Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft

Wissenschaftliche Berichte
FZKA 7176

Konzeption eines Emotionalen Pädagogischen Agenten und Realisierung von ausgewählten Komponenten

P. Real, C. Düpmeier, M. Ruchter

Institut für Angewandte Informatik

November 2005

Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft
Wissenschaftliche Berichte
FZKA 7176

Konzeption eines Emotionalen Pädagogischen Agenten und Realisierung von ausgewählten Komponenten

Peter Real[*)]**, Clemens Düpmeier, Markus Ruchter**

Institut für Angewandte Informatik

[*)] von der Universität Karlsruhe (TH), Fachbereich Informatik, genehmigte Diplomarbeit

Forschungszentrum Karlsruhe GmbH, Karlsruhe
2005

Impressum der Print-Ausgabe:

Als Manuskript gedruckt

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

Mitglied der Hermann von Helmholtz-Gemeinschaft
Deutscher Forschungszentren (HGF)

ISSN 0947-8620

urn:nbn:de:0005-071761

Kurzfassung:

Mobile Applikationen wie mobile Naturführer werden zunehmend in unterschiedlichen Anwendungsbereichen eingesetzt. Die Anwender dieser Geräte sind häufig Laien, die das Gerät in der Freizeit kurzzeitig einsetzen wollen. Zu ihrer Unterstützung wird eine neuartige intuitive Interaktionstechnik benötigt. Animierte Interface Agenten bieten eine natürliche Unterstützung wie der Benutzer sie sonst von einem menschlichen Führer erwarten würde. Dieser Führer soll sowohl den Anwender bei seiner Tour durch ein Naturgebiet unterstützen, als auch aus umweltpädagogischer Sicht die Bindung zwischen dem Anwender und den betrachteten Naturphänomenen verstärken. Eine kontext-basierte Architektur und ausgewählte Implementierungsaspekte für einen SVG basierten pädagogischen Agenten werden in dieser Arbeit präsentiert. Der Agent ist in der Lage, dem Benutzer abhängig vom aktuellen Kontext angepasste Tourinhalte zu präsentieren. Dafür stehen verschiedene multimodale Kommunikationsformen zur Verfügung, sei es über Sprache, Gestik, Mimik oder eher traditionell über Text und Bilder. Auf den Konzepten aufbauend wurde eine Implementierung einer grafischen Darstellung des animierten Agenten unter SVG durchgeführt.

Design of an emotional pedagogical agent and realisation of selected components

Abstract:

Today mobile applications like mobile nature guide systems are more and more frequently used in different areas of application. The users of these systems are mostly non-experts, who want to use the system temporarily during their leisure time. This raises a new demand for intuitive interaction methods that give assistance to the user. Animated interface agents can offer a natural assistance that is traditionally expected from a human nature guide. This guide helps the user to take a tour through a nature reserve, but it can also enhance the binding between the user and the surrounding natural environment. Based on an extensive review of the literature this thesis presents a context-based architecture and selected implementation aspects for a SVG-based pedagogical agent. Depending on the current context the agent is able to present tour content that is adapted for the current situation. The presentation uses several different communication media, speech, gesture, mimic or traditional forms like text and picture. The graphical representation uses the SVG format. Based on these concepts a prototypic Pocket PC application of the agent was implemented.

INHALTSVERZEICHNIS

ABBILDUNGSVERZEICHNIS III

TABELLENVERZEICHNIS IV

KAPITEL 1: EINLEITUNG 1

KAPITEL 2: STAND DER FORSCHUNG UND WICHTIGE BEGRIFFLICHKEITEN 6

2.1 AGENTEN 7
2.1.1 Agent und Intelligenter Agent 7
2.1.2 Präsentations Agent 9
2.1.3 Interface Agent 10
2.1.4 Pädagogischer Agent 10
2.1.5 Conversational Agent 11
2.2 MULTIMODALE KOMMUNIKATION 12
2.2.1 Gesten 13
2.2.2 Mimik 15
2.2.3 Sprache 15
2.3 EMOTIONEN 21
2.4 BESCHREIBUNGS- UND SKRIPTSPRACHEN 25
2.5 CHARAKTER ANIMATION 27
2.5.1 2D-Animation 28
2.5.2 Koordination von Animationen 31
2.5.3 Layered Modell 31
2.5.4 Skelett-basiertes Modell 32
2.5.5 Animationsübergänge zwischen Animationsabschnitten 32
2.5.6 Synchronisation von Animation und Sprache 33
2.6 KONTEXT 34
2.7 VERHALTENSPLANUNG 43
2.7.1 Rekursive hierarchische Planung 45
2.7.2 Kontext-sensitive Planung der Animation 46
2.7.3 BEAT 47
2.8 ZUSAMMENFASSUNG UND VERGLEICH 48

KAPITEL 3: EXISTIERENDE SYSTEME 50

3.1 LAURA 50
3.2 GINA 51
3.3 HERMAN-THE-BUG 52
3.4 SMARTKOM 53
3.5 WEITERE LIFE-LIKE CHARACTERS 54
3.6 AGENT-FRAMEWORKS 55

KAPITEL 4: SYSTEMKONZEPT 58

4.1 USE-CASES 58
4.2 SYSTEM ARCHITEKTUR 67
4.3 DIRECTOR 72
4.4 EDITOR UND VORVERARBEITUNG DER ANIMATIONSERSTELLUNG 77
4.5 KONTEXT-MANAGER 79
4.6 SZENE-PLAYER 82

KAPITEL 5: IMPLEMENTIERUNG 88

5.1 IMPLEMENTIERUNGSUMGEBUNG 88
5.2 DIRECTOR 90
5.3 PUPPET UND ANIMATION 91
5.4 ANIMATIONS-COMPILER 94
5.5 IMPLEMENTIERUNGSPROBLEME 96

KAPITEL 6: FAZIT UND AUSBLICK 97

6.1 FAZIT 97
6.2 AUSBLICK 97

LITERATURVERZEICHNIS 100

ABKÜRZUNGEN 105

LINKS 106

Abbildungsverzeichnis

Abbildung 1: Der MobiNaf Guide 1
Abbildung 2: Mobile Nature Guide 4
Abbildung 3: Agenten Architektur 8
Abbildung 4: Life-like character, Wissen, Geist, Körper 11
Abbildung 5: Wahrnehmung und Erzeugung von menschlicher Kommunikation 12
Abbildung 6: Iconic gesture 14
Abbildung 7: Auf das wesentliche reduzierte Mimiken 15
Abbildung 8: Beispiel Ablauf für Erzeugung von Mundanimation und Sprachausgabe 16
Abbildung 9: Circumplex-Model 22
Abbildung 10: Funktion für Emotionsintensität 23
Abbildung 11: Traurigkeit und Freude 23
Abbildung 12: Emotionsintensitätsfunktion 24
Abbildung 13: CML Verwendung 26
Abbildung 14: Workflow Animation 29
Abbildung 15: Layered Modell 32
Abbildung 16: Bewegungsgraphen 33
Abbildung 17: Allgemeiner Kontext und relevanter Kontext 36
Abbildung 18: Kontexthierachie 39
Abbildung 19: Systemübersicht mit Präsentations-,Verhaltensplanung und Player 44
Abbildung 20: Graph mit Präsentationszielen 45
Abbildung 21: Zustandsautomat für Präsentationsteile 46
Abbildung 22: BEAT architecture 48
Abbildung 23: Laura Client-Server Architektur 51
Abbildung 24: Gina 52
Abbildung 25: Architektur von Herman-the-Bug 53
Abbildung 26: Oberfläche für das mobile Szenario 54
Abbildung 27: Rapid-Paper-Prototyping 58
Abbildung 28: Benutzer Use-cases 60
Abbildung 29: Benutzer hat Fragen an den Charakter 61
Abbildung 30: Autor Use-cases 63
Abbildung 31: Quiz präsentieren 64
Abbildung 32: Präsentation von POI-Informationen 65
Abbildung 33: Gesamt-Architektur Übersicht 68
Abbildung 34: Skript-Akt-Szene Struktur 69
Abbildung 35: Basisarchitektur Übersicht 73
Abbildung 36: Schnittstelle zwischen Mediator und Director 74
Abbildung 37: Schichten über der Karte 76
Abbildung 38: Prozess von der Erstellung bis zur Darstellung von Animationen 79
Abbildung 39: Context model für den Mobile Nature Guide 80
Abbildung 40: Kontext-Manager Architektur 81
Abbildung 41: Klassendiagramm Puppet 84
Abbildung 42: Zustandsübergangsdiagramm des Players 85
Abbildung 43: Konzept-Skizze für Skelett-Model 86
Abbildung 44: Bisherige Architektur zur Anzeige der Stationsinformationen 89
Abbildung 45: Klassendiagramm des Directors in der bestehenden Implementierung 90
Abbildung 46: Ranger-Charakter 91
Abbildung 47: Verschiedene Handformen 93
Abbildung 48: Der Ranger präsentiert die Stationsbeschreibung 94

Tabellenverzeichnis

Tabelle 1: Sprach Parameter für verschiedene Emotionen 18
Tabelle 2: Phonme-Viseme Tabelle 19
Tabelle 3: TTS Systeme 20
Tabelle 4: Positive und negative Emotionen 21
Tabelle 5: Übersicht 2D/3D Animationssoftware 30
Tabelle 6: Übersicht von relevanten Kontexten für einen pädagogischen Agenten 40
Tabelle 7: Kontext Widget für die Raumüberwachung 42
Tabelle 8: Übersicht life-like characters 55
Tabelle 9: Frameworks 56
Tabelle 10: Die ersten identifizierten Komponenten: 67

Kapitel 1: Einleitung

Es ist ein schöner Dienstag im März, die Temperatur ist bereits angenehm warm für Anfang Frühling. Michael besucht mit seiner Schulklasse der Unterstufe von der „Walter-Hildebrandt“-Realschule das Naturschutzzentrum Rappenwört. Letztes Jahr hatte er bereits mit seiner Schulklasse eine Exkursion in das Zentrum gemacht. Dieser Tag ist ihm immer noch negativ in Erinnerung. Damals waren sie mit 25 Kindern da und er konnte die Tourführerin nicht verstehen, da er ständig hinter einer Reihe von anderen Kindern stand. Deswegen hat er Bedenken, ob es nicht heute wieder genauso öde wird. Dieses Jahr wurde die Schulklasse eingeladen, da das Zentrum ein neues elektronisches System testen möchte. Die Kinder werden in Gruppen von jeweils drei Schülern aufgeteilt. Jede Gruppe bekommt einen PDA („Persönlicher digitaler Assistent“) zugeteilt. Beim Einschalten des PDA wird Michael von einem animierten gezeichneten Vogel begrüßt. Der Vogel stellt sich vor „Hallo! Mein Name ist MobiNaf. Willkommen an diesem schönen Frühlingstag im Naturschutzzentrum Rappenwört.“ Dabei bewegt sich der Schnabel und Körper. Der Text wird in einer Sprechblase angezeigt (Abbildung 1).

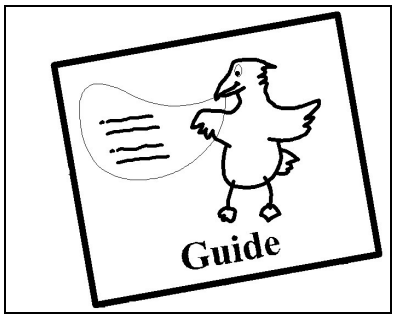

Abbildung 1: Der MobiNaf-Guide

„Bevor wir anfangen das Gebiet zu erkunden, sollten wir uns besser kennen lernen. Bitte sag mir deinen Namen.“ Michael schreibt M-I-C-H-A-E-L auf das Display und drückt dann „Weiter“. „Hallo Michael. Gibt es noch mehr Personen in deiner Gruppe?“ Michael trägt auch noch die Namen der anderen beiden Kinder in das Gerät ein, der Vogel begrüßt beide. „Macht ihr einen Schulausflug?“ Michael kreuzt „Ja“ an. „Wollt ihr, dass wir den Wald um das Naturschutzzentrum herum erkunden?“ Michael markiert noch mal „Ja“. „Ihr habt Glück! Für einen Ausflug ist heute ein phantastischer Tag. Ich kann euch eine Reihe von Touren anbieten. Bitte wählt euch eine aus.“ Es wird eine Reihe von Touren angezeigt: „Schülertour 1: Die Natur erwacht.“, „Schülertour 2: Auf der Jagd nach dem Frühling“ etc. Michael wählt die zweite Tour aus. „Das wird spannend. Dabei kann man viel erleben! Also Michael, Anna, Daniel, ihr müsst mir eine Sache versprechen. In diesem Gebiet leben viele meiner Freunde, von denen wir einige heute kennen lernen werden. Ihr seid Gäste in ihrem Wald, deswegen geht sorgsam mit den Tieren und Pflanzen um, wenn

wir durch den Wald gehen. Wir werden jetzt zur ersten Station gehen. Geht zuerst zum Eingang des Zentrums.“ Am Eingang des Zentrums sind die Kinder etwa 20 m von den anderen Kindern entfernt. Eine Stimme kommt aus dem Gerät „So, jetzt sind wir unter uns. Jetzt kann ich etwas lauter sprechen. Die nächste Station ist nicht mehr weit. Geht weiter zur nächsten Kreuzung. Schaut euch auf dem Weg die Sträucher an. Könnt ihr die gelben Blüten erkennen? Das sind Blüten von Haselnussträuchern.“ Die Kinder laufen bis zur ersten Station. „So, wir haben die erste Station erreicht. Hier gibt es einen Baum mit einem Zwillingsstamm. Kann mir jemand von euch sagen, was das für ein Baum ist?“ Neben dem Photo des Baums wird eine Liste mit mehreren Baumnamen angezeigt: Eiche, Buche, Linde. Anna wählt Linde aus. „Linde ist leider falsch. Auf dem Bild siehst du eine Linde. Schau dir die Rinde an, sie hat eine andere Farbe als der Baum, neben dem wir stehen. Dieser Baum ist eine Buche. Ich werde euch etwas zu seiner Geschichte erzählen …“

Anhand der einleitenden Geschichte soll gezeigt werden, um welche Thematik es in dieser Arbeit geht. Das System, das die Kinder in der Geschichte verwenden, basiert auf einer erweiterten Version eines Mobilen Naturführers, der am Institut für Angewandte Informatik (IAI) des Forschungszentrums Karlsruhe entwickelt wird. Das Konzept des so genannten MobiNaf (Mobiler Naturführer) Systems wurde erstmals in [Ruchter 2003] vorgestellt und von Sobek [Sobek 2004] in einer Basisform weitgehend implementiert. Das System soll Nutzern aus verschiedenen Zielgruppen bei ihrem Erlebnis von Natur und Umwelt in so genannten Erlebnisgebieten als Mobiler Naturführer begleiten. Das Projekt MobiNaf wird vom Umweltministerium (UM), dem Ministerium für Ernährung und Ländlichen Raum (MLR) und der Landesanstalt für Umweltschutz (LfU) Baden-Württemberg sowie der Stadt Karlsruhe unterstützt und im Rahmen des LIFE-Projektes „Lebendige Rheinauen bei Karlsruhe“ von der EU gefördert. Zur Entwicklung und zum Test des Systems existiert eine Kooperation mit dem Naturschutzzentrum Rappenwört, das die Arbeiten inhaltlich begleitet. Das Naturschutzzentrum und seine Umgebung dient als ein konkretes Erlebnisgebiet, in dem das MobiNaf-System mit Gästen des Naturschutzgebietes getestet wird.

Dabei soll die Erfahrung der MobiNaf Nutzer durch auf ihr persönliches Interesse und ihren Erfahrungshintergrund zugeschnittene, ortsbezogene Umweltinformationen bereichert werden. Eines der Hauptziele des Systems ist es, das Umweltbewusstsein der Nutzer zu steigern, indem nicht nur ihr Umweltwissen erweitert wird, sondern auch die Einstellung der Nutzer gegenüber Natur und Umwelt sowie ihre Bereitschaft zum umweltgerechten Handeln verbessert werden. Wie umweltpsychologische Studien gezeigt haben, ist es hierzu wichtig, dass die Nutzer eine emotionale Bindung zur Natur und Umwelt basierend auf konkreten Umwelterfahrungen aufbauen [Schultz 2000]. Hierzu soll der elektronische Naturführer virtuelle Naturinformation mit konkreter Naturerfahrung im Erlebnisgebiet verknüpfen helfen.

Zwei Begriffe sind aus dem vorgehenden Abschnitt besonders herauszuheben:

- „zugeschnittene Umweltinformationen“
- „emotionale Bindung“

Ein guter menschlicher Führer wird nach Möglichkeit die Teilnehmer der Führung mit in seine Präsentation einbeziehen [Ham 1992]. Dabei erfolgt bei ihm bewusst oder unbewusst eine Anpassung der zu vermittelnden Informationen an die Interessen und den Kenntnisstand der Besucher. So benötigt ein Kind andere Erklärungen von Sachverhalten als ein Erwachsener. Dies versteht man unter Informationen, die auf eine bestimmte Zielgruppe zugeschnitten sind.

Mit dem zweiten Begriff „emotionale Bindung“ ist eine Identifikation der Besucher mit der Natur gemeint. Dieser Begriff kommt aus der Umweltpädagogik. Die Identifikation oder Bindung kann durch positive Erlebnisse mit der Natur während einer Tour entstehen. Die Motivation, eine emotionale Bindung zwischen Natur und Besucher zu erreichen, ergibt sich aus dem Ziel, ein umweltgerechteres Verhalten des Besuchers zu bewirken. Eine erfolgreiche Umweltkommunikation präsentiert nicht nur Fakten, sondern erzeugt eine anhaltende Bindung des Besuchers mit der Natur. Ein Naturführer kann idealer Weise diese Bindung erreichen, indem er die Informationen so präsentiert, dass sie sowohl unterhaltsam als auch emotional bindend sind. Zusätzlich gibt er dem Besucher das Gefühl, dass die Präsentation für ihn persönlich erarbeitet wurde. Thema des MobiNaf Projektes ist die Entwicklung eines mobilen elektronischen Naturführers, der diese zwei wesentlichen Anliegen berücksichtigt: d. h. zielgruppengerechte Informationen für Besucher von Natur Erlebnisgebieten liefert, wobei zusätzlich noch eine emotionale Bindung zwischen Besucher und Naturgebiet durch den elektronischen Naturführer hergestellt wird.

Zielgruppe des Systems sind Anwender, die keine Experten sind und sich auch nicht ausgiebig mit der Bedienung eines mobilen Systems beschäftigen wollen, wie Schüler oder Familien. Dafür ist eine neue, intuitive Interaktionsmethode zwischen Gerät/Software und Benutzer erforderlich. Vorbild dabei ist der menschliche Naturführer, der z. B. in einem Naturschutzzentrum die Besuchergruppen durch das Gebiet führt und ihnen dabei Naturphänomene erläutert.

