiten überwiesen, soll eine `InsufficientFundsException`, die als neue Klasse in der selben Datei definiert wird, geworfen werden.

```
1  namespace bank
2  {
3     public class Account
4     {
5        private float balance;
6
7        private float minimumBalance = 10.00F;
8
9        public void Deposit(float amount)
10       {
11          balance+=amount;
12       }
13
14       public void Withdraw(float amount)
15       {
16          balance-=amount;
17       }
18
19       public void TransferFunds(Account destination, float amount)
20       {
21          destination.Deposit(amount);
22          Withdraw(amount);
23       }
24
25
26       public float Balance
27       {
28          get{ return balance; }
29       }
30
31       public float MinimumBalance
32       {
33          get{ return minimumBalance;}
34       }
35    }
36
37    public class InsufficientFundsException : ApplicationException
38    {
39    }
40
41 }
```

Listing 15.15: NUnit: erweiterte Beispielklasse

Um das Werfen bestimmter Exceptions zu validieren, bietet NUnit ein spezielles Attribut an. Listing 15.16 zeigt die um einen Testfall erweiterte Testklasse. Dort wird eine Überweisung von zu wenig Einheiten simuliert. Über das zusätzliche Attribut `[ExpectedException(typeof(InsufficientFundsException))]` zeigt der Testfall an, dass eine `InsufficientFundsException` erwartet wird. Tritt diese Exception nicht auf, gilt der Test als nicht bestanden. Führt man NUnit mit dem zweiten Testfall aus, schlägt der Test erwartungsgemäß fehl, weil die Klasse `Account` bei einer Überweisung nicht überprüft, ob der Auftrag genügend Werteinheiten umfasst. Die Programmierer müssten nun dafür sorgen, dass die Methode `Account.TransferFunds(Account, float)` bei einem Überweisungswert von weniger als 10 Einheiten eine `InsufficientFundsException` wirft.

```
1  using NUnit.Framework;
2
3  namespace bank
4  {
5     [TestFixture]
6     public class AccountTest
7     {
8        [Test]
9        public void TransferFunds()
10       {
11          Account source = new Account();
12          source.Deposit(200.00F);
13          Account destination = new Account();
14          destination.Deposit(150.00F);
15
16          source.TransferFunds(destination, 100.00F);
17          Assert.AreEqual(250.00F, destination.Balance);
18          Assert.AreEqual(100.00F, source.Balance);
19       }
20
21       [Test]
22       [ExpectedException(typeof(InsufficientFundsException))]
23       public void TransferWithInsufficientFunds()
24       {
25          Account source = new Account();
26          source.Deposit(200.00F);
27          Account destination = new Account();
28          destination.Deposit(150.00F);
29          source.TransferFunds(destination, 300.00F);
30       }
31
32    }
33 }
```

Listing 15.16: NUnit: erweiterte Testfallklasse

15.4.1.5 Temporäres Ignorieren von Testfällen

Werden für einen Testfall die Anforderungen in Frage gestellt, aber nicht verbindlich geklärt, sollte der Testfall beim automatischen Test ignoriert werden. Anstatt den Testfall zu löschen oder auszukommentieren, bietet NUnit die Möglichkeit, Testfälle mit dem Attribut `[Test, Ignore('Erklärung für das Ignorieren')]]` zu versehen. Diese Tests werden von NUnit beim Durchlaufen der automatischen Tests als „nicht getestet“ gekennzeichnet.

15.4.2 Logging mit C#

Zur Ablaufverfolgung eines Programms bietet sich der in Visual Studio integrierte Debugger an. Möchte man allerdings das laufende Programm nicht permanent überwachen, so sind Logdateien das Mittel der Wahl. Der Logging-Mechanismus in C# ist relativ einfach gehalten. Es wird unterschieden zwischen Logs eines Debug-Programms und eines Release-Programms. Die Mechanismen sind sich sehr ähnlich, nur die Klassen der Logging-Methoden unterscheiden sich. Für Logging im Debug-Programm lautete die anzusprechende Klasse `System.Diagnostics.Debug` und für Release-Programme `System.Diagnostics.Trace`. Feingranulare Einstellungen, bspw. ein Log-Level, sind in C# nicht vorgesehen. Es können aber System-Events geloggt werden, bspw. der Staus einer Internetverbindung.

```
1  using System;
2  using System.IO;
3  using System.Diagnostics;
4
5  namespace LoggingTest
6  {
7     class DebugTester
8     {
9        static void Main(string[] args)
10       {
11          //file io.
12          Stream debugFile = File.Create("TestDebug.txt");
13          TextWriterTraceListener fileWriter
14             = new TextWriterTraceListener(debugFile);
15
16          //console io
17          TextWriterTraceListener consoleWriter
18             = new TextWriterTraceListener(System.Console.Out);
19
20          //ausgabe registrieren
21          Debug.Listeners.Add(fileWriter);
22          Debug.Listeners.Add(consoleWriter);
23
24          //test
25          Debug.Write("Debugtest");
26
27          //ausschreiben
28          Debug.Flush();
29
30          //ausschreiben und ßschlieen
31          fileWriter.Flush();
32          fileWriter.Close();
33
34          consoleWriter.Flush();
35          consoleWriter.Close();
36
37          //auf tastendruck warten
38          System.Console.Read();
39       }
40    }
41 }
```

Listing 15.17: C#-Logging

Listing 15.17 zeigt einfaches loggen über die Klasse `System.Diagnostics.Debug`. Die Ausgaben erfolgen in eine Datei und auf die Konsole. Er werden von `System.Diagnostics.TraceListener` abgeleitete Typen registriert. Über diese können Ausgaben direkt erfolgen. Empfehlenswert ist es aber, damit auch alle Listener die gleichen Ausgaben machen, die Ausgaben direkt über die `System.Diagnostics.Debug`-Klasse laufen zu lassen. Zur Debug-Ausgabe in eine `System.Windows.Forms.TextBox` muss ein eigener Listener, abgeleitet von `System.Diagnostics.TraceListener`, geschrieben werden.

Für die Implementierung des Agentenspiels lässt der C#-Log-Mechanismus einige Eigenschaften vermissen. Beispielsweise kann kein so genanntes Log-Level festgelegt werden, mit dem die Log-Information feingranularer gefiltert werden können. Allgemeine Filter nach bestimmten Ausdrücken oder ähnlichem sucht man ebenfalls vergebens. Aus diesem Grund wird ein eigener Log-Mechanismus entwickelt, der dem C#-Log-Mechanismus ähnelt, aber reicher an Filtermöglichkeiten der Log-Informationen ist. Der für das Agentenspiel verwendete Log-Mechanismus wird in Abschnitt 14.4.1 auf Seite 318 näher beschrieben.

15.5 Integrationstests

In diesem Abschnitt wird die Durchführung der Integrationstests beschrieben. Ein Integrationstest dient zum Testen des Zusammenspiels der einzelnen Module eines Softwaresystems. Voraussetzung für die Durchführung eines solchen Tests ist, dass alle Module bereits erfolgreich einen Modultest durchlaufen haben. Als Orientierungshilfe für die Auswahl der durchzuführenden Integrationstests bieten die im Implementierungsplan auf Seite 160 dargestellten Anwendungsfälle an. Die Reihenfolge, in der die einzelnen Integrationstests durchzuführen sind, wird im so genannten Testplan festgehalten. Als Vorlage für den Aufbau diesen Testplans und für den Aufbau der Testprotokolle der einzelnen Integrationstests dient der IEEE Standard 829-1998 (*IEEE Standard for Software Test Documentation*).

15.5.1 Testplan

Innerhalb dieses Abschnitts wird der Testplan für die Durchführung der Integrationstests beschrieben. Der Testplan beinhaltet dabei Angaben über den Zeitraum der Testphase, eine Auflistung testrelevanter Dokumente und eine Benennung von Kriterien, die einen Fehlschlag dieser Testphase kennzeichnen. Darüber hinaus enthält der Testplan eine Beschreibung des generelle Ablaufs der Testphase. Hierzu werden alle durchgeführten Tests in chronologischer Reihenfolge aufgelistet.

Zeitraum:

04.07.-21.09.05

Referenzen zu anderen Dokumenten:

- IEEE Standard 829-1998 (*IEEE Standard for Software Test Documentation*)
- Implementierungsplan (siehe Abschnitt 10.1 auf Seite 160)

Pass/Fail-Kriterien

Die Testphase gilt als bestanden, wenn sämtliche Integrationstests mindestens einmal erfolgreich durchlaufen wurden.

Zeitplan/Ablaufplan

- Verwaltung der Clients
  - Anmelden von Clients (siehe 15.5.2.1 auf Seite 374)
  - Abmelden von Clients (siehe 15.5.2.2 auf Seite 375)
- Positionsbestimmung der Clients
  - Senden der Position (siehe 15.5.3.1 auf Seite 379)
  - Startpunkt (siehe 15.5.3.2 auf Seite 381)
  - Client aktivieren (siehe 15.5.3.3 auf Seite 382)
  - Sehen von Goodies (siehe 15.5.3.4 auf Seite 384)

  - Sehen von Startpunkten (siehe 15.5.3.5 auf Seite 386)
  - Sehen von Missionszielen (siehe 15.5.3.6 auf Seite 387)
  - Sehen von Mitspielern (siehe 15.5.3.7 auf Seite 388)
  - Sehen von Gegnern (siehe 15.5.3.8 auf Seite 389)
  - Sehen von Kameras (siehe 15.5.3.9 auf Seite 390)
- Kartendarstellung
  - Karte zoomen (siehe 15.5.4.1 auf Seite 392)
  - Darstellung von POIs und eigener Position (siehe 15.5.4.2 auf Seite 393)
- Kontext
  - Qualifizierte Positionsdaten (siehe 15.5.5.1 auf Seite 395)
  - Ausgabe des Kontexts (siehe 15.5.5.2 auf Seite 396)
- Multimodale Ausgabe
  - Akustische Ausgabe (siehe 15.5.6.1 auf Seite 398)
  - Vibrationsausgabe (siehe 15.5.6.2 auf Seite 399)
  - Berechnung der Ausgabemedien (siehe 15.5.6.3 auf Seite 399)
- Funktionalitäten
  - Tarnen (siehe 15.5.7.1 auf Seite 402)
  - Neutralisieren (siehe 15.5.7.2 auf Seite 405)
  - Aufklären (siehe 15.5.7.3 auf Seite 408)
  - Abhören (siehe 15.5.7.4 auf Seite 412)
- Nachrichten
  - Systemnachrichten an Clients senden (siehe 15.5.8.1 auf Seite 415)
  - Nachrichten verfassen/versenden (siehe 15.5.8.2 auf Seite 416)
  - Nachrichten empfangen/anzeigen (siehe 15.5.8.3 auf Seite 419)
  - Nachrichten löschen (siehe 15.5.8.4 auf Seite 423)
  - Nachrichten beantworten (siehe 15.5.8.5 auf Seite 425)
  - Nachrichten weiterleiten (siehe 15.5.8.6 auf Seite 428)
- Ziel des Spiels
  - Missionsziele anzeigen/lösen (siehe 15.5.9.1 auf Seite 431)
  - Spielende (siehe 15.5.9.2 auf Seite 432)
- weitere Testfälle
  - Konfigurationsdatei (siehe 15.5.10.1 auf Seite 434)

15.5.2 Testfälle: Verwaltung der Clients

15.5.2.1 Anmelden von Clients

Ein Client soll eine Verbindung zum Server herstellen. Der Server soll die eingehende Anfrage bearbeiten und den Client als Spieler registrieren. Während der Anmeldung werden Name und Fähigkeitslevel vom Client zum Server übertragen. Der Server gibt dem Spieler eine eindeutige ID, ordnet ihn einem Team zu und sendet diese Daten an den Client zurück. Dann erzeugt der Server einen Startpunkt für den Spieler und sendet diesen als Point-of-Interest an den Client. Hat der Client die Anmeldung vollendet, wechselt er zur Kartenansicht. In der GUI des Servers werden der Spieler und sein Startpunkt angezeigt.

Verantwortliche(r): MP

Pass-Kriterien:

- die Internetverbindung zwischen Server und Client wird hergestellt
- auf Serverseite ist der Client als neuer Spieler registriert
- der Client hat eine eindeutige ID vom Server zugewiesen bekommen
- die Hauptsansicht des Clients zeigt die Kartenansicht
- die bereits angemeldeten Clients erhalten das Spielerprofil des neuen Clients zugeschickt
- der sich anmeldende Client erhält die Spielerprofile der bereits registrierten Spieler zugeschickt

Fail-Kriterien:

- Client oder Server stürzen während der Anmeldung ab
- Internetverbindung zwischen Client und Server kann nicht hergestellt werden
- Anmeldedaten werden nicht zum Server übertragen
- Server registriert keinen Spieler oder erzeugt keinen Startpunkt
- die Aktualisierung der ID des Spielers oder seiner Teamzugehörigkeit wird nicht vorgenommen, bzw. wird nicht zurück an den Client gesendet und dort verarbeitet
- es wird kein Startpunkt für den Spieler angelegt und an den Client gesendet

---

1. Testlauf

Datum: 01.08.2005

Stand der Software: Sehr früher Stand der Software; Client und Server können fehlerfrei gestartet werden, es wurden aber noch keine anderen Testfälle positiv durchlaufen.

Tester: SK, MP

Setup:

- Windows PC mit Windows XP Betriebssystem als Server
- iPAQ 5450, WLAN
- GPS-Maus

Voraussetzungen: Der Server ist hochgefahren und ein Spiel zu dem Szenario aus der Datei `offis.xml` gestartet.

Testablauf:

1. Verbindungen des Clients zu WLAN Access Point und GPS Maus herstellen
2. Starten des Clients und Auswahl der Szenariodatei `offis.xml` im Setup-Dialog
3. Bestätigen des Setups durch klicken auf OK und beobachten der Reaktion

Auswertung: Der Client meldete sich beim Server an. In der GUI des Servers wurde das Anlegen eines Spieler-Objekts für den Client bestätigt. Der Client wechselte wie erwartet in die Kartenansicht. Der Testlauf wurde bestanden.

---

2. Testlauf

Datum: 02.09.2005

Stand der Software: Die Implementierung der Features ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup: *siehe 1. Testlauf, nur in anderem WLAN Netzwerk*

Voraussetzungen: *siehe 1. Testlauf*

Testablauf: *siehe 1. Testlauf*

Auswertung: Wie beim ersten Testlauf bestand der Client den Testfall.

15.5.2.2 Abmelden von Clients

Ein angemeldeter Client soll das Spiel verlassen. Dabei soll zum Einen der Client korrekt heruntergefahren werden, so dass alle belegten Ressourcen freigegeben werden. Zum Anderen muss der Server alle für den Spieler angelegten Objekte löschen.

Verantwortliche(r): MP

Pass-Kriterien:

- die mit dem Client assoziierten Spielobjekte werden aus den Spielobjekten des Servers entfernt
- alle Threads des Clients beenden sich, alle Ressourcen (z.B. Zugriff auf das Dateisystem) werden freigegeben.
- alle mit dem Client assoziierten Daten, wie das Spielerprofil, sein Point-of-Interest, usw. werden von den anderen Clients entfernt

Fail-Kriterien:

- Server stürzt ab
- Server löscht die für den Spieler angelegten Objekte nicht
- der Client hat nicht alle Ressourcen freigegeben, so dass er nicht ein zweites Mal gestartet werden kann

---

1. Testlauf

Datum: 01.08.2005

Stand der Software: Sehr früher Stand der Software; Client und Server können fehlerfrei gestartet werden, der Client kann sich beim Server anmelden.

Tester: SK, MP

Setup:

- Windows PC mit Windows XP Betriebssystem als Server
- iPAQ 5450, WLAN
- GPS Maus

Voraussetzungen: Der Client hat sich erfolgreich beim Server angemeldet.

Testablauf:

1. Beenden des Client durch drücken auf das X in einer der Hauptansichten.

Auswertung: Der Client meldete sich beim Server ab. In der GUI des Servers wurde der abgemeldete Client nicht mehr angezeigt. Der Testlauf wurde bestanden.

---

2. Testlauf

Datum: 02.09.2005

Stand der Software: Die Implementierung der Features ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup: *siehe 1. Testlauf, nur in anderem WLAN Netzwerk*

Voraussetzungen: *siehe 1. Testlauf*

Testablauf: *siehe 1. Testlauf*

Auswertung: Wie beim ersten Testlauf bestand der Client den Testfall.

---

3. Testlauf In diesem Testlauf soll überprüft werden, ob ein Spieler alle aufgenommenen Goodies ablegt, wenn er sich abmeldet. Darüber hinaus soll getestet werden, ob die abgelegten Goodies von einem anderen Spieler wieder aufgenommen werden können.

Datum: 07.09.2005

Stand der Software: Die Implementierung der Features ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: JP

Setup: Für den Test werden ein Server und zwei Clients benötigt. In diesem Testlauf werden alle Clients über den Pocket PC 2003 Emulator von Microsoft gestartet. Um dies zu ermöglichen, werden zwei PCs benötigt.

Konfiguration PC 1 (Server und Client 1)

- Intel Pentium M (1,6 GHz), 1 GB RAM
- Microsoft Windows XP Home Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PC 2 (Client 2)

- AMD Athlon XP 1800+ (1,53 GHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Der Server ist gestartet und das Szenario *offis.xml* wurde geladen. Beide Clients sind auf dem Server angemeldet (unter den Namen „Client 1“ und „Client 2“) und sind verschiedenen Teams zugeordnet.

Testablauf:

1. Client 1: Startpunkt aufsuchen, um aktiviert zu werden
2. Client 1: Goodie oben links auf dem Spielfeld suchen und aufnehmen
3. Client 1: Goodie unten rechts auf dem Spielfeld suchen und aufnehmen
4. Client 1: im MenuPanel überprüfen, ob beide Goodies angezeigt werden
5. Client 2: Startpunkt aufsuchen, um aktiviert zu werden
6. Client 2: Client 1 in der unteren rechten Ecke suchen
7. Client 1: abmelden / Spiel beenden
8. Client 2: Goodies in der rechten unteren Ecke finden
9. Client 2: erstes Goodie aufnehmen
10. Client 2: zweites Goodie aufnehmen
11. Client 2: im MenuPanel überprüfen, ob beide Goodies angezeigt werden

Auswertung: Der Test wurde erfolgreich durchgeführt. Client 1 legte die aufgenommenen Goodies an der Stelle ab, an der er sich beim Verlassen des Spiels befand. Die Goodies konnten dann von Client 2 gefunden und aufgenommen werden.

15.5.3 Testfälle: Positionsbestimmung der Clients

15.5.3.1 Senden/Anzeigen der Position

Die geographische Position eines Spielers, sowie die im Rahmen des Agentenspiels für diesen Spieler „sichtbaren“ Mitspieler, sollen auf der Übersichtskarte während des Spielverlaufs angezeigt werden. Die Informationen für die Bestimmung dieser Positionen werden von dem GPS-Navigationssystem empfangen. Für die Darstellung der Positionen auf der Karte werden die GPS-Koordinaten der Spieler auf die Karte abgebildet.

Verantwortliche(r): FJ

Pass-Kriterien:

- Sehen der eigenen Position an der Position auf der Übersichtskarte (in Form eines Punktes oder Icons), die der wirklichen Position des Spielers entspricht
- Sehen der Positionen von Mitspielern an den Positionen auf der Übersichtskarte (in Form eines Punktes oder Icons), die den wirklichen Positionen der Mitspieler entspricht
- der Server empfängt ein `PositionUpdatedEvent`

Fail-Kriterien:

- die eigene Position wird nicht auf der Karte angezeigt
- die Positionen der Mitspieler werden nicht auf der Karte angezeigt
- ein `PositionUpdateEvent` wird nicht vom Server empfangen

---

1. Testlauf

Datum: 16.08.2005

Stand der Software: Die Software befindet sich auf einem fortgeschrittenen Entwicklungsstand. Das Darstellen der eigenen Position auf der Karte ist implementiert, die Funktionen zum Darstellen der Positionen der Mitspieler fehlen.

Tester: FJ

Setup: Aufgrund der fehlenden Implementierung des Sehens von Mitspielern wird für diesen Testlauf lediglich ein PC für die Ausführung des Servers sowie ein Pocket PC für das Ausführen des Clients benötigt.

- Pocket PC (HP iPaq 5450), für den eine Bluetooth-Verbindung zum GPS-Empfänger (Falcom NAVI-1) aufgebaut ist, über den das `GPSLocationRetriever`[20]-Modul die GPS-Koordinaten empfängt.
- WLAN-Anbindung des Pocket PC für die Kommunikation mit dem Server.

[20] Dieses Modul dient der Positionsbestimmung per GPS.

- Laptop, auf dem der Server ausgeführt wird. Das Laptop ist mit dem „Windows XP“-Betriebssystem ausgestattet und verfügt über einen „Intel Pentium M“-Prozessor mit 1.6 GHz sowie 512 Megabyte DDR-Ram.

Voraussetzungen: Der Server wurde gestartet und ein Szenario ausgewählt, zu dem sich der Client verbinden soll.

Testablauf:

1. Herstellen der GPS und der WLAN-Verbindung
2. Anmelden des Clients beim Server, auf dem das Szenario `offis.xml` gestartet ist
3. Anzeigen der Karte in der GUI
4. Abmelden des Clients

Auswertung: Nach dem Anmelden des Clients wurde die Karte in der GUI angezeigt. Nach einer kurzen Verzögerung wurde die erste Position von dem GPS-Receiver empfangen. Die eigene Position wurde korrekt auf der Karte angezeigt. Die protokollierte Kommunikation zwischen Client und Server bestätigt, dass ein `PositionUpdateEvent` vom Client gesendet und vom Server empfangen wurde. Die Positionen der Mitspieler wurden aufgrund der fehlenden Implementierungen dieser Funktion nicht auf der Karte angezeigt. Somit wurden auch keine Positionsdaten vom Server an die Clients der Mitspieler gesendet. Der Test gilt somit als nicht bestanden.

---

2. Testlauf

Datum: 20.09.2005

Stand der Software: Die Software befindet sich auf dem finalen Entwicklungsstand. Die für diesen Testlauf notwendigen Funktionen sind implementiert.

Tester: FJ

Setup: Für den Test werden ein Server und drei Clients benötigt. Zwei der Clients werden auf einem Pocket PC mit „Windows Mobile 2003“-Betriebssystem von Microsoft gestartet, ein dritter im Pocket PC 2003 Emulator von Microsoft. Der Client auf dem Emulator wird lediglich gestartet, damit sich die beiden Clients auf den Pocket PCs im selben Team befinden. Beide Clients auf den Pocket PCs werden mit den GPS-Empfängern verbunden, der Client des Emulators wird mit dem CenterMapLocationRetriever[21]-Modul gesteuert. Der Server wird auf einem stationären PC gestartet, auf dem auch der Client auf dem Pocket PC Emulator 2003 von Microsoft gestartet wird.

- 2 Pocket PCs (HP iPaq 5450), für die jeweils eine Bluetooth-Verbindung zum GPS-Empfänger (Falcom NAVI-1) aufgebaut ist, über den das `GPSLocationRetriever`[22]-Modul die GPS-Koordinaten empfängt.

[21] Dieses Modul dient dazu eine Position durch das Klicken auf der Karte festzulegen.
[22] Dieses Modul dient der Positionsbestimmung per GPS.

- WLAN-Anbindung der beiden Pocket PC für die Kommunikation mit dem Server.
- Laptop, auf dem der Server und ein Client auf dem Emulator ausgeführt werden. Das Laptop ist mit dem „Windows XP“-Betriebssystem ausgestattet und verfügt über einen „Intel Pentium M“-Prozessor mit 1.6 GHz sowie 512 Megabyte DDR-Ram.

Voraussetzungen: Der Server wurde gestartet und ein Szenario ausgewählt, zu dem sich die Clients verbinden sollen.

Testablauf:

1. Herstellen der GPS- und der WLAN-Verbindung der Pocket PCs
2. Anmelden des Clients beim Server, auf dem das Szenario `grenadierweg.xml` gestartet ist
3. Anzeigen der Karte in der GUI auf beiden Clients und Beobachtung der Positionsveränderungen
4. Beobachtung der Log-Ausgaben in der Konsole des Servers
5. Abmelden des Clients

Auswertung: Nach dem Anmelden der beiden Clients wurden die Karten in der GUI angezeigt. Nach einer kurzen Verzögerung wurden die erste Positionen von den GPS-Receivern empfangen. Die eigene Position sowie die Position des Mitspielers wurden auf beiden Geräten korrekt auf der Karte angezeigt. Die protokollierte Kommunikation zwischen Client und Server bestätigt, dass ein `PositionUpdateEvent` vom Client gesendet, vom Server empfangen und an den jeweils anderen Client verschickt wurde. Der Test gilt somit als erfolgreich absolviert.

15.5.3.2 Startpunkt übermitteln

Nachdem sich ein Client zum Server verbunden hat, muss er einem Team zugordnet werden und einen Startpunkt erhalten. Dieser wird dem Client als Point-of-Interest (Poi) übermittelt und ihm auf der Karte nur im passiven Zustand angezeigt.

Verantwortliche(r): SK

Pass-Kriterien:

- Der Spieler wird einem Team zugeordnet.
- Der Spieler sieht das Poi des Startpunktes.

Fail-Kriterien:

- Es stehen keine zwei Startpunkte auf dem Server zur Verfügung.
- Das Spielobjekt *Player* des Clients kann seinen Startpunkt nicht sehen.
- Der Poi des Startpunktes wird dem Client nicht übermittelt.

---

1. Testlauf

Datum: 11.08.2005

Stand der Software: Sehr früher Stand der Software; Client und Server können fehlerfrei gestartet werden, der Client kann sich beim Server anmelden.

Tester: SK

Setup:

- PC mit Windows 2000 Betriebssystem als Server
- Microsoft Visual Studio .NET 2003 (MSVS)
- Microsoft Pocket PC 2003 Emulator
- Variante der Client-Software mit „Click and Follow“ Location Retriever

Voraussetzungen: Der Server ist hochgefahren und ein Spiel zu dem Szenario aus der Datei `scenario.xml` gestartet. Der Client ist gestartet und auf dem Server angemeldet.

Testablauf:

1. Prüfen, ob auf dem Server zwei Startpunkte zur Verfügung stehen.
2. Auf dem Server prüfen, dass der Client einem Team zugeteilt wurde.
3. Auf dem Server prüfen, dass das *Player*-Spielobjekt des Clients seinen Startpunkt sehen kann.
4. Auf dem Client prüfen, ob der richtige Startpunkt als Poi dargestellt wird.

Auswertung: Keine der Fail-Kriterien traten ein. Der Spieler wird einem Team zugeordnet und sieht seinen Startpunkt. Der Test wurde erfolgreich durchgeführt.

15.5.3.3 Client aktivieren

Befindet sich der Client im passiven Zustand, muss er sich durch Erreichen seines Startpunktes aktivieren können. Dazu muss der Startpunkt angezeigt werden. Nach der Aktivierung darf der Startpunkt nicht mehr angezeigt werden, dafür aber alle Pois die im passiven Zustand nicht sichtbar sind.

Verantwortliche(r): SK

Pass-Kriterien:

- der Spieler aktiviert sich durch Erreichen seines Startpunktes
- der Startpunkt wird aus der Liste der Pois nach dem Aktivieren entfernt
- die anderen Pois werden sichtbar

Fail-Kriterien:

- der Server zeigt keine Reaktion auf die Nähe des Clients zum Startpunkt
- der Zustand des *Player*-Spielobjekts auf dem Server wird nicht auf aktiv gesetzt
- das *Player*-Spielobjekt sieht noch seinen Startpunkt
- das *Player*-Spielobjekt sieht die anderen Pois nicht
- die Poi-Events werden nicht an den Client übermittelt
- die Pois werden nicht entsprechend der Events auf dem Client dargestellt

---

1. Testlauf

Datum: 11.08.2005

Stand der Software: Sehr früher Stand der Software; Client und Server können fehlerfrei gestartet werden, der Client kann sich beim Server anmelden.

Tester: SK

Setup:

- PC mit Windows 2000 Betriebssystem als Server
- Microsoft Visual Studio .NET 2003 (MSVS)
- Microsoft Pocket PC 2003 Emulator
- Variante der Client-Software mit „Click and Follow“ Location Retriever

Voraussetzungen: Der Server ist hochgefahren und ein Spiel zu dem Szenario aus der Datei `scenario.xml` gestartet. Der Client ist gestartet und auf dem Server angemeldet.

Testablauf:

1. Mit Hilfe des „Click and Follow“ Location Retrievers wird der Client in die Nähe des Startpunktes bewegt.
2. Der Server stellt die Nähe fest.
3. Prüfen, dass das *Player*-Spielobjekt auf dem Server auf *aktiv* gesetzt wird.
4. Prüfen, dass das *Player*-Spielobjekt auf dem Server andere Pois sieht, aber nicht mehr den eigenen Startpunkt.
5. Prüfen, dass die entsprechenden Poi-Events dem Client mitgelteilt werden.
6. Prüfen, dass das Startpunkt-Poi nicht mehr auf dem Client dargestellt wird, jedoch die anderen Pois wieder.

Auswertung: Keine der Fail-Kriterien traten ein. Der Spieler konnte sich an seinem Startpunkt aktivieren und die richtigen Pois wurden angezeigt. Der Test wurde erfolgreich durchgeführt.

15.5.3.4 Sehen von Goodies

Dieser Test überprüft das korrekte Erspähen von Goodies.

Verantwortliche(r): MP

Pass-Kriterien: Ist der Spieler aktiv und enttarnt, sieht er alle Goodies, die sich in seinem Sichtradius oder dem einer seiner Aufklärungskameras befinden.

Fail-Kriterien: Der Spieler darf das Goodie nicht sehen, wenn es sich nicht in seinem oder dem Aufklärungsradius einer seiner Kameras befindet.

---

1. Testlauf

Datum: 12.08.2005

Stand der Software: Fortgeschrittener Stand der Software. Der Integrationstest *Senden / Anzeigen der Position* (15.5.3.1) wurde erfolgreich durchgeführt. Dem Client bekannte Points-of-Interests werden in der Kartenansicht dargestellt.

Tester: MP

Setup:

- Pocket PC 2003 Emulator mit `MapCenterLocationRetriever`[23]
- Windows PC, auf dem Emulator und Server ausgeführt werden.

Voraussetzungen:

- der Client hat sich beim Server angemeldet
- die eigene Position wird angezeigt
- der Sichtradius des Spielers wird korrekt angezeigt
- der Spieler befindet sich im Spielzustand aktiv
- die für den Test benötigten Spielobjekte - in diesem Fall ein Goodie - wurden erzeugt
- das Goodie ist auf der linken oberen Ecke der Karte positioniert

[23] Dieses Modul dient als Testersatz für die Positionsbestimmung per GPS. Die aktuelle Position wird berechnet aus dem Mittelpunkt des sichtbaren Kartenausschnitts.

Testablauf:

1. Die Kartenansicht wird so eingestellt, dass die linke, obere Ecke sichtbar ist.

Auswertung: Das Goodie wurde angezeigt. Der Testfall gilt als bestanden.

---

2. Testlauf Der Testlauf muss erneut durchgeführt werden, weil sich die Anzeigebedingungen für Goodies verändert haben. Beim 1. Testlauf sah der Spieler noch alle Goodies auf der Karte. Beim aktuellen Stand der Software sollte er nur noch Goodies sehen, die sich in seinem Sichtradius befinden.

Datum: 02.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- Windows PC als Server
- HP iPAQ 5450, WLAN
- GPS-Maus

Voraussetzungen:

- der Spieler nimmt an dem Spiel teil und befindet sich im Spielzustand aktiv
- es existieren nicht aufgesammelte Goodies auf der Karte, die der Spieler erreichen kann
- der Spieler ist nicht in Sichtweite eines Goodies

Testablauf:

1. Der Spieler begibt sich in die Nähe eines Goodies, so dass es sich in seinem Sichtradius befindet. Das Programm reagiert korrekt, indem es die Position des Goodies durch ein Goodie-Symbol markiert, und mittels multimodaler Ausgabe die Sichtung des Goodies mitteilt.
2. Der Spieler entfernt sich von dem Goodie, bis es den Sichtradius verlässt. Das Goodie-Symbol verschwindet und das Programm generiert eine entsprechende multimodale Meldung.
3. Der Spieler setzt eine Kamera in die Nähe des Goodies, so dass sich das Goodie im Sichtradius der Kamera befindet und entfernt sich dann von dem Goodie. Das Goodie-Symbol wird korrekterweise weiterhin angezeigt.
4. Der Spieler tarnt sich und kann jetzt das Goodie nicht mehr sehen.
5. Nachdem sich der Spieler wieder enttarnt hat, sieht er das Goodie wieder.
6. Der Spieler wird deaktiviert. Daraufhin verschwindet das Goodie-Symbol von der Karte.

Auswertung: Das Goodie wurde in den korrekten Situationen angezeigt. Der Test gilt als bestanden.

15.5.3.5 Sehen von Startpunkten

Dieser Test überprüft das korrekte Sehen von Startpunkten.

Verantwortliche(r): MP

Pass-Kriterien: Ist der Spieler im Spielzustand passiv, sieht er den Startpunkt seines Teams.

Fail-Kriterien:

- der Spieler sieht seinen Startpunkt, wenn er sich nicht im Spielzustand passiv befindet
- der Spieler sieht den Startpunkt des anderen Teams

---

1. Testlauf

Datum: 02.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- Windows PC als Server
- HP iPAQ 5450, WLAN
- GPS-Maus

Voraussetzungen: Der Spieler nimmt an dem Spiel teil und befindet sich im Spielzustand passiv.

Testablauf:

1. Der Spieler sieht seinen Startpunkt. Anhand der Server-GUI wird bestätigt, dass es sich dabei um den Startpunkt seines eigenen Teams handelt.
2. Der Spieler aktiviert sich und der Startpunkt wird korrekterweise nicht mehr angezeigt.

Auswertung: Der richtige Startpunkt wurde in den korrekten Situationen angezeigt. Der Test gilt als bestanden.

15.5.3.6 Sehen von Missionszielen

Dieser Test überprüft das korrekte Sehen von Missionszielen.

Verantwortliche(r): MP

Pass-Kriterien: Missionsziele können nur vom Commander gesehen werden, wenn er sich im Spielzustand aktiv befindet und nicht getarnt ist, und sich das Missionsziel in seinem Sichtradius oder dem einer seiner Aufklärungskameras befindet.

Fail-Kriterien:

- das Missionsziel ist im Sichtradius des Commanders und wird nicht angezeigt, obwohl der Commander enttarnt und aktiv ist
- das Missionsziel ist im Sichtradius einer Kamera des Commanders und wird nicht angezeigt, obwohl der Commander enttarnt und aktiv ist
- das Missionsziel wird anderen Spielern angezeigt
- das Missionsziel wird angezeigt, obwohl der Commander getarnt ist und/oder sich nicht im Spielzustand aktiv befindet und/oder es sich nicht im Sichtradius des Commanders bzw. seiner Kamera befindet

---

1. Testlauf

Datum: 02.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- Windows PC als Server
- HP iPAQ 5450, WLAN
- GPS-Maus

Voraussetzungen: Der Spieler nimmt an dem Spiel teil und befindet sich im Spielzustand passiv.

Testablauf:

1. Der Spieler sieht seinen Startpunkt. Anhand der Server-GUI wird bestätigt, dass es sich dabei um den Startpunkt seines eigenen Teams handelt.
2. Der Spieler aktiviert sich und der Startpunkt wird korrekterweise nicht mehr angezeigt.

Auswertung: Das Missionsziel wurde in den korrekten Situationen angezeigt. Der Test gilt als bestanden.

15.5.3.7 Sehen von Mitspielern

In diesem Test wird überprüft, ob Mitspieler in den korrekten Situationen gesehen werden.

Verantwortliche(r): MP

Pass-Kriterien: Ist der Spieler aktiviert und enttarnt, sieht er alle aktivierten, nicht getarnten Mitspieler.

Fail-Kriterien:

- der Spieler sieht einen Mitspieler, obwohl er selbst und/oder der Mitspieler getarnt ist/sind
- der Spieler sieht einen Mitspieler, obwohl er selbst und/oder der Mitspieler sich nicht im Spielzustand aktiv befindet/befinden

---

1. Testlauf

Datum: 05.09.2005

Stand der Software: Die Implementierung der Features ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- PC mit Windows XP, AMD Athlon XP 2000+, 512 MB DDR RAM
- HP iPAQ 5450 mit Internetverbindung über ActiveSync
- Pocket PC 2003 Emulator

Voraussetzungen: Zwei Spieler A und B aus dem gleichen Team nehmen an dem Spiel teil und sind passiv.

Testablauf:

1. Die Spieler werden nah beieinander positioniert, so dass sie sich im Sichtradius des jeweils anderen befinden. Die Spieler sehen sich nicht.
2. Die Spieler werden über den Server aktiviert und sehen nun ihren jeweiligen Mitspieler.
3. Spieler A tarnt sich. Die Spieler können sich nun nicht mehr sehen.

4. A enttarnt sich wieder. Die Spieler sehen einander wieder.

5. A entfernt sich von B, so dass sich beide nicht mehr im Sichtradius des jeweils anderen befinden. Die Spieler sehen sich weiterhin, da sie im gleichen Team sind.

Auswertung: Der Test gilt als bestanden.

15.5.3.8 Sehen von Gegnern

In diesem Test wird überprüft, ob das Sehen von gegnerischen Spielern ordnungsgemäßg funktioniert. Es gibt keine direkten Unterschiede beim Sehen von Gegnern und Mitspielern, deshalb werden die Pass- und Fail-Kriterien aus *Sehen von Mitspielern* (Abschnitt:15.5.3.8, S.389) übernommen. Zusätzliche Kriterien sind hier aufgelistet.

Verantwortliche(r): MP

Pass-Kriterien:

- ist der Spieler aktiviert und enttarnt, sieht er alle aktivierten, nicht getarnten Gegner, die sich in seinem Aufklärungsradius befinden
- ist der Spieler aktiviert und enttarnt, sieht er außerdem alle aktivierten Gegner, die sich im Aufklärungsradius einer seiner Kameras befinden

Fail-Kriterien: Der Spieler sieht einen Mitspieler, obwohl er selbst und/oder der Gegner sich nicht im Spielzustand aktiv befindet/befinden.

---

1. Testlauf

Datum: 02.09.2005

Stand der Software: Die Implementierung der Features ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- PC mit Windows XP, AMD Athlon XP 2000+, 512 MB DDR RAM
- HP iPAQ 5450 mit Internetverbindung über ActiveSync
- Pocket PC 2003 Emulator

Voraussetzungen: Zwei Spieler A und B aus verschiedenen Teams nehmen an dem Spiel teil und sind passiv.

Testablauf:

1. Die Spieler werden nah beieinander positioniert, so dass sich sich im Sichtradius des jeweils anderen befinden. Die Spieler sehen sich nicht.
2. Die Spieler werden über den Server aktiviert und sehen nun ihren jeweiligen Gegner.
3. Spieler A tarnt sich. Die Spieler können sich nun nicht mehr sehen.
4. A enttarnt sich wieder. Die Spieler sehen einander wieder.
5. A platziert eine Kamera und entfernt sich von B, so dass sich beide nicht mehr im Sichtradius des jeweils anderen befinden. A kann B weiterhin sehen, B sieht A aber nicht. Dafür sieht B die Kamera von A.
6. Nun tarnt sich B, aber A sieht ihn aufgrund der Kamera weiterhin.

Auswertung: Der Test gilt als bestanden.

15.5.3.9 Sehen von Kameras

Dieser Test überprüft das korrekte Erspähen von Points-of-Interests durch Kameras.

Verantwortliche(r): MP

Pass-Kriterien: Der Spieler sieht neben seiner eigenen Kamera auch alle Kameras, die sich in seinem oder dem Aufklärungsradius einer seiner Kameras befinden. Einschränkungen dazu werden durch die Fail-Kriterien spezifiziert.

Fail-Kriterien:

- der Spieler sieht eine Kamera, die nicht seine ist, obwohl er getarnt ist
- der Spieler sieht eine Kamera, obwohl er sich nicht im Spielzustand aktiv befindet
- der Spieler sieht eine Kamera, die sich nicht in seinem oder dem Aufklärungsradius einer seiner Kameras befindet

---

1. Testlauf

Datum: 02.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- Windows PC als Server, AMD Athlon XP 2000+, 512 MB DDR RAM
- HP iPAQ 5450 mit Internetverbindung über ActiveSync
- Pocket PC 2003 Emulator

Voraussetzungen: Es nehmen zwei Spieler A und B an dem Spiel teil, gehören verschiedenen Teams an und sind passiv.

Testablauf:

1. Spieler A aktiviert sich und setzt eine Kamera. Der Spieler sieht die Kamera und deren Aufklärungsradius.
2. Spieler A tarnt sich und sieht nun den Aufklärungsradius der Kamera nicht mehr, die Kamera an sich wird aber weiterhin angezeigt.
3. Spieler B begibt sich in den Sichtradius der Kamera. Da er sich noch im Spielzustand passiv befindet, sieht er weder die Kamera, noch sieht die Kamera ihn.
4. Spieler B wird nun serverseitig aktiviert. Er sieht die Kamera.
5. Spieler A enttarnt sich und kann nun Spieler B sehen, da er sich im Aufklärungsradius der Kamera befindet.
6. Nun tarnt sich Spieler B und kann in Folge dessen die Kamera nicht mehr sehen. Spieler A sieht Spieler B aufgrund der Kamera weiterhin.
7. Spieler B setzt eine eigene Kamera. Die Kameras erspähen sich gegenseitig, so dass der nicht getarnte Spieler A auch die Kamera des Spielers B sehen kann.

Auswertung: Der Test gilt als bestanden, da die Pass-Kriterien erfüllt und keines der Fail-Kriterien verletzt wurde.

15.5.4 Kartendarstellung

15.5.4.1 Zoomen

Die Karten der einzelnen spielbaren Szenarien liegen in verschiedenen Auflösungen, resp. Zoomstufen vor. Während des Spielverlauf kann es, je nach Situation, nötig sein, näher an das Spielgeschehen heranzuzoomen oder sich durch ein Herauszoomen einen Überblick zu verschaffen. Zum Ausführen des Zooms stehen im MapPanel jeweils ein Knopf zum Hereinzoomen (Beschriftung „+“) und ein Knopf zum Herauszoomen (Beschriftung „-“) zur Verfügung.
Durch den folgenden Test wird überprüft, ob das Hinein- bzw. Herauszoomen in den verschiedenen Szenarien fehlerfrei funktioniert.

Verantwortliche(r): DN

Pass-Kriterien:

- beim Klicken auf den Knopf zum Hineinzoomen, wird der Kartenausschnitt vergrößert dargestellt (gibt es keine größere Zoomstufe für die aktuelle Karte, so wird der angezeigte Kartenausschnitt beibehalten)
- beim Klicken auf den Knopf zum Herauszoomen, wird der Kartenausschnitt verkleinert dargestellt (gibt es keine kleinere Zoomstufe für die aktuelle Karte, so wird der angezeigte Kartenausschnitt beibehalten)

Fail-Kriterien:

- beim Klicken auf den Knopf zum Hineinzoomen, wird der Kartenausschnitt verkleinert
- beim Klicken auf den Knopf zum Herauszoomen, wird der Kartenausschnitt vergrößert
- beim Erreichen der kleinsten bzw. größten Zoomstufe führt ein abermaliges Betätigen des jeweiligen Knopfes zu einem Absturz des Programms

---

1. Testlauf

Datum: 07.09.2005

Stand der Software: Die Implementierung der grafischen Benutzungsoberfläche, sowie aller für das Zoomen benötigten Klassen ist abgeschlossen. Alle NUnit-Tests auf Clientseite sowie im Paket *Common* werden erfolgreich durchlaufen.

Tester: DN

Setup: Für den Test wird ein Client benötigt, der keine Verbindung zu einem Server herstellen muss. In diesem Testlauf wird auf einem PC der Pocket PC 2003 Emulator von Microsoft gestartet, auf dem der NABB Client mit dem „Click and Follow“ LocationRetriever läuft.

Konfiguration PC (Client)

- AMD Athlon 2600+, 1 GB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator
- Variante der Client-Software mit „Click and Follow“ Location Retriever

Voraussetzungen: Der Client wurde gestartet und es wurde ein Szenario geladen. Die aktuelle Ansicht ist das MapPanel.

Testablauf:

1. Die Karte befindet sich in der höchsten Zoomstufe.
2. Anwählen des Knopfes zum Hineinzoomen.
3. Anwählen des Knopfes zum Herauszoomen, bis die kleinste Zoomstufe erreicht wurde.
4. Anwählen des Knopfes zum Herauszoomen.
5. Anwählen des Knopfes zum Hineinzoomen, bis die größte Zoomstufe erreicht wurde.

Auswertung: Der Test wurde mit jedem aktuell vorhandenen Szenario (Lerigauweg, Grenadierweg, Offis, Lerchenweg und Uni Oldenburg) erfolgreich durchgeführt. Das Hinein- bzw Herauszoomen haben fehlerfrei funktioniert. Befand sich die Karte in der kleinsten bzw. größten Zoomstufe, so waren die jeweiligen Knöpfe zum Hinein- bzw Herauszoomen ausgegraut.

15.5.4.2 Darstellung von POIs und eigener Position

Im Agentenspiel auftretende Points-of-Interest (Pois) sollen auf der Karte als transparente Bitmaps angezeigt werden.

Verantwortliche(r): AB

Pass-Kriterien:

- die Pois werden mit dem jeweiligen korrekten Bitmap dargestellt
- die Bitmaps werden mit transparenten Hintergund dargestellt

Fail-Kriterien:

- es werden keine Bitmaps gezeichnet
- die Bitmaps haben keinen transparenten Hintergrund

1. Testlauf

Datum: 22.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: AB

Setup: Für den Test werden ein Server und zwei Clients benötigt. Um alle möglichen Kombinationen von anzeigbaren Pois zu testen, müssen sich die Clients einmal im gleichem Team und einmal in unterschiedlichen Teams befinden.

Setup:

- PC mit Windows XP
- Dell Axim X30 mit Internetverbindung über WLAN
- Pocket PC 2003 Emulator

Voraussetzungen: Der Server ist gestartet, Szenario *offis.xml* ist geladen und zwei Clients sind auf dem Server unter den Namen *Client 1* und *Client 2* angemeldet. Für die Verteilung der Fähigkeitspunkte werden die Standardeinstellungen beibehalten.

Testablauf:

1. Client 1 und 2 begeben sich zum Startpunkt.
2. Client 1 begibt sich in die Nähe von Client 2.
3. Client 1 begibt sich in die Nähe eines Goodies.
4. Client 1 führt die Funktion Tarnen aus.
5. Client 1 führt die Funktion Abhören aus.
6. Client 1 (im Team Infiltrator) begibt sich in die Nähe des Missionsziels.

Auswertung: Der Test verlief erfolgreich. Alle Pois wurden korrekt im zugehörigen transparentem Bitmap dargestellt.

15.5.5 Testfälle: Kontext

15.5.5.1 Qualifizierte Positionsdaten

Neben den Daten über den eigenen Standort enthalten die Informationen des GPS-Systems noch Angaben wie z.B. die Bewegungsgeschwindigkeit und die Blickrichtung des Empfängers sowie Informationen über die Qualität der empfangenen Signale. Diese werden für die Bestimmung des Kontexts eines Spielers einbezogen. Es wird getestet, ob sich die qualitativen Informationen der GPS-Daten ändert, wenn man die Empfangsbedingungen verändert.

Verantwortliche(r): FJ

Pass-Kriterien:

- Veränderung der GPS-Informationen über die Bewegungsgeschwindigkeit, Bewegungsrichtung und die Angaben über die Qualität der GPS-Daten bei einem Wechsel der Bewegungsgeschwindigkeit, der Bewegungsrichtung und der Empfangsbedingungen (z.B. durch einen Standortwechsel von draußen in ein Gebäude)
- Protokollierung der einzelnen Qualitätsänderungen (Richtung, Geschwindigkeit und Angaben über die Qualität der Signale) in der Log-Datei
- Benachrichtigung der Observer bei Änderung der Qualitäten (Protokollierung der Aufrufe in einer Log-Datei)

Fail-Kriterien:

- die GPS-Informationen ändern sich nicht trotz veränderter Bedingungen
- die Observer werden trotz Qualitätsänderung nicht benachrichtigt
- Protokolldatei weist keine Änderung der qualitativen Informationen auf

---

1. Testlauf

Datum: 08.09.2005

Stand der Software: Der Software werden keine neuen Funktionen mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern.

Tester: FJ

Setup:

- Pocket PC (HP iPaq 5450), für den eine Bluetooth-Verbindung zum GPS-Empfänger (Falcom NAVI-1) aufgebaut ist, über den das `GPSLocationRetriever`[24]-Modul die GPS-Koordinaten empfängt
- WLAN-Anbindung des Pocket PC für die Kommunikation mit dem Server

[24] Dieses Modul dient der Positionsbestimmung per GPS.

- Laptop, auf dem der Server ausgeführt wird. Das Laptop ist mit dem „Windows XP“-Betriebssystem ausgestattet und verfügt über einen „Intel Pentium M“-Prozessor mit 1.6 GHz sowie 512 Megabyte DDR-Ram.

Voraussetzungen: Der Server wurde gestartet und ein Szenario ausgewählt, zu dem sich der Client verbinden soll.

Testablauf:

1. Positionierung im Freien (bei wolkenlosem Himmel) in WLAN-Reichweite.
2. Herstellen der GPS- und der WLAN-Verbindung.
3. Anmelden des Clients beim Server, auf dem das Szenario `offis.xml` gestartet ist.
4. Änderung der Bewegungsgeschwindigkeit und der Bewegungsrichtung.
5. Änderung der Empfangsbedingungen durch Standortwechsel vom Freien in ein Gebäude.
6. Abmelden des Clients.

Auswertung: Die Einträge der Protokolldatei haben ergeben, dass das GPS-System Daten über die Änderungen der Bewegungsrichtung und der Bewegungsgeschwindigkeit geliefert hat. Es haben sich ebenfalls die Angaben über die Qualität der empfangenen Signale nach dem Standortwechsel vom Freien in ein Gebäude geändert. In allen Fällen wurden die Observer über die Veränderungen der qualitativen Informationen benachrichtigt. Der Test gilt somit als erfolgreich absolviert.

15.5.5.2 Ausgabe des Kontexts

Während des gesamten Spielverlaufs werden Informationen über den, für den Spielverlauf relevanten Kontext eines Spielers gesammelt und ausgewertet (interpretierter Kontext). Dieser wird z.B. für die Berechnung der Ausgabemodalitäten (siehe auch Abs. 15.5.6.3 auf Seite 399) herangezogen. Die Informationen über den Kontext eines Spielers sollen zudem in der Kontext-Ansicht der GUI ausgegeben werden.

Verantwortliche(r): FJ

Pass-Kriterien:

- die Informationen über den Kontext eines Spielers werden in der GUI in der Kontext-Ansicht ausgegeben
- wenn sich der Kontext eines Spielers ändert wird die Ausgabe des Kontexts in der Kontext-Ansicht der GUI aktualisiert

Fail-Kriterien:

- der Kontext wird nicht in der Kontext-Ansicht der GUI ausgegeben
- obwohl sich der Kontext eines Spielers ändert wird die Kontext-Ansicht der GUI nicht aktualisiert

---

1. Testlauf

Datum: 19.09.2005

Stand der Software: Der Software werden keine neuen Funktionen mehr hinzugefügt (finale Version), sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Die Berechnung und die Ausgabe des Kontexts sind vollständig implementiert.

Tester: FJ

Setup:

- Pocket PC (HP iPaq 5450), für den eine Bluetooth-Verbindung zum GPS-Empfänger (Falcom NAVI-1) aufgebaut ist, über den das `GPSLocationRetriever`[25]-Modul die GPS-Koordinaten empfängt
- WLAN-Anbindung des Pocket PC für die Kommunikation mit dem Server
- Laptop, auf dem der Server ausgeführt wird. Das Laptop ist mit dem „Windows XP“-Betriebssystem ausgestattet und verfügt über einen „Intel Pentium M“-Prozessor mit 1.6 GHz sowie 512 Megabyte DDR-Ram.

Voraussetzungen: Der Server wurde gestartet und ein Szenario ausgewählt, zu dem sich der Client verbinden soll.

Testablauf:

1. Herstellen der GPS- und der WLAN-Verbindung.
2. Anmelden des Clients beim Server, auf dem das Szenario `grenadierweg.xml` gestartet ist.
3. Anzeigen der Kontextansicht in der GUI.
4. Abwarten, bis der GPS-Empfänger Signale liefert (aus denen der Kontext berechnet wird).
5. Erzeugen einer Kontextänderung des Spielers, z.B. durch Änderung der Bewegungsgeschwindigkeit (von stehend zu zügig gehend).
6. Beobachtung der Kontextansicht.
7. Abmelden des Clients.

Auswertung: Nach dem Anmelden des Clients wurde zur Kartenansicht in der GUI gewechselt. Nachdem der GPS-Receiver die erste Positionsinformationen geliefert hat, wurde die eigene Position in der Karte angezeigt. Durch einen Wechsel zur Kontextansicht wurde bestätigt, dass sich der Spieler im Zustand „Stehend“ (*Standing*) befindet. Wenn sich der Spieler zügig fortbewegt, wird statt „Stehend“ der Zustand „Bewegend“ (*Moving*) in der Kontextansicht ausgegeben. Der Test gilt somit als erfolgreich absolviert.

[25] Dieses Modul dient der Positionsbestimmung per GPS.

15.5.6 Testfälle: Multimodale Ausgabe

15.5.6.1 Akustische Ausgabe

Es sollen auf den mobilen Endgeräten Audiosamples abgespielt werden.

Verantwortliche(r): AB

Pass-Kriterien: Das gewünschte Audiosample wird abgespielt.

Fail-Kriterien:

- Client stürzt ab
- es wird kein Audiosample abgespielt

---

1. Testlauf

Datum: 15.08.2005

Stand der Software: Der Software werden keine neuen Funktionen mehr hinzugefügt (finale Version), sie befindet sich in der Phase des Testens und des Behebens von Fehlern.

Tester: AB

Setup:

- PocketPC Dell Axim X30
- Windows Mobile 2003
- ein beliebiges Audiosample im Wave PCM Format.

Voraussetzungen: Der Spieler ist dem Spiel beigetreten, das abzuspielende Testsample ist auf dem PocketPC vorhanden.

Testablauf:

1. Starten der Anwendung auf dem PocketPC, starten des Spiels und Anmelden mit voreingestellten Werten des Spielers.
2. Auslösen eines Events, das eine Audionachricht abspielt.

Auswertung: Das zu Testzwecken eingesetzte Audiosample wurde korrekt wiedergegeben. Der Testlauf wurde bestanden.

15.5.6.2 Vibrationsausgabe

Dieser Test soll die Ansteuerung der Vibrationsausgabe testen. Dabei ist zu beachten, dass der Vibrationsalarm über eine virtuelle LED angesteuert wird, deren Adresse auf verschiedenen PDA unterschiedlich sein kann. Außerdem kann es sein, dass der PDA an sich keinen Vibrationsalarm besitzt. Dieser Test kann also lediglich prüfen, ob der in NABB enthaltenen Code bezüglich der Vibration ordnungsgemäß arbeitet.

Verantwortliche(r): MP

Pass-Kriterien: Der Vibrationsalarm wird erfolgreich angesteuert.

Fail-Kriterien: Der Vibrationsalarm kann der Einstellung korrekten virtuellen LED nicht angesteuert werden, obwohl der PDA über Vibrationsalarm verfügt.

---

1. Testlauf

Datum: 07.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: MP

Setup:

- Windows PC als Server
- HP iPAQ 5450
- GPS-Maus

Voraussetzungen: Der Spieler ist dem Spiel beigetreten; in der Konfigurationsdatei ist die korrekte Adresse zur Ansteuerung des Vibrationsalarms für den PDA eingetragen.

Testablauf: Der Spieler platziert ein *Beacon* auf dem Spielfeld. Die Entfernung zu dem *Beacon* wird u.a. über den Virbationsalarm dargestellt.

Auswertung: Der Vibrationsalarm wurde erfolgreich eingesetzt. Der Test wurde bestanden.

15.5.6.3 Berechnung der Ausgabemodalität in Abhängigkeit vom Kontext

Als kontextsensitives und multimodales Spiel ermöglicht NABB die Berechnung der Ausgabemodalitäten und Ausgabe aller Informationen in Abhängigkeit des aktuellen Kontexts. Dabei kann der Kontext zum einen vom Programm berechnet (z.B. über die aktuelle Bewegungsgeschwindigkeit den Kontext „rennend“

berechnen) und zum anderen vom Benutzer eingegeben sein (z.B. ob akustische Ausgaben erlaubt werden sollen). Als Rückmeldung, welche der verfügbaren Ausgabemedien (Audioausgabe, Sprachausgabe, TickerPanel, MailBox und Haptische Ausgabe) in Abhängigkeit des Kontext als geeignet erachtet werden, werden diese durch ein Label im LabelPanel dargestellt. Ist das jeweilige Label schwarz hinterlegt, so wurde die Modalität als nicht geeignet erachtet.
Durch den Test soll nun die korrekte Funktionsweise der Berechnung der Ausgabemodalität anhand des aktuellen Kontext überprüft werden.

Verantwortliche(r): DN

Pass-Kriterien:

- die Einstellungen im ContextPanel haben die korrekten Auswirkungen auf die Berechnungen der Ausgabemodalitäten
- wenn man sich im Kontext „rennend“ befindet, werden Ereignisse nicht über das TickerPanel und die MailBox ausgegeben

Fail-Kriterien:

- wenn man sich im Kontext „rennend“ befindet, werden Ereignisse trotzdem über das TickerPanel únd die MailBox ausgegeben
- obwohl eine der „Allow-Optionen“ im ContextPanel deaktiviert wurde, werden Ereignisse bzw. Zustände über diese Modalität ausgegeben
- trotz Auswahl von „Prefer Voice over Audio“ im ContextPanel wird ein Audiosignal anstatt einer Sprachdatei ausgegeben

---

1. Testlauf

Datum: 07.09.2005

Stand der Software: Die Implementierung aller für den Test benötigten Klassen ist abgeschlossen.

Tester: DN

Setup: Für den Test werden ein Server und zwei Clients benötigt. In diesem Testlauf wird ein Client über den Pocket PC 2003 Emulator von Microsoft gestartet und der andere über einen iPAQ von Hewlett Packard. Der Server läuft auf dem selben PC, auf dem auch der eine Client läuft. Da zum Testen keine Internetverbindung aufgebaut werden konnte (Infrastruktur nicht vorhanden) und der PDA somit in seiner Station stehen musste, die keine Vibration unterstützt, wurden die Vibrationsausgaben an eine Status-LED des PDA weitergeleitet.

Konfiguration PC (Server und Client 1)

- AMD Athlon 2600+, 1 GB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PDA (Client 2)

- HP iPAQ 5450

Voraussetzungen: Der Server ist gestartet und das Szenario *offis.xml* wurde geladen. Beide Clients sind auf dem Server angemeldet (unter den Namen „Client 1“ und „Client 2“) und sind verschiedenen Teams zugeordnet.

Testablauf:

1. Client 2 deaktiviert im ContextPanel die Punkte „Allow visual output“, „Allow auditive output“ und „Allow haptic output“
2. Client 2 begibt sich in den Sichtradius von Client 1

⇒ Client 2 bekommt dies nur auf der Übersichtskarte dargestellt, nicht jedoch akustisch, haptisch oder über das TickerPanel. Client 1 hingegen bekommt diese Informationen auf allen 3 Kanälen ausgegeben, wobei die Audioausgabe das Geräusch von Schritten beinhaltet.

3. Client 1 aktiviert im ContextPanel die Funktion „Prefer Voice over Audio“
4. Client 1 entfernt sich aus dem Sichtradius von Client 2 und begibt sich sofort wieder hinein

⇒ Client 1 bekommt nun als Audioausgabe die Sprachdatei „Gegner in Sicht“

5. Client 1 entfernt sich aus dem Sichtradius von Client 2
6. Client 2 setzt eine Aufklärungskamera und entfernt sich von dieser
7. Client 1 rennt in den Sichtradius der Aufklärungskamera

⇒ Client 1 bekommt die Sprachausgabe „Kontakt“ ausgegeben, im TickerPanel erscheint keine Meldung

Auswertung: Der Test wurde erfolgreich durchgeführt. Auch das Einstellen aller Kombinationsmöglichkeiten im ContextPanel (z.B. Aktivierung einer Ausgabe) führt zu den korrekten Ergebnissen.

15.5.7 Testfälle: Funktionalitäten

15.5.7.1 Tarnen

Um einem NABB-Spieler die Möglichkeit zu bieten, sich unbemerkt an eine bestimmte Position zu begeben oder sich von dieser zu entfernen, wurde der Tarnmodus eingeführt. Getarnte Spieler sind für gegnerische und eigene Spieler nicht mehr auf der Karte sichtbar. Ein getarnter Spieler sieht im Gegenzug nur noch seine eigene Position auf der Karte, nicht aber die von anderen Spielern oder Goodies. Darüber hinaus stehen einem getarnten Spieler nur noch die Fähigkeiten Neutralisieren und Enttarnen zur Verfügung.

Verantwortliche(r): JP

Pass-Kriterien:

- Spieler kann sich tarnen
- Spieler kann sich enttarnen
- im getarnten Zustand kann die Fähigkeit Neutralisieren aktiviert werden

Fail-Kriterien:

- Spieler kann sich nicht (ent-)tarnen
- im getarnten Zustand sind gegenerische Spieler und Goodies immer noch sichtbar
- die Fähigkeit Neutralisieren kann im getarnten Zustand nicht aktiviert werden
- im getarnten Zustand können die Fähigkeiten Abhören oder Aufklären aktiviert werden
- nach dem Enttarnen steht die Fähigkeit Tarnen sofort wieder zur Verfügung

---

1. Testlauf

Datum: 16.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt. Sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Tarnen“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden ein Server und zwei Clients benötigt. Sowohl der Server als auch die Clients werden auf dem selben Rechner gestartet. Um dies zu ermöglichen, müssen die Clients auf zwei verschiedenen Emulatoren laufen. So wird Client 1 auf dem Pocket PC 2002 und Client 2 auf dem Pocket PC 2003 Emulator von Microsoft gestartet.

Konfiguration PC (Server, Client 1 und Client 2)

- AMD Athlon 1800+ (1,53 GHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2002 Emulator
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Der Server ist gestartet, Szenario *offis.xml* ist geladen und zwei Clients sind auf dem Server unter den Namen *Client 1* und *Client 2* angemeldet. Für die Verteilung der Fähigkeitspunkte werden die Standardeinstellungen beibehalten. Die Clients gehören verschiedenen Teams an. Beide Clients haben ihre Startpunkte erreicht und sind aktiviert.

Testablauf:

1. Client 1: zum Haupteingang des OFFIS begeben
2. Client 2: zum Haupteingang des OFFIS begeben, bis gegnerischer Spieler im Sichtradius zu erkennen ist
3. Client 1: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenem Sichtradius befindet
4. Client 2: Tarnmodus aktivieren
5. Client 2: überprüfen, ob gegnerischer Spieler von Karte verschwindet
6. Client 1: überprüfen, ob gegnerischer Spieler von Karte verschwindet
7. Client 2: überprüfen, ob bis auf Enttarnen und Neutralisieren alle Fähigkeiten deaktiviert sind
8. Client 2: Neutralisieren aktivieren
9. Client 1: überprüfen, ob neutralisiert
10. Client 2: enttarnen und überprüfen, ob neutralisierter Spieler von Karte verschwunden ist
11. Client 2: überprüfen, ob alle Fähigkeiten wieder verfügbar sind (Tarnen, Neutralisieren und Abhören werden wieder aufgeladen)

Auswertung: Der Test verlief erfolgreich. Getarnte Spieler sind für gegenerische Spieler nicht mehr sichtbar. Die Fähigkeit Neutralisieren kann auch im getarnten Zustand ausgeführt werden.

---

2. Testlauf

Datum: 19.09.2005

Stand der Software: siehe 1. Testlauf

Tester: JP

Setup: Für den Test werden ein Server und zwei Clients benötigt. Der Server wird auf einem Windows PC gestartet. Die Clients laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden.

Konfiguration PC (Server)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003

Konfiguration PDA 1 (Client 1)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

Voraussetzungen: Der Server ist gestartet, Szenario *lerigauweg.xml* ist geladen und zwei Clients sind auf dem Server unter den Namen *Client 1* und *Client 2* angemeldet. Für die Verteilung der Fähigkeitspunkte werden die Standardeinstellungen beibehalten. Die Clients gehören verschiedenen Teams an. Beide Clients haben ihre Startpunkte erreicht und sind aktiviert.

Testablauf:

1. Client 1: zur Kreuzung Lerigauweg/Kaspersweg begeben
2. Client 2: zur Kreuzung Lerigauweg/Kaspersweg begeben
3. Client 1: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenen Sichtradius befindet
4. Client 2: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenen Sichtradius befindet
5. Client 2: Tarnmodus aktivieren
6. Client 1: überprüfen, ob gegnerischer Spieler von Karte verschwindet
7. Client 2: überprüfen, ob gegnerischer Spieler von Karte verschwindet und nur noch eigene Position (ausgegraut) angezeigt wird

8. Client 2: überprüfen, ob bis auf Enttarnen und Neutralisieren alle Fähigkeiten deaktiviert sind
9. Client 2: zur Position des gegnerischen Spielers begeben
10. Client 2: Neutralisieren aktivieren
11. Client 1: überprüfen, ob neutralisiert
12. Client 2: enttarnen und überprüfen, ob neutralisierter Spieler von Karte verschwunden ist
13. Client 2: überprüfen, ob alle Fähigkeiten wieder verfügbar sind (Tarnen, Neutralisieren und Abhören werden wieder aufgeladen)

Auswertung: Der Test verlief ohne Probleme. Der getarnte Spieler wurde nicht mehr auf der Karte des anderen Spielers angezeigt und konnte diesen neutralisieren. Während des Tarnens konnten lediglich die Fähigkeiten Enttarnen und Neutralisieren aktiviert werden.

15.5.7.2 Neutralisieren

Das Neutralisieren von Mitspielern gehört neben Tarnen, Aufklären und Abhören zu den vier Fähigkeiten einer Spielers. Der Test besteht aus vier Aspekten. Auf dem Client wird getestet, ob die „Neutralisieren"-Fähigkeit korrekt auflädt, sich aktiviert und deaktiviert und der Server neutralisierte Spieler auf den Spielerzustand passiv setzt. Des weiteren muss ein neutralisierter Spieler alle bis dahin aufgesammelten *Goodies* verlieren, damit andere Spieler diese wieder einsammeln können. Die *Goodies* werden an seinem Standort platziert. Als letzer Aspekt wird das Versenden einer Auswertungsnachricht, den sogenannten „NABB stats", getestet.

Verantwortliche(r): SA

Pass-Kriterien:

- die „Neutralisieren"-Fähigkeit lädt auf und aktiviert sich
- es wird ein Neutralisierungradius gezeichnet
- alle Spieler im Neutralisierungradius werden nach fünf Sekunden auf passiv gesetzt

Fail-Kriterien:

- neutralisierte Spieler verlieren ihre aufgenommenen Goodies nicht
- der neutralisierende Spieler erhält keine Nachricht mit dem Neutralisierungsergebnis, wenn ein Spieler tatsächlich neutralisiert wurde
- der neutralisierende Spieler wird selbst passiv nach einem Neutralisierungsvorgang

---

1. Testlauf In diesem Testlauf soll überprüft werden, ob ein neutralisierter Spieler alle aufgenommenen Goodies verliert.

Datum: 07.09.2005

Stand der Software: Alle für die Ausführung der Funktion Neutralisieren relevanten Programmteile sind implementiert.

Tester: JP

Setup: Für den Test werden ein Server und zwei Clients benötigt. In diesem Testlauf werden alle Clients über den Pocket PC 2003 Emulator von Microsoft gestartet. Um dies zu ermöglichen, werden zwei PCs benötigt.

Konfiguration PC 1 (Server und Client 1)

- Intel Pentium M (1,6 GHz), 1 GB RAM
- Microsoft Windows XP Home Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PC 2 (Client 2)

- AM Athlon XP 1800+ (1,53 GHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Der Server ist gestartet und das Szenario *offis.xml* wurde geladen. Beide Clients sind auf dem Server angemeldet (unter den Namen „Client 1“ und „Client 2“) und sind verschiedenen Teams zugeordnet.

Testablauf:

1. Client 1: Startpunkt aufsuchen und aktiviert werden
2. Client 1: Goodie oben links auf dem Spielfeld suchen und aufnehmen
3. Client 1: Goodie unten rechts auf dem Spielfeld suchen und aufnehmen
4. Client 1: im MenuPanel überprüfen, ob beide Goodies angezeigt werden
5. Client 2: Startpunkt aufsuchen und aktiviert werden
6. Client 2: Client 1 in der unteren rechten Ecke suchen
7. Client 2: Client 1 neutralisieren
8. Client 1: im MenuPanel überprüfen, ob Goodies verloren wurden
9. Client 1: Startpunkt aufsuchen und aktiviert werden

10. Client 2: Goodies in der rechten unteren Ecke finden
11. Client 2: erstes Goodie aufnehmen
12. Client 2: zweites Goodie aufnehmen
13. Client 2: im MenuPanel überprüfen, ob beide Goodies angezeigt werden
14. Client 1: Client 2 in der unteren rechten Ecke suchen
15. Client 1: Client 2 neutralisieren
16. Client 2: im MenuPanel überprüfen, ob Goodies verloren wurden

Auswertung: Der Test wurde erfolgreich durchgeführt. Jeder Client verlor die aufgenommenen Goodies an der Stelle, an der er neutralisiert wurde. Die Goodies konnten dann vom jeweils anderen Client gefunden und aufgenommen werden.

---

2. Testlauf In diesem Testlauf wird überprüft, ob alle Spieler, sowohl Teammitglieder als auch andere Spieler, aber nicht der neutralisierende Spieler, die sich im Neutralisierungsradius befinden, auf den Spielzustand passiv gesetzt werden. Außerdem wird das Aufladen und Aktivieren der Fähigkeit „Neutralisieren“ getestet. Abschließend wird die Versendung und der Inhalt der Auswertungnachricht nach einem Neutralisationsvorgang, bei dem mindestens ein Spieler neutralisiert worden ist, überprüft.

Datum: 15.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion Neutralisieren relevanten Programmteile sind vollständig implementiert.

Tester: SA, AB, FJ

Setup:

- Pocket PC (HP iPaq 5450), für den eine Bluetooth-Verbindung zum GPS-Empfänger (Falcom NAVI-1) aufgebaut ist
- WLAN-Anbindung des Pocket PC für die Kommunikation mit dem Server
- Laptop, auf dem der Server und ein Client ausgeführt werden. Das Laptop ist mit dem „Windows XP“-Betriebssystem ausgestattet und verfügt über einen „Intel Pentium M“-Prozessor mit 1.6 GHz sowie 512 Megabyte DDR-Ram.
- ein zweiter PC für den dritten Client
- auf den beiden PCs muss Microsoft Visual Studio .NET 2003 und Microsoft Pocket PC 2003 Emulator installiert sein

Voraussetzungen:

- der Server ist gestartet und das Szenario *grenadierweg.xml* wurde geladen
- das Anmelden bei dem Server und das Versenden von *Events* funktioniert
- die Verbindung zum Server bleibt bestehen
- die Spieler befinden sich nicht an ihrem Startpunkt

Testablauf:

1. Herstellen der GPS und der WLAN-Verbindung für einen Pocket PC und Herstellen einer WLAN-Verbindung für zwei Emulatoren.
2. Konfiguration des neutralisierenden „Defenders“ mit vier Punkten auf der Eigenschaft „Neutralisieren“.
3. Anmelden aller Clients beim Server, auf dem das Szenario `grenadierweg.xml` gestartet ist.
4. Alle Spieler bewegen sich zum unteren Rand der Karte und befinden sich alle in Sichtweite und in Neutralisierungsreichweite des neutralisierenden „Defenders“.
5. Der neutralisierende „Defender“ aktiviert die voll aufgeladene und aktive „Neutralisieren“-Funktion.
6. Ein Neutralisierungskreis wird für ungefähr fünf Sekunden gezeichnet.
7. Sein Mitspieler und der „Infiltrator“ sind beide neuralisiert, befinden sich im Spielzustand passiv.
8. Der neutralisierende „Defender“ überprüft die Nachrichten-Ansicht und findet die Auswertungsnachricht des Neutralisieren-Vorgangs (NABB-Stats).
9. In der „NABB stats“-Nachricht sind sowohl der neutralisierte „Infiltrator“ und der neutralisierte „Defender“ aufgeführt.

Auswertung: In keinem Punkt des Tests konnte ein *Fail*-Kriterium erreicht werden. Alle *Pass*-Kriterien sind erreicht worden. Der Test gilt als bestanden.

15.5.7.3 Aufklären

Einem aktiven NABB-Spieler werden die Positionen von Goodies, Missionszielen oder gegnerischen Spielern nur angezeigt, wenn sich diese in seinem Sichtradius befinden. Diesen Sichtradius kann der Spieler durch das Setzen von Aufklärungskameras erweitern. Jedem Spieler stehen eine begrenzte Anzahl von Kameras zur Verfügung, die an beliebigen Positionen abgelegt werden können. Eigene Aufklärungskameras können wieder aufgenommen und gegnerische Aufklärungskameras können zerstört werden.

Verantwortliche(r): JP

Pass-Kriterien:

- Kameras können abgelegt werden
- Kameras können aufgenommen werden
- gegnerische Kameras können zerstört werden
- Hinweis erfolgt, wenn gegenerischer Spieler sich im Aufklärungsradius der eigenen Kamera befindet
- Hinweis erfolgt, wenn sich eine gegenerische Kamera im eigenen Sichtradius befindet

Fail-Kriterien:

- Kameras können nicht abgelegt oder aufgenommen werden
- gegnerische Kameras können nicht zerstört werden
- kein Hinweis, wenn gegenerischer Spieler sich im Aufklärungsradius der eigenen Kamera befindet
- kein Hinweis, wenn sich eine gegenerische Kamera im eigenen Sichtradius befindet
- zerstörte Kameras werden dem Spieler nicht wieder zur Verfügung gestellt
- neutralisierte Spieler werden weiterhin benachrichtigt, wenn ein gegenerischer Spieler den Aufklärungsradius einer Kamera betritt

---

1. Testlauf

Datum: 16.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt. Sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Aufklären“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden ein Server und zwei Clients benötigt. Sowohl der Server als auch die Clients werden auf demselben Rechner gestartet. Um dies zu ermöglichen, müssen die Clients auf zwei verschiedenen Emulatoren laufen. So wird Client 1 auf dem Pocket PC 2002 und Client 2 auf dem Pocket PC 2003 Emulator von Microsoft gestartet.

Konfiguration PC (Server, Client 1 und Client 2)

- AMD Athlon 1800+ (1,53 GHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2002 Emulator
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Der Server ist gestartet, Szenario *offis.xml* ist geladen und zwei Clients sind auf dem Server unter den Namen *Client 1* und *Client 2* angemeldet. Für die Verteilung der Fähigkeitspunkte werden die Standardeinstellungen beibehalten. Die Clients gehören verschiedenen Teams an. Beide Clients haben ihre Startpunkte erreicht und sind aktiviert.

Testablauf:

1. Client 1: zum Haupteingang des OFFIS begeben
2. Client 2: zum Haupteingang des OFFIS begeben
3. Client 1: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenen Sichtradius befindet
4. Client 2: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenen Sichtradius befindet
5. Client 2: von Position des gegnerischen Spielers entfernen, bis dieser nicht mehr im Sichtradius erkennbar ist.
6. Client 2: Aufklärungskamera setzen, Client 1 wird im Aufklärungsradius der Kamera angezeigt
7. Client 2: überprüfen, ob noch eine Kamera zur Verfügung steht
8. Client 1: auf Position des gegnerischen Spielers zubewegen, bis sich dieser im eigenen Sichtradius befindet
9. Client 1: Hinweis erfolgt, dass sich eine gegnerische Kamera im eigenem Sichtradius befindet
10. Client 1: gegnerischen Spieler neutralisieren
11. Client 2: Startpunkt aufsuchen, aktivieren lassen
12. Client 1: Kamera des gegnerischen Spielers suchen und zerstören
13. Client 2: überprüfen, ob wieder zwei Kameras zur Verfügung stehen
14. Client 2: zum letzten bekannten Standort von Client 1 begeben
15. Client 1: Aufklärungskamera setzen
16. Client 1: überprüfen, ob noch eine Kamera zur Verfügung steht
17. Client 1: sobald Client 1 im Aufklärungsradius der eigenen Kamera angezeigt wird, diese wieder aufnehmen

Auswertung: Der Test verlief ohne Probleme. Kameras können gesetzt, wieder aufgenommen und zerstört werden. Es werden Hinweise gegeben, wenn sich ein gegnerischer Spieler im Aufklärungsradius der eigenen Kamera befindet, oder wenn sich eine gegnerische Kamera im eigenen Sichtradius befindet.

---

2. Testlauf

Datum: 19.09.2005

Stand der Software: siehe 1. Testlauf

Tester: JP

Setup: Für den Test werden ein Server und zwei Clients benötigt. Der Server wird auf einem Windows PC gestartet. Die Clients laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden.

Konfiguration PC (Server)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003

Konfiguration PDA 1 (Client 1)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

Voraussetzungen: Der Server ist gestartet, Szenario *lerigauweg.xml* ist geladen und zwei Clients sind auf dem Server unter den Namen *Client 1* und *Client 2* angemeldet. Für die Verteilung der Fähigkeitspunkte werden die Standardeinstellungen beibehalten. Die Clients gehören verschiedenen Teams an. Beide Clients haben ihre Startpunkte erreicht und sind aktiviert.

Testablauf:

1. Client 1: zur Kreuzung Lerigauweg/Kaspersweg begeben
2. Client 2: zur Kreuzung Lerigauweg/Kaspersweg begeben
3. Client 1: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenen Sichtradius befindet
4. Client 2: Hinweis erfolgt, dass sich ein gegnerischer Spieler im eigenen Sichtradius befindet
5. Client 2: von Position des gegnerischen Spielers entfernen, bis dieser nicht mehr im Sichtradius erkennbar ist.
6. Client 2: Aufklärungskamera setzen, Client 1 wird im Aufklärungsradius der Kamera angezeigt

7. Client 2: überprüfen, ob noch eine Kamera zur Verfügung steht
8. Client 1: auf Position des gegnerischen Spielers zubewegen, bis sich dieser im eigenen Sichtradius befindet
9. Client 1: Hinweis erfolgt, dass sich eine gegnerische Kamera im eigenem Sichtradius befindet
10. Client 1: gegnerischen Spieler neutralisieren
11. Client 2: Startpunkt aufsuchen, aktivieren lassen
12. Client 1: Kamera des gegnerischen Spielers suchen und zerstören
13. Client 2: überprüfen, ob wieder zwei Kameras zur Verfügung stehen
14. Client 2: zum letzten bekannten Standort von Client 1 begeben
15. Client 1: Aufklärungskamera setzen
16. Client 1: überprüfen, ob noch eine Kamera zur Verfügung steht
17. Client 1: sobald Client 1 im Aufklärungsradius der eigenen Kamera angezeigt wird, diese wieder aufnehmen

Auswertung: Der Test verlief ohne Probleme. Der Sichtradius eines Spielers wird durch das Setzen einer Aufklärungskamera erweitert. Betritt ein gegnerischer Spieler den Aufklärungsradius einer Kamera, erhält dessen Besitzer eine entsprechende Warnung. Kameras können abgelegt, wieder aufgenommen und gegnerische Kameras können zerstört werden. Befindet sich eine gegnerische Kamera im Sichtradius eines Spielers, erhält dieser eine Warnung.

15.5.7.4 Nachrichten abhören

Das Abhören einer Nachricht gehört neben Tarnen, Aufklären und Abhören zu den vier Fähigkeiten eines Spielers. Der Test besteht aus zwei Aspekten. Auf dem Client wird getestet, ob die „Abhören“-Fähigkeit korrekt auflädt, sich aktiviert und deaktiviert und ob der Server abgehörte Nachrichten an den Abhörenden versenden.

Verantwortliche(r): SA

Pass-Kriterien:

- die „Abhören“-Fähigkeit lädt auf und aktiviert sich
- das Abhören kann unterbrochen werden
- alle Nachrichten werden abgehört

Fail-Kriterien:

- die „Abhören“-Fähigkeit lädt nicht auf oder aktiviert sich nicht
- es werden nicht alle oder keine Nachrichten abgehört
- die abgehörten Nachrichten sind nicht identisch mit den versendeten Nachrichten

---

1. Testlauf

Datum: 15.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Abhören“ relevanten Programmteile sind vollständig implementiert.