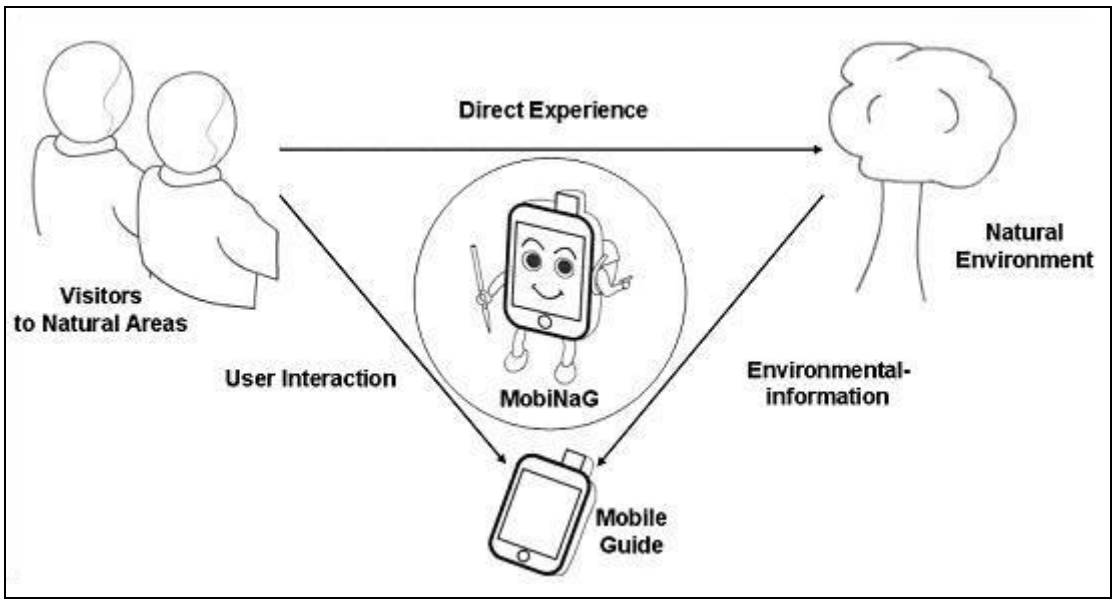

Abbildung 2: Mobile Nature Guide

Hierbei spielt vor allem der Einsatz eines emotionalen pädagogischen Agenten eine besondere Rolle. Dieser Typ von animierten Agenten wurde im Zusammenhang mit virtuellen Lernumgebungen entwickelt, um Engagement und Motivation der Lernenden zu steigern [Brna 2001]. Die Evaluation von Mulken [Mulken 1998] über den Effekt von pädagogischen Agenten hat gezeigt, dass sie eine wichtige Rolle für die Motivation des Lernenden übernehmen können. Diese Eigenschaft hat eine besondere Bedeutung im MobiNaf Projekt, wo besonders Kinder motiviert werden sollen, sich intensiv mit Naturphänomen zu beschäftigen. Die zentrale Position des Agenten als Vermittler von Umwelt-Informationen („Environmental Information“) und Motivator für das direkte Erfahren („Direct Experience“) der Natur ist in Abbildung 2 dargestellt. Der Vogel, der Michael und seine Mitschüler im Anwendungsszenario durch das Naturschutzgebiet führt, entspricht dem hier vorgestellten Konzept eines pädagogischen Agenten.

Im Rahmen dieser Arbeit soll ein Konzept für den im vorherigen Abschnitt motivierten emotionalen pädagogischen Agenten für das MobiNaf-Projekt erstellt werden. Dabei soll auch geklärt werden, wie der Agent in den mobilen Naturführer integriert werden kann. Wichtig sind dabei zum einen die Berücksichtigung der bestehenden Architektur und zum anderen die konzeptuelle Betrachtung der Neu-Gestaltung im Rahmen eines Re-Designs.

Wichtiger Teil der Implementierung ausgewählter Komponenten ist die Erstellung einer Komponente zur Darstellung des Agenten. Die grafische Gestaltung des Agenten soll als vektorgraphikbasierte Animation unter Einsatz von SVG erfolgen. Weiterhin wird eine prototypische Implementierung des Agenten unter Verwendung von C++ und XML auf einem PocketPC-PDA angestrebt. Dabei soll die Funktionalität von Teilen der Lösung demonstriert werden.

Schließlich soll in dieser Arbeit eine Übersicht über den aktuellen Forschungsstand und die verwendeten Technologien im Bereich der pädagogischen emotionalen Agenten gegeben werden.

Die Arbeit ist daher in zwei Hauptbereiche gegliedert. Kapitel 2 und 3 beinhalten eine theoretische Betrachtung der Thematik der pädagogischen Agenten. Dabei werden zuerst Konzepte aus den hier relevanten Forschungsgebieten über Agenten, multimodale Kommunikation, Emotionen, Beschreibungs- und Skriptsprachen, Charakter-Animation, Kontext und Verhaltensplanung vorgestellt und auf ihre Anwendbarkeit für das MobiNaf-Projekt überprüft. Danach werden die in Kapitel 2 eingeführten Begriffe in Kapitel 3 anhand von konkreten Systemen betrachtet. Dabei werden auch Probleme und Mängel dieser Systeme analysiert.

Der zweite Hauptbereich in Kapitel 4 und 5 beschäftigt sich mit der Anwendung der in Kapitel 2 und 3 vorgestellten Konzepte. In Kapitel 4 wird die im Rahmen dieser Arbeit entwickelte Architektur für den pädagogischen Agenten vorgestellt. Wichtige Konzepte und Teile aus der Architektur, die in das existierende MobiNaf-System integriert wurden, sind in Kapitel 5 dokumentiert.

Kapitel 6 gibt abschließend ein Resümee über den Stand des Systems und bietet eine Vorschau auf zukünftige Entwicklungen.

Kapitel 2: Stand der Forschung und wichtige Begrifflichkeiten

In diesem Abschnitt soll das Forschungsgebiet der life-like characters tiefergehend betrachtet und wichtige Begrifflichkeiten und Forschungsergebnisse eingeführt werden. Eine Schwierigkeit in dieser Disziplin ist, dass es keine Einigkeit über die Definition der einzelnen Bezeichnungen für die Formen von life-like characters gibt. Da die Literatur fast ausschließlich in englischer Sprache vorhanden ist, werden hier bis auf wenige Ausnahmen die englischen Begriffe verwendet. Die Begriffe werden bei der Einführung kurz erläutert. So gibt es verschiedene Namen für das gleiche Konzept sowie Ober- und Unterklassen:

- Embodied (Conversational) Agent (ECA), Life-like character, Animated (Interface) Agent: Agenten mit grafischer Repräsentation, deren Verhalten durch den Agenten gesteuert wird.
- Digital Puppet oder Avatar: Hauptsächlich nur grafische Repräsentation. Steuerung erfolgt durch den Benutzer oder anderen Systeme.
- Interface Agent: Agent zur Vereinfachung der Verwendung des Systems.
- Präsentations-Agent: Agent auf das Darstellen und Erzeugen von Präsentationen spezialisiert.
- Pädagogischer Agent oder Virtueller Lehrer: Spezialisierter Agent für die Vermittlung von Wissen.
- Relational Agent, Social Agent: Agent, der die langfristige Bindung zwischen Anwender und System beobachtet und verwendet.
- Anthropomorphic Agent: Agent, der spezielle menschliche Eigenschaften nachbilden soll.

Innerhalb dieser Arbeit wird der Begriff „life-like character“ als Oberbegriff verwendet. Der Begriff „Embodied Conversational Agent“ wird speziell für Dialog-Systeme - im Sinne von Konversation zwischen Mensch und Maschine, vertreten durch ein Abbild eines menschlichen Körpers - verwendet. Dieser Körper kann physikalisch (z. B. ein Roboter) oder virtuell sein. Interface Agent und Präsentations Agent sind Agenten für spezielle Anwendungsgebiete, wie in Kapitel 3.1 noch genauer dargelegt wird. Der Pädagogische Agent sollte zudem noch Eigenschaften haben, die eine bessere Vermittlung von Lehrinhalten ermöglichen. Animated Agent wird meistens synonym zu Embodied Agent verwendet. Relational und Social Agent haben den Fokus auf der emotionalen Bindung zu dem Benutzer. Die Abgrenzung zum pädagogischen Agenten ist schwer, da auch hier eine Bindung zwischen Schüler und Lehrer wichtig ist. Daher werden die Begriffe auch häufig synonym verwendet. Mit Puppet oder Avatar ist dagegen nur die grafische Darstellung eines Agenten gemeint, was dem Body entspricht. In Kapitel 2.1 sollen diese Begrifflichkeiten nun präziser erläutert und gegeneinander abgegrenzt werden.

2.1 Agenten

In diesem Kapitel werden die Hauptklassen von hier relevanten Agenten beschrieben. Bei vielen Systemen ist eine eindeutige Zuordnung schwierig, da der Agent zumeist mehrere Funktionen ausübt. Zuerst wird eine Definition der Begriffe Agent und intelligenter Agent mit typischen Architekturen gegeben. Häufig herrscht noch Unklarheit, ab welchen Kriterien die Bezeichnung Agent gerechtfertigt ist [Franklin 1996]. Im Anschluss an die Begriffsdefinition sollen die Spezialfälle der Präsentations-, Interface-, Pädagogischen- und Conversational Agenten beschrieben und voneinander abgegrenzt werden.

2.1.1 Agent und Intelligenter Agent

Es ist wichtig an dieser Stelle zu klären, was in dieser Arbeit unter einem Software-Agenten verstanden wird. Dies ist nötig um später zu zeigen, ob und wie die in Kapitel 4 entwickelte Architektur diese Merkmale erfüllt. Auch wird gerade im Bereich der life-like characters der Begriff „Agent“ in unterschiedlichen Ausprägungen verwendet. Eine praktische Definition kommt von Wooldridge in „Multiagent Systems“ [Wooldridge 2000]:

> „Ein Agent ist ein Computersystem, das in einer *Umgebung* eingebettet ist und das in dieser Umgebung in der Lage ist *autonome Aktionen* auszuführen um seine Design Ziele zu erfüllen.“

Eine Umgebung kann dabei sowohl physikalischer als auch virtueller Natur sein. Wichtig ist, dass der Agent über Sensoren Informationen über die Umgebung erfährt und daraus eine auszuführende Aktion ermitteln kann. Je nach Architektur und Arbeitsweise des Agenten wird zwischen proaktivem und reaktivem Verhalten eines Agenten unterschieden. Ein Agent kann damit durch die Eigenschaften Autonomie, Reaktivität und Proaktivität charakterisiert werden. Proaktivität bedeutet, dass das System selbstständig die Initiative ergreift. Das Verhalten hängt also weniger von äußeren Ereignissen ab. Stattdessen wird versucht, Ziele zu definieren und diese durch rationales Verhalten zu erreichen. Dagegen ist Reaktivität geprägt durch die Reaktion des Systems innerhalb einer bestimmten Zeit auf Veränderungen in der Umgebung.

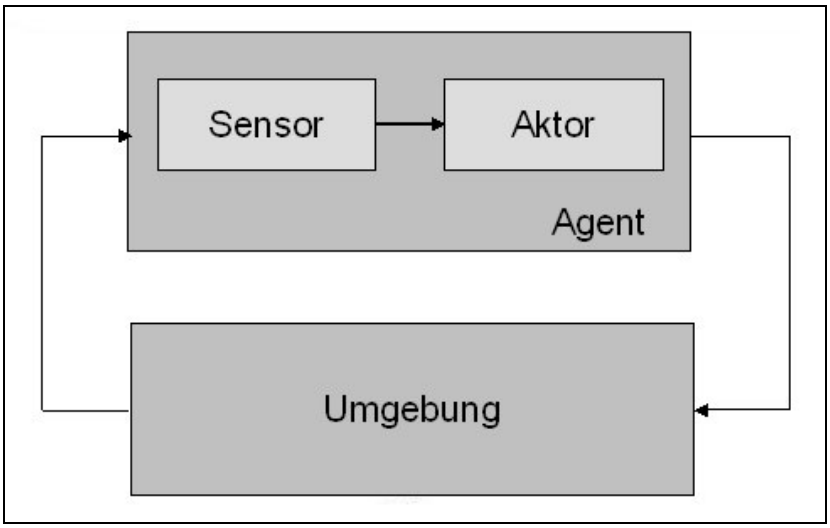

Abbildung 3: Agenten-Architektur

Selbst ein relativ einfaches reaktives System wie ein Temperaturregler kann nach Wooldridge als Agenten-System gesehen werden. Wichtig ist, dass der Agent fähig ist, flexibel und autonom auf die Umgebung zu reagieren. Resultierende Aktionen sollten aufgrund von aus der Umgebung extrahierten Merkmalen erzeugt werden, wie in Abbildung 3 dargestellt. Für den Prozess des Ableitens der Aktion kann man nach Wooldridge vier Architekturvarianten unterscheiden.

Logik-basierte Architekturen

Bei diesen Architekturen wird die resultierende Aktion per Deduktion über einer Regelbasis abgeleitet. Das Problem dieses Verfahrens ist, dass es je nach Größe der Regelbasis relativ aufwendig sein kann und dadurch kritische Echtzeitanforderungen nicht mehr eingehalten werden können. Genauso können Änderungen in der Umgebung während des Entscheidungsprozesses nicht mehr mit einkalkuliert werden.

Reaktive Architektur

Die Grundlage einer reaktiven Architektur ist die Idee, dass sich erstens intelligentes rationales Verhalten aus der Interaktion des Agenten mit seiner Umwelt und zweitens aus einer Verknüpfung von einfachen Verhaltensregeln ableiten lässt. Das bedeutet, dass jedes Verhalten einer Regel entspricht, durch die der aktuelle Umweltzustand direkt auf eine Aktion abgebildet wird. Mehrere Regeln können dabei gleichzeitig ausgeführt werden. Um eine Aktion auszuwählen, sind Regeln Prioritäten zugeordnet und es wird die Aktion mit der höchsten Priorität ausgewählt. Ein Problem ist hierbei, dass bei großen Regelmengen viele Prioritäten zu vergeben sind und hierbei das Verhalten schnell für den Entwickler nicht mehr nachvollziehbar ist.

BDI (Belief-Desire-Intention) Architektur

Die Idee der Belief-Desire-Intention-Architektur basiert auf dem Ablauf einer rationalen Entscheidungsfindung beim Menschen. In dem Prozess gibt es zwei entscheidende Schritte:

- Herausfinden, welche Ziele erreicht werden sollen. Dafür werden aus den Vorstellungen über die Umwelt („Beliefs“) die zukünftigen Wünsche („Desires“) erzeugt. Die *intentions* sind die konkreten Ziele, die aus den *desires* ausgewählt wurden.
- Ableiten der notwendigen Aktionen, durch die die ermittelten Ziele erreicht werden können.

Dieses Verfahren hat die Vorteile, dass es relativ intuitiv nachvollziehbar ist und bereits eine gute funktionale Aufteilung aufzeichnet. Das wesentliche Problem liegt darin, die einzelnen Funktionen effizient zu implementieren.

Geschichtete (Layered) Architektur

Da ein Agent häufig sowohl reaktives als auch proaktives Verhalten zeigen soll, liegt die Idee nahe, eine Gesamtarchitektur mit zwei getrennten Subsystemen für reaktives bzw. proaktives Verhalten zu erstellen. In so einem System gibt es minimal zwei Schichten, allerdings gibt es keinen Grund, nicht wesentlich mehr Schichten einzufügen. Dabei unterscheidet man zwischen horizontaler und vertikaler Schichtung. Bei horizontaler Schichtung arbeitet jede Schicht wie ein einzelner Agent, jede Schicht ist direkt mit dem Sensor verbunden. Beim vertikalen Modell werden die Sensorinformationen durch mehrere hintereinander liegende Schichten geleitet. Die Koordination, welche dieser Schichten die Kontrolle über das Gesamtverhalten hat, unterliegt einer zentralen Koordinierungseinheit. Für die Entscheidung kann ein Regelsystem verwendet werden.

2.1.2 Präsentations-Agent

Präsentations-Agenten erlauben die Nachahmung von menschlichen Präsentations-methoden [Mulken 1997]. Sie können z. B. Gesten und Sprache verwenden, um einen Zusammenhang zwischen präsentierten Bildern und Texten herzustellen [Lester 1997]. Nach Mulken muss ein Präsentations Agent verschiedene Kriterien erfüllen. Der Agent muss die für Präsentationen üblichen Gesten und Körperpositionen darstellen können. Außerdem darf er nicht den Anwender von den eigentlichen Inhalten ablenken. Speziell zur Beschreibung von multimodalen Präsentationen mit einem Präsentations-Agent wurde die Beschreibungssprache MPML (Multimodal Presentation Markup Language) entwickelt [Ishizuka 2000]. PPP Persona [André 1996] ist ein häufig zitiertes System zur auto-

matischen Erstellung von Präsentationen. Das System besteht aus einem Präsentations-Planer und der Persona-Engine zur Darstellung der generierten Präsentationen.

2.1.3 Interface Agent

Die Motivation für Interface-Agenten kommt aus dem Wunsch, die Bedienung von komplexen Benutzerinterfaces von Applikationen durch den Einsatz von Agenten zu vereinfachen. In der Regel ist das Erlernen von spezifischen Methoden notwendig, wie z. B. eine bestimmte Reihenfolge von Operationen, die aus einer verschachtelten Menustruktur ausgewählt werden müssen. Knopp [Knopp 2003] hat folgende Definition aufgestellt:

> „Interface-Agenten sind Computerprogramme, die den Benutzer bei der Bewältigung bestimmter Aufgaben zum Teil autonom unterstützen und bei der Bedienung des technischen Systems vermitteln."

Nach Schiaffino [Schiaffino 2003] sollten sich Interface-Agenten über die Zeit an den Benutzer anpassen, indem sie dessen Interessen und Arbeitsweise lernen. Smartakus des SmartKom-Projekts (s. Kapitel 3.4) und die aus Microsoft Office bekannte Büroklammer sind Beispiele für Interface-Agenten.

2.1.4 Pädagogischer Agent

Nach Johnson [Johnson 2000] soll ein pädagogischer Agent das Lernen in einer computer-basierten Lernumgebung vereinfachen. Der Agent braucht dabei nicht notwendigerweise ein Verständnis für die zu vermittelnden Lerninhalte. Gesten und Bewegungen des Agenten können die Aufmerksamkeit des Benutzers fokussieren. Durch den pädagogischen Agenten kann der Lernende zusätzlich motiviert werden, sich mit den Inhalten zu beschäftigen („persona effect") [Lester 1999]. Stone und Lester [Stone 1996] nennen drei Eigenschaften, die ein pädagogischer Agent erfüllen sollte:

- Contextuality: Das Verhalten des Agenten muss dem aktuellen Lernkontext und Benutzerkontext entsprechen.
- Continuity: Die grafische Darstellung muss visuell kohärent sein. Der Agent sollte direkt in die Lernumgebung eingebettet sein, so dass eine hohe Interaktion zwischen den Lerninhalten und dem Agent möglich ist.
- Temporality: Die Lerninhalte müssen an die Zeitressourcen des Schülers angepasst sein. Die wichtigsten Inhalte sollten in der zur Verfügung stehenden Zeit vermittelt werden.

2.1.5 Conversational Agent

Nach Li [Li 2004] ist ein Embodied Conversational Agent (ECA):

> „… a life-like virtual human capable of carrying on conversations with humans by both understanding and producing verbal and nonverbal behaviors."

Ein ECA sollte die menschliche verbale und nonverbale Interaktion emulieren können [Cassell 2000]. Dabei sind besonders die nonverbalen Kommunikationsformen wichtig. Durch Gesten und Mimik werden zum einen Gefühl und Bedeutungen ausgedrückt. Zusätzlich wird dadurch der Dialogablauf mitgesteuert (das sogenannte „turn-taking") [Argyle 1988]. Um einen möglichst natürlichen Dialog zu führen, ist ein so genannter Dialog-Planer („Discourse planer") notwendig [Bickmore, 2003].

Ein „Life-like character"-System besteht im Wesentlichen aus drei Komponenten: Wissen, Geist und Körper (Abbildung 4). In der Abbildung ist eine prinzipielle Grundaufteilung der Architektur dargestellt. Die Steuerung geschieht im Geist, hier wird das Verhalten auf der Basis des Wissens über die aktuelle und historische Situation des Charakters und das Faktenwissen entschieden. Ausgeführt wird das Verhalten durch den Körper, der die abstrakten Verhaltensanforderungen auf konkrete Aktionen umsetzt.

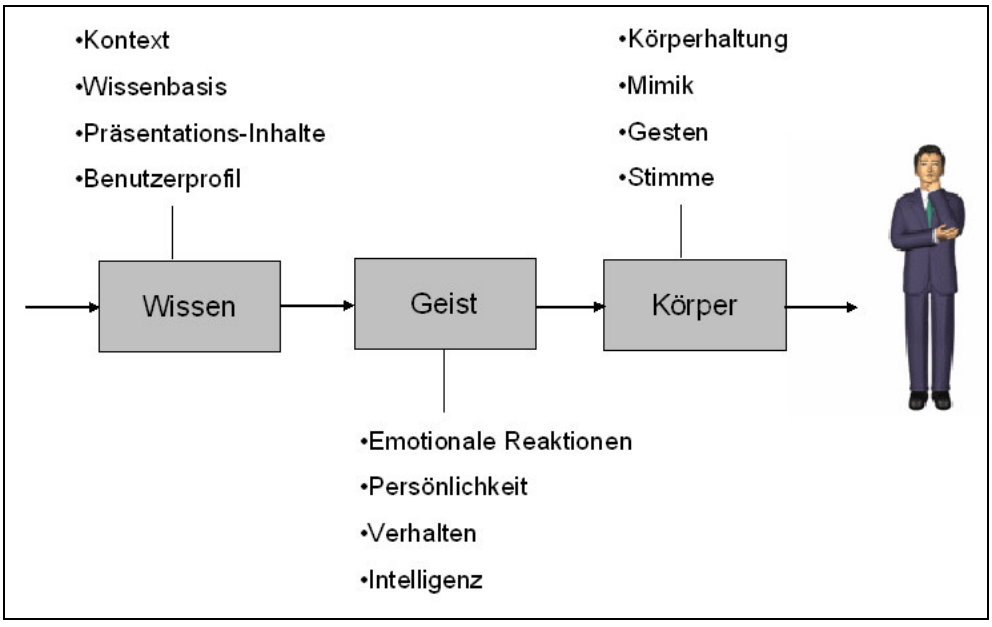

Abbildung 4: Life-like character, Wissen, Geist, Körper

Das Forschungsgebiet der life-like character Systeme vereinigt eine Reihe von unterschiedlichen Disziplinen aus Informatik, Linguistik, Psychologie, Soziologie und Design.

Im Folgenden soll jeweils eine kurze Übersicht über die einzelnen Unterbereiche gegeben werden. So werden die unterschiedlichen Kommunikationsformen eines life-like characters wie Blick, Gesten, Mimik und Sprache im Kapitel 2.2 „Multimodale Kommunikation“ beschrieben. Verschiedene Emotionsmodelle werden in Kapitel 2.3 vorgestellt. Es haben sich ein Reihe von Beschreibungssprachen für life-like character etabliert, die anschließend in Kapitel 2.4 dargestellt werden. Für eine Darstellung der Bewegungen des Charakters werden Animationstechniken eingesetzt. Diese werden in Kapitel 2.5 genauer betrachtet.

2.2 Multimodale Kommunikation

Aus der Sicht der Informatik ist die multimodale Kommunikation ein interdisziplinäres Thema aus dem Bereich der Mensch-Maschine-Kommunikation und künstlichen Intelligenz. Das Ziel sind benutzerfreundliche Systeme, die für den Menschen natürliche Kommunikationsformen bereitstellen. Informationen zwischen Menschen können in unterschiedlichen Modalitäten übertragen werden.

Im Wesentlichen werden drei unterschiedliche Sinne für den Empfang verwendet.

- Sehen: Optisch
- Fühlen: Haptisch
- Hören: Akustisch

Dem entsprechend gibt es die passenden „Sender“:

- Mund: Sprache
- Hand, Arme: Schrift, Zeichnungen, Gesten
- Gesicht, Körper: Mimik, Gesten, Blick

In Abbildung 5 ist der Zusammenhang von Wahrnehmung und Erzeugung bei menschlicher Kommunikation noch einmal dargestellt.

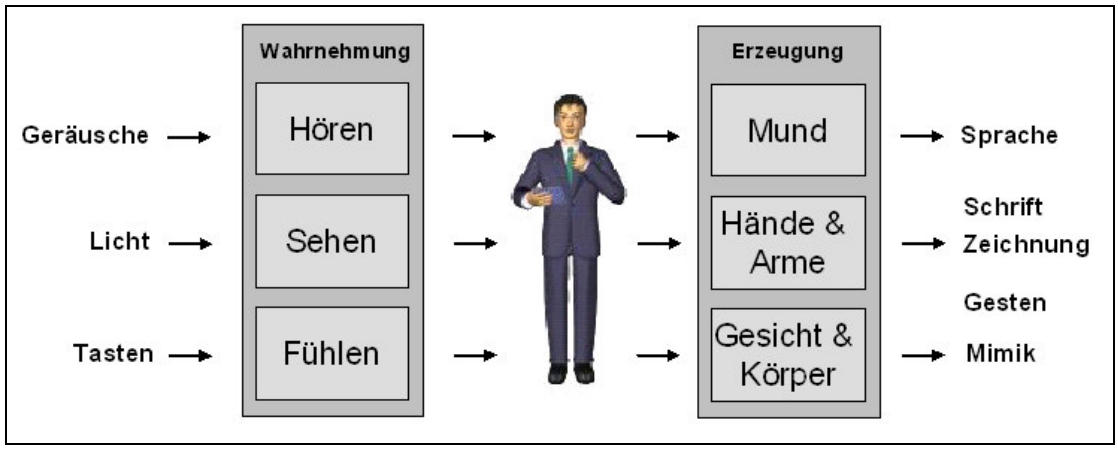

Abbildung 5: Wahrnehmung und Erzeugung von menschlicher Kommunikation

In dieser Arbeit sind im Wesentlichen Sprache, Gesten, Mimik und Blick und nicht die Wahrnehmung von Modalitäten relevant. Eine ganzheitliche Kommunikation wird im SmartKom-Projekt (s. Kapitel 3.4) angestrebt.

Der pädagogische Agent soll verbales und nonverbales Verhalten erzeugen können. Deswegen muss für jede Modalität verstanden werden, welche Bedeutung die Verwendung eines bestimmten Verhaltens hat.

Animierte Agenten haben dadurch, dass sie einen Körper besitzen, die Möglichkeit multimodale Kommunikationsformen zu nutzen. Ein effektiver Informationsaustausch ist wichtig für das Design eines Präsentationsagenten. Dabei muss die Zuordnung von verbalen und nonverbalen Verhalten dem Kommunikationsziel entsprechen. Der Mensch ist bei der Wahrnehmung von Kommunikationsfehlern, wie falsch zugeordneten Gesten, sehr empfindlich. Eine nicht kohärente Kommunikation kann leicht zu Frustration und zur Ablehnung des Systems führen.

2.2.1 Gesten

Spontane (ungeplante, unbewusste) Gesten begleiten die gesprochene Sprache in fast allen kommunikativen Situationen [Cassell 1998]. Das gilt für weitgehend alle Kulturen. Gesten erweitern den Inhalt der gesprochenen Sprache. Durch den Kontext der Geste kann der Inhalt dabei eine andere Bedeutung bekommen. Obwohl Gesten nicht essentiell für das Verständnis von Sprache sind, haben sie eine wichtige Bedeutung bei doppeldeutigen Ausdrücken oder in Situationen, bei denen z. B. durch zu viel Lärm die eigentliche Sprache übertont wird. Etwa 90% aller Gesten werden im Kontext von gesprochener Sprache verwendet.

Nach [Kendon 1980] bestehen Gesten üblicherweise aus 3 Phasen:

- Preparation („Vorbereitung“): Bewegung vom Ausgangspunkt hin zum Start der Geste.
- Stroke: Dies ist die eigentliche Geste und der energischste Teil der Bewegung.
- Retraction („Rückzug“): Bewegung zurück zur Ausgangsposition.

Der Ablauf einer Geste ist in Abbildung 6 beispielhaft für eine „Ziehen“ Geste dargestellt. In der Preparation wird der Arm aus der Ruheposition in die Startposition für den Stroke bewegt. Dann wird die eigentliche Ziehbewegung dargestellt. Danach bewegt sich der Arm wieder in die Ausgangsposition zurück.

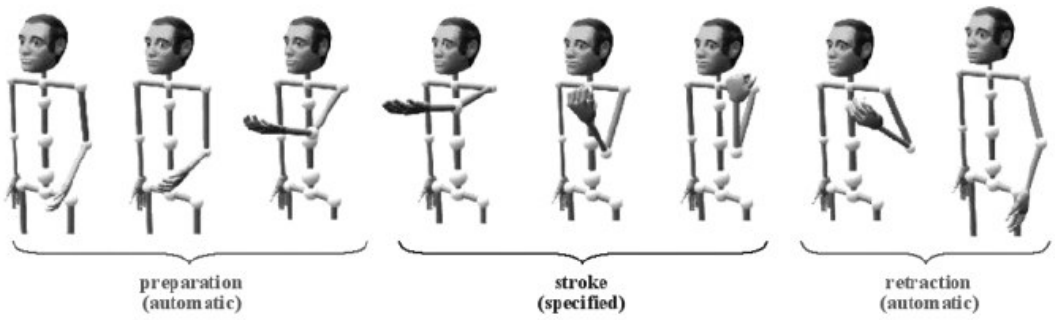

Abbildung 6: „Ziehen“ Geste (Iconic gesture)

In: Wachsmuth 2001

Dabei ist die Synchronisation zwischen den einzelnen Phasen der Geste mit der gesprochenen Sprache entscheidend. So ist meistens die stroke-Phase synchron zur der am stärksten betonten Silbe des dazugehörigen Sprachabschnitts. [Kendon 1980, McNeill 1992]. Gesten treten meistens parallel zu den semantisch passenden Textabschnitten auf und werden nicht bewusst ausgeführt und ebenfalls nicht bewusst wahrgenommen. In [Cassell 1996] findet sich eine Auflistung und Klassifikation von Gesten. Es werden bewusste und unbewusste Gesten unterteilt.

Bewusste Gesten:

- Emblematic gesture: Kulturell spezifizierte Gesten. (z. B. „V ... Victory“, „O.K.“, „Daumen hoch“-Gesten). Eigentlich relativ selten ausgeführte Gesten, die dafür meistens bewusst gemacht werden.
- Propositional gesture: Z. B. Abgrenzung eines symbolischen Raums („So groß ...“).

Unbewusste Gesten:

- Deictic gesture: Verdeutlichen Teile des präsentierten Materials durch Zeigen auf Objekte, auf die Bezug genommen wird („Dieser Stuhl“).
- Beat gesture: Schlagstock artige Bewegungen, die eine Art Rhythmus oder Strukturierung für den Text vorgibt („Dieser Lack funktioniert bei !VW und !Mercedes.“).
- Iconic gesture: Spezifizieren die Art und Weise, wie eine Aktion ausgeführt wird. In Abbildung 6 ist dargestellt, wie jemand etwas zieht.
- Metaphoric gesture: Repräsentieren eine nichtphysische Bedeutung, stattdessen kommt die Geste von einer alltäglichen Metapher (z. B. „Die Verhandlung ging immer weiter“ verbunden mit einer rollenden Handbewegung.).
- Contrast gesture: Gegensätze im Text. „Ich wollte !rote Äpfel kaufen, aber es gab nur !grüne.“

Das Verständnis für den semantischen Zusammenhang von Gesten und Sprache ist wichtig für Systeme, die automatisch Verhalten zu Texten hinzufügen (z. B. BEAT, CAST). Die Synchronisation und Kohärenz ist entscheidend für die Glaubwürdigkeit und Natürlichkeit des Charakters.

2.2.2 Mimik

Nach Wikipedia ist „Mimik die sichtbare Bewegung der Gesichtsoberfläche“. Emotionen zeigen sich sehr deutlich im Gesicht und sind für den Menschen kaum zu verbergen [Ekman 1972]. Die Hauptkomponenten sind Mund, Wangen, Augen, Augenbrauen und Stirn. [Bartneck 2001]. Interessanterweise ist kein besonders hoher Detailgrad notwendig für eine präzise Erkennung von Emotionen im Gesicht (Abbildung 7) [Etcoff 1992].

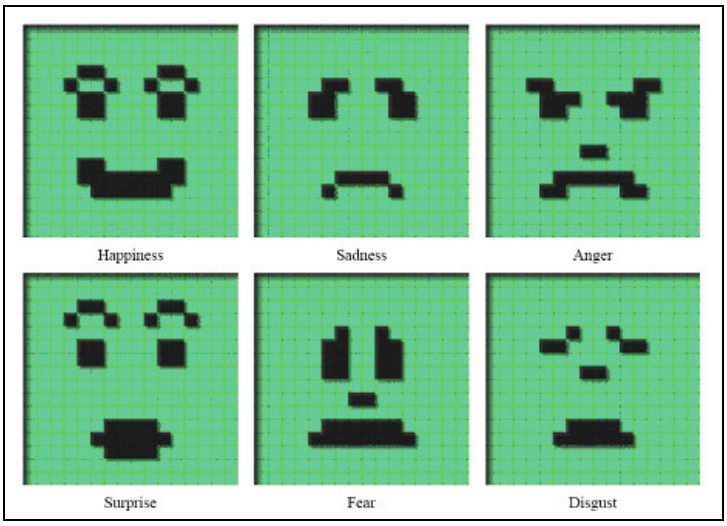

Abbildung 7: Auf das Wesentliche reduzierte Mimiken

In: Bartneck 2000

Etcoff hat sehr vereinfachte Bilder mit der mimischen Darstellung von Freude, Traurigkeit, Wut, Überraschung, Angst und Abscheu (von links nach rechts und oben nach unten) erzeugt und diese Bilder mit Gruppen getestet. Katsikis und Bartneck [Katsikis 1997, Bartneck 2002] haben in ihren Studien gezeigt, dass die in synthetischen Gesichtern dargestellten Emotionen mit einer hohen Wahrscheinlichkeit richtig erkannt werden. Interessant ist die Überlagerung von unterschiedlichen Emotionen („Ein lachendes und ein weinendes Gesicht“). Dies wird im Magicter Projekt [Magicter 2003] beachtet, wo verschiedene Emotionen nebeneinander aktiv sein können.

2.2.3 Sprache

Sprache ist mehr als nur die Übertragung von sachlichen Informationen. Hierbei werden ebenso Stimmungen und Meinungen übermittelt [Bartneck 2001]. Im Vergleich zu den Arbeiten zu Emotion und nonverbalen Verhalten gibt es relativ wenig Aktivitäten über die

Verwendung von Emotionen zur Spracherzeugung bei „life-like characters“ [Fleischmann 2002]. Ohne ein spezielles System zu betrachten, entspricht der grobe Ablauf bei der Erstellung von Sprachausgabe und Mundanimation meistens dem in Abbildung 8 dargestellten.

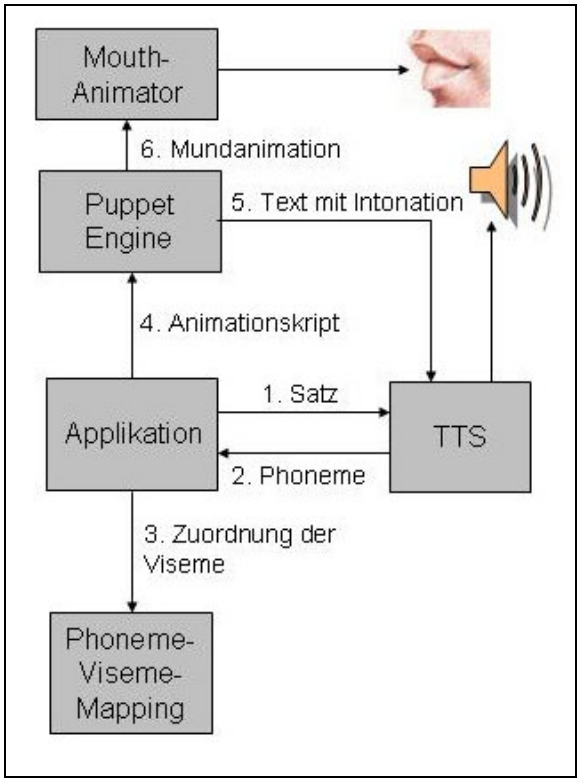

Abbildung 8: Beispiel-Ablauf für Erzeugung von Mundanimation und Sprachausgabe

Intonation

Unter Intonation versteht man den Tonhöhenverlauf innerhalb eines Satzes. Die Intonation ist wichtig, um Sprache lebendig erscheinen zu lassen. Benzmüller, Grice und Baumann [Benzmüller 2005] haben eine Beschreibungssprache für die deutsche Intonation entworfen. Die Intonation wird dabei durch L für einen tiefen Ton und H für einen hohen Ton beschrieben. Grundsätzlich wird dabei in der Beschreibungssprache zwischen Tonhöhenbewegungen, die dazu dienen Silben in Wörtern und Sätzen hervorzuheben, und solchen, die an Phrasengrenzen stattfinden, unterschieden (siehe THEME-Beispiel unten). Erstere heißen Akzenttöne („accent“), letztere Grenztöne („boundary tone“). Eine ähnliche Notation wird für die Erzeugung der Intonation bei BEAT verwendet [BEAT 2002]. Dabei werden allerdings keine Emotionen sondern nur Textabschnitte unterschieden (z. B. Betonung von „Auto“ bei „Heute habe ich mir ein Auto gekauft“). Das folgende Codebeispiel zeigt eine typische Regel zur Erzeugung der Intonation. Diese Regel ist Teil eines mehrstufigen Prozesses, so wurden in einem vorgehenden Schritt Satzfragmente zu Rhemes oder Themes zugeordnet. Das bedeutet grob, Theme ist ein Abschnitt, in dem ein Objekt neu eingeführt wird, und Rheme der Abschnitt, in dem bereits bekanntes wiederholt

wird. Bei dem Satz „Heute habe ich mir ein Auto gekauft.“ ist „ein Auto gekauft“ das Theme und „Auto“ das neue Objekt. Nach der Regel im Codebeispiel sollte eine von einem tiefen Ton zu einem hohen Ton wechselnde Betonung auf dem Wort „Auto“ liegen.

```
Within THEME:
 Suggest L+H* accent for NEW objects
 Suggest LH% boundary tone at end of Theme.
```

Sprachparameter und Emotion

Emotionen haben ebenfalls einen Effekt auf die Sprache. So wird die Stimme bei Angst schneller und wesentlich höher [Murray 1992]. Dazu wurde von Murray der Einfluss von einzelnen Parametern der Sprechweise (Tonhöhe, Lautstärke, Silben- und Lautdauer) und der Stimmqualität auf ihren Einfluss bei der menschlichen Wahrnehmung beobachtet.

Als Kriterien für das Zusammenspiel von Sprache und Emotionen unterscheidet Murray:

- Tempo („Speech rate“): Die Sprechgeschwindigkeit
- Durchschnittstonhöhe („Pitch average“)
- Tonhöhenbereich
- Tonhöhenveränderung
- Sprachqualität
- Artikulation
- Lautstärke

Tabelle 1 zeigt eine von Murray zusammengestellte Auflistung von qualitativen Eigenschaften der Emotionen Wut („Anger“), Freude („Happiness“), Traurigkeit („Sadness“), Angst („Fear“) und Disgust („Abscheu“) bzgl. der Kriterien. Die wichtigsten Kriterien davon sind Tempo, Lautstärke und Intonation [Bartneck 2001]. Nur durch Änderung der Intonation lassen sich bereits Emotionen auf die Sprache übertragen. Ein gutes Beispiel dafür ist ein Telefongespräch [Bartneck 2001]. Ein verärgerter Anrufer klingt anders als ein froher Anrufer. Die meisten Menschen sind in der Lage, den emotionalen Zustand eines Sprechers zu bewerten, ohne den Inhalt der Sprache zu verstehen.