Tester: SA, AB, FJ

Setup:

- Pocket PC (HP iPaq 5450), für den eine Bluetooth-Verbindung zum GPS-Empfänger (Falcom NAVI-1) aufgebaut ist
- WLAN-Anbindung des Pocket PC für die Kommunikation mit dem Server
- Laptop, auf dem der Server und ein Client ausgeführt werden. Das Laptop ist mit dem „Windows XP“-Betriebssystem ausgestattet und verfügt über einen „Intel Pentium M“-Prozessor mit 1.6 GHz sowie 512 Megabyte DDR-Ram.
- ein zweiter PC für den dritten Client
- auf beiden PCs muss Microsoft Visual Studio .NET 2003 und Microsoft Pocket PC 2003 Emulator installiert sein

Voraussetzungen:

- der Server ist gestartet und das Szenario *grenadierweg.xml* wurde geladen
- das Anmelden bei dem Server, verfassen, versenden und ansehen von Nachrichten funktioniert
- die Verbindung zum Server bleibt bestehen

Testablauf:

1. Herstellen der GPS- und der WLAN-Verbindung für einen Pocket PC und Herstellen einer WLAN-Verbindung für zwei Emulatoren.
2. Konfiguration des abhörenden Infiltrators mit vier Punkten auf der Eigenschaft „Abhören“ (100% aller Nachrichten werden dadurch abgehört).
3. Anmelden aller Clients beim Server, auf dem das Szenario `grenadierweg.xml` gestartet ist. Zwei Clients sind den „Defendern“ und einer den „Infiltratoren“ zugeordnet.
4. Die „Abhören“-Funktion des Infiltrator lädt auf.
5. Der Infiltrator aktiviert die „Abhören“-Funktion. Diese ist vollständig aufgeladen.
6. Die Option das Abhören vorzeitig zu beenden wird dem „Infiltrator“ angeboten, d.h. der *Wiretap*-Knopf ist mit dem Label „Stop“ beschriftet.

7. Ein Spieler des „Defender“-Teams sendet seinem Mitspieler vier Nachrichten.
8. Der „Infiltrator“ überprüft in der Nachrichten-Ansicht, ob er alle vier Nachrichten abhören konnte.
9. Die gesendeten und abgehörten Nachrichten werden miteinander verglichen.
10. Die „Abhören“-Funktion beendet der „Infiltrator“ oder der Server.
11. Die „Abhören“-Funktion des „Infiltrators“ lädt wieder auf.

Auswertung: In keinem Punkt des Tests konnte ein *Fail*-Kriterium erreicht werden. Alle *Pass*-Kriterien sind erreicht worden. Der Test gilt als bestanden.

15.5.8 Testfälle: Nachrichten

15.5.8.1 Systemnachrichten an Clients senden

Dem Server soll es ermöglicht werden, Systemnachrichten in Form von Textnachrichten an einen Client zu schicken. Die Textnachrichten werden dann über das Tickerpanel auf dem mobilen Endgerät ausgegeben.

Verantwortliche(r): AB

Pass-Kriterien: Systemnachrichten werden erfolgreich an den Client verschickt.

Fail-Kriterien:

- Server stürzt ab
- Client stürzt ab
- Systemnachricht kommt fehlerhaft beim Client an
- Systemnachricht kommt nicht beim Client an

---

1. Testlauf

Datum: 03.08.2005

Stand der Software: Alle für die Ausführung der Funktion „Systemnachrichten an Clients senden“ relevanten Programmteile sind vollständig implementiert.

Tester: AB

Setup:

- Windows PC als Server sowie Client
- Windows Mobile 2003 Emulator
- Windows XP
- Visual Studio 2003 .NET

Voraussetzungen:

- der Client hat sich beim Server angemeldet
- funktionstüchtiges Tickerpanel zum Anzeigen der empfangenen Nachricht

Testablauf:

1. Starten des Servers, laden der Szenariondatei `offis.xml` und starten des Spiels
2. Starten des Emulators, starten des Spiels und Anmelden mit voreingestellten Werten des Spielers
3. Auswahl und Anzeigen von Nachrichten im Tickerpannel
4. Beenden des Clients sowie Servers

Auswertung: Nachrichten werden korrekt an den Client übertragen und im Tickerpanel angezeigt.

15.5.8.2 Nachrichten verfassen/versenden

Die Kommunikation zwischen den Clients erfolgt in NABB über das Versenden und Empfangen von Nachrichten. Durch den folgenden Test wird überprüft, ob das Verfassen und Versenden einer neuen Nachricht auf einem NABB-Client korrekt funktioniert.

Verantwortliche(r): JP

Pass-Kriterien:

- neue Nachricht kann auf einem Client erstellt werden (Betreff und Nachrichtentext editierbar)
- Empfänger der Nachricht können hinzugefügt und wieder entfernt werden
- Nachricht kann versendet werden
- Absender erhält Rückmeldung, wenn Absendevorgang abgeschlossen ist (Betreff und Nachrichtentext werden gelöscht, „Send“-Button wird deaktiviert)
- Nachricht kommt bei Empfänger(n) an

Fail-Kriterien:

- Betreff oder Nachrichtentext sind nicht editierbar
- Empfänger können nicht hinzugefügt oder wieder gelöscht werden
- Nachricht kann nicht an alle Teammitglieder gesendet werden
- Nachricht kann an *Nicht*-Teammitglieder gesendet werden
- Absenden der Nachricht nicht möglich
- Absenden der Nachricht möglich, obwohl Client im Zustand *passive*
- Nachricht ohne gültigen Empfänger kann versendet werden
- Nachricht ohne Eintrag im Feld *Subject* kann versendet werden
- Nachricht kommt nicht bei Empfänger(n) an

- Nachricht wird auch an Clients gesendet, die nicht in der Empfängerliste der Nachricht aufgeführt sind
- Nachricht wird an Clients gesendet, die nicht zum Team des Absenders gehören und ihren Abhörmodus nicht aktiviert haben
- Nachricht kommt fehlerhaft beim Empfänger an

---

1. Testlauf

Datum: 31.08.2005

Stand der Software: Die Implementierung aller für den Nachrichtenaustausch verantwortlichen Klassen ist abgeschlossen. Die NUnit-Tests für das (De-)Serialisieren von *Message*-Objekten und *MessageEvent*-Objekten , sowie der Test *Systemnachrichten an Clients senden* (siehe 15.5.8.1 auf Seite 415) wurden erfolgreich durchlaufen. Der Test *Nachrichten empfangen / anzeigen* (siehe 15.5.8.3 auf Seite 419) ist noch nicht durchgeführt worden.

Tester: MO, JP

Setup: Für den Test werden ein Server und drei Clients benötigt. Die beiden Clients, zwischen denen Nachrichten ausgetauscht werden sollen, müssen dem gleichen Team zugehörig sein. Der Grund dafür, dass für diesen Test drei Clients benötigt werden liegt daran, dass es serverseitig nicht zulässig ist, lediglich zwei Clients des gleichen Teams anzumelden. In diesem Testlauf werden alle Clients über den Pocket PC 2003 Emulator von Microsoft gestartet. Um dies zu ermöglichen, werden drei PCs benötigt.

Konfiguration PC 1 (Server und Client 1)

- Intel Pentium 4 (3,06 GHz), 2 GB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PC 2 (Client 2)

- Intel Pentium 3 (804 MHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PC 3 (Client 3)

- AMD Athlon (908 MHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Die NUnit-Tests für das (De-)Serialisieren von *Message*-Objekten und *MessageEvent*-Objekten, sowie der Test *Systemnachrichten an Clients senden* (siehe 15.5.8.1 auf Seite 415) wurden erfolgreich durchlaufen. Der Server ist gestartet und das Szenario *Offis* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 2* dem Team *Defender* und *Client 2* dem Team *Infiltrator* angehören.

Testablauf:

1. Client 1: Wechsel in MailPanel/Compose
2. Client 1: *Test* in Feld *Subject* eintragen
3. Client 1: *This is a test* in das Nachrichtentextfeld eintragen
4. Client 1: Wechsel in MailPanel/Recipients
5. Client 1: Eintrag *Client 3* in ComboBox auswählen und durch Klicken auf *Add*-Button bestätigen
6. Client 1: Wechsel in MailPanel/Compose
7. Client 1: Klicken auf *Send*-Button
8. Client 3: Auf Farbänderung des MailTabs achten. Empfang einer neuen Nachricht muss signalisiert werden
9. Client 3: TickerPanel beobachten: Eintrag *You've got mail!* muss angezeigt werden

Auswertung: Der Test verlief erfolgreich. Das MailTab von *Client 3* änderte die Farbe und das TickerPanel zeigte den Eintrag *You've got mail!* an.

---

2. Testlauf

Datum: 19.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt. Sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Nachrichten verfassen/versenden“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden, wie für den ersten Testlauf, ein Server und drei Clients benötigt. Client 1 läuft auf dem Pocket PC 2003 Emulator von Microsoft, auf dem gleichen Rechner wie der Server. Client 2 und Client 3 laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden. Auch in diesem Test müssen die Clients, zwischen denen Nachrichten ausgetauscht werden sollen, dem gleichen Team zugehörig sein.

Konfiguration PC (Server und Client 1)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PDA 1 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 3)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

Voraussetzungen: Der Server ist gestartet und das Szenario *lerigauweg.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und *Client 2* dem Team Infiltrator angehören.

Testablauf: siehe 1. Testlauf

Auswertung: Der Test verlief ohne Probleme. Nachrichten können verfasst und versendet werden.

15.5.8.3 Nachrichten empfangen/anzeigen

Sowohl der NABB-Server, als auch der NABB-Client verfügen über die Fähigkeit Nachrichten an andere NABB-Clients zu senden. Auf dem Client werden empfangene Nachrichten im InBoxPanel des MailPanels angezeigt. Durch den folgenden Test wird überprüft, ob empfangene Nachrichten korrekt auf dem Client angezeigt werden.

Verantwortliche(r): JP

Pass-Kriterien:

- Empfangen einer neuen Nachricht wird auf dem Client angezeigt (Eintrag im TickerPanel, Farbänderung des MailTabs)
- empfangene Nachricht wird in die ReceivedMessageListBox im InBoxPanel des MailPanels eingetragen (enthält Absender und *Subject* der Nachricht)
- bei Anwahl einer Nachricht der ReceivedMessageListBox wird der zugehörige Nachrichteninhalt in der MessageContentTextBox angezeigt

Fail-Kriterien:

- Eingang einer neuen Nachricht wird nicht über eine Farbänderung des MailTabs signalisiert
- Empfangene Nachricht wird nicht in der ReceivedMessageListBox angezeigt
- Eintrag der ReceivedMessageListBox enthält kein Präfix, das auf den Absender schliessen läßt
- Eintrag der ReceivedMessageListBox kann nicht angewählt werden
- Nachrichteninhalt wird nach Anwahl einer Nachricht nicht in der MessageContentTextBox angezeigt

---

1. Testlauf

Datum: 31.08.2005

Stand der Software: Die Implementierung aller für den Nachrichtenaustausch verantwortlichen Klassen ist abgeschlossen. Die NUnit-Tests für das (De-)Serialisieren von *Message*-Objekten und *MessageEvent*-Objekten, sowie der Test *Systemnachrichten an Clients senden* (siehe 15.5.8.1 auf Seite 415) wurden erfolgreich durchlaufen. Der erste Testlauf des Tests „Nachrichten verfassen / versenden“ (siehe 15.5.8.2 auf Seite 416) ist erfolgreich durchgeführt worden. Eine Rückmeldung über das Empfangen einer Nachricht erfolgt sowohl über den Eintrag *You've got mail!* im TickerPanel, als auch über eine Farbänderung das MailTabs.

Tester: JP

Setup: Für den Test werden ein Server und drei Clients benötigt. Die beiden Clients, zwischen denen Nachrichten ausgetauscht werden sollen, müssen dem gleichen Team zugehörig sein. Der Grund dafür, dass für diesen Test drei Clients benötigt werden liegt daran, dass es serverseitig nicht zulässig ist, lediglich zwei Clients des gleichen Teams anzumelden. In diesem Testlauf werden alle Clients über den Pocket PC 2003 Emulator von Microsoft gestartet. Um dies zu ermöglichen, werden drei PCs benötigt.

Konfiguration PC 1 (Server und Client 1)

- Intel Pentium 4 (3,06 GHz), 2 GB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PC 2 (Client 2)

- Intel Pentium 3 (804 MHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PC 3 (Client 3)

- AMD Athlon (908 MHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Die NUnit-Tests für das (De-)Serialisieren von *Message*-Objekten und *MessageEvent*-Objekten, sowie der Test *Systemnachrichten an Clients senden* (siehe 15.5.8.1 auf Seite 415) wurden erfolgreich durchlaufen. Der Server ist gestartet und das Szenario *offis.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und Client 2 dem Team Infiltrator angehören.

Testablauf:

1. gleicher Ablauf, wie im ersten Testlauf des Tests *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416)
2. Client 3: Wechsel in MailPanel/Inbox
3. Client 3: Liste empfangener Nachrichten muss Eintrag *[Client 1] Test* enthalten
4. Client 3: Diesen Eintrag anklicken
5. Client 3: Nachrichtentext *This is a test* muss im Nachrichtentextfeld angezeigt werden

Auswertung: Der Test verlief erfolgreich. Das MailTab von *Client 3* änderte die Farbe und das Ticker-Panel zeigte den Eintrag *You've got mail!* an. Die Liste der empfangenen Nachrichten enthielt den Eintrag *[Client 1] Test*. Durch Anklicken dieses Eintrages wurde der Text *This is a test* im Nachrichtentextfeld angezeigt.

2. Testlauf

Datum: 19.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Nachrichten empfangen/anzeigen“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden, wie für den ersten Testlauf, ein Server und drei Clients benötigt. Client 1 läuft auf dem Pocket PC 2003 Emulator von Microsoft, auf dem gleichen Rechner wie der Server. Client 2 und Client 3 laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden. Auch in diesem Test müssen die Clients, zwischen denen Nachrichten ausgetauscht werden sollen, dem gleichen Team zugehörig sein.

Konfiguration PC (Server und Client 1)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PDA 1 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 3)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

Voraussetzungen: Der Server ist gestartet und das Szenario *lerigauweg.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und *Client 2* dem Team *Infiltrator* angehören.

Testablauf: siehe 1. Testlauf

Auswertung: Der Test verlief ohne Probleme. Nachrichten können empfangen und angezeigt werden.

15.5.8.4 Nachrichten löschen

Jeder NABB-Client verfügt über eine Nachrichtenansicht zur Darstellung der empfangenen Nachrichten. Eine Übersicht aller empfangenen Nachrichten bietet hier eine *ListBox*. Jeder Eintrag dieser *ListBox* stellt eine Nachricht dar und besteht aus zwei Teilen:

- einem Präfix, dient der Identifikation des Absenders
- dem Betreff der Nachricht

Bei Auswahl einer Nachricht aus der *ListBox* wird der Delete-Button aktiviert, der das Löschen der Nachricht auf dem Client ermöglicht. Wird der Delete-Button gedrückt, so wird der ausgewählte Nachrichteneintrag aus der *ListBox* und die zugehörige Nachricht vom Server entfernt und der Delete-Button wieder deaktiviert. Im folgenden Test sollen einzelne Nachrichten auf dem Client gelöscht werden.

Verantwortliche(r): JP

Pass-Kriterien:

- Nachrichten der ListBox der Nachrichtenansicht können ausgewählt werden
- nach Auswahl eines Eintrages der ListBox wird der Delete-Button aktiviert
- durch Anklicken des Delete-Buttons wird der ausgewählte Eintrag der ListBox aus dieser entfernt
- nach Löschen einer Nachricht wird der Delete-Button wieder deaktiviert

Fail-Kriterien:

- Nachrichten der ListBox können nicht ausgewählt werden
- nach Auswahl eines Eintrages der ListBox wird der Delete-Button nicht aktiviert
- das Anklicken des Delete-Buttons führt nicht zum Entfernen des markierten Eintrages aus der ListBox
- nach Löschen einer Nachricht wird der Nachrichtentext weiterhin im Nachrichtentextfeld angezeigt

---

1. Testlauf

Datum: 15.08.2005

Stand der Software: Die Implementierung der Logik der Nachrichtenansicht ist noch nicht fertig. Es können Nachrichten über die ListBox ausgegeben, deren Inhalt aber noch nicht über die TextBox ausgegeben werden. Die Logik zum Löschen einer Nachricht vom Client durch Anklicken des Delete-Buttons nach Auswahl einer Nachricht ist implementiert.

Tester: JP

Setup: Für den Test werden ein Server und ein Client benötigt. Server und Client laufen auf dem gleichen PC. Der Client wird über den Pocket PC Emulator 2003 von Microsoft gestartet.

Konfiguration PC (Server und Client)

- Intel Pentium M (1,6 GHz), 1 GB RAM
- Microsoft Windows XP Home
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Der Server ist hochgefahren das Szenario *offis.xml* gestartet. Der Client ist gestartet und auf dem Server unter dem Namen *Client 1* angemeldet.

Testablauf:

1. Client1: Wechsel in MailTab/Inbox
2. Client1: beliebigen Nachrichteneintrag der ListBox markieren.
3. Client1: Anklicken des Delete-Buttons.

Auswertung: Der Nachrichteneintrag wurde aus der Liste der empfangen Nachrichten gelöscht und der Nachrichteninhalt wird nicht mehr im Nachrichtentextfeld angezeigt.

---

2. Testlauf

Datum: 19.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Nachrichten löschen“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden ein Server und ein Client benötigt. Der Client läuft auf einem PDA. Die GPS-Signale empfängt dieser über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Maus als auch das GPRS-fähige Handy sind über Bluetooth mit den PDA verbunden.

Konfiguration PC (Server)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003

Konfiguration PDA (Client)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Voraussetzungen: Der Server ist gestartet und das Szenario *lerigauweg.xml* geladen. Der Client ist auf dem Server angemeldet (unter den Namen *Client 1*).

Testablauf: siehe 1. Testlauf

Auswertung: Der Nachrichteneintrag wurde aus der Liste der empfangen Nachrichten gelöscht und der Nachrichteninhalt wird nicht mehr im Nachrichtentextfeld angezeigt.

15.5.8.5 Nachrichten beantworten

Die Funktion *Nachrichten beantworten* dient dazu, möglichst unkompliziert auf eine empfangene Nachricht antworten zu können. Im folgenden soll die korrekte Funktionsweise getestet werden.

Verantwortliche(r): JP

Pass-Kriterien:

- beantwortete Nachrichten kommen korrekt beim Empfänger an
- gleiche Pass-Kriterien wie für Test *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416)

Fail-Kriterien:

- *Reply*-Button wird nicht aktiviert, wenn Nachricht in der Liste der empfangenen Nachrichten angeklickt wird
- nach Anklicken des *Reply*-Buttons erfolgt kein Wechsel ins *SelectRecipientsPanel*
- gleiche Fail-Kriterien wie für Test *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416)

---

1. Testlauf

Datum: 31.08.2005

Stand der Software: Die Implementierung aller für den Nachrichtenaustausch verantwortlichen Klassen ist abgeschlossen. Die Tests *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416) und *Nachrichten empfangen / anzeigen* (siehe 15.5.8.3 auf Seite 419) wurden fehlerfrei durchlaufen.

Tester: MO, JP

Setup: Das Setup entspricht dem des ersten Testlaufs des Tests *Nachrichten empfangen/anzeigen* (siehe 15.5.8.3 auf Seite 419)

Voraussetzungen: Die Tests *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416) und *Nachrichten empfangen / anzeigen* (siehe 15.5.8.3 auf Seite 419) wurden fehlerfrei durchlaufen. Der Server ist gestartet und das Szenario *offis.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und *Client 2* dem Team *Infiltrator* angehören.

Testablauf:

1. gleicher Ablauf, wie im ersten Testlauf des Tests *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416)
2. Client 3: Wechsel in MailPanel/Inbox
3. Client 3: Liste empfangener Nachrichten muss Eintrag *[Client 1] Test* enthalten
4. Client 3: diesen Eintrag anklicken
5. Client 3: *Reply*-Button anklicken
6. Client 3: Ansicht wechselt zu MailPanel/Compose
7. Client 3: *Subject* enthält Eintrag *Re: Test*
8. Client 3: Nachrichtentextfeld enthält Eintrag *This is a test*
9. Client 3: *Send*-Button ist aktiv
10. Client 3: Wechsel in MailPanel/SelectRecipientsPanel
11. Client 3: Liste der Empfänger muss Eintrag *Client 1* enthalten
12. Client 3: Wechsel in MailPanel/ComposePanel
13. Client 3: *Send*-Button klicken
14. Client 1: Wechsel in MailPanel/Inbox
15. Client 1: Liste empfangener Nachrichten muss Eintrag *[Client 1] Test* enthalten
16. Client 1: diesen Eintrag anklicken
17. Client 1: Nachrichtentext *This is a test* muss im Nachrichtentextfeld angezeigt werden

Auswertung: Der Test verlief erfolgreich. Durch Drücken des *Reply*-Buttons wird der Absender der Nachricht automatisch in die Empfängerliste eingetragen, so dass diesem eine Antwort geschickt werden kann.

---

2. Testlauf

Datum: 19.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Nachrichten beantworten“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden, wie für den ersten Testlauf, ein Server und drei Clients benötigt. Client 1 läuft auf dem Pocket PC 2003 Emulator von Microsoft, auf dem gleichen Rechner wie der Server. Client 2 und Client 3 laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden. Auch in diesem Test müssen die Clients, zwischen denen Nachrichten ausgetauscht werden sollen, dem gleichen Team zugehörig sein.

Konfiguration PC (Server und Client 1)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PDA 1 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 3)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

Voraussetzungen: Der Server ist gestartet und das Szenario *lerigauweg.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und *Client 2* dem Team Infiltrator angehören.

Testablauf: siehe 1. Testlauf

Auswertung: Der Test verlief ohne Probleme. Empfangene Nachrichten konnten beantwortet werden.

15.5.8.6 Nachrichten weiterleiten

Die Funktion *Nachrichten weiterleiten* dient dazu, den Inhalt empfangener Nachrichten, mit möglichst geringem Aufwand an beliebige Mitspieler weiterzuleiten. Im folgenden soll die korrekte Funktionsweise getestet werden.

Verantwortliche(r): JP

Pass-Kriterien:

- weitergeleitete Nachricht kommt korrekt bei Empfänger(n) an
- gleiche Pass-Kriterien wie für Test *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416)
- *Forward*-Button wird aktiviert, wenn Nachricht in der Liste der empfangenen Nachrichten angeklickt wird
- nach Anklicken des *Forward*-Buttons erfolgt ein Wechsel ins *SelectRecipientsPanel*

Fail-Kriterien:

- *Forward*-Button wird nicht aktiviert, wenn Nachricht in der Liste der empfangenen Nachrichten angeklickt wird
- nach Anklicken des *Forward*-Buttons erfolgt kein Wechsel ins *SelectRecipientsPanel*
- gleiche Fail-Kriterien wie für Test *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416)

---

1. Testlauf

Datum: 31.08.2005

Stand der Software: Die Implementierung aller für den Nachrichtenaustausch verantwortlichen Klassen ist abgeschlossen. Die Tests *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416) und *Nachrichten empfangen / anzeigen* (siehe 15.5.8.3 auf Seite 419) wurden fehlerfrei durchlaufen.

Tester: MO, JP

Setup: Das Setup entspricht dem des ersten Testlaufs des Tests *Nachrichten empfangen/anzeigen* (siehe 15.5.8.3 auf Seite 419)

Voraussetzungen: Die Tests *Nachrichten verfassen / versenden* (siehe 15.5.8.2 auf Seite 416) und *Nachrichten empfangen / anzeigen* (siehe 15.5.8.3 auf Seite 419) wurden fehlerfrei durchlaufen. Der Server ist gestartet und das Szenario *offis.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und *Client 2* dem Team *Infiltrator* angehören.

Testablauf:

1. Client 1: Wechsel in MailPanel/Inbox
2. Client 1: Eintrag *[Server] Willkommen Client 1* in der Liste der empfangenen Nachrichten auswählen
3. Client 1: *Forward*-Button anklicken (vorher nicht möglich, da inaktiv)
4. Client 1: MailPanel beobachten: Ansicht wechselt in *SelectRepipientsPanel*
5. Client 1: Eintrag *Client 3* in ComboBox auswählen und durch Klicken auf *Add*-Button bestätigen
6. Client 1: Wechsel in MailPanel/Compose
7. Client 1: Klicken auf *Send*-Button
8. Client 3: auf Farbänderung des MailTabs achten. Empfang einer neuen Nachricht muss signalisiert werden
9. Client 3: TickerPanel beobachten: Eintrag *You've got mail!* muss angezeigt werden
10. Client 3: Wechsel in MailPanel/Inbox
11. Client 3: Liste empfangener Nachrichten muss Eintrag *[Client 1] Willkommen Client 1* enthalten
12. Client 3: diesen Eintrag anklicken
13. Client 3: Nachrichtentext *Willkommen Client 1* muss im Nachrichtentextfeld angezeigt werden

Auswertung: Der Test verlief erfolgreich. Das MailTab von *Client 3* änderte die Farbe und das Ticker-Panel zeigte den Eintrag *You've got mail!* an. Die Liste der empfangenen Nachrichten enthielt den Eintrag *[Client 1] Willkommen Client 1*. Durch Anklicken dieses Eintrages wurde der Text *Willkommen Client 1* im Nachrichtentextfeld angezeigt.

---

2. Testlauf

Datum: 19.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt. Sie befindet sich in der Phase des Testens und des Behebens von Fehlern. Alle für die Ausführung der Funktion „Nachrichten weiterleiten“ relevanten Programmteile sind vollständig implementiert.

Tester: JP

Setup: Für den Test werden, wie für den ersten Testlauf, ein Server und drei Clients benötigt. Client 1 läuft auf dem Pocket PC 2003 Emulator von Microsoft, auf dem gleichen Rechner wie der Server. Client 2 und Client 3 laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden. Auch in diesem Test müssen die Clients, zwischen denen Nachrichten ausgetauscht werden sollen, dem gleichen Team zugehörig sein.

Konfiguration PC (Server und Client 1)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Konfiguration PDA 1 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 3)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

Voraussetzungen: Der Server ist gestartet und das Szenario *lerigauweg.xml* geladen. Alle Clients sind auf dem Server angemeldet (unter den Namen *Client 1*, *Client 2* bzw. *Client 3*), wobei *Client 1* und *Client 3* dem Team *Defender* und *Client 2* dem Team *Infiltrator* angehören.

Testablauf: siehe 1. Testlauf

Auswertung: Der Test verlief erfolgreich. Das MailTab von *Client 3* änderte die Farbe und das Ticker-Panel zeigte den Eintrag *You've got mail!* an. Die Liste der empfangenen Nachrichten enthielt den Eintrag *[Client 1] Willkommen Client 1.* Durch Anklicken dieses Eintrages wurde der Text *Willkommen Client 1* im Nachrichtentextfeld angezeigt.

15.5.9 Testfälle: Ziel des Spiels

15.5.9.1 Missionsziele anzeigen/lösen

In diesem Test wird das Anzeigen und Lösen von Missionszielen geprüft. Nur der Commander kann ein Missionziel sehen und lösen. (Siehe Test „Sehen von Missionszielen“, 15.5.3.6) Gibt er den richtigen Code ein, ist das Missionsziel gelöst.

Verantwortliche(r): SK

Pass-Kriterien:

- der Commander hat die Möglichkeit, in der Nähe des Missionsziels die Lösung im Menu-Panel einzugeben
- bei Eingabe der richtigen Lösung wird er über den Erfolg informiert

Fail-Kriterien:

- das Missionsziel wird nicht angezeigt, wenn der Commander in der Nähe ist
- die richtige Lösung führt nicht dazu, dass das Missionsziel gelöst ist

---

1. Testlauf

Datum: 11.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: SK

Setup:

- PC mit Windows 2000 Betriebssystem als Server
- Microsoft Visual Studio .NET 2003 (MSVS)
- Microsoft Pocket PC 2003 Emulator
- HP iPAQ 5450 mit Internetverbindung über ActiveSync

Voraussetzungen: Der Commander nimmt an dem Spiel teil und befindet sich im Spielzustand aktiv.

Testablauf:

1. Der Commander bewegt sich in die Nähe des Missionsziels.
2. Wechseln in die Ansicht „Menu“.
3. Eingabe der richtigen Lösung.
4. Klicken auf „solve“-Button.
5. In der Ansicht „Mail“ die Nachricht über das erfolgreiche Lösen des Missionsziels ansehen.

Auswertung: Das Missionsziel kann vom Commander mit der richtigen Lösung gelöst werden. Er erhält eine Nachticht über den Erfolg. Der Test verlief erfolgreich.

15.5.9.2 Spielende

In diesem Test wird das Spielende überprüft. Dazu gibt der Commander die richtige Lösung für das Missionsziel ein. Er erhält eine Nachricht vom Server über das Ende des Spiels. Alle anderen Spieler werden ebenfalls über das Spielende informiert und darüber, ob sie gewonnen oder verloren haben. Der Zustand aller Spieler wechselt zu „finished“.

Verantwortliche(r): SK

Pass-Kriterien:

- der Zustand aller Spieler wechselt zu „finished“
- alle Spieler erhalten eine Nachricht ob sie gewonnen oder verloren haben

Fail-Kriterien:

- der Zustandswechsel bleibt aus
- einer oder mehrere Spieler erhalten keine Nachricht

---

1. Testlauf

Datum: 11.09.2005

Stand der Software: Die Implementierung der Funktionen ist abgeschlossen, die Software befindet sich in der Phase der Integrationstests und der Qualitätssicherung.

Tester: SK

Setup:

- PC mit Windows 2000 Betriebssystem als Server
- Microsoft Visual Studio .NET 2003 (MSVS)
- Microsoft Pocket PC 2003 Emulator
- HP iPAQ 5450 mit Internetverbindung über ActiveSync

Voraussetzungen: Der Commander nimmt an dem Spiel teil und befindet sich im Spielzustand aktiv. Er hat die richtige Lösung für das Missionsziel eingegeben. (Siehe: Test „Missionsziele anzeigen/lösen“, 15.5.9.1.) Ein gegnerischer Spieler befindet sich im Spiel.

Testablauf:

1. Der Commander gibt die richtige Lösung für das Missionsziel ein.
2. Der Zustand beider Spieler wird auf dem Server als „finished“ angezeigt.
3. Der Zustand beider Clients wird als „finished“ dargestellt.
4. Der Commander hat eine Nachricht über den Sieg der Infilitratoren bekommen.
5. Der andere Mitspieler hat eine Nachricht über die Niederlage der Defender bekommen.

Auswertung: Die Eingabe der richtigen Lösung hat das Spiel für beide Spieler beendet. Der Test war erfolgreich.

15.5.10 Testfälle: Weitere optionale Anwendungsfälle

15.5.10.1 Konfigurationsdatei

Die Konfigurationsdatei speichert verschiedene Parameter mit denen NABB zur Startzeit initialisiert wird. Im Einzelnen sind das der vollständige Pfad zu einer Szenariodatei, die Punkte auf die Eigenschaften, der Spielernamen, der Comm-Port für die GPS-Maus, die Verwendung der GPS-Maus, die Verwendung von Vibration, die Server-IP und der Server-Port und ob sich zum Server verbunden werden soll. Die Speicherung dieser Informationen erleichtert dem Benutzer oder Tester den Umgang mit NABB, da er ein einmal eingegebenes Profil wieder verwenden kann oder nur minimale Änderungen vornehmen muss. Die Logik zum Einlesen und Speichern der Konfigurationsdatei befindet sich in der Klasse `SetupData`.

Verantwortliche(r): SA

Pass-Kriterien:

- Alle Einstellungen aus der vorliegenden Konfigurationsdatei werden übernommen. Existiert noch keine Konfigurationsdatei, werden Standardwerte verwendet.
- Alle Einstellungen, die in die Konfigurationsdatei geschrieben werden sollen, werden bei einem regulären Spielende (kein Abbruch des Programm durch Entfernen aus dem Speicher), in der Konfigurationsdatei gesichert. Wenn diese nicht vorhanden ist, wird sie angelegt.
- Der Lademechanismus zeigt sich robust gegen fehlende Einträge und belegt diese mit Standardwerten, ebenso ist die Reihenfolge der Einträge nicht erheblich für das korrekte Einlesen der Konfigurationsdatei. Ein gewisser Schutz muss gegen ungültige IPs und Ports gegeben sein.

Fail-Kriterien:

- Einstellungen werden aus der Konfigurationsdatei nicht (vollständig) übernommen
- Einträge werden nicht mit Standardwerten initialisiert, wenn sie in der Konfigurationsdatei fehlen.
- die Einstellungen werden bei einem regulären Beenden von NABB nicht vollständig in der Konfigurationsdatei gesichert
- es können ungültige Ports und IPs angegeben werden

---

1. Testlauf

Datum: 16.09.2005

Stand der Software: Der Software werden keine neuen Eigenschaften mehr hinzugefügt, sie befindet sich in der Phase des Testens und des Behebens von Fehlern.

Tester: SA

Setup: Für den Test werden ein Server und ein Client benötigt. In diesem Testlauf wird der Client über den Pocket PC 2003 Emulator von Microsoft gestartet.

Konfiguration PC (Server und Client)

- Athlon XP 1800+, 0,5 GB Ram, Geforce 4 mit 0,064 GB Ram
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003
- Microsoft Pocket PC 2003 Emulator

Voraussetzungen: Der Server wurde gestartet und ein Szenario ausgewählt, zu dem sich der Client verbinden soll. Das Anmelden bei dem Server muss funktionieren.

Testablauf:

- Abschnitt 1: Anlegen einer Konfigurationsdatei mit Standardwerten.
  1. Starten einer neuen Emulatorinstanz. Es existiert keine Konfigurationsdatei.
  2. NABB starten. Überprüfen, ob das `SetupPanel` mit Standardwerten gefüllt ist. Zu finden sind diese in der Klasse `SetupData::InitDefaults`.
  3. Überprüfen, ob eine Datei mit dem Namen *NabbConfig.txt* im Verzeichnis von NABB angelegt ist und in dieser die Standardwerte gespeichert sind.
  4. Beenden von NABB.
- Abschnitt 2: Verändern der Einstellungen in der Konfigurationsdatei.
  1. Starten des Servers.
  2. Starten einer neuen Emulatorinstanz mit NABB.
  3. Im `SetupPanel` die IP des Servers eintragen, das korrekte Szenario und den Namen ändern. Die Eigenschaftspunkte nach Belieben verändern.
  4. Erfolgreich beim Server anmelden.
  5. NABB beenden.
  6. Überprüfen, ob in der Konfigurationsdatei die vorgenommen Einstellungen übernommen sind.
- Abschnitt 3: Robustheit des Einlesemechanismus.
  1. In der vorhandenen Konfigurationsdatei eine ungültige IP (zu lang und/oder mit Bytes größer 255) und einen ungültigen Port eintragen (größer 65536 oder kleiner 1024).
  2. Verschiedene Einträge löschen („Scenario," „Player"), damit die Reihenfolge nicht erhalten bleibt.
  3. NABB starten.
  4. Überprüfen, ob die Standard-IP und der Standard-Port im `SetupPanel` gesetzt sind und nicht die falschen Einträge. Ebenso überüfen, ob die verbliebenen, nicht gelöschten Einträge (z.B. „Neutralize", „Recon", ...) trotz der veränderten Reihenfolge korrekt übernommen sind.
  5. NABB beenden.
  6. Überprüfen, ob eine vollständige Konfigurationsdatei vorliegt, in der die vormals falschen und gelöschten Einträge durch die Standardwerte ersetzt sind.

Auswertung: In keinem Punkt des Tests konnte ein *Fail*-Kriterium erreicht werden. Alle *Pass*-Kriterien sind erreicht worden. Der Test gilt als bestanden. Abschließend ist noch zu erwähnen, dass nicht jede möglich Kombination von fehlenden Werten oder ungültigen Werten getestet worden ist. Dieses würde den Aufwand nicht rechtfertigen. Es kann aber angenommen werden, dass die getesteten Fälle die ungetesteten abdecken.

15.6 Abschlusstests

In diesem Abschnitt werden die Abschlusstests dokumentiert. Innerhalb eines Abschlusstests werden komplette Spielabläufe durchlaufen. Die Spielabläufe setzen sich dabei aus einer Reihe von Integrationstests zusammen. Das Ziel der Durchführung von Abschlusstests besteht zum einen darin, mögliches Fehlverhalten bei längerer Spieldauer zu identifizieren, sowie die *Spielbarkeit* des Agentenspiels generell zu bewerten.

15.6.1 1. Testlauf

Ziel dieses Testlaufs ist es, die grundlegenden Funktionen von NABB, wie Tarnen (*Cloak*), Neutralisieren (*Neutralize*), Aufklären (*Recon*) und Abhören (*WireTab*) zu testen. Zu diesem Zweck soll ein komplettes Spielszenario durchgespielt werden, bei welchem alle Grundfunktionen von NABB mindestens einmal durchlaufen werden. Zu den Grundfunktionen zählen neben den eingangs genannten Funktionen auch das Anmelden eines Clients beim Server, das Aufnehmen, Ablegen und Aktivieren von Goodies, sowie das Lösen des Missionsziels.
Im folgenden Abschnitt wird der Ablauf dieses ersten Abschlusstests am OFFIS[26], Oldenburg protokolliert. Neben Angaben zur verwendeten Hard- und Software, enthält dieses Testprotokoll eine detaillierte Beschreibung des Testablaufs, sowie ein Fazit des Testlaufs.

Datum: 08.09.05

Ort: Escherweg, Oldenburg

Setup:

Für den Test werden ein Server und zwei Clients benötigt. Der Server wird auf einem Windows PC gestartet. Die Clients laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden.

Konfiguration PC (Server)

- AMD Athlon (908 MHz), 512 MB RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003

Konfiguration PDA 1 (Client 1)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

[26] Oldenburger Forschungs- und Entwicklungsinstitut für Informatikwerkzeuge und -systeme

Konfiguration PDA 2 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

NABB-Testlauf am Escherweg		
Zeit-punkt	**Spieler:** Client 1 **Team:** Defender **Rolle:** Defender	**Spieler:** Client 2 **Team:** Infiltrator **Rolle:** Commander
1.	PDA, GPS Maus und Handy per Bluetooth miteinander verbinden	PDA, GPS Maus und Handy per Bluetooth miteinander verbinden
2.	Programm NABB starten	Programm NABB starten
3.	Szenario auswählen: *offis.xml*	Szenario auswählen: *offis.xml*
4.	IP-Adresse NABB-Server eintragen: *134.106.52.219*	IP-Adresse NABB-Server eintragen: *134.106.52.219*
5.	Spielernamen eintragen: *Client 1*	Spielernamen eintragen: *Client 2*
6.	Verwendung von GPS bestätigen	Verwendung von GPS bestätigen
7.	Spiel starten: OK-Button klicken	Spiel starten: OK-Button klicken
8.	Überblick verschaffen: soweit wie möglich aus der Karte zoomen	Überblick verschaffen: soweit wie möglich aus der Karte zoomen
9.	zum Startpunkt (linke untere Ecke der Karte) begeben, um aktiviert zu werden	zum Startpunkt (rechte Seite der Karte, oberhalb der Einfahrt zum OFFIS-Parkplatz) begeben um aktiviert zu werden
10.	zur rechten unteren Ecke der Karte begeben, Goodie finden und aufnehmen	zur linken oberen Ecke der Karte begeben, Goodie finden und aufnehmen
11.	zum Haupteingang des OFFIS gehen und Kamera setzen und dies dem Infiltrator mitteilen (heranwinken)	auf Signal des Defenders warten, dann in Richtung Haupteingang des OFFIS begeben
12.	wenn Infiltrator in Sichtweite der Kamera, diesem dies signalisieren (Hand heben)	wenn Defender Signal zum Anhalten gibt, stehen bleiben und Tarnen
13.	Karte beobachten, Infiltrator muss von der Karte verschwinden (hat sich getarnt), dies dem Infiltrator mitteilen (heranwinken)	auf Signal des Defenders warten, dann zur Position des Defenders begeben
14.	Kamera wieder aufnehmen	sobald Defender auf Karte sichtbar, diesen neutralisieren
15.	wenn neutralisiert, wieder zum Startpunkt (linke untere Ecke der Karte) begeben und aktiviert werden	Goodie an Position des Defenders suchen und aufnehmen, Kamera setzen
16.	zum Haupteingang des OFFIS begeben	zum Missionsziel begeben (oberhalb des kleinen Hörsaals des OFFIS), sobald dieser erreicht wurde, eine Kamera setzen
17.	Kamera des Infiltrators finden und zerstören	in „Menu“-Ansicht wechseln und beide Goodies aktivieren
18.	zum Missionsziel begeben (oberhalb des kleinen Hörsaals des OFFIS)	in „Mail“-Ansicht wechseln und Nachrichten mit Lösungshinweisen lesen
19.	sobald Infiltrator in Sichtradius, diesen Neutralisieren	in „Menu“-Ansicht wechseln und versuchen das Missionsziel zu lösen. Falls dabei ein Defender entdeckt wird, diesen neutralisieren

Tabelle 15.1: NABB-Testlauf am Escherweg

Fazit:

Der Testlauf konnte nicht komplett durchlaufen werden. Das Gelände rund um das OFFIS erwies sich als nicht geeignet für die Durchführung des Tests. Die Positionsbestimmung der einzelnen Clients funktionierte zwischen dem OFFIS und den angrenzenden Gebäuden nur unzureichend, was auf eventuelle Verfälschungen der GPS-Daten innerhalb der Häuserschluchten zurückgeführt wurde. Darüber hinaus riss die Verbindung zwischen *PDA 2* und dem Siemens S55 mehrmals ab. Im Verlaufe des Testlaufs wurde festgestellt, das die GPS-Mäuse einige Minuten brauchen, bevor sie sinnvolle GPS-Daten für die Clients auf den PDAs liefern.
Das Testszenario konnte, aufgrund der soeben beschriebenen Schwierigkeiten nur teilweise durchlaufen werden. So konnte sich nur *Client 2* beim Server anmelden. *Client 1* musste über den Server aktiviert werden. Da eine weitere Durchführung des Testlaufs ohne eine funktionsfähige Positionsbestimmung unmöglich war, wurde entschieden, den Testlauf abzubrechen und zunächst ein geeigneteres Testgelände zu suchen.

15.6.2 2. Testlauf

Für den zweiten Testlauf entschied sich die Projektgruppe für ein neues Testgelände (Lerigauweg, Oldenburg). Der Testablauf des ersten Testlaufs wurde größtenteils beibehalten und lediglich den lokalen Gegebenheiten des neuen Testgeländes angepasst.
Im folgenden Abschnitt wird der Ablauf dieses zweiten Abschlusstests protokolliert. Neben Angaben zur verwendeten Hard- und Software, enthält dieses Testprotokoll eine detaillierte Beschreibung des Testablaufs, sowie ein Fazit des Testlaufs.

Datum: 19.09.05

Ort: Lerigauweg, Oldenburg

Setup:

Für den Test werden ein Server und zwei Clients benötigt. Der Server wird auf einem Windows PC gestartet. Die Clients laufen jeweils auf einem PDA. Die GPS-Signale empfängt jeder Client über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über ein GPRS-fähiges Handy hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden.

Konfiguration PC (Server)

- AMD Athlon XP 2000+, 512 MB DDR RAM
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003

Konfiguration PDA 1 (Client 1)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

NABB-Testlauf am Lerigauweg		
Zeit-punkt	**Spieler:** Client 1 **Team:** Defender **Rolle:** Defender	**Spieler:** Client 2 **Team:** Infiltrator **Rolle:** Commander
1.	PDA, GPS Maus und Handy per Bluetooth miteinander verbinden	PDA, GPS Maus und Handy per Bluetooth miteinander verbinden
2.	Programm NABB starten	Programm NABB starten
3.	Szenario auswählen: *lerigauweg.xml*	Szenario auswählen: *lerigauwegs.xml*
4.	IP-Adresse NABB-Server eintragen: 217.186.162.231	IP-Adresse NABB-Server eintragen: 217.186.162.231
5.	Spielernamen eintragen: *Client 1*	Spielernamen eintragen: *Client 2*
6.	Verwendung von GPS bestätigen	Verwendung von GPS bestätigen
7.	Spiel starten: OK-Button klicken	Spiel starten: OK-Button klicken
8.	Überblick verschaffen: soweit wie möglich aus der Karte zoomen	Überblick verschaffen:soweit wie möglich aus der Karte zoomen
9.	zum Startpunkt (Kreuzung Kaspersweg/-Lerigauweg) begeben, um aktiviert zu werden	zum Startpunkt (Kreuzung Lerigauweg/Margarete-Gramberg Straße) begeben, um aktiviert zu werden
10.	Lerigauweg in Richtung Edewechter Landstraße bis zum Radweg (rechte Seite) folgen; dem Verlauf des Radwegs bis zum Ende des Parkplatzes auf der linken Seite folgen; Goodie finden und aufnehmen	Margarete-Gramberg Straße bis zum Ende folgen;Goodie finden und aufnehmen
11.	zurück zum Lerigauweg begeben;diesen in Richtung Margarete-Gramberg Straße folgen; Goodie finden und aufnehmen; Kamera setzen;Infiltrator heranwinken;	zurück zum Startpunkt begeben;auf Signal des Defenders warten;dann in Richtung des Defenders bewegen
12.	wenn Infiltrator in Sichtweite der Kamera, diesem dies signalisieren (Hand heben)	wenn Defender Signal zum Anhalten gibt, stehen bleiben und Tarnen
13.	Karte beobachten, Infiltrator muss von der Karte verschwinden (hat sich getarnt), dies dem Infiltrator mitteilen (heranwinken)	auf Signal des Defenders warten, dann zur Position des Defenders begeben
14.	Kamera wieder aufnehmen	sobald Position des Defenders erreicht, diesen neutralisieren
15.	wenn neutralisiert, wieder zum Startpunkt (Kreuzung Kaspersweg/Lerigauweg) begeben, um aktiviert zu werden	ein Goodie an Position des Defenders suchen und aufnehmen, Kamera setzen
16.	zurück zum Standort, an dem der Defender neutralisiert wurde	zum Missionsziel begeben (Stichstraße in Richtung Gehörlosenschule folgen);sobald dieser erreicht wurde, eine Kamera setzen
17.	Kamera des Infiltrators finden und zerstören; Goodie finden und aufnehmen	in „Menu“-Ansicht wechseln und beide Goodies aktivieren
18.	zum Missionsziel begeben (Stichstraße in Richtung Gehörlosenschule folgen)	in „Mail“-Ansicht wechseln und Nachrichten mit Lösungshinweisen lesen
19.	sobald Infiltrator in Sichtradius, diesen neutralisieren	in „Menu“-Ansicht wechseln und versuchen das Missionsziel zu lösen; falls dabei ein Defender entdeckt wird, diesen neutralisieren

Tabelle 15.2: NABB-Testlauf am Lerigauweg

Fazit:

Der Testlauf verlief erfolgreich. Es konnten sämtliche Punkte des Testablaufs durchlaufen werden. Die Anmeldung der Clients beim Server verlief ohne Probleme. Eine Positionsbestimmung auf den Clients, mittels der durch die GPS-Mäuse gelieferten Werte, war stets möglich. Lediglich der zweimalige Verlust des GPS-Signals durch die von *PDA 1* verwendeten GPS-Maus fiel negativ auf. Neben dem Anmelden der Clients beim Server, verlief auch der Test der Funktionen Tarnen (*Cloak*), Neutralisieren (*Neutralize*), Aufklären (*Recon*), Abhören (*WireTab*), Aufnehmen, Ablegen und Aktivieren von Goodies, sowie das Lösen des Missionsziels erfolgreich.

15.6.3 3. Testlauf

Der dritte und abschließende Testlauf wurde an der C.v.O. Universität Oldenburg (Standort Haarentor, Uhlhornsweg) durchgeführt, da dort auch die Abschlusspräsentation stattfinden sollte. Der Testablauf des ersten und zweiten Testlaufs wurde für diesen Testlauf geringfügig erweitert. Zusätzlich zu den beiden bisher eingesetzten mobilen Clients wurde ein dritter emulierter Client eingesetzt. Durch den Einsatz dieses dritten Clients sollte u.a. der Austausch und das Abfangen von Nachrichten getestet werden.
Im folgenden Abschnitt wird der Ablauf dieses dritten Abschlusstests protokolliert. Neben Angaben zur verwendeten Hard- und Software, enthält dieses Testprotokoll eine detaillierte Beschreibung des Testablaufs, sowie ein Fazit des Testlaufs.

Datum: 12.10.05

Ort: Uhlhornsweg, Oldenburg

Setup:

Für den Test werden ein Server und drei Clients benötigt. Der Server und ein Client werden auf einem Windows PC gestartet. Die beiden anderen Clients laufen jeweils auf einem PDA. Die GPS-Signale empfangen diese Clients über eine GPS-Maus. Eine Verbindung zum Server erfolgt über eine GPRS-Verbindung, die über GPRS-fähige Handys hergestellt wird. Sowohl die GPS-Mäuse als auch die GPRS-fähigen Handys sind über Bluetooth mit den PDAs verbunden.

Konfiguration PC (Server und Client 3)

- Intel Pentium M (1,6 GHz), 512 MB DDR-Ram
- Microsoft Windows XP Professional Version 2002 SP 2
- Microsoft Visual Studio .NET 2003

Konfiguration PDA 1 (Client 1)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Sony Ericsson K700i

Konfiguration PDA 2 (Client 2)

- Hewlett-Packard iPAQ 5450
- Falcom NAVI-1
- Siemens S55

NABB-Testlauf am Uhlhornsweg			
Zeitpunkt	**Spieler:** Client 1 **Team:** Infiltrator **Rolle:** Commander	**Spieler:** Client 2 **Team:** Defender **Rolle:** Defender	**Spieler:** Client 3 **Team:** Infiltrator **Rolle:** Infiltrator
1.	PDA, GPS Maus und Handy per Bluetooth miteinander verbinden	PDA, GPS Maus und Handy per Bluetooth miteinander verbinden	
2.	Programm NABB starten	Programm NABB starten	Programm NABB starten
3.	Szenario auswählen: *campus.xml*	Szenario auswählen: *campus.xml*	Szenario auswählen: *campus.xml*
4.	IP-Adresse NABB-Server eintragen:	IP-Adresse NABB-Server eintragen:	IP-Adresse NABB-Server eintragen:
5.	Spielernamen eintragen:	Spielernamen eintragen:	Spielernamen eintragen:
6.	Verwendung von GPS aktivieren	Verwendung von GPS aktivieren	Verwendung von GPS deaktivieren
7.		Fähigkeit Abhören auf Wert 4 setzen	
8.	Spiel starten: OK-Button klicken	Spiel starten: OK-Button klicken	Spiel starten: OK-Button klicken
9.	Überblick verschaffen: soweit wie möglich aus der Karte zoomen	Überblick verschaffen: soweit wie möglich aus der Karte zoomen	Überblick verschaffen: soweit wie möglich aus der Karte zoomen
10.	zum Startpunkt (Parkplatz gegenüber der Brücke zwischen A5 und AVZ) begeben, um aktiviert zu werden	zum Startpunkt (Fußgängerüberweg zwischen A14 und Brücke / Seite AVZ) begeben, um aktiviert zu werden	zum Startpunkt (Parkplatz gegenüber der Brücke zwischen A5 und AVZ) begeben, um aktiviert zu werden
11.		Abhörmodus aktivieren	
12.			zur Grünfläche zwischen A7 und A8 begeben, Goodie finden und aufnehmen
13.			Nachricht an Commander senden, dass Goodie gefunden wurde
14.	Nachricht vom Infiltrator wird empfangen	Nachricht vom Infiltrator wird abgehört	
15.	zum Fußgängerüberweg zwischen Einfahrt Parkplatz und Freilufthalle / Seite Parkplatz begeben, Goodie finden und aufnehmen	zur Grünfläche zwischen A7 und A8 begeben	Kamera setzen und auf Eintreffen des Defenders warten
16.		es werden ein Infiltrator und eine gegnerische Kamera im Sichtradius angezeigt	Defender wird zunächst im Sichtradius der Kamera und dann im eigenen Sichtradius angezeigt
17.		gegnerische Kamera suchen und zerstören	zerstörte Kamera ist wieder verfügbar
18.		zur Position des Infiltrators begeben, dann Tarnmodus aktivieren	Defender wird nicht mehr angezeigt (hat sich getarnt)

Tabelle 15.3: NABB-Testlauf am Uhlhornsweg (Teil 1)

NABB-Testlauf am Uhlhornsweg			
Zeitpunkt	**Spieler:** Client 1 **Team:** Infiltrator **Rolle:** Commander	**Spieler:** Client 2 **Team:** Defender **Rolle:** Defender	**Spieler:** Client 3 **Team:** Infiltrator **Rolle:** Infiltrator
19.		Infiltrator neutralisieren	wenn neutralisiert, überprüfen, ob Goodie wieder abgelegt wurde
20.		Goodie finden und aufnehmen	
21.	zum Standort des Missionsziels begeben (Grünfläche zwischen A5 und A13)	zum Standort des Missionsziels begeben (Grünfläche zwischen A5 und A13)	
22.	Kamera setzen und auf Eintreffen des Defenders warten		
23.	Defender wird zunächst im Sichtradius der Kamera und dann im eigenen Sichtradius angezeigt	es werden der Commander und eine gegnerische Kamera im Sichtradius angezeigt, zur Position des Commanders begeben	
24.	Kamera wieder aufnehmen und Tarnen aktivieren	Commander wird nicht mehr angezeigt (hat sich getarnt)	
25.	wenn Defender nah genug , diesen neutralisieren	wenn neutralisiert, überprüfen, ob Goodie wieder abgelegt wurde	
26.	Goodies aktivieren und Missionsziel lösen		

Tabelle 15.4: NABB-Testlauf am Uhlhornsweg (Teil 2)

Fazit:

Der Testlauf verlief erfolgreich. Es konnten sämtliche Punkte des Testablaufs durchlaufen werden. Die Anmeldung der Clients beim Server verlief ohne Probleme. Eine Positionsbestimmung auf den Clients, mittels der durch die GPS-Mäuse gelieferten Werte, war stets möglich. Lediglich der vom VPN-Client erzwungene Abbruch der WLAN-Verbindung nach ca. einer Stunde fiel negativ auf. Neben dem Anmelden der Clients beim Server, verlief auch der Test der Funktionen Tarnen (*Cloak*), Neutralisieren (*Neutralize*), Aufklären (*Recon*), Abhören (*WireTab*), Aufnehmen, Ablegen und Aktivieren von Goodies, sowie das Lösen des Missionsziels erfolgreich. Darüber hinaus konnten Nachrichten zwischen Client 1 und Client 2 problemlos ausgetauscht werden.

Kapitel 16

Zusammenfassung und Bewertung

Das folgende Kapitel bietet einen Rückblick über den Verlauf der Projektgruppe „eXplorer“. Nach einer kurz zusammenfassenden Wiedergabe sämtlicher Arbeitsschritte des vergangenen Jahres folgen Erfahrungsberichte, die Eindrücke aus der Bearbeitung einzelner Arbeitspakete reflektieren. Abschließend erfolgt eine Bewertung des Projekts sowie des resultierenden Softwareprodukts und ein Ausblick auf denkbare Erweiterungen.

16.1 Zusammenfassung

Als Einstieg in die für viele Teilnehmer noch eher unklare Thematik des Projekts wurde gleich beim ersten Gruppentreffen nach einer kurzen Vorstellung des Projekts und der Teilnehmer die erste Aufgabe gestellt. Die Grundlagen für eine *kontextsensitive Umgebungserkundung mit mobiler, multimodaler Unterstützung* sollten in einer Seminarphase erarbeitet werden, für die Vorträge und schriftliche Ausarbeitungen über folgende Themen angefertigt und an einem Wochenendseminar auf Norderney vorgestellt wurden: *Accessibility*, *Evaluation*, *Interaktion*, *Kontext*, *Mobile Anwendungen*, *Multimodalität*, *Personalisierung*, *Tangible Bits* und *Usability*.
Nach der Vorstellung der einzelnen Themen, musste geklärt werden, was für Produkte denkbar wären, die alle genannten Themenbereiche überspannen. Aus diesem Grunde wurden zunächst einige Szenarien entwickelt, von denen dann drei noch recht lange parallel verfolgt wurden, bis die Entscheidung letztendlich auf das Szenario mit dem Arbeitstitel *Agentenspiel* fiel. Mit voller Konzentration auf das gewählte Szenario wurde im nächsten Schritt daran gearbeitet, ein passendes Produkt zu skizzieren. Dabei sollten die Beschreibungen allerdings so untechnisch wie möglich bleiben.
Die Produktsskizze diente dann als Grundlage für die Anforderungsdefinition, in welcher Projektziel, Systemeinsatz und -umgebung, funktionale und nichtfunktionale Anforderungen sowie Anforderungen an eine Benutzungsschnittstelle und Dokumentation festgehalten wurden.
Ein detaillierteres Bild des zu entwickelnden Produkts sollte eine Fallstudie liefern. Sie sollte zentrale Technologien des bisher noch untechnischen Gedankenkonstrukts aufdecken und zu einer prototypischen Umsetzung der Anforderungsdefinition führen. Dank der hieraus gewonnenen Erkenntnisse konnten nun Technologiestudien folgen, in denen zunächst mögliche technische Lösungen zusammengetragen und deren Eignung anhand eines zuvor erstellten Kriterienkatalogs in Technikreferaten vorgestellt wurde. Darauf basierend erfolgte schließlich die Auswahl der zu verwendenden Technologien.
Um die anschließende Entwurfsphase möglichst erfolgreich gestalten zu können, wurde sie mit einer kurzen Vortragsreihe über Design Patterns eingeleitet. Nach der Identifizierung möglicher Programmzustände be-

gannen dann parallel Arbeiten an der Systemarchitektur und der Nutzungsschnittstelle. Im Anschluss an den Entwurf der Systemarchitektur konnte der Feinentwurf in Angriff genommen werden. Die letzte Entwurfsphase umfasste schließlich Aufgaben des Reengineerings. Hier wurden aufgrund von Erkenntnissen des parallel durchgeführten Qualitätsmanagements und teilweise auch der Implementierung Anpassungen am Entwurf vorgenommen.
Neben dem Entwurf begleitete das Qualitätsmanagement auch die Implementierung. Vorbereitend für die Implementierung und gleichzeitig Konzepte des Entwurfs überprüfend wurden als Teil des Qualitätsmanagements Machbarkeitsstudien zu Hard- und Software durchgeführt. Zum einen wurden hier die Einsetzbarkeit einzelner Technologien getestet und zum anderen als Hilfestellung für die Implementierung bereits Prototypen bezüglich deren Verwendung erstellt. Die Ergebnisse der Implementierung wurden in drei Phasen evaluiert: *Modultests*, *Integrationstests* und *Abschlusstests*.
In der Implementierungsphase wurden anfangs Coding Conventions festgelegt und die Priorisierung der einzelnen Module vorgenommen bevor dann die eigentliche Programmierarbeit in zwei Phasen begann. Zunächst wurde ein Softwaregerüst erstellt, welches im Anschluss nach und nach um Funktionalitäten einzelner Anwendungsfälle erweitert wurde. Das letzte Arbeitspaket Abschluss befasste sich dann mit der Abnahme der Implementierung, einer Projektanalyse, dem Abschlussbericht und der Projektpräsentation.
Außer den bereits genannten Aufgaben liefen noch einige über das ganze Jahr parallel, die im Projektplan im Arbeitspaket Projektkoordination zusammengefasst wurden. Hierzu gehören: Gruppentreffen, Projektmanagement, Sitzungsmoderation, Qualitätssicherung, Webauftritt, Dokumentation, Evaluation, Eventmanagement sowie Hilfestellung zu TeX und UML.

16.2 Erfahrungsberichte

In diesem Abschnitt werden die Erfahrungen erläutert, die die Gruppe während der Durchführung des Projektes gemacht hat.

16.2.1 Erfahrungen aus der Vorbereitungsphase

Die Vorbereitungsphase der Projektgruppe umschließt die Arbeitspakete A, B und C, dabei sind Grundlagenbildung, Ideenfindung und deren Ausformulierung die zentralen Punkte der Vorbereitungsphase.

Die Arbeit der Projektgruppe hat mit der Erstellung der Seminararbeiten und deren Präsentation auf Norderney begonnen. Die Kompaktheit des Norderneyaufenthalts hat der Projektgruppe gut gefallen, da die Seminarphase dadurch schnell abgeschlossen wurde und die Gruppe sich auf nachfolgende Tätigkeiten konzentrieren konnte. Die Gruppe bedauert, dass eine Korrektur der Seminararbeiten viel Zeit in Anspruch nahm oder gar nicht erfolgte. Dies ist einem Lerneffekt abträglich, außerdem kam die Gruppe in diesem Zusammenhang zu der Feststellung, dass die Ergebnisse der Seminararbeiten nicht optimal für spätere Arbeiten genutzt wurden. Hier wird Verbesserungspotential gesehen.

An der Erarbeitung der Szenarien war die ganze Projektgruppe beteiligt. Das wurde auf der einen Seite als positiv empfunden, weil somit alle Ergebnisse von der Gruppe diskutiert wurden und jeder seinen Ideen und Wünschen einfließen lassen konnte, auf das zügige Vorankommen der Projektarbeit wirkte sich dieser Sachverhalt allerdings negativ aus. Es wurde zu wenig parallel und nicht zielgerichtet genug gearbeitet. Der verstärkte Einsatz von Kleingruppen mit klar definierten Teilaufgaben hätte diesen Sacherverhalt vermieden. Die frühe Simulation des Agentenspiels als Papierversion war eine hilfreiche Erfahrung, allerdings hätten die Ergebnisse besserer verwertet werden können, d.h. im ersten Schritt ausformuliert

positiv	verbesserungswürdig
sehr kompakte Seminarphase	mehr Erkenntnisse aus Seminaren nutzen
kreative Phase	zielgerichteter arbeiten
gutes Papierspiel	mehr schriftlich festhalten
toller Flyer erarbeitet	Produktskizze weglassen

Tabelle 16.1: Erfahrungen aus der Vorbereitungsphase

werden müssen. Gefehlt hat der Projektgruppe auch eine Auswertung des aufgenommenen Videos. Im Allgemeinen hat die Projektgruppe sehr konkret und detailliert das Agentenspiel ausgearbeitet, allerdings ohne es ausführlich genug schriftlich festzuhalten. Das hätte in Form eines fachlichen Modells für das Agentenspiel realisiert werden können.

Die Produktskizze ist von der Projektgruppe als notwendiges Dokument für die Entwicklung von Software aufgefasst worden. Nach Auffassung der Projektgruppe fehlt es der Produktskizze an inhaltlicher Tiefe und konkreten Aussagen. Die Projektgruppe merkte an, dass es zu einem Auftrag, der nicht klar festegelegt ist, sinnvoller gewesen wäre auf die Produktskizze zu verzichten, oder sie allenfalls als ein internes Dokument zu verwenden und die Anforderungsdefinition als erstes offizielles Dokument anzusehen. Die zeitliche Anordnung vor der Technologiestudie wurde als ungünstig empfunden, da diese Erkenntnisse brachte, die auf die Entwicklung des Agentenspiels Einfluss genommen hat. Die Entwicklung des Flyers als Werbemaßnahme hat im Nachhinein enttäuscht, da dieser nicht eingesetzt wurde. Der Flyer an sich wurde aber als gelungen bewertet.

16.2.1.1 Fazit

Aus der Analyse der Vorbereitungsphase wird deutlich, dass die Projektgruppe noch nicht eingearbeitet war und sich schwer tat, aus der Themenvorgabe einen konkreten Arbeitsauftrag zu formen. Die Projektgruppe ist sich im Klaren, dass kreative Phasen leicht als unproduktiv angesehen werden. Trotzdem zeugen die gesammelten Ideen, wenn auch nicht perfekt ausformuliert, von großen Einfallsreichtum. Die Tabelle 16.1 auf Seite 451 fasst die wichtigsten Punkte der Vorbereitungsphase unteren positiven und verbesserungswürdigen Gesichtpunkten zusammen.

16.2.2 Erfahrungen aus der Erstellung der Anforderungsdefinition

Im allgemeinen wird eine vollständige Anforderungsdefinition für die Entwicklung von Software als unabdingbar angesehen. Im Verlauf der Projektgruppe stellte sich jedoch heraus, dass es sehr schwierig ist, bei Projekten, die im besonderem Masse kreative Ideenfindung fordern, früh eine konkrete Anforderungsdefinition zu formulieren. Im Projektplan der Projektgruppe war die Anforderungsdefinition deutlich zu früh angesiedelt. Bei nachträglicher Betrachtung hätte das Projekt wahrscheinlich von einem evolutionären Vorgehen profitiert, wobei eher die Rahmenbedingungen als Anforderungen festgehalten werden. Wenn dann der genaue Umfang der Software feststeht (und durch die Evulotionsschritte schon teilweise implementiert ist) kann eine konkrete und feinkörnige Anforderungsdefinition aufgestellt werden. Ein solches Vorgehen würde dem Umstand Rechnung tragen, dass eine Projektgruppe auch immer Forschung betreibt und nicht wie bei der klassischen Softwareentwicklung konkrete, mit einem Kunden verhandelbare Anforderungen im Vordergrund stehen.

16.2.2.1 Fazit

Die Anforderungsdefinition hätte in einem perfekten Projektverlauf später verfasst werden müssen. Durch den frühen Zeitpunkt wurde einiges an Zeit vertan, Anforderungen zu definieren, über dessen Stellenwert noch keine konkreten Aussagen gemacht werden konnten. Jedoch hat die Projektgruppe die in der Anforderungsdefinition aufgestellten Anwendungsfälle, die nichtfunktionalen Anforderungen und die Anforderungen an die Benutzungsschnittstelle mit Erfolg umsetzen können.

16.2.3 Erfahrungen aus der Erstellung der Fallstudie

Zu Beginn der Planung des Projekts wurden die Arbeiten zu dem Arbeitspaket 2 als eine Vorbereitungsphase für die Arbeiten zum Arbeitspaket 4, dem Entwurf, angesehen. Als Ergebnis sollte ein Grobkonzept erstellt werden, dass nach der Phase der Technologiestudien zu einem Entwurfsdokument verfeinert werden sollte.

Nach dem Abschluss der Arbeiten zu diesem Arbeitspaket haben wir jedoch festgestellt, dass das entstandene Dokument nicht die gewünschte Vorarbeit leistet, um durch eine Verfeinerung den Entwurf der Software zu erhalten. Statt dessen haben wir eher unbeabsichtigt die Erstellung einer Softwarearchitektur betrieben, die jedoch nicht ausreichend genug geplant werden konnte, da sie nicht als solche vorgesehen war. Letztendlich hätte dieser Teil des Dokuments als gut konzipierte und ausgearbeitete Softwarearchitektur ein Bestandteil des Entwurfs werden sollen. Ähnliches gilt für die Abschnitte zum Informationsdesign und Interaktionsdesign. Diese Teile des Dokuments hätte mehr auf das Projekt zugeschitten werden sollen. Eine konkretere Erarbeitung zur Gestaltung der Benutzerschnittstelle der zu erstellenden Software hätte dadurch ebenfalls Bestandteil des Entwurfs werden können. Aufgrund der unausgereiften Bearbeitung wurde auch der Titel des Dokuments von „Grobkonzept“ auf „Fallstudie“ geändert.

Generell werden die Arbeiten zu diesem Arbeitspaket als sehr wichtig erachtet. Ebenfalls die zeitliche Einordnung in die Arbeiten des gesammten Projekts wurde als richtig eingestuft. Die Erarbeitung dieses Dokuments wurde zum Zeitpunkt der Erstellung jedoch noch als Abarbeitung eines lästigen Pflichtdokuments angesehen. Dies spiegelt sich auch in der Art und Weise wieder, wie die Arbeiten zu diesem Arbeitspaket teilsweise durchgeführt wurden. Die Beschreibunge zu den einzelnen Modulen der Architektur des Agentenspiels und des Frameworks wurden vom Inhalt und vom Umfang her nach einem bestimmten Schema abgearbeitet. Die Aussagekraft wurde dadurch deutlich herabgesetzt.

Rückblickend ist die Gruppe einheitlich der Meinung, dass die Arbeiten zu diesem Arbeitspaket anders hätten angegangen werden sollen, um die Erstellung eines Pflichtdokuments zu vermeiden. Es wurde vorgeschlagen, bei einer erneuten Erarbeitung der Architektur bei der Beschreibung der Module sich mehr auf die Schnittstellen zwischen diesen als auf die Aufgaben und Funktionalitäten zu konzentrieren.

Eine Arbeit in Kleingruppen ähnlich dem Pair-Programming während der Implementierungsphase wurde als weitere Verbesserung angeregt. Module könnten zur Bearbeitung aufgeteilt und zur Ausarbeitung und Verfeinerung auf die Gruppen verteilt werden. Dies hätte zum einen eine höhere Qualität der Arbeiten als auch eine stärkere Verantwortung für einzelne Teile der Architektur zur Folge.

16.2.3.1 Fazit

Rückblickend lässt sich festhalten, dass die Arbeiten zum Arbeitspaket 2 generell als wichtig angesehen werden, die Ausführung jedoch nicht optimal gelaufen ist. Das Ziel dieser Phase hätte gerade im Hinblick auf die Abschnitte „Interaktionsdesign“ und „Informationsdesign“ einen stärkeren Bezug auf die zu

entwickelnde Software haben sollen. Das Grobkonzept als Ergebnisdokument hätte ein Bestandteil des Entwurfs werden können.

16.2.4 Erfahrungen mit den Technologiestudien

Die Technologiestudien wurden nach der Erstellung des Grobkonzeptes angefertigt. Innerhalb der Technologiestudien wurden relevante Technologien, die der Realisierung des Agentenspiels dienen sollten, gesichtet und anhand eines Kriterienkatalogs bewertet. Die Technologienstudien endeten mit einer Ergebnispräsentation vor der gesamten Projektgruppe. Es war die einhellige Meinung der Gruppe, dass die Erstellung der Technologiestudien, wenn auch zur Findung relevanter Technologien sinnvoll, im Verlauf der Projektgruppe zu spät erfolgte. Die aus den Technologiestudien gewonnenen Erkenntnisse wären schon im Vorfeld für die Entscheidungsfindung, welche Technologien wie eingesetzt werden, wichtig gewesen. Außerdem wurde der Sinn einer Technologiestudie mit dem Thema Kontext in Frage gestellt, da notwendige Entscheidungen in Bezug auf dieses Thema erst im Entwurf getroffen wurden. Die Ausarbeitungen der Technologiestudien war notwendig, erfolgte aber, so die Meinung der Gruppe, in zu großem Umfang. Kürzere Ausarbeitungen wären ausreichend gewesen, besonders da im Abschluss der Technologiestudien eine Ergebnispräsentation vor des Gruppe erfolgte. Als positiv wurde die Arbeit in Zweierteams bewertet, wodurch die Qualität der einzelnen Studien, wie auch die Motivation der Gruppenmitglieder gefördert wurde.

16.2.4.1 Fazit

Die Erstellung von Technologiestudien ist äußerst sinnvoll, sie sollten aber möglichst früh durchgeführt werden. Am besten bietet es sich hier an, die Technologiestudien mit anderen Arbeitspaketen, wie beispielsweise dem Grobkonzept zu verknüpfen. Somit würde man einen optimalen Informationsrückfluss in der Entwicklung der Software erreichen. Es hat sich auch als vorteilhaft erwiesen, innerhalb der Technologiestudien kleine Machtbarkeittests (Prototypen) durchzuführen.

16.2.5 Erfahrungen aus der Entwurfsphase

Die Ausarbeitung des Entwurfs begann nach der Fertigstellung der Technologiestudien. Es wurde zunächst mit einer Einarbeitung in Design Pattern begonnen, die sich dann im Rahmen einer Präsentation gegenseitig vorgestellt wurden. Dies wurde von den Teilnehmern der Projektgruppe als positiv befunden, da eine gute Grundlage über bisher meist unbekannte Entwurfsmuster geschaffen wurde, die im weiteren Verlauf eingesetzt werden konnten und wurden.
Als generelles Problem in der Entwurfsphase erwies sich die Werzeugauswahl zur Erstellung von UML Diagrammen. Das Arbeiten mit Microsoft Visio wurde als behindernd empfunden, da die Umsetzung der UML-Unterstützung äußerst schlecht gelungen ist. Mangels frei verfügbarer Alternativen musste die Gruppe jedoch wohl oder übel mit diesem Problem leben und hat dieses auch letztendlich gut gemeistert. Aufgrund der mangelnden Erfahrungen der Teilnehmer im Umgang mit Projekten dieser Größenordnung lief die Planung der einzelnen (internen) Deadlines innerhalb der Entwurfsphase nicht immer perfekt und diese Deadlines wurden teilweise überschritten. Diese teilweise Überschreitung der Deadlines liegt jedoch zu einem gewissen Teil daran, dass beim Feinentwurf des Agentenspiels viel zu sehr ins Detail gegangen wurde. Nach Meinung der Gruppe hätte es im Nachhinein vollkommen ausgereicht, die öffentlichen Methoden der einzelnen Klassen detailliert zu beschreiben und nicht, wie geschehen, zusätzlich jedes private Attribut und jede private Methode. Im Gegenzug wurde festgestellt, dass die Schnittstellen der einzelnen Klassen bzw. Module besser entworfen werden könnten, anstatt lediglich modulorientiert vorzugehen.

Das Zusammenspiel der einzelnen Module wurde zwar beschrieben, doch für zukünftige Projekte sollte dies einen noch wichtigeren Status erhalten. (z.B. in Form von Sequenzdiagrammen für alle wichtigen Anwendungsfälle)
Als absolut sinnvoll wurde der durchgeührte Architekturentwurf empfunden und es wurde festgestellt, dass dieser Arbeitsschritt immer am Beginn einer Entwurfsphase stehen sollte, um ein solides Grundgerüst für die weitere Entwurfsphase zu schaffen.

16.2.5.1 Fazit

Der Entwurf hat eine gute und solide Grundlage für die anschließende Implementierungsphase geleistet. Dennoch gibt es die beschriebenen Verbesserungsmöglichkeiten für zukünftige Projekte, um die Entwurfsphase noch optimaler nutzen zu können. Besonders das Konzentrieren auf die Schnittstellen und das Zusammenspiel der einzelnen Klassen untereinander anstelle der viel zu detallierten Klassenbeschreibungen ist hierbei sicherlich ein lohnenswerter Ansatz.

16.2.6 Erfahrungen aus dem Qualitätsmanagement

Das Ziel des Qualitätsmanagements (QM) bestand darin, sicherzustellen, dass die von der Projektgruppe während der Entwurfs-, Implementierungs- und Testphase erstellten Arbeiten den vorgegebenen Qualitätskriterien genügen.
So wurden parallel zur Erstellung des Entwurfs Machbarkeitsstudien erstellt, durch welche bereits frühzeitig die Einsetzbarkeit der unterschiedlichen zur Verfügung stehenden Hard- und Softwarelösungen überprüft wurde. Zur Einschätzung des Aufwands, der für den Einsatz der untersuchten Hard- und Softwarelösungen zu betreiben war, wurden, sofern dies möglich war, einfache Softwareprototypen erstellt. Diese Vorgehensweise bot der Projektgruppe zudem die Möglichkeit, sich im Umgang mit dem später eingesetzten .NET Framework und hier speziell mit der Programmiersprache C# vertaut zu machen. Von der Projektgruppe wurde diese Vorgehensweise als positiv angesehen. Es wurde jedoch angemerkt, dass die Erstellung der Machbarkeitsstudien bereits zu einem früheren Zeitpunkt hätte stattfinden sollen, da bezüglich der Einsetzbarkeit einiger Hard- und Softwarelösungen lange Zeit Unklarheit bestand, was die Projektplanung unnötig erschwerte.
Zu Beginn der Implementierungsphase wurden einige Techniken vorgestellt, durch deren Einsatz die Erstellung von qualitativ hochwertigem, d.h. möglichst fehlerfreiem Quellcode sichergetsellt werden kann. Hierzu zählen eine Einführung in NUnit und das Logging in C#. Darüber hinaus wurden während der Implementierungsphase eine Reihe von Integrationstests durchgeführt, die dem Zweck dienten, das Zusammenspiel einzelner Module untereinander zu testen. Die Integrationstests orientierten sich hierbei an den Anwendungsfällen des Implementierungsplans auf Seite 160. Diese Vorgehensweise wurde von der Projektgruppe äußerst positiv bewertet.
Eine weitere Aufgabe des Qualitätsmanagements bestand in der Planung, Durchführung und Auswertung der Abschlusstests. Zu diese Zweck wurden komplette Spielabläufe festgelegt und anschließend von der Gruppe unter realistischen Bedingungen (im Freien mit mobilen Endgeräten) durchlaufen. Die Verwendung eines vorgegebenen Spielablaufs für die Durchführung der Abschlusstests wurde von den Mitgliedern der Projektgruppe als die sinnvollste Möglichkeit angesehen, die fertige Anwendung zu testen.
Abschließend wurde von der Projektgruppe angemerkt, dass die Trennung von Qualitätssicherung (QS) und QM wenig sinnvoll war. Es herrschte Einigkeit darüber, dass es besser gewesen wäre, wenn das QM bereits mit Beginn der Projektgruppe begonnen hätte. Planung und Durchführung der einzelnen Teilaufgaben des QM, speziell die Integrationstests, wurden positiv bewertet.

16.2.6.1 Fazit

Für die Durchführung eines erfolgreichen Softwareprojekts ist ein vernünftigen QM unabdingbar. Das QM sollte dabei mit Beginn des Projekts einsetzen und neben der Validierung und Verifikation des erstellten Quellcodes, auch für Konsistenz und Korrektheit der erstellten Dokumente verantwortlich zeichnen. Dabei sollte sich das QM auf allgemein anerkannte Vorgehensweisen stützen und stets auf Einhaltung existierender Standards achten.

16.2.7 Erfahrungen mit der Implementierung

16.2.7.1 Planung / Tatsächlicher Ablauf

Eine weit verbreitete Meinung ist, dass die Implementierung anfangs nur schleppend voranzukommen schien, so dass zeitweilig sogar die Befürchtung umging, nicht fertig zu werden, also am Ende kein wirklich laufendes Produkt abliefern zu können, das wenigstens die geforderten Grundfunktionen bieten würde. Durchweg positiv vermerkt wurde, dass sich mit voranschreitender Projektzeit, jeder mehr und mehr reinzuhängen schien und dann doch mehr fertiggestellt werden konnte, als zwischenzeitlich angenommen wurde. Teilweise wird dies auch damit erklärt, dass die Planungsrichtung anfangs modulorientiert verlief. Die Verteilung der Implementierungsaufgaben nach Modulen erschien vielen hinderlich und weniger produktiv. Erst nach einem Wechsel zu einer anwendungsorientierten Planungsrichtung schien die Implementierung schneller voranzuschreiten. Die Aufgaben wurden für die meisten weniger abstrakt und es bot sich ein Blick auf das große Ganze jenseits des eigenen Moduls, so dass am Ende der Implementierung nahezu jeder an jedem Modul programmiert hat und ein besseres Verständnis des Zusammenspiels der einzelnen Module miteinander erlangte. Dies schien sich auch in einer steigenden Qualität des Codes wiederzuspiegeln. Nichtsdestotrotz wurde auch angemerkt, dass die anfängliche Modulorientiertheit durchaus auch seine Berechtigung hat. Auf sie zu verzichten, hätte zu schmutzigem Programmieren mit unfertigen Modulen führen können, wenn jeder nur noch darauf geachtet hätte, dass sein Anwendungsfall funktioniert. Letzendlich hat es sich bewährt, erst scheinbar mühsam das Modulgerüst aufzubauen, und die komplizierteren Teile erst anschließend anwendungsorientiert hinzuzufügen. Allgemein herrscht rückblickend die Meinung, dass die Implementierung gut gelaufen ist. Der gute Verlauf der Implementierung wird nicht zuletzt auch auf das gut empfundene Arbeiten in angehnemer Atmosphäre und ein stärkeres Teamgefühl als in der Entwurfsphase zurückzuführen sein. Als wichtigen Faktor hierfür wurden die vielen Gruppentreffen auch zu sozialen beziehungsweise sportlichen Anlässen identifiziert. Auch die Kommunikation innerhalb der Gruppe wurde insbesondere durch die Programmiertreffen, ob mit der gesamten Gruppe oder in Zweierteams, gefördert. Insgesamt schien sie immer etwas vernachlässigt worden zu sein. So wurde das dafür vorgesehene Forum nur phasenweise von allen regelmäßig genutzt. Bei Problemen oder Unklarheiten hätte öfter und schneller eine Email verfasst werden können, wobei angemerkt wurde, dass auf Rundmails allgemein mehr Feedback hätte kommen können. Das zur Verfügung stehende Multimedialabor des OFFIS hätte wohl noch öfter für Programmiertreffen mit allen Teilnehmern genutzt werden können. Auf diese Weise wäre die Implementierung durch schnellere Absprachen möglicherweise beschleunigt werden können.

Nachdem der Wechsel von modulorientierter zu anwendungsorientierter Implementierung und größere Programmiertreffen in der Gruppe weitestgehend als gute Entscheidung aufgefasst werden, gehen die Meinungen zum verwendeten Observer-Pattern auseinander. Zum einen wird ihm zu Gute gehalten, dass es für leicht verständliche Abläufe sorgt und schon früh dazu veranlasst, modulübergreifend zu arbeiten. Als weniger vorteilhaft kann dessen Verwendung allerdings in Anbetracht des in der verwendeten Programmiersprache C# bereits enthaltenen Eventmodells angesehen werden. Das interne Eventmodell

wäre insofern komfortabler gewesen, als dass es nicht selbst hätte geschrieben werden müssen und durch seine vergleichsweise feingranulare Struktur zudem noch performanter als das gewählte Observer-Pattern sein könnte.