Tabelle 1: Sprach Parameter für verschiedene Emotionen [Murray 1992]

	Anger	Happiness	Sadness	Fear	Disgust
Speech rate	Slightly faster	Faster or slower	Slightly slower	Much faster	Very much slower
Pitch average	Very much higher	Much higher	Slightly lower	Very much higher	Very much lower
Pitch range	Much wider	Much wider	Slightly narrower	Much wider	Slightly wider
Intensity	Higher	Higher	Lower	Normal	Lower
Voice quality	Breathy, chest tone	Breathy, blaring	Resonant	Irregular voicing	Grumbled chest tone
Pitch change	Abrupt, on stressed syllables	Smooth, upward inflections	Downward inflections	Normal	Wide, downward terminal inflections
Articulation	Tense	Normal	Slurring	Precise	normal

Phoneme

Laut Wikipedia ist ein Phonem „die kleinste bedeutungstragende Einheit eines Sprachsystems“. Anders formuliert sind Phoneme die Geräusche, aus der die menschliche Sprache aufgebaut ist. In der englischen Sprache gibt es etwa 35 Phoneme, wogegen die deutsche Sprache 42 Phoneme hat. [Aschenberner, 2005]. Die meisten TTS Systeme (wie z. B. Microsoft TTS) sind in der Lage, automatisch zu einem Textabschnitt die für die Betonung notwendigen Phoneme zu generieren. [Microsoft-TTS[1]].

Viseme

Ein Visem ist die Form des Mundes beim Sprechen eines Phonems. Daher können Phoneme auf Viseme abgebildet werden („Phone-Viseme Mapping“). Mehrere Phoneme werden mit dem gleichen Visem gesprochen. Viseme sind wichtig für eine synchrone Darstellung der Lippen während einer Sprachausgabe. In Tabelle 2 ist die Zuordnung von Phonemen und Viseme für die deutsche Sprache dargestellt. Dabei ist deutlich, dass mehrere Phoneme auf das gleiche Viseme abgebildet werden.

[1] Microsoft Speech SDK: http://www.microsoft.com/mind/0299/cutting/cutting0299.asp

Tabelle 2: Phonme-Viseme Tabelle [Aschenberner 2005]

No.	Phoneme (BOSS)	Viseme	Example
1	p, b	**P**	Pause, Bitte
2	t, d, k, g	**T**	Tonne, Dach, König, Gier
3	n, @n, l, @l	**N**	Nadel, raten, Liebe, Igel
4	m	**M**	Mutter
5	f, v	**F**	Finder, Vase
6	s, z	**S**	Fass, Sein
7	S, Z, tS, dZ	**Z**	Schein, Garage, Tscheche, Dschungel
8	h, r, x, N	**R**	Hase, Reden, Dach, Wange
9	j, C	**C**	Junge, Wicht
10	i:, I, e:, E:, E	**E**	Bier, Tisch, Weg, Räte, Menge
11	a:, a	**A**	Wagen, Watte
12	o:, O	**O**	Wolle, Wogen
13	u:, U	**U**	Buch, Runde
14	@, 6	**Q**	Bitte, Weiher
15	y:, Y, 2:, 9	**Y**	Tür, Mütter, Goethe, Götter

TTS-Systeme

Die sprachliche Ausgabe eines Textes innerhalb einer Applikation erfolgt über eine Text-to-Speech (TTS)-Komponente. In Tabelle 3 ist eine kleine Auswahl von TTS-Systemen beschrieben, wobei der Fokus auf PDA-Lauffähigkeit, deutsche Sprachausgabe und automatischer Intonations-Generierung liegt. Es gibt eine Vielzahl von kommerziellen und universitären Systemen. Eine wesentlich ausführlichere Liste wird von Bernhard Frötschtl [Frötschl[2]] bereitgestellt.

[2] Auflistung von deutschen TTS Systemen: http://www.8hertz.de/tts/tts.html

Tabelle 3: TTS Systeme

	SVOX	Festival[3]	Mary[4]	BOSS[5]	Sakrament TTS[6]	IMS[7]
Languages	Deutsch Englisch, Französisch	Englisch, Deutsch, Spanish	Deutsch	Deutsch	Englisch, Russisch	Deutsch
PDA	Ja (SVOX Smart)	Flite TTS Pocket PC implement-tation[8]	n.a.	n.a.	WinCE	n.a.
Automatische Intonation	Betonung und Tempo manuell kontrollierbar	Ja, sogar für Gesang	Ja	Ja	n.a.	Ja
Development kit	SVOX-SDK	Ja	Ja	Ja	n.a.	Ja
Bemerkung	Kommerzielle Variante des Systems der ETH Zürich. Kostenloses SDK mit Testlizenz erhältlich.	Sourcecode ermöglicht Kom-pilierung für PDA.	Java Applet zum Testen der Intonation, XML basierte Beschreibung der Intonation	Client-Server Architektur, C++	Läuft auf PDA	Deutsche Festival Variante, Teil von Smartkom

Am vielversprechensten für den mobilen Einsatz im deutschsprachigen Raum sieht nach Tabelle 3 das SVOX System aus. Mit SVOX smart bietet die SVOX AG eine speziell auf Windows CE optimierte Lösung an. Der Nachteil ist nur, dass es zu einem ein kommerzielles System ist und zum anderem keine automatische Intonation beherrscht. Die mobile Festival Variante Flite, war bei einem Test nicht überzeugend von der Sprachqualität, was aber an einer nicht optimierten Windows CE Version liegen kann. Dafür ist der Quellcode über eine Open-Source-Lizenz frei verfügbar. Eine andere Variante ist die Erzeugung von Audiodateien vor dem Ablauf des Systems. Das Speichern der Sprachausgabe in separaten Audiodateien beherrschen alle Systeme. Diese Variante ist nur sinnvoll bei wenig dynamischen Inhalten, da bei jeder Änderung die Datei neu erzeugt werden muss.

[3] Festival: http://www.cslu.ogi.edu/tts/demos/index.html
[4] Mary TTS: http://mary.dfki.de/
[5] BOSS: http://www.ikp.uni-bonn.de/dt/forsch/phonetik/boss/index.html
[6] Sakrament TTS PDA: http://www.sakrament-speech.com/products/tts/pda/
[7] IMS: http://www.ims.uni-stuttgart.de/phonetik/synthesis/synthesis_demo.html
[8] Flite PocketPC: http://www.viksoe.dk/code/flite.htm

2.3 Emotionen

In diesem Abschnitt soll das Zusammenspiel von Emotionen und Verhalten betrachtet werden. Dabei wird unterschieden werden, welche Arten von Emotionen es gibt und wie ein Modell für die maschinelle Repräsentation von Emotionen aussehen kann.

Aus der Psychologie gibt es drei bedeutende Emotionsmodelle, die zu zwei verschiedenen Klassen gehören:

- Kategoriale Modelle: Die Emotionen werden in diskrete Basis-Emotionen eingeteilt
  - OCC Model of emotions [Ortony 1988]: 22 Emotions-Typen (s. Tabelle 4)
  - P. Ekman emotion model [Ekman1992]: 6 Grundemotionen, Angst, Wut, Traurigkeit, Freude, Abneigung, Überraschung
- Dimensionale Modelle: Zwischen den Emotionen gibt es einen kontinuierlichen Übergang
  - Valence/ arousal model: Emotionen werden durch 2 Dimensionen charakterisiert. Positive/negatives Gefühl und Stärke des Gefühls. Dieses Model wird auch Circumplex-Modell genannt.[Russell 1980]

Tabelle 4: Positive und negative Emotionen [Otorny 1988]

Positive		Negative	
Happiness		Sadness	
Happy-for		Resentment	
Gloating		Pity	
Hope		Fear	
Relief		Disappointment	
Satisfaction		Fears-confirmed	
Pride		Shame	
Admiration		Reproach	
Gratification		Remorse	
Gratitude		Anger	
Love /Liking		Hate/Disliking	

Das OCC-Model von Otorny wird häufig verwendet um den emotionalen Zustand eines „life-like characters“ zu beschreiben. Dafür werden 11 positive und 11 negative Kategorien zur Aufteilung benutzt.

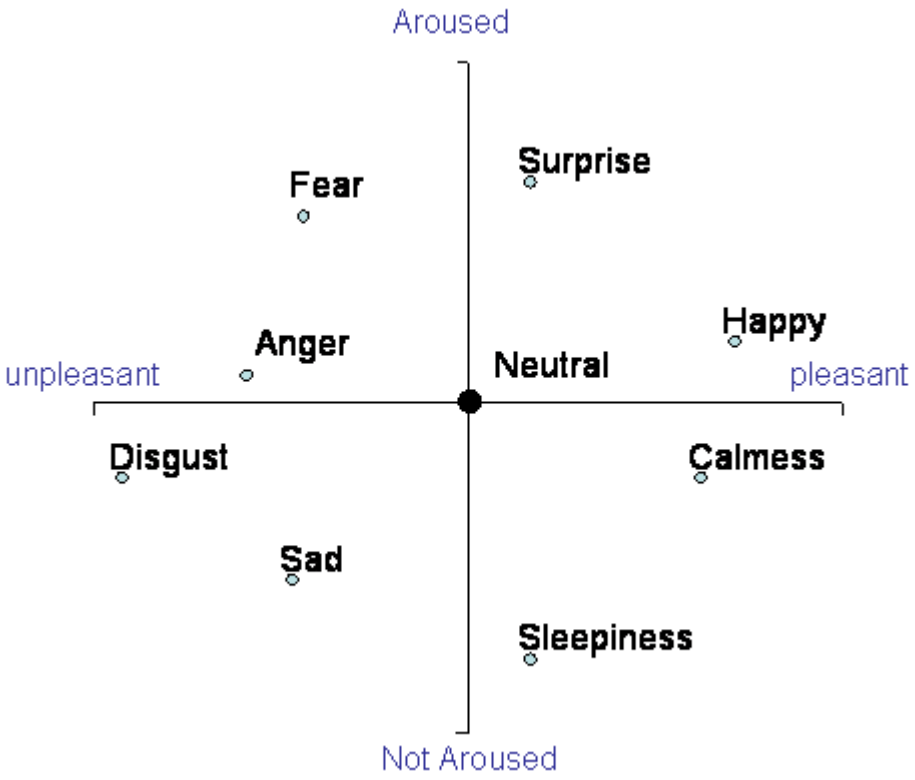

Abbildung 9: Circumplex-Model

In: Russell 1980

Das Circumplex-Model von Russell verteilt Emotionen in einem 2-dimensionalen Raum. Die horizontale Achse beschreibt das Empfinden der Emotionen von angenehm bis unangenehm. Auf der vertikalen Achse findet sich ein Faktor für die Erregtheit („aroused“). Im Gegensatz zu Emotionskategorien beschreibt der Faktor, wie stark Emotionen miteinander in Verbindung stehen. So stehen gegensätzliche Emotionen wie glücklich („happy“) und traurig („sad“) an gegenüberliegenden Enden.

Wie beeinflussen Emotionen das Verhalten? Nach Otorny [Otorny 1988] ist die direkte Relation zwischen Emotionen und Verhalten schwach. Die Verhaltensreaktion hängt vielmehr stark von der Intensität der Emotion ab. Die Emotionsintensität kann als nicht-lineare Funktion entsprechend Abbildung 10 dargestellt werden [Picard 2000]. Die doppelte Menge von positivem Input (wie z. B. schöner Musik) macht einen Menschen nicht doppelt so glücklich. Die Verhaltensreaktion hängt weiter ab von einer Intensitätsschwelle. Die Intensitätsschwelle kann sich dabei abhängig vom Kontext verändern. Außerdem muss die Intensitätsfunktion einen Grenzwert haben, da die Emotionsintensität nicht unendlich groß werden kann. Die Steigung der Intensitäts-Funktion hängt vom Charakter der Person ab. So wird eine aggressivere Person schneller von einem ruhigen Zustand in einen aggressiven Zustand wechseln.

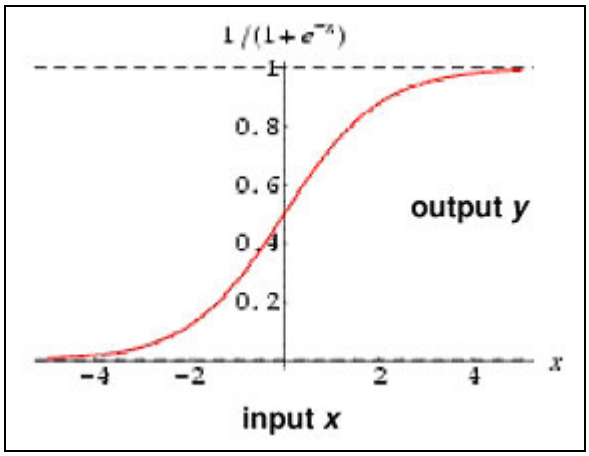

Abbildung 10: Funktion für Emotionsintensität

In: Picard 2000

Daher kann Verhalten abhängig von dem Überschreiten einer Emotionsintensitätsschwelle gemacht werden. Ein einfaches Emotionsmodell kann damit durch Regeln entsprechend dem folgenden Beispiel repräsentiert werden. Sobald ein Schwellwert überschritten wird, wird ein emotionaler Zustand gesetzt.

```
If  Freude(input_x) > Schwellwert_zufrieden THEN
setFreude(Zufrieden)
```

Passend zu dem emotionalen Zustand wird das Verhaltensskript für die Animation und die Betonung der Sprachausgabe angepasst. So können nach außen abfallende Augenbrauen und ein geschlossener Mund, Traurigkeit darstellen. Wogegen ein geöffneter Mund mit etwas hoch gezogenen Augenbrauen Freude ausstrahlt (Abbildung 11).

Abbildung 11: Traurigkeit und Freude

In: Bickmore 2002

Emotionen haben eine beschränkte Gültigkeitsdauer. Deswegen wird ein Intensitätsdegressionsfaktor eingeführt. Dieser reduziert die Emotion über die Zeit auf einen neutralen Wert.

Es gibt also verschiedene Parameter von denen das Aktivieren einer Emotion abhängt:

- Ein emotionaler Grundzustand (Ist eine Person eher fröhlich oder depressiv?).
- Die Steigung der Emotionsintensitätsfunktion.
- Der Schwellwert
- Der Degressionsfaktor (Abbau der Emotionsintensität).

Ushida, Hirayama und Nakajima [Ushida 1998] haben ein Emotionsmodell mit einer Emotionsintensitätsfunktion aufgestellt. Die Bestimmung der Emotion ist dabei ein mehrstufiger Prozess. Die aktuelle Situation wird abgebildet auf Emotionsfaktoren, die Eingangsfaktoren für die Intensitätsfunktion sind. Eine genaue Beschreibung des Modells würde hier zu weit führen, kann aber in [Ushida 1998] nachgelesen werden. Hier soll nur noch die Intensitätsfunktion in Abbildung 12 gezeigt werden. Wenn die Funktion einen bestimmten Schwellwert überschreitet und keine andere Intensitätsfunktion einen höheren Wert hat, dann ist die Emotion aktiv. In der Funktion ist γ der Degressionskoeffizient und δ der Emotionsfaktor.

$$E_i(t) = \frac{1}{1 + \exp\{(-X(t) + 0.5)/0.1\}}$$

$$X(t) = X(t-1) + \delta - \gamma + \sum_j W_{ji} E_j(t-1)$$

Abbildung 12: Emotionsintensitätsfunktion

In: Usdhida 1998

Für die Verwendung von Emotionen im Bereich der Mensch-Maschine-Kommunikation gibt es das Forschungsgebiet des Affective Computing[9]. Der Fokus liegt auf der Wahrnehmung (Sensorik) und Verwendung von Emotionen durch einen Computer. Ein Teilgebiet davon sind die Relational Agents, die auch zu den life-like characters gehören. Relational Agents haben das Ziel eine soziale/ emotionale Bindung zu dem Benutzer aufzubauen. Anwendungen sind dabei z. B. eine Unterstützung bei der Einübung neuer Verhaltensweisen über einen Zeitraum (z. B. mehr Sport zu treiben oder einen

[9] Affective Computing website: http://affect.media.mit.edu/

medizinischen Plan einhalten). Eine sehr umfassende Darstellung über Relational Agents findet sich in [Bickmore 2002].

2.4 Beschreibungs- und Skriptsprachen

Beschreibungs- und Skriptsprachen ermöglichen die manuelle Verhaltensplanung und bilden die Schnittstelle zur Ansteuerung einer Animations-Engine, die die Animationen eines life-like characters steuert. In diesem Abschnitt soll ein Vergleich der unterschiedlichen Sprachen gegeben werden, die bereits entwickelt wurden. Es gibt eine Reihe von Skript- und Beschreibungssprachen mit ähnlichen Zielen aber unterschiedlichen Ansätzen [Arafa 2003]. Nach Arafa können die Sprachen nach folgenden Kriterien kategorisiert werden:

- Ansatz: Was sollte mit der Sprache beschrieben werden? (Animation, Präsentation oder Dialog)
- Format: Welches Format wurde gewählt? XML-basiert?
- Spezifikationselemente: Welche Eigenschaften können beschrieben werden? (Umgebung, Stimme, Charakter, etc.)
- Animations-Kontrolle: Welche Kontrolle über die Animation ist möglich? (Synchronisation, Parametrisierung, etc.)
- Glaubwürdigkeits-Parameter: Kann dem Charakter „Tiefe“ gegeben werden? (Emotionen und Persönlichkeit)
- Character-Parts: Welche Körperbereiche werden beschrieben? (Gesicht, Körper und Sprache)

Hier sollen zwei unterschiedliche Sprachen, exemplarisch beschrieben werden. Eine ausführliche Übersicht wurde von Arafa [Arafa 2004] für das Buch „Life-Like Agents“ [Prendinger 2004] erstellt.

CML (Character Markup Language)

Das Entwurfsziel war die Definition von abstrakten Beschreibungen für Persönlichkeit, Emotion und Verhalten [Arafa 2002, 2003]. Dabei sollte die Kommunikation der Verhaltens- und Emotionsgenerierung mit der Animations-Engine standardisiert werden. Diese Beschreibungen werden abgebildet auf synchronisierte Animationen (Abbildung 13).

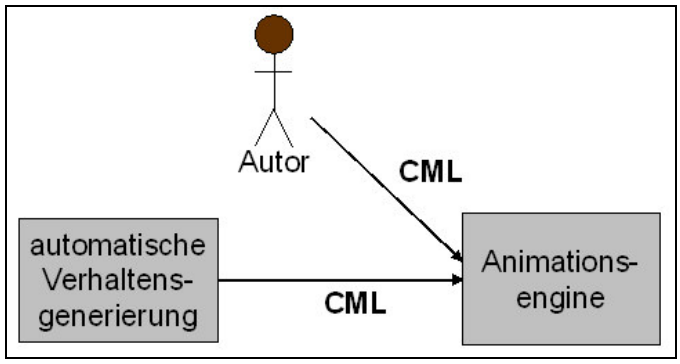

Abbildung 13: CML-Verwendung

Neues Verhalten kann aus der Kombination von Basisverhalten erzeugt werden. Es gibt Beschreibungen für alle von McNeill (s. Kapitel 2.2.1 „Gesten“) definierten Gesten. Bewegungen können von Emotionen beeinflusst werden (z. B. trauriges/ zufriedenes Laufen). Die Emotionskoeffizienten basieren auf dem Circumplex-Modell von Russell (s. Kapitel 2.3). Animationen sind sowohl für 2D als auch 3D Charaktere möglich. Für die Gesichtsanimation wurde der MPEG-4 Standard verwendet. Dadurch ist eine exakte Beschreibung von Mimiken möglich. Die Synchronisation zwischen Audio und Animation geschieht durch SMIL-Elemente, wodurch der sequentielle und parallele Ablauf beschrieben wird. Die Umgebung kann über Objekte mit einbezogen werden.