Als weiteren Punkt wurde genannt, man hätte eventuell zu viele Features eingebaut. Vorteilhafter hätte es sein können, sich zunächst auf wenigere zu beschränken, und im Gegenzug weniger Bugs gegenüber zu stehen. Dies sei bereits ein Problem des Entwurfs gewesen, von dem zu viel zu übernehmen versucht wurde. Teilweise negativ empfunden wurde das geringe Interesse der Betreuer an der Implementierung.

16.2.7.2 Entwicklungsumgebung / verwendete Tools

Ein bedeutender Faktor für die Implementierung ist, dass auf eine Entwicklungsumgebung gesetzt wurde, mit der vorher noch kein Teilnehmer zu tun hatte. So boten die Programmiersprache C#, Programmierung mobiler Geräte und das verwendete Compact Framework für alle neue Erfahrungen, die mit den üblichen Eingewöhungsschwierigkeiten einhergingen. Dabei stellte sich besonders die Verwendung von PDAs als aufwendig und zeitintensiv heraus. Viel Zeit wurde darin investiert ActiveSync zur Datenübertragung zwischen PC und PDA überhaupt zum Laufen zu bringen, Anwendungen zu starten und eine Verbindung mit einem Server herzustellen. Ein weiteres Problem entstand aus dem Zusammenspiel von Visual Studio und WinCVS. Ein häufig aufgetretener Fehler war, dass nicht immer alle Dateien als verändert markiert worden schienen und demzufolge auch nicht mit ins CVS eingecheckt wurden. Während für den Einchecken den im Anschluss keine Probleme erkennbar waren, konnte das Projekt bei allen Anderen nicht mehr kompilieren. Keine Probleme traten hingegen auf, wenn man statt WinCVS das CVS-Plugin von Eclipse benutzte, was vielleicht für spätere Projekte im Hinterkopf zu behalten ist. Die Verwendung des in Visual Studio enthaltenen GUI-Editors zur Erstellung der GUI auf dem Client hätte eventuell viel Zeit sparen können. Dazu muss allerdings auch gesagt werden, dass eine so erstellte GUI wahrscheinlich schlechter zu reenginieren wäre und der GUI-Editor auch wegen seiner Codeformatierung eher mit Vorsicht zu genießen ist.

16.2.7.3 Pair Programming

Bei der Implementierung wurde auf das Prinzip des Pair Progammings gesetzt, d.h. es wurde sich zum Programmieren in Zweierteams zusammengefunden und die Aufgaben dann gemeinsam bearbeitet. Der dadurch vielleicht zu erwartene Ausfall von Arbeitskraft sollte durch höhere Qualität und somit letztendlich weniger Aufwand ausgegelichen werden. Allgemein wurde dies von der Gruppe gut angenommen und festgestellt, dass es besonders Programmierern mit weniger Erfahrung zu Gute kommt. So bekommt man schnelleres Feedback bei Fehlern und steigert schon dadurch, dass vier Augen mehr sehen als zwei die Qualität des Codes. Außerdem lösen sich viele Probleme, an denen man alleine oft lange grübeln würde, oft schon bei der Diskussion mit dem Partner. Eine weitere positive Eigenschaft des Pair Programmings ist die Überwindung des inneren „Schweinehunds“. Beim Programmieren zu Hause bieten sich zu viele Gelegenheiten von der Arbeit abgelenkt zu werden, während man, wenn man sich schon zum Programmieren getroffen hat, besser am Ball bleibt und die Aufgaben durchzieht. Neben der erhöhten Motivation wird zudem der Vorteil, zum durchdachten Arbeiten gezwungen zu werden, identifiziert. Unüberlegtes „Drauflos-Hacken“, welches zu schwer verständlichem Code oder Fehlern führen könnte, wird so vermieden. Die zunächst vorherrschende Meinung, ein stärkerer und ein schwächerer Programmierer wären eine ideale Mischung wurde im Laufe der Implementierung teilweise abgelegt. Hier wird der stärkere Partner gebremst und man sollte sich bei solchen Teams vielleicht eher auf den Austausch über die zu erledigenden Aufgaben beschränken und die Implementierung dann doch einzeln erfolgen. Dies trifft auch auf Fleißaufgaben zu, bei denen Problem und Lösung bereits bekannt sind, wie beispielsweise Feinjustierungen

an der GUI. Besonders ausgezahlt hat sich das Pair Programming hingegen bei logischen Problemen, die mehr Überlegung erfordern.

16.2.7.4 NUnit

Die ursprüngliche Vorgabe ausschließlich testgetrieben zu implementieren wurde zu einer freiwilligen Angelegenheit. Dabei sollten vor Erstellung eines Moduls zunächst in NUnit Testfälle angelegt werden, die, würden sie erfolgreichen verlaufen, die korrekte Fertigstellung des eigentlich zu implementierenden Moduls sicherstellen sollen. Dies erschien vielen erst gewöhnungsbedürftig und der Sinn dieser Maßnahme wollte nicht jedem sofort einleuchten. Im Laufe der Implementierung wurde NUnit in der gesamten Gruppe mehr und mehr zu schätzen gelernt. Tatsächlich wurde erkannt, dass mit NUnit durchaus Zeit eingespart werden kann, da man die investierte Zeit schnell wieder rausbekommt, wenn man nicht immer zum Testen den PDA-Emulator starten muß. Manche Teile konnten sogar nur mit Hilfe von NUnitTests zum Laufen gebracht werden. Allerdings verlangen die NUnit-Tests auch ein gewisses Maß an Disziplin, da sie während der Implementierung immer wieder an den aktuellen Stand des Projekts angepasst werden müssen. Begrenzt ist ihr Nutzen außerdem bei verteilten Anwendungen und bei Tests die sehr viele Vorbedingungen haben. Dennoch hat NUnit letztendlich alle überzeugt; Eine Einsicht die gruppenintern als *Evolutionsschritt eines Informatikers* angesehen wird.

16.2.7.5 Kollektive Code-Ownership

Die Tatsache, dass niemand einen Teil des Codes exklusiv für sich hatte, sondern vielmehr jeder an allem gearbeitet hat, störte niemanden und wurde allemein als gut empfunden. Dies ist vor allem darauf zurückzuführen, dass Änderungen stets abgesprochen und sinnvoll waren. Nur vereinzelt sind so aus Mangel an Sorgfalt Fehler aufgetreten. Erleichtert wurde der projektübergreifende Implementierungseinsatz aller durch gute Kommentare im Code. Dennoch wurde angemerkt, dass zu häufig Kommentare fehlten.

16.2.8 Erfahrungen aus dem Projektabschluss

In der Planungsphase wurde ein erheblich größerer Aufwand für dieses Paket vermutet. Dies lag vor allem daran, dass viele Punkte aus dem Paket Projektabschluss, die laut Literatur dazugehören, nicht weiter verfolgt worden sind und von den Betreuern für unnötig erachtet wurden. Der Projektabschluss ist trotzdem ein wichtiger Teil eines Projektes. Hervorzuheben sind hier die Erfahrungsberichte, von denen in späteren Projekten profitiert werden könnte. An der Planung des Projektabschlusses würde nichts geändert werden müssen, da in zukünftigen Projekten in der Berufswelt alle dazugehörigen Punkte wichtig sind, was in dem Fall einer Projektgruppe an der Universität, wie von den Betreuern angeregt, nicht der Fall ist.

16.2.9 Erfahrungen aus der Projektplanung

16.2.9.1 Projektkontrolle

Zu Beginn des Projekts bestand der einzige Kontrollmechanismus darin, dass zu beginn jeder Gruppensitzung ein so genannte Blitzlicht abgehalten wurde. Dabei erzählte jeder Teilnehmer, was er seit dem vergangenen Treffen für die Projektgruppe getan hatte. Dies reichte jedoch nicht aus, weil nicht transparent genug war, wer welche Aufgaben zu erledigen hatte. Das führte unter anderem dazu, dass Teilnehmern den heranrückenden Deadlines erst zu spät gewahr wurden, um sie noch einhalten zu können.

Als Reaktion darauf versuchte die Gruppe, vergebene Aufgaben in einer Aufgabenliste zu sammeln. Diese wurde jedoch nicht gut angenommen, so dass die Idee bald wieder verworfen wurde. Die Situation verbesserte sich spürbar, als die Gruppe anfing, Aufgaben nur noch kurzfristig zu verteilen, und sie samt Deadline in den Protokollen vermerkte. Zusätzlich wurden die Aufgabenlisten von den Protokollen immer auf die Tagesordnungen zu den folgenden Gruppensitzungen übertragen, so dass offensichtlich wurde, wer welche Aufgaben zu erledigen hatte und was davon tatsächlich erledigt worden war.

Ein zweiter Kontrollmechanismus wurde in Form von Stundenzetteln eingeführt, nachdem aufgefallen war, dass die Arbeitsleistung der einzelnen Gruppenmitglieder zum Teil deutlich auseinander lag. In die Stundenzettel wurde jede Tätigkeit mit Datum, Dauer, zugehörigem Arbeitspaket und Kommentar vermerkt. Zunächst wurden sie jedoch nur mäßig benutzt. Erst als die Einträge der vergangenen Woche zu jedem Donnerstags-Treffen in Form einer Wochenstatistik zusammengefasst und ausgeteilt wurden, begannen alle, die Stundenzettel konsequent zu nutzen. Da jeder nun die Möglichkeit hatte, seine Arbeitsleistung zu überwachen und sie mit denen der anderen zu vergleichen, leistete bald jeder die geforderte Stundenzahl und die Leistung der einzelnen glich sich einander an.

16.2.9.2 Teamarbeit und Kommunikation

Während des Informatikstudiums wird darauf hingewiesen, dass Teamarbeit und zwischenmenschliche Kommunikation ein wesentlicher Teil der Projektarbeit ist. Die Projektgruppe traf sich darum zweimal wöchentlich, um sich auszutauschen, Probleme zu diskutieren und neue Aufgaben zu definieren und zu verteilen. Neben den Treffen war der Email-Verteiler das zweite wesentlich Kommunikationsmedium innerhalb der Gruppe. Zusätzlich wurde ein Forum aufgesetzt, in dem vor allem während des Entwurfs viele Probleme diskutiert wurden. Dazu wurde eine Telefonliste ausgeteilt, so dass einige Probleme auch telefonisch geklärt werden konnten.

Die Kommunikation der Teilnehmer wurde weiterhin dadurch gefördert, dass Aufgaben oft in kleinen Teams erledigt wurden. Ein weiterer Vorteil war, dass es den meisten Teilnehmer leichter viel, sich auf die Arbeit zu konzentrieren, wenn sie diese in kleinen Teams erledigten.

16.2.9.3 Flexibilität

Die Gruppe empfand es als positiv, dass die Projektplanung während des Projekts flexibel an die aktuellen Entwicklungen angepasst wurde. So war es z.B. ein mutiger aber im Nachhinein sinnvoller Schritt, auf die Entwicklung eines Frameworks zu Gunsten einer ausgereiften Anwendung zu verzichten. Es wurde auch als positiv genannt, dass die Entwurfsphase beendet wurde, als sich die Gruppe zu sehr in Details verstrickte. Während der Implementierung lösten sich die Detail-Probleme zumeist von allein oder viel leichter, weil man bereist ein besseres Verständnis für die Anwendung hatte. Ebenfalls als positiv, wurde das Setzen realistischer Ziele genannt. Als einigermaßen deutlich wurde, welche Anwendungsfälle mit der geforderten Arbeitsleistung von 20 Stunden/Woche bis zum Ende des Projekts umzusetzen sein würden, richtete die Gruppe ihre Planung daran aus. In Folge dessen wurde das Projekt pünktlich fertig, ohne dass die Teilnehmer Sonderschichten einlegen mussten.

16.2.9.4 Ideen für zukünftige Projekte

Im Folgenden ist beschrieben, wie die Gruppe mit der gewonnenen Erfahrung ein ähnliches Projekt in Zukunft planen würde. Er soll damit eine Hilfestellung für die Planung weiterer Projekte sein.

Es wurde als gut empfunden, dass die Seminarphase straff organisiert war, und es strikte Deadlines für

die Abgabe der Seminararbeiten gab. So wurde keine Arbeit unnötig aufgeschoben und mit in das Projekt hineingetragen. Die Phase der Ideenfindung wurde allgemein als zu ausgedehnt empfunden. Außerdem sollte im Rahmen des Qualitätsmanagement schon nach kurzer Zeit die Umsetzbarkeit der Ideen getestet werden, so dass nicht unnötig viel Zeit in deren detaillierte Ausformulierung gesteckt wird. Aus der Ideenphase sollte dann ein Dokument entstehen, das als fachliches Modell, oder zumindest als Vorlage dafür verwendet werden kann. Dank dieses Dokuments könnte dann eventuell auf das Schreiben einer Anforderungsdefinition verzichtet werden, wenn die Betreuer nicht fordern, die Implementierung bestimmter Features oder Software-Qualitätsmerkmale vertraglich zu regeln. Die erste Aktion zur Umsetzung der Idee, sollte eine objektorientierte Analyse der Ergebnisse aus der Ideenfindungsphase sein. Was in diesem Projekt das Grobkonzept bzw. die Fallstudie und die Technologiestudie geleistet haben, könnte noch besser im Rahmen des Architekturentwurfs stattfinden. Während der Fallstudie fanden bereits der Entwurf der Struktursicht und deren Bewertung statt. Die Technologiestudie würde durch die Risikoanalyse ersetzt werden. Den Feinentwurf bewertete die Gruppe im Nachhinein als zu detailliert ausgearbeitet. Stattdessen sollte der Fokus auf die Beschreibung der Schnittstellen und der Dynamik des Systems beschränkt werden. Bei der Implementierung würde die Gruppe in Zukunft von vorne herein auf Anwendungsfälle bezogene Aufgaben verteilen.

16.2.9.5 Fazit

Die Projektplanung hat zwei wesentliche Ziele der Projektgruppe erreicht. Zum einen ist das Projekt rechtzeitig fertig geworden und erfüllt die geforderten Anforderungen, wobei nahezu alle Gruppenmitglieder einen großen Beitrag geleistet haben. Zum anderen wurden Lernschritte im Bezug auf Planung und Projektkontrolle gemacht, von denen die wichtigsten in diesem Erfahrungsbericht festgehalten wurden. Im Hinblick auf diese Punkte kann man das Projekt als vollen Erfolg bezeichnen.

16.3 Bewertung und Ausblick

Dieser Abschnitt fasst die Bewertung des Projekts und des entstandenen Produkts zusammen, und gibt einen Ausblick über denkbare Erweiterungen des Softwareprodukts.

16.3.1 Bewertung des Produkts

Mit NABB hat die Projektgruppe ein stabiles und einfach zu bedienendes Softwareprodukt geschaffen. Es wurden nahezu alle in der Anforderungsdefinition geforderten Anwendungsfälle und diverse zusätzliche umgesetzt. Die Aspekte der Aufgabenstellung wurden allesamt in dem Spiel umgesetzt:

- **Kontextsensitivität**: Das Spiel nimmt Änderungen am Kontext des Spielers wahr, wie z.B. seine Bewegungsgeschwindigkeit, die Nähe anderer Spieler oder die Aktivierung bestimmter Spezialfähigkeiten.
- **Umgebungserkundung**: Die Software setzt diesen Aspekt über die Positionsbestimmung mittels GPS um. Der Spieler erkundet die Umgebung nach Mitspielern und virtuellen Spielobjekten.
- **Mobile Unterstützung**: NABB wird ausschließlich auf mobilen Endgeräten gespielt, die von einem zentralen Spielserver koordiniert werden.
- **Multimodalität**: Es liegt in der Natur von Spielen, dass diese medial reichhaltige Anwendungen sind. Das Spiel wählt anhand des aktuellen Spielkontext geeignete Ausgabemodalitäten aus, um

den Spieler über das Auftreten von Spielereignissen und -zuständen zu informieren. Es bedient sich dabei bei allen hardwareseitig zur Verfügung stehenden Ausgabekanälen.

In den Integrationstests hat die Gruppe bewiesen, dass das Programm die geforderten Anwendungsfälle fehlerfrei umsetzt. Während den Testphasen hat das Programm außerdem die Stabilität der Software unter Beweis gestellt.

Als Schwachpunkt stellte sich die Internetverbindung über Mobiltelefon heraus. Während der Testphasen brach die Verbindung oft ab, so dass das Spiel neu gestartet werden musste. Außerdem erwies sich die Handhabung der Hardware bestehend aus PDA, GPS Empfänger und Mobiltelefon als relativ umständlich. Zum einen gestaltet sich die Einrichtung der Hardware gewöhnungsbedürftig und zum anderen existierten nicht genügend Hardwaregeräte, um das Spiel mit einer großen Anzahl an Spielern auf seine Spielbarkeit und seinen Spielspaß hin testen zu können.

16.3.2 Bewertung des Projekts

Nach der üblichen Eingewöhnungsphase arbeitete die Gruppe gut und intensiv zusammen. Trotz der fehlenden Implementierungserfahrung der meisten Teilnehmer gelang es der Gruppe, zum vorgegebenen Termin eine stabile und gut getestete Software abzuliefern, die nahezu alle Anforderungen aus dem Pflichtenheft und diverse weitere Features umsetzt. Die Gruppe hat flexibel auf Probleme reagiert und die gesammelten Erfahrungen in Erfahrungsberichten festgehalten. Jeder Teilnehmer arbeitete aktiv bis zum Ende an der Projektgruppe mit und trug seinen Teil zum Gelingen des Projekts bei, ohne dass dabei der Spaß an der Arbeit verloren ging. Auch außerhalb der Projektarbeit herrschte reger Kontakt zwischen den Teilnehmern. So wurde z.B. wöchentlich für das PG Fußballturnier trainiert, wobei sich die Projektgruppe einen hervorragenden zweiten Platz erkämpfte.

16.3.3 Ausblick

In der kreativen Vorbereitungsphase der Projektgruppe wurden Ideen zur Umsetzung des Projekttitels in einen Arbeitsauftrag gesammelt und diskutiert. Viele der Ideen sind in der ein oder anderen Form in das Agentenspiel eingeflossen, andere wurden, in der Regel aus Zeitgründen, nicht umgesetzt. Des Weiteren hat die Phase der Implementierung neue Anregungen geliefert. Da diese Ideen ein Teil der Projektarbeit darstellen und sie deshalb nicht verloren gehen sollen, ordnet dieser Abschnitt diese Ideen nach drei logischen Aspekten. Der erste Aspekt, „Erweiterung des Agentenspiels“, befasst sich mit Punkten, die das bestehende Spiel nach Meinung der Projektgruppe aufwerten würden, ohne jedoch seine Semantik zu verändern. Der zweite Aspekt, „Erweiterung der Spielidee“, führt Punkte auf, die Einfluss auf die Spiellogik haben und damit in der Regel aufwendiger zu realisieren sind. Der dritte und letzte Aspekt, „Technische Erweiterungen“, behandelt technische Erweiterungen des Agentenspiels, die sich für eine breite Nutzung des Spiels, auch durch behinderte Menschen, positiv auswirken würden.

Erweiterung des Agentenspiels

- Das Agentenspiel erhält eine Beschreibung der Rahmenhandlung, die als Hintergrundgeschichte dient.
- Zu jedem Szenario gibt es eine separate Geschichte, die auf die Hintergrundgeschichte des Agentenspiels aufsetzt, sowie eine ausführliche Missionsbeschreibung.
- Zum Spielstart wird ein „Splash-Screen“ eingeblendet.

- Es gibt eine Onlinehilfe und Tooltipps im Spiel.
- Anfängern steht ein Tutorial zur Verfügung.
- Die Darstellung von der Karte und den POIs kann auf Wunsch auch akustisch erfolgen.

Erweiterung der Spielidee

- Es gibt einen Spielverwaltungsserver als Metaserver, der die einzelnen regionalen Spielserver kennt und diesem zur Auswahl auflistet.
- Neben reinen Textnachrichten können auch Medien wie Photos, Videos und Audionachrichten verschickt werden.
- Goodies repräsentieren, neben Codefragmenten zur Lösung des Missionsziels, weitere Effekte wie bspw. die Veränderung von Fähigkeitspunkten.
- Die grafische Benutzerschnittstelle passt sich der Veränderung des Kontexts an, bspw. durch größere Buttons bei hoher Geschwindigkeit.
- Es gibt einen Szenarioeditor zur komfortablen Erstellung oder Änderung von Szenarien.

Technische Erweiterungen

- Die Eingabe kann multimodal erfolgen.
- Es gibt eine Spracherkennung und -Ausgabe.
- Das Agentenspiel wird auf Mobiltelefone portiert, bspw. für Symbian oder J2ME.

Die Projektgruppe schätzt die Punkte der ersten beiden Aspekte, „Erweiterung des Agentenspiels“ und „Erweiterung der Spielidee“, in einem überschaubaren Zeitraum als realisierbar ein, die technische Erweiterungen des Agentenspiels jedoch erscheinen aufwendig. Verfügbare Spracherkennungssoftware funktionierte bis zum dem Zeitpunkt des Ablaufs der Projektgruppe noch nicht zuverlässig. Für eine Portierung auf andere Plattformen, die nicht das .NET CF unterstützen, muss das Agentenspiel neu geschrieben werden. Als idealistisches Fernziel sieht die Projektgruppe die Spielbarkeit des Agentenspiels für sehbehinderte Menschen. Navigation in unbekanntem Gelände ist für Blinde aber ein Gebiet, auf dem noch Pionierarbeit verrichtet wird.

Kapitel 17

Handbuch

17.1 Einleitung

Vielen Dank, dass sie sich für NABB entschieden haben. NABB ist im Rahmen der Projektgruppe „Kontextsensitive Umgebungserkundung mit mobiler multimodaler Unterstützung“ entstanden. Die Projektgruppe bestand aus den folgenden Personen:

Stefan Andreßen
Arne Bartels
Frank Jellinghaus
Steffen Kruse
Olaf Lehde
Daniel Nüss
Michael Onken
Jens Peternel
Martin Pielot

17.1.1 Was ist NABB?

NABB ist ein Outdoor-Spiel, dass sich für Gruppen ab vier Personen eignet. Es gibt die beiden Teams der *Defender* und der *Infiltratoren*, auf die die Spieler gleichmäßig verteilt werden. Im Team der *Infiltratoren* nimmt ein Spieler die Rolle eines besonderen Agenten, des *Commander*, ein. Nur der *Commander* kennt das Missionsziel seines Teams und muss die Vorgehensweise seiner Mitspieler koordinieren, um das Missionsziel zu erfüllen und somit Spiel gewinnen zu können. Im Gegenzug versuchen die *Defender* die *Infiltratoren* an der Erfüllung ihrer Aufgaben zu hindern.

17.1.2 Was wird benötigt?

Um NABB spielen zu können wird für jeden Mitspieler ein PDA mit GPS-Empfänger benötigt. Zusätzlich wird ein Handy benötigt, über welches der PDA eine GPRS-Verbindung aufbauen kann. Ferner wird noch ein Server gebraucht.

17.2 Spielregeln und Ablauf

Bevor NABB gespielt werden kann, muss zunächst ein Server gestartet werden. Anschließend können die Clients gestartet werden und das Spiel kann beginnen.

17.2.1 Server

17.2.1.1 Starten des Servers

Zunächst müssem Sie den Server starten. Hierfür klicken Sie in Menüleiste auf Spiel und anschließend auf *Szenario laden...*(Abb. 17.1).

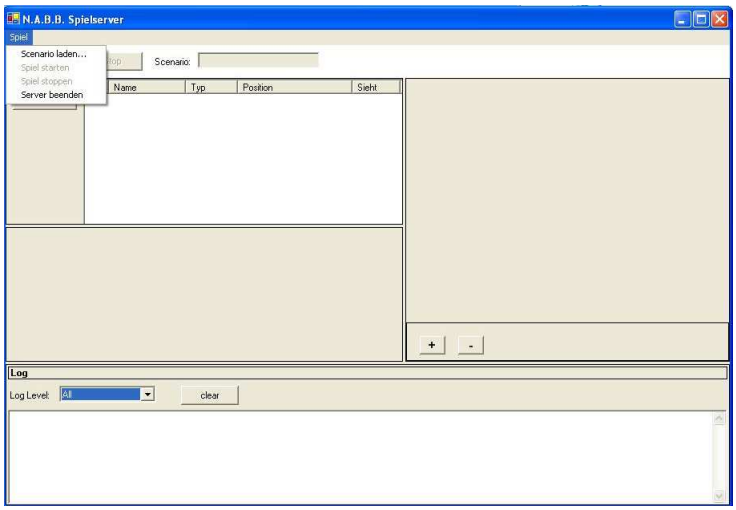


Abbildung 17.1: Serverstart

Es erscheint eine Liste von Szenarien, die geladen werden können (17.2). Klicken Sie auf das Szenario, das gespielt werden soll und öffnen Sie dieses. Das Fenster wird wieder geschlossen. Um den Server zu Starten klicken Sie auf den Button *Start*.

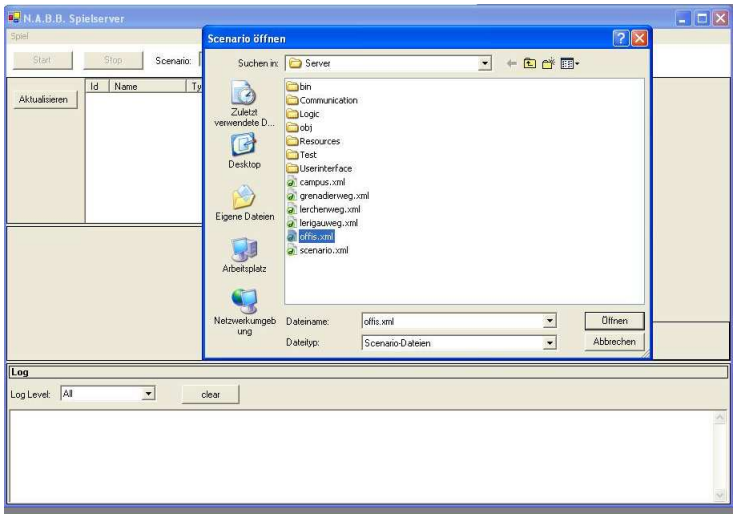


Abbildung 17.2: Szenarioauswahl

17.2.1.2 Funktionalitäten des Servers

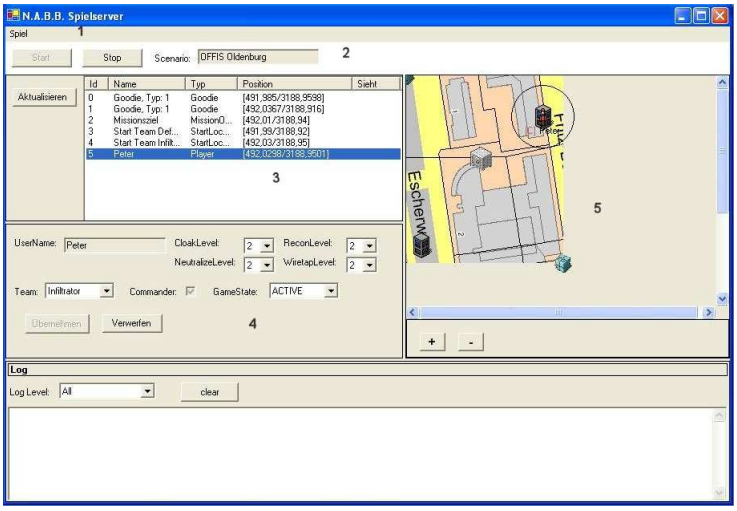


Abbildung 17.3: Server ist gestartet

In der Menüleiste (Abb. 17.3, Punkt 1) gibt es den Menüpunkt *Spiel*. Hier können Sie

- ein Szenario laden,
- das Spiel starten,
- das Spiel stoppen,
- und den Server beenden.

In der darunter liegenden Buttonleiste (Abb. 17.3, Punkt 2) können Sie ebenfalls das Spiel starten und stoppen. Zusätzlich steht hier, welches Szenario geladen ist.

Im mittleren Fenster (Abb. 17.3, Punkt 3) können Sie die Points-of-Interest (POI) sehen. POI können Spieler, Goodies, Missionsziele und Startpunkte sein. Zu erkennen sind

- die ID,
- der Name,
- der Typ und
- Position

des POI. Falls der POI ein Spieler ist, können sie noch erkennen, was der Spieler gerade sieht.

klicken Sie auf ein POI, so erhalten Sie im darunterliegenden Fenster (Abb. 17.3, Punkt 4) Informationen über das POI.

Auf der Karte (Abb. 17.3, Punkt 5)ist das Spielfeld zu sehen. Die Karte kann mit den darunterliegeden Buttons gezoomt werden. POIs werden auf der Karte durch Symbole dargestellt. In der rechten unteren Ecke ist das Symbol für ein Goodie (17.3.1), in der linken unteren Ecke ist das Symbol für einen Startpunkt, in der Mitte sieht man das Symbol für ein Missionsziel (17.9). Rechts in der Mitte sieht den Spieler mit Namen. Das *C* steht für Commander (Abschnitt 17.3), der Kreis um den Spieler ist sein Aufklärungsradius. In diesem Aufklärungsradius sieht der Spieler alle POIs.

17.2.2 Client

Die Mitspieler können jetzt den Client starten und das Spiel beginnen.

Abbildung 17.4: Start des Client

17.2.2.1 Charakter einrichten

Wenn Sie den Client gestartet haben, können Sie Ihren Spieler und das Szenario einrichten (Abb. 17.4). Im oberen Textfeld (Abb. 17.4, Punkt 1) müssen Sie das Szenario eingeben. Im zweiten Textfeld (Abb. 17.5, Punkt 2) müssen Sie die IP-Adresse des Servers angeben. Darunter (Abb. 17.4, Punkt 3) müssen Sie Ihren Spielernamen eingeben.

Im darunterliegenden Feld (Abb. 17.4, Punkt 4) können Sie Ihre Fähigkeitspunkte für die Fähigkeiten *Neutralisieren, Abhören, Aufklären und tarnen* vergeben. Je höher die Punktzahl bei einer Fähigkeit ist, desto höher ist die Chance, diese erfolgreich einzusetzen. Jedoch dürfen Sie maximal acht Punkte vergeben. Hierbei ist jedoch zu beachten, dass auf eine Fähigkeit mindestens 1 Punkt verteilt werden muss und maximal 4 Punkte verteilt werden können.

Neutralisieren/neutralize *Neutralisieren* kann in einem bestimmten Radius um den aktuellen Standort des Spielers eingesetzt werden. Dieser Radius beträgt bei einem Fähigkeitspunkt 20 Meter, bei 2 bzw. 3 Fähigkeitspunkten je 24 Meter und bei 4 Fähigkeitspunkten 28 Meter. Die Dauer des Neutralisierungsvorgangs beträgt 5 Sekunden. Alle Spieler, die sich nach diesen 5 Sekunden innerhalb des Neutralisierungsradius befinden, werden neutralisiert. Einem Versuch der Neutralisation kann man sich nur entziehen, indem man sich vor Ablauf der Neutralisationszeit aus dem Neutralisationsradius entfernt.
Wird man während des Spiels neutralisiert, so wird der Spielstatus auf Passiv gesetzt und man muss wieder zum Startpunkt, um sich wieder zu aktivieren. Sollte man Goodies (17.3.1) eingesammelt haben, so werden diese an der Stelle der Neutralisation wieder abgelegt.

Tarnen/cloak Ein Spieler kann sich während des Spielverlaufs tarnen und ist dann nicht mehr für seine Mitspieler und Gegner auf der Übersichtskarte zu sehen. Er kann allerdings auf seiner Karte auch keine *Points Of Interest* mehr sehen. Der Tarnmodus steht für eine bestimmte Zeit zur Verfügung. Bei einem Fähigkeitspunkt kann sich ein Spieler für 50 Sekunden tarnen, bei 2 Punkten für 70, bei 3 Punkte für 80 und bei 4 verteilten Fähigkeitspunkten für 100 Sekunden. Die einzige Möglichkeit einen getarnten Spieler zu enttarnen ist, wenn dieser den Sichtradius einer Aufklärungskamera (17.2.2.1) betritt.

Abhören/wiretap Wenn die Funktion *Abhören* aktiviert wird, können Nachrichten des gegnerischen Teams abgehört werden. Der Abhörmodus steht für eine bestimmte Zeit zur Verfügung. Bei einem Fähigkeitspunkt kann ein Spieler für 50 Sekunden abhören, bei 2 Punkten für 70, bei 3 Punkte für 80 und bei 4 verteilten Fähigkeitspunkten für 100 Sekunden.

Aufklären/recon Diese Fähigkeit erlaubt das Setzen von Aufklärungskameras auf dem Spielfeld, wobei die Kamera an der momentanen Position des Spielers abgelegt wird. Jede Kamera hat ihrerseits einen Sichtradius, in dem alle POIs sichtbar werden. So können selbst getarnte Spieler aufgespürt werden, sobald diese sich in den Sichtradius einer Aufklärungskamera bewegen. Bei einem Fähigkeitspunkt beträgt der Radius 20 Meter, bei 2 bzw. 3 Fähigkeitspunkten je 24 Meter und bei 4 Fähigkeitspunkten 28 Meter. Die Anzahl der Kameras entspricht den verteilten Punkten auf *Aufklären*. Platzierte Kameras , können jederzeit vom Spieler wieder eingesammelt werden, indem er zu deren Position bewegt und die entsprechende Funktion ausführt. Gegnerische Kameras können auf diese Weise deaktiviert werden.

Bestätigen Sie Ihre Eingeben mit dem *OK-Button*.

Und nun viel Spass mit NABB

17.3 Das Spiel

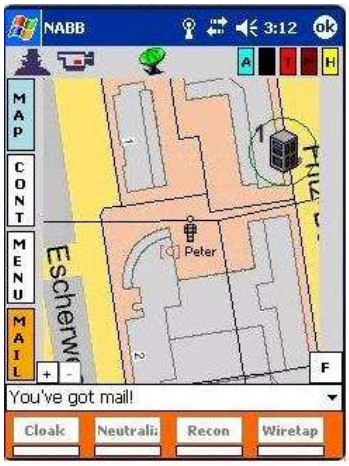


Abbildung 17.5: Client ist gestartet

Der Spieler wird über seine Teamzugehörigkeit informiert. Der erste Spieler, der dem Spiel als Mitglied der *Infiltratoren* beitritt, wird zum *Commander* ernannt.
Dann muss jeder Spieler seinen ihm zugewiesenen Startpunkt aufsuchen, damit seine Fähigkeiten aktiviert werden und er das Spielgeschehen aktiv beeinflussen kann. Der Startpunkt ist auf der Karte zu sehen (Abb. 17.5, Punkt 2). Die Symbole auf der Karte entsprechen den Symbole auf der Karte des Servers. Hat der Spieler den Startpunkt erreicht, so werden seine Fähigkeiten aufgeladen (Abb. 17.6).

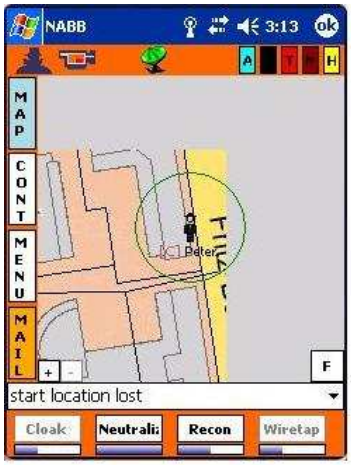


Abbildung 17.6: Spieler ist aktiviert

17.3.1 Goodies

Ein weiterer Teil des Spiels sind spezielle Objekte, sogenannte Goodies, die sich auf dem Spielfeld befinden und von den Spielern aufgespürt werden müssen. Diese können kleine Bonifikationen in Form von temporären Upgrades auf bestimmte Fähigkeiten oder Informationen für einen Spieler enthalten, die zur Erfüllung der Missionsziele dienen. Informationen für die Missionsziele können nur die Infiltratoren abrufen. Diese Informationen werden als Nachricht vom Server übermittelt. Um einen solchen Bonus erhalten

zu können, muss der Spieler zunächst ein Goodie einsammeln. Dazu muss er sich in die Nähe des auf der Karte angezeigten Goodies bewegen.

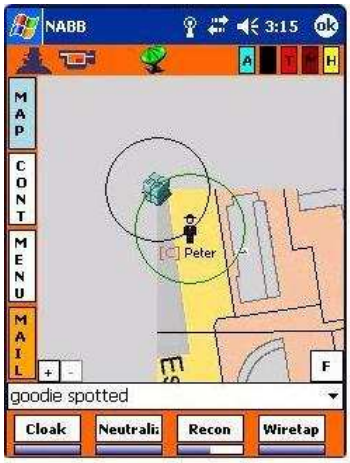


Abbildung 17.7: Goodie

17.3.2 Missionsziele

Die Hauptaufgabe des *Commanders* liegt darin, Missionen zu erfüllen. Diese Missionen bestehen aus dem Knacken eines virtuellen Safes. Wenn der *Commander* den Ort der Mission erreicht hat, wird ihm eine Aufgabe gestellt. Er hat drei Versuche, diese Aufgabe zu lösen. Schafft er dies nicht, so gilt die Mission als nicht erfüllt. Über den Erfolg oder Fehlschlag werden alle Spieler benachrichtigt.

Abbildung 17.8: Mitteilung über Erfolg oder Misserfolg

17.3.3 Nachrichten

Jeder Spieler kann, sofern er aktiv ist, Nachrichten schreiben, um seinen Mitspielern Informationen zukommen zu lassen. Diese Nachrichten können sowohl an einzelne Spieler des eigenen Teams als auch an das komplette eigene Team gesendet werden (Abb. 17.18).

Abbildung 17.9: Missionsziel

Der Spieler empfängt zusätzlich Nachrichten vom Server. In diesen stehen wichtige Mitteilungen über den Spielverlauf.

17.3.4 Spielende

Das Spielende ist erreicht, wenn der *Commander* neutralisiert wurde oder die Lösung zu einem Missionsziels vom *Commander* dreimal falsch eingegeben wurde. In diesen Fällen hat das Team der *Defender* gewonnen. Wenn alle Missionsziele erfüllt wurden ist das Team der *Infiltratoren* der Sieger. Ist das Spiel beendet, so färbt sich der Bildschirm gelb (17.10).

Abbildung 17.10: Spielende

17.3.5 Funktionalitäten des Clients

In diesem Abschnitt wird anhand von Screenshots erklärt, wie die Spieler einzelne Funktionen des Spiels einsetzen können. Auf dem Bildschirm befinden sich auf der linken Seite *tabs* (Abb. 17.11, Punkt 1). Anhand dieser werden die Funktionalitäten erläutert. Sollte etwas wichtiges passieren, sie erhalten z.B. eine Nachricht, dann leuchtet der ensprechende *Tab* auf (Abb. 17.11, Punkt 1).

Abbildung 17.11: Menü-Tabs

In jeder Ansicht ist die obere Menüleiste (Abb. refClientgestartet2, Punkt 1) zu sehen. Wenn das linke Symbol aufleuchtet, so befindet sich ein Gegner in ihrer Nähe. Leuchtet die Kamera auf, so befindet Sie sich im Aufklärungsradius eine gegnerischen Kamera. Die Satellitennschüssel zeigt, ob Sie mit dem Server verbunden sind (grün), oder ob die Verbindung nicht mehr besteht (rot). Die Tabs (Abb. 17.12, Punkt 3) sind ebenfalls immer zu sehen. Die Erklärung hierzu folgen im Anschluss. Der untere Bereich des Bildschirms ist ebenfalls immer präsent. Hier werden aktuelle Nachrichten angezeigt. Punkt 7 in Abbildung 17.12 zeigt den Status ihrer Fähigkeiten an. Desto mehr die Balken gefüllt sind, desto höher der Ladestatus der Fähigkeiten. Hier können Sie auch ihre Fähigkeiten durch einen Klick auf die jeweilige Fähigkeit benutzen .

17.3.6 Karte

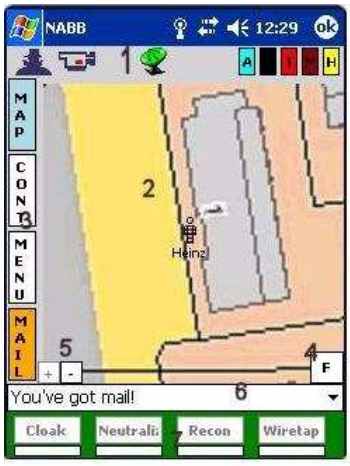


Abbildung 17.12: Kartendarstellung 2

Wenn Sie auf den Tab *Map* schalten sind folgende Funktionalitäten zu sehen:

- **Abb. 17.12, Punkt 2**: Hier ist die Karte zu sehen

- **Abb. 17.12, Punkt 4**: Mit diesem Button können Sie die Karte um sich herum zentrieren, anschließend bleiben Sie, solange die Funktion aktiviert ist, zentriert
- **Abb. 17.12, Punkt 5**: Mit diesen Buttons können Sie die Karte zoomen

17.3.7 Kontext

Der *Tab Cont* unterteilt sich in zwei Bereiche. Diese können durch die Reiter (17.13, Punkt 2) angesteuert werden.

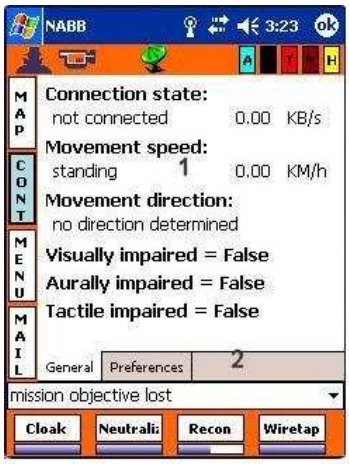


Abbildung 17.13: Kontext 1

Ist der Reiter *Genaral* aktiviert, so sind folgende Informationen erhältlich(17.13, Punkt 1):

- **Connection state**: gibt den Status der Verbindung an
- **Movement speed**: gibt die aktuelle Geschwigkeit an
- **Movement direction**: gibt die aktuelle Richtung an, in die Sie sich bewegen

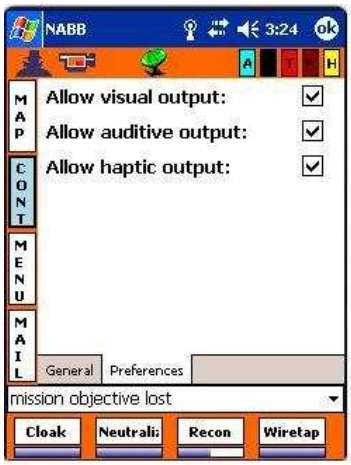


Abbildung 17.14: Kontext 2

Ist der Reiter *Preferences* aktiviert, so sind folgende Informationen erhältlich(17.13, Punkt 1):

- **Allow visual output**: hier können Sie entscheiden, ob Sie eine visuelle Ausgabe haben möchten
- **Allow auditive output**: hier können Sie entscheiden, ob Sie eine akkustische Ausgabe haben möchten
- **Allow haptic output**: hier können Sie entscheiden, ob Sie eine haptische Ausgabe haben möchten

17.3.8 Menü

Abbildung 17.15: Menü 1

Ist der *Tab Menü* aktiviert erhalten folgende Informationen, bzw. können folgende Funktionen ausführen:

- **Abb. 17.15, Punkt 1**: hier erhalten Sie Informationen über Verteilung ihrer Fähigkeitspunkte
- **Abb. 17.15, Punkt 2**: hier erhalten Sie Informationen über die Goodies, die Sie eingesammelt haben
- **Abb. 17.15, Punkt 3**: hier können Sie die eingesammelten Goodies aktivieren oder wieder ablegen

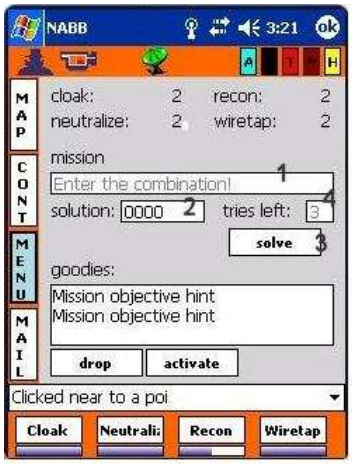


Abbildung 17.16: Menü 2

Befindet sich der *Commander* in der Nähe eines Missionsziels, so stehen ihm noch folgende Informationen zur Verfügung, bzw. folgende Funktionen können ausgeführt werden:

- **Abb. 17.15, Punkt 1**: in diesem Textfeld steht die Aufgabe des *Commanders*
- **Abb. 17.15, Punkt 2**: hier kann der Commander die Lösung angeben
- **Abb. 17.15, Punkt 3**: mit diesem Button übermittelt der *Commander* die Lösung zum Server
- **Abb. 17.15, Punkt 4**: in diesem Textfeld steht die Anzahl der verbliebenen Versuche

17.3.9 Nachrichten

Der *Tab Mail* gliedert sich in drei Unterbereiche, die durch Reiter (Abb. 17.17, Punkt 4) angesteuert werden können.

Abbildung 17.17: Nachrichtenfenster 1

Ist der Reiter *Inbox* aktiviert, so stehen ihnen folgende Funktionalitäten zur Verfügung:

- **Abb. 17.17, Punkt 1**: in dieser Box stehen alle empfangenen Nachrichten mit Sender und Betreff
- **Abb. 17.17, Punkt 2**: mit diesen Buttons können Sie ausgewählte Nachrichten löschen, weiterleiten oder auf die Nachricht antworten
- **Abb. 17.17, Punkt 3**: wenn Sie eine Nachricht in der oberen Box auswählen, können Sie hier die gesamte Nachricht lesen

Ist der Reiter *Compose* aktiviert, so können Sie Nachrichten schreiben und ihnen stehen folgende Funktionalitäten zur Verfügung:

- **Abb. 17.18, Punkt 1**: hier können Sie den Betreff der Nachricht eingeben
- **Abb. 17.18, Punkt 2**: mit dem Button können Sie den Betreff löschen
- **Abb. 17.18, Punkt 3**: hier können Sie den Nachrichtext eingeben
- **Abb. 17.18, Punkt 4**: mit dem Button können Sie die Nachricht senden

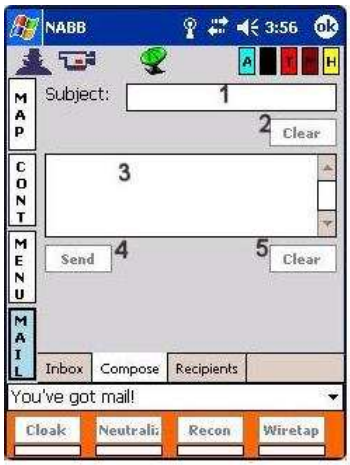


Abbildung 17.18: Nachrichtenfenster 2

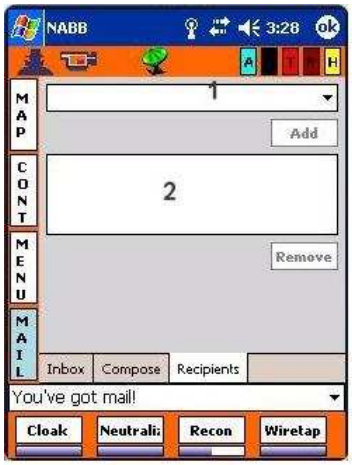


Abbildung 17.19: Nachrichtenfenster 3

- **Abb. 17.18, Punkt 5**: mit dem Button können Sie den Nachrichtentext löschen

Ist der Reiter *Recipients* aktiviert, so können Sie die Empfänger ihrer Nachricht hinzufügen und ihnen stehen folgende Funktionalitäten zur Verfügung:

- **Abb. 17.19, Punkt 1**: hier können Sie die Empfänger Ihrer Nachricht eingeben
- **Abb. 17.19, Punkt 2**: in dieser Box sehen Sie die Empfänger, an die Sie die Nachricht schicken möchten. Mit dem Button *remove* können Sie die markierten Empfänger wieder entfernen

Anhang A

Ergebnisdokumente der Vorbereitungsphase

A.1 Szenarien

Dieses Produkt entsteht im Rahmen der Projektgruppe „eXplorer“, die im Wintersemester 2004/2005 an der Universität Oldenburg im Department für Informatik angeboten wird. Aufgabenstellung ist die Entwicklung einer Informationsanwendung mit kontext-sensitiver Umgebungserkundung und mobiler, multimodaler Unterstützung.

Zu Beginn der Entwicklung einer Anwendung stehen die Ideenfindung und die Abstimmung von Meinungen zu den Entwicklungszielen der Projektgruppe. In den folgenden drei Abschnitten werden drei dieser Basisideen oder Szenarien in ihrer Rohfassung vorgestellt. Anschießend wird erläutert, für welches Szenario die Projektgruppe sich entschieden hat.

A.1.1 Szenario „Agentenspiel“

Susi, Kalle und Heiko sind auf dem Campus der Uni und wollen das Agentenspiel spielen. Sie starten ihre Operator (die unterstützende Anwendung auf dem mobilen Endgerät) und sehen, dass gerade eine neue Spielrunde beginnt. Außer ihnen loggen sich noch fünf weitere Spieler für die Runde ein. Jeder Spieler passt nun seine herausragenden Eigenschaften an. Je mehr Punkte er einer Eigenschaft zuweist, desto besser beherrscht er diese. Susi entscheidet sich für die vorgegebene „Werber“-Konfiguration, da sie vorrangig Teilnehmer der anderen Gruppe für die eigene Seite abwerben will. Da Heiko insbesondere Informationen über das gegnerische Team sammeln will, wählt er die „Hacker“-Konfiguration aus. Kalle stellt sich seine eigene Konfiguration zusammen, wobei er speziell der Aufklärungseigenschaft viele Punkte zuweist, um zuverlässigere Informationen über die Standorte der anderen Spieler zu erhalten.

Nachdem alle Spieler ihre Konfiguration ausgewählt haben, generiert der Computer auf dem Spielfeld verteilte Positionen und weist jedem Spieler eine dieser Positionen zu. Jeder Spieler erhält seine Fähigkeiten erst, wenn er den ihm zugewiesenen Punkt erreicht hat. Als Kalle seine Position erreicht, teilt ihm sein Operator mit, dass er in diesem Spiel die Zielperson „Mr. X“ sein wird. Susi wird seinem Team zugeteilt, Heiko dem Abwehrteam. „Mr. X“ wird damit beauftragt ein Unternehmen zu schädigen. Zum einen soll er sensible Daten aus einem Rechenzentrum entwenden, zum anderen im Büro des Chefs eine Abhöreinrichtung installieren. Das Abwehrteam weiß nur, dass „Mr. X“ in einer oder mehrerer von drei

angezeigten Zonen zuschlagen will. Kalle teilt Susi mit, dass er plant in das Rechenzentrum einzudringen und setzt sie auf Abwehragenten in dieser Region an. Susi folgt der Aufforderung und begibt sich in den Tarnmodus. So kann sie zwar selbst nicht auf die Satellitenortung zugreifen, kann aber auch selbst nicht geortet werden. Da Kalle nicht im Tarnmodus ist, kann er via Satellitenortung den Standort eines Abwehragenten in der Nähe des Rechenzentrums ausmachen und schickt eine entsprechende Nachricht an Susis Operator. Heikos Operator befindet sich gerade im Abhörmodus und fängt dank seiner „Hacker"-Konfiguration Kalles Nachricht an Susi ab. Als Heikos Warnung den gefährdeten Mitspieler erreicht, ist Susi bereits in Reichweite und wirbt letzteren ab. Da die Gegend nun gesichert ist, begibt sich Kalle zum Rechenzentrum und startet den Download der sensiblen Dateien.

Dies alarmiert alle Abwehragenten. Heiko weiß nun, dass Susi einen Gegner erfolgreich überzeugen konnte und markiert seinen ehem. Mitspieler als abgeworben. Nachdem das eine Ziel verloren ist, konzentrieren sich die Abwehragenten auf die verbleibenden zwei Zielregionen. Heiko schätzt nun eine mögliche Route, die „Mr. X" zu einer der Zielregionen nehmen könnte und legt sich dort auf die Lauer, indem er in den Tarnmodus schaltet. Mit dem Operator richtet er einen Ortungssatellit auf eine weitere mögliche Route aus, um dort getarnte Gegner entdecken zu können. Als Kalle diese Route passiert, bekommt Heiko dies mit und pirscht sich an ihn heran. Kalle ist völlig überrascht, als Heiko plötzlich hinter ihm steht und die Abwerbeprozedur initialisiert. Das System stellt fest, dass „Mr.X" enttarnt wurde und gibt das Abwehrteam als Sieger bekannt.

A.1.2 Szenario „Messe"

- **Kunde:**
  Der Messebesucher erhält seine elektronische Eintrittskarte auf sein mobiles Endgerät. Er speichert seine persönlichen Daten auf dieser Eintrittskarte. Die Anreise organisiert der Besucher über das „eXplorer"-System. Wenn er seine Route mit Start- und Zielpunkt und die gewünschte Reiseart ausgewählt hat, werden alle dafür notwendigen Dinge vom System übernommen (Tickets und Mietwagen buchen, usw.). Reist der Besucher einen Tag vor der Messe an, so kann er sich über die Messestadt informieren. Er wird z.B. zu Sehenswürdigkeiten oder Restaurants geführt. Reist der Besucher am Tag des Messebesuches mit dem eigenen PKW an, so wird er zu einem freien Parkplatz geführt.

  Am Eingang zum Messegelände checkt er mittels der elektronischen Eintrittskarte, die sich auf seinem mobilen Endgerät befindet, ein. Der Besucher kann nun zwischen dem Suchmodus und dem Stöbermodus wählen. Zunächst wählt er den Suchmodus. Seine Daten werden nun auch für Aussteller sichtbar und diese können dann dem Besucher gezielt Informationen zukommen lassen. Der Besucher wählt nun durch die Informationen die Aussteller aus, die ihn interessieren. Daraufhin wird die günstigste Route berechnet. In diese Kalkulation geht auch ein, ob an den Ständen der zu besuchenden Aussteller gerade großer Andrang herrscht. Ist diese der Fall wird dieser Stand auf das Ende der Route verlegt.

  Gegen Mittag bekommt der Besucher Hunger. Auf dem mobilen Endgerät wählt er den Menüpunkt „ESSEN". Nun werden auf dem Endgerät alle Restaurants und Imbissstände aufgeführt und die jeweiligen Speisekarten können eingesehen werden. Der Besucher entscheidet sich für ein Restaurant und wird dort hingeleitet. Wenn er beim Essen befindet, wechselt sein Status automatisch auf anonym, so kann es in Ruhe sein Mittagessen genießen.

  Nach dem Essen wird seine verbleibende Route auf Wunsch fortgesetzt. Sie wird von seinem derzeitigen Standpunkt ausgehend neu berechnet. Wenn er seinen Rundgang beendet hat, möchte er sich

noch ein bisschen zwanglos umsehen. Er wechselt in den anonym Stöbermodus (Schlendermodus). Er befindet sich in der Nähe einer Halle und wird über die Aussteller in der Halle in Stichworten informiert. Bei Interesse können nähere Informationen über die jeweiligen Aussteller abgerufen werden. Diese werden auch, wenn sich der Besucher in der Nähe des Standes befindet, auf dem mobilen Endgerät eingeblendet oder auf Wunsch akustisch ausgegeben. Die Abreise des Besuchers wird, wie die Anreise, vom „eXplorer“-System geregelt.

- **Mitarbeiter:**
  Mitarbeiter X der Firma Y betreut auf einer Messe einen Stand. Über das „eXplorer“ - System erhält er die Nachricht, das Kunde K die Halle soeben betreten hat. Dem Profil des Kunden kann er entnehmen, dass sich dieser für Produkte interessiert, die von Firma Y vertrieben werden. Er weist das „eXplorer“ - System an, Kunde Z eine Mitteilung über Firma Y und den Standort des Messestandes zuzuschicken.

  Kunde K schickt eine Nachricht zurück, in der er mitteilt gegen H Uhr am Stand einzutreffen, um sich über Produkt P der Firma Y zu unterhalten. Da es sich bei Kunde K um einen potenziellen Großkunden handelt, soll diesem eine besondere Betreuung zu teil werden. Zunächst mietet der Mitarbeiter X einen Seminarraum an und ordert zusätzlich einige Getränke. Als nächstes wird der Spezialist S für Produkt P der Firma Y für Zeitpunkt H zum angemieteten Seminarraum geschickt und ihm werden zusätzlich alle verfügbaren Informationen über Kunde K zugeschickt.

  Während der Spezialist S im Seminarraum ist, erkundigt sich ein anderer Kunde am Stand nach einem Produkt aus dem besonderen Wissensbereich des Spezialisten. Der Mitarbeiter, der währenddessen den Stand betreut, kann der Mitarbeiterkoordinationsfunktion entnehmen, dass der Spezialist frühestens in einer halben Stunde wieder am Stand sein kann und informiert den Kunden darüber. Daraufhin vereinbaren diese ein erneutes Treffen zu einem späteren Zeitpunkt.

  Ein langjähriger Geschäftspartner ist ebenfalls mit einem Stand auf der Messe vertreten. Mit dem „eXplorer“ System können beide in Kontakt treten und sich zum gemeinsamen Abendessen nach der Messe verabreden. Die Restaurantsuchfunktion hilft bei der Suche nach einem Restaurant, das den gemeinsamen Wünschen entspricht. Wird das Essen als Termin im gewählten Restaurant gespeichert, kann später die Route zum Restaurant angegeben werden.

A.1.3 Szenario „Einsatzkräfte“

Dieses Szenario geht von einer Katastrophe in einer urbanen Umgebung aus, bei der eine große Anzahl von unterschiedlichen Rettungskräften schnell und effektiv koordiniert werden müssen. Dabei soll die Kommunikation unter den Beteiligten und der Austausch von Daten erleichtert werden, möglichst unabhängig von den gegebenen Umständen.

Ausgangssituation Ein Airbus A380 stürzt in eine größere Chemiefabrik in der Nähe einer größeren Stadt. Durch die Explosion entsteht ein Feuer, aus dem giftige Gase entstehen. Das Feuer muss gelöscht, Verletzte geborgen und versorgt und Maßnahmen gegen mögliche Umweltverschmutzung ergriffen werden. Bewohner der Umgebung müssen in großem Rahmen evakuiert werden.

Ablauf

- Die werkseigene Feuerwehr löst Alarm aus und übermittelt für den Katastrophenfall vorbereitete Daten an die Feuerwehrleitstelle (Kartenmaterial) und beginnt mit einer ersten Bestandsaufnahme.

- Die Feuerwehrleitstelle löst Katastrophenalarm aus und alarmiert alle verfügbaren Hilfskräfte (THW, Polizei, Krankenhäuser etc.) Dabei werden die erhaltenen Informationen weitergeleitet.

- Das Kartenmaterial wird auf die mobilen Geräte der Feuerwehrleute gespielt.

- Die ersten Rettungskräfte treffen ein - Ein mobiles Netzwerk verbindet alle Einsatzkräfte vor Ort.

- Über dies Netzwerk erhalten die Rettungskräfte erste Daten der Bestandsaufnahme. (Betroffene Gebäude, Verletzte etc.)

- Eine Zentrale wird vor Ort eingerichtet und von dort werden Befehle erteilt.

- Weitere Kräfte treffen ein und erhalten, sobald sie in Reichweite sind, Daten wie z.B. den Sammelpunkt. Andersherum werden der Zentrale die Informationen über die Ankommenden mitgeteilt.

- Eine Mobile Sendeeinheit trifft mit ein. Diese erhöht die Reichweite des Netzwerks und verbessert die Eingabe und Anzeige von Daten in der Zentrale.

- Das Netzwerk wird plötzlich unterbrochen. Einige Rettungskräfte sind von der Zentrale abgeschnitten. Da sie aber untereinander in Reichweite sind, baut sich kurzfristig ein lokales Netz auf. Über dieses können sie Informationen weitergeben und abgleichen.

- Rettungskräfte setzen Markierer ein. Diese leuchten im Dunkeln oder im Rauch und zeigen sichere Wege an. Gleichzeitig sind sie passive Teilnehmer des Netzwerks und verkünden ihre Position. So können sie auch auf der Karte gesehen werden.

- Feuerwehrleute, die eine Pause machen, können schnell die Akkus ihrer mobilen Geräte tauschen. Dabei werden die Geräte erneut mit aktuellen Daten versorgt (falls diese noch nicht vorhanden waren) und alle gesammelten Daten werden übernommen.

- Um das Netzwerk zu verstärken, wird ein mobiler Einsatzwagen angefordert. Dieser stellt eine mobile Einsatzzentrale dar und verfügt über eine hohe Sendeleistung. So können wieder alle Rettungskräfte angeschlossen werden.

- Über den Einsatzwagen läuft die weitere Kommunikation nach außen, so können Maßnahmen zur Evakuierung ergriffen werden oder schon Informationen an Krankenhäuser über z.B. Art der Verletzungen weitergereicht werden.

A.1.4 Entscheidung

Nach ausführlicher Diskussion unter den Teilnehmern der Projektgruppe ist durch einen demokratischen und offenen Abstimmungsvorgang die Umsetzung des Szenarios „Agentenspiel“ beschlossen worden. Gegen das „Messe“-Szenario spricht vor allem die mangelnde Innovationskraft. Das Szenario „Einsatzkräfte“ wird zwar als sehr interessant und sogar für die Realität als nutzbar angesehen, allerdings ist die Projektgruppe der Meinung, dass ihr zur der Umsetzung dieses Szenarios die technischen und räumlichen Ressourcen fehlen. Das Agentenspiel erscheint der Projektgruppe als sinnvoller Kompromiss zwischen Machbarkeit und kreativen Potential.

A.2 Produktskizze

Dieses Produkt entsteht im Rahmen der Projektgruppe „eXplorer“, die im Wintersemester 2004/2005 an der Universität Oldenburg im Department für Informatik angeboten wird. Aufgabenstellung ist die Entwicklung einer Informationsanwendung mit kontext-sensitiver Umgebungserkundung und mobiler, multimodaler Unterstützung.

A.2.1 Problemdefinition, Projektziele

Dieses Dokument skizziert die wesentlichen Eigenschaften des Agentenspiels. Es treten zwei konkurrierende Gruppen mit unterschiedlichen Aufgabenstellungen gegeneinander an. Im Infiltrationsteam muss der „Commander“ bestimmte Missionsziele erfüllen und wird dabei von seinen Teamkollegen unterstützt. Das Abwehrteam versucht ihn am Erreichen dieser Missionsziele zu hindern. Das Infiltrationsteam gewinnt, wenn alle Missionsziele erfüllt wurden. Gelingt es dem Abwehrteam den „Commander“ zu fangen, gehen sie als Sieger aus dem Spiel hervor. Zum Erreichen der Spielziele stehen den Spielern verschiedene Funktionen zur Verfügung, wie z.B. das Abwerben von Spielern des gegnerischen Teams.

Als Spielfeld dient die reale Welt. Die Spieler können sich innerhalb einer vorgegebenen Region frei bewegen. Die Teilnahme an dem Spiel ist jedoch nur mittels mobilen Endgeräten möglich. Diese stehen mit einem Server in Verbindung, der die Schiedsrichterfunktion übernimmt. Über die mobilen Endgeräte erhalten die Spieler Informationen über den Status des Spiels und können diesen selbst aktiv beeinflussen.

A.2.2 Entwicklungs-, Einsatz-, Wartungsumgebung, Schnittstellen und Nebenbedingungen

Der Projektgruppe stehen das Medienlabor des OFFIS[1] und diverse mobile Endgeräte mit GPS-Empfänger als Entwicklungsumgebung zur Verfügung. Angestrebt wird eine Entwicklung der Anwendung in der Programmiersprache „Java“, da diese Sprache prinzipiell Plattformunabhängigkeit gewährleistet und die Teammitglieder die größte Erfahrung im Umgang mit „Java“ haben.

A.2.3 Benutzerprofil

Der primäre Nutzer ist der Spieler, für den das Spiel eine Freizeitaktivität darstellt. Er erwartet eine simple und multimedial reichhaltige Interaktion. Einige dieser Spieler können mit besonderen Administratorrechten ausgestattet sein, die ihnen die Möglichkeit geben, Spieleinstellungen vorzugeben und zu verändern.

A.2.4 Funktionsumfang, Außenverhalten

A.2.4.1 Datenhaltung

Auf einem Server lagern die Daten zu einer Spielrunde (z.B. Spielkarte, Missionsziele, Rundendauer, ...). Während einer Spielrunde speichert dieser außerdem die Daten der teilnehmenden Spielcharaktere (z.B. Position, Eigenschaftspunkte, Gruppenzugehörigkeit, ...).

[1] Oldenburger Forschungs- und Entwicklungsinstitut für Informatik-Werkzeuge und Systeme

A.2.4.2 Benutzungsoberfläche

Die Benutzungsoberfläche ist durch die Anzeigemöglichkeiten des mobilen Endgeräts beschränkt. Ein- und Ausgaben erfolgen multimodal. Die Anwendung bietet zwei Hauptansichten, die Übersichtskarte und die Charakter-/Funktionsübersicht. Die Übersichtskarte stellt die Spielgegend, die Missionsziele und die Standorte der teilnehmenden Spieler wahlweise visuell, akustisch und/oder haptisch dar. Die Charakter-/Funktionsübersicht bietet dem Spieler die Möglichkeit seine Charaktereigenschaften einzusehen, sowie seine Fähigkeiten einzusetzen.

A.2.4.3 Funktionalität

Auffinden von Spielservern Das System erlaubt eine Suche nach sich in der Region befindenden Spielservern. Der Spieler kann sich anhand angezeigter Server-Metainformationen für einen Server entscheiden.

Erstellung eines Spielcharakters Der Spieler kann seinen Charakter durch Punkteverteilung auf folgende Eigenschaften zusammenstellen:

- **Tarnen:** Je höher dieser Wert, desto ungenauer erscheint die Position des Spielers auf der Übersichtskarte des gegnerischen Teams.
- **Abwerben:** Je höher dieser Wert, desto erfolgversprechender und schneller verläuft eine Abwerbeprozedur.
- **Abhören:** Je höher dieser Wert, desto wahrscheinlicher können Nachrichten des gegnerischen Teams, auch über größere Entfernung, abgefangen werden.
- **Überprüfen:** Je höher dieser Wert, desto erfolgversprechender können Spieler auf ihre Teamzugehörigkeit geprüft werden.
- **Aufklären:** Je höher dieser Wert, desto genauer werden die Positionen der Mitspieler auf der Übersichtskarte angezeigt.

Versenden von Nachrichten Der Spieler kann Nachrichten sowohl an einzelne Spieler, als auch als Rundnachricht an das eigene Team oder alle teilnehmenden Spieler senden.

Übersichtskarte Die Positionen aller Spieler werden in bestimmten Zeitintervallen aktualisiert. Die Kartenansicht kann mittels folgender Funktionen gesteuert werden:

- **zoomen**
- **scrollen**
- **zentrieren**

Außerdem kann zu jedem beliebigen Standort die Distanz zur eigenen Position abgerufen werden.

Anwendung von Charakterfähgkeiten

- **Tarnen:** Verbirgt einen Spieler auf der Übersichtskarte vor dem gegnerischen Team, verhindert aber gleichzeitig, dass der getarnte Spieler die Positionen aller anderen Spieler angezeigt bekommt.
- **Abwerben:** Wirbt einen gegnerischen Spieler, der sich innerhalb einer bestimmten Reichweite befindet, für das eigene Team ab.
- **Abhören:** Erlaubt einem Spieler zwischen gegnerischen Teammitgliedern gesendete Nachrichten abzufangen und eventuell zu manipulieren.
- **Überprüfen:** Ein Spieler kann mittels dieser Fähigkeit die Teamzugehörigkeit eines Mitspielers validieren.
- **Aufklären:** Die Positionsanzeigen in einer begrenzten Region können temporär präzisiert, sowie getarnte Spieler sichtbar gemacht werden.
- **Missionsziel erfüllen:** Diese Fähigkeit ist abhängig von den Spieleinstellungen und steht nur einer einzigen Person, dem „Commander“ zur Verfügung. Befindet sich dieser Spieler in einem vorgegebenen Bereich der Übersichtskarte, so kann diese Spezialfähigkeit aktiviert werden.

A.2.5 Akzeptanzkriterien

Im Zuge der Entwicklung soll eine Anwendung mit kontextsensitiver Umgebungserkundung und mobiler, multimodaler Unterstützung realisiert werden. Eine gute Wartbarkeit wird durch einen modularen Aufbau der Anwendung garantiert. Die Anwendung soll ausreichend fehlerfrei laufen, um dessen Funktionsweise testen zu können.

A.2.6 Anmerkungen zur Produktskizze

Die Produktskizze ist zu Beginn des Projekts entstanden, daher sind bei der Realisierung des Agentenspiels verschiedenen Aspekte an die aktuellen Gegebenheiten angepasst worden. Im folgenden Abschnitt erfolgt eine Zusammenfassung der Abweichungen von Produktskizze und tatsächlicher Umsetzung.

Entwicklungs-, Einsatz-, Wartungsumgebung, Schnittstellen und Nebenbedingungen Basierend auf der Machbarkeitsstudie über die Kommunikation zwischen Java und C#, siehe dazu Seite 333, ist das Agentenspiel in C# statt in Java geschrieben worden. Die Ähnlichkeit zwischen Java und C# hat dies erleichtert. Die momentan nicht vorhandene Plattformunabhängigkeit von C# fällt nicht negativ ins Gewicht, da auf allen zur Verfügung stehenden mobilen Endgeräten die .NET CF-Plattform implementiert ist.

Benutzerprofil Anstelle einer Verteilung von Administratorrechten auf die Spieler verbleiben diese zentral beim Server. Diese Vorgehensweise wurde gewählt, da sie leichter zu implementieren ist und der Server sich natürlicherweise als Administratorplattform eignet.

Datenhaltung Die zeitliche Begrenzung einer Spielrunde, sowie die zentrale Datenhaltung auf dem Server wurden verworfen. Während des Testens des Spielablaufs wurde deutlich, dass eine zeitliche Begrenzung der Spieldauer für den Spielspaß keinen Mehrwert darstellt. Die zentrale Datenhaltung wurde zur Begrenzung des Übertragungsvolumens zwischen den Clients und dem Server aufgegeben, da ein höheres Datenaufkommen im Mobilfunkbereich auch höhere Kosten verursacht.

Benutzungsoberfläche Die Möglichkeit der Eingabe wurde auf eine Modalität (Haptische Eingabe) beschränkt, vgl. hierzu Abschnitt 13.5.3 auf Seite 301.

Auffinden von Spielservern Aufgrund der Zeitvorgaben und der begrenzten Ressourcen kam der Aufbau eines komplexen Servernetzes im Rahmen dieser Projektgruppe nicht in Frage.

Erstellung eines Spielcharakters und Anwendung von Charakterfähigkeiten Die zur Verfügung stehenden Fähigkeiten wurden im Verlauf der Entwicklung des Projekts mehrfach angepasst, um die Spielbarkeit des Agentenspiels zu verbessern. Eine aktualisierte Beschreibung der Fähigkeiten ist im Fachlichen Modell auf Seite 251 zu finden.

Anhang B

Grobkonzept

B.1 Einleitung

Im Rahmen dieses Arbeitspaketes wird eine Fallstudie erstellt, in der eine prototypische Umsetzung der Anforderungsdefinition stattfindet. Sie soll helfen, die Technologien auszumachen, die von zentraler Bedeutung für die Umsetzung sind und daher in einer Technologiestudie betrachtet werden müssen. Dieses Dokument ist als Ideensammlung gedacht. Im Rahmen des Arbeitspaketes entwickelte brauchbare Konzepte sollen im späteren Verlauf als Vorlage für die Erarbeitung des Entwurfsdokuments, die Entwicklung der Systemarchitektur und der Verfeinerung der Programmmodule, verwendet werden. Die Fallstudie stellt jedoch keine verbindliche Vorlage für den Entwurf dar.

Abschnitt B.2: In diesem Abschnitt wird einleitend eine graphische Übersicht über die Module gegeben, die für die Realisierung des Agentenspiels notwendig erscheinen. Im weiteren Verlauf dieses Kapitels werden die Funktionen und Einsatzzwecke der Module kurz erläutert und umgangssprachlich beschrieben. Es werden Überlegungen angestellt, welche Schnittstellen die Module untereinander haben und auf welche Besonderheiten (Hardwarevoraussetzungen, potentielle Fehlerquellen, etc.) zu achten sind.

Abschnitt B.3: Ausgehend von dieser Grobstruktur des Agentenspiels werden in diesem Abschnitt diejenigen Module identifiziert, die Bestandteile des Frameworks für die Entwicklung kontextsensitiver, umgebungserkundender, multimodaler und mobiler Software werden können. Die Beschreibungen zu den Modulen der Frameworkstruktur sollen, wie bei den Beschreibungen zum Agentenspiel, einen Überblick über die Funktionen und Aufgaben dieser Einheiten geben.

Basierend auf den Anforderungen an die Leistungen der Module werden die Technologien identifiziert, die als geeignet für die Entwicklung und Realisierung der Funktionalitäten des Frameworks und des Agentenspiels erscheinen. Die Untersuchung dieser Technologien wird Gegenstand des folgenden Arbeitspakets „Technologiestudie“ (AP3 / siehe Kapitel 12 auf Seite 193) sein.

Abschnitt B.4: Im vierten Kapitel werden Daten zusammengetragen, die während des Spielverlaufs gehalten und organisiert werden müssen. Diesbezüglich werden zwei grundsätzliche Architekturansätze zur Datenhaltung vorgestellt, die **Client/Server-Architektur** und der Architekturansatz einer **verteilten Datenhaltung**.

Abschnitt B.5: In den ersten drei Kapiteln wurden die Vorbereitungen auf die Entwurfsphase getroffen, indem die Module identifiziert wurden, die die funktionalen Anforderungen des Spiels erfüllen sollen. In den folgenden Kapiteln soll überprüft werden, ob dieses Konzept zur Erfüllung der nichtfunktionalen Anforderungen an das Agentenspiel geeignet ist. Dazu werden in diesem Abschnitt Überlegungen angestellt, ob sich die Spielidee mit dem Grobkonzept und dem Framework umsetzen lässt.

Abschnitt B.6: Diese Abschnitt beschäftigt sich mit dem Kontextmodell.

Abschnitt B.7: In diesem Kapitel wird das Interaktionsdesign beschrieben. Es geht hierbei ebenso wie in dem folgenden Abschnitt darum, wie die notwendigen Spielinformationen für einen Mitspieler darzustellen sind.

Abschnitt B.8: Dieser Abschnitt befasst sich mit dem Informationsdesign.

B.2 Modulbeschreibung des Agentenspiels

In dieser Teilaufgabe sollen die grundlegenen Module des Agentenspiels identifiziert werden, die zur Umsetzung der Anforderungen benötigt werden. Die Aufgaben, die sie erfüllen sollen, werden umgangssprachlich beschrieben. Es werden verschiedene Vorschläge ausgearbeitet, anhand derer dann ein möglichst optimales Basiskonzept erarbeitet werden soll. Abb. B.1 zeigt die identifizierten Module in einer Übersicht. Die folgenden Abschnitte beschreiben die Module im Einzelnen.

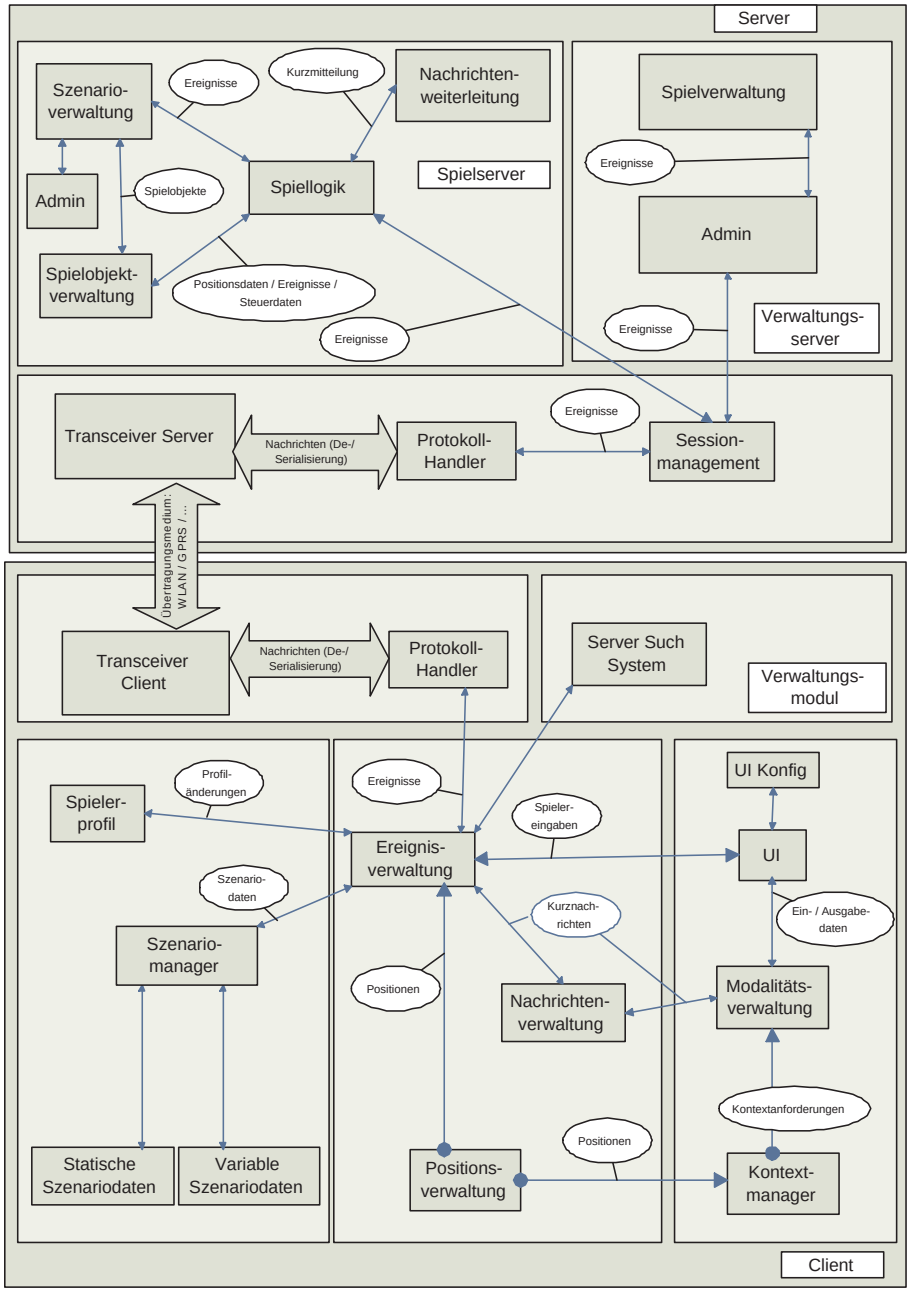


Abbildung B.1: Modulübersicht des Agentenspiels

B.2.1 Logikmodul

B.2.1.1 Positionsverwaltung

Funktionsbeschreibung Das Positionsverwaltungsmodul ermittelt unter Zuhilfenahme geeigneter Hardware die aktuelle Position des mobilen Endgerätes. Je nach eingesetzter Technik kann die Positionsinformation aus Koordinaten oder einer Zellen-ID (Positionsbestimmung durch Abstandmessung zu Funkzellen) bestehen. Geeignete Hardware könnte ein GPS-Modul oder ein WLAN-Modul sein. GPS-Module liefern die aktuelle Position auf Anfrage in Form von absoluten Koordinaten. Wird die Position über ein WLAN-Modul berechnet, müssen Daten wie Sender-ID, Signallaufzeiten oder Feldstärken berücksichtigt werden.

Vorbedingung Voraussetzung für die korrekte Funktion des Moduls ist das Vorhandensein geeigneter Hardware zur Positionsbestimmung. Bei Verwendung von WLAN müssen zusätzlich Daten über die genauen Standpunkte der Accesspoints sowie evtl. Feldstärkedaten zur Verfügung gestellt werden.

Datenbasis Bei Verwendung von GPS werden keine zusätzlichen Daten benötigt, da die Positionsdaten direkt von der Hardware geliefert werden und an das anfragende Modul weiterreichen lassen. Bei Positionsbestimmung über ein WLAN-Modul muss auf Daten, wie die IDs und Positionsdaten der Accesspoints, sowie die Empfangsstärken der Accesspoint an unterschiedlichen Positionen, zugegriffen werden.

Software-Schnittstellen Das Positionsverwaltungsmodul stellt eine Funktion zu Verfügung, die die aktuelle Position zurückliefert.

Hardware-Schnittstellen Eingabegeräte können je nach verwendeter Methode der Positionsbestimmung ein GPS-Gerät oder ein WLAN-Modul sein.

Fehler Fehler können bei Verwendung des Positionsverwaltungsmodul auftreten, falls die Positionsbestimmung z.B. aufgrund ungünstiger Umweltbedingungen oder eines Hardwareausfalls nicht möglich ist. Tritt ein solcher Fehler auf, muss die zuletzt ermittelte Position zur Verfügung gestellt und der Anwender über den Abbruch der Positionsbestimmung informiert werden.

B.2.1.2 Nachrichtenverwaltung

Funktionsbeschreibung Das Nachrichtenverwaltungsmodul verwaltet die vom System und von Spielern gesendeten und empfangenen Nachrichten. Hierfür bietet das Nachrichtenverwaltungsmodul über das Userinterface-Modul eine Schnittstelle an, über das der Anwender empfangene Nachrichten lesen und löschen, sowie neue Nachrichten erstellen und versenden kann.