Der folgende Code zeigt ein Beispiel für ein CML-Skript. Der Charakter James soll zuerst zu einem Objekt mit dem Bezeichner „product1“ laufen. Nach dieser Aktion soll er parallel eine Zeigegeste auf „product1“ ausführen und „Dies ist Produkt 1“ sagen. Am Anfang der gesamten Handlung soll die Emotion „Zufriedenheit“ eine Intensität von 0.3 haben, die mit einem Faktor von 0.5 über die Zeit abnimmt.

```
<cml>
 <character name="James" personality="extravert" gender="m" base-
animation-file="butler.liv">
       <happy intensity="0.3" decay="0.5" priority="1">
             <move-to order="1" speed="default" object="product1">
             <sync type="par" order="1" priority="0">
                   <point-to object="product1"/>
                   <utterance>"Dies ist Produkt 1"</utterance>
             </sync>
       </happy>
 </character>
</cml>
```

BEAT (Beschreibungsprache)

BEAT ist nicht nur eine Sprache sondern ein gesamtes System zur automatischen Generierung von Gesten, Mimik und Intonation zu Text [Cassell 2004]. Dazu wird der Text über mehrere Stufen um Annotationen erweitert. In der letzten Stufe werden dem Skript genaue Zeiten für den Ablauf zugeordnet. Dieses Skript kann dann an ein

Animationssystem für die Ausgabe übergeben werden. Die Stärke von BEAT ist die Erweiterbarkeit. So kann relativ leicht neues Verhalten hinzugefügt werden. Das bedeutet auch, dass die Sprache nicht standardisiert ist. Emotionale Ausdrücke können nicht explizit angegeben werden. Dies ist bereits implizit in den Animationsanweisungen enthalten (z. B. durch eine spezielle Augenbrauenausrichtung). Der Vorteil der Sprache ist die intuitive Syntax. Es können Viseme, Blick, Gesten, Augenbrauen und Körperhaltung spezifiziert werden. Die Aktionen werden an einer absoluten Zeitleiste ausgerichtet.

Der folgende Beispielcode zeigt drei Animationen mit absolutem und Ereignis basierten Timing (s. Kapitel 2.5.1 „2-D-Animation“).

```
<VISEME time=0.0 spec='A'>
<GAZE word=1 time=0.0 soec=AWAY_FROM_HEARER>
<R_GESTURE_START word=3 time=0.517 spec=BEAT>
```

Es soll ein Viseme (s. Kapitel 2.2.3 „Sprache“) für den Buchstaben ‚A’ angezeigt werden. Zum gleichen Zeitpunkt soll sich der Blick weg vom Benutzer drehen. Beim Start des dritten Wortes oder zum Zeitpunkt 0.517 wird eine Beat Geste mit der rechten Hand animiert werden. (s. Kapitel 2.2.1 “Gesten”). An dem Beispiel ist erkennbar, dass BEAT eine einfache und intuitiv verständliche Syntax hat.

2.5 Charakter Animation

Durch die von einem Player (Animations-Darstellungs-Komponente) abgespielten Animationen werden die Gesten, Mimiken und Mundbewegungen dargestellt. Das interne Modell und die Verhaltensgenerierung können auf einem noch so hohen Niveau ablaufen, bringen aber wenig, wenn die Animationen schlecht sind. Die Animationen sind letztendlich das, was der Anwender zu sehen bekommt. Animationsmodelle können nach folgenden Merkmalen unterteilt werden:

- Erweiterbarkeit: Wie leicht können neue Animationen hinzugefügt werden?
- Modularität: Können neue Animationen aus bestehenden Basis-Animationen erstellt werden?
- Charakter-Modell: 2D, 3D-Modell oder kein Modell
- Animations-Modell: Eine limitierte Anzahl von Animationen vorgegeben oder freie Bewegungsmöglichkeiten?
- Anzahl der Übergänge zwischen Animationen: In wie vielen Posen kann der Charakter sich befinden? In wie vielen Zuständen können die Animationen anfangen und enden?

Wichtig für die Animation ist die Synchronisation der Sprachausgabe mit der restlichen Präsentation. Hier soll nur die 2D-Animation und die zugehörige Synchronisation betrachtet werden.

Weiter gibt es zwei Typen von Verhaltens-Animationen:

- Idle („untätig“, z. B. Augen drehen): Dies sind Aktionen zum Füllen von untätigen Zuständen. Diese Animationen sind wichtig um die Glaubwürdigkeit und Lebendigkeit des Charakters zu erzeugen.
- Directed („gerichtet“): Dabei soll durch das Verhalten eine spezifische Botschaft übermittelt werden (s. Kapitel 2.2).

Im Folgenden soll auf die oben genannten Merkmale in Bezug auf 2D-Animationen genauer eingegangen werden.

2.5.1 2D-Animation

Grundsätzlich gibt es zwei Methoden eine Animation aufzubauen:

- Frame-based (Einzelbild-basiert): Eine Animation besteht aus einer Anzahl von Bildern, die hintereinander abgespielt werden. Üblich sind etwa 16-25 Bilder pro Sekunde (16-25 fps).
- Time-based (Zeit-basiert): Ein Objekt, das time-based animiert ist, bewegt sich von einem Punkt A nach Punkt B immer in der gleichen Zeit, unabhängig von der Bildrate. Eventuell macht das Objekt dadurch Sprünge, falls das Programm die nötige Bildrate nicht berechnen kann.

Die meisten bei „life-like characters“ verwendeten Animationssysteme basieren auf einem 2D-Modell, das eine limitierte Anzahl von Basisanimationen verwendet. Mit einer Basisanimation ist hier ein Abschnitt gemeint, der nicht weiter unterteilt werden kann (z. B. eine Idle-Geste wie das Wackeln mit dem Fuß). Um weiche Übergänge zwischen Basisanimationen zu ermöglichen, müssen sie in definierten Zuständen starten und enden, diese Zustände werden häufig als „Pre-/Post-Conditions“ bezeichnet [Lester 1997]. Die Animation besteht damit aus einer Grafik und einer Bewegung. Bei time-based Animationen kann zwischen keys und in-betweens unterschieden werden. Keys sind festgelegte Zustände einer Animation und in-betweens die Übergänge zwischen den Zuständen. Der Computer interpoliert die Übergänge automatisch. Dieser Vorgang wird als tweening bezeichnet. Bei Vektor-basierten Formaten wie SVG und Flash reicht es, die keys zu definieren. Bei einem traditionellen Bitmap-Format wie z. B. Gif muss die gesamte Animation vom Designer Bild für Bild erstellt werden. Hier sollen im Weiterem nur noch Vektorformate betrachtet werden. Die Grafiken orientieren sich meistens an 2D Comic Charakteren. Gerade für pädagogische Systeme, die häufig Kinder als Zielgruppe haben, hat sich das als praktikable Lösung gezeigt [Shawn 2004].

Durch die Zusammenstellung der Animationen wird eine Animationsbibliothek erstellt. Sinnvoll ist eine Unterteilung in eine Standardanimationen-Bibliothek und eine Domain spezifische [Shawn 2004]. Es gibt üblicherweise mindestens folgende Animationen als Teil einer Basis-Animations-Bibiliothek: Idle, Gesten, Mund, Augen, Augenbrauen, Bewegung und Körperhaltung. Eine Standardanimationen-Bibliothek enthält Animationen die in unterschiedlichen Kontexten verwendet werden können, wie z. B. winken und zeigen. Dagegen stehen in der Domain spezifischen Bibliothek Animationen die nur in einem spezifischen Kontext verwendbar sind, das kann z. B. eine „Rührbewegung“ für einen Koch sein.

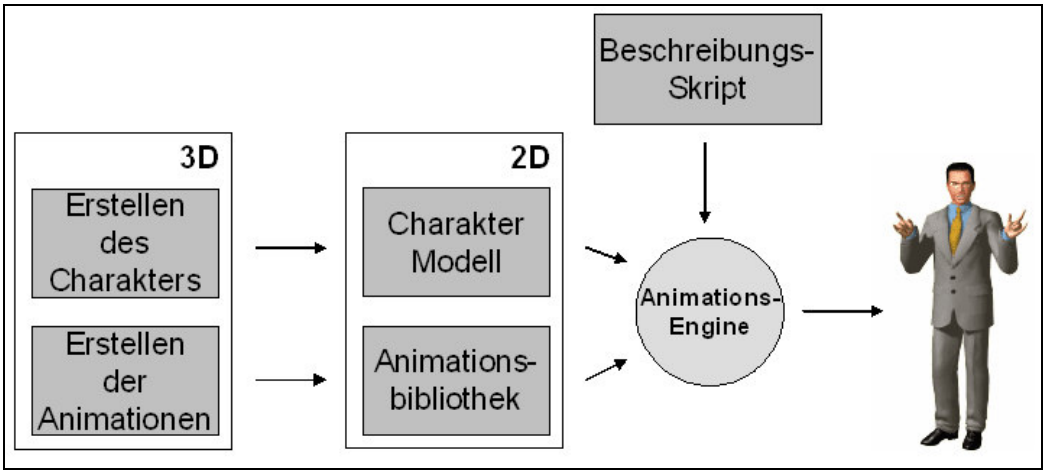

Abbildung 14: Workflow Animation

Eine Herausforderung ist die 2D-Darstellung von menschlichen Bewegungen. 3D-Bewegungen in Richtung des Benutzers oder vom Benutzer weg müssen ebenfalls als 2D-Animationen dargestellt werden. Die manuelle Erstellung ist je nach Detailgrad sehr aufwändig. Deswegen werden häufig die Charakter-Modelle zuerst in einem 3D-Animationsprogramm erstellt und dann in ein 2D-Vektorformat (wie z. B. Flash oder SVG) exportiert. So wurde die Figur Laura [Bickmore 2003] mit der Hilfe von Poser (einem Werkzeug für die 3D-Charaktererstellung, s. Tabelle 4) erstellt und dann nach Flash exportiert. Die Figur Skip [Shaw 2004] wurde mit Maya (ebenfalls ein Werkzeug für die 3D-Modellierung, s. Tabelle 4) erstellt. Ein üblicher Ablauf ist in Abbildung 14 beschrieben. Der Charakter wird in einem 3D Programm modelliert. Danach werden mit Hilfe des Modells die Animationen erstellt. Das Charakter-Modell und die Animationen werden separat als 2D-Versionen gespeichert. Die Animations-Engine verwendet das Modell und die Animationen, um das vom Autor erstellte Beschreibungsskript grafisch darzustellen.

Tabelle 5: Übersicht 2D/3D-Animationssoftware

	Poser[10]	Swift 3D[11]	Blender[12]	Toon Studio[13]	Maya 3D[14]
SVG/ Flash Renderer	Nicht direkt, aber Flash, Tools für Konvertierung existieren	Ab Version 4.5 SVG-Export, sehr guter Flash Renderer	Anscheinend möglich, keine Informationen ob mit Animationen	Flash und Plazmic Mobile SVG[15]	Export nach SVG und Flash.
SVG Input	Nein	Ja	SVG path import mit Python Skript	Nein	Nein
Open Source	Nein	Nein	Ja	Nein	Nein
2D/3D	3D	3D	3D	2D	3D
Bemerkung	3D-Animation von Figuren ist der Hauptfokus. Einsatz von physikalischen und biomechanischen Modellen.	Fokus auf Web-Vektoranimation.	Open-Source 3D-Renderer. Sehr mächtig, hoher Lernaufwand.	Fokus ist die 2D Animation von Zeichentrick-szenen.	Sehr mächtiger 3D-Renderer.

In Tabelle 5 ist eine Übersicht von Systemen zur Unterstützung bei der Erstellung der Figur und Animationen von 2D/3D-Charakteren aufgelistet. Der Fokus wurde dabei auf die Eigenschaft gelegt, ob das Rendern in eines der 2D-Vektorformate SVG oder Flash möglich ist. Poser ist das am meisten auf die Erstellung von grafischen 3D-Figuren und Animationen spezialisierte Produkt. Durch fertige Basismodelle und integrierte physikalische und biomechanische Modelle können Animationen schnell erstellt werden. Swift 3D's Spezialität ist die Erstellung von 3D-Animationen für Flash. Blender und Maya3D sind mächtige 3D-Modellierungswerkzeuge, die kaum oder wenig Unterstützung für Flash oder SVG haben. ToonStudio's Fokus liegt auf der Erstellung von Zeichentrickfilmen. Wegen der leichten Bedienbarkeit von Poser ist es das für den Designer komfortabelste Produkt, das Problem ist, dass die Ausgabe nur nach Flash möglich ist. Es ist möglich eine Flash-Animation bildweise in eine SVG-Animation zu übertragen. Die resultierende Animation ist also frame-based. Diese Methode wurde für den Charakter Gina (s. Kapitel 3.2) verwendet.

[10] Poser 6: http://www.e-frontier.com/go/products/poser
[11] Swift 3D: http://www.erain.com/products/swift3d/
[12] Blender 3D: http://blender3d.org
[13] Toon Boom Studio: http://www.toonboom.com/products/toonBoomStudio/
[14] Maya: http://www.alias.com/glb/eng/products-services/family_details.jsp?familyId=3900009
[15] Plazmic Mobile SVG: www.plazmic.com/

In Abbildung 14 ist der Workflow für die Erstellung der Animationen bis zur Verwendung für den Charakter dargestellt. Der Charakter hat hier getrennte Beschreibungen für die Animationen und die Figur. Zuerst muss der Designer den Charakter erstellen. Hier soll der Grund-Charakter in 3D sein, dadurch sind Gesten wesentlich leichter zu modellieren. Zu dem Charakter werden Basis-Animationen entworfen, die später den Beschreibungen der Skriptsprache entsprechen. Die Basis-Animationen und der Charakter werden in das 2D-Format transformiert. Für die eigentlichen Animationssequenzen müssen die Basisanimationen von der Animations-Engine den Informationen aus dem Beschreibungsskript zugeordnet werden.

2.5.2 Koordination von Animationen

Für die Koordination, wann eine Animation startet, gibt es drei Verfahren:

- Time-based: Jede Basisanimation bekommt eine absolute Startzeit zugeordnet.
- Event-based: Der Anfangszeitpunkt hängt von einem Event, wie z. B. dem Ende der vorausgehenden Animationssequenz, ab. Dadurch können alle Animationen relativ zu der Startanimation beschrieben werden.
- Time und Event based: Es können auch beide Koordinationsarten verknüpft werden. Das bedeutet, eine Animation startet zu einem bestimmten Zeitpunkt, aber die darauf folgenden Animationen werden alle durch Ereignisse aktiviert.

Der Vorteil der time-based-Animationskoordination ist die Einfachheit. Während des Abspielens sind keine Koordinationsmechanismen nötig. Der Nachteil ist, dass Änderungen innerhalb der Animation schwierig sind. Dies ist genau der Vorteil der Event-based-Koordination. Animationssegmente können ausgetauscht werden, ohne das die Gesamtanimation neu erstellt werden muss.

2.5.3 Layered-Modell

Ein geschichtetes Modell einer Animationsfigur verteilt die Animationen auf mehrere Ebenen. Dies hat den Vorteil, dass bei einer Animation nur die Ebene neu gezeichnet werden muss, in der die Veränderung stattfindet. Der Nachteil ist, dass Abhängigkeiten zwischen Körperteilen nicht beachtet werden (wie z. B. Verbindung von Arm und Oberkörper). In Abbildung 15 ist ein solches Modell dargestellt. Es gibt 2 Schichten mit getrennten Animationen, den Körper („Torso“) und die Arme und den Kopf („Arms & Head“). Zusammengefügt entsteht daraus der gesamte Körper („Full body“). Daher muss die „Arms & Head“ Schicht für jeden unterschiedlichen Zustand der „Torso“-Schicht neu geladen werden. Dieses Modell wird für das System Laura verwendet [Bickmore 2003]. Ein weiterer Vorteil ist, dass es relativ schnell für eine kleine Anzahl von Animationen implementierbar ist.

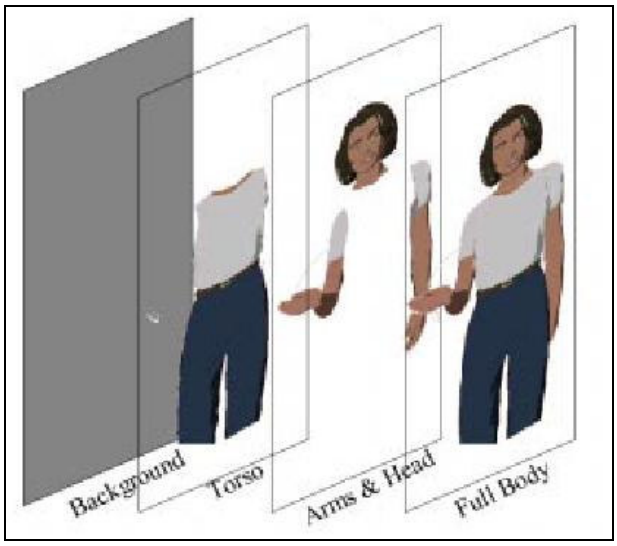

Abbildung 15: Layered Modell

In: Bickmore 2002

2.5.4 Skelett-basiertes Modell

Ein Skelett besteht aus starren Elementen und Gelenken. Im Gegensatz zu einem absoluten Koordinatensystem für das gesamte Skelett hat jedes Gelenk ein eigenes lokales Koordinatensystem. Die Koordinatensysteme sind in eine Baumstruktur sortiert. So ist z. B. die Hand Teil des Unterarms, der wiederum Teil des Oberarms ist. Die lokalen Koordinatensysteme lassen sich durch Matrixmultiplikationen entlang des Pfades des Baumes erzeugen (z. B. Hand = Hüfte * Körper * Oberarm * Unterarm). Die Bewegung von Punkten im Raum wird gemäß der Kinematik beschrieben. Die Vorteile eines Skelett-Modells sind:

- Veränderungen der Position eines Körperteils übertragen sich auf die Unterkomponenten (Eine Verschiebung des Oberkörpers, verschiebt auch die Arme).
- Es können leicht Bedingungen für Gelenke angegeben werden (z. B. der Armwinkel darf nicht größer 180 Grad sein).
- Eine Pose kann durch die Koordinaten des Wurzelelements (z. B. der Hüfte) und die Winkel der einzelnen Gelenke beschrieben werden.

2.5.5 Animationsübergänge zwischen Animationsabschnitten

Angenommen, es wurde eine Animationsbibliothek mit einer beschränkten Anzahl von Animationen erstellt. Anfangs- und Endpositionen einer Animation heißen Pose. Die Animationen haben Vor- und Nachbedingungen, die für eine weiche Animation erfüllt sein sollten. Hierbei entsteht ein Problem. Wie kann man von einer Pose zur nächsten Anfangspose kommen? Dieses Problem wird umso schwieriger, je mehr Posen es gibt.

Die Basis-Animationen können als Übergänge in einem Graphen betrachtet werden und die Posen als Zustände. Dadurch sollte sich ein zusammenhängender Graph wie in Abbildung 16 bilden. Diese Graphen werden Bewegungsgraphen (Motion Graphs) [Kovar 2002] genannt. Um die Komplexität der Bewegungsgraphen zu reduzieren, verwenden viele Animationssysteme nur Animationen mit nur einem oder sehr wenigen Übergängen. Bei nur einem Zustand kann dieser auch als neutraler Zustand bezeichnet werden, in den die Figur oder das Körperteil nach jeder Animation zurückkehrt.

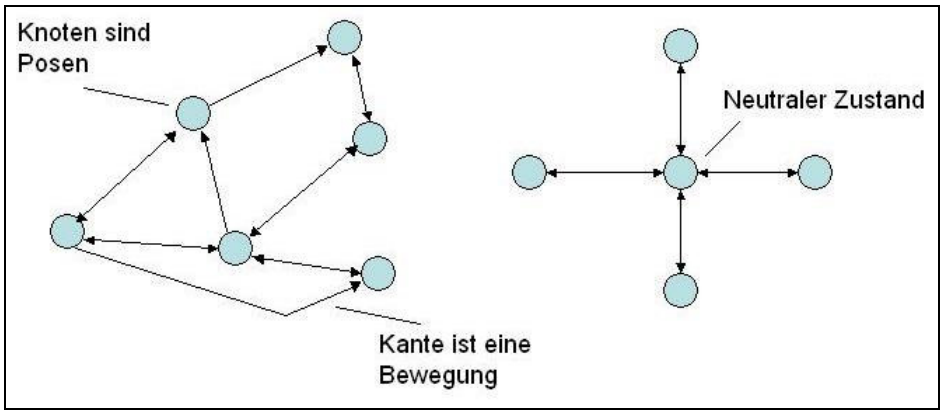

Abbildung 16: Bewegungsgraphen

Um das Problem der Übergänge zu lösen, muss der kürzeste Pfad zwischen 2 Posen innerhalb des Graphen gefunden werden. Der kürzeste Pfad ist aus zwei Gründen sinnvoll. Zu einem müssen Zeitkriterien bis zum Startzeitpunkt der Folgeanimation eingehalten werden. Zusätzlich entspricht es der menschlichen Bewegung zwischen zwei Gesten nur eine kurze Übergangsbewegung zu machen. Für das Suchen des kürzesten Pfads innerhalb eines Graphens gibt es z. B. den Dijkstra Algorithmus mit einer Laufzeit von $O(n^2)$. Allerdings wird n (Anzahl der Posen) nicht sehr hoch sein, da der Designer alle Posen und Übergänge (d. h. Animationen) zwischen Posen erstellen müsste.

2.5.6 Synchronisation von Animation und Sprache

Zur Synchronisation von Animation und Sprache gibt es zwei Möglichkeiten:

- Absoluter Zeitplan: Auf der Basis von Wort und Phoneme Dauern wird ein Ablaufplan mit absoluten Zeiten für die gesamte Animation erstellt bevor die Animation startet. Es gibt also nur eine Synchronisation beim Start der Animation und Sprachausgabe.
- Ereignis-basierter Ablaufplan: Die Animation-Engine wartet bis ein bestimmtes Wort gesprochen wird und startet dann die Animation. Dafür sind Echtzeit-Benachrichtigungen, welches Wort gerade gesprochen wurde, der TTS (Text-to-Speech) Engine nötig.

Der absolute Zeitplan ist einfacher zu implementieren, der Nachteil dabei ist die fehlende Synchronisation während der Laufzeit. Es kann passieren, dass durch unterschiedliche Ablaufszeiten, Animation und Sprachausgabe asynchron laufen. Dadurch entsteht für den Betrachter ein unnatürlicher Effekt.

2.6 Kontext

Dem Gebiet des „context-aware computing“ ist in den letzten Jahren einige Bedeutung zugekommen. Besonders im Zusammenhang mit mobilen und adaptiven Systemen ist das Ausnutzen von Kontextinformationen besonders relevant. Grund dafür ist eine zunehmende Beherrschbarkeit der Komplexität von Anwender-Systemen, die bessere Performance der Systeme und eine bessere Verfügbarkeit der für kontext-berücksichtigende Systeme notwendigen Sensorik. Das Ziel des Einsatzes von Kontextinformationen sind Systeme, die besser an den individuellen Benutzer angepasst sind, wodurch für den Anwender ein höherer Nutzen und eine vereinfachte Bedienung eines Systems erzielt werden. Im folgenden wollen wir den Begriff Kontext genauer erklären.

Kontext ist allgemein betrachtet die Umgebung bzw. der Zusammenhang in dem sich ein Objekt befindet. Das entspricht der Definition des Begriffes Kontext in Merriam-Webster’s Collegiate Dictionary[16]:

„Context is defined as the interrelated conditions in which something exists or occurs.”

Das Problem ist, dass diese Definition sehr vage bleibt. Dazu kommt noch, dass der Begriff in unterschiedlichen Fachgebieten mit einer anderen Bedeutung verwendet wird. So bedeutet Kontext im Bereich der Künstlichen Intelligenz [Liebermann 2000] etwas anderes als bei Betriebssystemen, Programmiersprachen oder in der Literaturwissenschaft. Im Rahmen dieser Arbeit ist besonders der Kontextbegriff im Bereich des „Mobile Computing“ relevant. Um das Wort Kontext etwas weniger abstrakt zu betrachten, soll zuerst der Begriff anhand von einem Beispiel dargestellt werden.

Beispiel: PDA im Wald

Ein Besucher ist mit seinem PDA auf einer Tour durch ein Waldgebiet. Der PDA kann automatisch Informationen zu den Baumsorten im Umkreis anzeigen. Dabei werden die Informationen abhängig von dem Datum und der Uhrzeit angepasst. So wird im Herbst auf

[16] Merriam-Webster’s Collegiate Dictionary: www.m-w.com

andere Eigenschaften des Baums aufmerksam gemacht als im Frühling. Bestimmte Gebiete des Waldes dürfen besonders im Frühling nicht betreten werden. Sobald der Benutzer eines dieser Gebiete betritt, erfolgt eine Warnung. Bei anderen Gebieten erfolgt eine Vorsichtsmeldung, aber das Betreten ist erlaubt. Wenn der Benutzer häufig Informationen zu einer bestimmten Sorte von Bäumen abfragt sollen ihm auch ähnliche Bäume automatisch angezeigt werden.

Was fällt dabei auf? Einmal die räumliche Umgebung, ein Waldgebiet mit unterschiedlichen Baumsorten und Zonen mit verschiedenen Zutrittsrechten. Weiterhin werden das Datum und die Tageszeit verwendet. Der PDA muss entscheiden, welche Baumsorten in seiner Umgebung sind. Ebenso müssen Datum und Uhrzeit bekannt sein. Informationen über die vom Besucher betrachteten Bäume müssen gespeichert werden.

Hierbei kann man bereits wichtige Eigenschaften des Kontextbegriffes erkennen. Die Bäume müssen erkannt werden, also braucht das System entweder eigene oder in der Umgebung installierte Sensoren. Dafür sind optische Sensoren denkbar. Ebenso könnte bereits die Position aller Bäume in einer Datenbank enthalten sein, die dann mit der Position des Geräts verglichen wird. Dafür wäre eine Komponente nötig, die die aktuelle Position liefert. Eventuell sind die angezeigten Bauminformationen falsch, wenn ein Baum nicht korrekt identifiziert wurde. Die gemessenen Werte müssen erst interpretiert werden. So muss die Position mit dem Datum kombiniert werden, um festzustellen ob eine Zone zum aktuellen Datum betreten werden darf. Das System muss auf das Eintreten von bestimmten Kontexten, wie z. B. das Erkennen eines Baums, reagieren können.

In dem Beispiel können einige Begriffe herausgestellt werden:

- Sensoren: erfassen Eigenschaften der Umgebung (Kontextinformationen).
- Kontextabhängige Aktion: Reaktion des Systems auf den Kontext.
- Kontextverarbeitung: für die Interpretation von Zusammenhängen (z. B. Regeln).
- Kontext kann kombiniert werden, um neuen Kontext zu erzeugen.
- Kontext hat eine gewisse Fehlerwahrscheinlichkeit.

Kontextumgebung

Kontext ist zunächst einmal alles das, was das System umgibt: das ist der allgemeine Kontext oder die Umgebung. Es ist nötig zwischen relevantem und unrelevantem Kontext zu unterscheiden. Das passiert durch eine geeignete Auswahl von Sensoren. Hier wird also schon gefiltert, welche Zustände der Umgebung (Kontextinformationen) in das Kontext-System eingehen sollen. Diese Filterung der Gesamtinformationen durch ausgewählte Sensorik oder Kontextinformationserfassung ist in Abbildung 17 zu sehen.

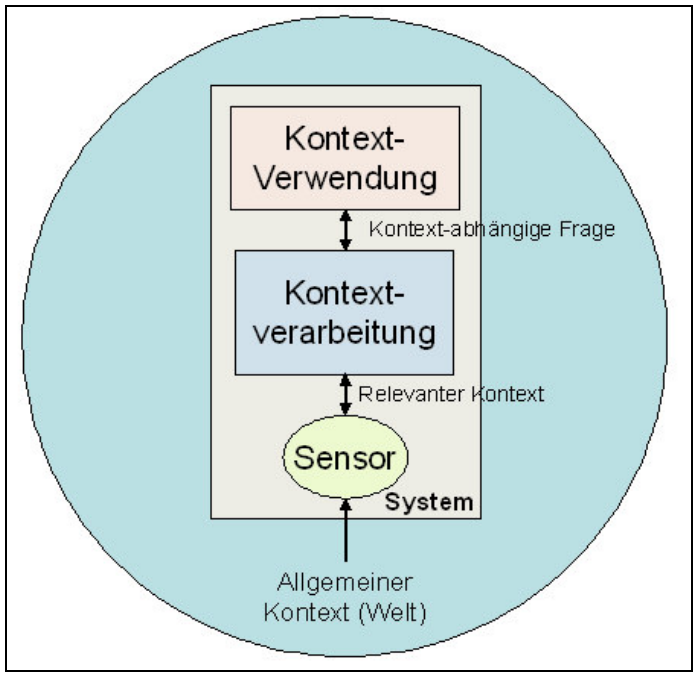

Abbildung 17: Allgemeiner Kontext und relevanter Kontext

An diesem Bild wird eine generelle Schwierigkeit beim Kontextbegriff deutlich. Während es für den Menschen vollkommen normal ist, eine Vielzahl von Kontextinformationen zu einer Zeit bewusst oder unbewusst zu verarbeiten, muss die Verarbeitung in einem technischen System viel expliziter durchgeführt werden. Anforderungen an ein Kontext-abhängiges System sind schnell formuliert, aber die Realisierung ist sehr schwierig, da für alle notwendigen Kontextinformationen zunächst einmal eine geeignete Erfassung der Kontextinformation realisiert werden muss. So kann ein Mensch schnell implizit erkennen, dass eine Person an einer Bushaltestelle steht, weil sie wahrscheinlich auf den Bus wartet. Ein Computersystem braucht hier bereits entweder eigene Sensoren oder eine geeignete Infrastruktur mit Sensoren, die überhaupt erst mal erfasst, dass eine Person an einer Bushaltestelle steht. Hierzu benötigt das System Definitionen über Bushaltestellen und Menschen. Eventuell ändert sich die Interpretation auch abhängig von der Zeit. Was passiert, wenn die Person nachts an der Bushaltestelle steht? Wartet Sie auch dann noch auf den Bus oder auf einen Bekannten, der sie abholt?

Es ist wichtig, zwischen dem Objekt, das vom Kontext umgeben ist, dem Kontext selbst und der Frage, die durch den Kontext beantwortet werden soll, zu unterscheiden.

So ist z. B. die Frage „Hat der Benutzer einer Führung Station 4 erreicht?“ nur im Kontext der Führung, in dem der Benutzer sich befindet, beantwortbar. Bei dieser Frage ist:

- Das Objekt der Benutzer.
- Die kontextabhängige Frage „Hat der Benutzer eine Station erreicht?“.
- Die relevanter Kontextinformation zur Beantwortung der Frage ist die augenblickliche Position des Benutzers.
- Außerdem ist zusätzliches Wissen nötig: Station 4 befindet sich an der geografischen Position X, Y.

Angenommen der Benutzer trägt einen PDA bei sich, der einen GPS Empfänger hat. Dadurch ist die aktuelle Position des Benutzers feststellbar. Mit einem einfachen Regelsystem könnte in diesem Fall die Frage beantwortet werden. Bei den meisten mobilen Systemen ist es unnötig zwischen dem Benutzer und dem System zu unterscheiden, da der Benutzer direkt das System mit sich trägt. Die GPS-Koordinate des Systems entspricht also auch dem Ort des Benutzers.

Nach Einführung der Thematik werden nun einige Kontextdefinitionen näher betrachtet.

Kontextdefinitionen

In [Schmidt 2002] werden vier Charakteristiken für Kontextdefinitionen unterschieden:

- Scope: Ist ein spezieller Kontext (wie z. B. Ort) oder ein allgemeiner Kontext gemeint („die Welt“).
- Seperation: Wie stark ist die Trennung zwischen Kontext und Applikation und die Trennung zwischen Kontextermittlung und Kontextbereitstellung?
- Abstraction: Ist der Kontext spezifisch für eine Applikation oder generell verwendbar?
- Relation: Auf was ist der Kontext bezogen? Die Anwendung, das Gerät, den Benutzer?

Man kann beobachten, dass der Kontextbegriff sich über die Zeit weiterentwickelt hat. Dabei war die Entwicklung meistens abhängig von neuen Sensoren und neuen Anwendungsbereichen für kontextbasierte Systeme.

Als einer der ersten versucht Schilit [Schilit 1994] den Kontextbegriff über Beispiele zu definieren. Danach können drei verschiedene Kontextklassen unterschieden werden:

- „Computing context“: Drucker, Monitor, Netzwerk Anbindung.
- „User context“: Benutzer Profil, Ort, Leute in der Umgebung.
- „Physical context“: Helligkeit, Geräusche, Temperatur

Zusätzlich wurde von Chen und Kotz [Chen 2000] vorgeschlagen,

- „Time context“: Tageszeit, Monat, Jahr

als zusätzlichen Kontext hinzuzufügen.

In Cheverst [Cheverst 1998] wird unterschieden zwischen:

- personal context: Benutzerinteressen, finanzielle Möglichkeiten, aktueller Ort
- environmental context: Tageszeit, Öffnungszeiten von Attraktionen, Wetter.

Interessant ist bei Cheverst, dass Kontext eine Gültigkeitsdauer hat. Davon hängt auch die Frage ab, wie häufig der Kontext neu bestimmt werden muss. So werden die meisten „personal contexts“ (Benutzerkontexte) bei einem mobilen System während der Anwendung konstant bleiben, wogegen der Ort sich ständig verändert.

Häufig werden für mobile Projekte hauptsächlich Ort, Zeit und das User Profile (Benutzerinteressen, finanzielle Möglichkeiten, etc.) als Kontext verwendet [Chen 2000]. Der Ort ist der wichtigste Kontext für mobile Anwendungen, da er sich während der Benutzung ständig ändert und relativ einfach zu bestimmen ist. Die Position ist heutzutage in vielen Situationen über einen handelsüblichen GPS Empfänger ermittelbar. Deswegen ist die Verarbeitung des Ortkontextes auch relativ gut verstanden. Neben GPS gibt es noch eine Reihe von anderen Technologien, um die Position zu messen. Schmidt [Schmidt 2002] hat eine Auflistung von unterschiedlichen Technologien zusammengestellt.

Chen und Kotz haben eine praktische und allgemeine Kontextdefinition aufgestellt.

> “Context is the set of environmental states and settings that either determines an application’s behaviour or in which an application event occurs and is interesting to the user.”

Speziell für mobile Kontexte wurde von Abowd [Abowd 2002] ein mehrschichtiges Kontextmodell vorgeschlagen (Abbildung 18). Bei einem pragmatischen Kontexteinsatz

liegt die Schwierigkeit meistens darin, die relevanten Kontexte auszuwählen, die am besten den Einsatzbereichs des Systems abdecken. Deswegen war das Ziel des Modells eine möglichst strukturierte Darstellung für die Auswahl von Kontexten zu ermöglichen. Kontext wird in einer Hierarchie in immer spezifischere Kontexte aufgeteilt.

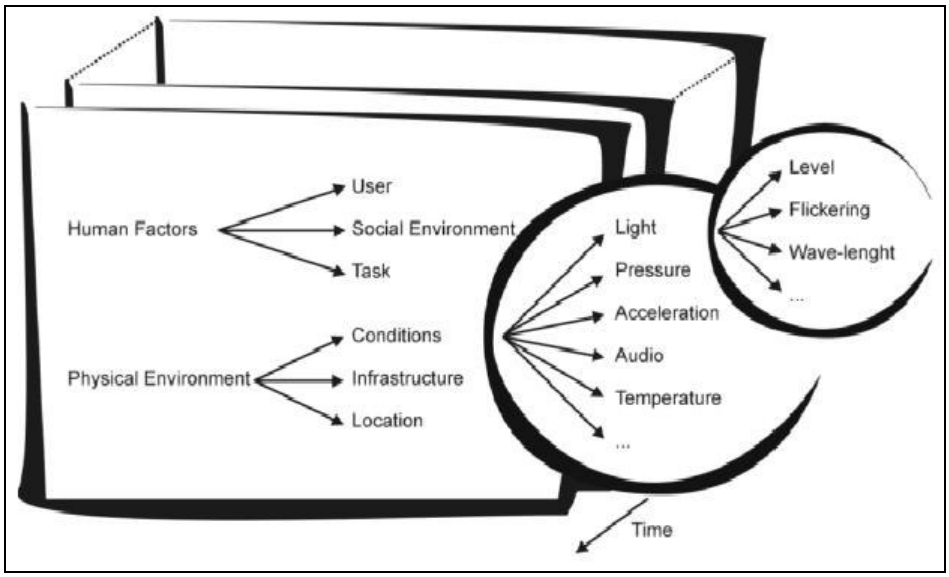

Abbildung 18: Kontexthierachie

In: Schmidt 2002

In Tabelle 6 wurden Kontextbeispiele und mögliche Aktionen zusammengestellt, die für einen pädagogischen Agenten relevant sein könnten. Dabei verwenden die meisten entweder den Ort oder das Nutzerprofil. Da ein elektronischer Naturführer ortsabhängige Informationen präsentieren soll (s. Kapitel 1), muss bekannt sein, wo das Gerät sich befindet. Dadurch kann eine Beziehung zwischen präsentierten Informationen und der Natur erreicht werden. Dabei ist ebenso der Benutzerkontext wichtig, da die Informationen zusätzlich auf den Erfahrungshintergrund und das persönliche Interesse des Benutzers zugeschnitten sein sollen.

Tabelle 6: Übersicht von relevanten Kontexten für einen pädagogischen Agenten

Kontext-Kategorie	Kontext	Sensor Input	Agenten Verhalten
Ort/Position	Benutzer bewegt sich.	Position ändert sich.	Benutzer schaut wahrscheinlich gerade nicht auf das Display, also kein Verhalten nötig.
Nutzerprofil (Demographie)	Benutzer ist 42 Jahre alt.	Benutzer-Profil-Abfrage beim Start des Programms.	Inhalte auf das Alter anpassen. Hier Inhalte für einen Erwachsenen anzeigen.
Ort/Position	Benutzer ist an einem POI.	Position bleibt über einen Zeitraum konstant, Position entspricht einem bestimmten Ort.	Präsentation von Inhalten zu diesem POI.
Sozialer Kontext	Benutzer ist in einer Gruppe.	Abfrage beim Start des Programms, Geräuschsensor, Kamera.	Aufforderung zum Vorlesen (falls keine Sprachausgabe), Gruppenspiele, Gruppenaktionen.
Ort/Position, Nutzerprofil (Interesse/ Kenntnisse)	Benutzer bewegt sich schnell.	Position ändert sich stark.	Benutzer hat eventuell wenig Zeit. Eventuell Inhalt auf das Wesentliche reduzieren. (Mit vorheriger Abfrage).
Nutzerprofil (Interesse/ Kenntnisse)	Benutzer hat hohen Kenntnisstand.	Abfrage beim Start oder Test.	Komplexere Inhalte präsentieren.
Nutzerprofil (Kontexthistorie)	Benutzer hält sich nur kurz an den Stationen auf.	Position, Zeit, Kontextgeschichte.	Abfragen, ob präsentierte Inhalte tatsächlich den Benutzer-interessen entsprechen.
Nutzerprofil (Kontexthistorie)	Benutzer hat Verständnis-Schwierigkeiten .	Texte werden gelesen. Aufenthaltsdauer an Station ausreichend lang, aber Quiz und Feedback negativ.	Inhalte anpassen. Niveau reduzieren. Mehr Grundlagen vermitteln.
Nutzerprofil (Kontexthistorie)	Benutzer hat Motivationsprobleme.	Texte werden kurz oder nicht gelesen. Aufenthaltsdauer an der Station kurz.	Mehr Spiele präsentieren. Kürzere Texte und eventuell mehr Grundlagen. Alter abfragen. Falsche Altersgruppe eingestellt?

Höherwertige Kontexte

Kontext kann mit anderen Kontexten oder Informationen zu höherwertigen Kontexten interpretiert werden. Bei Chen [Chen 2000] wird zwischen „Low-Level“ und „High-Level“-Kontexten unterschieden. So kann der Orts-Kontext und der Zeit-Kontext zu einem gemeinsamen Kontext der Aufenthaltsdauer an einem Ort kombiniert werden. Von Abowd [Abowd 1999] wird dies auch als Kontextabstraktion bezeichnet. Häufig ist es für Anwendungen sinnvoller bereits interpretierte Informationen zu erhalten. So kann es für einen „Tour guide“ praktischer sein, bereits Straßen oder Gebäudenamen statt GPS-Koordinaten zu erhalten.