Datenbasis Empfangene und versendete Nachrichten müssen auf dem mobilen Endgerät gespeichert werden.

Softwareschnittstelle Das Nachrichtenverwaltungsmodul stellt Funktionen zur Verfügung, die das Senden und Empfangen sowie das Verwalten von Nachrichten ermöglichen.

Hardwareschnittstelle Die Hardwareschnittstelle für die Ein- und Ausgabe von Nachrichten wird von dem eingesetzten mobilen Endgerät zur Verfügung gestellt.

B.2.1.3 Ereignisverwaltung

Funktionsbeschreibung Das Ereignisverwaltungsmodul koordiniert und verwaltet alle Ereignisse auf dem Client. Trift ein Ereignis über den „Transceiver Client"- und „Protokoll-Handler"-Modul ein oder wird ein Ereignis durch den Anwender ausgelöst, leitet das Ereignisverwaltungsmodul das Ereignis and das Modul weiter, das für die Verarbeitung des Ereignisses verantwortlich ist.

Vorbedingung Das Ereignisverwaltungsmodul stellt keine Vorbedingungen. Es organisiert den Aufruf der Module abhängig von den eintreffenden Ereignissen.

Software-Schnittstellen Als zentrales Modul kommuniziert das Ereignisverwaltungsmodul mit folgenden Modulen auf dem Client:

- Datenmodul
- Verwaltungsmodul
- Ausgabemodul
- Kommunikationsmodul

B.2.2 Datenmodule

B.2.2.1 Spielerprofil

Funktionsbeschreibung Das Spielerprofilmodul ist verantwortlich für die Speicherung aller Profilbezogenen Daten und deren Aktualisierung.

Vorbedingung Der Spieler gibt zu Beginn des Spieles seine Daten korrekt ein, d. h. die Charaktererstellung ist erfolgreich verlaufen.

Datenbasis Folgende Daten werden gespeichert:

- Der Alias des Spielers (Initial vom Spieler eingegeben).
- Die Merkmalsausprägungen der Attribute (Initial vom Spieler eingegeben + Effekte im Spiel, wie bspw. Goodies).
- Status des Spielers, also ob er getarnt oder neutralisiert ist und ob er sich im Abhörmodus befindet.
- Die Teamzugehörigkeit.

Software-Schnittstellen Das Spielerprofilmodul benötigt selbst keine Schnittstellen, muss aber seinen Daten zugänglich machen und eine Schnittstelle zur Aktualisierung dieser anbieten.

Hardware-Schnittstellen Das Spielerprofil benötigt ausschließlich herkömmlichen Arbeitsspeicher für seine Daten.

Fehler Im Spielerprofil selbst sollten keine Fehler auftreten können (funktionierende Hardware vorausgesetzt), nur durch Setzen eines falschen Wertes können nicht korrekte Daten gespeichert werden. Darauf hat diese Modul aber keinen Einfluss.

B.2.2.2 Variable Szenariodaten

Funktionsbeschreibung Hier werden alle im Spiel dynamisch anfallen Daten (Bilder, Klangdateien, etc.) gespeichert und zur schnellen Verfügbarkeit und Einsparung von Übertragungbandbreite zwischengespeichert.

Vorbedingung Ein Spiel wurde korrekt initialisiert und es liegen keine Übertragungfehler vor.

Datenbasis Eine abstakte Datenklasse stellt die Basis für die Speicherung der unterschiedlichen binären Formate da. Die konkreten Datenklassen enthalten Metainformationen über das Objekt, wie bspw. den Typ, und alle konkreten Datenobjekte (Bilder, Klangdatein, ...) bekommen eine im Spiel eindeutige Nummer zugewiesen.

Software-Schnittstellen Der Cache für dynamische Daten benötigt eine Schnittstelle zur Übertagung von Daten. Er hält seine Daten abrufbar.

Hardware-Schnittstellen Der Cache für dynamische Daten benötigt herkömmlichen Arbeitsspeicher für seine Daten und eine physikalische Verbindung zum Server, um Daten nachzufordern.

Fehler Bei dem dynamischen Nachladen kann es zu Übertragungsfehlern und zu Verbindungsabbrüchen kommen.

B.2.2.3 Statische Szenariodaten

Funktionsbeschreibung Dieses Modul speichert alle Daten, die zu Beginn des Spieles bekannt sind. Dazu gehören die Karte, Texte der Hilfe, verschiedene Spielobjekte wie Alarmanlage oder auch der Safe, usw. Diese Daten können vor dem Spiel geladen oder auch zum Startzeitpunkt nachgeladen werden.

Vorbedingung Die Übertragung der Daten verlief erfolgreich.

Datenbasis Eine abstakte Datenklasse stellt die Basis für die Speicherung der unterschiedlichen binären Formate da. Die konkreten Datenklassen enthalten Metainformationen über das Objekt, wie bspw. den Typ.

Software-Schnittstellen Der Szenariomanager für statische Daten benötigt eine Schnittstelle zur Übertagung von Daten. Er hält seine Daten abrufbar.

Hardware-Schnittstellen Der Szenariomanager für statische Daten benötigt herkömmlichen Arbeitsspeicher für seine Daten und eine physikalische Verbindung zum Server, um Daten nachzufordern.

Fehler Es kann zu Übertragungsfehler kommen.

B.2.3 Ein-/Ausgabemodule

B.2.3.1 User Interface

Funktionsbeschreibung Das User Interface (UI) gibt die benutzerrelevanten Daten des Spiels aus. Diese Ausgabe kann sowohl visuell als auch auditiv (und evtl. haptisch) erfolgen. Ausserdem nimmt das User Interface Benutzereingaben auf und leitet diese an die Ereignisverwaltung weiter.

Vorbedingung Das Modul User Interface muss bei der Ereignisverwaltung als EventListener registriert sein, um Ereignisse zu empfangen. Zur Anzeige der benutzerrelevanten Daten braucht das User Interface eben diese. Ausserdem muss der Benutzer eine Aktion ausgeführt haben, bevor diese an die Ereignisverwaltung weitergeleitet werden kann.

Datenbasis Als Daten hat das Modul die vom Benutzer auf dem User Interface ausgeführte Aktion.

Software-Schnittstellen Das User Interface kommuniziert zum Einen über Ereignisse mit dem System, um Benutzereingaben an die Ereignisverwaltung weiterzuleiten und zum Anderen über eine Schnittestelle zur Modalitätsverwaltung um die Ausgabedaten zu empfangen.

Hardware-Schnittstellen Das Navigationsmodul stellt das Spielgebiet und die aktuellen Positionen aller (für den jeweiligen Benutzer) sichtbaren Spielobjekte mittels einer 2D-Karte dar.

B.2.3.2 User Interface-Konfiguration

Funktionsbeschreibung Das Modul UI-Konfiguration ermöglicht dem Benutzer die individuelle Anpassung der Benutzerschnittstelle. Grafische Ausgabeelemente, wie Buttongröße, Schriftgröße und -farbe sollen ebenso verändert werden können wie die Zuordnung von Klängen zu Ereignissen. Ausserdem soll die Belegung der Menüleiste mit Funktionen frei konfigurierbar sein.

Datenbasis Die Konfiguration der Benutzeroberfläche wird in einer Datei gespeichert.

B.2.3.3 Kontextmanager

Funktionsbeschreibung Der Kontextmanager ermittelt aus den Positionsdaten des Spielers, dessen Profil und seinen Präferenzen die Kontextanforderungen an das User Interface. Dazu speichert der Kontextmanager die aktuellen und die zuletzt von der Ereignisverwaltung übermittelten Positionsdaten des Spielers.

Vorbedingung Der Kontextmanager ist bei der Positionsverwaltung als EventListener registriert.

Datenbasis Der Kontextmanager speichert die aktuellen und die zuletzt von der Positionsverwaltung übermittelten Positionsdaten des Spielers.

Software-Schnittstellen Dieses Modul besitzt eine Schnittstelle zur Ereignisverwaltung, über die es die von der Positionverwaltung übermittelten Positionsdaten erhält, wann immer sich diese ändern. Ausserdem sendet der Kontextmanager die berechneten Kontextinformationen an die Modalitätsverwaltung.

Fehler Fehler in der Kontextberechnung können dann entstehen, wenn die Positionsdaten (aktuell und/oder zuletzt übermittelt) fehlerhaft sind.

B.2.3.4 Modalitätsverwaltung

Funktionsbeschreibung Die Modalitätsverwaltung empfängt von der Ereignisverwaltung die Ergebnisse der Spiellogikberechnung, die vom User Interface dargestellt werden sollen. Anhand der vom Kontextmanager berechneten Kontextanforderungen ermittelt die Modalitätsverwaltung die passende(n) Ausgabemodalität(en) dieser Ergebnisse.

Vorbedingung Das Modul muss bei der Ereignisverwaltung als EventListener registriert sein, um Ereignisse zu empfangen. Ausserdem müssen zur Modalitätsbestimmung gültige Kontextanforderungen vorliegen.

Software-Schnittstellen Die Modalitätsverwaltung empfängt von der Ereignisverwaltung die vom User Interface auszugebenden Daten, von der Nachrichtenverwaltung die auszugebenden Nachrichten, von der Kontextsensitiven Hilfe eben diese und vom Kontextmanager die Kontextanforderungen.

Fehler Fehler in der Modalitätsverwaltung können dann entstehen, wenn die Kontextanforderungen fehlerhaft sind.

B.2.4 Client/Server Kommunikation

B.2.4.1 Protokoll Handler

Funktionsbeschreibung Der Protokoll Handler ist für die Kodierung des Protokolls verantwortlich. Er übersetzt Ereignisse in Befehl-Strings des Protokolls und zurück.

Vorbedingung Ein Transceiver wurde für die Kommunikation ausgewählt.

Datenbasis Datenbasis ist das Protokoll selber. Ereignisse müssen Befehlen zugeordnet werden.

Software-Schnittstellen Der Protokoll Handler erhält und sendet Protokolldaten über das „Tranceiver“-Modul. Serverseitig existiert eine Schnittstelle zum „Session-Management“-Modul, auf dem Client direkt eine zur Ereignisverwaltung.

Fehler Das Protokoll sollte bis zu einem bestimmten Grad Fehlertolerant ausgelegt sein. Stimmen Daten nicht mit den erwarteten überein, wird die Kommunikation wiederholt. Erst bei mehreren Fehlschlägen wird ein Fehler ausgelöst.

B.2.4.2 Transceiver

Funktionsbeschreibung Die „Transceiver"-Module übernehmen die Übertragung der Daten transparent. Dabei kann es für jede Übertragungstechnik ein anderes, spezielles Modul geben. Entsprechend der gewählten Übertragungstechnik wird das Modul ausgewählt und eingesetzt.

Vorbedingung Übertragungstechnik und Transceiver wurden gewählt.

Software-Schnittstellen Transceiver kommunizieren mit dem „Protokoll-Handler"-Modul und den Treibern der jeweiligen Übertragungstechnik.

Fehler Anhaltende Fehler sollten als Verbindungsabbruch gewertet werden.

B.2.4.3 Sessionmanagement

Funktionsbeschreibung Das „Session Management"-Modul begleitet den Lebenszyklus der Kommunikation mit allen Clients. Über dieses Modul können alle Clients vom Server identifiziert und angesprochen werden. Das Session Management kümmert sich auch um das Einloggen neuer Clients sowie um Aufräumungsarbeiten bei einem Verbindungsabbau oder eines Timeouts.

Vorbedingung Der Server muss funktionstüchtig sein und Verbindungen entgegennehmen können.

Fehler Bei anhaltenden Fehlern sollte die Session eines Client beendet werden.

B.2.5 Spielserver

B.2.5.1 Spiellogik

Funktionsbeschreibung Die Spiellogik verbindet die anderen Module des Spielservers zu einem System, indem sie die auftretenden Ereignisse und Nachrichten weiterleitet.

Datenbasis Die Ereignisverwaltung speichert Referenzen auf die anderen Module, so dass sie die Ereignisse weiterreichen kann.

Software-Schnittstellen Die Ereignisverwaltung bietet Schnittstellen zur Verwaltung von Event-Listenern an (Registrierung, Deregistrierung) und empfängt auftretende Ereignisse.

B.2.5.2 Spielobjektverwaltung

Funktionsbeschreibung Diese Modul hält alle Spielobjekte vor. Diese werden über auftretende Ereignisse informiert (meist die Positionsänderung eines Spielers) und können dadurch eigene Ereignisse auslösen.

Vorbedingung Die Spielobjektverwaltung ist bei der Ereignisverwaltung als EventListener registriert.

Datenbasis

- **Missionsziele**: Die Missionsziele werden durch Positionsänderungen des Spielers ausgelöst, wenn ein Spieler dabei in eine bestimmte Reichweite kommt. Das Missionsziel kann dann entweder das Ereignis auslösen, dass es erfüllt sei, oder es löst ein Ereignis zum Stellen eines Rätsels aus. Die vom Spieler eingegebene Lösung wird dann ebenfalls als Ereignis von den Missionszielobjekten aufgenommen.
- **Goodies**: Goodies reagieren wie Missionsziele auf Positionsänderungen. Sie lösen Ereignisse aus, die dann z.B. die Fähigkeiten des Spielers verändern.
- **Kameras**: Kameras reagieren ebenfalls auf Positionsänderungen und teilen dann ihrem Besitzer die Sichtung andere Spieler mittels Ereignisse mit.
- **Neutralisationskreise**: spielen eine Sonderrolle, da sie während ihres Bestehens selbst ständig ihren Wirkungsradius ändern.
- **Spieler**: Sind die umfangreichsten Spielobjekte, da hier auf eine Menge an Veränderungen die Spieldaten reagiert werden muss, z.B. die Positionen der anderen Spieler, das Einsetzen von Fähigkeiten oder eine Neutralisation selbiger.
- **Startpunkt**: Startpunkte reagieren auf die Positionsänderung eines bestimmten Spielers, indem sie ihm seine Fähigkeiten zurückgeben, wenn er in Reichweite kommt und dann selbst verschwinden.

Software-Schnittstellen Die Spielobjektverwaltung kommuniziert über Ereignisse mit dem System und bietet der Szenarioverwaltung die Möglichkeit, neue Spielobjekte hinzuzufügen.

Fehler Fehler können durch unbekannte Ereignisse entstehen. Diese werden von den Spielobjekten ignoriert.

B.2.5.3 Szenarioverwaltung

Funktionsbeschreibung Die Szenarioverwaltung hält alle Daten über das Szenario vor und erzeugt daraus die Spielobjekte wie Missionsziele, Startpunkte und Goodies. Sie überwacht außerdem die Rahmenbedingungen des Szenarios, wie z.B. die Erfüllung der Missionsziele in der korrekten Reihenfolge, oder ob sich Spieler dem Rand des Spielfelds nähern.

Vorbedingung Szenarioverwaltung ist bei der Ereignisverwaltung als Event-Listener registriert.

Datenbasis Die Szenarioverwaltung hält alle statische Daten zu dem aktiven Szenario vor und weiß darüber hinaus, zu welchem Zeitpunkt weitere Spielobjekte erzeugt werden sollen. Außerdem wird der aktuelle Zustand des Szenarios gespeichert und überwacht.

Software-Schnittstellen Die Szenarioverwaltung kommuniziert über Ereignisse mit dem System und kann der Spielobjektverwaltung neue Spielobjekte hinzufügen.

Hardware-Schnittstellen Das Modul muss Zugriff auf einen persistenten Speicher haben, auf dem die statischen Szenariodaten bereitliegen

Fehler Fehler können auftreten, wenn die statischen Szenariodaten nicht korrekt sind. In dem Fall, sollte das Spiel gar nicht erst für Clients öffentlich werden, sondern dem Administrator eine Fehlermeldung angezeigt werden.

B.2.5.4 Nachrichtenweiterleitung

Funktionsbeschreibung Die Nachrichtenweiterleitung ließt versendete Kurznachrichten ein und leitet sie an alle berechtigten Empfänger weiter. Sie kann außerdem dafür verwendet werden, große Daten, wie z.B. Bilder zwischenzuspeichern und nur bei bedarf dem jeweiligen Spieler zur Verfügung zu stellen. In diesem Modul wird auch die Funktionalität „Abhören“ umgesetzt.

Vorbedingung Nachrichtenweiterleitung ist bei der Ereignisverwaltung als Event-Listener registriert.

Datenbasis Die Nachrichtenweiterleitung hält die Nachrichten selbst als Datum vor, solange sie nicht korrekt weitergeleitet wurde. Zu überlegen bleibt auch, ob Daten wie Bilder bis zum Abruf vorzuhalten, um den Datenverkehr während des Nachrichtenversendens zu minimieren. Dazu muss die Nachrichtenweiterleitung darüber in Kenntnis gesetzt sein, welcher Spieler zu welchem Team gehört und welcher Spieler seine Abhörfähigkeit aktiviert hat.

Software-Schnittstellen Dieses Modul kommuniziert mittels Ereignisse mit dem System.

B.2.6 Verwaltungsserver

B.2.6.1 Spielverwaltung

Funktionsbeschreibung Die Spielverwaltung dient zum Starten des Spielservers. Auf dem Verwaltungsserver, ist identisch mit der Spielverwaltung, werden alle wichtigen Daten für ein Spiel gespeichert. Diese wären alle vorhandenen Karten, Highscores und fest vorgegebene Spielerprofile. Neue Karten und Szenarien können hier eingepflegt werden.

Vorbedingung Vorraussetzung für ein korrektes Funktionieren ist das Vorhandensein geeigneter Hardware und die Karten müssen korrekt mit den spezifischen Eigenarten der Spielfelder, z.B. Häuser und gegebenenfalls Seen, eingepflegt sein.

Datenbasis Es werden die Daten der Spielfelder benötigt und die Adressen der Spielserver.

Software-Schnittstellen Es existiert eine Schnittstelle zum „Sessionmanagement“-Modul zum Weiterleiten von Daten zum Starten des Spielservers.

Hardware-Schnittstellen Der Verwaltungsserver muss über genug Festplattenspeicher verfügen, um die zum Starten des Spielservers relevanten Daten zu speichern.

Fehler Fehler können bei einem Hardwareausfall auftreten.

B.2.7 Verwaltungsmodule

B.2.7.1 Server Such System

Funktionsbeschreibung Über das „Server Such System“ kann nach Verwaltungsservern gesucht werden, die Informationen zu laufenden Agentenspielen liefern.

Vorbedingung Das „Server Such System“ hat die Adresse eines oder mehrerer Verwaltungsserver gespeichert, oder kann diese Adressen von einem dritten Server beziehen.

Datenbasis Es muss dauerhaft die Adresse (bzw. eine Liste mit Adressen) eines (oder mehrerer) Verwaltungsserver gespeichert werden, um zu jeder Zeit Informationen über laufende Spiele abrufen zu können. Alternativ kann das System über die Adresse eines dritten Servers Adressen von Verwaltungsservern beziehen.

Software-Schnittstellen Das „Server Such System“ kommuniziert mit dem System über Ereignisse. Es muss Adressen (Listen mit Adressen) von Servern über die Schnittstelle beziehen können.

Fehler Das „Server Such System“ kann nicht eingesetzt werden, wenn es keine Adresse gespeichert hat, mit der potentiell ein Server adressierbar ist. Des Weiteren tritt ein Fehler auf, wenn das System über keine der vorgehaltenen Adressen einen Verwaltungsserver (oder einen Server der diese Adressen liefert) adressieren kann.

B.3 Funktionalitäten des Frameworks

Diese Teilaufgabe verallgemeinert die in Abschnitt 1 gewonnenen Erkenntnisse. Hier sollen die Funktionalitäten der identifizierten Module abstrakt beschrieben werden, um in einem Framework universell eingesetzt werden zu können.

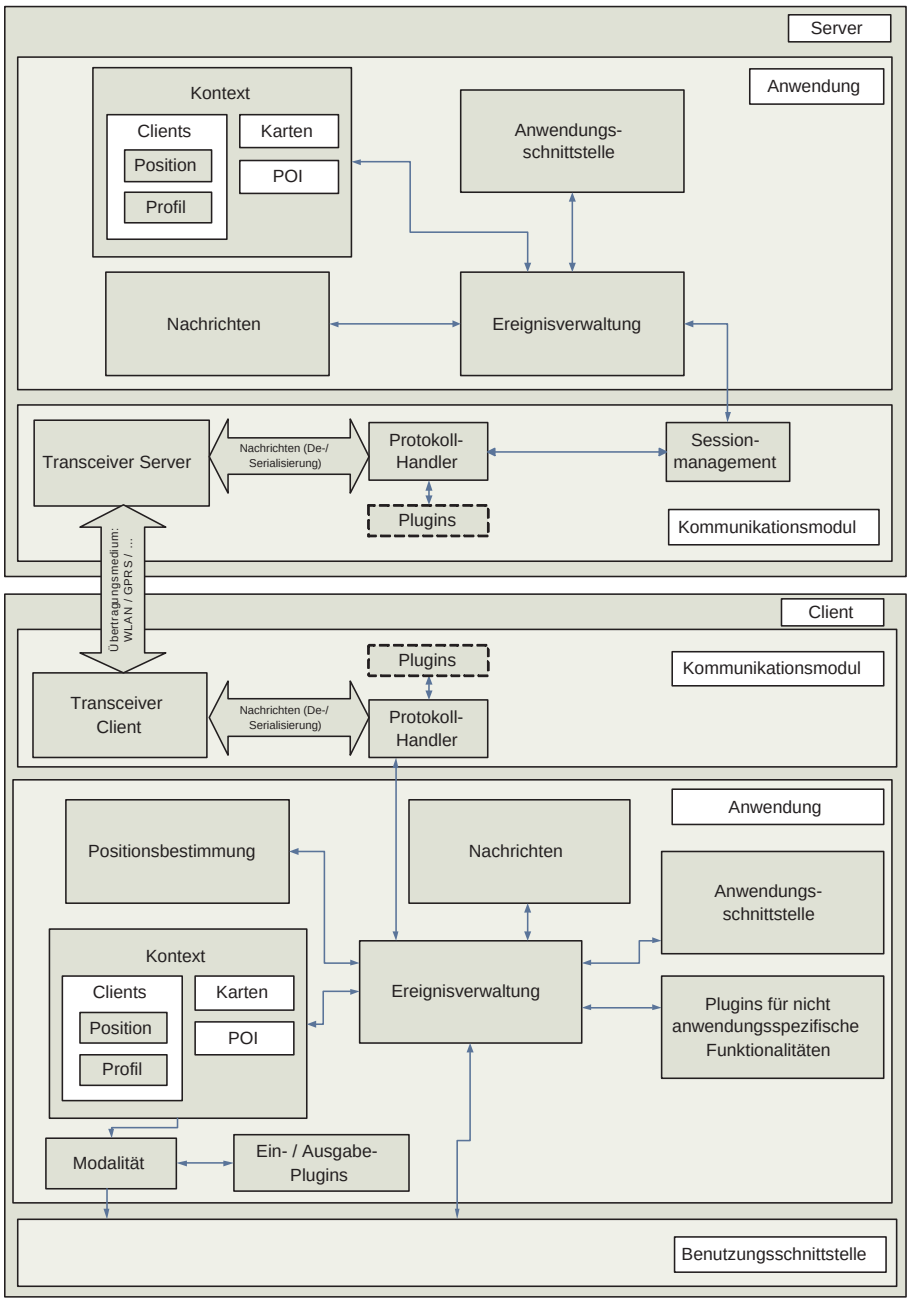


Abbildung B.2: Modulübersicht des Frameworks

B.3.1 Server - Anwendungsmodul

B.3.1.1 Kontext

Funktionsbeschreibung In diesem Modul werden alle Daten gespeichert, die von der Anwendung zur Bestimmung des Kontextes benötigt werden.

Vorbedingung Kontextmodul muss geladen, bzw. auf dem Server installiert sein.

Datenbasis Das Kontextmodul besteht aus drei Modulen:

- Karten: Hier werden die Karten der Anwendungsumgebung verwaltet
- POI (Points Of Interests): Dieses Modul ist für alle nicht statischen Daten der Anwendungsumgebung verantwortlich.
- Clients: Dieses Modul hält sämtliche Daten der Clients vor und besteht aus zwei Modulen:
  - Position: Hier werden die Positionsdaten der Clients gespeichert
  - Profil: Alle individuellen Daten der Clients, wie z.B. *Nickname*, IP-Adresse, usw. werden durch dieses Modul verwaltet.

Software - Schnittstelle Die Daten des Kontextmoduls müssen für andere Module abrufbar sein. Des Weiteren wird eine Schnittstelle zur Aktualisierung der Kontextinformationen vorhanden sein.

Hardware - Schnittstelle Es muss ausreichend Speicherplatz zur Verfügung stehen.

Fehler Beim Abspeichern der Daten muss auf Konsistenz geachtet werden. So müssen die Daten für jeden Client eindeutig bestimmt werden können.

B.3.1.2 Anwendungsschnittstelle

Funktionsbeschreibung An dieser Stelle wird vom Framework eine Schnittstelle für mögliche Anwendungen bereitgestellt.

Software - Schnittstelle Die Anwendungsschnittstelle steht über die Ereignisverwaltung mit den anderen Modulen in Kontakt.

B.3.1.3 Nachrichten

Funktionsbeschreibung Dieses Modul dient dazu, den Nachrichtenaustausch der Clients untereinander zu gewährleisten. Des Weiteren können von hier Systemnachrichten an die Clients geschickt werden.

Vorbedingung Verbindung zu den Clients muss vorhanden sein.

Datenbasis Die Nachrichten müssen bis zur erfolgreichen Weiterleitung zwischengespeichert werden.

Software - Schnittstelle Die Kommunikation mit anderen Modulen erfolgt über die Ereignisverwaltung.

B.3.1.4 Ereignisverwaltung

Funktionsbeschreibung Die Ereignisverwaltung koordiniert den Datenaustausch der einzelnen Module des Anwendungsmoduls.

Datenbasis Die Ereignisverwaltung speichert Referenzen auf die auf die einzelnen Module des Anwendungsmoduls, damit sie Ereignisse weiterleiten kann.

Software - Schnittstelle Die Ereignisverwaltung bietet Schnittstellen zur Verwaltung von Event-Listenern (Registrierung, Deregistrierung) an und empfängt auftretende Ereignisse.

B.3.2 Server - Kommunikationsmodul

B.3.2.1 Transceiver Server

Funktionsbeschreibung „Transceiver"-Module übernehmen die Übertragung der Daten transparent. Dabei kann es für jede Übertragungstechnik ein anderes, spezielles Modul geben. Entsprechend der gewählten Übertragungstechnik wird das Modul ausgewählt und eingesetzt.

Vorbedingung Übertragungstechnik und Transceiver wurden gewählt.

Software - Schnittstelle Transceiver kommunizieren mit dem Protokoll Handler und den Treibern der jeweiligen Übertragungstechnik.

Fehler Anhaltende Fehler sollten als Verbindungsabbruch gewertet werden.

B.3.2.2 Protokoll Handler

Funktionsbeschreibung Der Protokoll Handler ist für die Kodierung des Protokolls verantwortlich. Er übersetzt Ereignisse in Befehl-Strings des Protokolls und zurück.

Vorbedingung Ein Transceiver wurde für die Kommunikation ausgewählt.

Datenbasis Datenbasis ist das Protokoll selber. Ereignisse müssen Befehlen zugeordnet werden.

Software - Schnittstelle Der Protokoll Handler erhält und sendet Protokolldaten über das „Tranceiver"-Modul. Serverseitig existiert eine Schnittstelle zum „Session Management"-Modul.

Fehler Das Protokoll sollte bis zu einem bestimmten Grad fehlertolerant ausgelegt sein. Stimmen Daten nicht mit den erwarteten überein, wird die Kommunikation wiederholt. Erst bei mehreren Fehlschlägen wird ein Fehler ausgelöst.

B.3.2.3 Plugins

Funktionsbeschreibung Es ist vorgesehen, Schnittstellen bereit zu stellen, um die Internetkommunikation erweitern oder anpassen zu können.

B.3.2.4 Session Management

Funktionsbeschreibung Das „Session-Management"-Modul begleitet den Lebenszyklus der Kommunikation mit allen Clients. Über dieses Modul können alle Clients vom Server identifiziert und angesprochen werden. Das „Session-Management"-Modul kümmert sich auch um das Einloggen neuer Clients sowie um Aufräumungsarbeiten beim Verbindungsabbau oder Time-Out.

Vorbedingung Der Server muss funktionstüchtig sein und Verbindungen entgegennehmen können.

Fehler Bei anhaltenden Fehlern sollte die Session eines Clients beendet werden.

B.3.3 Client - Anwendungsmodul

B.3.3.1 Anwendungsschnittstelle

Funktionsbeschreibung Die Anwendungsschnittstelle verbindet das Framework mit den anwendungsspezifischen Modulen, die zusätzlich auf dem Client benötigt werden.

Datenbasis Die Datenbasis hängt von der jeweiligen Anwendung ab.

Software - Schnittstelle Die Kommunikation mit anderen Modulen erfolgt über die Ereignisverwaltung.

B.3.3.2 Nachrichten

Funktionsbeschreibung Das Nachrichtenmodul ermöglicht das Empfangen und Versenden von Nachrichten. Gesendete und empfangene Nachrichten werden hier verwaltet und auch lokal gespeichert.

Datenbasis Die Möglichkeit zur Speicherung gesendeter und empfangener Nachrichten muss gegeben sein.

B.3.3.3 Ereignisverwaltung

Funktionsbeschreibung Das Ereignisverwaltungsmodul koordiniert und verwaltet alle Ereignisse auf dem Client. Es leitet die Ereignisse zwischen den Modulen des Clients weiter.

Software - Schnittstelle Als zentrales Modul kommuniziert das Ereignisverwaltungsmodul mit allen Frameworkmodulen des Clients.

B.3.3.4 Positionsbestimmung

Funktionsbeschreibung Das Positionsverwaltungsmodul ermittelt unter Zuhilfenahme geeigneter Hardware die aktuelle Position des mobilen Endgerätes. Je nach eingesetzter Technik kann die Positionsinformation aus Koordinaten oder einer Zellen-ID bestehen. Geeignete Hardware könnte ein GPS-Modul oder ein WLAN-Modul sein. GPS-Module liefern die aktuelle Position auf Anfrage in Form von absoluten Koordinaten. Wird die Position über ein WLAN-Modul berechnet, müssen Daten wie Sender-ID, Signallaufzeiten oder Feldstärken berücksichtigt werden.

Vorbedingung Voraussetzung für die korrekte Funktion des Moduls ist das Vorhandensein geeigneter Hardware zur Positionsbestimmung. Bei Verwendung von WLAN müssen zusätzlich Daten über die genauen Standpunkte der Accesspoints sowie evtl. Feldstärkedaten zur Verfügung gestellt werden.

Datenbasis Bei Verwendung von GPS werden keine zusätzlichen Daten benötigt, da die Positionsdaten direkt von der Hardware geliefert werden und an das anfragende Modul weiterreichen lassen. Bei Positionsbestimmung über ein WLAN-Modul muss auf Daten, wie die IDs und Positionsdaten der Accesspoints, sowie die Empfangsstärken der Accesspoint an unterschiedlichen Positionen, zugegriffen werden.

B.3.3.5 Plugins für nicht anwendungsspezifische Funktionalitäten

Funktionsbeschreibung Es ist vorgesehen, Schnittstellen bereit zu stellen, um das Framework mit anderen Modulen zu erweitern.

B.3.3.6 Kontext

Funktionsbeschreibung Dieses Modul enthält alle vom Client benötigten Kontextdaten. Es handelt sich hierbei um eine lokale Kopie des Kontextmoduls des Servers.

Vorbedingung Es muss eine Verbindung zum Server bestehen, damit die Daten des Kontextmoduls des Clients mit denen des Servers abgeglichen werden können.

Datenbasis Das Kontextmodul besteht aus drei Modulen:

- Karten: Hier werden die Karten der Anwendungsumgebung verwaltet

- POI (Points Of Interests): Dieses Modul ist für alle nicht-statischen Daten der Anwendungsumgebung, verantwortlich.
- Clients: Dieses Modul hält die für den Client notwendigen Daten der anderen Clients vor und besteht aus zwei Modulen:
  - Position: Hier werden die Positionsdaten der Clients gespeichert
  - Profil: Alle individuellen Daten der Clients, wie z.B. *Nickname*, IP-Adresse, usw. werden durch dieses Modul verwaltet.

B.3.3.7 Modalität

Funktionsbeschreibung Dieses Modul berechnet aus den Daten des Kontextmoduls die augenblicklichen Modalitätseigenschaften des Client.

Vorbedingung Das Kontextmodul muss für die Berechnung der Modalitätseigenschaften bereits Daten (wenn auch nur Defaultwerte) enthalten.

Software - Schnittstelle Das Modalitätsmodul empfängt Daten vom Kontextmodul und sendet Daten an die Benutzungsschnittstelle.

Fehler Fehlerhafte Berechnungen des Modalitätsmoduls können aufgrund fehlerhafter, vom Kontextmodul empfangener Daten, auftreten.

B.3.3.8 Ein-/Ausgabe Plugins

Funktionsbeschreibung Es ist vorgesehen, Schnittstellen bereitzustellen, um die Nutzungsschnittstelle erweitern oder anpassen zu können.

B.3.4 Client - Kommunikationsmodul

B.3.4.1 Transceiver Client

Funktionsbeschreibung „Transceiver“-Module übernehmen die Übertragung der Daten transparent. Dabei kann es für jede Übertragungstechnik ein anderes, spezielles Modul geben. Entsprechend der gewählten Übertragungstechnik wird das Modul ausgewählt und eingesetzt.

Vorbedingung Übertragungstechnik und Transceiver wurden gewählt.

Software - Schnittstelle Transceiver kommunizieren mit dem „Protokoll-Handler“-Modul und den Treibern der jeweiligen Übertragungstechnik.

Fehler Anhaltende Fehler sollten als Verbindungsabbruch gewertet werden.

B.3.4.2 Protokoll Handler

Funktionsbeschreibung Das „Protokoll-Handler“-Modul ist für die Kodierung des Protokolls verantwortlich. Er übersetzt Ereignisse in Befehl-Strings des Protokolls und zurück.

Vorbedingung Ein Transceiver wurde für die Kommunikation ausgewählt.

Datenbasis Datenbasis ist das Protokoll selber. Ereignisse müssen Befehlen zugeordnet werden.

Software - Schnittstelle Das „Protokoll-Handler“-Modul erhält und sendet Protokolldaten über das „Transceiver“-Modul. Der Protokoll-Handler auf dem Client besitzt eine direkte Schnittstelle zur Ereignisverwaltung.

Fehler Das Protokoll sollte bis zu einem bestimmten Grad fehlertolerant ausgelegt sein. Stimmen Daten nicht mit den erwarteten überein, wird die Kommunikation wiederholt. Erst bei mehreren Fehlschlägen wird ein Fehler ausgelöst.

B.3.4.3 Plugins

Funktionsbeschreibung Es ist vorgesehen, Schnittstellen bereit zu stellen, um die Internetkommunikation erweitern oder anpassen zu können.

B.3.5 Client - Benutzungsschnittstelle

Funktionsbeschreibung Die Ausgabe der benutzerrelevanten Daten erfolgt über die Benutzungsschnittstelle und kann visuell und auditiv (evtl. auch haptisch) erfolgen. Benutzereingaben werden von der Benutzerschnittstelle an die Ereignisverwaltung weitergeleitet.

Vorbedingung Das Modul „Benutzerschnittstelle“ muss bei der Ereignisverwaltung als Event-Listener registriert sein, um Ereignisse zu empfangen.

Software - Schnittstelle Die Benutzungsschnittstelle kommuniziert über Ereignisse mit dem System, um Benutzereingaben an die Ereignisverwaltung weiterzuleiten oder von dort Daten zu empfangen. Es existiert ferner eine Verbindung zum Modalitätsmodul, welches die Benutzungsschnittstelle mit Daten bezüglich der Modalität des Clients versorgt und somit Einfluss auf das Ein- und Ausgabeformat nimmt.

B.4 Datenhaltung

In diesem Abschnitt soll die Art der Datenhaltung beleuchtet werden. Um das Problem der Datenhaltung zu lösen, stehen uns grundsätzliche zwei Architekturansätze zur Verfügung. Den Einsatz einer **Client/Server-Architektur** oder den Einsatz einer **verteilten Datenhaltung**

- **Client/Server-Architektur**
  Beim Einsatz einer Client/Server-Architektur fordert der Client Dienste des Servers an, wobei der Client die Anwendung repräsentiert und der Server für die Bereitstellung von Funktionalität und Daten zuständig ist. Der Ansatz der mobilen Client/Server-Architektur unterscheidet sich kaum von der klassischen Client/Server-Architektur. Einzige Unterschiede sind, dass der Zugriff auf Datenbank-Server durch den Client drahtlos erfolgt und das es mobilen Clients ermöglicht wird, auch auf replizierten Daten zu arbeiten. Aufgrund der verringerten Bandbreite und der unsicheren Verfügbarkeit durch die drahtlose Kommunikation, führt eine solche Lösung zu weiteren Problemen und soll darum nicht in Betracht gezogen werden.

- **Verteilte Datenhaltung**
  Hauptidee bei einer verteilten Datenhaltung ist die Verteilung der Daten und Datenbankfunktionalität auf verschiedene Endgeräte (Knoten), wobei jeder Knoten über eine eigene Datenbank verfügt. Das führt dazu, dass die notwendige Übertragungskapazität bei einer verteilten Datenhaltung weitaus geringer ist als bei der Client/Server-Architektur. Alle Knoten bilden dabei das verteilte System. Bei Mobilen verteilten Systeme kommt aber erschwerend hinzu, dass mobile Knoten häufig vom Gesamtsystem getrennt sind. Das macht die Einführung von Datenredundanz nötig, damit notwendige Daten auch im nicht verbundenen Zustand zur Verfügung stehen und den Abgleich von redundant vorhandenen Daten, an denen Änderungen vorgenommen wurden.

Wir haben uns für die verteilte Datenhaltung entschieden, da wir die Übertragungskapazität möglichst gering halten wollten und das Spiel auch fortgeführt werden kann, wenn die Übertragung einmal abbricht. Die nachfolgenden zwei Tabellen B.1 und B.2 stellen eine grobe Übersicht über die in den einzelnen Modulen zu speichernden Daten dar, und listen auf, ob sich die Daten während des Spiels verändern (veränderlich), oder ob sie während des Spiels konstant blieben (statisch).

Modul	Welche Daten	Art der Daten
Spielverwaltung	Daten der Spielfelder	statisch
	Adressen der Spielfelder	statisch
Ereignisverwaltung	Referenzen der Module	veränderlich
Spielobjektverwaltung	Missionsziele	veränderlich
	Goodies	veränderlich
	Kameras	veränderlich
	Neutralisationskreise	veränderlich
	Spieler	veränderlich
	Startpunkt	statisch
Szenarioverwaltung	statische Szenariodaten	statisch
	aktueller Zustand des Szenarios	veränderlich
Nachrichtenweiterleitung	Nachrichten	veränderlich
Admin	-	-

Tabelle B.1: Datenhaltung des Servers

Modul	Welche Daten	Art der Daten
Server Such System	Adresse der Verwaltungsserver	statisch
Positionsverwaltung	-	-
Nachrichtenverwaltung	empfangene Nachrichten	veränderlich
	gesendete Nachrichten	veränderlich
Ereignisverwaltung	Referenzen der Module	veränderlich
Kontextsensitive Hilfe	-	-
UI Konfiguration	Einstellung der UI	veränderlich
Modalitätsverwaltung	-	-
Kontextmanager	aktuelle Position	veränderlich
	Positionsgeschichte	veränderlich
Spielerprofil	Name	statisch
	Attribute	statisch
	Status	veränderlich
	Team	statisch
Cache f. dynamische Daten	variable Daten	veränderlich
Szenariomanager	statische Szenariodaten	statisch

Tabelle B.2: Datenhaltung des Clients

B.5 Umsetzung der Spielidee mit dem Grobkonzept

In diesem Abschnitt soll untersucht werden, in wiefern unsere in der Anforderungsdefinition identifizierten Anwendungsfälle durch unser Grobkonzept realisiert werden können.

B.5.1 Systemfunktionen

B.5.1.1 Systemnachricht

Voraussetzung Der Spieler muss einem Spiel beigetreten sein.

Ablauf

1. Die Szenarioverwaltung erzeugt eine Nachricht an einen Spieler.
2. Dieses wird an die Ereignisverwaltung und anschließend der Nachrichtenweiterleitung weitergeleitet.
3. Von der Nachrichtenweiterleitung wieder zurück zur Ereignisverwaltung und von dort zur Ereignisverwaltung des Clients.
4. Von hier gehts über die Nachrichtenverwaltung zum UI des Clients.

Ausgabe/Nachbedingung Die Nachricht erscheint auf dem UI des Spielers.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.2 Spielbeitritt

B.5.2.1 Spiel beitreten

Voraussetzung Der Benutzer muss das Programm gestartet haben und es muss eine Verbindung zu einem Verwaltungsserver möglich sein. Ausserdem muss der Benutzer ein verfügbares Spiel ausgewählt haben.

Ablauf

1. Der Benutzer wählt im User Interface den Menüpunkt zum Beitreten des ausgewählten Spiels.
2. Das User Interface erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
3. Über das Kommunikationsmodul des Clients und das Kommunikationsmodul des Server wird das Ereignis an die Ereignisverwaltung des Spielservers gesendet.
4. Die Ereignisverwaltung leitet das Ereignis an die Szenarioverwaltung weiter, die neue Spielobjekte (Spieler, Startposition ...) erzeugt und der Spielobjektverwaltung hinzufügt.
5. Dem neu hinzugefügten Client wird über das Sessionmanagement eine eindeutige ID zugeordnet, um eine eindeutige Kommunikation zu diesem Client während des Spielablaufs zu ermöglichen.

6. Die Szenarioverwaltung erzeugt ein neues Ereignis, das den Erfolg der Aktion signalisieren soll.
7. Dieses Ereignis wird über die Ereignisverwaltung und Kommunikationsmodul des Servers sowie das Kommunikationsmodul und die Ereignisverwaltung des Clients an den Kontextmanager des Clients gesendet.
8. Dieser generiert die Kontextanforderungen und sendet diese an die Modalitätsverwaltung.
9. Die Modalitätsverwaltung ermittelt aus den Kontextanforderungen die Ausgabedaten und leitet diese an das User Interface weiter.
10. Im User Interface werden diese Daten dann ausgegeben.

Ausgabe/Nachbedingung Eine Ausgabe muss eigentlich nicht erfolgen. Um den Erfolg des Beitritts zu signalisieren ist jedoch eine Ausgabe in Form von näheren Instruktionen zur Rolle im Spiel wünschenswert.
Als Nachbedingung muss gelten, dass der Spieler und seine Startposition als Spielobjekte zur Spielobjektverwaltung hinzugefügt wurden und dem Client eine eindeutige ID zugeordnet wurde.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Das Framework stellt ebenfalls die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kommunikationsmodul auf Clientseite und Kommunikationsmodul, Ereignisverwaltung und Kontext auf Serverseite zur Verfügung.

B.5.2.2 Spiel auswählen

Voraussetzung Der Benutzer muss das Programm gestartet haben und es muss eine Verbindung zu einem Verwaltungsserver möglich sein. Ausserdem muss der Verwaltungsserver die Informationen über die verfügbaren Spiele halten das aktuelle Fenster die Anzeige der verfügbaren Spiele sein.

Ablauf

1. Der Benutzer wählt im User Interface ein verfügbares Spiel aus und aktiviert den Menüpunkt zur Anzeige näherer Informationen.
2. User Interface erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
3. Über das Kommunikationsmodul des Clients und das Kommunikationsmodul des Servers wird das Ereignis an die Spielverwaltung des Verwaltungsserver gesendet.
4. Die Spielverwaltung erzeugt anhand der näheren Informationen zum ausgewählten Spiel ein angemessenes Ereignis.
5. Dieses Ereignis wird über das Kommunikationsmodul des Servers und das Kommunikationsmodul des Clients an die Ereignisverwaltung des Clients gesendet.
6. Das Ereignis wird über den Kontextmanager und die Modalitätsverwaltung an das User Interface weitergeleitet.
7. Im User Interface werden die näheren Informationen zum ausgewählten Spiel ausgegeben.

Ausgabe/Nachbedingung Es erfolgt eine Ausgabe der näheren Informationen über das vom Benutzer ausgewählte Spiel.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Anwendungsschnittstelle zur Verfügung.

B.5.2.3 Server erstellen

Voraussetzung Der Benutzer muss das Programm gestartet haben und über die notwendigen Rechte verfügen, um einen neuen Spielserver zu erstellen. Ausserdem muss eine Verbindung zu einem Verwaltungsserver bestehen und das zu spielende Szenario muss auf dem Spielserver zur Verfügung stehen.

Ablauf

1. Der Benutzer wählt im Verwaltungsmodul den Menüpunkt „Server erstellen“
2. Danach müssen alle Spielrelevanten Daten, wie z.B. Anzahl der Spiele, Dauer des Spiels oder die zu spielende Karte eingegeben und bestätigt werden.
3. Die Verwaltungsserver-Fernsteuerung erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
4. Über das Kommunikationsmodul des Client und das Kommunikationsmodul des Server wird das Ereignis an die Spielverwaltung des Verwaltungsservers gesendet.
5. Die Spielverwaltung erstellt dann einen neuen Spielserver anhand der vom Benutzer eingegebenen Daten.

Ausgabe/Nachbedingung Es muss keine Ausgabe erfolgen. Lediglich der Bildschirm zur Servererstellung sollte verlassen werden.
Als Nachbedingung muss gelten, dass der Spielserver anhand der vom Benutzer eingegebenen Daten gestartet wurde und über das „Server Such System“ des Client aufzufinden ist.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.2.4 Genaue Position der Alarmanlage sehen

Voraussetzung Der Spieler muss einem Spiel beigetreten und vom System als „Mister X“ ausgewählt worden sein.
Ansonsten erfordert dieser Anwendungsfall keinerlei Aktionen vom Benutzer.

Ablauf Der Ablauf dieses Anwendungsfalls ist implementierungsabhängig. Grundsätzlich sollte der Ablauf jedoch folgendermaßen sein:

1. Beim Spielstart wird von der Ereignisverwaltung ein geeignetes Ereignis an das Datenmodul gesendet. Daraufhin werden die statischen Szenariodaten aus dem Datenmodul geladen.

2. Es folgt eine Abfrage anhand der Profilinformationen, ob der Spieler „Mister X“ ist.

3. Ist die Abfrage positiv, so werden die Szenariodaten inklusive der genauen Positionen der Alarmanlage(n) vom User Interface angezeigt.

Ausgabe/Nachbedingung Die genauen Positionen der Alarmanlage(n) werden permanent auf der Karte angezeigt. Wird eines der Missionsziele im Spielverlauf erfüllt, oder eine Alarmanlage deaktiviert, so ist die dazugehörige Alarmanlage nicht mehr sichtbar.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.2.5 Fähigkeitspunkte anpassen und Charakter annehmen

Voraussetzung Der Spieler muss einem Spiel beigetreten sein und hat bereits seinen Benutzernamen eingegeben. Dieser „Username“ ist bereits im Profil gespeichert worden. Dem Benutzer wird ein Bildschirm angezeigt, auf dem er die Punkte für die Fähigkeiten „Abhören“, „Aufklären“, „Neutralisieren“ und „Tarnen“ einstellen kann.

Ablauf

1. Der Spieler wählt im User Interface seine Verteilung der Fähigkeitspunkte und bestätigt diese.

2. Das User Interface erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.

3. Die Ereignisverwaltung ruft eine „Setter-Methode“ im Datenmodul auf, die die Merkmalsausprägungen der einzelnen Attribute anhand der eingestellten Fähigkeitenpunkte im Profil aktualisiert.

Ausgabe/Nachbedingung Es muss keine Ausgabe erfolgen. Lediglich der Bildschirm zur Punkteverteilung sollte verlassen werden.
Als Nachbedingung muss gelten, dass die Punkteverteilung bei keiner Fähigkeit den Minimalwert (zur Zeit 1) unterschreitet und den Maximalwert (zur Zeit 4) überschreitet.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.2.6 Fähigkeiten aktivieren

Voraussetzung Der Spieler muss einem Spiel beigetreten sein und bereits seine Fähigkeitspunkte verteilt haben. Ferner muss er seinen Startpunkt erreicht haben.

Ablauf

1. Die Positionsbestimmung übermittelt die Positionsdaten an die Ereignisverwaltung.
2. Ist die korrekte Position erreicht, werden die Fähigkeiten aktiviert.
3. Der Status wird an die Ereignisverwaltung des Servers übermittelt.
4. Die Ereignisverwaltung schickt an das User Interface eine entsprechende Nachricht, die das Aktivieren der Fähigkeiten für den Spieler bestätigt.

Ausgabe/Nachbedingung Es erfolgt die Bestätigung der Aktivierung als Ausgabe. Die Fähigkeiten sind aktiviert.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.2.7 Verfügbare Spiele finden

Voraussetzung Der Benutzer muss das Programm gestartet haben und es muss eine Verbindung zu einem Verwaltungsserver möglich sein.

Ablauf

1. Der Benutzer wählt im Verwaltungsmodul den Menüpunkt „Verfügbare Spiele finden“.
2. Das „Server Such System“ erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
3. Über das Kommunikationsmodul des Clients und das Kommunikationsmodul des Server wird das Ereignis an die Spielverwaltung des Verwaltungsservers gesendet.
4. Die Spielverwaltung erzeugt anhand aller verfügbaren Spiele ihrerseits ein angemessenes Ereignis.
5. Dieses Ereignis wird über das Kommunikationsmodul des Servers und das Kommunikationsmodul des Clients an die Ereignisverwaltung des Clients gesendet.
6. Diese leitet das Ereignis an den Kontextmanager weiter, der die Kontextanforderungen (z.B. aktueller Standort des Benutzers) für die Modalitätsverwaltung erstellt.
7. Die Modalitätsverwaltung ermittelt anhand der Kontextanforderungen die Ausgabedaten und sendet diese an das User Interface, welches diese Daten dann ausgibt.

Ausgabe/Nachbedingung Es erfolgt eine Ausgabe aller Spiele, die in räumlicher Nähe zur aktuellen Position des Benutzers stattfinden.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Anwendungsschnittstelle zur Verfügung.

B.5.2.8 Vorgefertigten Charakter auswählen

Voraussetzung Der Spieler muss einem Spiel beigetreten sein und hat bereits seinen Usernamen eingegeben. Dieser Username ist bereits im Profil gespeichert worden. Ausserdem muss eine Liste von vorgefertigten Charakteren, mindestens jedoch einer, im Szenariomanager für statische Daten vorhanden sein. Diese Liste wird dem Spieler auf der Benutzungsoberfläche angezeigt.

Ablauf

1. Der Spieler wählt im User Interface einen vorgefertigten Charakter aus und bestätigt diese Auswahl.
2. Das User Interface erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
3. Die Ereignisverwaltung ruft eine „Setter-Methode“ im Datenmodul auf, die die Merkmalsausprägungen der einzelnen Attribute anhand der Fähigkeitenpunkte des ausgewählten Charakters im Profil aktualisiert.

Ausgabe/Nachbedingung Es muss keine Ausgabe erfolgen. Lediglich der Bildschirm zur Charakterauswahl sollte verlassen werden.
Als Nachbedingung gilt, dass die Fähigkeitspunkte des vorgefertigten Charakters korrekt in das Spielerprofil eingetragen wurden.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.3 Nachrichten

B.5.3.1 Nachricht lesen und schreiben

Voraussetzung Der Spieler muss einem Spiel beigetreten sein.

Ablauf

1. Der Spieler wählt im User Interface Nachricht lesen, bzw. schreiben aus.
2. Der Spieler kann eine Nachricht schreiben und wählt die Empfänger aus.
3. Die Nachricht wird an die Nachrichtweiterleitung des Spielservers übermittelt.
4. Anschließend wird die Nachricht an die Nachrichtenverwaltung der jeweiligen Clients übermittelt.

5. Die Empfänger bekommen ein Signal, dass sie eine Nachricht erhalten haben.

6. Die Empfänger können die Nachricht abrufen und lesen und bei Wunsch anschließend löschen.

Ausgabe/Nachbedingung Die entsprechenden Spieler können die Nachricht abrufen, lesen und ggf. löschen.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.3.2 Nachricht abhören

Voraussetzung Die Spieler sind einem Spiel beigetreten. Der Spieler, der eine Nachricht abhören möchte, befindet sich in diesem Modus und ist in Reichweite von zwei Spielern, die sich eine Nachricht schicken.

Ablauf

1. Nachricht wird von der Nachrichtweiterleitung des Spielservers über die jeweilige Ereignisverwaltung an das Ausgabemodul des Clients geleitet.

Ausgabe/Nachbedingung Nachricht wurde abgefangen und der Spieler kann die abgefangene Nachricht lesen.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.4 Details auf der Karte sehen

B.5.4.1 Ungenaue Position der Alarmanlage sehen

Voraussetzung Der Spieler muss einem Spiel als „Abwehragent“ beigetreten sein.
Ansonsten erfordert dieser Anwendungsfall keinerlei Aktionen vom Benutzer.

Ablauf Der Ablauf dieses Anwendungsfalls ist implementierungsabhängig. Grundsätzlich sollte der Ablauf jedoch folgendermaßen sein:

1. Beim Spielstart wird von der Ereignisverwaltung ein geeignetes Ereignis an das Datenmodul gesendet. Daraufhin werden die statischen Szenariodaten aus dem Datenmodul geladen.

2. Es folgt eine Abfrage anhand der Profilinformationen, ob der Spieler „Abwehragent“ ist.

3. Ist die Abfrage positiv, so werden die Szenariodaten inklusive der ungenauen Positionen der Alarmanlage(n) vom User Interface angezeigt.

Ausgabe/Nachbedingung Die ungenauen Positionen der Alarmanlage(n) werden permanent auf der Karte angezeigt. Wird eines der Missionsziele im Spielverlauf erfüllt, oder eine Alarmanlage deaktiviert, so ist die dazugehörige Alarmanlage nicht mehr sichtbar.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.4.2 Teammitglied oder Gegner auf Karte sehen

Voraussetzung Der Spieler muss einem Spiel beigetreten sein.

Ablauf

1. Die Positionsverwaltung des Clients übermittelt die Positionsdaten an die Spielobjektverwaltung des Servers (über die Ereignisverwaltung des Clients und des Servers).
2. Die Positionsdaten werden von der Spielobjektverwaltung an die Modalitätsverwaltung des Ausgabemoduls übermittelt (wieder über die jeweilige Ereignisverwaltung).

Ausgabe/Nachbedingung Der Spieler sieht die anderen Spieler auf der Karte.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.4.3 Karte zoomen

Voraussetzung Der Spieler muss einem Spiel beigetreten sein und das aktuelle Fenster muss die Kartenansicht sein. Der Spieler darf sich nicht im Tarnmodus befinden, da sonst die Kartenansicht nicht verfügbar ist und außerdem muss die aktuelle Spielkarte in geeigneter Form auf dem Client verfügbar sein.

Ablauf Im Folgenden soll nur beispielhaft dargestellt werden, wie dieser Anwendungsfall durch unser Grobkonzept realisiert werden könnte.

1. Der Spieler wählt die Funktion vergrößern (oder verkleinern) aus.
2. Das User Interface nimmt die Benutzeraktion auf und verarbeitet diese intern.
3. Die Karte wird in der angepassten Zoomstufe vom User Interface angezeigt.

Ausgabe/Nachbedingung Als Ausgabe wird dem Benutzer die Karte in der neuen Zoomstufe angezeigt.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über das Modul Benutzungsschnittstelle zur Verfügung.

B.5.4.4 Karte zentrieren

Voraussetzung Der Spieler muss einem Spiel beigetreten sein und das aktuelle Fenster muss die Kartenansicht sein. Der Spieler darf sich nicht im Tarnmodus befinden, da sonst die Kartenansicht nicht verfügbar ist und außerdem muss die aktuelle Spielkarte in geeigneter Form auf dem Client verfügbar sein.

Ablauf Im Folgenden soll nur beispielhaft dargestellt werden, wie dieser Anwendungsfall durch unser Grobkonzept realisiert werden könnte.

1. Der Spieler wählt einen Punkt auf der Karte aus. (z.B. durch Anklicken)
2. Das User Interface nimmt die Benutzeraktion auf und verarbeitet diese intern.
3. Die Karte wird in der angepassten Version vom User Interface angezeigt.

Ausgabe/Nachbedingung Als Ausgabe wird dem Benutzer die Karte mit dem durch ihn bestimmten Mittelpunkt angezeigt.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über das Modul Benutzungsschnittstelle zur Verfügung.

B.5.4.5 Genaue Positionen der Missionsziele sehen

Voraussetzung Der Spieler muss einem Spiel beigetreten und vom System als „Mister X“ ausgewählt worden sein.
Ansonsten erfordert dieser Anwendungsfall keinerlei Aktionen vom Benutzer.

Ablauf Der Ablauf dieses Anwendungsfalls ist implementierungsabhängig. Grundsätzlich sollte der Ablauf jedoch folgendermaßen sein:

1. Beim Spielstart wird von der Ereignisverwaltung ein geeignetes Ereignis an das Datenmodul gesendet. Daraufhin werden die statischen Szenariodaten aus dem Datenmodul geladen.
2. Es folgt eine Abfrage anhand der Profilinformationen, ob der Spieler „Mister X“ ist.
3. Ist die Abfrage positiv, so werden die Szenariodaten inklusive der genauen Positionen der Missionsziele vom User Interface angezeigt.

Ausgabe/Nachbedingung Die genauen Positionen der Missionsziele werden permanent auf der Karte angezeigt. Wird eines der Missionsziele im Spielverlauf erfüllt, so ist dieses Missionsziel nicht mehr sichtbar.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.4.6 Genaue Position der Alarmanlage sehen

Voraussetzung Der Spieler muss einem Spiel beigetreten und vom System als „Mister X“ ausgewählt worden sein.
Ansonsten erfordert dieser Anwendungsfall keinerlei Aktionen vom Benutzer.

Ablauf Der Ablauf dieses Anwendungsfalls ist implementierungsabhängig. Grundsätzlich sollte der Ablauf jedoch folgendermaßen sein:

1. Beim Spielstart wird von der Ereignisverwaltung ein geeignetes Ereignis an das Datenmodul gesendet. Daraufhin werden die statischen Szenariodaten aus dem Datenmodul geladen.
2. Es folgt eine Abfrage anhand der Profilinformationen, ob der Spieler „Mister X“ ist.
3. Ist die Abfrage positiv, so werden die Szenariodaten inklusive der genauen Positionen der Alarmanlage(n) vom User Interface angezeigt.

Ausgabe/Nachbedingung Die genauen Positionen der Alarmanlage(n) werden permanent auf der Karte angezeigt. Wird eines der Missionsziele im Spielverlauf erfüllt, oder eine Alarmanlage deaktiviert, so ist die dazugehörige Alarmanlage nicht mehr sichtbar.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten u.a. über die Module Benutzungsschnittstelle, Ereignisverwaltung und Kontext zur Verfügung.

B.5.5 Spielfunktionen

B.5.5.1 Tarnen

Voraussetzung Der Spieler muss einem Spiel beigetreten sein und hat seine Fähigkeiten aktiviert.

Ablauf

1. Der Spieler wählt im User Interface die Funktion „Tarnen“ aus.
2. Das User Interface erzeugt ein entsprechendes Ereignis und leitet dieses an die Ereignisverwaltung des Clients und des Spielservers weiter.
3. Es wird ein Ereignis erzeugt, dass der Spieler auf seiner Karte keine anderen Positionen mehr erkennen kann.
4. Der Spielserver sendet das Ereignis an alle anderen Clients weiter, so dass der getarnte Spieler auf keinem Client mehr zu erkennen ist.

Ausgabe/Nachbedingung Für die Zeit des Tarnens kann der Spieler auf der Karte keine anderen Positionen erkennen. Der Spieler ist für niemanden sichtbar. Nach dem Ablauf der Zeit erhält der Spieler eine entsprechende Nachricht.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.5.2 Neutralisieren

Voraussetzung Die Fähigkeiten des Spielers müssen aktiviert sein.

Ablauf

1. Der Spieler wählt im User Interface die Funktion „Neutralisieren“ aus.
2. Das User Interface erzeugt ein angemessenes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
3. Das Ereignis wird an die Ereignisverwaltung des Spielservers weitergeleitet.
4. Die Ereignisverwaltung des Spielservers erfragt sich bei der Spielobjektverwaltung die aktuelle Position des Spielers (zusätzlich auch die Fähigkeitspunkte des Spielers).
5. Vom aktuellen Punkt aus erzeugt die Ereignisverwaltung ein Gebiet, indem der Spieler neutralisieren kann und sendet dieses Gebiet an das User Interface des Clients.
6. Der Spieler wählt nun ein Zielgebiet aus.
7. Dieses wird wieder an die Ereignisverwaltung des Servers übertragen.
8. Die Ereignisverwaltung sendet nun ein Warnsignal an betroffene Spieler im Zielgebiet aus (über die Ereignisverwaltung des jeweiligen Clients und der Modalitätsverwaltung an das User Interface).
9. Die Ereignisverwaltung des Spielservers überprüft die Zeit, wie lange sich die Spieler im Zielgebiet befinden.
10. Die Ereignisverwaltung des Spielservers sendet einen Statusbericht an alle Spieler, die an dieser Aktion beteiligt waren.

Ausgabe/Nachbedingung Ausgabe ist der Statusbericht. Der/Die Gegenspieler ist/sind gegebenenfalls neutralisiert.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.5.3 Missionsziel erfüllen

Voraussetzung „Mister X“ hat die Position, an der sich das Missionsziel befindet, erreicht.

Ablauf

1. Vom Server wird eine Nachricht mit einem Rätsel an „Mister X“ geschickt.
2. „Mister X“ gibt die Antwort ein und diese wird an die Spiellogik weitergeleitet.
3. Die Spiellogik überprüft das Ergebnis und erzeugt einen Wahrheitswert, der an das Ausgabemodul weitergeleitet wird.
4. War das Ergbnis falsch, so erhält „Mister X“ einen neuen Versuch.
5. War das Ergebnis richtig, dann ist das Missionsziel erfüllt.

Ausgabe/Nachbedingung Das Missionsziel wurde erfüllt oder das Spiel gilt (nach einer bestimmten Anzahl an Versuchen) als verloren.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.5.4 Aufklären

Voraussetzung Die Fähigkeiten des Spielers müssen aktiviert sein.

Ablauf

1. Der Spieler wählt im User Interface die Funktion „Aufklären“ aus und setzt eine Kamera.
2. Das User Interface erzeugt ein entsprechendes Ereignis und leitet dieses an die Ereignisverwaltung weiter.
3. Das Ereignis wird an die Ereignisverwaltung des Spielservers gesendet und von dort an die Spielobjektverwaltung weitergeleitet.
4. Die Spielobjektverwaltung schickt nun die Positionsdaten der Spieler, die sich im Umkreis der Kamera befinden, an die Modalitätsverwaltung des Clients (über die jeweilige Ereignisverwaltung).
5. Nach Ablauf einer Frist wird die Kamera deaktiviert und der Spielserver sendet ein entsprechendes Ereignis an die Ereignisverwaltung des Clients.
6. Von dort wird das Ereignis an das User Interface weitergeleitet (über die Modalitätsverwaltung).

Ausgabe/Nachbedingung Der Spieler kann das Gebiet um die Kamera für einen bestimmten Zeitraum beobachten.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.5.5 Alarmanlage deaktivieren

Voraussetzung Der Spieler muss im Team von „Mister X“ sein, darf aber nicht „Mister X“ selbst sein.

Ablauf

1. Die Positionsverwaltung sendet die Position des Spielers an die Ereignisverwaltung.
2. Der Spieler erreicht den Punkt, an dem sich die Alarmanlage befindet.
3. Das Eintreffen wird an die Ereignisverwaltung des Spielservers übertragen.
4. Der Server übermittelt die Aufgabe, durch die die Alarmanlage deaktiviert werden kann, an die Ereignisverwaltung des Client.
5. Dies wird an das User Interface weitergeleitet.
6. Der Spieler gibt den Code ein.
7. Dieser wird wieder an die Ereignisverwaltung des Spielservers übertragen.
8. Das Ergebnis wird an die Ereignisverwaltung der Clients der Spieler des Teams von „Mister X“ übermittelt und von dort an das jeweilige User Interface weitergeleitet.

Ausgabe/Nachbedingung Als Ausgabe erfolgt ein Statusbericht über die Alarmanlage.
Als Nachbedingung muss gelten, dass der Status der Alarmanlage bekannt ist.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.5.5.6 Goodies sammeln

Voraussetzung Der Spieler muss einem Spiel beigetreten sein.

Ablauf

1. Der Spieleserver erzeugt ein Goodie.
2. Die Position und die Funktion des Goodies wird an den Client übermittelt.
3. Dessen Position wird auf dem Client angezeigt.
4. Wenn ein Spieler die Position des Goodies erreicht hat, wird ihm das Ereignis des Goodies gutgeschrieben oder angezeigt.
5. Bei einer Gutschreibung wird das Spielerprofil geändert.

Ausgabe/Nachbedingung Ein Spieler hat das Goodie eingesammelt und das entsprechende Ereignis bekommen.

Fazit Dieser Anwendungsfall kann durch unser Grobkonzept realisiert werden. Auch das Framework stellt die hierfür notwendigen Funktionalitäten zur Verfügung.

B.6 Kontextmodell

Die erste Version unseres Kontextmodells, die in der Technologiestudie vorgestellt wurde, beinhaltet einige Aspekte, die hier noch einmal überarbeitet worden sind. Wir haben uns bei dem hier vorgestellten Kontextmodell bemüht, den Kontext klar von den Daten zu trennen. Die Daten werden erhoben und eventuell gespeichert, müssen deshalb aber nicht unbedingt zum Kontext selbst gehören.

Was macht Informationen zu Kontext und berechtigt sie, in das Kontextmodell aufgenommen zu werden? Es wurde vereinbart Informationen genau dann Kontext zu nennen, wenn sie die Interaktion des Benutzers mit der Anwendung in einer bestimmten Situation unmittelbar beeinflussen. So sind zum Beispiel Informationen über eigene Kameras an sich nicht mehr Kontext, sondern Teil der Datenhaltung. Allerdings werden diese Informationen als Grundlage zur Bestimmung des Kontexts „sichtbare Personen“ herangezogen. Informationen über sichtbare Personen bestimmen dann unmittelbar, welche Spieler auf der eigenen Karte angezeigt werden.

Die Einflüsse und Aspekte, die den Kontext des Agentenspiels beeinflussen können bzw. zu ihm gehören, wurden in Benutzermodell, Umgebungsmodell und Anwendungsmodell unterteilt. Tabelle B.3 auf Seite 521 zeigt diese Unterteilung.

Benutzermodell	**Umgebungsmodell**	**Anwendungsmodell**
Name	Helligkeit	aktuelles Fenster
Einstellungen	Geräuschpegel	Netzanbindung
Modus	Position	
Rolle	Geschwindigkeit	
Sicherheit / Unsicherheit des Benutzers		
Aufmerksamkeit des Benutzers		

Tabelle B.3: Modelleinteilung

Zum Benutzermodell gehört alles, was den Benutzer identifiziert und vom ihm direkt oder indirekt erstellte Daten. Physische Eigenschaften zählen zum Umgebungsmodell und das Anwendungsmodell schließlich spiegelt den Zustand des laufenden Programms wieder. Aus diesen Modellen lässt sich der Kontext des Agentenspiels, der in Abschnitt B.6.1 erläutert wird, ableiten.

B.6.1 Kontextaspekte

Bei dem Kontext wird zwischen Frameworkkontext und Spielkontext unterschieden. Der Spielkontext enthält natürlich den Frameworkkontext. Des Weiteren wird unterschieden zwischen dem Basiskontext (den Daten) und dem berechneten Kontext (aus Daten berechnete Ergebnisse). Tabelle 1.2 zeigt die Kontexteinteilung.

	Framework	**Agentenspiel**
Basiskontext	Position Positionsgeschichte Geschwindigkeit(per Sensor) Helligkeit (evtl.) Geräuschpegel (evtl.) aktuelles Fenster Einstellungen(allgemein) Netzanbindung	Rolle Modus Einstellungen (spielspezifisch)
Berechneter Kontext	Sicherheit / Unsicherheit des Benutzers Aufmerksamkeit des Benutzers Qualität der Positionsinfo Verfügbarkeit der Positionsinfo räumliche Nähe zu Objekten sichtbare Objekte Geschwindigkeit (per Positionsgeschichte)	Einsatzbereitschaft der Fähigkeiten

Tabelle B.4: Kontexteinteilung

- **Position**
  Enthält Informationen über den derzeitigen Aufenthaltsort des Clients.

- **Positionsgeschichte**
  Enthält eine Ansammlung bisheriger Positionen des Clients mit zugehörigen Zeitpunkten.

- **Geschwindigkeit**
  Enthält Informationen über die aktuelle Geschwindigkeit des Clients.

- **Helligkeit**
  Informationen über die Helligkeit erlauben Rückschlüsse über eventuelle Einschränkungen visueller Interaktionsmöglichkeiten mit dem Benutzer und sind Voraussetzung für eine automatische Anpassung der Darstellung sowie der Modalität im Allgemeinen.

- **Geräuschpegel**
  Informationen über den Geräuschpegel erlauben Rückschlüsse über eventuelle Einschränkungen akustischer Interaktionsmöglichkeiten mit dem Benutzer und sind Voraussetzung für eine automatische Anpassung der akustischen Ausgabe sowie der Modalität im Allgemeinen.

- **aktuelles Fenster**
  Für die kontextsensitive Hilfe wird angegeben, welches Fenster oder welcher Bildschirm gerade von der Anwendung ausgegeben wird, beziehungsweise welche andere Modalität gerade aktiv ist.

- **Einstellungen**
  Einstellungen erlauben es dem Benutzer explizite Vorgaben zur Interaktion anzugeben. So könnte beispielsweise eine bestimmte generell unerwünschte Modalität ausgeschlossen werden.

- **Netzanbindung**
  In dynamischer Umgebung unterliegen die Kommunikationsmöglichkeiten starken Schwankungen. Hier wird angegeben, welche Kanäle momentan zur Verfügung stehen.

- **Sicherheit / Unsicherheit des Benutzers**
  Informationen über die Sicherheit des Benutzers im Umgang mit der Anwendung sind für die kontextsensitive Hilfe und eine dynamische Anpassung der Applikation an den Benutzer interessant.

- **Aufmerksamkeit des Benutzers**
  Informationen über die Aufmerksamkeit des Benutzers sind für die Wahl der Modalität wichtig. Unaufmerksame Benutzer könnten etwa zusätzlicher akustischer oder haptischer Hinweise bedürfen.

- **Qualität der Positionsinformationen**
  Erlaubt Annahmen über die Plausibilität und Genauigkeit der aktuellen Positionsinfomationen.

- **Verfügbarkeit der Positionsinformationen**
  Enthält Informationen darüber, welche Möglichkeiten zur Positionsbestimmung verfügbar sind, beziehungsweise wie aktuell die letzten Positionsinformationen sind.

- **räumliche Nähe zu Objekten**
  Enthält Informationen über die Nähe des Clients in Relation zu anderen Objekten.

- **sichtbare Objekte**
  Enthält Informationen darüber, welche Objekte für den Benutzer sichtbar sind und dementsprechend auf der Karte angezeigt werden.

- **Rolle**
  Die Rolle eines Spielers enthält dessen Teamzugehörigkeit inklusive Unterscheidung zwischen „Mister X“ und seinen Teammitgliedern, sowie zwischen neutralisiert und aktiv.

- **Modus**
  Im Modus eines Spielers wird festgehalten, ob dieser gerade getarnt ist oder den Abhörmodus aktiviert hat.

- **Einsatzbereitschaft der Fähigkeiten**
  Gibt Auskunft darüber, ob eine Fähigkeit einsatzbereit ist oder noch aufgrund der letzten Ausführung beziehungsweise bedingt durch den Modus blockiert ist.

B.7 Interaktionsdesign

B.7.1 Interaktionsdesign

Interaktionsdesign beschäftigt sich mit der Gestaltung von Benutzerschnittstellen. Der Austausch von Informationen zwischen Mensch und Maschine kann dabei auf verschiedenen Kanälen stattfinden. Die meisten Informationen werden in der Regel über die visuelle Schnittstelle ausgetauscht. Die allgemeinen Entwurfsprinzipien zielen aus diesem Grund hauptsächlich auf die Erstellung einer graphischen Benutzerschnittstelle ab. In dem Abschnitt über multimodale Benutzerschnittstellen werden deshalb zusätzlich einige Prinzipien vorgestellt, die speziell bei der Entwicklung multimodaler Schnittstellen beachtet werden sollten.

Für den Entwurf einer Mensch-Maschine-Schnittstelle gibt es keine festgelegten Regeln und Gesetzmäßigkeiten in Bezug auf ihr Verhalten, ihre Anordnung, ihr Aussehen, usw., an die sich ein Entwickler halten muss, um diese zu erstellen. Aus diesem Grund existieren Entwurfsprinzipien, an denen sich ein Entwickler orientieren kann, um eine geeignete Schnittstelle zwischen der von ihm erstellten Anwendung und den zukünftigen Benutzern zu erstellen. Im Gegensatz zu Styleguides sind Entwurfsprinzipien allgemeine Richtlinien. Durch Styleguides wird festgelegt, wie beispielsweise Schaltflächen aussehen und benutzt werden sollen. Sie sind in der Regel plattformspezifisch, und werden meist von Firmen eingesetzt, um einer Familie von Softwareprodukten ein einheitliches Aussehen zu verleihen.

B.7.2 Entwurfsprinzipien

B.7.2.1 Allgemeine Prinzipien

Informiere dich über potentielle Benutzer und ihre Aufgaben Dies ist das wichtigste aller Prinzipien, und wird an vielen Stellen über Entwurfsprinzipien an erster Stelle genannt. Die zentrale Fragestellung lautet: „Wer soll das Programm benutzen?“. „Die Kenntnis der Benutzer muss am Anfang entwickelt (und ausgebaut) werden“[1]. Der Entwickler soll sich nicht nur auf seine Intuition verlassen, sondern möglichst die Personen, die das Programm letztendlich einsetzen an dem Entwicklungsprozess beteiligen. Ist das Programm auf eine allgemeine Zielgruppe ausgerichtet, „sollte man exemplarisch einige Anwender interviewen“[2].

Hilf Benutzern, ein mentales Modell zu entwickeln Je besser ein Benutzer ein mentales Modell von der Funktionsweise eines Programms erstellen kann, desto schneller kann er dessen Bedienung erlernen und behalten. Der Grad des Lernens lässt sich hauptsächlich durch Anzahl der Fehler erkennen, die ein Benutzer bei der Bedienung eines Programms macht. Ein wichtiger Begriff ist in diesem Zusammenhang der der „Konsistenz“. Damit ist beispielsweise gemeint, „wie aus Bezeichnungen Abkürzungen gebildet werden, wie die Maus reagiert, wie Icons aussehen, wo in einem Dialog bestimmte Elemente erscheinen“ oder „der Aufbau von Dialogen (Position von Eingabefeldern und Buttons, Benennung von Buttons)“[3].

Sprich die Sprache des Benutzers Mit diesem Prinzip ist gemeint, eine Schnittstelle aus der Sicht der Anwender und nicht aus der Sicht der Entwickler zu erstellen. Entwickler verständigen sich in der Regel durch fachspezifische Begriffe, die der Anwender nicht verstehen wird. Deshalb sollte beispielsweise bei der Entwicklung von Schaltflächen darauf geachtet werden, dass deren Beschriftung eine Beschreibung

[1] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.56
[2] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.56
[3] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.57

der Funktionsweise der dahinterliegenden Programmausführung beinhaltet („Benutzer denken nicht in Menüpunkten, sondern in ihren Funktionen“[4]).

Mach Systemzustände sichtbar und unterscheidbar Dieses Prinzip betrifft die Offenlegung von Informationen über den Zustand eines Systems für einen Benutzer, der für diesen bei der Benutzung eines Programms relevant ist. Bei der Anwendung eines Textverarbeitungsprogramms beispielsweise sollte der Benutzer darüber informiert sein, in welcher Schriftart, mit welcher Schriftstärke und -eigenschaft er seinen Text verfasst. Weniger relevante Informationen hingegen sollten dem Benutzer verborgen bleiben, um dessen Aufmerksamkeit nicht von seinen wesentlichen Aufgaben abzulenken.

Verdeutliche die jeweils möglichen Aktionen Dieses Prinzip bezieht sich in erster Linie auf graphische Benutzerschnittstellen, durch die einem Benutzer sichtbar gemacht wird, welche Aktionen er durchführen kann und welche nicht (ausgegraute Schaltflächen). Eine ungeeignete Bedienung eines Programms wäre gegeben, wenn sich der Benutzer an die Mittel der Kommunikation erinnern müsste, beispielsweise an die Eingabe von kryptischen Kommandos über eine Befehlszeile. Nicht aktivierbare Schaltflächen haben den Nachteil, dass der Benutzer häufig nicht versteht, warum er eine Aktion nicht ausführen kann. Wenn keine Erklärung dazu mitgeliefert wird kann der Benutzer lediglich durch Experimentieren erfahren, unter welchen Bedingungen die Aktion wieder zur Verfügung steht. Wenn für die Eingabe in ein Formularfeld Werte erwartet werden, dann muss dem Benutzer klar sein, welcher Wertebereich zulässig ist und welche Syntax erwartet wird. Hilfreich dafür ist ein vorhandenes Angebot von Standardwerten.

Strukturiere die Benutzungsschnittstelle Bedienelemente, die aufgrund ihrer funktionalen Ähnlichkeit zu einer Gruppe zusammengefasst werden können, sollten durch ihre Positionierung diese Zusammengehörigkeit bereits ausdrücken. „Das Zusammengehören von Bedienelementen zu einer Gruppe kann vermittelt werden, indem ein geeigneter Name als Beschriftung der Gruppe eingesetzt wird (Preim / Entwicklung interaktiver Systeme)“[5]. Mit einer Menüstruktur kann dieser Effekt leicht erreicht werden, indem funktional zusammengehörige Elemente in einem Menüpunkt zusammengefasst sind. Die Gestaltung einer Benutzerschnittstelle sollte aber nicht ausschließlich nach funktionalen, sondern auch nach ästethischen Gesichtspunkten vorgenommen werden. Der Anwender soll sich bei der Benutzung eines Programms auch wohlfühlen.

Stelle eine erkennbare Rückkopplung sicher Ein Benutzer sollte, wenn er einem Programm ein Kommando erteilt hat, Informationen darüber erhalten, dass das Programm das Kommando akzeptiert hat und mit der Ausführung des Befehls beschäftigt ist. Diese Ausgabe kann über eine Statusanzeige erfolgen, über eine Veränderung von Schaltflächen oder über eine Textselektion. Dauert die Abarbeitung einer Aktion länger als ca. 10 Sekunden, so sollte der Benutzer auch auf die Dauer der Ausführungszeit hingewiesen werden.

Gestalte die Schnittstelle adaptierbar Durch dieses Prinzip sollen die Unterschiede zwischen verschiedenen Benutzergruppen hinsichtlich ihrer Erfahrungen (Anfänger, Fortgeschrittene, Profis) im Umgang mit einem System, aber auch hinsichtlich ihrer (kognitiven) Fähigkeiten und Denkstrategien berücksichtigt werden. *„(a) Denk an Anfänger, an gelegentliche und an erfahrene Benutzer“* verweist darauf, dass Anfänger eher darauf angewiesen sind, den Umgang mit einer Anwendung durch Ausprobieren zu erforschen, während erfahrene Benutzer Wert auf eine möglichst effiziente Nutzbarkeit eines Programms

[4] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.57
[5] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.59

legen. Demzufolge sollten beispielsweise Anfängern Schaltflächen angeboten werden, dessen Funktionen intuitiv verständlich sind, während erfahrene Benutzer auf komplexe, aber zeitsparende Kommandos zurückgreifen wollen. „*(b) Denk an unterschiedliche kognitive Fähigkeiten der Benutzer*“ zielt darauf ab, das Benutzer in ihren Wahrnehmungsfähigkeiten eingeschränkt sein können. Deshalb sollte beispielsweise die Schrift einer graphischen Oberfläche für Menschen mit einer Sehschwäche in ihrer Form, Farbe und Größe anpassbar sein. „*(c) Denk an unterschiedliche Umgebungen und Hardwarevoraussetzungen*“ verweist darauf, dass graphische Oberflächen auf unterschiedlichen Systemen auch unterschiedlich dargestellt werden. Deswegen sollte einem Benutzer die Möglichkeit gegeben werden, dass er sich die Schnittstelle individuell gestalten kann. Das beinhaltet beispielsweise „Bedienelemente unterdrücken oder initiieren“[6] zu können. Wichtig ist dieser Punkt allein deswegen, weil die Anwendung auf unterschiedlich großen Bildschirmen mit unterschiedlichen Auflösungen laufen kann, wodurch eine Umgestaltung des Oberfläche notwendig werden könnte. Verschiedenen Benutzern sollte ermöglicht werden, ihre individuellen Einstellungen dauerhaft zu speichern (User Profile), um auch den verschiedenen Geschmäckern der Benutzer im Umgang mit einer Anwendung gerecht zu werden.