Kontextgeschichte

Wird der Kontext über einen Zeitraum beobachtet, entsteht eine Kontextgeschichte, die wiederum als Kontextquelle verwendet werden kann. So kann die Aufenthaltsdauer eines Benutzers über eine Reihe von Stationen etwas über das Interesse des Benutzers an einzelnen Stationen oder an der Tour generell aussagen. In Abbildung 18 ist die Zeit daher bewusst orthogonal zu den restlichen Kontexten eingetragen.

Kontextdynamik und Kontextgenauigkeit

Seltener in der Literatur beschriebene Kontexteigenschaften sind die Kontextdynamik und Kontextgenauigkeit. Beide Eigenschaften werden bisher kaum in Systemen berücksichtigt.

- Häufigkeit der Änderung: Wie häufig ändert sich der Kontext während der Interaktion mit dem Gerät? Bestimmte Kontexte wie das Benutzerprofil bleiben während der Anwendung konstant, wogegen Kontexte wie die Position dynamisch sind [Abowd 1999]. Dabei ist auch die Frage interessant, ob der Kontext kontinuierlich oder diskret erfasst wird? [Brown 1997].
- Genauigkeit: Wenn ein Sensor einen Wert aus der Umgebung misst, kommt es üblicherweise zu Abweichungen vom tatsächlichen Wert. So hat das GPS-Signal eine Ungenauigkeit von 5-10 m, je nach System. Diese Ungenauigkeit besteht auch im Kontext weiter. Gerade bei Anwendungen, die höherwertige Kontexte erzeugen, kann es von Vorteil sein, eine Wahrscheinlichkeit für die Korrektheit des Kontextes einzufügen [Abowd 1999].

Context-Toolkit

Ein bereits sehr weit entwickeltes Context-Framework ist das „Context-Toolkit“ von Salber, Dey und Abowd [Salber 1997]. Von der Idee her, entspricht es den von grafischen Oberflächen bekannten GUI-Widgets.

Interessant ist dabei die Trennung von:

- Kontextgewinnung (Sensor): Eingang in das Kontextmodell. Noch nicht interpretierter Kontext (Temperatur, geografische Position, ...).
- Kontextverwendung (Applikation): Die Applikationslogik ist getrennt von der Kontextverarbeitung.
- Kontext-Interpretation: Kontext wird durch die Kombination von mehreren Kontexten und eventuell zusätzlichen Faktenwissen auf eine höhere Abstraktionsebene transformiert.
- Kontext-Aggregation: Mehre Kontexte, die logisch zusammengehören, bekommen eine gemeinsame Repräsentation (z. B. der Kontext von einem Raum).

Kontext-Widgets kapseln die Informationen zu einer Kontext-Eigenschaft und ermöglichen einen „query“-(„Nachfrage“) und „notify“ („Benachrichtigung“)-Mechanismus. Dadurch wird die Komplexität der eigentlichen Kontextgewinnung vor der Anwendung „versteckt“. Durch die Abstraktion erhält die Anwendung genau die für sie notwendigen Informationen. Ein weiterer Vorteil ist, dass Kontext-Widgets ähnlich wie GUI-Widgets in anderen Anwendungen wieder verwendet und dadurch Kontextbibliotheken erzeugt werden können. Dadurch soll eine leichtere Entwicklung von „context-aware“-Anwendungen realisiert werden. Tabelle 7 zeigt ein Beispiel für ein Kontext Widget aus [Abowd 1999]. Dargestellt ist ein Widget, das einen Raum überwacht und eine Applikation benachrichtigt, sobald eine Person den Raum betritt oder verlässt.

Tabelle 7: Kontext Widget für die Raumüberwachung

Widget Class	IdentityPresence
Attrributes	
Ort ID Uhrzeit	Ort den das Widget beobachtet ID des letzten Benutzers Ankunftszeit es letzten Benutzers
Callbacks	
PersonKommt (ort, id, uhrzeit) PersonGeht (ort, id, uhrzeit)	Ausgelöst wenn Person Raum betritt Ausgelöst wenn Person Raum verlässt

Kontextgewinnung

Kontext kann meistens auf zwei Arten gewonnen werden: Zum einem autonom von dem Gerät selber und zum anderem über eine Kontext-Infrastruktur. Als Beispiel soll wieder ein Benutzer mit einem PDA im Wald dienen. Das System soll ihm Informationen zu den Bäumen in seiner Umgebung liefern. Dies kann z. B. durch einen pädagogischen Agenten geschehen, der abhängig vom Kontext diese Informationen zusammenstellt und präsentiert. Nun gibt es zwei Varianten, wie das System diesen Kontext erfahren kann. Das Gerät kann mit eigenen Sensoren ausgestattet sein, durch die es in der Lage ist die Baumart zu erkennen. Vorteil dieser Lösung ist die Flexibilität und Unabhängigkeit von einer Infrastruktur. Das Problem liegt in der Technik. Ein System zu bauen, das diese technischen Kriterien erfüllt ist schwierig. Die zweite Variante ist eine Kontext Infrastruktur aufzubauen. So könnte jeder Baum mit einem RFID[17]-Chip ausgerüstet sein. Entweder enthält der Chip direkt die gesuchten Informationen oder eine ID für eine weitergehende Suche in einer Datenbank. Die Möglichkeiten sind dabei vielfältig. Der Vorteil ist, dass das eigentliche mobile Gerät weniger technisch aufwendig ausgerüstet sein muss. Das Problem hierbei ist, dass damit eine Abhängigkeit zu der Infrastruktur entsteht.

[17] RFID: Radio Frequency Identification. Kleine und preiswerte Chips die über Funk ausgelesen werden können.

Adaptive und Context-aware Computing

Zum Ende dieses Kapitels soll noch kurz der Unterschied und die Überschneidung der Begrifflichkeiten von Adaptive und Context-aware computing betrachtet werden. Diese beiden Themen sind sehr eng miteinander verbunden und werden in der Literatur zum Teil nicht eindeutig getrennt.

So hat Schilit [Schilit 1994] folgende Definition für Context-aware Software eingeführt:

> "Context aware software adapts according to the…environment and react to changes to the environment."

Weiterhin gibt es in Chen, Kotz [Chen 2000] die Definition:

> "Active context awareness: an application automatically adapts to discovered context, by changing the applications behaviour. "

In beiden Definitionen passt sich die Anwendung aktiv an den Kontext an. Eine Unterscheidung zwischen Kontext-Verarbeitung und Kontext-Verwendung (wie z. B. bei den Kontext-Widgets) ist häufig softwaretechnisch sinnvoll. Adaptive Computing ist an der Verwendung von Kontext-Informationen interessiert; diese Informationen werden von der Kontext-Komponente bereitgestellt. Beide Themen ergänzen sich. Bei der Adaption des Systems soll das System speziell an die Bedürfnisse von Benutzern oder auch anderen Systemen angepasst werden. Dazu werden Kontextinformationen benötigt.

In diesem Kapitel wurde weniger auf konkrete Kontext-basierte Anwendungen eingegangen. Detaillierte Übersichten von Projekten im Umfeld des Context-aware Computing finden sich in [Chen 2000] und [Schmidt 2002].

2.7 Verhaltensplanung

Ein Agent braucht eine Steuerung, die beschreibt, was in einem bestimmten Zustand passieren soll. Eine Steuerung kann auf zwei Arten realisiert werden: zum einen die automatische Erstellung von Präsentationsskripten oder die manuelle Erstellung durch den Autor. Bei den manuellen Systemen muss jedes Verhalten vom Autor spezifiziert werden. Manuell beschriebene Puppets existieren von Oddcast, Haptek und Microsoft [Oddcast, Haptek, MSAgent].

Die manuelle Erstellung wurde im Abschnitt „Beschreibungssprachen" betrachtet. Das manuelle Erstellen von Skripten ist aufwendig und fehleranfällig. Daher gibt es mehrere

Versuche diesen Schritt ganz oder teilweise in den Agenten zu verlagern. Letztendlich ist es das Ziel dem Autor die Arbeit zu erleichtern.

Nach André [André 1999] kann das Verhalten eines life-like character durch folgende Formel beschrieben werden:

Persona behaviour := directives + self behaviour

Anweisungen (directives) kommen von dem Autor oder dem System, das den Charakter verwendet. Eigenes Verhalten (self behavior) sind „Idle"-Aktionen und -Reaktionen auf Benutzeraktionen. Dementsprechend kann die Präsentationsplanung in zwei Komponenten getrennt werden:

- Presentation Engine: Was soll dargestellt werden?
- Puppet Engine: Wie soll es dargestellt werden?

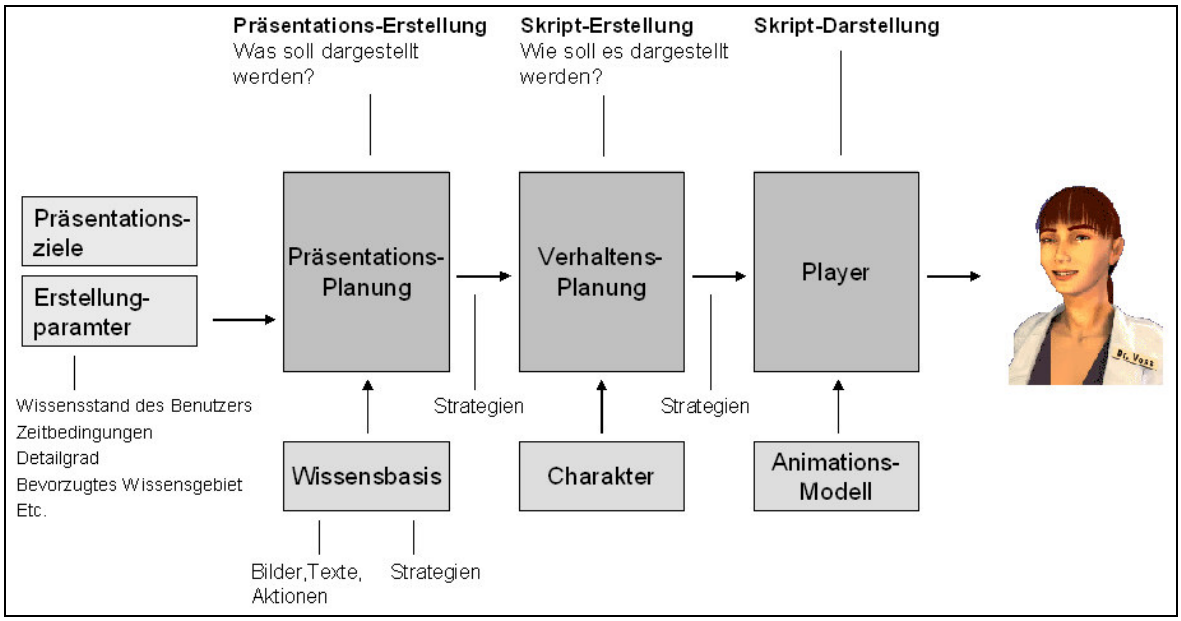

Abbildung 19: Systemübersicht mit Präsentations-,Verhaltensplanung und Player

Dies führt zu einer Architektur, wie sie in Abbildung 19 dargestellt ist. Die Presentation Engine (oder Planungs-Komponente) bekommt Ziele und Erstellungsparameter übergeben. Ziel kann z. B. die Vermittlung von Wissen über Laubbäume sein. Parameter sind der aktuelle Wissensstand des Benutzers und Zeitbedingungen, die von der Präsentation erfüllt sein müssen (z. B. der Benutzer hat nur 10 Minuten Zeit). Aus einer Wissensbasis mit Materialien für diese Präsentation wird ein Drehbuch (Ablaufplan) erzeugt. In einem zweiten Schritt muss dieses Drehbuch auf den konkreten Charakter abgebildet werden. Dabei ist entscheidend welches Verhalten der Charakter anbietet und wie das Drehbuch auf

das Verhalten abgebildet werden kann. Dabei wird auch ein Zeitplan für den Ablauf erstellt. Der Player übernimmt die Darstellung der Animation. Die dafür benötigten Animationen werden über das Animationsmodell bereitgestellt.

2.7.1 Rekursive hierarchische Planung

Der Präsentationsagent PPP-Persona verwendet einen Planer basierend auf einem rekursiven hierarchischen Planer [André 1999]. Das Erstellen einer Präsentation ist ein zielgerichtetes Handeln (z. B. Präsentation von Informationen über Laubbäume). Das eingehende abstrakte Präsentationsziel wird in Teilziele aufgelöst (z. B. Einleitung, Hauptteil, Überleitung etc. präsentieren). Diese werden wieder in Teilziele zerlegt, bis den Teilzielen konkrete Aktionen zugeordnet werden können. Eine Aktion ist z. B. das Photo von einer Eiche anzeigen. Das Problem hierbei ist die Erzeugung einer rhetorischen und zeitlichen Struktur der Präsentation. Präsentationsabschnitte sind semantisch eng verbundene Themen. Die rhetorische Struktur kann in einem gerichteten azyklischen Graphen, wie in Abbildung 20 dargestellt werden. Das Wurzelelement ist das abstrakte Ziel (z. B. Information vermitteln), die Knoten sind Teilziele und die Blätter konkrete Aktionen (z. B. Photo anzeigen, Text anzeigen, Geste ausführen). Die zeitliche Struktur beschreibt den Ablauf der Präsentation entlang einer Zeitachse.

Die Erstellung und Präsentation von Information erfolgt unter bestimmten Präsentationszielen (wie z. B. “Zusammenfassung präsentieren“) und Bedingungen (Zeitvorgabe, Wissenstand, etc.).

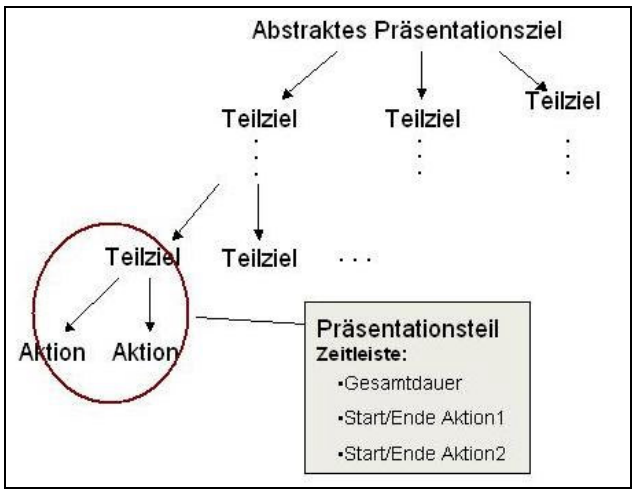

Abbildung 20: Graph mit Präsentationszielen

Einzelne Präsentationsabschnitte (das unterste Teilziel) sind Zustände innerhalb einer Präsentation. Abbildung 21 zeigt einen Zustandsautomat der durch die Präsentationsteile gebildet wird. Die Zustandsknoten bestehen aus einem Abschnitt zusammen mit der Gesamtdauer. Die Zustandsübergänge werden entweder durch das Ablaufen einer Zeit, einer

Benutzeraktion (z. B. „Next“-Klick) oder Kontextereignisse ausgelöst. Wird ein Zustandsübergang während der Darstellung eines Abschnitts ausgelöst, dann wird die Darstellung bei einer Rückkehr in diesen Zustand an dieser Stelle fortgesetzt (z. B. wenn der Benutzer zusätzliche Informationen zu einem Abschnitt verlangt und danach wieder die eigentliche Präsentation fortsetzt) (Abbildung 21).

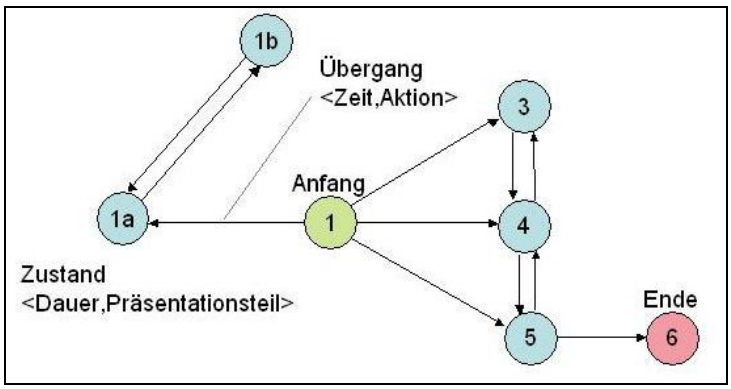

Abbildung 21: Zustandsautomat für Präsentationsteile

Für das Erstellen einer Präsentation werden so genannte Strategien verwendet. Eine Strategie beschreibt, durch welche Unterziele und Aktionen ein Ziel erfüllt werden kann. Außerdem sind zeitliche Kriterien angegeben (z. B.: Bild wird angezeigt bevor die Sprachausgabe startet). Der Planer sucht die Strategie für ein bestimmtes Ziel und versucht die Strategie zu erfüllen. Das Ergebnis des Planers ist ein Drehbuch, das die mögliche Abfolge von Präsentationsabschnitten beschreibt. Der nächste Schritt ist die Zuordnung von Aktionen für die grafische Figur.

2.7.2 Kontext-sensitive Planung der Animation

Der Animations-Planer (Script Engine) entspricht hier einem Regisseur, der ein Drehbuch bekommt, in dem steht, welche Objekte (Akteure) es auf der Bühne gibt und wie die Objekte miteinander interagieren, aber das dafür noch keinen genauen Ablauf definiert. Das Drehbuch kommt von der Presentation-Engine (s. Abbildung 19). Wie der Regisseur ist daher der Animations-Planer dafür zuständig, den Akteuren genau mitzuteilen, wann und wo die Aktionen stattfinden sollen. Eine Beispiel für einen Animations-Planer ist die PPP-Persona [André 1999].

> Therefore, the operators are formulated from the point of view of a director who orchestrates the interplay of the character with the display of all other media objects. [Andre 1999]

Eine Kontext-sensitive Geste wie „zeigen“ muss erst an die konkrete Bildschirmaufteilung angepasst werden. Dafür ist es eventuell noch nötig, move-to (Positionsänderungs-) Aktionen hinzuzufügen (z. B. wenn der Charakter erst eine Position erreichen muss, bevor die Zeigeaktion angezeigt werden kann). Hierzu stehen Animationsdefinitionen zur Verfügung, die Vor- und Nachbedingungen haben. Dadurch sollen die Übergänge zwischen Animationen ermöglicht werden. Zusätzlich können Bedingungen angegeben werden, in welchem Zustand der Charakter gerade sein muss (z. B. iconified, zoomed, etc.)

```
(defprimitive bottomupjumping
 :pre ((leftarm standard)(rightarm standard)
          (iconified no)(bodydir front)
          (bodypos stand)(stick off))
 :post((posy -=1))
 :gesture 42)
```

In dem oberen Codebeispiel sind die pre- und post-Bedingungen für die Animation hochspringen („bottomupjumping“) beschrieben. Der linke und rechte Arm muss vorher in einer Standard-Position sein. Zusätzlich muss der Charakter nach vorne gedreht (bodydir front), stehend (bodypos stand) und der Zeigestab muss ausgeschaltet sein (stick off).

Eine Besonderheit der PPP-Persona ist, dass alle möglichen Animationsübergänge vorher in einen endlichen Zustandsautomat geladen werden. Dadurch können mögliche Folge-animationen während der Laufzeit schnell gefunden werden (vgl. Kapitel 2.5.1 “2D-Animation“). Das System wurde von Shawn [Shawn 2004] um Wahrscheinlichkeiten für die Animationen erweitert. Es gibt mehrere Animationen für die gleiche Aktion. Die dargestellte Animation wird nach der Wahrscheinlichkeit ausgewählt. Dadurch gibt es weniger Wiederholung bei gleichem Verhalten und der Charakter bekommt eine höhere Natürlichkeit. Die Darstellung der Aktion erfolgt im folgenden Schritt durch einen Player, der das erstellte Animationsskript grafisch darstellt.