Kombiniere visuelle Interaktion mit sprachbasierter Interaktion Dieses Prinzip besagt, dass die (durchaus redundante) Kombination von textuellen Beschreibungen mit einem Bild (Icon) zu einem besseren Verständnis für den Benutzer bei der Ausführung einer Funktion führt. Vielfach wird diese Kombination in Menü eingesetzt, eine weitere Möglichkeit besteht aber auch darin, beim Überfahren eines Icons mit der Maus einen Text nach einer kurzen zeit einzublenden (Tooltipp).

Vermeide, dass Benutzer sich zu viele Dinge merken müssen Dieses Prinzip soll daran erinnern, dass „das Kurzzeitgedächtnis nur eine sehr begrenzte Kapazität hat“[7] und der Mensch sich ca. sieben Dinge merken kann. Daher ist es ungünstig, wenn sich ein Benutzer viele einzelne Teile eines Kommandos merken muss, um eine Aktion ausführen zu können. „Der Erfolg von Fenstersystemen beruht vor allem auf der Tatsache, dass mehrere Aufgaben parallel erledigt werden können und in den dazugehörigen Fenstern jeweils der Bearbeitungszustand klar erkennbar ist“[8].

Ermögliche es, Aktionen abzubrechen und rückgängig zu machen Der Lernprozess mit dem Umgang eines Systems wird erleichtert, wenn der Benutzer Funktionen eines Programms ausprobieren und trotzdem wieder den Zustand vor der Ausführung einer Aktion herstellen kann, wenn eine Aktion nicht das vom Anwender gewünschte Ergebnis liefert (Trial-and-Error). Eine optimale Lösung ist es, wenn der Benutzer alle seine Aktionen bis zum Ausgangszustand rückgängig machen kann. „Wenn ein UnDo nicht möglich ist, sollten Benutzer gewarnt werden. Dies gilt insbesondere für das Überschreiben oder Löschen von Dateien“[9]. Bei zeitaufwendigen Aktionen sollte dem Anwender immer die Möglichkeit gegeben sein, die ausgeführte Aktion abbrechen und rückgängig machen zu können.

Erleichtere es, Fehler zu erkennen, zu diagnostizieren und zu begeben Wenn bei der Ausführung eines Programms ein Fehler aufgetreten ist, sollte dem Benutzer die Möglichkeit gegeben sein, diesen Fehler selbst zu beheben. Deshalb sollten die Meldungen über den Fehler kurz aber möglichst aussagekräftig gehalten werden. Auf keinen Fall darf die Fehlermeldung dem Benutzer einen Vorwurf machen, dass er diesen Fehler verursacht habe. Zu der eigentlichen Meldung, dass ein Fehler aufgetreten ist sollten

[6] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.64
[7] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.67
[8] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.68
[9] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.68

dem Benutzer zusätzliche Informationen und mögliche Problemlösungen (beispielsweise in einem zusätzlichen Dialog, der über eine Schaltfläche erreichbar ist) angeboten werden. erkennt das Programm, dass der Benutzer sich z.B. bei der Eingabe eines Kommandos lediglich vertippt hat, kann er direkt über eine Statusanzeige auf das korrekte Kommando aufmerksam gemacht werden. „Gerade für Anfänger ist es wichtig, dass Auswege aus der Situation angeboten werden“[10]. Der Benutzer soll bei Hinweisen durch Systemmeldungen im Mittelpunkt stehen, und nicht der Algorithmus, bei dessen Ausführung das Problem aufgetreten ist.

Vermeide es, den Benutzer zu überraschen Der Benutzer sollte, gerade wenn er unter Zeitdruck arbeitet, sich auf die Arbeitsweise einer Anwendung verlassen können und nicht durch unerwartete Reaktionen überrascht werden. Dieses Prinzip hilft dem Anwender zudem bei der Entwicklung eines mentalen Modells eines Programms. Eine mögliche Umsetzung dieses Prinzips ist beispielsweise ein Preview-Bereich in einer Anwendung, die dem Benutzer zeigt, welche Auswirkungen die Ausführung einer Aktion haben wird, noch bevor die Aktion tatsächlich ausgeführt wurde.

Beachte die wichtigsten Bedienhandlungen besonders Dieses Prinzip bezieht sich auf die Entwicklung einer Benutzerschnittstelle. Es wird herausgestellt, dass es wichtiger ist, sich auf die Entwicklung der häufigsten und wichtigsten Interaktionen zwischen Mensch und Computer zu konzentrieren, als alle Interaktionen mit gleichem Aufwand zu entwickeln. „Ähnlich wie für die Befehlssätze eines herkömmlichen CISC-Prozessors gilt für die Interaktionsmöglichkeiten die 80:20-Regel, wonach in 80 Prozent aller Anwendungsfälle nur eine Teilmenge von 20 Prozent der Interaktionsmöglichkeiten genutzt wird“[11].

Erkläre die Bedienung des Programms durch Beispiele und weniger durch Formalismen Für den Benutzer ist es oftmals hilfreicher anhand von Beispielen zu einer Lösung eines Problems zu kommen, als durch die Erläuterung der Bedeutung einzelner Parameter. Als Beispiel kann u.a. die Vorgabe von Standardwerten in Formularen dienen, die dem Benutzer verdeutlichen, in welcher Syntax Parameter erwartet werden. In diesem Zusammenhang ist eine Kombination aus formaler Erläuterung unter Bezugnahme auf ein konkretes Anwendungsbeispiel für den Benutzer eine große Hilfe.

B.7.2.2 Prinzipien für multimodale Benutzerschnittstellen

Mehrere Modalitäten müssen synchronisiert werden Aufgrund des Unterschiedes, dass Sprache zeitlich begrenzt Informationen liefert, eine graphische Ausgabe hingegen räumlich begrenzt aber dauerhaft Informationen zur Verfügung stellt, ist eine Synchronisation von gleichen Informationen notwendig, die über verschiedene Kanäle transportiert werden. Als Beispiel dafür kann ein Benutzer einen Punkt auf einer Karte markieren, wahrend er gleichzeitig ein Frage stellt, um Informationen über diesen Punkt zu erhalten. Eine Navigationssoftware sollte dem Benutzer sowohl graphisch als auch auditiv signalisieren, wenn er seine Fahrtroute ändern sollte.

Multimodale Interaktion sollte einem langsamen Qualitätsabfall unterliegen Dieses Prinzip besagt, dass eine Kommunikation zwischen zwei Menschen möglich sein sollte, selbst wenn ein oder mehrere Kommunikationskanäle nicht (oder nur eingeschränkt) zur Verfügung stehen sollten. Die Qualität der Kommunikation wird zwar abnehmen, aufgrund einer hohen Redundanz bleibt sie aber trotzdem durchführbar. In der Mensch-Maschine Interaktion sollte sichergestellt werden, dass eine Kommunikation

[10] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.70
[11] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.71

auch dann aufrechterhalten werden kann, wenn sich die Modalitäten des Menschen im Laufe der Zeit ändern sollten.

Multimodale Schnittstellen sollten einen gemeinsamen Interaktionsstatus haben Wenn ein Benutzer die Art der Ein- und Ausgabe mit einem Gerät wechselt, dann sollte der neu zu nutzende Kommunikationskanal darauf vorbereitet sein, und vorherige Interaktionen berücksichtigen. Hat ein Anwender einem Programm das Kommando erteilt, eine Liste mit Informationen auditiv auszugeben, dann sollte das Programm in der Lage sein, diese Informationen ebenfalls graphisch ausgeben zu können.

Multimodale Benutzerschnittstellen sollten sich dem Kontext des Benutzers anpassen Multimodale Benutzerschnittstellen sollten sich dem Kontext des Benutzers anpassen, um diesem zu jeder Zeit die optimale Möglichkeit zu bieten, eine Aufgabe zu lösen. Der relevante Kontext bezieht sich in diesem Zusammenhang auf die Wünsche und Fähigkeiten des Benutzers, die technischen Voraussetzungen des Gerätes, die verfügbare Bandbreite der Netzanbindung des Gerätes, die verfügbare Interaktionsbandbreite zwischen Benutzer und Gerät und weiteren Einschränkungen, die von der Aktivität des Benutzers abhängig sind.

B.7.3 Evaluation

Benutzerschnittstellen können durch verschiedene Methoden evaluiert werden. Um einen ersten Eindruck einer Schnittstelle zu erhalten werden Prototypen erstellt, über deren Handhabung mit Nutzern und Experten diskutiert werden kann. Sukzessive wird dieser Prototyp weiterentwickelt, bis er die für die Bedienung der Software gewünschten Eigenschaften aufweist. Aufbauend auf der Entwicklung der Prototypen kann die Heuristische Evaluierung durchgeführt werden. Dabei werden die Prototypen von Experten begutachtet, die beispielsweise einen unvollständigen Prototypen gedanklich eher vervollständigen können als ein Anwender, und daher Vorschläge für die weitere Gestaltung unterbreiten kann. Der Test mit den Endbenutzern wird Empirische Evaluierung genannt. Dieses Verfahren wird in den letzten Phasen der Entwicklung angewendet. Stehen mehrere Prototypen zur Diskussion kann hier die Entscheidung für eine bestimmte Benutzerschnittstelle fallen. Eine weitere Testmöglichkeit ist die Formale Evaluierungsmethode. Bei diesem Vorgehen werden die Zeiten getestet die ein Anwender braucht, um ein bestimmtes Ziel zu erreichen. Im Vorfeld sind Zeiten festgelegt, mit welcher Bearbeitungsdauer gerechnet werden kann.

Gegenüber der Heuristischen und der Empirischen Evaluierung, bei denen vor allem die subjektive Meinung von Experten und Benutzern als Bewertungsgrundlage dient, ist die Formale gekennzeichnet durch eine objektive Beurteilung einer Schnittstelle aufgrund von Messdaten. Die Aussagekraft dieser Messdaten ist aber wiederum abhängig von der Auswahl der Testpersonen, ob diese bezüglich ihrer Erfahrungen richtig eingeschätzt werden und welche Vorkenntnisse sie aufweisen. Die Auseinandersetzung mit Experten und Benutzern hat sich als wichtiges Kriterium bei der Beurteilung einer Benutzerschnittstelle herausgestellt, weil dadurch unterschiedliche Sichtweisen herausgestellt werden deren Ergebnisse sich ergänzen.

B.7.4 Adaptive Systeme

Im Gegensatz zu *adaptierbaren* Systemen, wo der Benutzer die Möglichkeit hat, Präferenzen einzustellen, geht die Initiative zur Anpassung bei *adaptiven* Systemen vom System selber aus. Dabei werden die Ein- bzw. Ausgabemöglichkeiten verändert, oder Veränderungen vorgeschlagen, um den Benutzer in seiner gegenwärtigen Aufgabe zu unterstützen. Eine solche Adaption kann den aktuellen Dialogkontext

berücksichtigen oder auch auf Daten von vergangenen Dialogen mit demselben oder ähnlichen Benutzern zurückgreifen.

Adaptiven Systemen wird in der Literatur ein gewisser Bedarf an Intelligenz zugesprochen[12]. Dazu gehören des Erkennen des aktuellen Nutzungskontextes und die Erstellung eines Benutzermodells. Benutzermodellierung sammelt Informationen über die Aktionen des Benutzers und versucht daraus Erkenntnisse abzuleiten, um Annahmen über die Wünsche des Benutzers zu machen. Ein einfaches Beispiel ist das Anzeigen der zuletzt geöffneten Datei in einem Dateiauswahldialog oder das Vorbelegen von Eingabemasken mit den in der Vergangenheit am häufigsten eingegebenen Werten.

Jedoch muss auch bei adaptiven Systemen beachtet werden, dass schon beim Entwurf der Handlungsrahmen für alle Benutzer festgelegt wird und alle möglichen Aktionen berücksichtigt werden müssen. Es hat sich in der Vergangenheit häufig gezeigt, dass Software auf ganz andere Weise eingesetzt oder bedient wird als beim Entwurf ursprünglich antizipiert. Deshalb sollte der Benutzer letztendlich die Kontrolle über adaptive Prozesse behalten und immer die Möglichkeit haben diese Teilweise oder gänzlich Auszuschalten. Außerdem sollte der Benutzer informiert werden ob und in welchem Umfang ein Benutzermodell angelegt wird.

[12] Vgl. Preim, „Entwicklung interaktiver Systeme“, 1999, S.467

B.8 Informationsdesign

B.8.1 Vermittlung des aktuellen Zustands

Ein Spieler kann sich in diversen Zuständen befinden, wie z.B. getarnt sein. Da unser Ziel die Entwicklung multimodaler Benutzungsschnittstellen beinhaltet, sollte der aktuelle Zustand des Benutzers in beliebigen Modalitäten ausgegeben werden können. Dabei tritt das Problem auf, wie viele unterschiedliche Zustände kann ein Mensch in den verschiedenen Modalitäten unterscheiden kann.

- **Visuell**: Der Mensch ist in der Lage visuelle Informationen selektiv wahrzunehmen. Darum können theoretisch beliebig viele Teilaspekte eines Zustands gleichzeitig vom System dargestellt werden, es ist jedoch nicht garantiert, dass der Benutzer sie auch wahrnimmt. Der Sehsinn ist jedoch bei den meisten Menschen der am besten entwickelte Sinn und wird deshalb in der Mensch-Computer-Interaktion primär zur Übermittlung von Informationen benutzt.

- **Akustisch**: Bei gleichzeitig erklingenden akustischen Signalen kann der Mensch nur eine begrenzte Anzahl auseinander halten bzw. identifizieren. Das liegt zum einen daran, dass es dem Mensch schwieriger fällt, sich speziell auf einzelne Klänge zu konzentrieren, da der Hörsinn oft weniger trainiert ist, als der Sehsinn. Spezielle Gruppen wie z.B. sehbehinderte Menschen können aber eventuell genügend trainiert sein, um deutlich mehr Zustände unterscheiden zu können. Ein weiteres Problem ist, dass man akustische Signale nicht physikalisch ausblenden kann, wie man es mit Teilen einer grafischen Anzeige durch Verändern des Sichtfeldes tun kann. Man muss also aufpassen, den akustischen Kanal nicht zu überladen.

- **Haptisch**: Das Potential des Tastsinns liegt darin, dass er im Gegensatz zum Seh- oder Hörsinn nicht über einen „Punkt“ wie die Augen oder die Ohren, sondern über die gesamte Körperoberfläche des Menschen wahrgenommen wird. Informationen können also nicht nur durch Stärke, Richtung und Art der Kontaktoberfläche einer Berührung, sondern auch durch die Stelle auf der Benutzer-Oberfläche unterschieden werden. Im Gegensatz zu Akustik und Optik gibt es bei der Haptik keine Sprache. Es gibt auch viel weniger allgemeingültige Symbole (Vgl.: Klopfen an der Tür oder das Fragezeichen), was die gezielte Übermittlung von Informationen schwieriger macht. Haptische Signale kann der Mensch wie akustische Signale nicht physikalisch ausblenden.

Das Framework sollte auch die Möglichkeit bieten, Zustände durch beliebige Modalitäten zu vermitteln. Da aber sowohl haptische als auch akustische Signale vom Menschen nicht ausgeblendet werden können, zur Darstellung eines Zustands allerdings dauerhaft gesendet werden müssten, besteht die Gefahr von Reizüberflutung und Stress. Wir beschränken uns beim Agentenspiel daher für die Ausgabe des aktuellen Zustands auf visuelle Elemente, da diese ausgeblendet werden, wenn das Display des mobile Endgerät sich nicht im Blickfeld des Benutzers befindet. Es besteht keine Gefahr einer permanenten Reizüberflutung.

B.9 Fazit

Dieses Dokument soll der Gruppe als Vorstudie und Vorbereitung auf die folgenden Arbeitspakete „Technologiestudie“ und „Entwurf“ dienen. Die zur Umsetzung des Agentenspiels und des Frameworks einzusetzenden Technologien, sollen anhand der Struktur und Beschreibungen der Module identifiziert werden. Die Untersuchungen und Vorstellung der Technologien können als Grundlage für die folgenden Arbeiten zum Arbeitspaket „Entwurf“ verwendet werden. Die Arbeiten zu den nichtfunktionalen Anforderungen in den Kapiteln fünf bis acht sollen ebenfalls als Basis für das Arbeitspaket „Entwurf“ dienen. Diese Ergebnisse sollen in die Gestaltung einzelner Programmteile (Kontextmodell) sowie in die Gestaltung der multimodalen Nutzungsschnittstelle (Interaktionsdesign / Informationsdesign) eingehen.

Anhang C

Webseite

Projektbegleitend haben die Telinehmer der Projektgruppe eine Internetpräsenz[1] erstellt, welche die Projektgruppe nach außen präsentieren soll.

Die Webseite ist generell in folgende 4 Bereiche aufgeteilt:

- Startseite
- Projekt
- Das Spiel
- Über uns
- Impressum

Startseite In Abbildung C.1 auf Seite 534 ist die Startseite der Webseite abgebildet. Diese Seite gibt einen groben Überblick über die Organisation, den Inhalt und die Aufgabenstellung der Projektgruppe.

[1] erreichbar unter http://hurrikan.informatik.uni-oldenburg.de:20120/index.php

Abbildung C.1: Die Startseite der Webseite

Projekt Diese Seite ist aufgeteilt in die beiden Punkte *„Aufgabenstellung und Zielsetzung“* und *„Motivation“*. Ersterer beschreibt die grundlegende Aufgabenstellung und den Ablauf der ersten Monate innerhalb der Projektgruppe.
Unter dem Punkt *„Motivation“* werden die Motivation zur Umsetzung der Aufgabenstellung durch ein Spiel sowie die Ziele, die durch die Spielidee erreicht werden sollen, beschrieben.

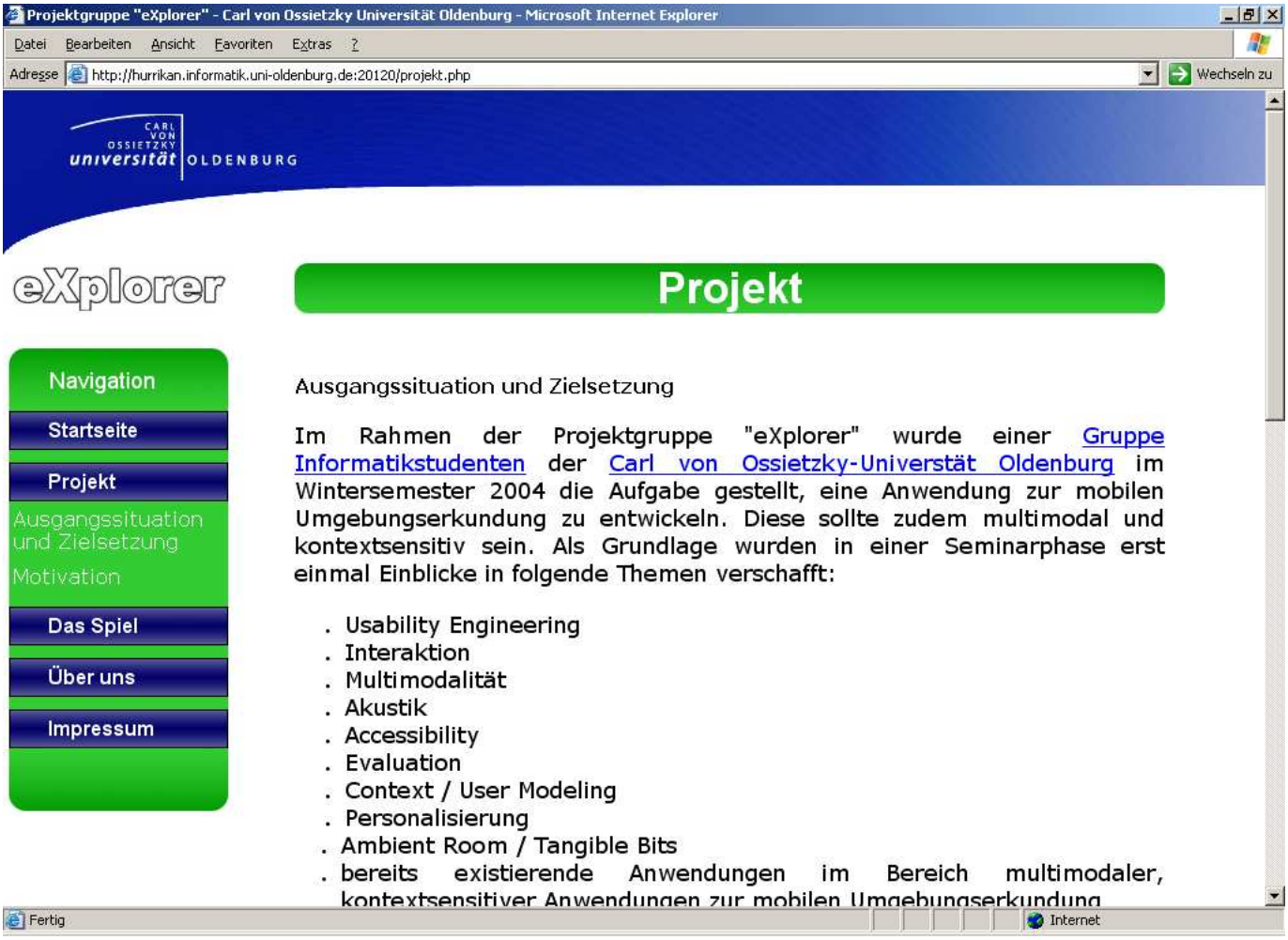


Abbildung C.2: Ein Ausschnitt der Seite zur Beschreibung des Projekts

Das Spiel Der Aufbau dieser Seite ist aufgeteilt in *„NABB-Story“*, *„Spielidee“* und *„Spielablauf“*. Unter dem ersten Punkt wird eine Hintergrundgeschichte zum Agentenspiel erzählt, die Menschen für das Spiel interessieren und an dieses fesseln soll.
„Spielidee“ erläutert die grundlegende Spielidee und die Verknüpfung von virtuellen Elementen mit der realen Welt.
Unter dem Punkt *„Spielablauf“* wird die Spielidee so weit verfeinert, dass die einzelnen Funktionalitäten des Spiels anhand des Spielablaufs erklärt werden.

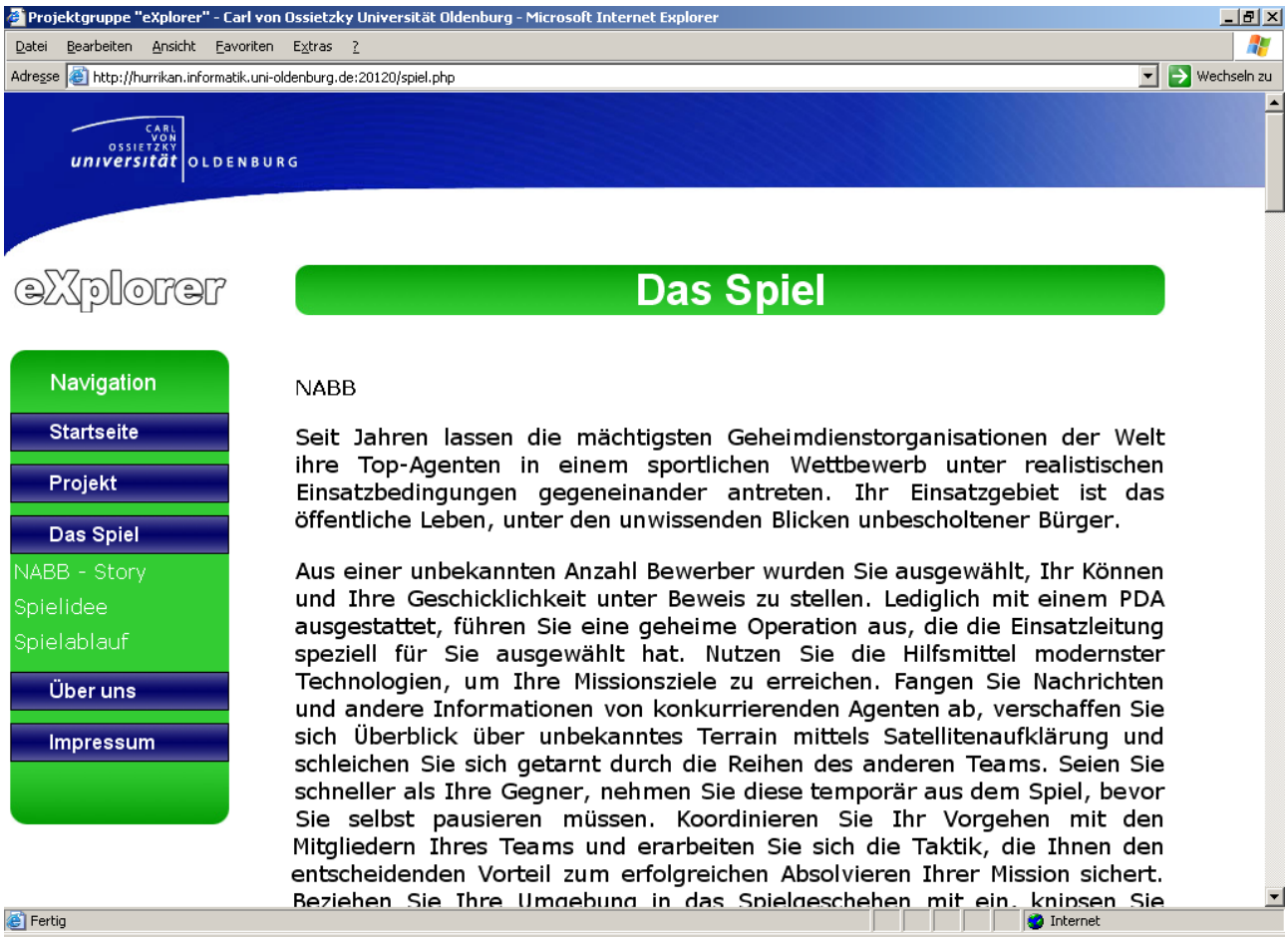


Abbildung C.3: Ein Ausschnitt der Seite zur Spielbeschreibung

Über uns Auf dieser Seite werden die Teilnehmer und Betreuer der Projektgruppe aufgelistet und es gibt ein Gruppenfoto, das beim Projektgruppenfussballturnier aufgenommen wurde.

Die Boll´sche Wisten

Obere Reihe (v.L.n.R.): Michael Onken, Frank Jellinghaus, Wilko Heuten, Daniel Nüss, Steffen Kruse, Olaf Lehde

Untere Reihe (v.L.n.R.): Susanne Boll, Stefan Andreßen, Arne Bartels, Jens Peternel, Martin Pielot

Fehlend: Palle Klante

(Aufgenommen am 17.08.2005 beim Projektgruppen-Fussballturnier des Departments für Informatik - weitere Bilder von den Boll´sche Wisten und vom Turnier gibt es hier [ca. 74 MB] zum Download)

Abbildung C.4: Ein Ausschnitt der Seite über die Projektgruppenteilnehmer

Impressum Diese Seite enthält rechtliche Hinweise sowie verschíedene Kontaktmöglichkeiten zur Projektgruppe.

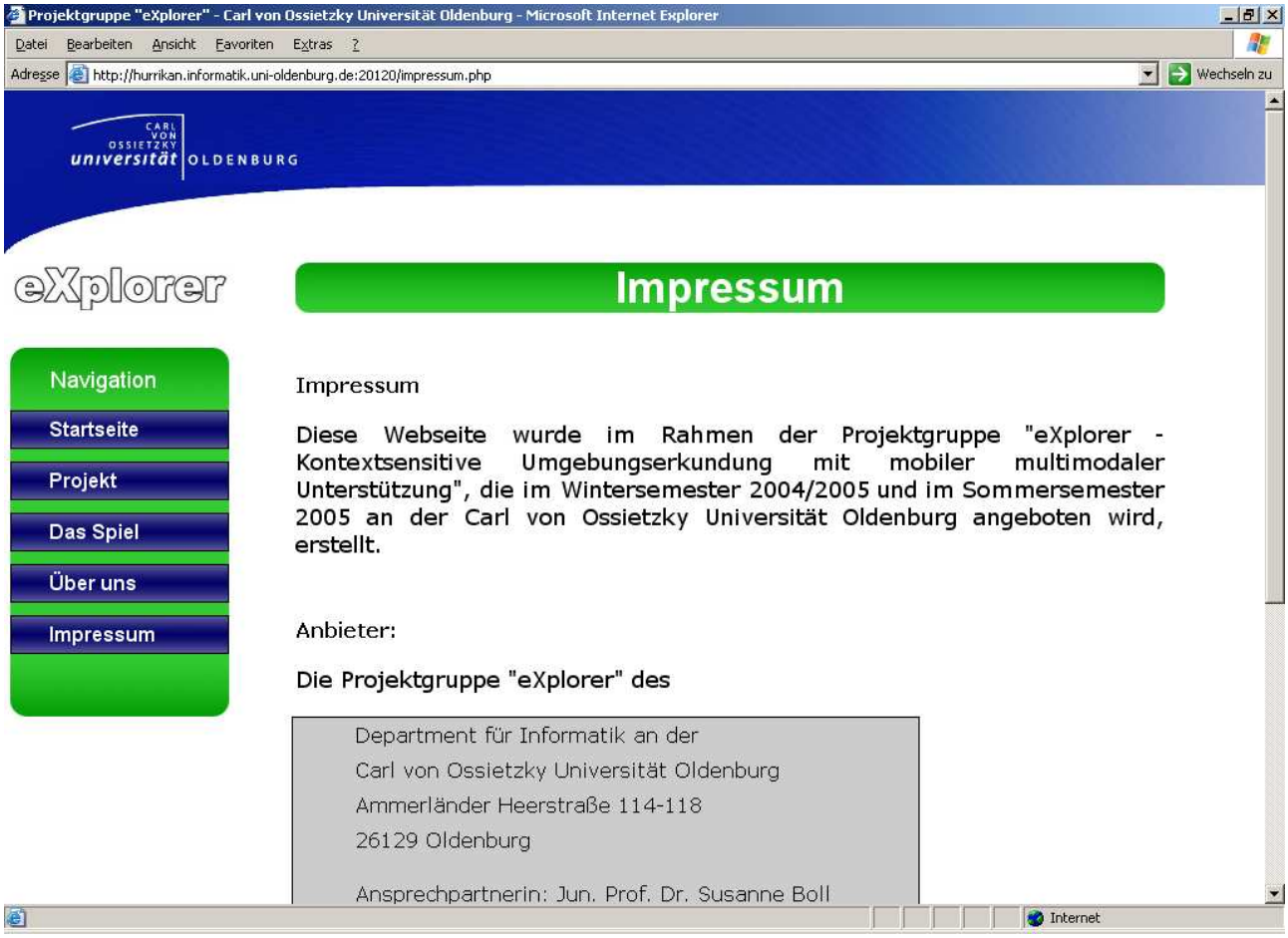


Abbildung C.5: Ein Ausschnitt des Impressums der Webseite

Anhang D

Loconex-Flyer

Der nachfolgende Flyer wurde von den Teilnehmern der Projektgruppe in der Absicht verfasst, die Arbeit der Projektgruppe nach außen zu repräsentieren. Dadurch wurde u.a. versucht, mögliche Sponsoren für Hardware o.ä. anzuwerben.

OLDENBURGER FORSCHUNGS- UND ENTWICKLUNGS-INSTITUT FÜR INFORMATIK -WERKZEUGE UND -SYSTEME

LoconeX

location-based, context-aware exploration

Designstudie eines Spiels auf Basis von ortsbasierter, kontext-sensitiver Umgebungserkundung

Die steigende Verbreitung ausgereifter Kommunikationstechnologie ermöglicht die Umsetzung neuartiger Spielkonzepte. Ziel dieses Projekts ist die Entwicklung eines Spiels, das die reale Welt mit virtuellen Inhalten verbindet. Die Spieler kommunizieren untereinander und interagieren mit dem Spiel mittels mobiler Endgeräte. Eine Besonderheit der Anwendung ist die Auswahl der Interaktionsmodalitäten anhand des Nutzerkontextes.

Das Spiel

Das Spielfeld ist die reale Welt. Die Spieler bilden Teams, die sich innerhalb einer vorgegebenen Region frei bewegen können. Dabei kann es sich je nach Szenario um den Campus einer Universität oder auch die belebte Fußgängerzone einer Innenstadt handeln. Mobile Endgeräte vermitteln den Spielern wichtige Informationen über die Umgebung, wie z.B. geographische Daten und die Positionen anderer Spieler. Im Verlauf des Spiels werden Aufgaben gestellt, die nur durch Kommunikation und Kooperation innerhalb der Teams gelöst werden können. Gleichzeitig bietet das Spiel diverse Möglichkeiten, die jeweils anderen Teams am Erreichen der Spielziele zu hindern.

Umgebungserkundung

Mobile Anwendungen zur Navigation in realen Umgebungen gewinnen immer mehr an Bedeutung. Eine Basisfunktionalität des Spiels ist die Darstellung der aktuellen Positionen eines Spielers, seiner Mitspieler und Spielobjekten auf einer Karte. Verwendet werden dazu Geographische Informationssysteme (GIS) in Verbindung mit GPS-Informationen. Spielobjekte sind dabei virtuell, aber an einen realen Ort gebunden. Das Spiel erlaubt es den Spielern mit den virtuellen Objekten zu interagieren und deren Zustand oder sogar deren Position zu beeinflussen.

Kontextsensitivität und Multimodalität

Bei der Benutzung einer Anwendung im mobilen Umfeld sind besondere Schwierigkeiten zu beachten: Der mobile Einsatz fordert vom Benutzer eine erhöhte Aufmerksamkeit, da sowohl mit der mobilen Anwendung interagiert werden muss, als auch eine erhöhte Konzentration zur Bewegung in der Umwelt erforderlich ist. Die kognitiven Ressourcen sind entsprechend eingeschränkt.
Die Auswahl dieser sollte aufgrund von Profiling-Informationen erfolgen, um ein möglichst konkretes und treffendes Angebot zu finden. Neben der Auswahl der Informationen sind auch die Art der Präsentation und die Möglichkeiten zur Interaktion sehr wichtig. Die verwendete Modalität muss der aktuellen Situation angemessen sein. Neben der Position des Benutzers muss also auch der Nutzungskontext beachtet werden. Um eine sinnvolle Darstellungsmodalität ermitteln zu können, erfasst das Spiel den Kontext des Spielers und bereitet die Informationen entsprechend auf.

Pervasive Gaming

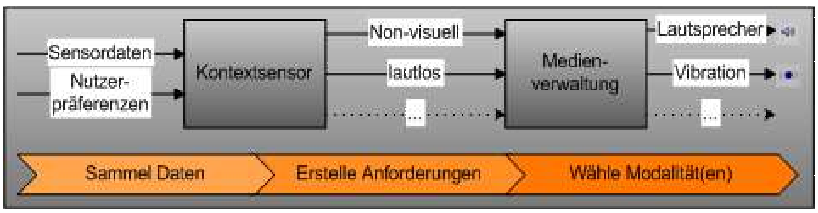
Pervasive Gaming ist das neuste Schlagwort im Bereich mobiler Spielentwicklung. Dabei handelt es sich um Spiele, die virtuelle Elemente mit der realen Welt verknüpfen. Spieler agieren in beiden Ebenen und bilden Teams um Aufgaben zu lösen. Vorangetrieben wird diese Entwicklung durch die steigende Verbreitung neuester Kommunikationstechnologien.

Realisierung

Grundlage für das Spiel ist ein Framework, dessen Entwicklung Teil dieses Projekts ist. Das Framework soll die Erstellung mobiler Anwendungen mit den Schwerpunkten kontext-sensitiver Umgebungserkundung, Kommunikation mobiler Endgeräte untereinander und multimodaler Nutzerinteraktion unterstützen.

Schlüsseltechnologien

Für das Spiel sollen die Standards GPRS und UMTS zum Einsatz kommen, weil diese flächendeckend verfügbar und für den mobilen Einsatz konzipiert sind. Zur Unterstützung können temporäre Personal Area Networks (PAN) über Bluetooth aufgebaut werden. Die Ortsbestimmung soll mittels GPS erfolgen, da keine zusätzliche Infrastruktur aufgebaut werden muss. Hierbei wird ein generisches Kontextmodell entwickelt, aus dem die Anforderungen an die Modalität einer Interaktion abgeleitet werden kann.



Wahl der Darstellungsmodalitäten abhängig vom Nutzerkontext

Ansprechpartner:
Jun. Prof. Dr. Susanne Boll

Telefon: 0441 9722-213
E-Mail: boll@offis.de

OFFIS
Escherweg 2
26121 Oldenburg
Telefon: 0441 9722-0
WWW: http://www.offis.de/

Neben dem Bereich „Multimedia und Internet-Informationsdienste" hat OFFIS vier weitere FuE-Bereiche:

BI IG HS SC

Abbildung D.1: Der Flyer

Anhang E

Inhalt der DVD

Dieses Kapitel beschreibt den Inhalt der zu diesem Abschlussbericht gehörenden DVD. Der Datenträger enthält folgende Ordner:

- **Binaries:** enthält die ausführbaren Dateien der im Rahmen dieses Projekts erstellten Programme. Das sind:
  - GPS: Prototyp zum Testen der GPS-Daten Auswertung
  - NABB Client: Client des Agentenspiels NABB
  - NABB Server: Server des Agentenspiels NABB
  - Stundenzettelauswertung: Programm zur Interpretation der Stundenzettel
- **Entwicklungswerkzeuge:** enthält die verwendeten freien Entwicklungswerkzeuge, die man neben Visual Studio .NET benötigt, um Anwendungen für das .NET Compact Framework zu erstellen. Die Anleitung, wie diese Programme einzurichten sind, befinden sich in der Datei *Installation der Entwicklungsumgebung.doc*.
- **Fotos:** enthält Fotos, die zu besonderen Anlässen der Projektgruppe gemacht wurde, wie z.B. die Projektgruppenfahrt und das Fußballturnier.
- **Repository:** enthält das CVS Repository der Projektgruppe. Im einzelnen sind dort gespeichert:
  - Arbeitspakete: die Quelldateien zu den Dokumenten der einzelnen Arbeitspakete
  - Ergebnisdokumente: die Dokumente zu den einzelnen Arbeitspaketen als PDFs oder PowerPoint Folien
  - Organisatorisches: speichert Protokolle, Tagesordnungen, Stundenzettel, Wochenstatistiken, Urlaubsplan und weitere organisatorische Dokumente
  - Software: enthält den Quellcode der im Rahmen dieses Projekt entstandenen Programme. Dies sind im einzelnen:
    * *GPS:* Prototyp zum Testen der GPS-Daten Auswertung
    * *NABB:* Client und Server des Agentenspiels NABB
    * *Stundenzettelauswertung:* Programm zur Interpretation der Stundenzettel

Anhang F

Klassenbescheibung

F.1 Namespace Client.Communication

F.1.1 Schnittstelle ClientCommunicationObserver

Interface für alle Klassen, die den Communication Layer observieren möchten.

Methoden

`public` **void Disconnected**( )

Wird aufgerufen, wenn die Verbindung zum Server unterbrochen wurde. Events, in der Warteschlange zur Versendung an den Server werden erst gesendet, wenn die Verbindung wieder hergestellt ist.

`public` **void KickedFromServer**( )

Wird genau dann aufgerufen, wenn der Server die Verbindung beendet hat.

F.1.2 Klasse ConcreteNetworkManager

Dieses Modul ist die zentrale Verwaltungsstelle der Kommunikationslogik auf Seiten des Clients und organisiert die Datenübertragung zwischen dem Client und einem Server. Initial werden Informationen über die Arten der Verbindungen zwischen Client und Server (es können verschiedene Verbindungen bestehen, die über Plugins hergestellt werden) ausgetauscht. Zudem werden Methoden zur Verwaltung der Plugins bereitgestellt. Desweiteren stellt dieses Modul den Mechanismus bereit, um Objekte an einen Server zu übertragen.

Konstruktoren

`public` **ConcreteNetworkManager**( )

Erzeugt ein neues ConcreteNetworkManager-Objekt.

Parameter

- `clientApplication` - Referenz auf die (Client-)Applikation

Methoden

`public` **void Attach**( )

Fügt eien neuen ClientCommunicationObserver der Liste hinzu.

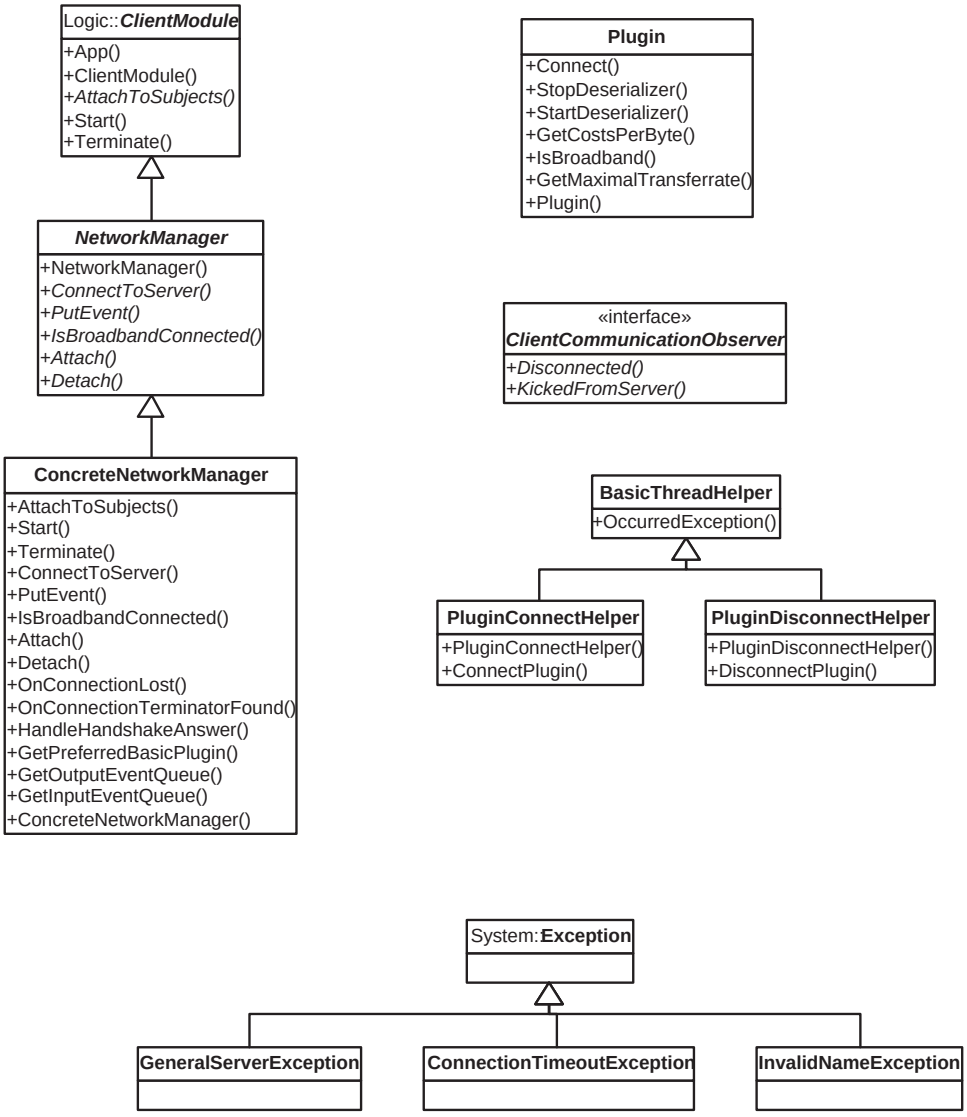


Abbildung F.1: UML Struktur des Namensraum Client.Communication

Parameter

- `observer` - der neue Observer auf die ClientCommunication

`public` **void AttachToSubjects**( )

Diese Methode muss aus der abstrakten Klasse NetworkManager überschrieben werden. Sie erfüllt in dieser Klasse jedoch keine Aufgabe.

`public` **void ConnectToServer**( )

Verbindet die Plugins für ein UserProfile mit dem Server.

Parameter

- `userProfile` - enthält Informationen über die Verbindung zwischen Client und Server

`public` **void Detach**( )

Entfernt einen ClientCommunicationObserver aus der Liste der Observer.

Parameter

* `observer` - zu entfernender Observer auf die ClientCommunication.

`public` **Common.Logic.EventQueue GetInputEventQueue**( )

Getter für die interne Queue (diese speichert die Events, die an den Server versendet werden sollen).

`public` **Common.Logic.EventQueue GetOutputEventQueue**( )

Getter für die extern Queue (diese speichert die Events, die vom Server kommen).

`public` **Common.Communication.BasicPlugin GetPreferredBasicPlugin**( )

Liefert eine ausgewähltes BasicPlugin. Ist vermutlich immer nur das GPRS-Plugin.

`public` **void HandleHandshakeAnswer**( )

Entscheidet anhand des HandshakeAnswer-Objekts (HandshakeAnswer.Flag), ob die Anmeldung erfolgreich war und die clientId gesetzt werden kann.

Parameter

* `handshakeAnswer` - die Antwort des Servers

`public` **bool IsBroadbandConnected**( )

Gibt an, ob ein Breitbandanschluss verfügbar ist.

`public` **void OnConnectionLost**( )

Wird vom Serializer bzw. Deserialiser über den C-Sharp-Eventmechanismus aufgerufen, wenn die Verbindung unterbrochen wurde, also bspw. wenn beim Lesen oder Schreiben des internen Sockets ein Fehler aufgetreten ist.

Parameter

* `sender` - der Aufrufer
* `plugin` - das Plugin, das die Verbindung verloren hat

`public` **void OnConnectionTerminatorFound**( )

Wird von einem Deserializer aufgerufen, wenn er ein ConnectionTerminator-Objekt deserialisiert hat. Der Server bestätigt vermutlich den Verbindungsabbruch, theoretisch ist es aber auch möglich, dass er einen Abbruch sebst initiiert. In diesem Fall ist das Flag des ConnectionTerminator-Objekts auf Closing gesetzt, sonst auf ClosingConfirm.

Parameter

* `sender` - der Aufrufer
* `connectionTerminator` - das ConnectionTerminator-Objekt

`public` **void PutEvent**( )

Fügt ein neues Event in die interne ClientEventQueue ein. Dieses wird daraufhin (bei Zeiten) serialisiert und dadurch an den Server gesendet.

Parameter

* `evt` - Das zu versendende Event.

`public` **void Start**( )

Hier wird der ConcreteNetworkManager gestartet, es werden Initialisierung vorgenommen und versucht eine Verbindung zum Server aufzubauen.

`public` **void Terminate**( )

Beendet den ConcreteNetworkManager und alle damit assoziierten Threads.

F.1.3 Klasse NetworkManager

Die abstrakte Klasse NetworkManager legt Methoden fest, die eine von NetworkManager abgeleitete Klasse implementieren muss. NetworkManager-Objekte stellen Mechanismen für den Verbindungsaufbau zwischen einem Client und einem Server sowie für das Versenden von Objekten bereit. Zudem kann über die Methode IsBroadbandConnected() abgefragt werden, ob die Verbindung eine Breitbandverbindung ist. Über die Methoden Attach() und Detach() werden Observer auf ein NetworkManager-Objekt registriert bzw. deregistriert.

Konstruktoren

`public` **NetworkManager**( )

Erzeugt ein neues NetworkManager-Objekt mit der übergebenen Referenz auf ein ClientApplication-Objekt.

Parameter

- `app` - Referenz auf die ClientApplication

Methoden

`public` **void Attach**( )

Über diese Methode sollen Module als Observer auf einen NetworkManager registriert werden.

Parameter

- `observer` - zu registrierender Observer

`public` **void ConnectToServer**( )

Über diese Methode soll eine Verbindung eines Clients zu einem Server hergestellt werden. Dazu notwendige Informationen werden in dem als Parameter übergebenen UserProfile-Objekt mitgeliefert.

Parameter

- `userProfile` - enthält Informationen über die aufzubauende Verbindung zwischen Client und Server.

`public` **void Detach**( )

Über diese Methode sollen Module, die nicht mehr länger Observer auf einen NetworkManager sein sollen, als Observer entfernt.

Parameter

- `observer` - zu entfernender Observer

`public` **bool IsBroadbandConnected**( )

Über diese Methode soll abgefragt werden können, ob die Verbindung zwischen einem Client und einem Server eine Breitbandverbindung ist.

`public` **void PutEvent**( )

Durch diese Methode soll das Versenden von Objekten von einem Client an einen Server realisiert werden.

Parameter

* `inputEvent` - zu versendendes Event-Objekt

F.1.4 Klasse Plugin

Basisklasse für alle konkreten PLugins (z.B. GPRS, WLAN, UMTS, etc.). Es wird die Verbindungslogik eines Plugins implementiert (z.B. über welche IP und welchen Port eine Verbindung zum Server hergestellt werden soll) und der Deserializer-Thread gestartet bzw. gestoppt. Die Erstellung neuer Plugins wird über PluginIDs verwaltet.

Konstruktoren

`public` **Plugin**( )

Erzeugt ein neues Plugin-Objekt mit dem als Parameter übergebenen ConnectionUtilitiesProvider-Objekt.

Parameter

* `connectionUtilitiesProvider` - das ConnectionUtilitiesProvider-Objekt

Methoden

`public` **void Connect**( )

Diese Methode stellt eine Verbindung des Plugins mit dem Server her.

Parameter

* `ip` - die Ip des Servers
* `port` - der Port auf dem der Server wartet
* `profile` - das für den Handshake zu sendende Profil

`public` **double GetCostsPerByte**( )

Deprecated.

`public` **int GetMaximalTransferrate**( )

Deprecated.

`public` **bool IsBroadband**( )

Deprecated.

`public` **void StartDeserializer**( )

Startet den Deserialisierungprozess dieses Plugins.

`public` **void StopDeserializer**( )

Stoppt den Deserialisierungsprozess dieses Plugins.

F.2 Namespace Client.Logic

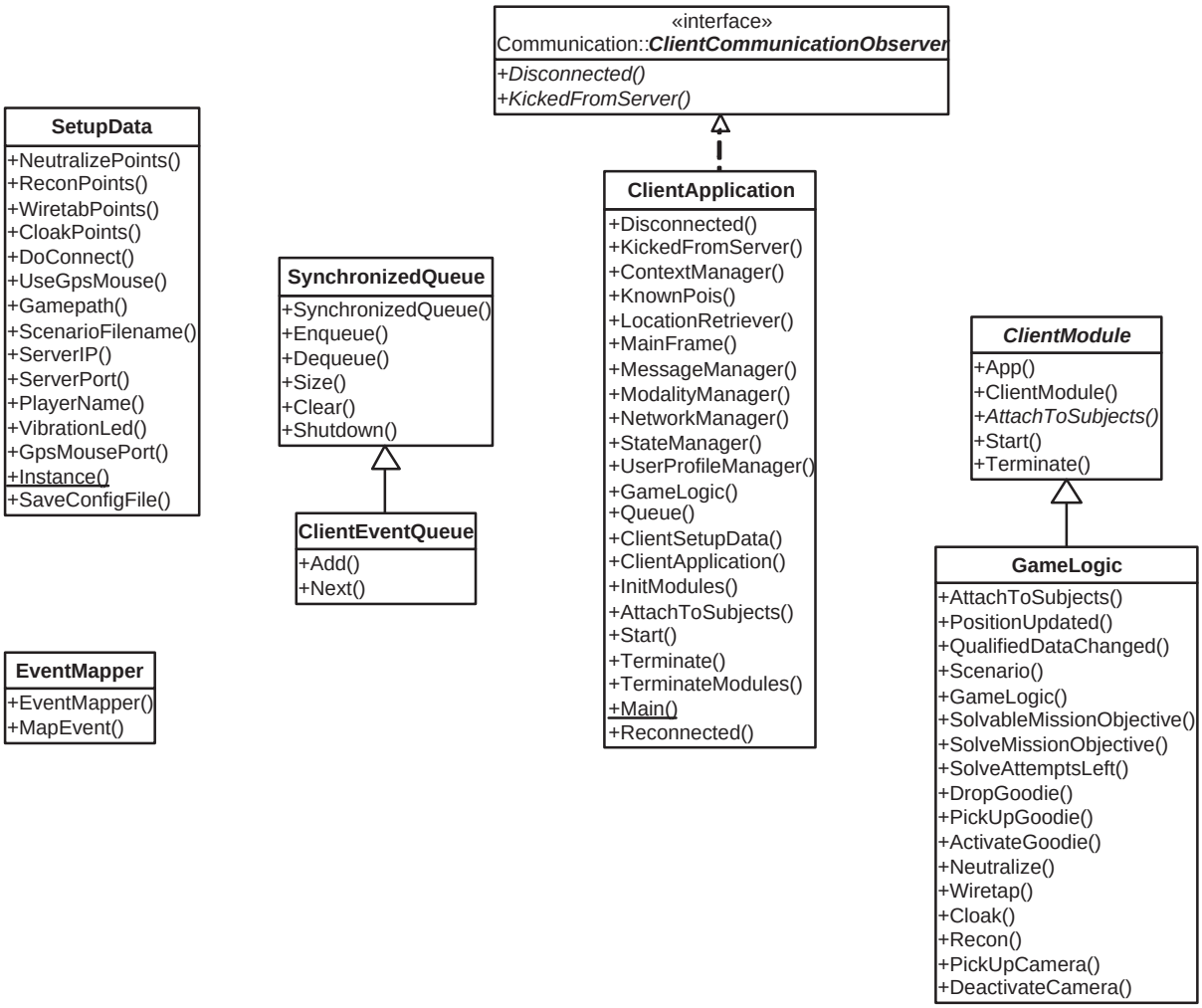


Abbildung F.2: UML Struktur des Namensraum Client.Logic

F.2.1 Klasse ClientApplication

Dieses Modul ist das Hauptmodul des Clients, das alle anderen Module erzeugt und koordiniert. Es speichert die Daten, mit denen die Anwendung initialisiert werden soll. Zusätzlich arbeitet es die Ereignisse ab und verteilt sie auf die entsprechenden Module.

Konstruktoren

`public` **ClientApplication**( )

Erzeugt eine neue ClientApplication.

Methoden

`public` **void AttachToSubjects**( )

Diese Methode wird im Konstruktor der Klasse ClientApplication nach dem Aufruf von InitModules() aufgerufen. Konkrete Applikationen sollen in dieser Methode, ihre Module als Observer bei den gewünschten anderen Modulen registrieren.

`public` **void Disconnected**( )

wird aufgerufen, wenn die Verbindung zum Server unterbrochen wurde. Events, in der Warteschlange zur Versendung an den Server werden erst gesendet, wenn die Verbindung wieder hergestellt ist.

`public` **void InitModules**( )

Diese Methode wird im Konstruktor der Klasse ClientApplication aufgerufen. Konkrete Applikationen sollen in dieser Methode alle Module initialisieren.

`public` **void KickedFromServer**( )

Wird genau dann aufgerufen, wenn der Server die Verbindung beendet hat.

`public` **void Main**( )

Eintrittspunkt der Anwendung

`public` **void Reconnected**( )

Wird aufgerufen, wenn eine unterbrochene Verbindung wieder hergestellt ist.

`public` **void Start**( )

Startet die Anwendung. Diese Methode muss in der Main() Methode der konkreten Applikation aufgerufen werden. In dieser Methode, sollen alle Threads, die zu der Applikation gehören, gestartet werden, und die GUI soll angezeigt werden.

`public` **void Terminate**( )

Beendet die Anwendung. Diese Methode soll aufgerufen werden, wenn das Programm durch den Nutzer (oder einen nicht behebbaren Fehler) beendet wird. Nach dem Aufruf dieser Methode sollen alle Threads beendet werden, und es sollen alle weiteren verwendeten Ressourcen freigegeben werden

`public` **void TerminateModules**( )

Beendet die Module des Clients. Die Methode wurde ausgelagert, damit sie aus dem EventMappingThread des Clients aufgerufen werden kann. So ist sichergestellt, dass der Client keine Events mehr verarbeitet, wenn die Module terminiert werden.

F.2.2 Klasse ClientEventQueue

Diese Klasse ist eine Implementierung der Schnittstelle EventQueue. Die Implementierung ist Thread-sicher. Es können mehrere Threads gleichzeitig auf die Add(Event) und Next() Methoden zugreifen.

Konstruktoren

`public` **ClientEventQueue**( )

Methoden

`public` **void Add**( )

Hängt der Warteschlange ein Event an.

Parameter

* `e` - Anzuhängendes Event

`public` **Common.Logic.Statemanager.Event Next**( )

Holt das vorderste Event aus der Warteschlange heraus. Ist gerade kein Event in der Warteschlange, blockiert die Methode, bis entweder ein Event zur Verfügung steht, oder die Methode Shutdown() aufgerufen wurde. In letzterem Fall liefert diese Methoden NULL statt eines Events zurück.

F.2.3 Klasse ClientModule

Abstrakte Oberklasse für alle Module des Clients. Über diese Klasse steuert der Client den Lifecycle der einzelnen Module.

Konstruktoren

`public` **ClientModule**( )

Initialisiert die Felder des Moduls, die zu der abstrakten Oberklasse gehören

Parameter

* `app` - Referenz auf den Client

Methoden

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll.

`public` **void Terminate**( )

Diese Methode wird von der Applikation aufgerufen, wenn sie sich beendet. Jedes Modul kann sie überschreiben, um vor dem Beenden z.B. belegte Ressourcen freizugeben.

F.2.4 Klasse GameLogic

Alle Aufrufe von Spielfunktionalitäten wir z.B. Tarnen oder Neutralisieren werden über diese Klasse ausgelöst. Die Aufgabe des GameLogic Moduls ist, Aufrufe bereits auszusortieren, wenn sie nach Kenntnisstand des Clients vom Server abgelehnt werden würde. Will z.B. ein Spieler seine Abhör-Fähigkeit aktivieren, ist aber gerade getarnt, ist das eine unzulässige Anfrage. Um das Netzwerk zu entlasten, wird die Anfrage bereits in diesem Modul aussortiert, und gar nicht erst an den Server geschickt.

Konstruktoren

`public` **GameLogic**( )

Erzeugt neues Spiellogik Modul.

Parameter

* `app` - Referenz auf die zugehörige Anwendung

Methoden

`public` **void ActivateGoodie**( )

Wird aufgerufen, wenn der Spieler ein Goodie aktiviert.

Parameter

* `goodie` - Das aktiviert GoodieItem

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void Cloak**( )

Wird aufgerufen, wenn Tarnen aktiviert werden soll.

`public` **void DeactivateCamera**( )

Wird aufgerufen, wenn eine gegenerische Kamera deaktivert werden soll.

Parameter

* `camera` - die Kamera, die deaktiviert werden soll

`public` **void DropGoodie**( )

Wird aufgerufen, wenn der Spieler ein Goodie fallen lässt.

Parameter

* `goodie` - Das abzulegende Goodie

`public` **void Neutralize**( )

Wird aufgerufen, um Neutralisieren zu starten.

`public` **void PickUpCamera**( )

Wird aufgerufen, wenn eine Kamera aufgenommen werden soll.

Parameter

* `camera` - die Kamera, die aufgenommen werden soll

`public` **void PickUpGoodie**( )

Wird aufgerufen, wenn der Spieler ein Goodie aufnimmt.

Parameter

* `goodie` - Das aufzunehmende Goodie

`public` **void PositionUpdated**( )

Wird aufgerufen, wenn der Client seine Position geändert hat. Die Positionsdaten umfasst Längengrad, Breitegrad und Höhe der aktuellen Spielerposition. Die Änderung der qualifizierten Daten, wie z.B. der Positionqualität wird über die Methode QualifiedDataChanged(QualifiedPosition) bekanntgegeben.

Parameter

* `position` - Positionsobjekt, das die aktuelle Spielerposition enthält

`public` **void QualifiedDataChanged**( )

Wird aufgerufen, wenn sich die qualifizierten Positionsdaten geändert haben. Die qualifizierten Daten umfassen die Qualität der Positionsdaten sowie Bewegungsrichtung und Bewegungsgeschwindigkeit des Spielers. Diese Methode wird jedoch nicht aufgerufen, wenn sich die geographische Position des Spielers geändert hat. In diesem Fall wird die Änderung über die Methode PositionUpdated(Position) bekanntgegeben.

Parameter

* `qualifiedPosition` -

`public` **void Recon**( )

Wird aufgerufen, wenn Aufklären gestartet werden soll.

`public` **Common.Logic.Pois.MissionObjectiveLocation SolvableMissionObjective**( )

Gibt ein MissionObjectiveLocation Objekt zurück, wenn dieses Missionsziel in der Nähe ist und vom Spieler gelöst werden kann. Sind zwei Missionsziele in der Nähe des Spielers, wird jenes zurückgegeben, dass in der Szenariodatei zuerst steht.

`public` **int SolveAttemptsLeft**( )

Gibt die Anzahl der übrigen Versuche zum lösen eines Missionsziels zurück.

`public` **void SolveMissionObjective**( )

Wird aufgerufen, wenn der Spieler versucht ein Missionsziel zu lösen.

Parameter

* `missionOb` - das Missionsziel
* `solution` - Die vom Spieler angegebene Lösung

`public` **void Wiretap**( )

Wird aufgerufen, um Abhören zu starten.

F.2.5 Klasse SetupData

Kapselt die Daten, die für ein Anmelden des Clients beim Server notwendig sind. Die Daten werden im SetupPanel vom Nutzer eingegeben.

Methoden

`public` **void SaveConfigFile**( )

Speichert den Inhalt des Attribute in der Cofigdatei zum erneuten laden.

F.2.6 Klasse SynchronizedQueue

Synchronisierte Warteschlange (FI-FO) für beliebige Objekte. Ist kein Objekt in der Warteschlange, blockiert die Dequeue() Methode beim Aufruf, bis wieder ein Objekt zur Verfügung steht, oder die Shutdown() Methode aufgerufen wurde.

Konstruktoren

`public` **SynchronizedQueue**( )

Erzeugt neue synchronisierte Warteschlange.

Methoden

`public` **void Clear**( )

Leert die Warteschlange.

`public` **object Dequeue**( )

Holt das vorderste Objekt aus der Warteschlange heraus. Ist gerade kein Objekt in der Warteschlange, blockiert die Methode, bis entweder ein Objekt zur Verfügung steht, oder die Methode Shutdown() aufgerufen wurde. In letzterem Fall liefert diese Methoden NULL statt eines Objekts zurück.

`public` **void Enqueue**( )

Fügt der Warteschlange ein Objekt hinzu.

Parameter

* `obj` - Anzuhängendes Objekt

`public` **void Shutdown**( )

Terminiert die Warteschlange. Die wartenden Objekte werden gelöscht. Eingehende Objekte werden ignoriert. Falls Threads in der Methode Dequeue() blockiert sind, werden sie aufgeweckt und liefern NULL statt einem Objekt zurück.

`public` **int Size**( )

Liefert die Anzahl der wartenden Events.

F.3 Namespace Client.Logic.Contextmanager

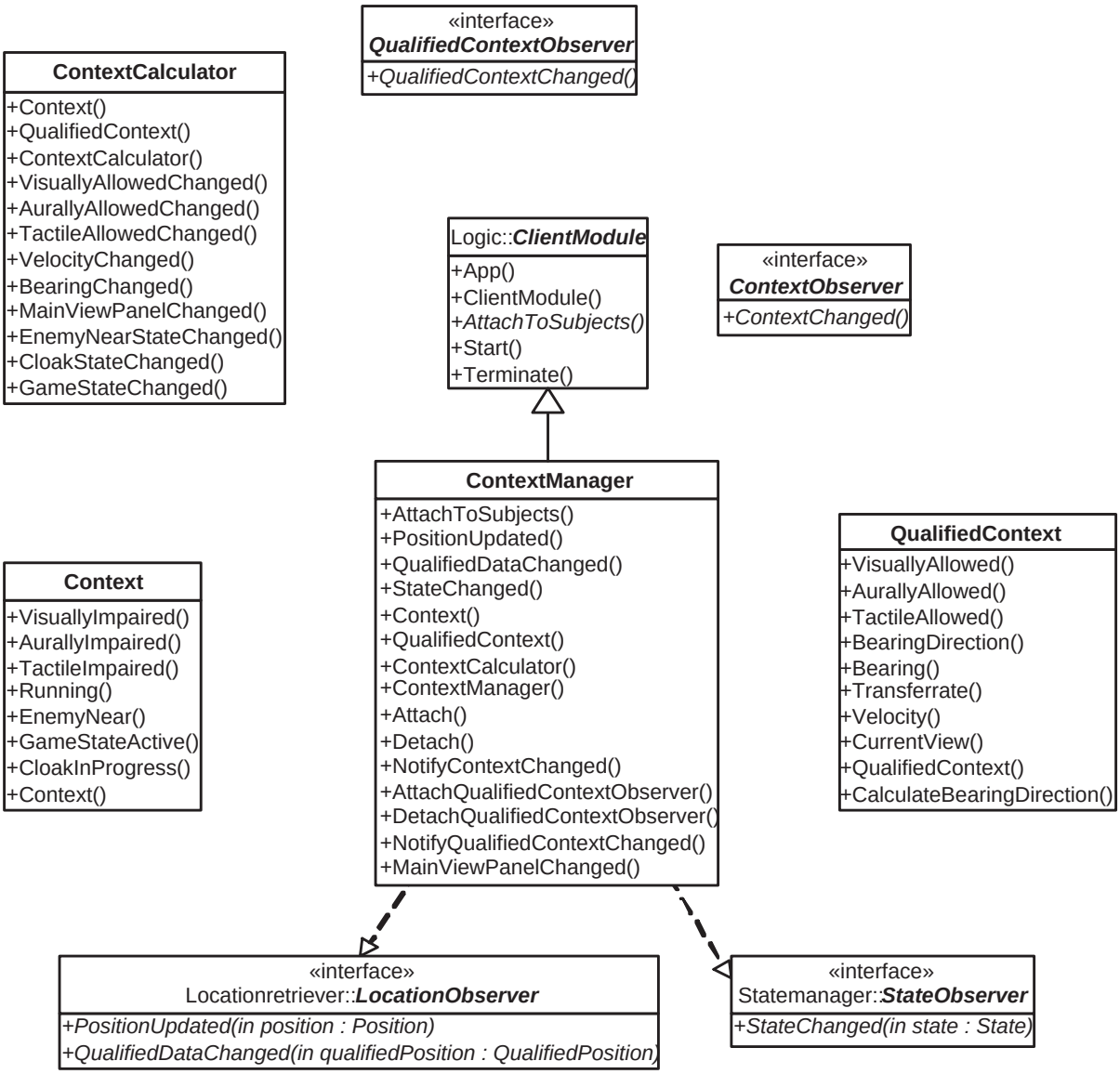


Abbildung F.3: UML Struktur des Namensraum Client.Logic.Contextmanager

F.3.1 Schnittstelle ContextObserver

Module, die über Änderungen des in der Klasse Context gespeicherten interpretierten Kontexts informiert werden wollen, implementieren diese Schnittstelle und können sich dann als Observer beim ContextManager registrieren.

Methoden

`public` **void ContextChanged**( )

Zeigt an, dass sich der Kontext geändert hat

F.3.2 Schnittstelle QualifiedContextObserver

Module, die über Änderungen des in der Klasse QualifiedContext gespeicherten qualifizierten Kontexts informiert werden wollen, implementieren diese Schnittstelle und können sich dann als Observer beim ContextManager registrieren.

Methoden

`public` **void QualifiedContextChanged**( )

Wird aufgerufen, wenn sich ein Attribut des qualifizierten Kontexts geändert hat.

F.3.3 Klasse Context

Stellt den interpretierten Kontext der Anwendung dar. Interpretiert bedeutet in diesem Fall, dass dieser Kontext aus den Daten des Qualifizierten Kontexts berechnet wird. Dabei werden qualifizierte Daten, wie z.B. die Bewegungsgeschwindigkeit, direkt in für die Anwendung relevante Kontext Informationen übersetzt. Erreicht ein Spieler eine bestimmte Bewegungsgeschwindigkeit, befindet er sich in dem interpretierten Kontext 'rennend'. Die Anwendung kann dann auf den resultierenden Verlust der Aufmerksamkeit reagieren, indem sie weniger Informationen über das Display ausgibt.

Konstruktoren

`public` **Context**( )

Erzeugt ein neues Objekt für den interpretierten Kontext.

F.3.4 Klasse ContextCalculator

Der ContextCalculator nimmt die Änderungen der beobachteten Kontext-Variablen entgegen und speichert sie in den qualifizierten Kontext. Gleichzeitig berechnet er den interpretierten Kontext und stößt bei Änderungen die Benachrichtigung der beim ContextManager registrierten ContextObserver an.

Konstruktoren

`public` **ContextCalculator**( )

Erzeugt neue Klasse zur Interpretation des Kontexts.

Parameter

- `contextMananger` - die Referenz auf den ContextManager, über den die Observer benachrichtigt werden sollen

Methoden

`public` **void AurallyAllowedChanged**( )

Wird aufgerufen, wenn der Spieler auditive Ausgaben (nicht mehr) einschränken will.

Parameter

- `allowed` - TRUE, wenn erlaubt, FALSE, wenn eingeschränkt

`public` **void CloakStateChanged**( )

Wird aufgerufen, wenn sich der Automat, der den Status der Fähigkeit Tarnen repräsentiert, geändert hat. Anhand des aktuellen Status wird dann der Kontext gesetzt. Dieser Kontext liefert Informationen dazu, ob die Fähigkeit verfügbar (oder nicht) oder gerade aktiviert ist (oder nicht).

Parameter

* `state` - der zugehörige Tarnen-Automat

`public` **void EnemyNearStateChanged**( )

Wird aufgerufen, wenn sich der Automat, der angibt ob sich ein Gegner des Spielers in der Nähe des Spielers befindet, geändert hat.

Parameter

* `state` - der zugehörige EnemyNear-Automat

`public` **void GameStateChanged**( )

Wird aufgerufen, wenn sich der Automat, der den Status des aktuellen Spielzustands des Spielers repräsentiert, geändert hat. Je nachdem, ob der Spieler gerade aktiv oder passiv ist, wird der Kontext dementsprechend gesetzt.

Parameter

* `state` - der zugehörige GameState-Automat

`public` **void TactileAllowedChanged**( )

Wird aufgerufen, wenn der Spieler taktile Ausgaben (nicht mehr) einschränken will.

Parameter

* `allowed` - TRUE, wenn erlaubt, FALSE, wenn eingeschränkt

`public` **void VisuallyAllowedChanged**( )

Wird aufgerufen, wenn der Spieler visuelle Ausgaben (nicht mehr) einschränken will.

Parameter

* `allowed` - TRUE, wenn erlaubt, FALSE, wenn eingeschränkt

F.3.5 Klasse ContextManager

Der ContextManager arbeitet vor allem als Observer. Indem er die anderen Module als Observer überwacht, sammelt er so viele Informationen wie möglich, die auf den aktuellen Kontext des Nutzers schließen lassen. Andere Module können den ContextManager auf Änderungen des Kontexts observieren.

Konstruktoren

`public` **ContextManager**( )

Erzeugt neuen Modul zur Analysierung des akutellen Kontext des Spielers.

Parameter

* `app` - Referenz auf die zugehörige Applikation

Methoden

`public` **void Attach**( )

Fügt dem ContextManager einen Observer auf den interpretierten Kontext hinzu.

Parameter

* `contextObserver` - hinzuzufügender Observer

`public` **void AttachQualifiedContextObserver**( )

Fügt dem ContextManager einen Observer auf den qualifizierten Kontext hinzu.

Parameter

* `contextObserver` - der hinzuzufügende Observer

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void Detach**( )

Entfernt einen Observer auf den interpretierten Kontext.

Parameter

* `contextObserver` - zu entfernender Observer

`public` **void DetachQualifiedContextObserver**( )

Entfernt einen Observer auf den qualifizierten Kontext.

Parameter

* `contextObserver` - der zu entfernender Observer

`public` **void MainViewPanelChanged**( )

Diese Methode wird aufgerufen, wenn sich die Ansicht des MainViewPanels geändert hat.

Parameter

* `mainViewPanel` - Das MainViewPanel, das sich geändert hat.

`public` **void NotifyContextChanged**( )

Benachrichtigt die Observer, dass sich der interpretierte Kontext geändert hat.

`public` **void NotifyQualifiedContextChanged**( )

Benachrichtigt die Observer, dass sich der qualifizierte Kontext geändert hat.

`public` **void PositionUpdated**( )

Wird aufgerufen, wenn der Client seine Position geändert hat. Die Positionsdaten umfasst Längengrad, Breitegrad und Höhe der aktuellen Spielerposition. Die Änderung der qualifizierten Daten, wie z.B. der Positionqualität wird über die Methode QualifiedDataChanged(QualifiedPosition) bekanntgegeben.

Parameter

* `position` - das Positionsobjekt, das die aktuelle Spielerposition enthält

`public` **void QualifiedDataChanged**( )

Wird aufgerufen, wenn sich die qualifizierten Positionsdaten geändert haben. Die qualifizierten Daten umfassen die Qualität der Positionsdaten sowie Bewegungsrichtung und Bewegungsgeschwindigkeit des Spielers. Diese Methode wird jedoch nicht aufgerufen, wenn sich die geographische Position des Spielers geändert hat. In diesem Fall wird die Änderung über die Methode PositionUpdated(Position) bekanntgegeben.

Parameter

* `qualifiedPosition` - die qualifizierten Positionsdaten

`public` **void StateChanged**( )

Wird aufgerufen, wenn sich ein Zustand geändert hat. Der Zustand kann über seine StateID einem bestimmten Automaten zugeordnet werden.

Parameter

* `state` - der Zustand, der sich geändert hat

F.3.6 Klasse QualifiedContext

Diese Klasse repräsentiert den qualifizierten Kontext des Nutzers. Sie speichert alle Details, die im weitesten Sinne zum Kontext gehören, wie z.B. die aktuelle Bewegungsgeschwindigkeit des Spielers. Aus den Daten dieser Klasse lässt sich der für die Spielaktionen relevante, interpretierte Kontext berechnen. Die Interpretation von Geschwindigkeit konnte z.B. 'rennend' bzw. 'nicht rennend' lauten.

Konstruktoren

`public` **QualifiedContext**( )

Erzeugt neuen qualifizierten Kontext.

Methoden

`public` **string CalculateBearingDirection**( )

Berechnet aus der Bewegungsrichtung eines Spielers die Bewegeungsrichtung als String-Repräsentation.

F.4 Namespace Client.Logic.Knownpois

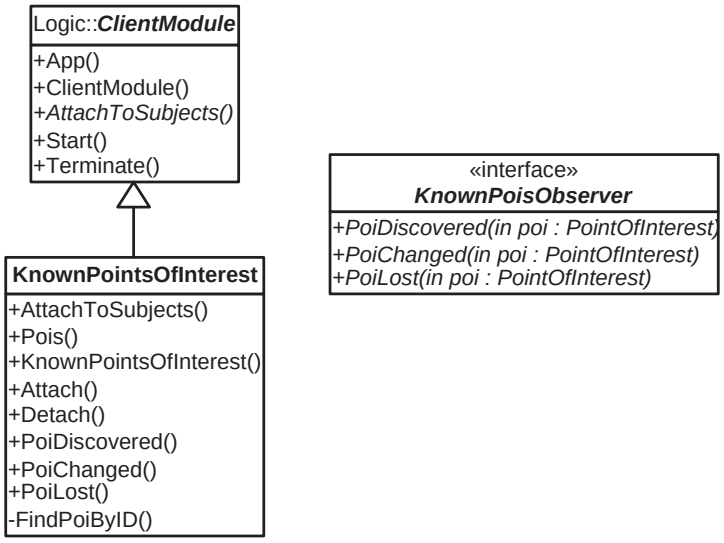


Abbildung F.4: UML Struktur des Namensraum Client.Logic.Knownpois

F.4.1 Schnittstelle KnownPoisObserver

Ein Modul, das diese Schnittstelle implementiert, kann sich als Observer beim KnownPointsOfInterest Modul registrieren. Es wird benachrichtigt, wenn ein Point-of-Interest sichtbar wird, sich seine Position wechselt, oder er aus dem Sichtbereich verschwindet.

Methoden

`public` **void PoiChanged**( )

Wird aufgerufen, wenn ein Point of Interest seinen Zustand ändert. Dabei kann es z.B. sich um eine Positionsänderung handeln.

Parameter

- `poi` - Point of Interest, der seinen Zustand (z.B. seine Position) geändert hat

`public` **void PoiDiscovered**( )

Wird aufgerufen, wenn dem Client ein neuer Point of Interest vom Server bekannt gegeben wird.

Parameter

- `poi` - Neu entdeckter Point of Interest

`public` **void PoiLost**( )

Wird aufgerufen, wenn ein Point of Interest aus dem Sichtbereich des Clients verschwindet. Der Server überträgt nun keine Zustandsänderungen des Point of Interest mehr an den Client, bis PoiDiscovered(PointOfInterest) für den Point of Interest wieder aufgerufen wird. Dem Client steht es frei, die dem Nutzer die zuletzt bekannte Position des Point of Interest weiterhin anzuzeigen, es wird aber empfohlen, dem Nutzer kenntlich zu machen, dass die angezeigte Position wahrscheinlich nicht mehr aktuell ist.

Parameter

* `poi` - Point of Interest, der ausser Sichtweite geraten ist

F.4.2 Klasse KnownPointsOfInterest

Dieses Modul speichert die zur Zeit für den Client sichtbaren Points-of-Interests und verfolgt deren Positionsänderungen.

Konstruktoren

`public` **KnownPointsOfInterest**( )

Erzeugt neues Modul zur Speicherung der sichtbaren Points-of-Interests.

Parameter

* `app` - Referenz auf die Anwendung

Methoden

`public` **void Attach**( )

Fügt den KnownPointsOfInterest einen KnownPoisObserver hinzu.

Parameter

* `observer` - Hinzuzufügender Observer

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert. Das KnownPointsOfInterest Modul registriert sich nicht bei anderen Modulen als Observer.

`public` **void Detach**( )

Fügt den KnownPointsOfInterest einen KnownPoisObserver hinzu.

Parameter

* `observer` - Zu entfernender Observer

`public` **void PoiChanged**( )

Wird aufgerufen, wenn ein Point-of-Interest seinen Zustand ändert. Dabei kann es z.B. sich um eine Positionsänderung handeln.

Parameter

* `poi` - Point of Interest, der seinen Zustand (z.B. seine Position) geändert hat

`public` **void PoiDiscovered**( )

Wird aufgerufen, um dem Client einen neuen Point of Interest bekannt zu machen.

Parameter

* `poi` - Neu entdeckter Point of Interest

`public` **void PoiLost**( )

Wird aufgerufen, wenn ein Point of Interest aus dem Sichtbereich des Clients verschwindet. Der Server überträgt nun keine Zustandsänderungen des Point of Interest mehr an den Client, bis PoiDiscovered(PointOfInterest) für den Point of Interest wieder aufgerufen wird. Dem Client steht es frei, die dem Nutzer die zuletzt bekannte Position des Point of Interest weiterhin anzuzeigen, es wird aber empfohlen, dem Nutzer kenntlich zu machen, dass die angezeigte Position nicht aktuell sein muss.

Parameter

* `poi` - Point of Interest, der ausser Sichtweite geraten ist

F.5 Namespace Client.Logic.Locationretriever

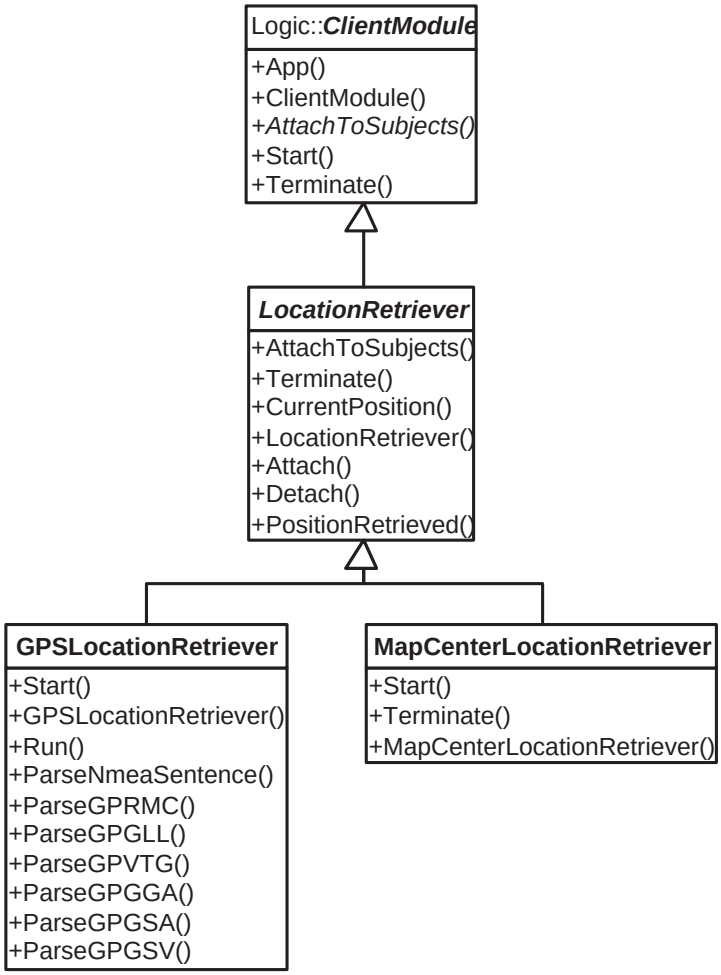


Abbildung F.5: UML Struktur des Namensraum Client.Logic.Locationretriever

F.5.1 Schnittstelle LocationObserver

Ein Modul, das diese Schnittstelle implementiert, kann sich als Observer beim LocationRetriever Modul registrieren. Es wird benachrichtigt, wenn sich die Positionsdaten oder qualitative Informationen eines Client verändern.

Methoden

`public` **void PositionUpdated**( )

Wird aufgerufen, wenn der Client seine Position geändert hat. Die Positionsdaten umfasst Längengrad, Breitegrad und Höhe der aktuellen Spielerposition. Die Änderung der qualifizierten Daten, wie z.B. der Positionqualität wird über die Methode QualifiedDataChanged(QualifiedPosition) bekanntgegeben.

Parameter

- `position` - Positionsobjekt, das die aktuelle Spielerposition enthält

`public` **void QualifiedDataChanged**( )

Wird aufgerufen, wenn sich die qualifizierten Positionsdaten geändert haben. Die qualifizierten Daten umfassen die Qualität der Positionsdaten sowie Bewegungsrichtung und Bewegungsgeschwindigkeit des Spielers. Diese Methode wird jedoch nicht aufgerufen, wenn sich die geographische Position des Spielers geändert hat. In diesem Fall wird die Änderung über die Methode PositionUpdated(Position) bekanntgegeben.

Parameter

* `qualifiedPosition` - Position, deren qualifizierte Daten sich geändert haben

F.5.2 Klasse GPSLocationRetriever

Dieses Modul verwaltet die Ansteuerung eines externen GPS-Empfangsgerät über einen virtuellen Comport (beispielsweise für eine Bluetooth-Verbindung). Zudem werden durch diese Klasse die empfangenen GPS-Signale ausgewertet, indem die von dem GPS-Empfänger gesendeten Bytes in einen String konvertiert werden, aus dem die relevanten Informationen (wie z.B. die geographische Länge, Breite und Höhe, die Bewegungsgeschwindigkeit und -richtung sowie Informationen übder die Qualität der empfangenen GPS-Signale) ausgelesen werden. In definierten Zeitabständen wird das LocationRetriever-Modul über den Empfang neuer Positionsdaten informiert.

Konstruktoren

`public` **GPSLocationRetriever**( )

Erzeugt ein neues Positionsbestimmungs-Modul.

Parameter

* `app` - die Anwendung, zu der dieses Modul gehört
* `port` - der (zu öffnende) COM-Port, auf dem die Daten zwischen GPS-Empfänger und Applikation übertragen werden

Methoden

`public` **void ParseGPGGA**( )

Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz (in diesem Fall $GPGGA) in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ',') und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind fest. Der $GPGGA-String enthält u.a. die Angaben (geographische) Länge, Breite und Höhe, sowie Informationen über die Qualität der empfangenen GPS-Signale, die Anzahl der Satelliten, von denen Signale empfangen wurden und die horizontale Genauigkeit.

Parameter

* `nmeaSentence` - zu parsender $GPGGA-String, der vom GPS-Empfänger erhalten wurde

`public` **void ParseGPGLL**( )

Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz (in diesem Fall $GPGLL) in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ',') und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind fest. Der $GPGLL-String enthält u.a. die Angaben (geographische) Länge und Breite.

Parameter

* `nmeaSentence` - zu parsender $GPGLL-String, der vom GPS-Empfänger erhalten wurde

`public` **void ParseGPGSA**( )

Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz (in diesem Fall $GPGSA) in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ',') und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind fest. Der $GPGSA-String enthält u.a. die Angaben über die Genauigkeit und die horizontale sowie die vertikale Genauigkeit der GPS-Informationen.

Parameter

* `nmeaSentence` - zu parsender $GPGSA-String, der vom GPS-Empfänger erhalten wurde

`public` **void ParseGPGSV**( )

Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz (in diesem Fall $GPGSV) in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ',') und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind fest. Der $GPGSV-String enthält u.a. die Angaben über die Anzahl der Satelliten, die momentan in Sichtweite des GPS-Empfängers liegen. (Es werden nicht zwangsläufig auch von allen Satelliten Signale empfangen und zur Positionsbestimmung herangezogen).

Parameter

* `nmeaSentence` - zu parsender $GGPGSV-String, der vom GPS-Empfänger erhalten wurde

`public` **void ParseGPRMC**( )

Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz (in diesem Fall $GPRMC) in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ',') und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind fest. Der $GPRMC-String enthält u.a. die Angaben (geographische) Länge, Breite und die momentane Bewegungsrichtung.

Parameter

* `nmeaSentence` - zu parsender $GPRMC-String, der vom GPS-Empfänger erhalten wurde

`public` **void ParseGPVTG**( )

Die Informationen der GPS-Strings werden in Abhängigkeit der Startsequenz (in diesem Fall $GPVTG) in Teilstrings unterteilt (Trennzeichen innerhalb der Strings ist ',') und in ein Positionsobjekt geschrieben. Die Positionen der Informationen innerhalb eines GPS-Strings sind fest. Der $GPVTG-String enthält u.a. die Angaben über die aktuelle Geschwindigkeit.

Parameter

* `nmeaSentence` - zu parsender $GPVTG-String, der vom GPS-Empfänger erhalten wurde

`public` **void ParseNmeaSentence**( )

Aktualisiert die zuletzt gespeicherte Position anhand des übergebenen NMEA-Satzes, der die neuen Positionsdaten enthält.

Parameter

* `nmeaSentence` - vollständiger Satz nach dem NMEA-Protokoll

`public` **void Run**( )

Diese Methode wird in einem eigenen Thread gestartet. Es wird eine Übertagung von Daten zwischen einem GPS-Empfänger und der Applikation initiiert und die dafür notwendigen Einstellungen vorgenommen. In festgelegten Zeitabständen werden Positionsdaten und weitere qualitative Informationen, die aus den GPS-Signalen entnommen werden, an die Observer weitergeleitet.

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll.

F.5.3 Klasse LocationRetriever

Diese Klasse ist die zentrale Verwaltungsstelle des LocationRetriever-Moduls. Sie speichert sowohl die Positionsdaten eines Spielers als auch Qualitätsinformationen der GPS-Daten, die in die Berechnung des Kontexts eines Spielers einfließen. Module können sich bei dieser Klasse sowohl als Observer auf Positionsänderungen als auch als Observer auf Veränderungen der Qualitätsinformationen (z.B. Bewegungsrichtung und -geschwindigkeit sowie die Qualität der empfangenen GPS-Signale) registrieren. Werden GPS-Signale empfangen werden die Änderungen der Informationen ausgewertet und gegebenenfalls die Observer benachrichtigt. Unabhängig vom Observerprinzip kann über das Property CurrentPosition auf die aktuelle Position sowie auf qualifizierte Positionsinformationen zugegriffen werden.

Konstruktoren

`public` **LocationRetriever**( )

Erzeugt neues Positionsbestimmungs-Modul.

Parameter

- `app` - Anwendung, zu der dieses Modul gehört

Methoden

`public` **void Attach**( )

Fügt dem LocationRetriever einen Observer hinzu, der auf Positionsänderungen wartet.

Parameter

- `locationObserver` - hinzuzufügender Observer

`public` **void AttachToSubjects**( )

Von ClientModule überschriebene Methode. Wird in diesem Fall nicht verwendet.

`public` **void Detach**( )

Entfernt einen Observer.

Parameter

- `locationObserver` - zu entfernender Observer

`public` **void PositionRetrieved**( )

Wird aufgerufen, wenn der implementierde LocationRetriever eine neue Position ermittelt hat. Unterscheidet sich die neue Position weit genug von der alten Position, werden die Observer benachrichtigt.

`public` **void Terminate**( )

Diese Methode wird von der Applikation aufgerufen, wenn sie sich beendet. Jedes Modul kann sie überschreiben, um vor dem Beenden z.B. belegte Ressourcen freizugeben.

F.5.4 Klasse MapCenterLocationRetriever

Der MapCenterLocationRetriever ist eine Spezialisierung des Moduls zur Positionsbestimmung, dass die Zentrierung der Karte als Position übernimmt. Es wurde zu Testzwecken implementiert.

Konstruktoren

`public` **MapCenterLocationRetriever**( )

Erzeugt neues Modul zur Positionsbestimmung.

Parameter

* `app` - Referenz auf Client

Methoden

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll.

`public` **void Terminate**( )

Beendet den Thread zur Positionsbestimmung durch das Zentrieren der Karte.

F.6 Namespace Client.Logic.Messagemanager

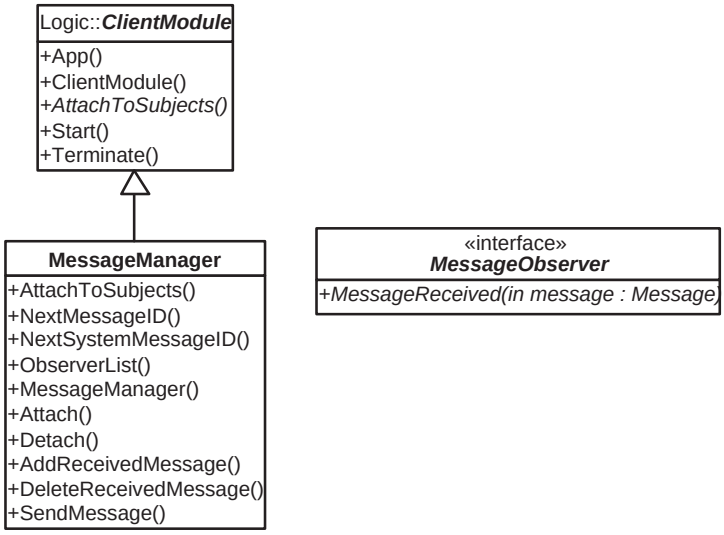


Abbildung F.6: UML Struktur des Namensraum Client.Logic.Messagemanager

F.6.1 Schnittstelle MessageObserver

Ein Modul, das diese Schnittstelle implementiert, kann sich als Observer beim MessageManager registrieren. Es wird benachrichtigt, wenn eine neue Nachricht empfangen wurde.

Methoden

`public` **void MessageReceived**( )

Wird aufgerufen, wenn eine neue Nachricht empfangen wurde.

Parameter

* `message` - empfangene Nachricht

F.6.2 Klasse MessageManager

Die Klasse MessageManager auf Seite des Clients dient dazu, die empfangenen Nachrichten zu verwalten. Nachrichten können zur Liste der empfangenen Nachrichten hinzugefügt oder aus dieser gelöscht werden. Zu versendende Nachrichten werden in MessageEvents verpackt an den NetworkManager weitergeleitet.

Konstruktoren

`public` **MessageManager**( )

Erzeugt neues MessageManager-Objekt.

Parameter

* `app` - Referenz auf die zugehörige Anwendung

Methoden

`public` **void AddReceivedMessage**( )

Fügt Nachricht zur Liste empfangener Nachrichten hinzu und teilt den Empfang der Nachricht allen angemeldeten MessageObservern mit.

Parameter

* `message` - empfangene Nachricht

`public` **void Attach**( )

Fügt dem MessageManager einen MessageObserver hinzu.

Parameter

* `messageObserver` - hinzuzufügender Observer

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void DeleteReceivedMessage**( )

Löscht Nachricht aus Liste der empfangenen Nachrichten.

Parameter

* `message` - zu löschende Nachricht

`public` **void Detach**( )

Entfernt einen MessageObserver des MessageManagers.

Parameter

* `messageObserver` - zu entfernender Observer

`public` **void SendMessage**( )

Verpackt zu versendende Nachricht in ein MessageEvent und leitet dieses an den NetworkMananger weiter.

Parameter

* `message` - zu versendende Nachricht

F.7 Namespace Client.Logic.Statemanager

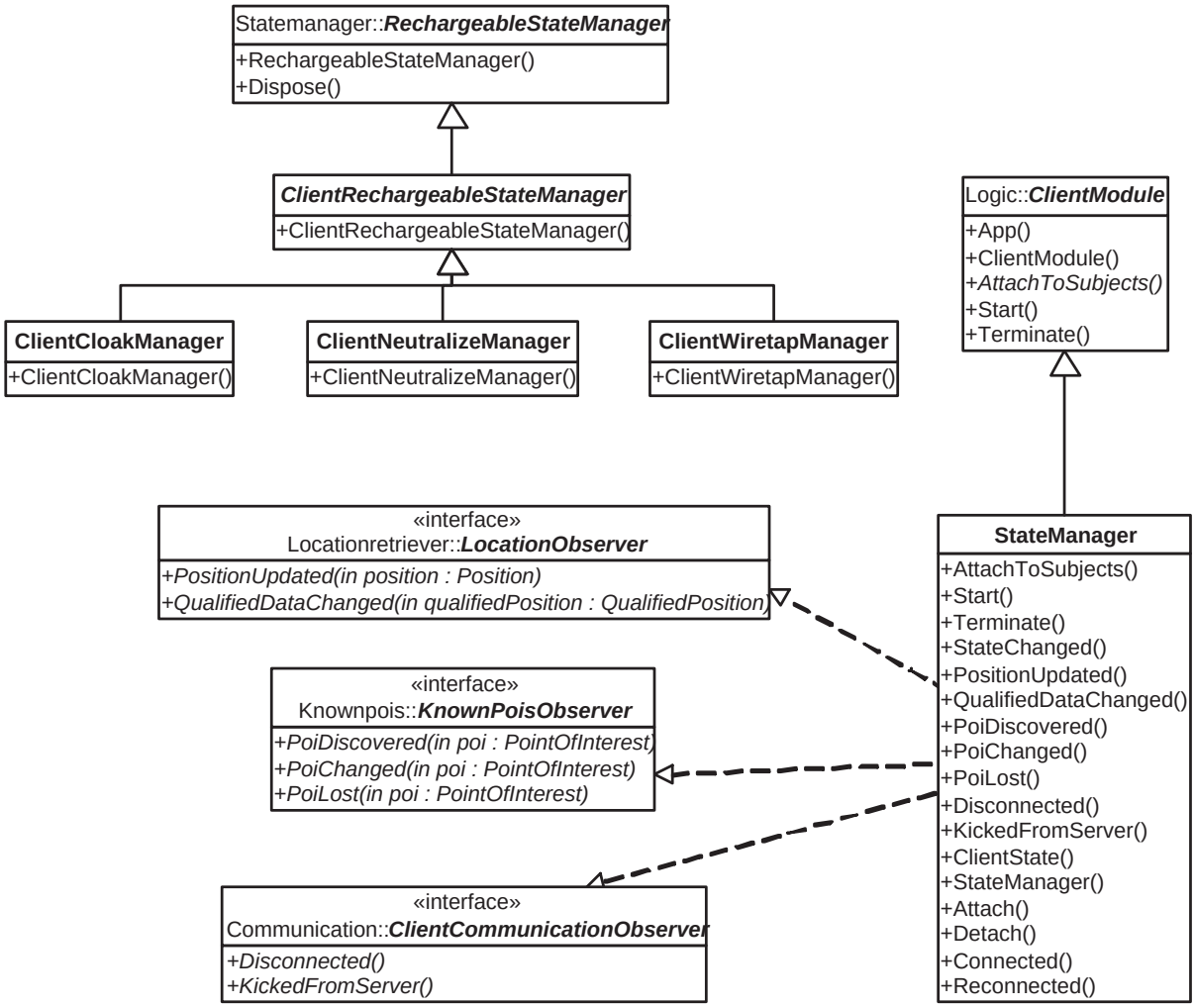


Abbildung F.7: UML Struktur des Namensraum Client.Logic.Statemanager

F.7.1 Klasse ClientCloakManager

Der ClientCloakManager verwaltet das Aufladen, Aktivieren und Deaktivieren der Fähigkeit Cloak, außerdem den gegenseitigen Ausschluß mit der Fähigkeit Wiretap. Zustandsveränderungen bekommt der ClientCloakManager als StateObserver automatisch geliefert.

Konstruktoren

`public` **ClientCloakManager**( )

Erzeugt ein neues ClientCloakManager-Objekt.

Parameter

* `app` - Referenz auf die Applikation des Clients
* `state` - der zu verwaltende State (Automat)

Methoden

`public` **void StateChanged**( )

Implementierung der RechargeableStateManager-Schnittstelle. Reagiert auf Veränderungen des Wiretap- und des CloakAutomatons. Die Fortschrittsbalken und der Cloak-Knopf werden aufgrund von Zustandsveränderungen angepasst und der GUI (dem MapPanel) wird mitgeteilt, ob ein getranter Spieler gezeichnet werden soll.

Parameter

* `state` - der veränderte Zustand.

F.7.2 Klasse ClientNeutralizeManager

Der ClientNeutralizeManager verwaltet das Aufladen, Aktivieren und Deaktivieren der Fähigkeit Neutralize. Zustandsveränderungen bekommt der ClientCloakManager als StateObserver automatisch geliefert.

Konstruktoren

`public` **ClientNeutralizeManager**( )

Erzeugt ein neues ClientNeutralizeManager-Objekt.

Parameter

* `app` - Referenz auf die Applikation des Clients
* `state` - der zu verwaltende State (Automat)

Methoden

`public` **void StateChanged**( )

Implementierung der RechargeableStateManager-Schnittstelle. Reagiert auf Veränderungen des NeutralizeAutomatons. Die Fortschrittsbalken und der NEutralize-Knopf werden aufgrund von Zustandsveränderungen angepasst und der GUI (dem MapPanel) wird mitgeteilt, ob ein Neutralisierungskreis gezeichnet werden soll.

Parameter

* `state` - der geänderte Zustand

F.7.3 Klasse ClientRechargeableStateManager

Basislasse für Behandlung einer Veränderung eines RechargeableState auf Clientseite. Der ClientRechargeableStateManager kümmert sich um die Behandlung von Timer-Events.

Konstruktoren

`public` **ClientRechargeableStateManager**( )

Erzeugt ein neues ClientRechargeableStateManager-Objekt.

Parameter

* `app` - Referenz auf die Applikation des Clients
* `state` - der zu verwaltende State (Automat)
* `rechargeInterval` - das Interval, indem der Timer-Rückruf stattfindet
* `downCoolingTime` - Auflagezeit der Fähigkeit

F.7.4 Klasse ClientWiretapManager

Der ClientWiretapManager verwaltet das Aufladen, Aktivieren und Deaktivieren der Fähigkeit Wiretap, außerdem den gegenseitigen Ausschluß mit der Fähigkeit Cloak. Zustandsveränderungen bekommt der ClientWiretapManager als StateObserver automatisch geliefert.

Konstruktoren

`public` **ClientWiretapManager**( )

Erzeugt ein neues ClientWireManager-Objekt mit einer ClientApplication-Instanz und einem Zustand.

Parameter

- `app` - Referenz auf die Applikation des Clients
- `state` - der zu verwaltende State (Automat)

Methoden

`public` **void StateChanged**( )

Implementierung der RechargeableStateManager-Schnittstelle. Reagiert auf Veränderungen des Wiretap- und des CloakAutomatons. Die Fortschrittsbalken und der Wiretap-Knopf werden aufgrund von Zustandsveränderungen angepasst und der GUI (dem MapPanel) wird mitgeteilt, ob ein abhörender Spieler gezeichnet werden soll.

Parameter

- `state` - der veränderte Zustand

F.7.5 Klasse StateManager

Der Statemanager verwaltet Zuständsveränderungen. Er observiert Module, die Zustandsveränderungen hervorrufen, reagiert auf Veränderungen und gibt diese weiter, indem er seine registrierten Observer benachrichtigt.

Konstruktoren

`public` **StateManager**( )

Erzeugt ein neues StateManager-Objekt mit einer Application-Instanz.

Parameter

- `app` - das Application-Objekt

Methoden

`public` **void Attach**( )

Fügt dem StateManager einen Observer hinzu.

Parameter

- `stateObserver` - der hinzuzufügende Observer

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void Connected**( )

Wird aufgerufen, wenn die Kommunikationsschicht initial eine Verbindung zum Server hergestellt und eine ClientID erhalten hat. Ab dem Zeitpunkt, zu dem diese Methode aufgerufen wird, können Events versendet werden.

Parameter

* `receivedClientID` - ID, die der Server dem Client zugewiesen hat.

`public` **void Detach**( )

Entfernt einen Obserer auf den StateManager

Parameter

* `stateObserver` - der zu entfernende Observer

`public` **void Disconnected**( )

Wird aufgerufen, wenn die Verbindung zum Server unterbrochen wurde. Events, in der Warteschlange zur Versendung an den Server werden erst gesendet, wenn die Verbindung wieder hergestellt ist.

`public` **void KickedFromServer**( )

Wird genau dann aufgerufen, wenn der Server die Verbindung beendet hat.

`public` **void PoiChanged**( )

Wird aufgerufen, wenn ein Point of Interest seinen Zustand ändert. Dabei kann es z.B. sich um eine Positionsänderung handeln.

Parameter

* `poi` - Point of Interest, der seinen Zustand (z.B. seine Position) geändert hat

`public` **void PoiDiscovered**( )

Wird aufgerufen, wenn dem Client ein neuer Point of Interest vom Server bekannt gegeben wird.

Parameter

* `poi` - neu entdeckter Point of Interest

`public` **void PoiLost**( )

Wird aufgerufen, wenn ein Point of Interest aus dem Sichtbereich des Clients verschwindet. Der Server überträgt nun keine Zustandsänderungen des Point of Interest mehr an den Client, bis PoiDiscovered(PointOfInterest) für den Point of Interest wieder aufgerufen wird. Dem Client steht es frei, die dem Nutzer die zuletzt bekannte Position des Point of Interest weiterhin anzuzeigen, es wird aber empfohlen, dem Nutzer kenntlich zu machen, dass die angezeigte Position nicht aktuell sein muss.

Parameter

* `poi` - Point of Interest, der ausser Sichtweite geraten ist

`public` **void PositionUpdated**( )

Wird aufgerufen, wenn der Client seine Position geändert hat. Die Positionsdaten umfasst Längengrad, Breitegrad und Höhe der aktuellen Spielerposition. Die Änderung der qualifizierten Daten, wie z.B. der Positionqualität wird über die Methode QualifiedDataChanged(QualifiedPosition) bekanntgegeben.

Parameter

* `position` - Positionsobjekt, das die aktuelle Spielerposition enthält

`public` **void QualifiedDataChanged**( )

Wird aufgerufen, wenn sich die qualifizierten Positionsdaten geändert haben. Die qualifizierten Daten umfassen die Qualität der Positionsdaten sowie Bewegungsrichtung und Bewegungsgeschwindigkeit des Spielers. Diese Methode wird jedoch nicht aufgerufen, wenn sich die geographische Position des Spielers geändert hat. In diesem Fall wird die Änderung über die Methode PositionUpdated(Position) bekanntgegeben.

Parameter

* `qualifiedPosition` - Position, deren qualifizierte Daten sich geändert haben

`public` **void Reconnected**( )

Wird aufgerufen, wenn eine unterbrochene Verbindung wieder hergestellt ist.

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll.

`public` **void StateChanged**( )

Implementierung der StateObserver-Schnittstelle. Benachrichtigt alle registrierten Observer über eine Veränderung eines Zustands.

Parameter

* `state` - der sich veränderte State

`public` **void Terminate**( )

Diese Methode wird von der Applikation aufgerufen, wenn sie sich beendet. Jedes Modul kann sie überschreiben, um vor dem Beenden z.B. belegte Ressourcen freizugeben.

F.8 Namespace Client.Logic.Userprofile

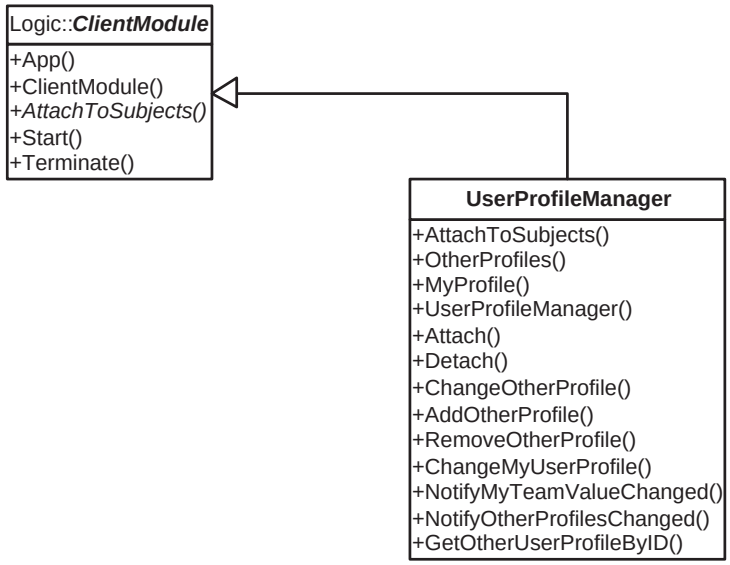


Abbildung F.8: UML Struktur des Namensraum Client.Logic.Userprofile

F.8.1 Klasse UserProfileManager

Der UserProfileManager verwaltet die Profile aller Spieler.

Konstruktoren

`public` **UserProfileManager**( )

Erzeugt einen neuen UserProfileManager.

Parameter

* `app` - die aktuelle ClientApplication

Methoden

`public` **void AddOtherProfile**( )

Ein neuer Spieler hat sich angemeldet, das Profil wird aufgenommen.

Parameter

* `profile` - das neue Profil

`public` **void Attach**( )

Fügt einen Observer auf Profile-Events hinzu.

Parameter

* `userProfileObserver` - der neue Observer

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void ChangeMyUserProfile**( )

Das eigene Profil hat sich geändert.

Parameter

* `profile` - das veränderte Profil

`public` **void ChangeOtherProfile**( )

Das Profil eines anderen Spielers hat sich geändert.

Parameter

* `profile` - das veränderte Profil

`public` **void Detach**( )

Entfernt einen UserProfileObserver.

Parameter

* `userProfileObserver` - der UserProfileObserver, der entfernt werden soll

`public` **Common.Logic.Userprofile.UserProfile GetOtherUserProfileByID**( )

Liefert das UserProfile der angegebenen ID oder null, wenn kein solches UserProfile existiert.

Parameter

* `ID` - die Id des gesuchten UserProfiles

`public` **void NotifyMyTeamValueChanged**( )

Benachrichtigt die Observer, dass sich die Teamzugehörigkeit eines Spielers geändert hat.

Parameter

* `profile` -

`public` **void NotifyOtherProfilesChanged**( )

Benachrichtigt die Observer, dass sich die Liste der Profile anderer Spieler geändert hat.

`public` **void RemoveOtherProfile**( )

Ein Spieler hat sich abgemeldet, das Profil wird entfernt.

Parameter

* `profile` - das Profil

F.9 Namespace Client.Userinterface

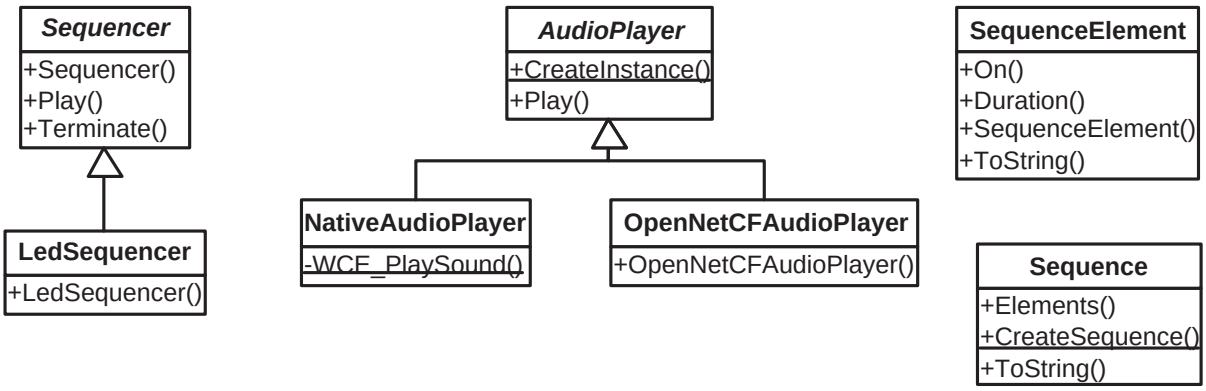


Abbildung F.9: UML Struktur des Namensraum Client.Userinterface

F.9.1 Klasse AudioPlayer

Abstrakte Oberklasse für ein Audio Ausgabegerät

Methoden

`public` **Client.Userinterface.AudioPlayer CreateInstance**( )

Instanziiert eine Implementierung des AudioPlayers. Die Methode wirft eine AudioNotAvailableException, wenn keine Instanz des AudioPlayers erzeugt werden kann.

`public` **void Play**( )

Spielt die Audio-Datei mit dem angegebenen Namen ab, sofern sie im Unterverzeichnis 'wav' des Clients liegt. Wird die Datei nicht gefunden, gibt die Methode ein Standard-Sample wieder. Wird auch dieses nicht gefunden, generiert die Methoden eine Ausnahme. Das Abspielen geschieht asynchron, d.h. die Methode kehrt sofort zurück und es können mehrere Samples gleichzeitig abgespielt werden.

Parameter

- `soundFile` - Name der Audio-Datei, ohne Pfadangabe

F.9.2 Klasse LedSequencer

Implementierung des Sequenzers, der die Sequenzen mittels virtueller oder echter LED ausgibt. Neben den physikalich vorhandenen LEDs ist auch u.a. beim Pocket PC iPAQ 5450 über LED Nr. 5 der Vibrationsalarm ansteuertbar.

Konstruktoren

`public` **LedSequencer**( )

Erzeugt neue Implementierung des Sequenzers, der die Sequenzen mittels virtueller oder echter LED ausgibt.

Parameter

- `ledNr` - Nummer der anzusteuernden LED

F.9.3 Klasse Sequence

Stellt eine Sequenz von An-Aus Befehlen und deren Dauer dar.

Konstruktoren

`public` **Sequence**( )

Methoden

`public` **Client.Userinterface.Sequence CreateSequence**( )

Parset eine Sequenz aus einem String, der eine Zeichenfolge der Symbole '.', '-' und dem Leerzeichen enthält. Enthält der String andere Symbole, wird eine ArgumentException geworfen.

Parameter

- `seq` - zu parsender String

F.9.4 Klasse SequenceElement

Element eine An-Aus Sequenz. Ein Element speichert seinen Befehl (An oder Aus) und die Dauer, für die der Befehl gültig sein soll.

Konstruktoren

`public` **SequenceElement**( )

Erzeugt neues Sequenzelement.

Parameter

- `on` - Befehl dieses Sequenzelements: true = An, false = Aus
- `duration` - Dauer, für die der Befehl mindestens gültig sein soll

F.9.5 Klasse Sequencer

Abstrakte Oberklasse für alle Klassen, die An-Aus Sequenzen darstellen. Der Sequenzer startet einen eigenen Thread, so dass die Sequenzen abgespielt werden können, ohne den aufrufenden Thread zu blockieren. Dies bedeutet aber auch, dass jeder eingesetzt Sequenzer mit der Methoden Terminate beendet werden muss, damit die Anwendung korrekt herunterfährt. Wird die Play() Methoden aufgerufen, während noch eine andere Sequenz abgespielt wird, wird die neue Sequenz in eine Warteschlange gestellt und erst nach Beendigung der vorigen Sequenz abgespielt.

Konstruktoren

`public` **Sequencer**( )

Erzeugt neuen Sequenzer und startet den internen Thread, mit dem die Sequenzen abgespielt werden.

Methoden

`public` **void Play**( )

Legt eine Sequenz von An-Aus Befehlen in der Warteschlange ab, aus der der interne Thread die abzuspielenden Sequenzen bezieht.

Parameter

* `sequence` - abzuspielende Sequenz

F.10 Namespace Client.Userinterface.Gui

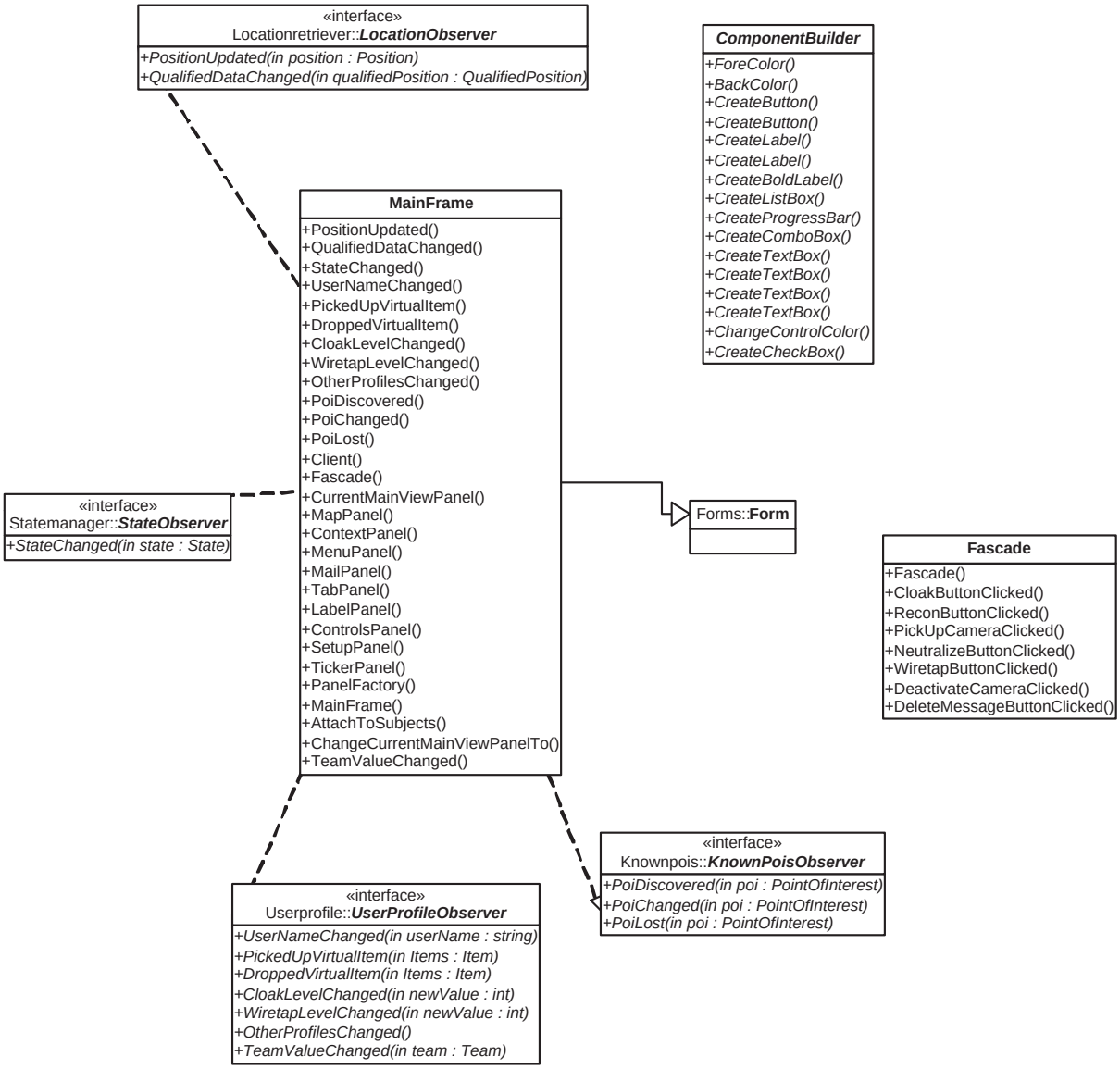


Abbildung F.10: UML Struktur des Namensraum Client.Userinterface.Gui

F.10.1 Klasse AbstractPanel

Diese Klasse dient als Basis für alle Panels, aus denen die GUI des Agentenspiels zusammengesetzt ist. Ziel ist es, den Aufbau und die Handhabung der Panels zu vereinheitlichen, indem gemeinsame benutzte Methoden in diese Klasse ausgelagert werden.

Konstruktoren

`public` **AbstractPanel**( )

Initialisiert das Panel mit der Standardhintergrundfarbe.

Parameter

* `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
* `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen

`public` **AbstractPanel**( )

Initialisiert das Panel mit der angegebenen Hintergrundfarbe.

Parameter

* `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
* `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen
* `backColor` - Hintergrundfarbe des Panels

F.10.2 Klasse ComponentBuilder

Diese Klasse ist die abstrakte Oberklasse der Klasse zur Erzeugung der einzelnen Komponenten der grafischen Benutzungsoberfläche (wie Button, TextBox o.ä.).

Methoden

`public` **System.Windows.Forms.Control ChangeControlColor**( )

Ändert die Farbe des übergebenen Controls.

Parameter

* `control` - Das zu ändernde Control
* `active` - Status des übergebenen Controls

`public` **System.Windows.Forms.Control CreateBoldLabel**( )

Erzeugt ein neues Label mit den übergebenen Werten, dessen Text in fetter Schrift ausgegeben wird.

Parameter

* `text` - Beschriftung des Labels
* `location` - Position der linken oberen Ecke des Labels, relativ zur Position des übergeordneten Panels
* `size` - Größe des Labels
* `fontSize` - Schriftgröße des Label-Textes

`public` **System.Windows.Forms.Control CreateButton**( )

Erzeugt einen neuen Button mit Standardschriftfarbe und angegebener Hintergrundfarbe.

Parameter

* `text` - Beschriftung des Buttons
* `location` - Koordinate der linken oberen Ecke des Buttons
* `size` - Größe des Buttons
* `BackGround` - Hintergrundfarbe des Buttons

`public` **System.Windows.Forms.Control CreateButton**( )

Erzeugt einen neuen Button mit der Standard Schrift- und Hintergrundfarbe.

Parameter

- `text` - Beschriftung des Buttons
- `location` - Koordinate der linken oberen Ecke des Buttons
- `size` - Größe des Buttons

`public` **System.Windows.Forms.CheckBox CreateCheckBox**( )

Erzeugt eine neue CheckBox mit den übergebenen Werten.

Parameter

- `location` - Position der linken oberen Ecke der CheckBox, relativ zur Position des übergeordneten Panels
- `size` - Größe der CheckBox

`public` **System.Windows.Forms.Control CreateComboBox**( )

Erzeugt eine neue ComboBox mit den übergebenen Werten..

Parameter

- `location` - Position der linken oberen Ecke der ComboBox, relativ zur Position des übergeordneten Panels
- `size` - Größe der ComboBox

`public` **System.Windows.Forms.Control CreateLabel**( )

Erzeugt ein neues Label mit den übergebenen Werten.

Parameter

- `text` - Beschriftung des Labels
- `location` - Position der linken oberen Ecke des Labels, relativ zur Position des übergeordneten Panels
- `size` - Größe des Labels
- `backColor` - Hintergrundfarbe des Labels

`public` **System.Windows.Forms.Control CreateLabel**( )

Erzeugt ein neues Label mit den übergebenen Werten.

Parameter

- `text` - Beschriftung des Labels
- `location` - Position der linken oberen Ecke des Labels, relativ zur Position des übergeordneten Panels
- `size` - Größe des Labels

`public` **System.Windows.Forms.Control CreateListBox**( )

Erzeugt eine neue ListBox mit den übergebenen Werten.

Parameter

- `location` - Position der linken oberen Ecke der ListBox.
- `size` - Größe der ListBox

`public` **OpenNETCF.Windows.Forms.ProgressBarEx CreateProgressBar**( )

Erzeugt eine neue Progress Control zur Anzeige des Füllstands der Fähigkeit mit den übergebenen Werten.

Parameter

* `progress` - Füllstand (0..100)
* `location` - Position der linken oberen Ecke des Controls, relativ zur Position des übergeordneten Panels
* `size` - Größe des Controls

`public` **System.Windows.Forms.Control CreateTextBox**( )

Erzeugt eine neue TextBox mit den übergebenen Werten.

Parameter

* `location` - Position der linken oberen Ecke der TextBox, relativ zur Position des übergeordneten Panels
* `size` - Größe der TextBox
* `multiline` - Gibt an, ob Schreiben in mehrer Zeilen erlaubt werden soll
* `wordWrap` - Gibt an, ob bei Bedarf ein Zeilenumbruch stattfinden soll
* `readOnly` - Gibt an, ob nur lesend auf die TextBox zugegriffen werden darf

`public` **System.Windows.Forms.Control CreateTextBox**( )

Erzeugt eine neue TextBox mit den übergebenen Werten.

Parameter

* `location` - Position der linken oberen Ecke der TextBox, relativ zur Position des übergeordneten Panels
* `size` - Größe der TextBox
* `multiline` - Gibt an, ob Schreiben in mehrer Zeilen erlaubt werden soll
* `wordWrap` - Gibt an, ob bei Bedarf ein Zeilenumbruch stattfinden soll

`public` **System.Windows.Forms.Control CreateTextBox**( )

Erzeugt eine neue TextBox mit den übergebenen Werten.

Parameter

* `location` - Position der linken oberen Ecke der TextBox, relativ zur Position des übergeordneten Panels
* `size` - Größe der TextBox

`public` **System.Windows.Forms.Control CreateTextBox**( )

Erzeugt eine neue TextBox mit den übergebenen Werten.

Parameter

* `text` - Standardmäßiger Inhalt der TextBox
* `location` - Position der linken oberen Ecke der TextBox, relativ zur Position des übergeordneten Panels
* `size` - Größe der TextBox
* `maxLength` - Maximale Eingabelänge der TextBox
* `wordWrap` - Gibt an, ob bei Bedarf ein Zeilenumbruch stattfinden soll

F.10.3 Klasse ComposeMessagePanel

Diese Klasse ist die abstrakte Oberklasse für das Panel, in dem der Betreff und der Text einer Nachricht eingegeben werden können.

Konstruktoren

`public` **ComposeMessagePanel**( )

Erzeugt einen neue Standardimplementierung eines ComposeMessagePanel.

Parameter

* `mainFrame` - Das übergeordnete Hauptanzeigefenster
* `componentBuilder` - Die Fabrik, aus der die Elemente des ComposeMessagePanel erstellt werden

F.10.4 Klasse ContextPanel

Diese Klasse ist die abstrakte Oberklasse des Panels, das zur Visualisierung des Kontexts dient. Die Klasse leitet sich von MainViewPanel ab und kann daher im Hauptansichtsbereich der GUI angezeigt werden. Das Panel implementiert die Observerschnittstelle des ContextManagers. Wird es als Observer beim ContextManager registriert, visualisiert es ausgewählte Elemente des aktuellen Kontexts. Dadurch soll zum einen die Fehlersuche erleichtert werden und zum anderen bietet es die Möglichkeit, die Einbindung von Kontext-sensivität in die Anwendung zu demonstrieren.

Konstruktoren

`public` **ContextPanel**( )

Erzeugt neue abstrakte Oberklasse des Panels, dass zur Visualisierung des Kontexts dient.

Parameter

* `mainFrame` - Referenz auf das Hauptfenster
* `componentBuilder` - Factory zur Erzeugung der GUI Komponenten

Methoden

`public` **void ContextChanged**( )

Wird aufgerufen, wenn sich ein Attribut des Kontext geändert hat.

`public` **void QualifiedContextChanged**( )

Wird aufgerufen, wenn sich ein Attribut des qualifizierten Kontext geändert hat.

F.10.5 Klasse ControlsPanel

Diese Klasse ist die abstrakte Oberklasse für das Panel, in dem Steuerungskomponenten für die 4 Grundfähigkeiten Tarnen, Abhören, Aufklären und Neutralisieren untergebracht sind.

Konstruktoren

`public` **ControlsPanel**( )

Initialisiert das Panel.

Parameter

* `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
* `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen
* `location` - Position der linken, oberen Ecke des Panels
* `size` - Höhe und Breite des Panels

Methoden

`public` **void SetCloakControlText**( )

Setzt die Beschriftung des Controls zum Auslösen der Fähigkeit Tarnen.

Parameter

* `text` - Die Beschriftung des Controls.

`public` **void SetCloakEnabled**( )

Blockiert/Deblockiert die Eingabekomponente zum Tarnen/Enttarnen.

Parameter

* `enabled` - true, wenn die Komponente Eingaben akzeptieren soll, false wenn nicht

`public` **void SetCloakProgress**( )

Aktualisiert den Aufladestatus der Fähigkeit Tarnen. Es werden nur Werte zwischen 0 (0%) und 1 (100%) akzeptiert. Bei anderen Werten ist die Anzeige nicht definiert.

Parameter

* `progress` - Aufladestatus

`public` **void SetNeutralizeEnabled**( )

Blockiert/Deblockiert die Eingabekomponente zum Neutralisieren.

Parameter

* `enabled` - true, wenn die Komponente Eingaben akzeptieren soll, false wenn nicht

`public` **void SetNeutralizeProgress**( )

Aktualisiert den Aufladestatus der Fähigkeit Neutralisieren. Es werden nur Werte zwischen 0 (0%) und 1 (100%) akzeptiert. Bei anderen Werten ist die Anzeige nicht definiert.

Parameter

* `progress` - Aufladestatus

`public` **void SetReconEnabled**( )

Blockiert/Deblockiert die Eingabekomponente zum Setzen einer Kamera.

Parameter

* `enabled` - true, wenn die Komponente Eingaben akzeptieren soll, false wenn nicht

`public` **void SetReconProgress**( )

Aktualisiert den Aufladestatus der Fähigkeit Aufklären. Es werden nur Werte zwischen 0 (0%) und 1 (100%) akzeptiert. Bei anderen Werten ist die Anzeige nicht definiert.

Parameter

* `progress` - Aufladestatus

`public` **void SetWiretapControlText**( )

Setzt die Beschriftung des Controls zum Auslösen der Fähigkeit Abhören.

Parameter

* `text` - Die Beschriftung des Controls.

`public` **void SetWiretapEnabled**( )

Blockiert/Deblockiert die Eingabekomponente zum Einschalten der Abhörfunktion.

Parameter

* `enabled` - true, wenn die Komponente Eingaben akzeptieren soll, false wenn nicht

`public` **void SetWiretapProgress**( )

Aktualisiert den Aufladestatus der Fähigkeit Abhören. Es werden nur Werte zwischen 0 (0%) und 1 (100%) akzeptiert. Bei anderen Werten ist die Anzeige nicht definiert.

Parameter

* `progress` - Aufladestatus

F.10.6 Klasse Fascade

Diese Klasse stellt die Fassade der Logik dar. Alle Eingaben der GUI werden über diese Klasse an die Logik weitergeleitet.

Konstruktoren

`public` **Fascade**( )

Erzeugt neue Fassade für die Logik

Parameter

* `app` - Referenz auf die Client-Applikation

Methoden

`public` **void CloakButtonClicked**( )

Wird aufgerufen, wenn der Nutzer auf das Eingabe-Control für die Fähigkeit Tarnen geklickt hat.

`public` **void DeactivateCameraClicked**( )

Wird aufgerufen, wenn der Nutzer auf der Übersichtskarte in die Nähe einer gegnerischen Kamera geklickt hat, um diese zu deaktivieren.

Parameter

* `camera` - Kamera, in deren Nähe geklickt wurde

`public` **void DeleteMessageButtonClicked**( )

Wird aufgerufen, wenn der Nutzer auf den Knopf zum Löschen einer empfangenen Nachricht gedrückt wurde.

Parameter

* `message` -

`public` **void NeutralizeButtonClicked**( )

Wird aufgerufen, wenn der Nutzer auf das Eingabe-Control für die Fähigkeit Neutralisieren geklickt hat.

`public` **void PickUpCameraClicked**( )

Wird aufgerufen, wenn der Nutzer auf der Übersichtskarte in die Nähe einer eigenen Kamera geklickt hat, um diese wieder aufzunehmen.

Parameter

* `camera` - Kamera, in deren Nähe geklickt wurde

`public` **void ReconButtonClicked**( )

Wird aufgerufen, wenn der Nutzer auf das Eingabe-Control für die Fähigkeit Aufklären geklickt hat.

`public` **void WiretapButtonClicked**( )

Wird aufgerufen, wenn der Nutzer auf das Eingabe-Control für die Fähigkeit Abhören geklickt hat.

F.10.7 Klasse LabelPanel

Die Klasse definiert die Schnittstelle für das LabelPanel. Das LabelPanel zeigt an, welche Ausgabemodalitäten zur Zeit verwendet werden und ob sich bestimmte Points-of-Interest (Gegner, gegerische Kameras) in der Nähe befinden.

Methoden

`public` **void OutputMediaChanged**( )

Wird aufgerufen, wenn sich die dem Kontext angemessenen Ausgabemedien geändert haben, weil sich z.B. der Kontext des Spielers geändert hat.

Parameter

* `contextAppropriateOutputMedia` - Liste der Ausgabemedien, in dem aktuellen Kontext zur Präsentation von Informationen angemessen sind

`public` **void StateChanged**( )

Wird aufgerufen, wenn sich ein Zustand geändert hat. Der Zustand kann über seine StateID einem bestimmten Automaten zugeordnet werden und gibt mittels CurrentStateID den Identifiert seines aktuellen Zustands preis.

Parameter

* `state` - Zustand, der sich geändert hat.

F.10.8 Klasse MailPanel

Das MailPanel definiert das Außenverhalten der Visualisierung der Nachrichtenverwaltung. Die Nachrichtenverwaltung besteht aus 3 Tabpages: dem MessageInboxPanel, dem SelectRecipientsPanel und dem ComposeMessagePanel, zwischen denen beliebig hin- und hergewechselt werden kann.

Konstruktoren

`public` **MailPanel**( )

Erzeugt ein neues MailPanel und initialisiert dessen Komponenten.

Parameter

* `mainFrame` - Das übergeordnete MainFrame
* `componentBuilder` - Der ComponentBuilder zur Erstellung der einzelnen Komponenten

F.10.9 Klasse MainFrame

Die Klasse MainFrame repräsentiert die Hauptkomponente der grafischen Benutzungsoberfläche und beinhaltet sämtliche anderen Komponenten der GUI.

Konstruktoren

`public` **MainFrame**( )

Erzeugt ein neues MainFrame zu der übergebenen Anwendung und initialisiert dieses mit den Standardeigenschaften.

Parameter

* `client` - Referenz auf die zugehörige Applikation

Methoden

`public` **void AttachToSubjects**( )

Weist das MainFrame an, sich als Observer bei allen gewünschten Modulen zu registrieren.

`public` **void ChangeCurrentMainViewPanelTo**( )

Wechselt das aktuelle Hauptanzeigefenster gegen das übergebene Hauptanzeigefenster aus.

Parameter

* `panel` - das neue Hauptanzeigefenster

`public` **void CloakLevelChanged**( )

Wird aufgerufen, wenn das CloakLevel neu gesetzt wurde.

Parameter

* `newValue` - Der neues Fähigkeitenwert.

`public` **void DroppedVirtualItem**( )

wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand abgelegt hat.

Parameter

* `Items` - virtueller Gegenständ, den der Nutzer abgelegt hat.

`public` **void OtherProfilesChanged**( )

Wird aufgerufen, wenn sich die Liste der Profile anderer Spieler ändert.

`public` **void PickedUpVirtualItem**( )

wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand aufgenommen hat.

Parameter

* `Items` - virtueller Gegenständ, den der Nutzer aufgenommen hat.

`public` **void PoiChanged**( )

Wird aufgerufen, wenn ein Point of Interest seinen Zustand ändert. Dabei kann es z.B. sich um eine Positionsänderung handeln.

Parameter

* `poi` - Point of Interest, der seinen Zustand (z.B. seine Position) geändert hat

`public` **void PoiDiscovered**( )

Wird aufgerufen, wenn dem Client ein neuer Point of Interest vom Server bekannt gegeben wird.

Parameter

* `poi` - neu entdeckter Point of Interest

`public` **void PoiLost**( )

Wird aufgerufen, wenn ein Point of Interest aus dem Sichtbereich des Clients verschwindet. Der Server überträgt nun keine Zustandsänderungen des Point of Interest mehr an den Client, bis PoiDiscovered(PointOfInterest) für den Point of Interest wieder aufgerufen wird. Dem Client steht es frei, die dem Nutzer die zuletzt bekannte Position des Point of Interest weiterhin anzuzeigen, es wird aber empfohlen, dem Nutzer kenntlich zu machen, dass die angezeigte Position nicht aktuell sein muss.

Parameter

* `poi` - Point of Interest, der ausser Sichtweite geraten ist

`public` **void PositionUpdated**( )

Wird aufgerufen, wenn der Client seine Position geändert hat. Die Positionsdaten umfasst Längengrad, Breitegrad und Höhe der aktuellen Spielerposition. Die Änderung der qualifizierten Daten, wie z.B. der Positionqualität wird über die Methode QualifiedDataChanged(QualifiedPosition) bekanntgegeben.

Parameter

* `position` - Positionsobjekt, das die aktuelle Spielerposition enthält

`public` **void QualifiedDataChanged**( )

Wird aufgerufen, wenn sich die qualifizierten Positionsdaten geändert haben. Die qualifizierten Daten umfassen die Qualität der Positionsdaten sowie Bewegungsrichtung und Bewegungsgeschwindigkeit des Spielers. Diese Methode wird jedoch nicht aufgerufen, wenn sich die geographische Position des Spielers geändert hat. In diesem Fall wird die Änderung über die Methode PositionUpdated(Position) bekanntgegeben.

Parameter

* `qualifiedPosition` - Position, deren qualifizierte Daten sich geändert haben

`public` **void StateChanged**( )

wird aufgerufen, wenn sich ein Zustand geändert hat. Der Zustand kann über seine StateID einem bestimmten Automaten zugeordnet werden und gibt mittels CurrentStateID den Identifiert seines aktuellen Zustands preis.

Parameter

* `state` - Zustand, der sich geändert hat.

`public` **void TeamValueChanged**( )

Wird aufgerufen, wenn sich die Teamzugehörigkeit eines Spielers geändert hat.

Parameter

* `team` -

`public` **void UserNameChanged**( )

wird aufgerufen, wenn sich das Pseudonym der Nutzers geändert hat

Parameter

* `userName` - neues Pseudonym

`public` **void WiretapLevelChanged**( )

Wird aufgerufen, wenn das WiretapLevel neu gesetzt wurd.

Parameter

* `newValue` - Der neues Fähigkeitenwert.

F.10.10 Klasse MainViewPanel

MainViewPanel ist die abstrakte Oberklasse für alle Hauptansichten. Während TabPanel, ControlsPanel, TickerPanel und LabelPanel immer sichtbar sind, kann immer nur eine Hauptansicht zu einem Zeitpunkt um Vordergrund sein. Über das TabPanel kann zwischen den verschiedenen Hauptansichten gewechselt werden.

Konstruktoren

`public` **MainViewPanel**( )

Erzeugt ein neues MainViewPanel und initialisiert dessen Größe und Position.

Parameter

* `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
* `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen

F.10.11 Klasse MapPanel

Das MapPanel definiert das Außenverhalten der Kartenansicht. Auf der Karte werden aller für den Client sichtbaren Spielobjekte, wie Spieler, Goodies, Kameras o.ä. dargestellt. Ausserdem stellt es die Möglichkeiten zur Interaktion des Benutzers mit der Karte (durch zoomen und verschieben) und mit den angezeigten Spielobjekten zur Verfügung.

Konstruktoren

`public` **MapPanel**( )

Ezeugt ein neues Map Panel zum übergebenen MainFrame.

Parameter

* `mainFrame` - Das übergeordnete Hauptanzeigefenster
* `componentBuilder` - Die Fabrik, aus der die Elemente des MapPanel erstellt werden

Methoden

`public` **Common.Logic.Pois.QualifiedPosition GetMapCenterPosition**( )

Liefert die GPS-Position des aktuellen Mittelpunkts des Kartenausschnitts.

`public` **System.Drawing.Point GPSToPixel**( )

Rechnet die übergebenen GPS-Koordinaten eines Objekts auf einen Punkt auf der Karte um.

Parameter

* `position` - Position eines Objektes, das auf der Karte dargestellt werden soll

`public` **void Init**( )

Initialisiert die Komponenten des MapPanels. Diese Methode wird im SetupPanel aufgerufen, wenn die Eingaben des Benutzers bestätigt wurden und dieser einem laufenden Spiel beigetreten ist.

`public` **Common.Logic.Pois.QualifiedPosition PixelToGPS**( )

Rechnnet einen Punkt auf der Karte in eine GPS Position um.

Parameter

* `x` - Entfernung des Punkts vom linken Kartenrand in Pixeln
* `y` - Entfernung des Punkts vom oberen Kartenrand in Pixeln

F.10.12 Klasse MenuPanel

Das MenuPanel definiert das Aussenverhalten des Hauptmenus. Hier werden der Benutzername, die Punkteverteilung der 4 Grungfähigkeiten sowie die aufgenommenen Goodies angezeigt. Die Goodies können im MenuPanel aktiviert und auch wieder abgelegt werden. Wenn der betreffende Spieler der Commander ist, wird, eine entsprechende Nähe zum Missionsziel vorausgesetzt, das Missionsziel angezeigt und dieses kann gelöst werden.

Konstruktoren

`public` **MenuPanel**( )

Erstellt ein neues MenuPanel zum übergebenen Hauptfenster und initialisiert dessen Komponenten.

Parameter

- `mainFrame` - Das übergeordnete Hauptanzeigefenster
- `componentBuilder` - Die Fabrik, aus der die Elemente des SetupPanel erstellt werden

F.10.13 Klasse MessageInboxPanel

Diese Klasse ist die abstrakte Oberklasse für das Fenster zur Übersicht über empfangene Nachrichten. Hier werden alle empfangenen Nachrichten angezeigt und können gelesen, gelöscht beantwortet oder weitergeleitet werden.

Konstruktoren

`public` **MessageInboxPanel**( )

Erzeugt einen neue Standardimplementierung eines MessageInboxPanel.

Parameter

- `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
- `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen

Methoden

`public` **void MessageReceived**( )

Wird aufgerufen, wenn eine neue Nachricht empfangen wurde.

Parameter

- `message` - empfangene Nachricht

F.10.14 Klasse PanelFactory

Die abstrakte Klasse PanelFactory ist nach dem Design Pattern AbstractFactory die abstrakte Fabrik für die Hauptkomponenten des MainFrames.

Konstruktoren

`public` **PanelFactory**( )

Erzeugt eine Standardimplementierung der PanelFactory.

Parameter

- `componentBuilder` - Factory, mit der die GUI Komponenten erzeugt werden sollen

Methoden

`public` **Client.Userinterface.Gui.ComposeMessagePanel CreateComposeMessagePanel**( )

Erzeugt ein neues Panel, in dem neue Nachrichten erstellt werden können.

`public` **Client.Userinterface.Gui.ContextPanel CreateContextPanel**( )

Erzeugt ein neues MainViewPanel, das den vom ContextManager generierten Kontext visualisiert

`public` **Client.Userinterface.Gui.ControlsPanel CreateControlsPanel**( )

Erzeugt ein neues Panel, in dem Steuerungskomponenten für die 4 Grundfähigkeiten Tarnen, Abhören, Aufklären und Neutralisieren untergebracht sind.

Parameter

* `location` - Koordinate der linken oberen Ecke des ControlsPanel
* `size` - Größe des ControlsPanel

`public` **Client.Userinterface.Gui.LabelPanel CreateLabelPanel**( )

Erzeugt ein neues Panel das angibt, welche Ausgabemodalitäten zur Zeit verwendet werden und ob sich bestimmte Points-of-Interest (Gegner, gegerische Kameras) in der Nähe befinden.

Parameter

* `location` - Koordinate der linken oberen Ecke des LabelPanel
* `size` - Größe des LabelPanel

`public` **Client.Userinterface.Gui.MailPanel CreateMailPanel**( )

Erzeugt ein neues MainViewPanel, das die Nachrichtenübersicht visualisiert.

`public` **Client.Userinterface.Gui.MapPanel CreateMapPanel**( )

Erzeugt ein neues MainViewPanel, das die Kartenansicht visualisiert.

`public` **Client.Userinterface.Gui.MenuPanel CreateMenuPanel**( )

Erzeugt ein neues MainViewPanel, das das Spielerprofil und die aufgenommenen Goodies visualisiert.

`public` **Client.Userinterface.Gui.MessageInboxPanel CreateMessageInboxPanel**( )

Erzeugt ein neues Panel, in dem die empfangenen Nachrichten angezeigt werden.

`public` **Client.Userinterface.Gui.SelectRecipientsPanel CreateSelectRecipientsPanel**( )

Erzeugt ein neues Panel, in dem die Empfänger einer Nachricht ausgewählt werden können.

`public` **Client.Userinterface.Gui.SetupPanel CreateSetupPanel**( )

Erzeugt ein neues MainViewPanel, das die Einstellungen vor Spielbeginn visualisiert

`public` **Client.Userinterface.Gui.TabPanel CreateTabPanel**( )

Erzeugt ein neues TabPanel, mit dem zwischen den einzelnen MainViewPanels umgeschaltet werden kann.

Parameter

* `location` - Koordinate der linken oberen Ecke des TabPanel
* `size` - Größe des TabPanel

`public` **Client.Userinterface.Gui.TickerPanel CreateTickerPanel**( )

Erzeugt ein neues Panel, über das Zustände und Ereignisse textuell ausgegeben werden können.

Parameter

* `location` - Koordinate der linken oberen Ecke des TickerPanel
* `size` - Größe des TickerPanel

F.10.15 Klasse SelectRecipientsPanel

Diese Klasse ist die abstrakte Oberklasse des Fensters zur Auswahl der Empfänger einer Nachricht. In diesem Fenster werden alle möglichen Empfänger einer Nachricht angezeigt und können als solche selektiert werden.

Konstruktoren

`public` **SelectRecipientsPanel**( )

Erzeugt ein neues Panel zur Auswahl der Empfänger einer Nachricht zum übergebenen MainFrame und ComponentBuilder

Parameter

- `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
- `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen

Methoden

`public` **void CloakLevelChanged**( )

Wird aufgerufen, wenn das CloakLevel neu gesetzt wurde.

Parameter

- `newValue` - Der neues Fähigkeitswert

`public` **void DroppedVirtualItem**( )

Wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand abgelegt hat.

Parameter

- `Items` - Virtueller Gegenständ, den der Nutzer abgelegt hat

`public` **void OtherProfilesChanged**( )

Wird aufgerufen, wenn sich die Liste der Profile anderer Spieler geändert hat.

`public` **void PickedUpVirtualItem**( )

Wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand aufgenommen hat.

Parameter

- `Items` - Virtueller Gegenständ, den der Nutzer aufgenommen hat

`public` **void TeamValueChanged**( )

Wird aufgerufen, wenn sich die Teamzugehörigkeit eines Spielers geändert hat.

Parameter

- `team` - Das neue Team des Spielers

`public` **void UserNameChanged**( )

Wird aufgerufen, wenn sich das Pseudonym des Nutzers geändert hat.

Parameter

* `userName` - neues Pseudonym

`public` **void WiretapLevelChanged**( )

Wird aufgerufen, wenn das WiretapLevel neu gesetzt wurde.

Parameter

* `newValue` - Der neues Fähigkeitswert

F.10.16 Klasse SetupPanel

Diese Klasse repräsentiert das erste Anzeigefenster der GUI. In diesem Fenster müssen die für den Spielstart relevanten Benutzereingaben getätigt werden. Diese umfassen die Eingabe eines Spielernamens, die Angabe der IP-Adresse des NABB-Servers, auf dem gespielt werden soll inklusiver der Portnummer, die Auswahl der zu spielenden Szenariodatei sowie die Verteilung der Fähigkeitspunkte auf die 4 zur Verfügung stehenden Hauptfähigkeiten 'Tarnen', 'Aufklären', 'Abhören' und 'Neutralisieren'. Nach Bestätigung der Benutzereingaben und Überprüfung deren Korrektheit werden die einzelnen Module des Spiels initialisiert und es wird versucht, eine Verbindung zum angegebenen NABB-Server aufzubauen. Existiert der angegebene Server, so tritt der Benutzer dem laufenden Spiel bei und kann an diesem teilnehmen.

Konstruktoren

`public` **SetupPanel**( )

Erzeugt ein neues Setup Panel zum übergebenen MainFrame und ComponentBuilder und initialisiert dieses mit den Standardeigenschaften.

Parameter

* `mainFrame` - Das übergeordnete Hauptanzeigefenster
* `componentBuilder` - Die Fabrik, aus der die Elemente des SetupPanel erstellt werden

Methoden

`public` **Client.Logic.SetupData CreateSetupData**( )

Erstellt aus den vom Benutzer eingegebenen Informationen die Daten, die für ein Anmelden des Clients beim Server notwendig sind.

`public` **void ShowConnectionTimeoutException**( )

Zeigt das Auftreten einer ConnectionTimeoutException in einer Messagebox an.

`public` **void ShowException**( )

Zeigt eine aufgetretene Exception an.

Parameter

* `e` - Die Exception.

`public` **void ShowGeneralServerException**( )

Zeigt das Auftreten einer GeneralServerException in einer Messagebox an.

`public` **void ShowInvalidNameException**( )

Zeigt das Auftreten einer InvalidNameException in einer Messagebox an.

F.10.17 Klasse TabPanel

Diese Klasse ist die abstrakte Oberklasse für das Panel, über das zwischen den möglichen Ansichten der Hauptansicht navigiert werden kann.

Konstruktoren

`public` **TabPanel**( )

Erzeugt ein neues TabPanel und initialisiert dieses mit den übergebenen Werten.

Parameter

- `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
- `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen
- `location` - Position der linken oberen Ecke des Panels
- `size` - Höhe und Breite des Panels

Methoden

`public` **void SetEnabled**( )

Gibt die Tabs des Panels für Eingaben frei, bzw. graut sie aus, so dass sie keine Eingabe mehr akzeptieren.

Parameter

- `enabled` - true, wenn die Tabs anwählbar sein sollen, false sonst

`public` **void SetMapTabEnabledOnStartup**( )

Setzt den Tab zur Kartenansicht als 'momentan ausgewählt'.

F.10.18 Klasse TickerPanel

Diese Klasse ist die abstrakte Oberklasse für das Panel, über das Zustände und Ereignisse textuell ausgegeben werden können.

Konstruktoren

`public` **TickerPanel**( )

Erzeugt ein neues TickerPanel und initialisiert dieses mit den übergebenen Werten.

Parameter

- `mainFrame` - Referenz auf die Hauptansicht, zu dem dieses Panel gehört
- `location` - Position der linken, oberen Ecke des Panels
- `size` - Höhe und Breite des Panels
- `componentBuilder` - Factory, mit der die GUI Komponenten dieses Panels erzeugt werden sollen

Methoden

`public` **void AppendText**( )

Ermöglicht die Ausgabe des übergebenen Textes im TickerPanel. Hierzu werden z.B. eingehende Events und Systemnachrichten an die tickerQueue angehängt und dann über das TickerPanel ausgegeben.

Parameter

* `text` - Eingegangenes Event oder Systemnachricht

F.11 Namespace Client.Userinterface.Modalitymanager

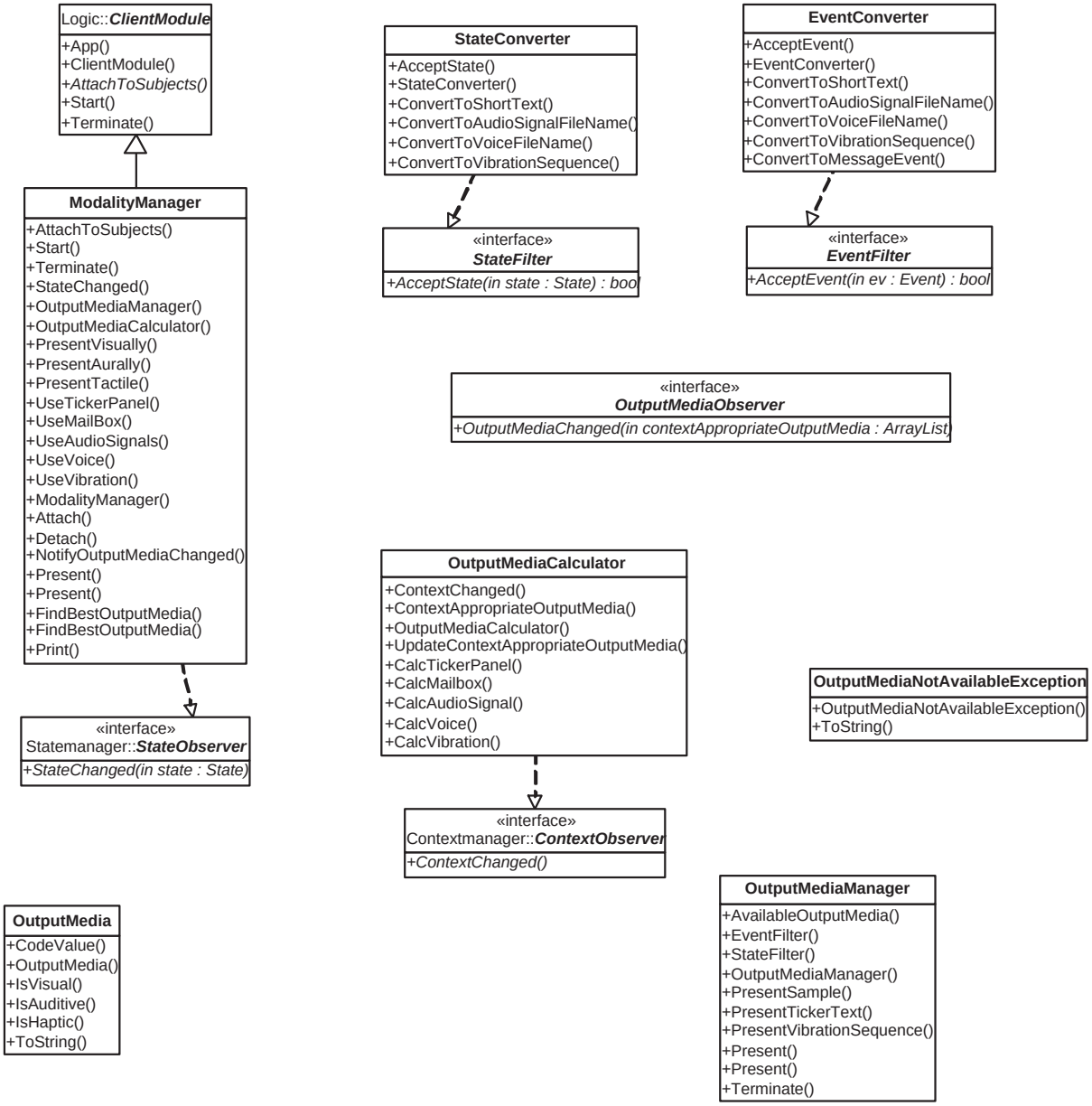


Abbildung F.11: UML Struktur des Namensraum Client.Userinterface.Modalitymanager

F.11.1 Schnittstelle OutputMediaObserver

Definiert die Schnittstelle für Module, die die Änderung der Ausgabemedien verfolgen wollen.

Methoden

`public` **void OutputMediaChanged**( )

Wird aufgerufen, wenn sich die dem Kontext angemessenen Ausgabemedien geändert haben, weil sich z.B. der Kontext des Spielers geändert hat.

Parameter

* `contextAppropriateOutputMedia` - Liste der Ausgabemedien, in dem aktuellen Kontext zur Präsentation von Informationen angemessen sind

F.11.2 Klasse ModalityManager

Der ModalityManager ermittelt aus den vom Kontextmanager berechneten Kontextanforderungen, den vom Nutzer eingestellten Präferenzen, den Möglichkeiten der Benutzungsschnittstelle eine oder mehrere geeignete Präsentationsmodalitäten für die Ausgabe von Zuständen und Ereignissen. Der ModalityManager übernimmt anhand bestimmter Kriterien die Auswahl der Präsentationsmodalität aller auszugebenden Informationen. Soll ein Event ausgegeben werden, wird die entsprechende Ausgabemethode des OutputMediaManagers aufgerufen.

Konstruktoren

`public` **ModalityManager**( )

Erzeugt neues ModalityManager Modul.

Parameter

* `clientApplication` - Anwendung zu der dieser ModalityManager erzeugt wird

Methoden

`public` **void Attach**( )

Fügt dem ModalityManager einen OutputMediaObserver hinzu.

Parameter

* `observer` - Hinzuzufügender Observer

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert. Der ModalityManager registriert sich beim ContextManager als Observer, um im Falle einer Änderung des Kontexts die Ausgabemodalitäten neu berechnen zu können. Er registriert sich außerdem beim StateManager als Observer, um bei einem Zustandswechsel den neuen Zustand präsentieren zu können.

`public` **void Detach**( )

Fügt dem ModalityManager einen OutputMediaObserver hinzu.

Parameter

* `observer` - Zu entfernender Observer

`public` **System.Collections.ArrayList FindBestOutputMedia**( )

Sucht das Ausgabemedium, über das der angegebene Zustand präsentiert werden soll.

Parameter

* `state` - Darzustellender Zustand, zu dem das passende Ausgabemedium gefunden werden soll

`public` **System.Collections.ArrayList FindBestOutputMedia**( )

Sucht das Ausgabemedium, über das das angegebene Ereignis präsentiert werden soll.

Parameter

* `e` - Darzustellendes Ereignis, zu dem das passende Ausgabemedium gefunden werden soll

`public` **void Present**( )

Stellt ein Event über die Nutzungsschnittstelle dar.

Parameter

* `e` - Darzustellendes Event.

`public` **void Present**( )

Stellt einen Zustand über die Nutzungsschnittstelle dar.

Parameter

* `state` - Darzustellender Zustand.

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll. Der ModalityManager nutzt diese Methode, um initial die zu verwendenden Ausgabemedien zu berechnen.

`public` **void StateChanged**( )

Wird aufgerufen, wenn sich ein Zustand geändert hat. Der Zustand kann über seine StateID einem bestimmten Automaten zugeordnet werden und gibt mittels CurrentStateID den Identifiert seines aktuellen Zustands preis.

Parameter

* `state` - Zustand, der sich geändert hat

`public` **void Terminate**( )

Diese Methode wird von der Applikation aufgerufen, wenn sie sich beendet. Jedes Modul kann sie überschreiben, um vor dem Beenden z.B. belegte Ressourcen freizugeben. Der ModalityManager nutzt diese Methode, um eventuell mit den Ausgabekanälen assoziierte Ressource freizugeben.

F.11.3 Klasse OutputMedia

Diese Klasse kapselt die Informationen zu einem Ausgabemedium.

Konstruktoren

`public` **OutputMedia**( )

Erzeugt ein neues Audgabemedium vom übergebenen Typ.

Parameter

* `mediaCode` - Kodierung des Ausgabekanals, siehe OutputMedia.MediaCode

Methoden

`public` **bool IsAuditive**( )

Überprüft, ob das Medium den Hörsinn anspricht.

`public` **bool IsHaptic**( )

Überprüft, ob das Medium den Tastsinn anspricht.

`public` **bool IsVisual**( )

Überprüft, ob das Medium den Sehsinn anspricht.

`public` **string ToString**( )

Liefert die textuelle Beschreibung des Ausgabemediums

F.11.4 Klasse OutputMediaCalculator

Diese Klasse berechnet anhand des Kontext und der zur Verfügung stehende Ausgabekanäle, den oder die geeigneten Ausgabekanäle.

Konstruktoren

`public` **OutputMediaCalculator**( )

Erzeugt neues OutputMediaCalculator Objekt.

Parameter

- `modalityManager` - Referenz auf den ModalityManager, damit die Observer benachrichtig werden können, wenn sich die angemessenen Ausgabenmedien ändern
- `manager` - Referenz auf den OutputMediaManager, damit die zur Verfügung stehenden Ausgabekanäle abgefragt werden können

Methoden

`public` **bool CalcAudioSignal**( )

Überprüft, ob die Ausgabe von Audiosignalen in dem augenblicklichen Kontexts des Spielers ein geeignetes Medium zur Präsentation von Informationen ist.

Parameter

- `context` - Interpretierter Kontext des Spielers

`public` **bool CalcMailbox**( )

Überprüft, ob die Mailbox in dem augenblicklichen Kontexts des Spielers ein geeignetes Medium zur Präsentation von Informationen ist.

Parameter

- `context` - Interpretierter Kontext des Spielers

`public` **bool CalcTickerPanel**( )

Überprüft, ob das TickerPanel in dem augenblicklichen Kontexts des Spielers ein geeignetes Medium zur Präsentation von Informationen ist.

Parameter

* `context` - Interpretierter Kontext des Spielers

`public` **bool CalcVibration**( )

Überprüft, ob die Ausgabe von Vibrationssignalen in dem augenblicklichen Kontexts des Spielers ein geeignetes Medium zur Präsentation von Informationen ist.

Parameter

* `context` - Interpretierter Kontext des Spielers

`public` **bool CalcVoice**( )

Überprüft, ob die Ausgabe von Sprachnachrichten in dem augenblicklichen Kontexts des Spielers ein geeignetes Medium zur Präsentation von Informationen ist.

Parameter

* `context` - Interpretierter Kontext des Spielers

`public` **void ContextChanged**( )

Zeigt an, dass sich der Kontext geändert hat.

F.11.5 Klasse OutputMediaManager

Klasse zur Verwaltung der Ausgabemedien.

Konstruktoren

`public` **OutputMediaManager**( )

Erzeugt neuen OutputMediaManager.

Parameter

* `app` - Referenz auf die Client Anwendung

Methoden

`public` **void Present**( )

Präsentiert einen Zustand über die angebenen Ausgabemedien.

Parameter

* `state` - Zu präsentierender Zustand
* `media` - Ausgabemedien, die zur Präsenation des Zustands verwendet werden sollen

`public` **void Present**( )

Präsentiert ein Ereignis über die angebenen Ausgabemedien.

Parameter

* `ev` - Zu präsentierendes Ereignis
* `media` - Ausgabemedien, die zur Präsenation des Ereignis verwendet werden sollen

`public` **void PresentSample**( )

Erzeugt einen AudioPlayer und spielt ein Sample ab.

Parameter

* `soundFile` - Das abzuspielende Sample

`public` **void PresentTickerText**( )

Zeigt eine Textnachricht im Tickerpanel an.

Parameter

* `text` - Der anzuzeigende Text

`public` **void PresentVibrationSequence**( )

Gibt eine Vibrationssequenz aus.

Parameter

* `vibrationSequence` - Die Vibrationssequenz

F.11.6 Klasse OutputMediaNotAvailableException

Wird geworfen, wenn eine Nachricht zur Ausgabe durch ein bestimmtes Medium umgewandelt werden soll, dass nicht zur Verfügung steht.

Konstruktoren

`public` **OutputMediaNotAvailableException**( )

Erzeugt neue Instanz der Ausnahme.

Parameter

* `code` - Kodierung des Ausgabemediums, das nicht zur Verfügung steht

Methoden

`public` **string ToString**( )

Überschreibt die Methode ToString() der Klasse ApplicationException, so dass zusätzlich das mit dieser Ausnahme assoziierte Ausgabemedium ausgegeben wird.

F.12 Namespace Common.Communication

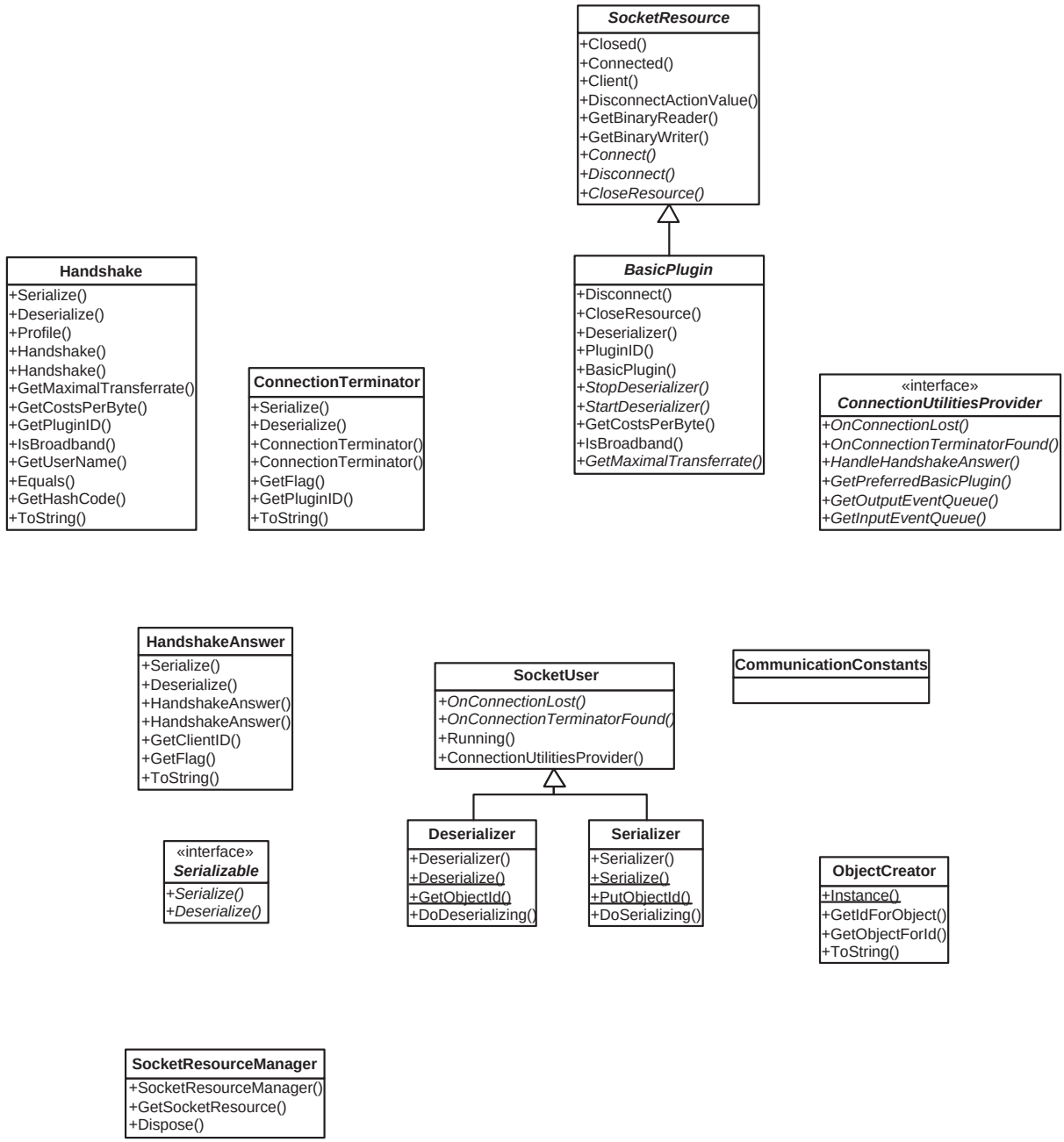


Abbildung F.12: UML Struktur des Namensraum Common.Communication

F.12.1 Schnittstelle ConnectionUtilitiesProvider

Die Scnittstelle ConnectionUtilitiesProvider definiert die delegate-Signaturen von SocketUser, die für die Events (ConnectionLost und ConnectionTerminatorFound, von Serializer und Deserializer ausgelöst) benötigt werden. Außerdem liefert ein ConnectionUtilitiesProvider die EventQueues und ein BasicPlugin zum senden oder empfangen von Events. Er definiert auch die Methodensignatur zu abarbeiten einer HandshakeAnswer. Auf Clientseite ist der ConcreteNetworkManager ein ConnectionUtilitiesProvider und auf Serverseite die NetworkSession.

Methoden

`public` **Common.Logic.EventQueue GetInputEventQueue**( )

Liefert die EventQueue des Clients. Alle Objekte aus ihr werden serialisiert und verschickt.

`public` **Common.Logic.EventQueue GetOutputEventQueue**( )

Liefert die EventQueue der Application. Deserialisierte Objekte werden ihr hinzugefügt.

`public` **Common.Communication.BasicPlugin GetPreferredBasicPlugin**( )

Liefert ein BasicPlugin zum senden oder empfangen von Objekten.

`public` **void HandleHandshakeAnswer**( )

Behandelt ein HandleHandshakeAnswer-Objekt. Entscheidet, ob noch ein Confirm gesendet werden muss, oder ob das BasicPlugin sofort geschlossen werden kann.

Parameter

- `handshakeAnswer` - das HandshakeAnswer-Objekt

`public` **void OnConnectionLost**( )

Behandelt ein ConnectionLost-Event.

Parameter

- `sender` - das object, welches das Event ausgelöst hat
- `plugin` - das betroffene Plugin

`public` **void OnConnectionTerminatorFound**( )

Behandelt ein ConnectionTerminatorFound-Event.

Parameter

- `sender` - das object, welches das Event ausgelöst hat
- `connectionTerminator` - das ConnectionTerminator-Objekt

F.12.2 Schnittstelle Serializable

Diese Schnittstelle definiert die Signaturen der Methoden, mit denen ein Objekt sich in seine serialisierte Form (eine Folge aus primitiven Datentypen) zerlegt und wie es sich aus seiner serialisierten Form wieder zusammensetzt. Jedes Objekt, das über den Communication-Layer versendet werden soll, muss diese Schnittstelle implementieren.

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transitiven primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-transitiven primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.12.3 Klasse BasicPlugin

Basisklasse für alle Plugins. Beinhaltet einen Deserializer zum empfangen von Serializable-Objekten und zum beenden der Verbindung. Das BasicPlugin verwaltet den Socket und den Deserializer-Thread.

Konstruktoren

`public` **BasicPlugin**( )

Erzeugt ein neues BasicPlugin-Objekt.

Parameter

* `pluginID` - Die zugeweisenen ID des Plugins.

Methoden

`public` **void CloseResource**( )

Beendet die Verbindung dieses Plugins zum Server, abhängig von DisconnectActionValue.

`public` **void Disconnect**( )

Beendet die Verbindung diese Plugins zum Server mit einem WArten auf ein Confirm.

`public` **double GetCostsPerByte**( )

Liefert die Kosten pro versendetem Byte.

`public` **int GetMaximalTransferrate**( )

Liefert die maximale Transferrate in Bytes pro Sekunde.

`public` **bool IsBroadband**( )

Liefert ein Boolean ob ein Breitbandanschluss vorliegt.

`public` **void StartDeserializer**( )

Startet den Deserialisierungprozess dieses Plugins.

`public` **void StopDeserializer**( )

Stoppt den Deserialisierungprozess dieses Plugins.

F.12.4 Klasse CommunicationConstants

Speichert einige konstante Werte für Threads.

Konstruktoren

`public` **CommunicationConstants**( )

F.12.5 Klasse ConnectionTerminator

Die Klasse ConnectionTerminator wird benutzt, um eine Verbindung definiert schließen bzw. das Schließen bestätigen zu können. Das Abmelden eines Plugins geschieht mit Hilfe des ConnectionTerminators. Die abmeldende Seite (meistens der Client) schickt einen Closing-ConnectionTerminator und erwartet einen ConnectionTerminator mit einem ClosingConfirm-Flag vom der abmeldenden Seite (meistens der Server).

Konstruktoren

`public` **ConnectionTerminator**( )

Erzeugt ein neues ConnectionTerminator-Object.

`public` **ConnectionTerminator**( )

Erzeugt ein neues ConnectionTerminator-Objekt mit den als Parameter übergebenen Attributen.

Parameter

- `pluginID` - Die ID eines Plugins.
- `flag` - Eine Flag-Enumeration.

Methoden

`public` **void Deserialize**( )

Implementiert die Methode Deserialize() des Interfaces Serializable.

Parameter

- `binaryReader` - Das BinaryReader-Objekt, das die Attribute der Klasse aus einem Input-Stream liest.

`public` **Common.Communication.ConnectionTerminator.Flag GetFlag**( )

Liefert eine Flag-Enumeration.

`public` **int GetPluginID**( )

Liefert die ID eines Plugins.

`public` **void Serialize**( )

Implementiert die Methode Serialize() des Interfaces Serializable.

Parameter

- `binaryWriter` - Das BinaryWriter-Objekt, das die Attribute der Klasse in einen Output-Stream schreibt.

`public` **string ToString**( )

Überschreibt die object.ToString()-Methode.

F.12.6 Klasse Deserializer

Der Deserializer emfängt Datenströme und serialisiert diese zu den von Event abgeleiteten Objekten. Mit Hilfe des ObjectCreators wird der korrekte Objekttyp instanziizert. Der Serialisierungsvorgang läuft innerhalb eines Threads, um andere Funktionen nicht zu blockieren.

Konstruktoren

`public` **Deserializer**( )

Erzeugt ein neues Deserializer-Objekt und registriert den ConnectionUtilitiesProvider für ConnectionLost- und ConnectionTerminatorFound-Events.

Parameter

* `connectionUtilitiesProvider` - der ConnectionUtilitiesProvider
* `basicPlugin` - das BasicPlugin, zu dem der Deserializer gehört

Methoden

`public` **void Deserialize**( )

Diese Methode deserialisiert ein Objekt, indem es die deserialize(binaryReader)- Methode beim zu deserialisierenden Objekt aufruft.

Parameter

* `binaryReader` - Das BinaryReader-Objekt.
* `serializable` - Das Serializable-Objekt.

`public` **void DoDeserializing**( )

Diese Methode wird innerhalb eines Threads aufgerufen. Die doDeserializing()- Methode überprüft alle 500 Millisekunden, ob ein InputStream beim Plugin-Objekt eingetroffen ist. Dieses Polling wird erzwungen durch die Tatsache, dass der Zugriff auf eine SocketResource synchronisiert werden muss. Innerhalb dieser Synchronisation darf sich der Thread aber nicht schlafen legen, da ansonsten keine Nachrichten mehr verschickt werden können (über diese SocketResource).

`public` **int GetObjectId**( )

Das BinaryReader-Objekt liest die Objekt-ID aus einem InputStream.

Parameter

* `binaryReader` - Das BinaryReader-Objekt.

F.12.7 Klasse Handshake

Mit einem Objekt des Typs Handshake meldet sich ein Client bei dem Server an. Das Handshake enthält Informationen über den Nutzer (das UserProfile) und über die Eigenschaften des Plugins. Anhand seiner ID wird das Plugin auf dem Server identifiziert.

Konstruktoren

`public` **Handshake**( )

Erzeugt ein neues Handshake-Objekt.

`public` **Handshake**( )

Erzeugt ein neues Handshake-Objekt mit den als Parameter übergebenen Attributen.

Parameter

- `userProfile` - Das UserProfile-Objekt.
- `costsPerByte` - Die Kosten einer Verbindung.
- `maximalTransferrate` - Die maximale Transferrate.
- `isBroadband` - Gibt an, ob die Verbindung eine Breitbandverbindung ist.
- `pluginID` - Die ID eines Plugins.

Methoden

`public` **void Deserialize**( )

Implementiert die Methode Deserialize() des Interfaces Serializable.

Parameter

- `binaryReader` - Das BinaryReader-Objekt, das die Attribute der Klasse aus einem Input-Stream liest.

`public` **bool Equals**( )

Vergleicht ein Objekt mit dem Handshake.

Parameter

- `obj` - das zu vergleichende Objekt

`public` **double GetCostsPerByte**( )

Liefert die Kosten einer Verbindung.

`public` **int GetHashCode**( )

`public` **int GetMaximalTransferrate**( )

Liefert die maximale Transferrate einer Verbindung.

`public` **int GetPluginID**( )

Liefert die ID eines Plugins.

`public` **string GetUserName**( )

Liefert den Benutzernamen.

`public` **bool IsBroadband**( )

Sagt aus, ob eine Breitbandverbindung vorliegt.

`public` **void Serialize**( )

Implementiert die Methode Serialize() des Interfaces Serializable.

Parameter

- `binaryWriter` - Das BinaryWriter-Objekt, das die Attribute der Klasse in einen Output-Stream schreibt.

`public` **string ToString**( )

Überschreibt die object.ToString()-Methode.

F.12.8 Klasse HandshakeAnswer

Die Klasse HandshakeAnswer wird benutzt, um den Zustand eines Handshakes darzustellen. Sie wird als Antwort auf einen Handshake vom Server gesendet.

Konstruktoren

`public` **HandshakeAnswer**( )

Erzeugt ein neues HandshakeAnswer-Objekt.

`public` **HandshakeAnswer**( )

Erzeugt ein neues HandshakeAnswer-Objekt mit den als Parameter übergebenen Attributen.

Parameter

- `clientID` - Die ID des Clients.
- `flag` - Eine Flag-Enumeration.

Methoden

`public` **void Deserialize**( )

Implementiert die Methode Deserialize() des Interfaces Serializable.

Parameter

- `binaryReader` - Das BinaryReader-Objekt, das die Attribute der Klasse aus einem Input-Stream liest.

`public` **int GetClientID**( )

Liefert die ID des Clients.

`public` **Common.Communication.HandshakeAnswer.Flag GetFlag**( )

Liefert eine Enumeration vom Typ Flag.

`public` **void Serialize**( )

Implementiert die Methode Serialize() des Interfaces Serializable.

Parameter

- `binaryWriter` - Das BinaryWriter-Objekt, das die Attribute der Klasse in einen Output-Stream schreibt.

`public` **string ToString**( )

Überschreibt die object.ToString()-Methode.

F.12.9 Klasse ObjectCreator

Der ObjectCreator ordnet Identifikationsnummern Klassennamen zu. Er dient zu Instanzierung versendeter Objekte anhand ihrer Identifikationsnummer. Benutzt wird dafür der Reflection-Mechanismus von C-Sharp. Außerdem kann die Identifikationsnummern eines Objekts zum Versenden abgefragt werden. Die benötigten Klassennamen (und Namensräume) werden aus der Datei 'SerializingRegistry.xml' geladen.

Methoden

`public` **int GetIdForObject**( )

Sucht nach dem Eintrag (Name des Objekts) in der internen Klassennamenlist. Dieser Eintrag ist die Identifikationsnummer.

Parameter

* `obj` - Zu diesem Objekt soll die Identifikationsnummer gefunden werden.

`public` **object GetObjectForId**( )

Instanziiert anhand des Klassennamens, der in der internen Klassennamensliste, an dem durch den Parameter spezifizierten Index zu finden ist, das Objekt.

Parameter

* `id` - Zu dieser Identifikationsnummer soll das passende Objekt instanziiert werden

`public` **string ToString**( )

Überschreibt die object.ToString()-Methode.

F.12.10 Klasse Serializer

Der Serializer serialisert Events und versendet diese dadurch. Der Serialisierungsvorgang läuft innerhalb eines Threads, um andere Funktionen nicht zu blockieren.

Konstruktoren

`public` **Serializer**( )

Erzeugt ein neues Serializer-Objekt mit den als Parameter übergebenen Attributen und registriert einen Callback auf den ConnectionUtilitiesProvider.

Parameter

* `connectionUtilitiesProvider` - das ConnectionUtilitiesProvider-Objekt

Methoden

`public` **void DoSerializing**( )

Diese Methode wird innerhalb eines Threads aufgerufen. Die doSerializing()- Methode blockiert und wartet darauf, das der Warteschlange neue, zu serialisierende Objekte hinzugefügt werden. Wird die Verbindung bei dem Serialisierungsprozess unterbrochen, wird ein ConnectionLost-Event ausgelöst.

`public` **void PutObjectId**( )

Das BinaryWriter-Objekt schreibt die Objekt-ID in einen OutputStream.

Parameter

* `binaryWriter` - Das BinaryWriter-Objekt.
* `objectId` - Die Objekt-ID.

`public` **void Serialize**( )

Diese Methode serialisiert ein Objekt, indem es die serialize(binaryWriter)- Methode beim zu serialisierenden Objekt aufruft.

Parameter

- `binaryWriter` - Das BinaryWriter-Objekt.
- `serializable` - Das Serializable-Objekt.

F.12.11 Klasse SocketResource

SocketResource dient als Basisklasse für die Plugins. Objekte des Typs SocketResource werden mit Hilfe des SocketResourceManagers threadsicher gemacht. Eine SocketResource kapselt einen Socket (hier TcpClient) und dessen Zustand.

Methoden

`public` **void CloseResource**( )

Vorgegebene Signatur der CloseResource-Prozedur. Wird aufgerufen, um die Verbindung eines von SocketResource abgeleitetes Objekt mit einem Host zu trennen. Wie diese Trennung aussieht, bestimmt die Enumeration DisconnectAction. Mindestens aber wird der lokale Socket geschlossen.

`public` **void Connect**( )

Vorgegebene Signatur der Connect-Prozedur. Wird aufgerufen, um ein von SocketResource abgeleitetes Objekt mit einem Host zu verbinden.

Parameter

- `ip` - Die IP des Hosts, kodiert als string.
- `port` - Der Port, auf dem der Host wartet.
- `profile` - Das UserProfile, das bei der Anmeldung mit übertragen wird.

`public` **void Disconnect**( )

Vorgegebene Signatur der Disconnect-Prozedur. Trennt den Client vom Server.

`public` **System.IO.BinaryReader GetBinaryReader**( )

Liefert das BinaryReader-Objekt. Bei Nutzung von GetBinary-Writer/Reader in unterschiedlichen Threads muss der Zugriff durch den SocketResourceManager synchronisiert werden.

`public` **System.IO.BinaryWriter GetBinaryWriter**( )

Liefert das BinaryWrite-Objekt. Bei Nutzung von GetBinary-Writer/Reader in unterschiedlichen Threads muss der Zugriff durch den SocketResourceManager synchronisiert werden.

F.12.12 Klasse SocketResourceManager

Der SocketRessourceManager synchronisiert über das Dispose-Pattern den Zugriff auf ein SocketResource-Objekt. Hintergrund ist, das Sockets nicht threadsicher sind, und sowohl Serializer als auch Deserializer zu gleicher Zeit SocketResource-Objekt verwenden könnten. Die Verwendung des SocketRessourceManager verhindert dies.

Konstruktoren

`public` **SocketResourceManager**( )

Erzeugt ein neues SocketResourceManager-Objekt für eine SocketResource und ruft für die SocketResource Monitor.Enter auf.

Parameter

* `socketResource` - das SocketResource-Objekt

Methoden

`public` **void Dispose**( )

Implementierung von IDisposable. Eine Finalisierung ist nicht mehr nötig.

`public` **Common.Communication.SocketResource GetSocketResource**( )

Liefert die SocketResource.

F.12.13 Klasse SocketUser

SocketUser ist die Basisklasse für Serializer und Deserializer. Von SocketUser abgeleitete Objekte laufen in einem eigenen Thread. Die Klasse SocketUser bietet CallBack-Mechanismen für die Threads an und speichert, ob der Thread aktiv ist.

F.13 Namespace Common.Logic

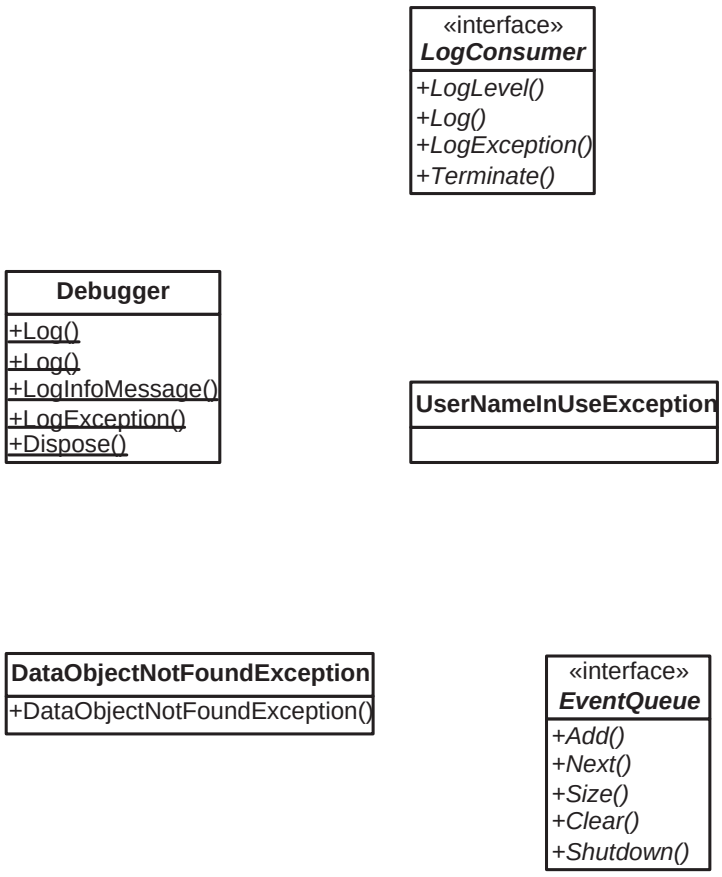


Abbildung F.13: UML Struktur des Namensraum Common.Logic

F.13.1 Schnittstelle EventQueue

Definiert die Schnittstelle für eine Warteschlange von Event Objekten. Events werden über Add(Event) in die Warteschlange eingefügt. Next() liefert das älteste Event aus der Warteschlange. Befindet sich zur Zeit des Aufrufs von Next() gerade kein Event in der Warteschlange, blockiert sie, bis entweder ein Event in die Warteschlange eingestellt wurde, oder die Warteschlange durch Shutdown() geschlossen wird.

Methoden

`public` **void Add**( )

Hängt der Warteschlange ein Event an.

Parameter

- `e` - Anzuhängendes Event

`public` **void Clear**( )

Leert die Warteschlange

`public` **Common.Logic.Statemanager.Event Next**( )

Holt das vorderste Event aus der Warteschlange heraus. Ist gerade kein Event in der Warteschlange, blockiert die Methode, bis entweder ein Event zur Verfügung steht, oder die Methode Shutdown() aufgerufen wurde. In letzterem Fall liefert diese Methoden NULL statt eines Events zurück.

`public` **void Shutdown**( )

Terminiert die Warteschlange. Die wartenden Events werden gelöscht. Eingehende Events werden ignoriert. Falls Threads in der Methode Next() blockiert sind, werden sie aufgeweckt und liefert NULL statt einem Event zurück.

`public` **int Size**( )

Liefert die Anzahl der wartenden Events.

F.13.2 Schnittstelle LogConsumer

Der LogConsumer ist die Definition der Schnittstellen für einen Logger. Ein Logger kann dem Debugger hinzugefügt werden, um dann die Nachrichten weiterzuverarbeiten, z.B. auf der Console auszugeben oder in eine Datei zu schreiben.

Methoden

`public` **void Log**( )

Gibt eine Nachricht über den Logger aus, wenn der angegebene Log-Level mindestens dem Log-Level dieses Loggers entspricht.

Parameter

- `msg` - auszugebende Nachricht
- `logLevel` - Log-Level der Nachricht

`public` **void LogException**( )

Gibt eine Ausnahme über den Logger aus, wenn der angegebene Log-Level mindestens dem Log-Level dieses Loggers entspricht.

Parameter

- `source` - Objekt in dem die Ausnahme aufgetreten ist
- `ex` - auszugebende Ausnahme
- `logLevel` - Log-Level der Ausnahme

`public` **void Terminate**( )

Gibt eventuell allokierte Ressourcen frei.

F.13.3 Klasse DataObjectNotFoundException

Wird aufgerufen, wenn aus einer Liste von Objekten ein Objekt mit einer ID herausgesucht werden soll, dass nicht gefunden werden kann.

Konstruktoren

`public` **DataObjectNotFoundException**( )

Erzeugt neue DataObjectNotFoundException.

Parameter

* `msg` - Fehlerbeschreibung

F.13.4 Klasse Debugger

Hauptklasse der eigens implementierten Logging API. Alle Module können über statische Methoden dieser Klasse ihre Aktionen loggen.

Konstruktoren

`public` **Debugger**( )

Methoden

`public` **void Dispose**( )

Gibt eventuell allokierte Ressourcen frei.

`public` **void Log**( )

Loggt eine Systemnachricht.

Parameter

* `obj` - Objekt, das diese Methode aufgerufen hat
* `msg` - Systemnachricht

`public` **void Log**( )

Loggt eine Systemnachricht.

Parameter

* `msg` - Systemnachricht

`public` **void LogException**( )

Loggt eine Ausnahme.

Parameter

* `sender` - Objekt, in dem die Ausnahme aufgetreten ist
* `e` - Ausnahme

`public` **void LogInfoMessage**( )

Loggt eine Debugnachricht.

Parameter

* `obj` - Objekt, das diese Methode aufgerufen hat
* `msg` - Debugnachricht

F.13.5 Klasse UserNameInUseException

Wird geworfen, wenn ein Spieler sich mit einem bereits vergebenen Namen beim Spiel anmelden will.

Konstruktoren

`public` **UserNameInUseException**( )

F.14 Namespace Common.Logic.Game

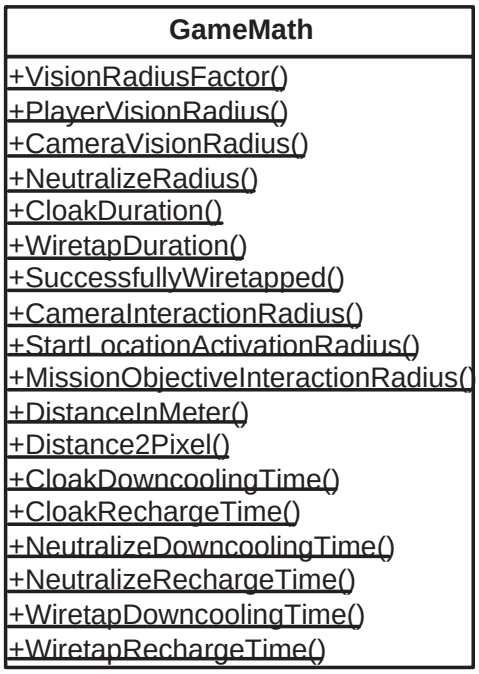


Abbildung F.14: UML Struktur des Namensraum Common.Logic.Game

F.14.1 Klasse GameMath

Diese Klasse dient zu Berechnung von Sichtradien und Entfernungen. Alle Distanzen sind Meter, soweit nicht anders angegeben.

Konstruktoren

`public` **GameMath**( )

Methoden

`public` **double CameraInteractionRadius**( )

Berechnet den Interaktionsradius einer Kamera (anhand seines Werts der Fähigkeit Aufklären) in Metern Wenn man sich innerhalb dieses Radius befindet, kann man seine eigenen Kameras aufnehmen und gegnerische Kameras deaktivieren.

Parameter

- `reconLevel` - Wert der Fähigkeit Aufklären

`public` **double CameraVisionRadius**( )

Berechnet den Sichtradius einer Kamera (anhand seines Werts der Fähigkeit Aufklären) in Metern

Parameter

- `reconLevel` - Wert der Fähigkeit Aufklären

`public` **int CloakDowncoolingTime**( )

Liefert die Zeit in ms, die 'Tarnen' braucht, um nach der letzten Benutzung wieder verfügbar zu sein.

`public` **int CloakDuration**( )

Liefert die maximale Tarndauer für einen Cloaklevel.

Parameter

* `cloakLevel` - Die Punkte auf Cloak.

`public` **int CloakRechargeTime**( )

Liefert die Zeit in ms, die 'Tarnen' braucht, um ???.

`public` **int Distance2Pixel**( )

Brechnet für einen Meterwert die entsprechende Anzahl an Pixel.

– **Usage**

* Die Karte muss rechteckig und genordet sein, ansonsten liefert diese Funktion keine korrekten Werte.

Parameter

* `distance` - Die Anzahl der Meter, entweder horizontal oder vertikal, nicht schräg.
* `lowerRight` - Die untere rechte GPS-Position der Karte
* `upperLeft` - Die obere linke GPS-Position der Karte
* `width` - Die Weite der Karte in Pixel
* `height` - Die Höhe der Karte in Pixel

`public` **double DistanceInMeter**( )

Berechnet näherungsweise die Distanz in Metern zwischen zwei GPS-Positionen.

Parameter

* `one` - Die erste Position.
* `two` - Die zweite Position.

`public` **double MissionObjectiveInteractionRadius**( )

Liefert den Interaktionsradius von Missionszielen

`public` **int NeutralizeDowncoolingTime**( )

Liefert die Zeit in ms, die 'Neutralisieren' braucht, um nach der letzten Benutzung wieder verfügbar zu sein.

`public` **double NeutralizeRadius**( )

Berechnet den Neutralisationsradius eines Spielers anhand seines Werts der Fähigkeit Neutralisieren in Metern.

Parameter

* `neutralizeLevel` - Wert der Fähigkeit Neutralisieren

`public` **int NeutralizeRechargeTime**( )

Liefert die Zeit in ms, die 'Tarnen' braucht, um ???.

`public` **double PlayerVisionRadius**( )

Berechnet den Sichtradius eines Spielers anhand seines Werts der Fähigkeit Aufklären

Parameter

* `reconLevel` - Wert der Fähigkeit Aufklären

`public` **double StartLocationActivationRadius**( )

Liefert den Aktivierungsradius von Startpositionen

`public` **bool SuccessfullyWiretapped**( )

Berechnet, ob das Abhören bei einem bestimmten Abhörlevel erfolgreich verlaufen ist.

Parameter

* `wiretapLevel` - Der Abhörlevel.

`public` **int WiretapDowncoolingTime**( )

Liefert die Zeit in ms, die 'Abhören' braucht, um nach der letzten Benutzung wieder verfügbar zu sein.

`public` **int WiretapDuration**( )

Liefert die maximale Abhördauer für einen Abhörlevel.

Parameter

* `wiretapLevel` - Die Punkte auf Abhören.

`public` **int WiretapRechargeTime**( )

Liefert die Zeit in ms, die 'Tarnen' braucht, um ???.

F.15 Namespace Common.Logic.Messagemanager

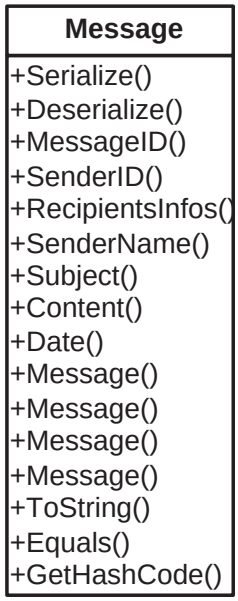


Abbildung F.15: UML Struktur des Namensraum Common.Logic.Messagemanager

F.15.1 Klasse Message

Die Klasse Message enthält alle relevanten Informationen die für den Nachrichtenaustausch zwischen Clients oder für das Versenden von Nachrichten vom Server zum Client benötigt werden.

Konstruktoren

`public` **Message**( )

Erzeugt neues Message-Objekt.

Parameter

* `messageID` - ID der Nachricht
* `senderID` - ClientID des Absenders
* `recipientsInfos` - Liste der Empfänger
* `senderName` - Name des Absenders
* `subject` - Betreff der Nachricht
* `content` - Nachrichtentext

`public` **Message**( )

Erzeugt neues Message-Objekt.

Parameter

* `messageID` - ID der Nachricht
* `systemMessage` - Text der Systemnachricht

`public` **Message**( )

Erzeugt neues Message-Objekt.

`public` **Message**( )

Dupliziert ein Message-objekt.

Parameter

* `m` - zu duplizierende Nachricht

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleich diese Message auf Gleichheit mit dem übergebenen Objekt.

Parameter

* `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

Liefert den Hashcode der Message.

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-transienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Erweitert den Betreff einer Nachricht um ein Präfix, das den Namen des Absenders enthält.

F.16 Namespace Common.Logic.Pois

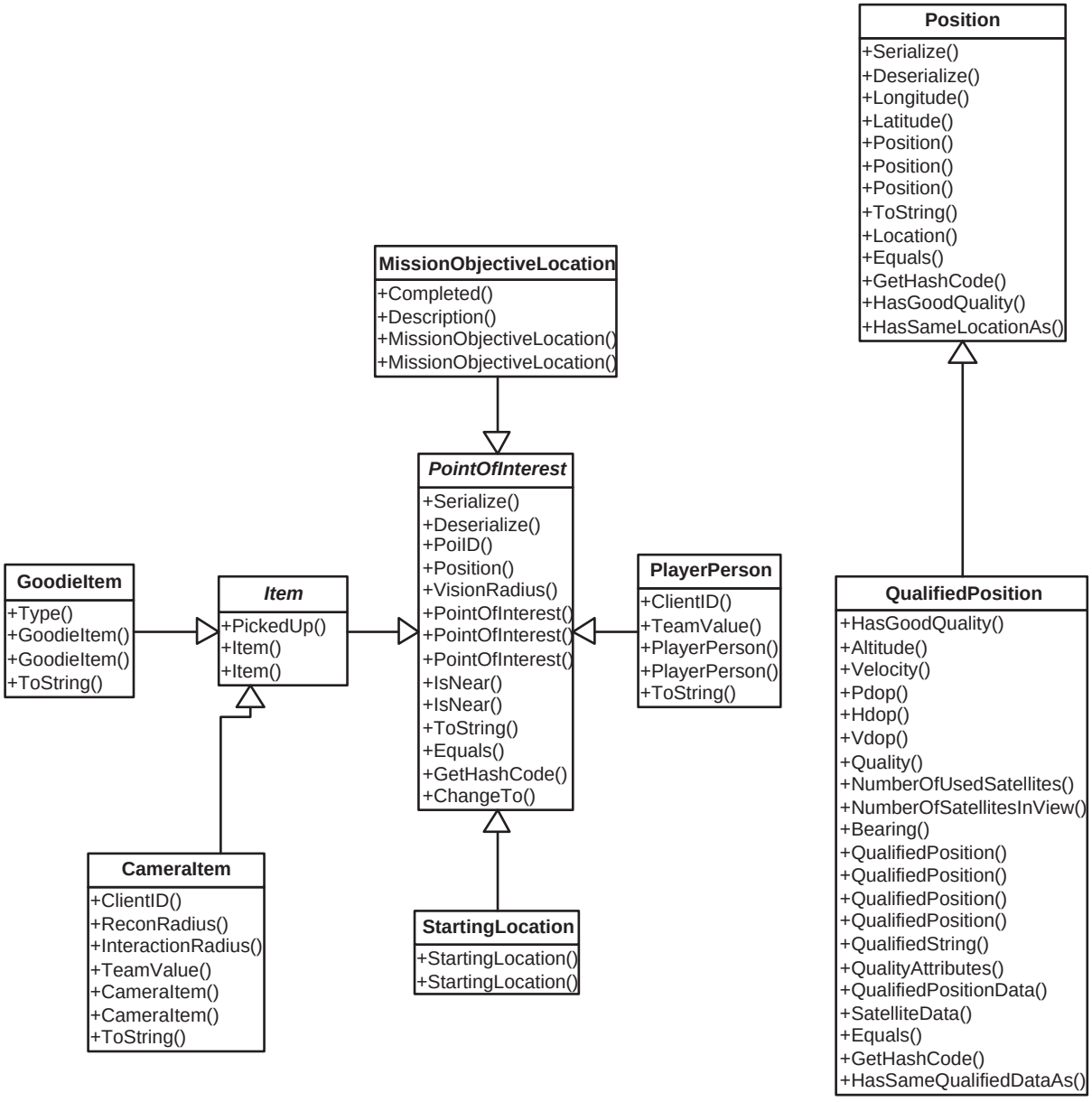


Abbildung F.16: UML Struktur des Namensraum Common.Logic.Pois

F.16.1 Klasse CameraItem

Diese Klasse repräsentiert eine Aufklärungskamera des Agentenspiels. Aufklärungskameras können von Spielern des Agentenspiels platziert und wieder eingesammelt werden. Wenn ein Spieler in die Nähe einer Kamera des gegnerischen Team gelangt, so kann er diese deaktivieren.

Konstruktoren

`public` **CameraItem**( )

Leerer Konstruktor, der benutzt wird, wenn die Felder dieses Objekts aus einem BinaryReader deserialiert werden sollen.

`public` **CameraItem**( )

Erzeugt ein neues Camera Objekt mit der übergebenen Werten

Parameter

- `poiID` - Identifier des Point-of-Interests
- `position` - Position des Point-of-Interest
- `visionRadius` - Sichtradius der Kamera
- `clientID` - ClientID des Besitzers der Kamera
- `team` - Teamzugehörigkeit der Kamera

Methoden

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

- `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert die textuelle Beschreibung dieses CameraItems

F.16.2 Klasse GoodieItem

Diese Klasse repräsentiert ein Goodie des Agentenspiels. Goodies sind virtuelle Objekte, die einen bestimmten Effekt besitzen und nur für einen bestimmten Zeitraum auf der Karte auftauchen. Begibt sich ein Spieler in die Nähe eines Goodie, so kann er dieses aufnehmen und aktivieren.

Konstruktoren

`public` **GoodieItem**( )

Leerer Konstruktor, der benutzt wird, wenn die Felder dieses Objekts aus einem BinaryReader deserialiert werden sollen.

`public` **GoodieItem**( )

Erzeugt neues Goodie

Parameter

- `poiID` - Identifier des Point-of-Interests
- `position` - Position des Point-of-Interest
- `visionRadius` - Sichtradius des Point-of-Interest, falls das POI keinen Sichtradius besitzt, soll der Wert 0 übergeben werden
- `type` - Typ des Goodies

Methoden

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

- `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert textuelle Beschreibung des Goodies, die zu seiner Darstellung im MenuPanel verwendet wird.

F.16.3 Klasse Item

Diese Klasse repräsentiert ein VirtualItem des Agentenspiels. Ein VirtualItem ist ein spezialisierter POI, der einen Goodie oder eine Kamera repräsentiert.

Konstruktoren

`public` **Item**( )

Leerer Konstruktor, der benutzt wird, wenn die Felder dieses Objekts aus einem BinaryReader deserialiert werden sollen.

`public` **Item**( )

Erzeugt ein neues Item. Initial ist das Item nicht aufgesammelt.

Parameter

- `poiID` - ID des Point-of-Interest
- `position` - Position des Point-of-Interest
- `visionRadius` - Sichtradius des Point-of-Interest

Methoden

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.16.4 Klasse MissionObjectiveLocation

Diese Klasse repräsentiert ein Missionsziel des Agentenspiels. Missionsziele können genau eine oder keine Alarmanlage haben, von der sie gesichert werden.

Konstruktoren

`public` **MissionObjectiveLocation**( )

Erzeugt neues MissionObjective Objekt.

`public` **MissionObjectiveLocation**( )

Erzeugt neue Instanz MissionObjective.

Parameter

* `poiID` - ID des Point-of-Interest
* `position` - Position des Point-of-Interest
* `visionRadius` - Sichtradius des Point-of-Interest. Soll auf dem Client kein Sichtradius angezeigt werden, muss 0 übergeben werden
* `description` - Textuelle Beschreibung des Missionsziels

Methoden

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.16.5 Klasse PlayerPerson

Dieses Klasse repräsentiert einen Spieler des Agentenspiels. Zu jedem Client, der mit dem Server verbunden ist, gibt es ein Spielerobjekt.

Konstruktoren

`public` **PlayerPerson**( )

erzeugt neues PlayerPerson Objekt.

`public` **PlayerPerson**( )

Erzeugt neues Point-of-Interest, das einen Spieler repräsentiert

Parameter

* `poiID` - ID des Point-of-Interests
* `position` - Position des Spielers
* `clientID` - ID des Clients, zu dem der Spieler gehört
* `visionRadius` - Sichtradius des Spielers
* `teamValue` - Team, zu dem der Spieler gehört

Methoden

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert textuelle Beschreibung des PointOfInterests.

F.16.6 Klasse PointOfInterest

Diese Klasse dient als Oberklasse aller für den Client sichtbaren POIs. Jeder POI wird durch eine eindeutige, vom Server vergebene ID identifiziert. Dazu kennt jeder POI seine Position und seinen Annäherungsradius. Ist ein anderer POI weniger weit von diesem POI entfernt als der Annäherungsradius, gilt der andere POI als 'sich in der Nähe befindlich'.

Konstruktoren

`public` **PointOfInterest**( )

Leerer Konstruktor, der benutzt wird, wenn die Felder dieses Objekts aus einem BinaryReader deserialiert werden sollen.

`public` **PointOfInterest**( )

Erzeugt neues Point-of-Interest ohne Sichtradius.

Parameter

- `poiID` - Identifier des Point-of-Interests
- `position` - Position des Point-of-Interest

`public` **PointOfInterest**( )

Erzeugt neues Point-of-Interest mit Sichtradius.

Parameter

- `poiID` - Identifier des Point-of-Interests
- `position` - Position des Point-of-Interest
- `visionRadius` - Sichtradius des Point-of-Interest, falls das POI keinen Sichtradius besitzt, soll der Wert 0 übergeben werden

Methoden

`public` **void ChangeTo**( )

Überträgt den Zustand des angegebenen POIs auf diesen POI

Parameter

- `poi` - POI, dessen Zustand übernommen werden soll

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleich dieses PointOfInterest auf Gleichheit mit dem übergebenen Objekt.

Parameter

* `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

Liefert den Hashcode des PointOfInterests

`public` **bool IsNear**( )

Überprüft, ob sich das übergebene Point-of-Interest im Sichtradius dieses Point-of-Interest befindet

Parameter

* `poi` - zu überprüfendes Point-of-Interest

`public` **bool IsNear**( )

Übberprüft, ob sich eine Position im Sichtradius dieses Point-of-Interest befindet

Parameter

* `position` - zu überprüfende Position

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert textuelle Beschreibung des PointOfInterests.

F.16.7 Klasse Position

Diese Klasse repräsentiert die Positionsdaten eines Objektes im Spiel. Die Position wird in Längengrad und Breitengrad in GPS-Minuten gespeichert.

Konstruktoren

`public` **Position**( )

Erzeugt ein neues Position Objekt

`public` **Position**( )

Erzeugt neues Position Objekt.

Parameter

* `longitude` - Längengrad in GPS-Minuten
* `latitude` - Breitengrad in GPS-Minuten

`public` **Position**( )

Erzeugt eine neue Position aus der gegeben Position

Parameter

* `pos` - zu klonende Position

Methoden

`public` **void Deserialize**( )

deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleich diese Position auf Gleichheit mit dem übergebenen Objekt.

Parameter

* `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

Generiert den Hashcode zu dieser Position.

`public` **bool HasGoodQuality**( )

Zeigt an, ob die Positionsdaten von ausreichender Qualität sind, um vom System weiterverarbeitet zu werden. Diese Methode kann von spezialisierten Positionsobjekten überschrieben werden. Diese Methode dieser Klasse liefert immer den Wert TRUE.

`public` **bool HasSameLocationAs**( )

Überprüft, ob Längengrad, Breitengrad und Höhe dieser Position mit der übergeben übereinstimmt.

Parameter

* `p` - zu vergleichende Position

`public` **string Location**( )

Liefert die aktuelle Position als String.

`public` **void Serialize**( )

serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert die Stringrepäsentation dieser Position.

F.16.8 Klasse QualifiedPosition

Diese Klasse erweitert die Klasse Position. Die Positionsdaten werden zuzüglich zu Längengrad und Breitengrad in GPS-Minuten um Informationen zur Höhe, Geschwindigkeit, Genauigkeit und Qualität der Positionsdaten, sowie Anzahl der in Sichtweite befindlichen Satelliten und die Bewegungsrichtung erweitert. Alle erweiterten Informationen werden von der GPS-Maus geliefert.

Konstruktoren

`public` **QualifiedPosition**( )

Erzeugt ein leeres QualifiedPosition-Objekt

`public` **QualifiedPosition**( )

Erzeugt ein QualifiedPosition-Objekt mit den als Parametern übergebenen Werten.

Parameter

- `longitude` - Der Längengrad
- `latitude` - Der Breitengrad

`public` **QualifiedPosition**( )

Erzeugt ein QualifiedPosition-Objekt mit den als Parametern übergebenen Werten.

Parameter

- `longitude` - Der Längengrad
- `latitude` - Der Breitengrad
- `altitude` - Die Höhe

`public` **QualifiedPosition**( )

Erzeugt eine neue QualifiedPosition aus der gegeben QualifiedPosition

Parameter

- `pos` - zu klonende Position

Methoden

`public` **bool Equals**( )

Vergleich diese QualifiedPosition auf Gleichheit mit dem übergebenen Objekt.

Parameter

- `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **bool HasGoodQuality**( )

Zeigt an, ob die Positionsdaten eine genügend hohe Qualität haben, um von dem Programm weiterverarbeitet zu werden. Die Berechnung beruht auf den gelieferten GPS-Daten über die Genauigkeit (pdop, hdop, vdop) und die Qualität (quality) der gesendeten Daten

`public` **bool HasSameQualifiedDataAs**( )

Überprüft, ob die qualifizierten Positionsdaten dieser Position denen der übergebenen entsprechen.

Parameter

* `p` - Position mit den zu vergleichenden qualifizierten Positionsdaten

`public` **string QualifiedPositionData**( )

Liefert die Stringrepräsentation von QualifiedPosition

`public` **string QualifiedString**( )

Liefert die Stringrepräsentation aller Daten dieser Position.

`public` **string QualityAttributes**( )

Liefert die Stringrepräsentation der Positionsqualität

`public` **string SatelliteData**( )

Liefert Anzahl der sichtbaren und zur Positionsbestimmung verwendeten Satelliten als String.

F.16.9 Klasse StartingLocation

Diese Klasse repräsentiert den Startpunkt eines Spielers des Agentenspiels. Wenn ein Spieler zu Beginn den Radius seines Startpunktes betritt, so werden seine Fähigkeiten aktiviert. Jeder Spieler hat einen eindeutigen Startpunkt und die Positionen der Startpunkte von verschiedenen Spielern sind unterschiedlich.

Konstruktoren

`public` **StartingLocation**( )

Leerer Konstruktor, der benutzt wird, wenn die Felder dieses Objekts aus einem BinaryReader deserialiert werden sollen.

`public` **StartingLocation**( )

Erzeugt ein neue StartingLocation

Parameter

* `poiID` - Identifier des Point-of-Interests
* `position` - Position der StartingLocation
* `visionRadius` - Sichtradius der StartingLocation

F.17 Namespace Common.Logic.Scenariodates

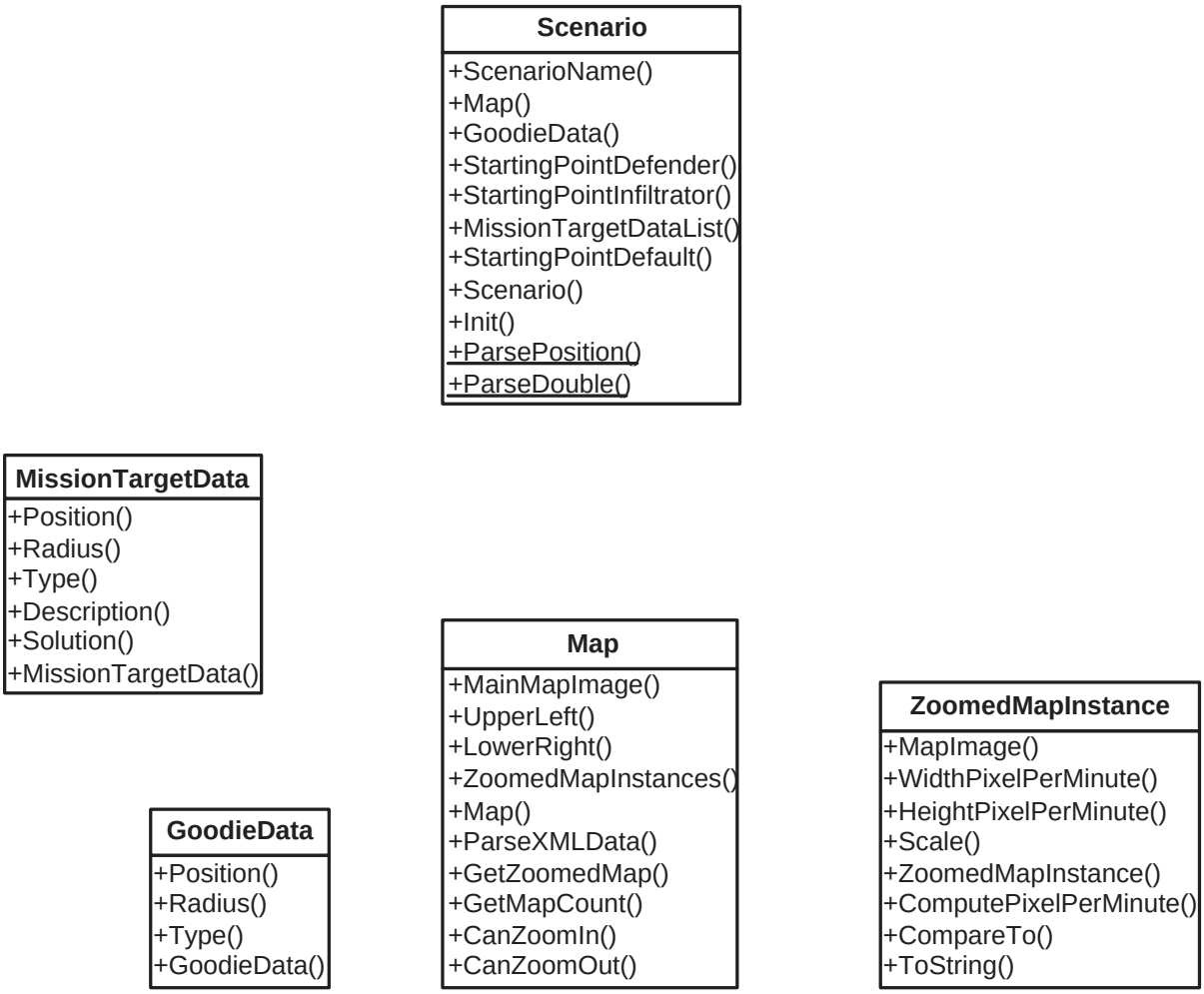


Abbildung F.17: UML Struktur des Namensraum Common.Logic.Scenariodates

F.17.1 Klasse GoodieData

Diese Klasse beschreibt die Daten eines Goodies aus einem Scenario.

Konstruktoren

`public` **GoodieData**( )

Erzeugt neues GoodieData Objekt.

Parameter

* `position` - die Position dieses Goodies
* `type` - der Typ dieses Goodies
* `radius` - der Radius dieses Goodies

F.17.2 Klasse Map

Das Map-Objekt enthält die Daten zu einer Karte und deren Zoomstufen.

Konstruktoren

`public` **Map**( )

Erzeugt neues Map Objekt aus einem XmlElement.

Parameter

* `mapEl` - das Element mit den Kartendaten
* `parentPath` - der Pfad der Karten-Datei

Methoden

`public` **bool CanZoomIn**( )

Zeigt an, ob noch eine weitere detailierte Zoomstufe der Karte existiert.

Parameter

* `index` - Index der aktuellen Zoomstufe

`public` **bool CanZoomOut**( )

Zeigt an, ob noch eine weitere gröbere Zoomstufe der Karte existiert.

Parameter

* `index` - Index der aktuellen Zoomstufe

`public` **int GetMapCount**( )

Liefert die Anzahl der Zoomstufen, die für diese Karte zur Verfügung stehen.

`public` **Common.Logic.Scenariodates.ZoomedMapInstance GetZoomedMap**( )

Liefert die i-te Karte aus der Liste verschieden gezoomter Karten.

Parameter

* `index` - Index der Karte

`public` **void ParseXMLData**( )

Diese Methode parst die einzelnen Elemente des mmapElements der XML-Datei und schreibt die Werte in die zugehörigen Attribute.

Parameter

* `mapEl` - Das Element mit den Kartendaten
* `parentPath` - Pfad, in dem die Bilder liegen

F.17.3 Klasse MissionTargetData

Diese Klasse beschreibt die Daten eines Missionsziels aus einem Scenario.

Konstruktoren

`public` **MissionTargetData**( )

Erzeugt neues MissionTargetData Objekt.

Parameter

* `position` - die Position dieses Missionsziels
* `type` - der Typ dieses Missionsziels
* `radius` - der Radius dieses Missionsziels
* `description` - die Beschreibung dieses Missionsziels

F.17.4 Klasse Scenario

Diese Klasse beschreibt die Bereitstellung der Szenariodaten.

Konstruktoren

`public` **Scenario**( )

Erzeugt neues Scenario Objekt.

Methoden

`public` **void Init**( )

Startet die Initialisierung aus einer Scenario XML-Datei.

Parameter

* `filePathString` - der Pfad der Scenario XML-Datei

`public` **double ParseDouble**( )

Wandelt einen double-Wert, der als string vorliegt, in einen double-Datentyp. Format ist ***,***.** (international).

– **Usage**
  * Für deutsches Format einfach FRACTION_SEPARATOR und THOUSAND_SEPARATOR vertauschen, Format ist dann ***.***,** (deutsch).

Parameter

* `doubleText` - Der double-Wert als string

`public` **Common.Logic.Pois.Position ParsePosition**( )

Erzeugt ein Position-Objekt aus einem Position-XmlElement.

Parameter

* `positionElement` - das Position-XmlElement

F.17.5 Klasse ZoomedMapInstance

Diese Klasse speichert alle Daten zu einer Zoomstufe der Karte.

Konstruktoren

`public` **ZoomedMapInstance**( )

Ezeugt eine neue ZoomedMapInstance.

Parameter

* `mapImage` - das Bild dieser Zoomstufe
* `scale` - die Skalierung dieser Zoomstufe

Methoden

`public` **int CompareTo**( )

Vergleicht die aktuelle Instanz mit einem anderen Objekt desselben Typs.

Parameter

* `obj` - ein Objekt, das mit dieser Instanz verglichen werden soll

`public` **string ToString**( )

Liefert die Bezeichnung dieser Karteninstanz.

F.18 Namespace Common.Logic.Statemanager

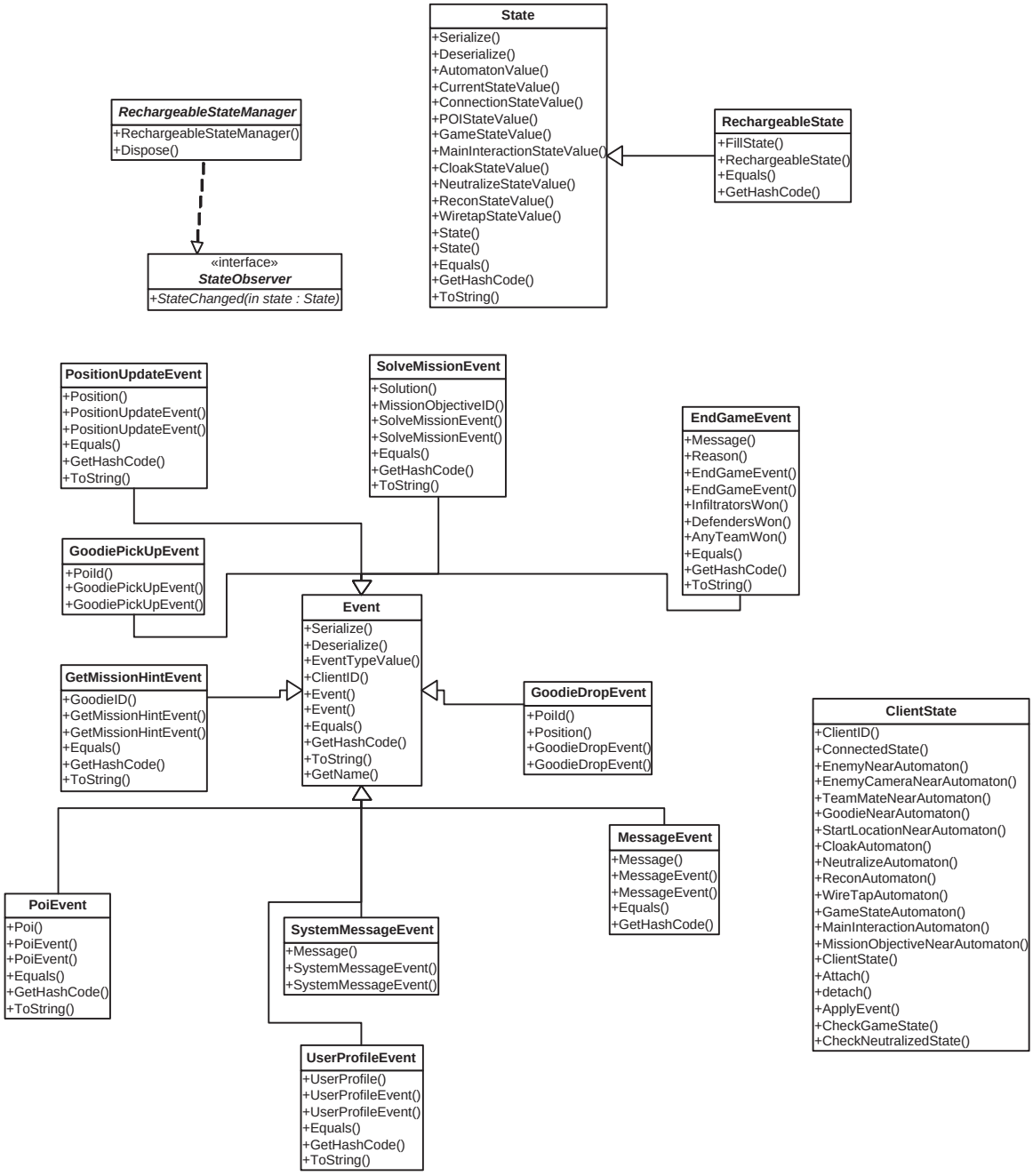


Abbildung F.18: UML Struktur des Namensraum Common.Logic.Statemanager

F.18.1 Schnittstelle StateObserver

Diese Schnittstelle wird von allen Klassen implementiert, die über Zustandsänderungen informiert werden wollen. Dazu registrieren sich die Klassen beim StateManager als Observer. Für jeden Zustandsautomaten stellt das Interface eine eigene Update-Methode bereit. Diese wird immer dann aufgerufen, wenn sich der Zustand im assoziierten Automaten geändert hat.

Methoden

`public` **void StateChanged**( )

Wird aufgerufen, wenn sich ein Zustand geändert hat. Der Zustand kann über seine StateID einem bestimmten Automaten zugeordnet werden und gibt mittels CurrentStateID den Identifiert seines aktuellen Zustands preis.

Parameter

* `state` - Zustand, der sich geändert hat.

F.18.2 Klasse ClientState

Das Modul ClientState hält für jeden Client die generischen Programmzustände vor. Für jeden Zustandsautomat verfügt der ClientState über eine eigene Zustandsklasse.

Konstruktoren

`public` **ClientState**( )

Erzeugt neues ClientState Objekt und setzt alle Zustandsautomaten initial in dem Startzustand.

Methoden

`public` **void ApplyEvent**( )

Wendet ein Ereignis auf die Zustandsautomaten an.

Parameter

* `e` - Ereigniss

`public` **void Attach**( )

Fügt dem ClientState einen Observer hinzu.

Parameter

* `stateObserver` - der Observer

`public` **void CheckGameState**( )

Überprüft, ob das Event den aktuellen Zustand des SpielzustandAutomaten verändert

Parameter

* `e` - Ereignis

`public` **void CheckNeutralizedState**( )

Überprüft, ob das Event den aktuellen Zustand des Neutralisierungsautomatena verändert.

Parameter

* `e` - Das Ereignis

`public` **void detach**( )

Entfernt einen Observer.

Parameter

* `stateObserver` - der zu entfernende Observer

F.18.3 Klasse EndGameEvent

Dieses Event wird vom Server an den Client geschickt, wenn das Spiel beendet wird.

Konstruktoren

`public` **EndGameEvent**( )

Erzeugt ein leeres EndGameEvent.

`public` **EndGameEvent**( )

Erzeugt ein neues EndGameEvent für den angegebenen Spieler und mit der angegebenen Nachricht.

Parameter

- `clientID` - Id des Spielers
- `message` - Nachricht für den Spieler
- `reason` - Grund für das Spielende

Methoden

`public` **bool AnyTeamWon**( )

Überprüft anhand der Begründung des Spielendes, ob eines der beiden Teams gewonnen hat.

`public` **bool DefendersWon**( )

Überprüft anhand der Begründung des Spielendes, ob die Defender gewonnen haben.

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht ein Event mit den übergebenen Objekt.

Parameter

- `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **bool InfiltratorsWon**( )

Überprüft anhand der Begründung des Spielendes, ob die Infiltratoren gewonnen haben.

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

- `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

F.18.4 Klasse Event

Die Klasse Event kapselt ein im Server aufgetretenes Ereignis. Events implementieren das Interface Serializable, können also über den Communication-Layer zwischen Server und Client ausgetauscht werden. Die Automaten prüfen ihre Übergangsbedingungen, und wechseln ggf. Aufgrund des Ereignisses in einen anderen Zustand.

Konstruktoren

`public` **Event**( )

Leerer Konstruktor, der vor dem Deserialisieren benutzt werden soll

`public` **Event**( )

Erzeugt neues Ereignis

Parameter

- `eventType` - Identifikation des Ereignis
- `clientID` - Client, auf dem das Ereignis aufgetreten ist

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht das Event mit dem angegebenen Objekt anhand seiner ID.

Parameter

- `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **string GetName**( )

Liefert die textuelle Beschreibung des Events

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

- `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert die Stringrepräsentation zu diesem GameObject.

F.18.5 Klasse GetMissionHintEvent

Dieses Event wird vom Client an den Server geschickt, wenn der Spieler einen Goodie mit einem Hinweis auf ein Missionsziel aktiviert.

Konstruktoren

`public` **GetMissionHintEvent**( )

Erzeugt ein leeres GetMissionHintEvent.

`public` **GetMissionHintEvent**( )

Erzeugt ein neues GetMissionHintEvent vom angegebenen Spieler und mit der angegebenen Id eines Goodies.

Parameter

- `clientID` - Id des Spielers
- `goodieID` - Id des Goodies

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

- `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht ein Event mit den übergebenen Objekt.

Parameter

- `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

- `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

F.18.6 Klasse GoodieDropEvent

Mit diesem Event wird der Server benachrichtigt, dass an einer bestimmten Position ein Goodie von einem Spieler fallen gelassen wurde.

Konstruktoren

`public` **GoodieDropEvent**( )

Erzeugt ein neues GoodieDropEvent für den Spieler mit der angegebenen clientID und dem Goodie der angegebenen poi Id. Das Goodie wird an der angegebenen Position abgelegt.

Parameter

* `clientID` - Id des Spielers
* `poiId` - poi Id des Goodies
* `position` - neue Position des Goodies

`public` **GoodieDropEvent**( )

Erzeugt ein neues GoodieDropEvent.

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.18.7 Klasse GoodiePickUpEvent

Dieses Event wird vom Client verschickt, wenn der Spieler ein Goodie aufgenommen hat.

Konstruktoren

`public` **GoodiePickUpEvent**( )

Erzeugt ein neues GoodiePickUpEvent für den Spieler mit der angegebenen clientID und dem Goodie der angegebenen poi Id.

Parameter

* `clientID` - ID des Clients
* `poiId` - poiId des Goodies

`public` **GoodiePickUpEvent**( )

Erzeugt ein neues GoodiePickUpEvent.

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.18.8 Klasse MessageEvent

Dieses Event wird verschickt wenn eine Nachricht versendet werden soll.

Konstruktoren

`public` **MessageEvent**( )

Erzeugt ein neues MessageEvent

Parameter

* `clientID` - ID des Versenders
* `message` - die zu versendende Nachricht

`public` **MessageEvent**( )

Erzeugt eine neues MessageEvent

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht dieses MessageEvent mit dem angegebenen Objekt anhand seiner ID.

Parameter

* `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.18.9 Klasse PoiEvent

Dieses Event wird den Clients geschickt wenn sich ein Poi verändert hat.

Konstruktoren

`public` **PoiEvent**( )

Erzeugt ein leeres PositionUpdateEvent

`public` **PoiEvent**( )

Erzeugt neues PositionUpdateEvent

Parameter

* `eventType` - Typ des Events
* `clientID` - ID des Clients
* `poi` - zu versendende Poi

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht das POIEvent mit dem übergebenen Objekt

Parameter

* `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

F.18.10 Klasse PositionUpdateEvent

Dieses Event wird verschickt, wenn sich die Position eines Clients verändert hat.

Konstruktoren

`public` **PositionUpdateEvent**( )

Erzeugt neues PositionUpdateEvent.

`public` **PositionUpdateEvent**( )

Erzeugt neues PositionUpdateEvent.

Parameter

* `clientID` - ID des Clients
* `position` - neue Position des Clients

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht dieses PositionUpdateEvent mit dem angegebenen Objekt anhand seiner Position. zu vergleichendes Objekt

Parameter

* `obj` -

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

F.18.11 Klasse RechargeableState

Ein State mit einem Füllstand für die Progressbars.

Konstruktoren

`public` **RechargeableState**( )

Erzeugt neuen Zustandsautomaten, der sich im Startzustand befindet.

Parameter

* `automaton` - Identifiziert den Automaten, den dieses Objekt repräsentiert.
* `initialState` - Der Startzustand des Automaten.

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht einen RechargeableState mit einem anderen Objekt.

Parameter

* `obj` - das zu vergleichende Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.18.12 Klasse RechargeableStateManager

Basisklasse für Behandlung einer Veränderung eines RechargeableState auf Server- und Clientseite. Zusätzlich zu der Schnittstellendefinition wird ein Timer verwaltet.

Konstruktoren

`public` **RechargeableStateManager**( )

Konstruiert einen neuen RechargeableStateManager.

Parameter

* `timerSpan` - Die Dauer zwischen den Timerauslösern
* `times` - Die Anzahl der Aufrufe
* `obj` - Das an OnTimerEmitted übergebene Objekt.

Methoden

`public` **void Dispose**( )

Stoppt, wenn er noch laufen sollte, den Timer.

`public` **void StateChanged**( )

Implementierung vom StateObserver.

Parameter

* `state` - Der sich geänderte Zustand

F.18.13 Klasse SolveMissionEvent

Dieses Event wird vom Client an den Server geschickt, wenn der Spieler ein Missionsziel lösen möchte.

Konstruktoren

`public` **SolveMissionEvent**( )

Erzeugt ein neues SolveMissionEvent.

`public` **SolveMissionEvent**( )

Erzeugt ein neues SolveMissionEvent.

Parameter

* `clientID` - ID des Clients
* `missionObjectiveID` - ID des Missionsziels
* `solution` - Lösung des Missionsziels

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht ein SolveMissionEvent mit einem anderen Objekt.

Parameter

* `obj` - das zu vergleichende Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

F.18.14 Klasse State

State ist die abstrakte Oberklasse aller Automaten. Die Klasse kapselt alle möglichen Zustände in der sich das Spiel befinden kann, dabei wird jeder Zustand durch einen Automaten repräsentiert. Jeder Automat wird durch seinen Namen identifiziert. Ausserdem besitzt jeder Automat eine Enumeration, die den augenblicklichen Zustand repräsentiert.

Konstruktoren

`public` **State**( )

Erzeugt undefinierten Zustand. Wird benutzt, wenn der Zustand aus einem Stream deserialisiert werden soll

`public` **State**( )

Erzeugt neuen Zustandsautomaten, der sich in dem angegebenen Zustand befindet.

Parameter

* `automaton` - Identifiziert den Automaten, den dieses Objekt repräsentiert.
* `initialState` - Identifiziert den aktuellen Zustand des repräsentierten Automaten.

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht einen State mit dem angegebenen Objekt anhand seiner Automaten.

Parameter

* `obj` - zu vergleichendes Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert die textuelle Beschreibung des States

F.18.15 Klasse SystemMessageEvent

Dieses Event wird verschickt wenn eine Systemnachricht versendet werden soll.

Konstruktoren

`public` **SystemMessageEvent**( )

Erzeugt ein leeres SystemMessageEvent.

`public` **SystemMessageEvent**( )

Erzeugt neues SystemMessageEvent.

Parameter

* `clientID` - Id des Spielers für den die Nachricht bestimmt ist
* `message` - Nachrichtinhalt

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

F.18.16 Klasse UserProfileEvent

Dieses Event wird verschickt wenn eine UserProfile versendet werden soll.

Konstruktoren

`public` **UserProfileEvent**( )

Erzeugt neues UserProfileEvent Objekt.

`public` **UserProfileEvent**( )

Erzeugt neues UserProfileEvent Objekt.

Parameter

* `eventType` - Typ des Events
* `clientID` - ID des Versenders
* `userProfile` - das zu versendende UserProfile

Methoden

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **bool Equals**( )

Vergleicht ein UserProfileEvent mit einem anderen Objekt.

Parameter

* `obj` - das zu vergleichende Objekt

`public` **int GetHashCode**( )

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

F.19 Namespace Common.Logic.Userinterface.Gui

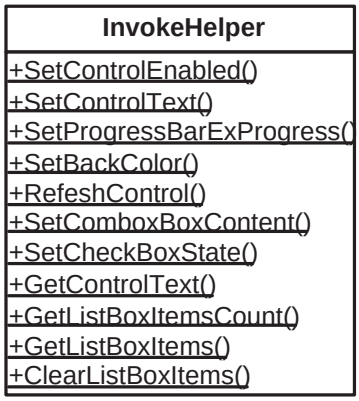


Abbildung F.19: UML Struktur des Namensraum Common.Logic.Userinterface.Gui

F.19.1 Klasse InvokeHelper

Die Klasse InvokeHelper stellt statische Hilfsmethoden zur threadsicheren Veränderung von Gui-Komponenten zur Verfügung.

Konstruktoren

`public` **InvokeHelper**( )

Methoden

`public` **void ClearListBoxItems**( )

Löscht threadsicher die Items einer ListBox

Parameter

* `box` - die Listbox

`public` **string GetControlText**( )

Holt threadsicher den Text eines Controls.

Parameter

* `control` - das Control

`public` **System.Collections.ArrayList GetListBoxItems**( )

Holt threadsicher die Items einer ListBox.

Parameter

* `box` - die Listbox

`public` **int GetListBoxItemsCount**( )

Holt threadsicher die Anzahl der Items einer ListBox.

Parameter

- `box` - die Listbox

`public` **void RefeshControl**( )

Ruft threadsicher Refresh() auf dem Control auf.

Parameter

- `control` - das neu zu zeichnende Control

`public` **void SetBackColor**( )

Setzt threadsicher die Hintergrundfarbe eines Controls.

Parameter

- `control` - Bei diesem Control wird die Hintergrundfarbe gesetzt.
- `color` - die neue Hintergrundfarbe

`public` **void SetCheckBoxState**( )

Ändert threadsicher den Zustand einer CheckBox.

Parameter

- `control` - zu ändernde CheckBox
- `check` - true, wenn sie angewählt sein soll, sonst false

`public` **void SetComboxBoxContent**( )

Füllt threadsicher eine ComboBox. Sie wird erst komplett entleert, dann neu gefüllt und am Schluss das aktuelle Item gesetzt.

Parameter

- `control` - ComboBox, die neu gefüllt werden soll
- `content` - ArrayList mit den Items
- `selectedItem` - momentan ausgewähle Item

`public` **void SetControlEnabled**( )

Setzt threadsicher den Enabled-Status eines Controls.

Parameter

- `control` - Das zu aktivierende bzw. zu deaktivierende Control.
- `enabled` - Der boolsche Werte, true für aktiviert, false für deaktiviert.

`public` **void SetControlText**( )

Setzt threadsicher den Text eines Controls, bspw. die Beschriftung eines Button.

Parameter

- `control` - Bei diesem Control wird der Text gesetzt.
- `text` - der Text

`public` **void SetProgressBarExProgress**( )

Setzt threadsicher den Fortschrittswert einer ProgressBarEx.

Parameter

- `control` - Diese ProgressBarEx wird verändert, also der Fortschrittsbalken gesetzt.
- `progress` - Der Fortschrittswert, liegt zwischen 0.0 und 1.0.

F.20 Namespace Common.Logic.Userprofile

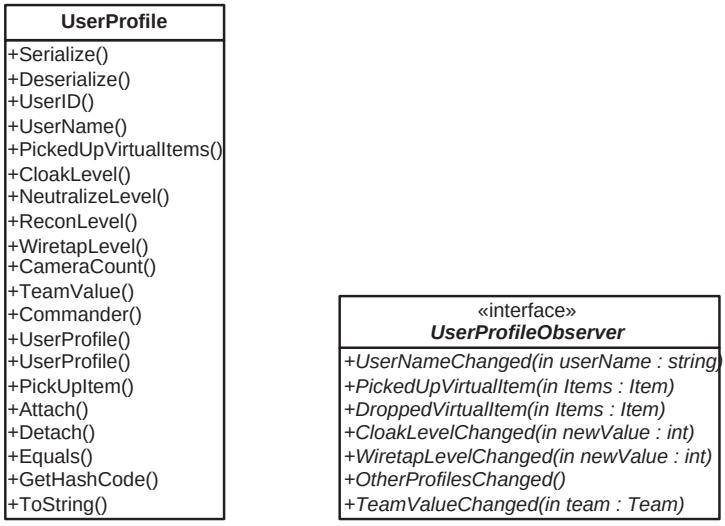


Abbildung F.20: UML Struktur des Namensraum Common.Logic.Userprofile

F.20.1 Schnittstelle UserProfileObserver

Ein Modul, das auf Änderungen des Benutzernamens oder der aufgenommenen virtuellen Objekte lauschen will, implementiert diese Schnittstelle und registriert sich beim UserProfile. Wenn sich der Benutzername oder die aufgenommenen virtuellen Objekte ändern, so wird das registrierte Modul über diese Änderungen benachrichtigt.

Methoden

`public` **void CloakLevelChanged**( )

Wird aufgerufen, wenn das CloakLevel neu gesetzt wurde.

Parameter

- `newValue` - Der neues Fähigkeitenwert.

`public` **void DroppedVirtualItem**( )

Wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand abgelegt hat.

Parameter

- `Items` - virtueller Gegenständ, den der Nutzer abgelegt hat.

`public` **void OtherProfilesChanged**( )

Wird aufgerufen, wenn sich die Liste der Profile anderer Spieler ändert.

`public` **void PickedUpVirtualItem**( )

Wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand aufgenommen hat.

Parameter

- `Items` - virtueller Gegenständ, den der Nutzer aufgenommen hat.

`public` **void TeamValueChanged**( )

Wird aufgerufen, wenn sich die Teamzugehörigkeit eines Spielers geändert hat.

Parameter

* `team` - Das neue Team des Spielers

`public` **void UserNameChanged**( )

Wird aufgerufen, wenn sich das Pseudonym der Nutzers geändert hat

Parameter

* `userName` - neues Pseudonym

`public` **void WiretapLevelChanged**( )

Wird aufgerufen, wenn das WiretapLevel neu gesetzt wurde.

Parameter

* `newValue` - Der neues Fähigkeitenwert.

F.20.2 Klasse UserProfile

Das UserProfile repräsentiert das für das Spiel spezifisches Nutzerprofil. Im UserProfile werden neben Name und Teamzugehörigkeit auch die einzelnen Fähigkeitslevel und alle aufgesammelten virtuellen Gegenstände gespeichert.

Konstruktoren

`public` **UserProfile**( )

Erzeugt neues UserProfile Objekt.

`public` **UserProfile**( )

Erzeugt neues UserProfile.

Parameter

* `name` - Name des Spielers
* `cloakLevel` - Repräsentiert das Level der Tarnfähigkeit. Die Variable wird mit dem Wert '1' initialisiert und nimmt nur Werte zwischen '1' und '5' an. Siehe dazu auch die Konstanten LEVEL_XYZ. Pro Fähigkeitspunkt steigt die Aufladegeschwindigkeit der Fähigkeit.
* `neutralizeLevel` - Repräsentiert das Level der Neutralisationsfähigkeit. Die Variable wird mit dem Wert '1' initialisiert und nimmt nur Werte zwischen '1' und '5' an. Siehe dazu auch die Konstanten LEVEL_XYZ. Pro Fähigkeitspunkt steigt der Neutralisationsradius und die Aufladegeschwindigkeit.
* `reconLevel` - Repräsentiert das Level der Aufklärungsfähigkeit. Die Variable wird mit dem Wert '1' initialisiert und nimmt nur Werte zwischen '1' und '5' an. Siehe dazu auch die Konstanten LEVEL_XYZ.
* `wireTapLevel` - Repräsentiert das Level der Abhörfähigkeit. Die Variable wird mit dem Wert '1' initialisiert und nimmt nur Werte zwischen '1' und '5' an. Siehe dazu auch die Konstanten LEVEL_XYZ. Pro Fähigkeitspunkt steigt die Wahrscheinlichkeit, dass eine gegnerische Nachricht abgefangen wird.

* `team` - Teamzugehörigkeit des Spielers. TEAM_NONE ist der Standardwert bei der Initialisierung der Klasse.
* `commander` - Zeigt an, ob der Spieler der Commander des Infiltrator-Teams ist. In diesem Fall ist der Wert der Eigentschaft true. Standardmäßig ist die Eigentschaft false.

Methoden

`public` **void Attach**( )

Fügt denm UserProfile einen UserProfileObserver hinzu.

Parameter

* `observer` - hinzuzufügender Observer

`public` **void Deserialize**( )

Deserialisiert das Objekt aus dem übergebenen BinaryReader. Dazu werden die nicht-transienten primitiven Datentypen des Objekts aus dem Reader ausgelesen. Dies muss in der gleichen Reihenfolge geschehen, in der die Methode Serialize(BinaryWriter) des Objekts die Datentypen serialisiert.

Parameter

* `binaryReader` - Reader, aus dem das Objekt deserialisiert werden soll.

`public` **void Detach**( )

Entfernt einen Observer..

Parameter

* `observer` - zu entfernender Observer

`public` **bool Equals**( )

Vergleicht das UserProfile mit einem anderen Objekt.

Parameter

* `obj` - zu vergleichende Objekt

`public` **int GetHashCode**( )

`public` **void PickUpItem**( )

Fügt dem UserProfile ein virtuelles Item hinzu.

Parameter

* `item` -

`public` **void Serialize**( )

Serialisiert das Objekt in den übergebenen BinaryWriter. Dazu werden die nicht-tranisienten primitiven Datentypen des Objekts in den Writer geschrieben. Die Methoden muss konsistent zu der Methode Deserialize(BinaryReader) des Objekts sein, da sonst das Objekt nicht korrekt übertragen werden kann.

Parameter

* `binaryWriter` - Writer, der die serialisierte Version des Objekts aufnimmt.

`public` **string ToString**( )

Liefert die textuelle Beschreibung des UserProfiles

F.21 Namespace Server.Communication

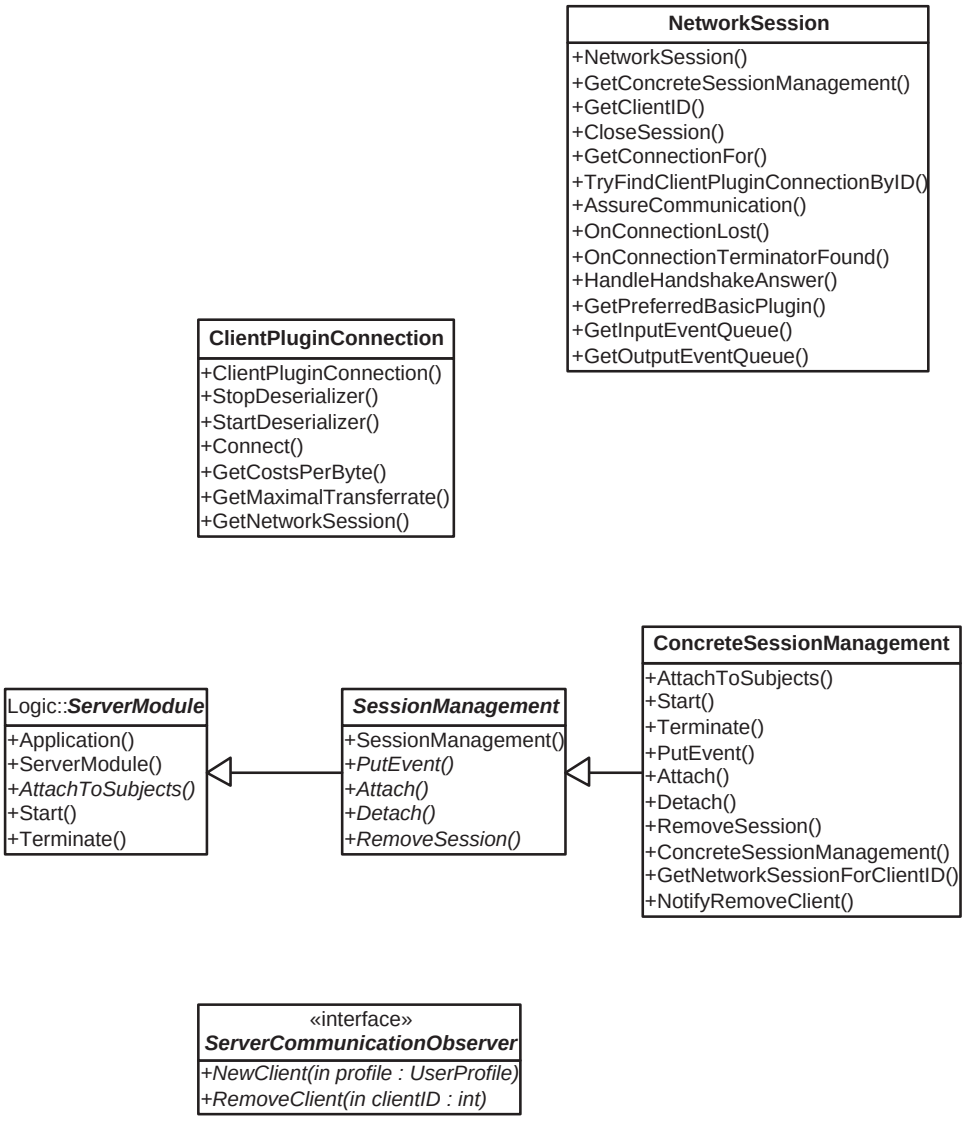


Abbildung F.21: UML Struktur des Namensraum Server.Communication

F.21.1 Schnittstelle ServerCommunicationObserver

Klassen, die den Communication-Layer auf Serverseite observieren möchten, müssen vom der ServerCommunicationObserver-Schnittstelle abgeleitet werden.

Methoden

`public` **void NewClient**( )

Diese Methode wird aufgerufen, wenn sich ein neuer Client beim Server anmeldet.

Parameter

* `profile` - Das User-Profile des Clients.

`public` **void RemoveClient**( )

Diese Methode wird aufgerufen, wenn ein Client die letzte Verbindung zum Server getrennt hat, oder eine Communication mit ihm nicht mehr möglich ist.

Parameter

* `clientID` - Identifiziert den Client, der sich vom Server abgemeldet hat.

F.21.2 Klasse ClientPluginConnection

Die Klasse ClientPluginConnection ist das Gegenstück eines Clientplugins.

Konstruktoren

`public` **ClientPluginConnection**( )

Erzeugt ein neues ClientPluginConnection-Objekt.

Parameter

* `networkSession` - Die NetworkSession, der diese ClientPluginConnection zugeordnet ist.
* `costsPerByte` - Die Kosten dieser Verbindung.
* `maximalTransferrate` - Die maximale Transferrate dieser Verbindung.
* `pluginID` - Die Plugin-ID dieser Verbindung.

Methoden

`public` **void Connect**( )

Wird vom Server aus nicht unterstützt. Nur der Client baut Verbindungen auf.

Parameter

* `ip` - die IP
* `port` - der Port
* `profile` - das Profil

`public` **double GetCostsPerByte**( )

Liefert die Kosten der Verbindung dieses Plugins.

`public` **int GetMaximalTransferrate**( )

Liefert die maximale Transferrate der Verbindung dieses Plugins.

`public` **Server.Communication.NetworkSession GetNetworkSession**( )

Liefert die NetworkSession dieses Plugins.

`public` **void StartDeserializer**( )

Startet den Deserialisierungprozess dieses Plugins.

`public` **void StopDeserializer**( )

Stoppt den Deserialisierungprozess dieses Plugins.

F.21.3 Klasse ConcreteSessionManagement

Die Klasse ConcreteSessionManagement stellt das Herzstück des Communication-Layers auf Serverseite dar. Sie erwartet eingehende Verbindungen, führt den Handshake durch und benachrichtigt alle ServerCommunication-Observer, wenn sich ein neuer Spieler angemeldet hat. Für jeden Spieler wird eine NetworkSession erstellt, die das Senden und Empfgangen von Nachrichten an bzw. von diesem Speiler übernimmt. Auch das Abmelden eines Client wird über den Observer mitgeteilt. Die IDs der Clients generiert das ConcreteSessionManagement. Über die Schittstelle des SessionManagements interagieren die anderen Klassen mit dem ConcreteSessionManagement.

Konstruktoren

`public` **ConcreteSessionManagement**( )

Erzeugt eine neues ConcreteSessionManagement-Object.

Parameter

* `server` - Die ServerApplication.
* `port` - Der zu belauschende Port.

Methoden

`public` **void Attach**( )

Registriert einen ServerCommunicationObserver.

Parameter

* `observer` - der Observer

`public` **void AttachToSubjects**( )

Wird aus Compiler-Gründen überschrieben.

`public` **void Detach**( )

Entfernt einen ServerCommunicationObserver.

Parameter

* `observer` - der Observer

`public` **Server.Communication.NetworkSession GetNetworkSessionForClientID**( )

Liefert, für den durch die clientID spezifizierten Client die zugehörige NetworkSession. Wird die Network-Session nicht gefunden, so wird eine ArgumentException geworfen.

Parameter

* `clientID` - Spezifiziert den Client.

`public` **void NotifyRemoveClient**( )

Benachrichtigt alle Server CommunicationObserver davon, das ein bestimmter Client entfernt werden soll.

Parameter

* `clientID` - Die Id des Clients, der entfernt werden soll.

`public` **void PutEvent**( )

Fügt ein zu verschickendes Event in die Queue ein.

Parameter

* `inputEvent` - Das Event, welches verschickt werden soll.

`public` **void RemoveSession**( )

Wird aufgerufen, wenn ein Client entgültig aus dem Spiel genommen werden soll, bspw. die Communication mit ihm abgebrochen wurde.

Parameter

* `clientID` - Die clientId der zu entfernenden Session.

`public` **void Start**( )

Start das Erwarten eingehenden Verbindungen der Clients.

`public` **void Terminate**( )

Gibt alle Ressourcen frei und beendet alle Verbindungen zu den Clients.

F.21.4 Klasse NetworkSession

Für jeden Client gibt es einen NetworkSession. Diese verwaltet die ClientPluginConnections, den Serializer und die EventQueue der zu versendenden Events. Die NetworkSession ist Ansprechpartner für den Serializer und die Deserialiser der ClientPluginConnections, um Rückmeldung über den Zustand einer Verbindung geben zu können, bspw. über deren Abbruch. Die Schnittstelle hierzu liefert ConnectionUtilitiesProvider.

Konstruktoren

`public` **NetworkSession**( )

Erzeugt ein neues NetworkSession-Objekt.

Parameter

* `concreteSessionManagement` - Das SessionManagement, welches diese Networksession speichert.
* `clientID` - Die ID des Clients, zu dem dieser Networksession gehört.

Methoden

`public` **void AssureCommunication**( )

Stellt sicher, das Serializer bzw. Deserializer laufen, wenn mindestens eine Verbindung besteht.

`public` **void CloseSession**( )

Schließt alle ausgehenden Verbingungen und beendet der Serialisierungprozeß für diese NetworkSession.

`public` **int GetClientID**( )

Getter für clientID.

`public` **Server.Communication.ConcreteSessionManagement GetConcreteSessionManagement**( )

Getter für concreteSessionManagement.

`public` **Server.Communication.ClientPluginConnection GetConnectionFor**( )

Liefert für einen Handshake die passende ClientPluginConnection und bereitet die ClientPluginConnection für den Datentransfer vor.

Parameter

* `handshake` -
* `tcpClient` - Der aktive Socket.

`public` **Common.Logic.EventQueue GetInputEventQueue**( )

Getter für queue.

`public` **Common.Logic.EventQueue GetOutputEventQueue**( )

Getter für die Queue der Applikation.

`public` **Common.Communication.BasicPlugin GetPreferredBasicPlugin**( )

Liefert eine ausgewählte ClientPluginConnection (ist ein BasicPlugin). Bisher gibt es nur eine.

`public` **void HandleHandshakeAnswer**( )

Ist auf dem Server nicht implementiert, da sich der Server niemals aktiv zu einem Client verbindet.

Parameter

* `handshakeAnswer` - Die HandshakeAnswer.

`public` **void OnConnectionLost**( )

Wird aufgerufen, wenn ein BasicPlugin keine Verbindung mehr hat. Das Plugin wird geschlossen und die Observer benachrichtigt, dass der Client entfernt werden soll.

Parameter

* `sender` - Das Objekt, welches das Event gesendet hat.
* `plugin` - Das BasicPlugin, für das die Verbindung unterbrochen worden ist.

`public` **void OnConnectionTerminatorFound**( )

Beendet eine Verbindung auf Anfrage. Das Plugin wird geschlossen und die Observer benachrichtigt, dass der Client entfernt werden soll.

Parameter

* `sender` - Der Aufrufer.
* `connectionTerminator` - Die Anfrage zum schließen einer Verbindung.

`public` **Server.Communication.ClientPluginConnection TryFindClientPluginConnectionByID**( )

Versucht, eine ClientPluginConnection in der Liste anhand der pluginID zu finden.

Parameter

* `pluginID` - Nach dieser pluginID wird gesucht.

F.21.5 Klasse SessionManagement

Die SessionManagement-Schnitstelle stellt die Interaktionsmöglichkeiten für andere Klassen mit dem Communication-Layer zu Verfügung.

Konstruktoren

`public` **SessionManagement**( )

Erzeugt ein neues SessionManagement-Objekt mit einer ServerApplication.

Parameter

* `server` - das ServerApplication-Objekt.

Methoden

`public` **void Attach**( )

Registriert einen ServerCommunicationObserver.

Parameter

* `observer` - der Observer

`public` **void Detach**( )

Entfernt einen ServerCommunicationObserver aus der Liste.

Parameter

* `observer` - der Observer

`public` **void PutEvent**( )

Versendet ein Event an seinen Empfänger.

Parameter

* `inputEvent` - das zu versendende Event

`public` **void RemoveSession**( )

Wird aufgerufen, wenn ein Client entgültig aus dem Spiel genommen werden soll, bspw. wenn er sich zu lange außerhalb des Spielfeldes aufgehalten hat.

Parameter

* `clientID` - Die clientId der zu entfernenden Session.

F.22 Namespace Server.Logic

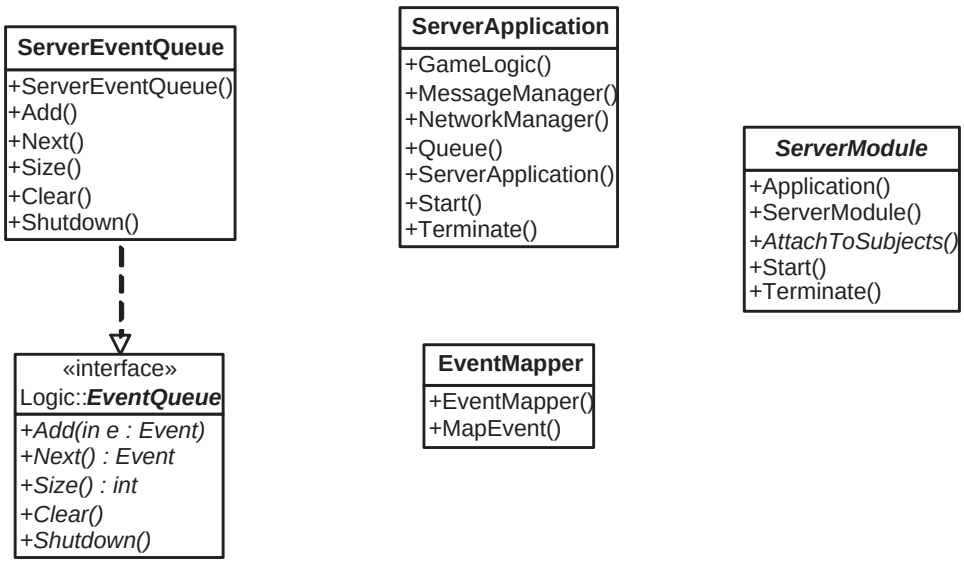


Abbildung F.22: UML Struktur des Namensraum Server.Logic

F.22.1 Klasse EventMapper

Diese Klasse stellt die Verbindung zwischen Events und Methodenaufrufen in der GameLogic her.

Konstruktoren

`public` **EventMapper**( )

Erzeugt neues NabbEventMapper Objekt.

Parameter

- `gameLogic` - die zugehörige GameLogic

Methoden

`public` **void MapEvent**( )

In dieser Methode werden Events identifiziert und die entsprechenden Methoden in der GameLogic aufgerufen.

Parameter

- `e` - zuzuordnendes Event

F.22.2 Klasse ServerApplication

NabbServer ist die Hauptklasse für den Server. Sie enthält Methoden zum Steuern des Servers und dem Event-Verarbeitungsprozess.

Konstruktoren

`public` **ServerApplication**( )

Erzeugt neues NabbServer-Objekt.

Methoden

`public` **void Start**( )

Startet die Anwendung. Diese Methode muss in der Main() Methode der konkreten Applikation aufgerufen werden. In dieser Methode, sollen alle Threads, die zu der Applikation gehören, gestartet werden, und die GUI soll angezeigt werden.

`public` **void Terminate**( )

Beendet die Anwendung. Diese Methode soll aufgerufen werden, wenn das Programm durch den Nutzer (oder einen nicht behebbareb Fehler) beendet wird. Nach dem Aufruf dieser Methode sollen alle Threads beendet werden, und es sollen alle weiteren verwendeten Ressourcen freigegeben werden

F.22.3 Klasse ServerEventQueue

Diese Klasse ist eine Implementierung der Schnittstelle EventQueue. Die Implementierung ist Thread-sicher. Es können mehrere Threads gleichzeitig auf die Add(Event) und Next() Methoden zugreifen.

Konstruktoren

`public` **ServerEventQueue**( )

Erzeugt neues ServerEventQueue Objekt.

Methoden

`public` **void Add**( )

Hängt der Warteschlange ein Event an.

Parameter

- `e` - anzuhängendes Event

`public` **void Clear**( )

Leert die Warteschlange.

`public` **Common.Logic.Statemanager.Event Next**( )

Holt das vorderste Event aus der Warteschlange heraus. Ist gerade kein Event in der Warteschlange, blockiert die Methode, bis entweder ein Event zur Verfügung steht, oder die Methode Shutdown() aufgerufen wurde. In letzterem Fall liefert diese Methoden NULL statt eines Events zurück.

`public` **void Shutdown**( )

Terminiert die Warteschlange. Die wartenden Events werden gelöscht. Eingehende Events werden ignoriert. Falls Threads in der Methode Next() blockiert sind, werden sie aufgeweckt und liefert NULL statt einem Event zurück.

`public` **int Size**( )

Liefert die Anzahl der wartenden Events.

F.22.4 Klasse ServerModule

Diese Klasse muss von allen Modulen erweitert werden, die beim Server registriert werden sollen und Observer bedienen.

Konstruktoren

`public` **ServerModule**( )

Erzeugt neues ServerModule Objekt.

Parameter

* `app` -

Methoden

`public` **void AttachToSubjects**( )

Weißt das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll.

`public` **void Terminate**( )

Diese Methode wird von der Applikation aufgerufen, wenn sie sich beendet. Jedes Modul kann sie überschreiben, um vor dem Beenden z.B. belegte Ressourcen freizugeben.

F.23 Namespace Server.Logic.Game

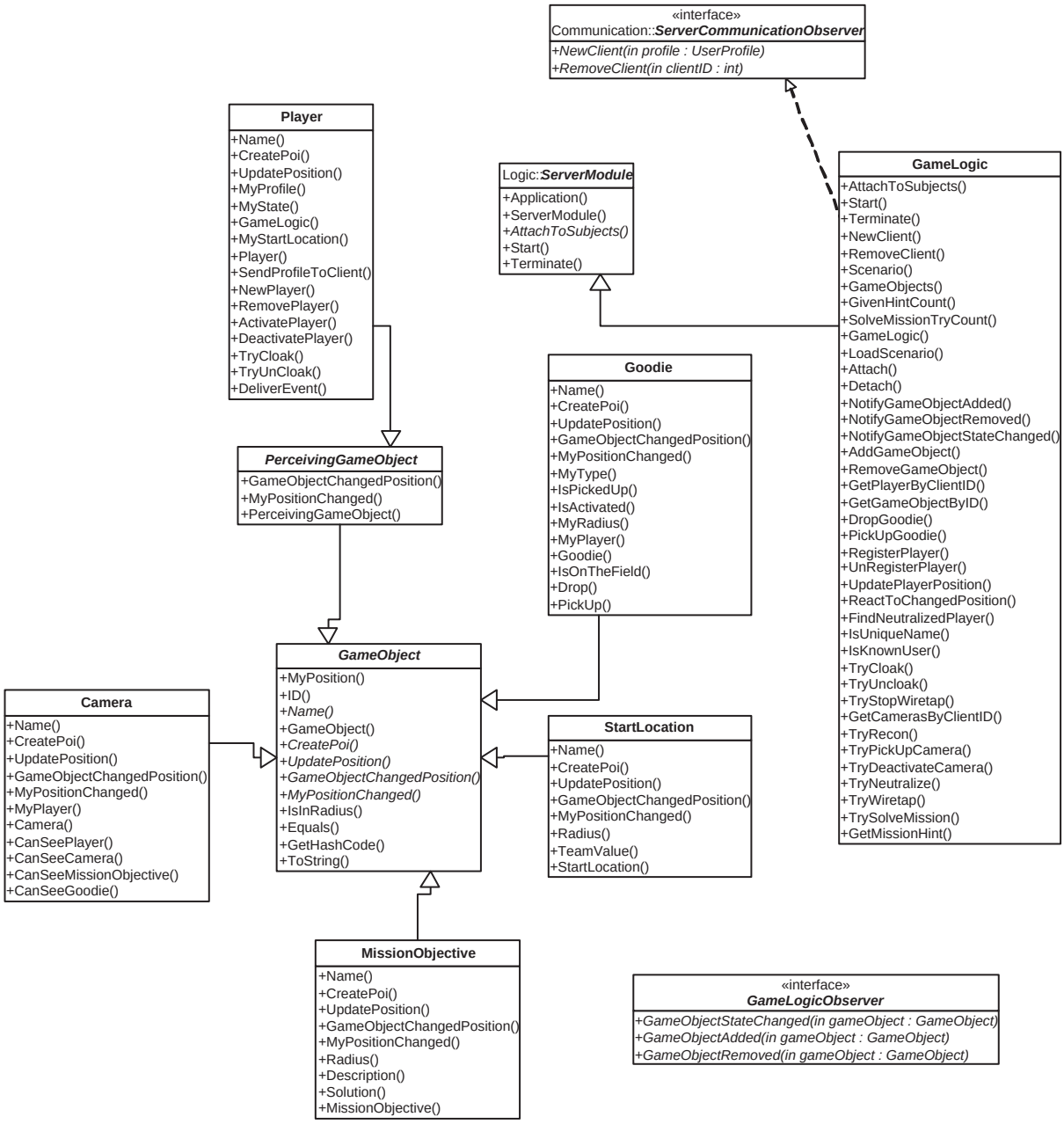


Abbildung F.23: UML Struktur des Namensraum Server.Logic.Game

F.23.1 Schnittstelle GameLogicObserver

Diese Schnittstelle wird von allen Klassen implementiert, die über Zustandsänderungen der GameLogic informiertwerden wollen. Dazu registrieren sich die Klassen bei dieser als Observer. Das Interface stellt für wichtige Änderungen der Spielobjekte eine eigene Update-Methode bereit.

Methoden

`public` **void GameObjectAdded**( )

Wird aufgerufen, wenn ein Spielobjekt hinzugefügt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectRemoved**( )

Wird aufgerufen, wenn ein Spielobjekt entfernt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectStateChanged**( )

Wird aufgerufen, wenn sich der Zustand eines Spielobjekts geändert hat.

Parameter

* `gameObject` - das betroffene Spielobjekt

F.23.2 Klasse Camera

Dieses Spielobjekt stellt eine Aufklärungskamera dar. Wie bei allen Spielobjekten werden die aktuelle Position und die ID gespeichert.

Konstruktoren

`public` **Camera**( )

Erzeugt ein neues Kamera-Spielobjekt.

Parameter

* `gameLogic` - Referenz auf Spiellogik
* `position` - Position der Kamera
* `myPlayer` - der Spieler, der diese Aufklärungskamera platziert hat

Methoden

`public` **bool CanSeeCamera**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen einer Kamera. Es überprüft, ob die angegebene Kamera von diesem Spielobjekt gesehen wird.

Parameter

* `camera` - Kamera, deren Zustand sich geändert hat

`public` **bool CanSeeGoodie**( )

In dieser Methode wird geprüft, ob diese Camera das Goodie sehen kann.

Parameter

* `goodie` - Goodie, das überprüft wird

`public` **bool CanSeeMissionObjective**( )

In dieser Methode überprüft dieses Spielobjekt, ob es das Missionsziel sehen kann.

Parameter

* `missionObjective` - Missionsziel

`public` **bool CanSeePlayer**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen eines anderen Spielers. Es überprüft, ob der angegebene Spieler von diesem Spielobjekt gesehen wird.

Parameter

* `player` - Spieler, dessen Zustand sich geändert hat

`public` **Common.Logic.Pois.PointOfInterest CreatePoi**( )

Erzeugt ein neues PointOfInterest für dieses GameObject. Die Id des GameObjects wird als PoiId verwendet.

`public` **void GameObjectChangedPosition**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen eines anderen GameObjects. Sie wird aufgerufen wenn eine Positionsänderung des angegebenen GameObjects stattgefunden hat. Sie ist leer, da Kameras nur indirekt (über das Spieler-Ojekt) auf Positionsänderungen reagieren.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void MyPositionChanged**( )

Diese Methode wird aufgerufen wenn sich die Position dieses GameObjects relativ zu dem übergebenen Object verändert hat. Dabei steht das andere Object still. Hier wird MyPositionChanged des Spieler-Objekts dieser Kamera aufgerufen, da dort überprüft wird, ob diese Kamera etwas sieht, oder nicht.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void UpdatePosition**( )

Setzt dieses GameObject an einen neuen Ort. Je nach Art des GameObjects, wird die detection-Logik ausgeführt und entsprechende Events an Clients verschickt.

Parameter

* `newPosition` - die neue Position dieses GameObjects

F.23.3 Klasse GameLogic

Die GameLogic enthält die Hauptmethoden der Spiellogik. Eingehende Events verursachen Methodenaufrufe in dieser Klasse, um den Spielzustand zu ändern.

Konstruktoren

`public` **GameLogic**( )

Erzeugt eine neues GameLogic Modul.

Parameter

* `app` - Referenz auf die Serverapplikation

Methoden

`public` **void AddGameObject**( )

Mit dieser Methode werden neue Spielobjekte dem Spiel hinzugefügt.

Parameter

* `gameObject` - das neue Spielobjekt

`public` **void Attach**( )

Fügt der Liste der Observer ein neuen GameLogicObserver hinzu.

Parameter

* `observer` - der neue GameLogicObserver

`public` **void AttachToSubjects**( )

Weißt das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void Detach**( )

Entfernt einen GameLogicObserver aus der Liste der Observer.

Parameter

* `observer` - der zu entfernende GameLogicObserver

`public` **void DropGoodie**( )

Legt das Goodie mit der angegebenen Id an der Position ab.

Parameter

* `goodieId` - Id des Goodies
* `pos` - neue Position des Goodies

`public` **System.Collections.ArrayList FindNeutralizedPlayer**( )

Gibt die Lister aller Spieler zurück, die sich im Neutralizierungs- radius des angegebenen Spielers befinden.

Parameter

* `neutralisingPlayer` - von diesem Spieler geht die Neutralisierung aus

`public` **System.Collections.ArrayList GetCamerasByClientID**( )

Liefert alle Kameras, die dem angegebenen Spieler gehören.

Parameter

* `player` - Spieler

`public` **Server.Logic.Game.GameObject GetGameObjectByID**( )

Liefert das Spielobjekt zu der angegebene Objekt-ID. Ist keine Spielobjekt für diese Objekt-ID gespeichert, wird eine DataObjectNotFoundException ausgelöst.

Parameter

* `objectID` - Objekt-ID

`public` **void GetMissionHint**( )

Wird aufgerufen wenn der Spieler mit der angegebenen Id das Goodie mit der angegebenen Id aktiviert und einen Hint für den Code des Missionsziels bekommen soll.

Parameter

* `clientID` - Id des Spielers
* `goodieID` - Id des Goodies

`public` **Server.Logic.Game.Player GetPlayerByClientID**( )

Liefert das Player Objekt zu der angegebene Client-ID. Ist keine Player Objekt für diese Client-ID gespeichert, wird eine DataObjectNotFoundException ausgelöst.

Parameter

* `clientID` - Client-ID

`public` **bool IsKnownUser**( )

Überprüft, ob das angegeben UserProfile bekannt ist.

Parameter

* `profile` - zu überprüfendes Spielerprofil

`public` **bool IsUniqueName**( )

Überprüft, ob der angegebene Name bereits von einem registrierten Spieler benutzt wird.

Parameter

* `name` - Name, der auf Einzigeartigkeit überprüft werden soll

`public` **void LoadScenario**( )

Läd das Scenario aus der angegebenen Datei.

Parameter

* `scenarioFileName` - Pfad zur Scenario XML-Datei

`public` **void NewClient**( )

Diese Observer Methode wird von der Comm-Layer aufgerufen wenn sich ein neuer Spieler anmeldet. Das Profil enthält eine gültige UserID und UserName.

Parameter

* `profile` - Profile das der Server dem Client zugewiesen hat

`public` **void NotifyGameObjectAdded**( )

Wird aufgerufen, wenn ein Spielobjekt hinzugefügt wurde und alle GameLogicObserver informiert werden sollen.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void NotifyGameObjectRemoved**( )

Wird aufgerufen, wenn ein Spielobjekt entfernt wurde und alle GameLogicObserver informiert werden sollen.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void NotifyGameObjectStateChanged**( )

wird aufgerufen, wenn sich der Zustand eines Spielobjekts geändert hat und alle GameLogicObserver informiert werden sollen.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void PickUpGoodie**( )

Setzt den Goodie-Status des Goodies mit der angegebenen Id auf 'isPickedUp'.

Parameter

* `goodieId` - Id des Goodies
* `clientId` - Id des Clients, der das Goodie aufnimmt

`public` **void ReactToChangedPosition**( )

In dieser Methode wird die Sichtbarkeit von dem angegebenen GameObject gegen alle anderen GameObjects und umgekehrt geprüft.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void RegisterPlayer**( )

Fügt der Benutzerverwaltung einen neuen Spieler hinzu. Erzeugt für jeden anderen Client ein USER_ADDED UserProfileEvent in der NetworkManager queue. Der Client erscheint als erstes an der default Position aus dem Scenario.

Parameter

* `userProfile` - Das NabbUserProfile des neuen Spielers.

`public` **void RemoveClient**( )

Diese Observer Methode wird von der Comm-Layer aufgerufen wenn sich ein Spieler abmeldet.

Parameter

* `clientID` - Die Id des Spielers der sich abmeldet

`public` **void RemoveGameObject**( )

Mit dieser Methode werden Spielobjekte aus dem Spiel entfernt.

Parameter

* `gameObject` - das zu entfernende Spielobjekt

`public` **void Start**( )

Diese Methode wird von der Applikation aufgerufen, wenn das Modul seine eventuell vorhandenen internen Threads starten soll.

`public` **void Terminate**( )

Diese Methode wird von der Applikation aufgerufen, wenn sie sich beendet. Jedes Modul kann sie überschreiben, um vor dem Beenden z.B. belegte Ressourcen freizugeben.

`public` **void TryCloak**( )

Versucht den Spieler, der durch das Ereignis referenziert wird, zu tarnen.

Parameter

* `request` - Tarnanfrage-Ereignis

`public` **void TryDeactivateCamera**( )

Versucht eine Kamera zu deaktivieren.

Parameter

* `request` - Kameradeaktivieren-Ereignis

`public` **void TryNeutralize**( )

Anfrage des Client, um einen Neutralisationsvorgang einzuleiten.

Parameter

* `request` - das empfangene Event

`public` **void TryPickUpCamera**( )

Versucht eine Kamera aufzunehmen.

Parameter

* `request` - Kameraaufnehmen-Ereignis

`public` **void TryRecon**( )

Versucht eine Kamera zu platzieren.

Parameter

* `request` - Kamerasetzen-Ereignis

`public` **void TrySolveMission**( )

Wird aufgerufen wenn der Spieler mit der angegebenen Id versucht das angegebene Missionsziel zu lösen.

Parameter

* `clientID` - Id des Spielers
* `missionObjectiveID` - Missionsziel
* `solution` - String mit der Lösung

`public` **void TryStopWiretap**( )

Stoppt das Abhören.

Parameter

* `request` - Die Anfrage

`public` **void TryUncloak**( )

Versucht den Spieler, der durch das Ereignis referenziert wird, zu enttarnen.

Parameter

* `request` - Enttarnanfrage-Ereignis

`public` **void TryWiretap**( )

Anfrage des Client, um einen Abhörvorgang einzuleiten.

Parameter

* `request` - das empfangene Event

`public` **void UnRegisterPlayer**( )

Entfernt einen Spieler aus der Benutzerverwaltung. Erzeugt für jeden anderen Client ein USER_REMOVED UserProfileEvent in der NetworkManager queue. Sollte der Spieler noch Goodies bei sich tragen, werden diese abgelegt. Alle dem Spieler gehörenden Spielobjekte werden entfernt und die Clients benachrichtigt.

Parameter

* `clientID` - Die Id des Spielers.

`public` **void UpdatePlayerPosition**( )

Wird aufgerufen, wenn der Spieler mit der angegebenen ID seine Position geändert hat.

Parameter

* `clientID` - ID des Spielers
* `position` - die neue Position

F.23.4 Klasse GameObject

Die Klasse GameObject bildet die abstrakte Grundlage für alle Spielobjekte. Es werden allgemeine Methoden zur Verfügung gestellt, um die Position der Spielobjekte zur verwalten und auf andere Spielobjekte reagieren zu können.

Konstruktoren

`public` **GameObject**( )

erzeugt neues PerceivingGameObject Objekt.

Parameter

* `gameLogic` - Referenz auf das zugehörige Modul
* `position` - Initialer Standort des Spielobjekts

Methoden

`public` **Common.Logic.Pois.PointOfInterest CreatePoi**( )

Erzeugt ein neues PointOfInterest für dieses GameObject. Die Id des GameObjects wird als PoiId verwendet.

`public` **bool Equals**( )

Vergleicht dieses Spielobjekt mit dem angegebenen Objekt anhand seiner ID.

Parameter

* `obj` - zu vergleichendes Objekt

`public` **void GameObjectChangedPosition**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen eines anderen GameObjects. Sie wird aufgerufen wenn eine Positionsänderung des angegebenen GameObjects stattgefunden hat.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **int GetHashCode**( )

liefert den Hashcode zu diesem GameObject

`public` **bool IsInRadius**( )

Überprüft, ob die Distanz in Metern dieses GameObjects zu der angegebenen Position kleiner gleich der angegeben Distanz in Metern ist.

Parameter

* `position` - Position, zu der die Entfernung berechnet wird.
* `distance` - maximale Entfernung zu der Position in Metern

`public` **void MyPositionChanged**( )

Diese Methode wird aufgerufen wenn sich die Position dieses GameObjects relativ zu dem übergebenen Object verändert hat. Dabei steht das andere Object still.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **string ToString**( )

Liefert die Stringrepräsentation zu diesem GameObject.

`public` **void UpdatePosition**( )

Setzt dieses GameObject an einen neuen Ort. Je nach Art des GameObjects, wird die detection-Logik ausgeführt und entsprechende Events an Clients verschickt.

Parameter

* `newPosition` - die neue Position dieses GameObjects

F.23.5 Klasse Goodie

Diese Klasse enthält alle Daten und Methoden für ein Goodie.

Konstruktoren

`public` **Goodie**( )

Erzeugt ein neues Goodie.

Parameter

* `gameLogic` - Referenz auf das zugehörige Modul
* `position` - die Position des Spielers
* `type` - der Typ des Goodies
* `radius` - der Radius des Goodies

Methoden

`public` **Common.Logic.Pois.PointOfInterest CreatePoi**( )

Erzeugt ein neues GoodieItem PointOfInterest für dieses Goodie. Die Id des Goodies wird als PoiId verwendet.

`public` **void Drop**( )

Setzt dieses Goodie an der angegebenen Position ab, wenn es von einem Spieler aufgenommen ist.

Parameter

* `pos` - neue Position des Goodies

`public` **void GameObjectChangedPosition**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen eines anderen GameObjects. Sie wird aufgerufen wenn eine Positionsänderung des angegebenen GameObjects stattgefunden hat. Sie ist leer, da Goodies nicht auf Positionsänderungen reagieren.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **bool IsOnTheField**( )

Diese Methode gibt an, ob sich das Goodie auf dem Spielfeld befindet. Dies ist genau dann der Fall, wenn es sichtbar und nicht von einem Spieler aufgenommen worden ist.

`public` **void MyPositionChanged**( )

Diese Methode wird aufgerufen wenn sich die Position dieses GameObjects relativ zu dem übergebenen Object verändert hat. Dabei steht das andere Object still. Sie ist leer, da Goodies nicht auf Positionsänderungen reagieren.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void PickUp**( )

Wird aufgerufen, um dieses Goodie vom angegebenen Spieler aufnehmen zu lassen.

Parameter

* `player` - der Spieler, der dieses Goodie erhält

`public` **void UpdatePosition**( )

Setzt dieses Goodie an einen neuen Ort. Goodies dürfen nur im Spielfeld erscheinen oder abgelegt werden.

Parameter

* `newPosition` - die neue Position dieses Goodies

F.23.6 Klasse MissionObjective

Diese Klasse enthält alle Daten und Methoden für ein Missionsziel.

Konstruktoren

`public` **MissionObjective**( )

Erzeugt neues MissionObjective Objekt.

Parameter

* `gameLogic` - Referenz auf das zugehörige Modul
* `position` - Die Position des Spielers
* `type` - der Typ dies Missionsziels
* `radius` - der Radius des Missionsziels
* `description` - die Beschreibung des Missionsziels
* `solution` - die Lösung für das Missionsziel

Methoden

`public` **Common.Logic.Pois.PointOfInterest CreatePoi**( )

Erzeugt ein neues PointOfInterest für dieses GameObject. Die Id des GameObjects wird als PoiId verwendet.

`public` **void GameObjectChangedPosition**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen eines anderen GameObjects. Sie wird aufgerufen wenn eine Positionsänderung des angegebenen GameObjects stattgefunden hat.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void MyPositionChanged**( )

Diese Methode wird aufgerufen wenn sich die Position dieses GameObjects relativ zu dem übergebenen Object verändert hat. Dabei steht das andere Object still.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void UpdatePosition**( )

Setzt dieses GameObject an einen neuen Ort. Je nach Art des GameObjects, wird die detection-Logik ausgeführt und entsprechende Events an Clients verschickt.

Parameter

* `newPosition` - die neue Position dieses GameObjects

F.23.7 Klasse PerceivingGameObject

Ein PerceivingGameObject hat die Fähigkeit, andere GameObjects warzunehmen. Dazu gehören z.B. Spieler. PerceivingGameObject führen eine Liste der im Moment für sie sichtbaren GameObjects. In den Methoden GameObjectChangedPosition und MyPositionChanged wird überprüft ob das angegebene GameObject gesehen wird und gegen die breits gesehenen GameObject geprüft.

Konstruktoren

`public` **PerceivingGameObject**( )

Erzeugt neues PerceivingGameObject Objekt.

Parameter

* `gameLogic` - Referenz auf das zugehörige Modul
* `position` - Initialer Standort des Spielobjekts

Methoden

`public` **void GameObjectChangedPosition**( )

In dieser Methode reagiert dieses GameObject auf Veränderungen eines anderen GameObjects. Wenn eine Positionsänderung des angegebenen GameObjects stattgefunden hat, wird überprüft, ob dieses GameObject das angegebene sieht oder nicht. Dabei muss die Liste der für dieses GameObject sichtbaren GameObjects aktualisiert werden. Veränderungen werden über Events an den entsprechenden Client verschickt. Ist das Objekt bekannt und sichtbar, wird die Reaktion durch die Methode GameObjectMoved(GameObject) ausgeführt. War das Objekt nicht bekannt, ist aber jetzt sichtbar, reagiert dieses Spielobjekt durch die Methode GameObjectDiscovered(GameObject). War das Objekt bekannt, ist aber nicht mehr sichtbar, wird die Methode GameObjectLost(GameObject aufgerufen). War das Objekt nicht bekannt und ist jetzt auch nicht sichtbar, findet keine Reaktion statt.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

`public` **void MyPositionChanged**( )

Diese Methode wird aufgerufen wenn sich die Position dieses GameObjects relativ zu dem übergebenen Object verändert hat. Dabei steht das andere Object still. Es wird überprüft, ob dieses GameObject das angegebene sieht oder nicht. Dabei muss die Liste der für dieses GameObject sichtbaren GameObjects aktualisiert werden. Veränderungen werden über Events an den entsprechenden Client verschickt. Ist das Objekt bekannt und sichtbar, findet keine Reaktion statt, weil es sich nichts geändert hat. War das Objekt nicht bekannt, ist aber jetzt sichtbar, reagiert dieses Spielobjekt durch die Methode GameObjectDiscovered(GameObject). War das Objekt bekannt, ist aber nicht mehr sichtbar, wird die Methode GameObjectLost(GameObject aufgerufen). War das Objekt nicht bekannt und ist jetzt auch nicht sichtbar, findet keine Reaktion statt.

Parameter

* `gameObject` - das GameObject, dessen Zustand sich geändert hat

F.23.8 Klasse Player

Das Player-Objekt representiert einen Spieler. Es verfügt über ein Profil in dem die Daten des Spielers gespeichert werden. Wie bei allen Spielobjekten werden die aktuelle Position, die ID und die von diesem Spielobjekt gesehenen Spielobjekte gespeichert.

Konstruktoren

`public` **Player**( )

Erzeugt einen neuen Spieler mit dem angegebenen Profil und der angegebenen Position.

Parameter

- `gameLogic` - Referenz auf das zugehörige Modul
- `position` - die Position des Spielers
- `profile` - das Profil dieses Spielers

Methoden

`public` **void ActivatePlayer**( )

Aktiviert diesen Spieler. Der Zustand wird auf aktive gesetzt und der Client per Event informiert.

`public` **Common.Logic.Pois.PointOfInterest CreatePoi**( )

Erzeugt ein neues PointOfInterest für dieses GameObject. Die Id des GameObjects wird als PoiId verwendet.

`public` **void DeactivatePlayer**( )

Deaktiviert diesenn Spieler. Der Zustand wird auf passive gesetzt und der Client per Event informiert.

`public` **void DeliverEvent**( )

Dieser Methode leitet ein Ereignis an die Netzwerkschicht und den ClientState dieses Spieler weiter.

Parameter

- `e` - weiterzuleitendes Event

`public` **void NewPlayer**( )

Diese Methode erzeugt ein Ereignis, dass den zugehörigen Client über einen neuen Spieler informiert.

Parameter

- `player` - der neue Spieler

`public` **void RemovePlayer**( )

Diese Methode erzeugt ein Ereignis, dass den zugehörigen Client informiert, das ein anderer Spieler entfernt wurde.

Parameter

- `player` - der entfernte Spieler

`public` **void SendProfileToClient**( )

Schickt das eigene Profil an den Client zwecks Update.

`public` **void TryCloak**( )

Versucht diesen Spieler zu tarnen. Sendet ein Cloaked Event falls tarnen möglich, sonst ein CloakUnavailable Event.

`public` **void TryUnCloak**( )

Versucht diesen Spieler zu enttarnen. Sendet ein Cloaked Event falls tarnen möglich, sonst ein CloakUnavailable Event.

`public` **void UpdatePosition**( )

Setzt dieses GameObject an einen neuen Ort. Je nach Art des GameObjects, wird die detection-Logik ausgeführt und entsprechende Events an Clients verschickt.

Parameter

* `newPosition` - die neue Position dieses GameObjects

F.23.9 Klasse StartLocation

Diese Klasse enthält alle Daten und Methoden für einen Startpunkt eines Spielers. Beim Erreichen des Startpunkts wird der Zustand des Spielers auf aktiv gesetzt. Er wird nur angezeigt, wenn der Spieler sich im Zustand passiv befindet.

Konstruktoren

`public` **StartLocation**( )

Erzeugt neuen Startpunkt.

Parameter

* `gameLogic` - Referenz auf das zugehörige Modul
* `position` - Initialer Standort des Spielobjekts
* `radius` - Radius, innerhalb dessen ein Spieler aktiviert wird
* `team` - ID des Teams, zu dem dieser Startpunkt gehört

Methoden

`public` **Common.Logic.Pois.PointOfInterest CreatePoi**( )

Erzeugt ein neues PointOfInterest für dieses GameObject. Die Id des GameObjects wird als PoiId verwendet.

`public` **void GameObjectChangedPosition**( )

Das angegebene Spielobjekt hat sich bewegt. StartLocations können keine Spielobjekte sehen, reagieren aber auf die Bewegung von Spielern, falls ein Spieler aktiviert werden muss.

Parameter

* `gameObject` - das bewegte Spielobjekt

`public` **void MyPositionChanged**( )

Dieses StartLocation-Objekt befindet sich an einer neuen Position. Es wird überprüft ob es sich in Reichweite eines angegebenen Spielers ist, falls dieser aktiviert werden muss.

Parameter

* `gameObject` - Spielobjekt zu dem eine Positionsveränderung stattgefunden hat

`public` **void UpdatePosition**( )

Setzt dieses GameObject an einen neuen Ort. Je nach Art des GameObjects, wird die detection-Logik ausgeführt und entsprechende Events an Clients verschickt.

Parameter

* `newPosition` - die neue Position dieses GameObjects

F.24 Namespace Server.Logic.Messagemanager

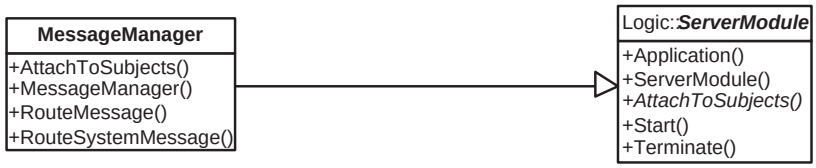


Abbildung F.24: UML Struktur des Namensraum Server.Logic.Messagemanager

F.24.1 Klasse MessageManager

Die Klasse MessageManager auf der Seite des Servers dient dazu, Nachrichten an die jeweiligen Empfänger weiterzuleiten. Darüber hinaus werden Nachrichten an jeden berechtigten Abhöragenten, der nicht zum Team des Absenders der Nachricht gehört, weitergeleitet.

Konstruktoren

`public` **MessageManager**( )

Erzeugt neues MessageManager-Objekt.

Parameter

* `server` - Referenz auf die zugehörige Anwendung

Methoden

`public` **void AttachToSubjects**( )

Weist das Modul an, sich als Observer bei allen gewünschten anderen Modulen zu registrieren. Wenn diese Methode aufgerufen wird, sind alle über die Application erreichbaren Module bereits initialisiert.

`public` **void RouteMessage**( )

Erstellt für jeden Eintrag der Empfängerliste der Nachricht ein neues MessageEvent, bestehend aus ID des Empfängers und der Nachricht selbst. Darüber hinaus wird für jeden berechtigten Abhöragenten, der nicht zum Team des Absenders der Nachricht gehört, ebenfalls ein solches MessageEvent erstellt. Abgehörte Nachrichten werden durch einen entsprechenden Eintrag im Betreff gekennzeichned. Alle MessageEvents werden an den NetworkManager weitergeleitet.

Parameter

* `message` - weiterzuleitende Nachricht

`public` **void RouteSystemMessage**( )

Erstellt aus übergebener ClientID und Systemnachricht ein SystemMessageEvent und leitet dieses an den NetworkMananger weiter.

Parameter

* `clientID` - ID des Empfängers
* `message` - zu versendende Nachricht

F.25 Namespace Server.Logic.Statemanager

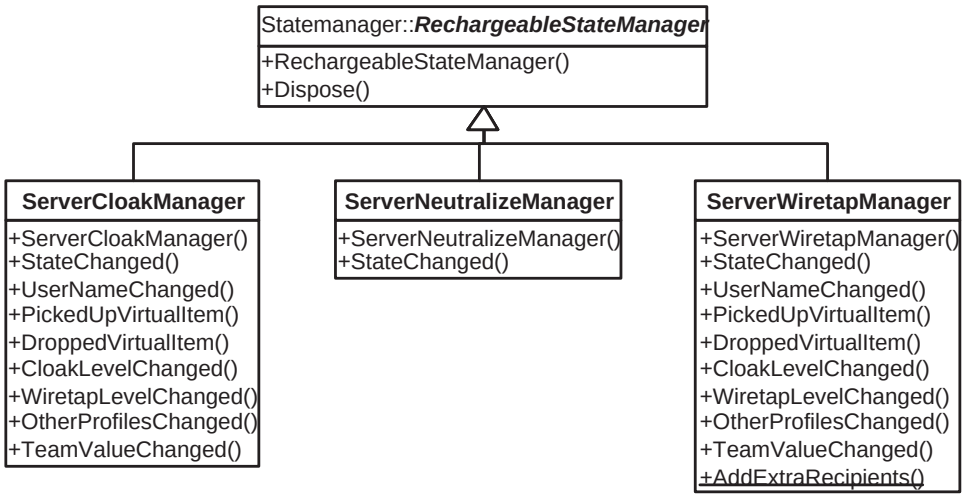


Abbildung F.25: UML Struktur des Namensraum Server.Logic.Statemanager

F.25.1 Klasse ServerCloakManager

Verwaltet das Tarnen eines Spielers auf Serverseite. Veränderungen im Benutzerprofil bekommt der ServerCloakManager durch den UserProfileObserver mit. Interessant ist dabei nur die Veränderung des CloakLevels, da sich dadurch die Tarndauer ab dem nächsten Tarnvorgang verändert.

Konstruktoren

`public` **ServerCloakManager**( )

Erzeugt ein neues ServerCloakManager-Objekt.

Parameter

- `player` - Das Playerobjekt des zugehörigen Spielers.

Methoden

`public` **void CloakLevelChanged**( )

Wird aufgerufen, wenn das CloakLevel neu gesetzt wurd.

Parameter

- `newValue` - Der neues Fähigkeitenwert.

`public` **void DroppedVirtualItem**( )

wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand abgelegt hat.

Parameter

- `Items` - virtueller Gegenständ, den der Nutzer abgelegt hat.

`public` **void OtherProfilesChanged**( )

Wird aufgerufen, wenn sich die Liste der Profile anderer Spieler ändert.

`public` **void PickedUpVirtualItem**( )

wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand aufgenommen hat.

Parameter

* `Items` - virtueller Gegenständ, den der Nutzer aufgenommen hat.

`public` **void StateChanged**( )

Implementierung der RechargeableStateManager-Schnittstelle. Reagiert auf Veränderungen des Cloak-Automatons. Tarnt oder enttarnt sich der Spieler, so wird das über den Aufruf ReactToChangedPosition bei der Gamelogic den anderen Spielern mitgeteilt. Der Timer für die maximale Tarndauer wird beim tarnen gesetzt und beim enttarnen gestoppt.

Parameter

* `state` - der geänderte Zustand

`public` **void TeamValueChanged**( )

Wird aufgerufen, wenn sich die Teamzugehörigkeit eines Spielers ändert.

Parameter

* `team` -

`public` **void UserNameChanged**( )

wird aufgerufen, wenn sich das Pseudonym der Nutzers geändert hat

Parameter

* `userName` - neues Pseudonym

`public` **void WiretapLevelChanged**( )

Wird aufgerufen, wenn das WiretapLevel neu gesetzt wurd.

Parameter

* `newValue` - Der neues Fähigkeitenwert.

F.25.2 Klasse ServerNeutralizeManager

Verwaltet das Neutralisieren eines Spielers auf Serverseite und versendet die NABB stats.

Konstruktoren

`public` **ServerNeutralizeManager**( )

Erzeugt eine neues ServerNeutralizeManager-Objekt.

Parameter

* `player` - Das Playerobjekt des zugehörigen Spielers.

Methoden

`public` **void StateChanged**( )

Implementierung der RechargeableStateManager-Schnittstelle. Reagiert auf Veränderungen des Neutralize-Automatons. Wird neutralisiert (Timer ist ausgelöst), so werden alle Spieler in Neutralisierungreichweite dieses Spielers mittels DeactivatePlayer auf passiv gesetzt und der neutralisierende Spieler erhält, wenn mindestens ein Spieler neutralisiert wurde, die NABB stats. Der Timer für die Neutralisierungsdauer wird beim NeutralizeState.InProgress geladen.

Parameter

* `state` - der geänderte Zustand

F.25.3 Klasse ServerWiretapManager

Verwaltet das Abhören eines Spielers auf Serverseite. Veränderungen im Benutzerprofil bekommt der ServerWiretapManager durch den UserProfileObserver mit. Interessant ist dabei nur die Veränderung des WiretapLevels, da sich dadurch die Abhördauer ab dem nächsten Abhörvorgang verändert. Der ServerWiretapManager sorgt mit der Methode AddExtraRecipients, die vom MessageManager aufgerufen wird, dafür, dasss abhörende Spieler (eventuell) die versendete Nachricht auch bekommen.

Konstruktoren

`public` **ServerWiretapManager**( )

Erzeugt ein neues ServerWiretapManager-Objekt.

Parameter

* `player` - Das Playerobjekt des abhörenden Spielers.

Methoden

`public` **void AddExtraRecipients**( )

Trägt in eine Nachricht eventuell zusätzliche Recipients ein. Wenn ein Spieler abhört, hat er die Chance die Nachricht auch zu erhalten, wenn er nicht schon in der Liste steht.

Parameter

* `app` - Die ServerApplication.
* `message` - Die von einem Client gesendete Nachricht

`public` **void CloakLevelChanged**( )

Wird aufgerufen, wenn das CloakLevel neu gesetzt wurd.

Parameter

* `newValue` - Der neues Fähigkeitenwert.

`public` **void DroppedVirtualItem**( )

wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand abgelegt hat.

Parameter

* `Items` - virtueller Gegenständ, den der Nutzer abgelegt hat.

`public` **void OtherProfilesChanged**( )

Wird aufgerufen, wenn sich die Liste der Profile anderer Spieler ändert.

`public` **void PickedUpVirtualItem**( )

wird aufgerufen, wenn der Nutzer einen virtuellen Gegenstand aufgenommen hat.

Parameter

* `Items` - virtueller Gegenständ, den der Nutzer aufgenommen hat.

`public` **void StateChanged**( )

Implementierung der RechargeableStateManager-Schnittstelle. Reagiert auf Veränderungen des Wiretap-Automatons. Der Timer für die maximale Abhördauer wird beim aktivieren gesetzt und beim deaktivieren gestoppt.

Parameter

* `state` - der geänderte Zustand.

`public` **void TeamValueChanged**( )

Wird aufgerufen, wenn sich die Teamzugehörigkeit eines Spielers ändert.

Parameter

* `team` -

`public` **void UserNameChanged**( )

wird aufgerufen, wenn sich das Pseudonym der Nutzers geändert hat

Parameter

* `userName` - neues Pseudonym

`public` **void WiretapLevelChanged**( )

Wird aufgerufen, wenn das WiretapLevel neu gesetzt wurd.

Parameter

* `newValue` - Der neues Fähigkeitenwert.

F.26 Namespace Server.Userinterface

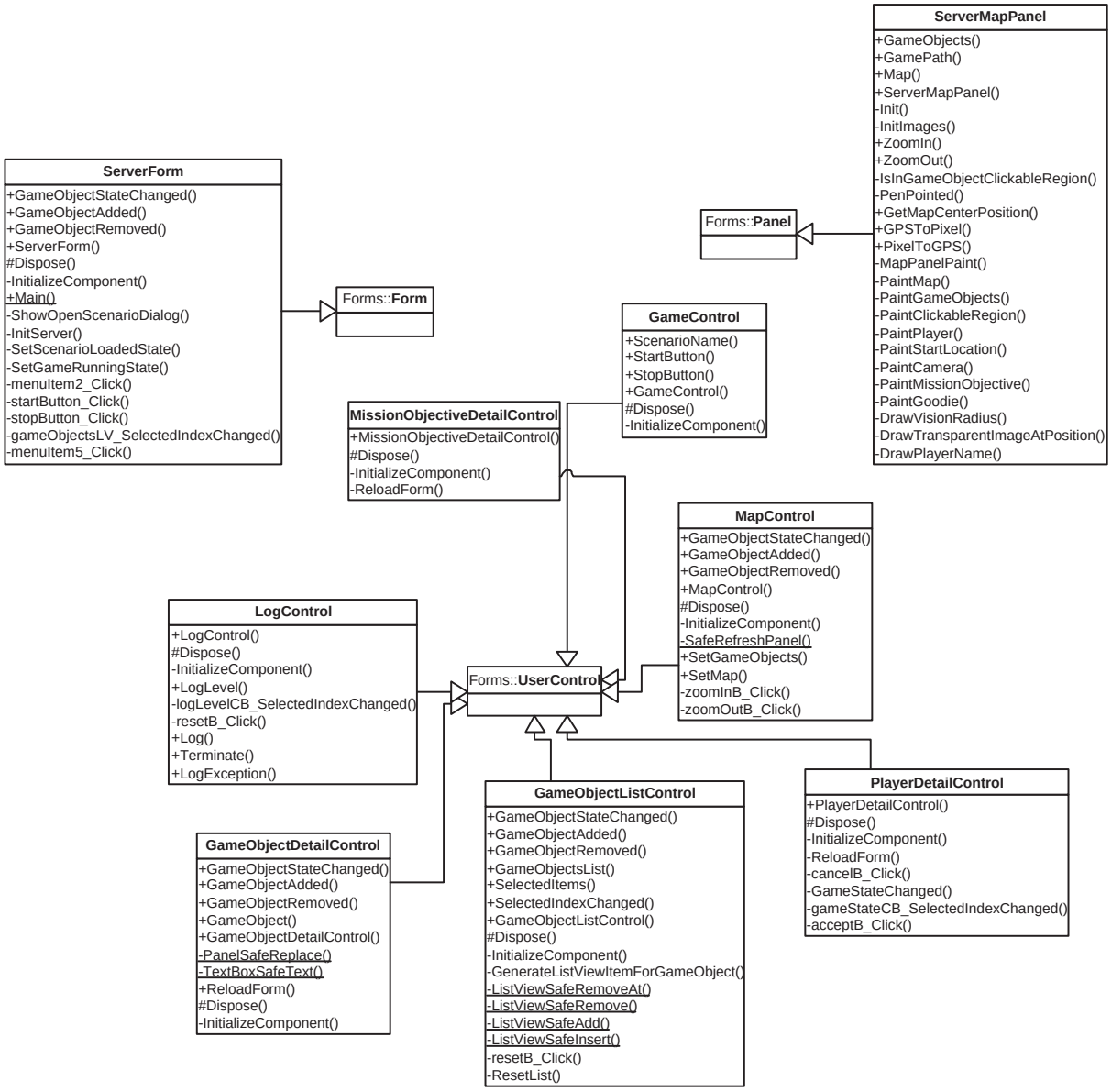


Abbildung F.26: UML Struktur des Namensraum Server.Userinterface

F.26.1 Klasse GameControl

GameControl ist ein Gui-Element zur Steuerung eines Szenarios.

Konstruktoren

`public` **GameControl**( )

Erzeugt ein neues GameControl-Element.

F.26.2 Klasse GameObjectDetailControl

Das GameObjectDetailControl Gui-Element zeigt Details von Spiel-Objekten an.

Konstruktoren

`public` **GameObjectDetailControl**( )

Erzeugt ein neues GameObjectDetailControl.

Methoden

`public` **void GameObjectAdded**( )

wird aufgerufen, wenn ein Spielobjekt hinzugefügt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectRemoved**( )

wird aufgerufen, wenn ein Spielobjekt entfernt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectStateChanged**( )

wird aufgerufen, wenn sich der Zustand eines Spielobjekts geändert hat.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void ReloadForm**( )

Zeigt das GameObjectDetailForm erneut an.

F.26.3 Klasse GameObjectListControl

GameObjectListControl ist ein Gui-Element zur Anzeige einer Liste von Spielobjekten.

Konstruktoren

`public` **GameObjectListControl**( )

Erzeugt ein neues GameObjectListControl

Methoden

`public` **void GameObjectAdded**( )

wird aufgerufen, wenn ein Spielobjekt hinzugefügt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectRemoved**( )

wird aufgerufen, wenn ein Spielobjekt entfernt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectStateChanged**( )

wird aufgerufen, wenn sich der Zustand eines Spielobjekts geändert hat.

Parameter

* `gameObject` - das betroffene Spielobjekt

F.26.4 Klasse LogControl

LogControl ist ein Gui-Element zur Anzeige von Log-Ausgaben.

Konstruktoren

`public` **LogControl**( )

Erzeugt ein neues Log-Control.

Methoden

`public` **void Log**( )

Gibt eine Nachricht über den Logger aus, wenn der angegebene Log-Level mindestens dem Log-Level dieses Loggers entspricht.

Parameter

* `msg` - auszugebende Nachricht
* `logLevel` - Log-Level der Nachricht

`public` **void LogException**( )

Gibt eine Ausnahme über den Logger aus, wenn der angegebene Log-Level mindestens dem Log-Level dieses Loggers entspricht.

Parameter

* `source` - Objekt in dem die Ausnahme aufgetreten ist
* `ex` - auszugebende Ausnahme
* `logLevel` - Log-Level der Ausnahme

`public` **void Terminate**( )

Gibt eventuell allokierte Ressourcen frei.

F.26.5 Klasse MapControl

MapControl ist ein Gui-Element zur Anzeige einer zoombaren Karte zu einem Szenario.

Konstruktoren

`public` **MapControl**( )

Erzeugt ein neues MapControl-Objekt.

Methoden

`public` **void GameObjectAdded**( )

wird aufgerufen, wenn ein Spielobjekt hinzugefügt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectRemoved**( )

wird aufgerufen, wenn ein Spielobjekt entfernt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectStateChanged**( )

wird aufgerufen, wenn sich der Zustand eines Spielobjekts geändert hat.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void SetGameObjects**( )

Mit dieser Methode wird die Liste der auf der Karte anzuzeigenden Spielobjekte gesetzt.

Parameter

* `gameObjects` - ArrayList der Spielobjekte

`public` **void SetMap**( )

Mit dieser Methode wird die anzuzeigende Karte gesetzt.

Parameter

* `map` - die Karte

F.26.6 Klasse MissionObjectiveDetailControl

MissionObjectiveDetailControl ist ein Gui-Element zur Anzeige von Missionszieldetails.

Konstruktoren

`public` **MissionObjectiveDetailControl**( )

Erzeugt ein neues MissionObjectiveDetailControl für das angegebene Missionsziel.

Parameter

* `missionObjective` - das Missionsziel

F.26.7 Klasse OtherDetailControl

OtherDetailControl ist ein Gui-Element zur Anzeige von beliebiegen Spielobjekten.

Konstruktoren

`public` **OtherDetailControl**( )

Erzeugt ein neues MissionObjectiveDetailControl für das angegebene Missionsziel.

Parameter

* `gameObject` - das Missionsziel

F.26.8 Klasse PlayerDetailControl

PlayerDetailControl ist ein Gui-Element zur Anzeige von Details eines Spielerobjekts.

Konstruktoren

`public` **PlayerDetailControl**( )

Erzeugt ein neues PlayerDetailControl für den angegebenen Spieler.

Parameter

* `player` - der anzuzeigende Spieler

F.26.9 Klasse ServerForm

ServerForm stellt eine graphische Oberfläche zur Steuerung eines NABB Spielservers bereit. Der Benutzer kann Szenarios laden und starten, erhält eine Übersicht über die im Spiel befindlichen Spielobjekte. Ausserdem wird die Karte mit allen Spielobjekten angezeigt.

Konstruktoren

`public` **ServerForm**( )

Erzeugt ein neues ServerForm-Objekt.

Methoden

`public` **void GameObjectAdded**( )

wird aufgerufen, wenn ein Spielobjekt hinzugefügt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectRemoved**( )

wird aufgerufen, wenn ein Spielobjekt entfernt wurde.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void GameObjectStateChanged**( )

wird aufgerufen, wenn sich der Zustand eines Spielobjekts geändert hat.

Parameter

* `gameObject` - das betroffene Spielobjekt

`public` **void Main**( )

Die Main-Methode zum starten des Servers. Die Server-GUI wird dargestellt.

F.26.10 Klasse ServerMapPanel

Das MapPanel definiert die Kartenansicht für die Server-Gui. Auf der Karte werden alle Spielobjekte, wie Spieler, Goodies, Kameras o.ä. dargestellt.

Konstruktoren

`public` **ServerMapPanel**( )

Ezeugt ein neues Map Panel.

Parameter

* `map` -

Methoden

`public` **Common.Logic.Pois.QualifiedPosition GetMapCenterPosition**( )

Liefert die GPS-Position des aktuellen Mittelpunkts des Kartenausschnitts.

`public` **System.Drawing.Point GPSToPixel**( )

Rechnet die übergebenen GPS-Koordinaten eines Objekts auf einen Punkt auf der Karte um. Die X-Koordinate des Objekts ist seine aktuelle Latitude minus die des Kartennullpunktes geteilt durch das Verhältnis von Pixeln pro GPS-Minute. Die Y-Koordinate errechnet sich analog.

Parameter

* `position` - Position eines Objektes, das auf der Karte dargestellt werden soll

`public` **Common.Logic.Pois.QualifiedPosition PixelToGPS**( )

Rechnnet einen Punkt auf der Karte in eine GPS Position um.

Parameter

* `x` - Entfernung des Punkts vom linken Kartenrand in Pixeln
* `y` - Entfernung des Punkts vom oberen Kartenrand in Pixeln

`public` **void ZoomIn**( )

Diese Methode wird aufgerufen, um die Karte größer darzustellen.

`public` **void ZoomOut**( )

Diese Methode wird aufgerufen, um die Karte kleiner darzustellen.

F.27 Namespace System.Runtime.CompilerServices

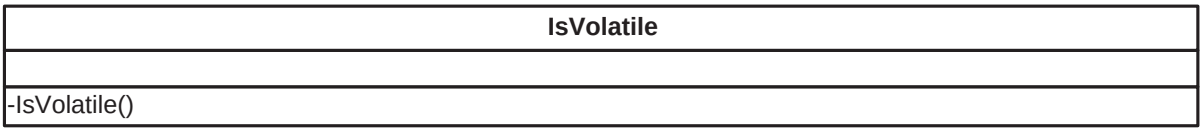


Abbildung F.27: UML Struktur des Namensraum System.Runtime.CompilerServices

F.27.1 Klasse IsVolatile

Siehe Fehlerbeschreibung unter folgender URL: http://blog.opennetcf.org/ncowburn/default,date,2004-11-16.aspx

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AK02] Alfred Kobsa, Josef Fink: *User Modeling for Personalized City Tours*. 2002.

[AKD99] A. K. Dey, G. D. Abowd: *Towards a Better Understanding of Context and Context-Awareness*. 1999.

[ALB+04] Africano, Diana, Peter Lundholm, Sara Berg, Frederik Nilbrink, Kent Lindbergh und Anna Persson: *Designing Tangible Interfaces for Children's Collaboration*. In: *CHI '04: Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Inc., April 2004.

[AS01a] Albrecht Schmidt, Hans-Werner Gellersen: *Modell, Architektur und Plattform für Informationssysteme mit Kontextbezug*. 2001.

[AS01b] Albrecht Schmidt, Hans-Werner Gellersen: *Modell, Architektur und Plattform für Informationssysteme mit Kontextbezug*. 2001.

[Bau97] Baumgartner, P.: *Evaluation vernetzten Lernens*. In: *Virtueller Campus: Forschung und Entwicklung für neues Lehren und Lernen*, Seiten 131–146. Waxmann Verlag, 1997.

[BB00] Buckler, Frank und Holger Buxel: Mobile Commerce Report. `http://www.profit-station.de/star/study2/sty02a_g.htm`, 2000. Abgerufen am: 25.10.2004.

[BD02] Bradley, N. A. und M. D. Dunlop: *Investigating context-aware clues to assist navigation for visually impaired people*. In: *Proccedings of Workshop on Building Bridges: Interdisciplinary Context-Sensitive Computing, University of Glasgow*, 2002.

[BD03a] Bradley, N. und M. Dunlop: *Towards a multidisciplinary user-centric design framework for context-aware applications*. In Proceedings of 1st UKUbiNet Workshop, September 2003.

[BD03b] Bradley, N. A. und M. D. Dunlop: *A pathway to independence: Wayfinding Systems which adapt to adapt a visually impaired person's context*. 2003.

[BD04] Bradley, N. A. und M. D. Dunlop: *Investigating design issues of context-aware mobile guides for people with visual impairments*. In: SchmidtBelz, B. und K. Cheverst (Herausgeber): *Proceedings of workshop on HCI in Mobile Guides at MobileHCI04*, September 2004.

[Bec99] Beck, Kent: *Extreme Programming Explained*. Addison-Wesley, 1999.

[Ber02] Beringer, N.: *PROMISE - A Procedure for Multimodal Interactive System Evaluation*. 2002.

[Ber03] Berkley, Jeffrey J.: *Whitepaper: Haptic Devices*, 2003.

[BFJ+] Buchanan, George, Sarah Farrant, Matt Jones, Harold Thimbleby, Gary Marsden und Michael Pazzani: *Improving Mobile Internet Usability.* Technischer Bericht, Middlesex University and University of Cape Town and AdaptiveInfo, `http://www10.org/cdrom/papers/pdf/p230.pdf`.

[BGG02] *Gesetz zur Gleichstellung behinderter Menschen.* `http://behindertenbeauftragter.de/gesetzgebung/behindertengleichstellun%gsgesetz/gesetzestext`, 2002.

[BL03] Birkelbach, Joerg und Markus Lemcke: *Chancengleichheit. Behinderte in der digitalen Welt.* c't Magazin für Computertechnik, Seiten 88–91, 4/2003. online erreichbar unter `http://www.heise.de/ct/03/04/088/default.shtml`.

[Bod92] Bodendorf, Freimut: *Benutzermodelle - ein konzeptioneller Überblick.* 1992.

[Bro97] Brockhaus: *Die Enzyklopädie in 24 Bänden: Band 6.* 1997.

[Bun03] Bundesamt, Statistisches: *Schwerbehinderte in Deutschland.* `http://www.destatis.de/`, Februar 2003.

[Car83] Card, S.: *The psychology of human computer interaction.* 1983.

[CK00] Chen, Guanling und David Kotz: *A Survey of Context-Aware Mobile Computing Research.* Technischer Bericht TR2000-381, November 2000.

[CM01] Christian Mueller, et al.: *Recognizing Time Pressure and Cognitive Load on teh Basis of Speech: An Experimental Study.* 2001.

[CM04] Cohen, Philip R. und David R. McGee: *Tangible Multimodal Interfaces for Safety-Critical Applications.* Communications of the ACM, 47(1), January 2004.

[COJ+02] Chang, Angela, Sile O'Modhrain, Rob Jacob, Eric Gunther und Hiroshi Ishii: *ComTouch: Design of a Vibrotactile Communication Device.* In: *DIS02, London.* Association for Computing Machinery, Inc., 2002.

[Cra03] Crawford, Chris: *The art of interactive design.* No Starch Press, San francisco, 2003.

[DCSW85] Dirlich, G., C.Freska, U. Schwatlo und K. Wimmer: *Kognitive Aspekte der Mensch-Computer-Interaktion.* Springer Verlag, 1985.

[DE998] *Definitionen von Evaluation.* http://www-is.informatik.uni-oldenburg.de/~dibo/teaching/mm/buch/node72.html, 1998. Retrieved 2004-10-23.

[DFGB93] Dix, Alan, Janet Finlay, Abowd Gregory und Russel Beale: *Human-Computer-Interaction.* Prentice Hall International, 1993.

[e.V03] e.V., Bankakademie: *Kompendium Management in Banking and Finance.* Bankakademie Verlag GmbH, 2003.

[EVA04] *Definitionen für Begriffe rund um Evaluation.* http://www.evaluieren.de/evaluat.ion/definiti.htm, 2004. Retrieved 2004-10-23.

[FB97] Fitzmaurice, George W. und William Buxton: *An Empirical Evaluation of Graspable User Interfaces: towards specialized, spacemultiplexed input.* In: *CHI'97: Proceedings of the ACM Conference on Human Factors in Computing Systems*, Seiten 43–50, 1997.

[fbusS04] Studierende, Zentrum für blinde und sehbehinderte: *Zugänglichkeit von Linux für Blinde.* `http://www.fh-giessen.de/bliz/projekte/linux-zugaenglichkeit.php`, 2004.

[fG04] GbR fluSoft: *Braillezeile InfoDot 40 pro.* `http://www.flusoft.de/braille/id40pro/index.html`, 2004.

[fIA04] IT Accommodation, Center for: *Section 508.* `http://www.section508.gov/index.cfm?FuseAction=Content\&ID=3`, 2004.

[FIB95] Fitzmaurice, George W., Hiroshi Ishii und William Buxton: *Bricks: Laying the Foundations for Graspable User Interfaces.* In: *CHI'95: Proceedings of the ACM Conference on Human Factors in Computing Systems*, May 1995.

[GHJV94] Gamma, Erich, Richard Helm, Ralph Johnson und John Vlissides: *Design Pattern: Elements of Reusable Object-Oriented Software.* Addison- Wesley, 1994.

[Hae04] Haenel, Matthias: *Chancen und Probleme Sehgeschädigter bei der Nutzung moderner Medien.* `http://www.matthias-haenel.de/`, 2004.

[Han02] Hanisch, Armin: *GoTo C#.* Addison-Wesley, 2002.

[HB98] Hugh Beyer, Karen Holtzblatt: *Contextual Design.* Academic Press, Morgan Kaufmann Publishers, 1998.

[HBC+96] Hewett, Thomas T., Ronald Baecker, Stuart Card, Tom Carey, Jean Gasen, Marilyn Mantei, Gary Perlman, Gary Strong und William Verplank: *Curricula for Human-Computer Interaction.* Technischer Bericht, ACM Special Interest Group on Computer-Human Interaction (SIGCHI) Curriculum Development Group, 1996.

[Hed02] Hedicke, Volkmar: *Mensch-Maschine-Systemtechnik.* Symposion Publishing GmbH, 2002.

[Hei] Heinecke, Andreas M.: Skript zur Vorlesung IS1 Interaktive Systeme 1 „Mensch-Computer-Interaktion". `http://www.informatik.fh-gelsenkirchen.de/paul/fhge/lehre/ws0203_isy1/i%s1-1.pdf`. Abgerufen am 10.2004.

[Hel02] Hellbusch, Jan Eric: *Microsoft Active Accessibility.* `http://www.barrierefreies-webdesign.de/knowhow/msaa/einfuehrung.php`, 2002.

[HK03] Ho, Shuk Ying und Sai Ho Kwok: *The attraction of personalized service for users in mobile commerce: an empirical study.* SIGecom Exch., 3(4):10–18, 2003.

[HND00] Hofmeister, Christine, Robert Nord und Soni Dilip: *Applied Software Architecture.* Addison- Wesley, 2000.

[Hoh02] Hohmann, Sandra: *Mensch-Maschine-Interface, Studien zu einer Theorie der Mensch-Computer-Interaktion.* Duisburg, 2002.

[Hue04] Huenermund, Holger: *Ratgeber Behinderung.* `http://www.behinderung.org/`, 2004.

[Huf04] Huffstadt, Karsten: Human-Computer-Interaction, 2004.

[Hut95] Hutchins, Edwin: *Cognition in the Wild*. MIT Press, Cambrige, 1995.

[HZ97] Haas, Rolf und Holger Ziegelbauer: Personenbezug. `http://www.heise.de/ix/artikel/1997/09/044/`, 1997. Abgerufen am: 25.10.2004.

[Inc04a] Incobs(Hrsg.): *Screenreader - Stand der Technik*. `http://www.incobs.de/hilfsmittelinfos/screenreader/sctechnik.php`, 2004.

[Inc04b] Incobs(Hrsg.): *Zusammenfassung der Testergebnisse*. `http://incobs.dias.de/produkttests/screenreader/zusammenfassung.php`, 2004.

[ISO04] *Bluffers' Guide to ISO 9241*. http://www.userfocus.co.uk/resources/iso9241/, 2004. Retrieved 2004-10-23.

[IU97] Ishii, Hiroshi und Brygg Ullmer: *Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms*. In: *CHI'97: Proceedings of the ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Inc., March 1997.

[KCH+] Klockar, Tomas, David A. Carr, Anna Hedman, Tomas Johansson und Fredrik Bengtsson: *Usability of Mobile Phones*. Technischer Bericht, Department of Computer Science and Electrical Engineering Luleå University of Technology, `http://www.sm.luth.se/~david/papers/MobileUsabilityHFT03.pdf`.

[KKB04] Klante, Palle, Jens Krösche und Susanne Boll: *AccesSights - A Multimodal Location-Aware Mobile Tourist Information System*. In: *ICCHP'04: Computers Helping People with Special Needs*. Springer Verlag, July 2004.

[Kla99] Klante, P.: *Evaluation von Web-basierter Lernsoftware*. 1999.

[Kob00] Kobsa, Alfred: *Generic User Modeling Systems*. 2000.

[KR] Ketola, Pekka und Mika Röykkee: *The Three Facets of Usability In Mobile Handsets*. Technischer Bericht, Nokia, `http://www.cs.colorado.edu/~palen/chi_workshop/papers/ketola.pdf`.

[Kum05] Kumar, S. Senthil: What's up with BeginInvoke? `http://www.codeproject.com/csharp/begininvoke.asp`, 2005. Abgerufen am: 17.09.2005.

[Man01] Manhart, Klaus: NetworkWorld / Enterprise Application / Personalisierung ist ein Muss. `http://www.tecchannel.de/netzwerk/networkworld/enterpriseapplication/11%43/index.html`, 2001. Abgerufen am: 25.10.2004.

[Mat03] Mattern, Friedemann: *Total vernetzt - Szenarien einer Informatisierten Welt*. Springer-Verlag Berlin Heidelberg, 2003.

[May99] Mayhew, Debroah J.: *The Usability Engineering Lifecycle*. Academic Press, Morgan Kaufmann Publishers, 1999.

[MD03] Mazalek, Ali und Glorianna Davenport: *A Tangible Platform for Documenting Experiences and Sharing Multimedia Stories*. In: *Proceedings of the 1st ACM WS on Experiential Telepresence (ETP 03)*. Association for Computing Machinery, Inc., November 2003.

[ME01] Maria Ebling, Guerney Hunt, Hui Lei: *Issues for Context Services for Pervasive Computing*, 2001.

[Men03] Mensch, Aktion: *Einfach für Alle.* `http://www.einfach-fuer-alle.de/artikel/gestaltung-hilft/`, 2003.

[Mil56] Miller, G.A.: *The Magical Number Seven, Plus or Minus Two.* The Psychological Review, 1956.

[Möl03] Möller, Jan: Häufig gestellte Fragen zum Datenschutz mit P3P. `http://www.datenschutzzentrum.de/faq/p3p.htm`, 2003. Abgerufen am: 25.10.2004.

[MPR00] Manber, Udi, Ash Patel und John Robison: *Experience with personalization of Yahoo!* Commun. ACM, 43(8):35–39, 2000.

[Nie93] Nielson, Jakob: *Usability Engineering.* Academic Press Limited, 1993.

[Nie96] Nielsen, Jakob: *Accessible Design for Users With Disabilities.* `http://www.useit.com/alertbox/9610.html`, 1996.

[Nie00] Nielson, Jakob: *Designing Web Usability.* New Riders Publishing, 2000.

[Nie03] Nielsen, Jakob: *Usability 101: Introduction to Usability.* `http://www.useit.com/alertbox/20030825.html`, 2003.

[Nor88] Norman, Donald A.: *The psychology of everyday things.* 1988.

[OEO94] Oberquelle, H, E. Eberleh und R. Oppermann: *Einführung in die Software-Ergonomie. Gestaltung graphisch-interaktiver Systeme: Prinzipien, Werkzeuge, Lösungen.* de Gruyter, Berlin, 1994.

[PBG04] Posch, Torsten, Klaus Birken und Michael Gerdom: *Basiswissen Softwarearchitekturen.* dpunkt.verlag GmbH, 2004.

[Pet00] Petzhold, Charles: *Windowsprogrammierung 5.Auflage.* Microsoft Press Deutschland, 2000.

[Pre94] Preece, J.: *Human-computer interaction.* Addison-Wesley, 1994.

[Pre99a] Preim, Bernhard: *Entwicklung interaktiver Systeme: Grundlagen, Fallbeispiele und innovative Anwendungsfelder.* Springer Verlag, Bonn, 1999.

[Pre99b] Preim, Dr. B.: *Entwicklung interaktiver Systeme.* Springer-Verlag, 1999.

[Rei94] Reiterer, Oppermann &: *Software-ergonomische Evaluation.* In: *Einführung in die Software-Ergonomie*, Seiten 335–371. de Gruyter, 1994.

[Rei03] Reichenbacher, T.: *Adaptive Methods For Mobile Cartography.* 2003.

[Ric81] Rich, Elaine: *Users are individuals: individualizing user models.* 1981.

[Rie00a] Riecken, Doug: *Introduction: personalized views of personalization.* Commun. ACM, 43(8):26–28, 2000.

[Rie00b] Riecken, Doug: *Personalized communication networks.* Commun. ACM, 43(8):41–42, 2000.

[RMI04] Ryokai, Kimiko, Stefan Marti und Hiroshi Ishii: *I/O Brush: Drawing with Everyday Objects as Ink.* In: *CHI'04: Proceedings of the ACM Conference on Human Factors in Computing Systems.* Association for Computing Machinery, Inc., April 2004.

[Roy70] Royce, Winston W.: *Managing the Development of Large Software Systems, Concepts and Techniques*. In: *Proceedings of IEEE WESCON*, 1970.

[RPI04] Raffle, Hayes Solos, Amanda J. Parkes und Hiroshi Ishii: *Topobo: A Constructive Assembly System with Kinetic Memory*. In: *CHI'04: Proceedings of the ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Inc., April 2004.

[Sch91] Schmauks, Dagmar: *Deixis in der Mensch-Maschine-Interaktion*. Niemeyer Verlag, 1991.

[See03] Seeberger, Klaus: *Barrierefreies Internet*. `http://www.barrierefreiesinternet.de/`, 2003.

[Shn98] Shneiderman, B.: *Designing the User interface*. Addison-Wesley, 1998.

[SR03] Slatin, John M. und Sharron Rush. Addison-Wesley, 2003.

[Thi90] Thimbleby, H.: *User Interface Design*. ACM Press Frontier Series, 1990.

[TJK00] Timpe, K.-P., T. Jürgensohn und H. Kolrep: *Mensch-Maschine-Systemtechnik*. Symposion publishing, 2000.

[TJK02] Timpe, Klaus-Peter, Thomas Jürgensohn und Harald Kolrep: *Mensch-Maschine-Systemtechnik*. Symposion, 2002.

[UI99a] Ullmer, Brygg und Hiroshi Ishii: *mediaBlocks: Tangible Interfaces for Online Media*. In: *CHI'99: Proceedings of the ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Inc., May 1999.

[UI99b] Underkoffler, John und Hiroshi Ishii: *Urp: A Luminous-Tangible Workbench for Urban Planning and Design*. In: *CHI'99: Proceedings of the ACM Conference on Human Factors in Computing Systems*. Association for Computing Machinery, Inc., May 1999.

[UI00] Ullmer, Brygg und Hiroshi Ishii: *Emerging Frameworks for Tangible User Interfaces*. IBM Systems Journal, 39(3), 2000.

[UI01] Ullmer, Brygg und Hiroshi Ishii: *Emerging Frameworks for Tangible User Interfaces*, Seiten 597–601. Addison-Wesley, 2001.

[Ull02] Ullmer, Brygg Anders: *Tangible Interfaces for Manipulating Aggregates of Digital Information*. Doktorarbeit, Massachusetts Institute of Technology, 2002.

[Unbnt] Unbekannt: GUI application hangs when using non-[OneWay]-Events. `http://www.thinktecture.com/Resources/RemotingFAQ/HANDLING_EVENTS_HANGS%_APPLICATION.html`, Unbekannt. Abgerufen am: 25.10.2004.

[urla] Authoring for Small-Screen Rendering (SSR). `http://www.opera.com/products/mobile/dev/`. Abgerufen am 24.10.2004.

[urlb] Benutzerfreundlichkeit.de. `http://www.benutzerfreundlichkeit.de/index.php?id=7`. Abgerufen am 24.10.2004.

[urlc] Contextual Design: How We Design. `http://www.incent.com/cd/cdp.html`. Abgerufen am 24.10.2004.

[urld] Gestalt Principles of Perception. http://www.usask.ca/education/coursework/skaalid/theory/gestalt/gestalt%.htm. Abgerufen am 24.10.2004.

[urle] International Electrotechnical Commission. http://www.iec.ch/. Abgerufen am 24.10.2004.

[urlf] ISO Online. http://www.iso.org/iso/en/ISOOnline.openerpage. Abgerufen am 24.10.2004.

[urlg] Japanese Products Map the Mobile Road Ahead. http://www.useit.com/alertbox/20010429.html. Abgerufen am 24.10.2004.

[urlh] Ten Usability Heuristics. http://www.useit.com/papers/heuristic/heuristic_list.html. Abgerufen am 24.10.2004.

[urli] The Usability Methods Toolbox. http://jthom.best.vwh.net/usability/. Abgerufen am 24.10.2004.

[urlj] Timo Jokela. http://www.tol.oulu.fi/~tjokela/publications.htm. Abgerufen am 24.10.2004.

[urlk] Usability First. http://www.usabilityfirst.com/intro/index.txl. Abgerufen am 24.10.2004.

[urll] User centred design standards. http://www.usability.serco.com/trump/resources/standards.htm. Abgerufen am 24.10.2004.

[urlm] Web Usability Standards. http://www.userfocus.co.uk/articles/ISO23973.html. Abgerufen am 24.10.2004.

[W3Ca] W3C: Composite Capabilities/Preferences Profiles. http://www.w3.org/Mobile/CCPP. Abgerufen am: 25.10.2004.

[W3Cb] W3C: Platform for Privacy preferences (P3P) project. http://www.w3.org/P3P. Abgerufen am: 25.10.2004.

[W3Cc] W3C: XML Encryption. http://www.w3.org/encryption. Abgerufen am: 25.10.2004.

[Wah01] Wahlster, Wolfgang et al.: *Real: Ein ressourcenadaptierendes mobiles Navigationssystem.* 2001.

[Wan93] Wandermacher, H.: *Softwareergonomie.* Berlin, 1993.

[WBHK02] Wagner, Matthias, Wolf-Tilo Balke, Robert Hirschfeld und Wolfgang Kellerer: *A Roadmap to Advanced Personalization of Mobile Services.* Proc. Federated Conferences CoopIS/ODBASE/DOA, 2002.

[Wei91] Weiser, Mark: *The Computer for the 21st Century.* Scientific American, 265(3):94–104, 1991.

[Wei98] Weiser, Mark: *The Future of Ubiquitous Computing on Campus.* Communications of the ACM, 41(1), January 1998.

[wik] Wikipedia, die freie Enzyklopädie. http://de.wikipedia.org/wiki/Hauptseite. Abgerufen am 10.2004.

[wik04] *Barrierefreies Internet*. `http://de.wikipedia.org/wiki/Accessibility`, 2004.

[Win03] WINKLER, SANDRA: *Diplomarbeit: Gibt es ein barrierefreies Web?*, Juli 2003. online erreichbar unter `http://www.sandra-winkler.de/barrierefrei/`.