2.7.3 BEAT

Hier liegt der Fokus auf der automatischen Generierung von Animationen zu einem Text. Dies sind also Gesten, Mimiken und weitere Verhaltensweisen, die gemeinsam mit gesprochener Sprache verwendet werden. Kontext-sensitive Animationen können über Objekte (z. B. Image, TextBox) berücksichtigt werden. Grundlage ist die Extraktion von linguistischen Informationen aus dem Text. Eine Stärke des Systems ist der modulare und erweiterbare Aufbau. So können relativ einfach neue Verhaltensgeneratoren bzw. Filter hinzugefügt werden. Das System ist in der Lage, verschiedene Skripte zu erzeugen und ist dadurch gut für neue Charaktere erweiterbar. Die gesamte Datenverarbeitung basiert auf XML-Dokumenten. Das Programm selber ist in Java implementiert.

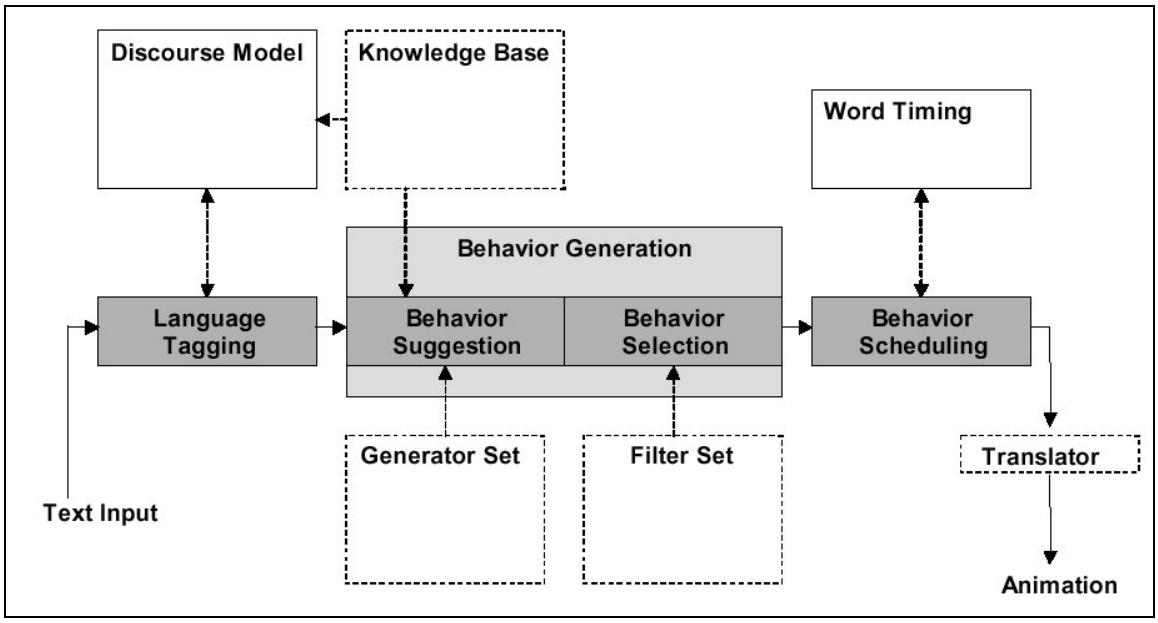

Abbildung 22: BEAT architecture

In: Cassell 2001

In Abbildung 22 ist die Architektur des BEAT-Systems dargestellt. Hier ist gut die modulare Struktur erkennbar. Im ersten Schritt, dem so genannten „Language Tagging" werden Meta-Informationen über die Inhalte der Sprache in den Text eingefügt. Dabei werden Sätze in „Theme" und „Rheme" unterteilt (s. Kapitel 2.2.3 „Sprache"). Über die Wissensbasis („Knowledge Base") stehen semantische Informationen zur Verfügung. Damit können auch Kontraste gekennzeichnet werden (gut … böse, schwarz … weiß). Die Wissensbasis verwendet WordNet[18] für die semantischen Informationen. Als nächster Schritt, werden Regeln zur Verhaltensgenerierung („Behavior Suggestion") ausgewertet. Dadurch entsteht eine Überproduktion von Vorschlägen für das Verhalten, die anhand von Konflikten und Prioritäten gefiltert werden. So können z. B. alle Gesten unterhalb einer bestimmten Priorität herausgefiltert werden, um einen lethargischen Charakter zu imitieren. Der abschließende Schritt ist das Zuordnen von Startzeiten zu dem generiertem Verhalten.

2.8 Zusammenfassung und Vergleich

Im diesem Kapitel wurde eine große Bandbreite an Forschungsgebieten und Technologien vorgestellt, die im Bereich der Pädagogischen Agenten oder „life-like characters" relevant sind. Hier soll jetzt betrachtet werden, welche Verfahren für eine spätere Verwendung innerhalb der Architektur besonders interessant sein könnten.

[18] WordNet online lexical reference system: http://wordnet.princeton.edu/

Die skelettbasierte Animation, die bisher hauptsächlich in 3D-Animationen eingesetzt wird, bietet einen interessanten Ansatz zur Realisierung von Animationen in 2D-Vektorformaten. Dieses Verfahren reduziert den Aufwand der Animationserstellung und ermöglicht die Trennung von Animationen und Texturen. Der Nachteil ist die Abbildung von 3D-Bewegungen auf eine 2D-Darstellung.

Bei den Kontextverfahren ist besonders das Konzept der Kontext-Widgets interessant. Das Verfahren der Interpretation und Aggregation ist gut durchdacht und nicht auf spezielle Kontexte festgelegt. Das Verfahren bietet sich besonders im mobilen Bereich an, da bereits im Konzept der Kontext-Widgets die Verarbeitung besonders für mobile und verteilte Systeme bedacht wurde.

Aus der Betrachtung der Verhaltens-Beschreibungssprachen ist deutlich geworden, dass die Sprachen unterschiedliche Anwendungsfälle abdecken und dementsprechend komplex sein können. Die BEAT-Sprache zeichnet sich besonders durch die einfache und intuitive Syntax aus, was ein großer Vorteil für den Autor ist, der die Sprache zur Beschreibung seiner Texte verwenden muss. Zusätzlich steht mit dem BEAT-System ein gut erweiterbares System zur automatischen Generierung von Verhaltensbeschreibungen zur Verfügung. Der Nachteil des Systems ist, dass es bisher nur auf Englisch verfügbar ist.

Bei der Untersuchung von unterschiedlichen Ansätzen der multimodalen Kommunikation ist klar geworden, wie vielfältig dieses Thema ist. Die grafische Repräsentation des Charakters muss die Fähigkeit anbieten, eine möglichst große Anzahl von Verhaltensweisen auf Animationen abzubilden, um dem Autor ein breites Spektrum an Verhaltensbeschreibungen anzubieten. Dafür sollte eine erweiterbare Animationsbibliothek zur Verfügung stehen.

Die Aufteilung des Systems in Puppet und Planer ähnlich wie bei der PPP Persona ist sinnvoll, da beide Komponenten unterschiedliche Aufgaben haben (Darstellung und Planung) und durch eine Trennung auch austauschbar sind. Diese Aufteilung sollte daher in der Architektur berücksichtigt werden.

Von den TTS-Systemen scheint SVOX und das deutsche Festival System IMS für den Einsatz im MobiNaf Projekt am vielversprechensten, da von beiden sowohl deutsche Spracherzeugung als auch PocketPC-C++ -APIs angeboten werden.

Im folgenden Kapitel werden die hier eingeführten Begriffe und Kenntnisse auf existierende Systeme angewendet. Dabei sollen Vor- und Nachteile der Systeme im Bezug auf die Aufgabenstellung dieser Arbeit beschrieben werden.

Kapitel 3: Existierende Systeme

Es gibt bereits eine große Anzahl von Implementierungen von verschieden Varianten von „life-like characters“. In diesem Kapitel sollen vier davon ausführlich vorgestellt werden. Die hier präsentierten Systeme kommen aus unterschiedlichen Bereichen und haben verschiedene Schwerpunkte. Laura und Gina sollen das Verhalten der Benutzer über einen längeren Zeitraum beeinflussen und beobachten; damit gehören sie sowohl in den Bereich der Relationalen als auch der Pädagogischen Agenten. Herman-the-bug ist einer der frühsten Pädagogischen Agenten. SmartKom ist ein sehr aufwändiges Projekt, bei dem der Agent nur ein Bereich einer Reihe von Forschungsthemen ist. Übergeordnetes Thema von SmartKom ist die multimodale Mensch-Maschinen-Kommunikation in unterschiedlichen Anwendungskontexten.

Danach folgt eine Tabelle mit kurzen Beschreibungen zu weiteren „life-like characters“-Projekten. Die Systeme werden auf ihre Anwendbarkeit für das MobiNaf-Projekt betrachtet.

3.1 Laura

Laura wurde 2001 als Teil der Doktorarbeit von T. Bickmore [Bickmore 2003] über Relationale Agenten entworfen. Der Fokus war, die Beziehung der Anwender zu dem Agenten zu erforschen. Über einen längeren Zeitraum wurde eine Gruppe von Studenten von Laura bzgl. ihres täglichen Sportprogramms „betreut“. Laura ist eine als Flash-Applikation realisierte Figur, die ein von BEAT [Cassell 2004] erzeugtes Animations-Skript darstellen kann. Das BEAT-System von Cassell und Bickmore kann automatisch formale Beschreibungen von Gesten, Mimiken und Intonation zu Text erzeugen [s. Kapitel 2.7.3]. Diese können dann über die Flash-Applikation abgespielt werden. Für die möglichst natürliche Kommunikation mit den Benutzern wurde für Laura ein aufwendiges Dialogsystem entwickelt. Das System verwendet ein Benutzerprofil, um die Relation mit dem Benutzer über einen längeren Zeitraum zu verfolgen. Laura hat nur eine beschränkte Animationsbibliothek zur Verfügung. Die einzelnen Animationen wurden dafür umso aufwendiger speziell für diesen Anwendungsfall ausgearbeitet. Das System wurde in Java und XML als Client-Server-Architektur erstellt (Abbildung 23). Die grafische Darstellung erfolgt in einem Browser, der mit der eigentlichen Applikation über ein Netzwerk verbunden ist.

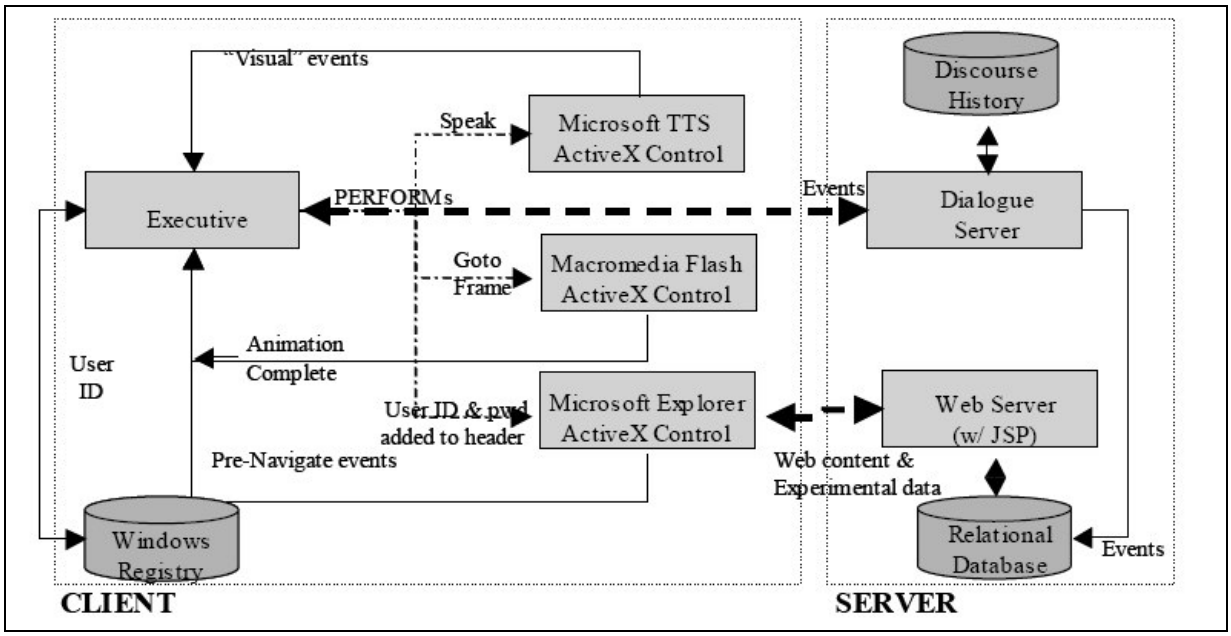

Abbildung 23: Laura Client-Server Architektur

In: Bickmore 2002

3.2 Gina

Gina ist ein pädagogischer Agent aus dem Jahr 2004. Der Anwendungsfokus liegt hier ähnlich wie bei Laura auf der Vermittlung von Informationen durch wechselseitige Interaktion zwischen Agenten und Anwender über einen längeren Zeitraum. Der Agent ist Teil eines PDA-basierten Systems zur Unterstützung von Müttern von Krebspatienten. Ziel ist es, die meistens nicht mit Computern vertrauten Personen in dem richtigen Umgang mit dem System zu schulen. Dafür verwendet das System die grafische Repräsentation einer Frau, die vom Oberkörper aufwärts dargestellt ist. Der Agent beobachtet dabei, wie vertraut ein Anwender mit dem System ist. Dementsprechend werden die Hilfestellungen erweitert oder reduziert. Gina ist in Flash und C# auf einem Dell Axim X3 PocketPC implementiert worden. Das System basiert auf dem Digital Puppet System von Shaw [Shaw 2004]. Das System ermöglicht eine schnelle Erstellung von Charakteren, basierend auf einem bewährtem Workflow. Die Animationen sind in Szenen aufgeteilt, die immer am Stück abgespielt werden. Die Charaktere werden in Maya erstellt und dann in Flash frame-based-Animationen (s. Kapitel 2.5.1) gespeichert. Die Animationen sind jeweils Animationssequenzen, die an einem Stück abgespielt werden. Die Sequenzen sind auf spezielle Hilfestellungen betreffend der Benutzung des Geräts ausgelegt. In Abbildung 24 ist eine Darstellung von Gina auf dem PDA Bildschirm zu sehen. Die Ausgabe von Texten erfolgt sowohl über das Abspielen von Audiodateien als auch das Einblenden von Texten.

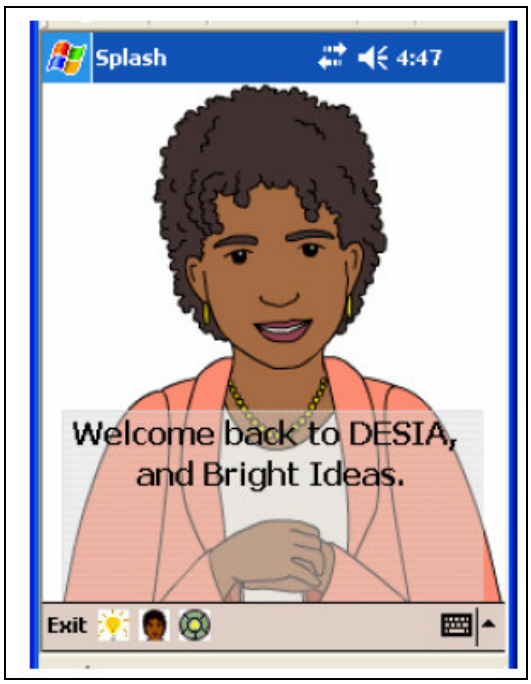

Abbildung 24: Gina

In: Johnson 2004

3.3 Herman-the-bug

„Herman-the-bug“ ist der pädagogische Agent der „design-a-plant“-Lernumgebung. Der Agent wurde bereits 1996 erstellt. Ziel ist es, Schülern Wissen über botanische Anatomie und Physiologie zu vermitteln. Dabei sollen Pflanzen für spezielle Umgebungen von den Schülern erzeugt werden. Der Agent erstellt dafür abhängig vom Benutzerprofil und der aktuellen Situation automatisch passende Hilfestellungen und Hinweise. Die Steuerung des Agentenverhaltens basiert bei Herman-the-bug auf der „coherence-structured behavior sequencing engine“. Grundlage ist ein digitales Drehbuch mit untereinander vernetzten Abspielszenen („networked storyboard“), basierend auf 30 fertig animierten Sequenzen und 160 Audioclips. Animationen wurden zu Sequenzen zusammengefasst, die immer einen speziellen thematischen Bereich der Lerninhalte abdecken. Dabei sollten Sequenzen immer mit dem gleichen Bild anfangen und enden, um Sprünge zwischen Animationen zu vermeiden. Dadurch sind diese Sequenzen beliebig kombinierbar. Ansonsten gibt es eine Metrik für die Übereinstimmung von zwei Bildern, d. h. es sind auch Sequenzen kombinierbar die nicht genau zusammen passen. Sequenzen sind weiter durch einen ontologischen Index kategorisiert. Die Sequenz-Engine wertet diesen Index aus, um das Verhalten für den aktuellen Lern-Kontext auszuwählen. Durch dieses spezielle Design ist der Agent sehr domain-spezifisch und nicht einfach in andere Bereiche transformierbar. Die Animationssequenzen wurden für spezielle Lerninhalte erstellt und entsprechen kurzen animierten Filmen. In Abbildung 25 ist der Verhaltensraum („Behavior Space“) mit den Animationen und Audioausgaben abgebildet. Die Behavior Sequencing Engine wählt entsprechend dem aktuellen Kontext eine passende Animation aus. Ein Kontext könnte z.

B. eine falsche Antwort in einem Test sein. Der Agent entscheidet daraufhin, ob im aktuellen Kontext ein Hinweis für die richtige Antwort dargestellt werden soll.

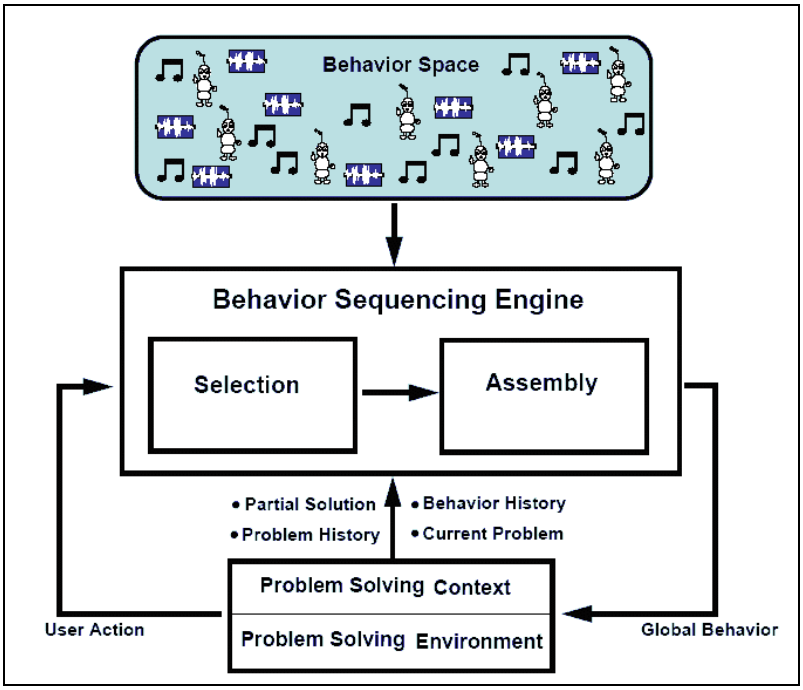

Abbildung 25: Architektur von Herman-the-Bug

In: Stone 1996

3.4 SmartKom

SmartKom ist ein multimodales Dialog-System, dass Gesten, Sprache und Mimik verstehen und verwenden kann. Das Projekt läuft seit 1999 am Deutschen Forschungszentrum für Künstliche Intelligenz (DFKI) [Wahlster 2003]. Der Anwender soll Aufgaben an einen life-like character delegieren können. SmartKom definiert drei Anwendungszenarien: mobil, home und public. Das mobile Szenario beschäftigt sich mit dem Einsatz als Begleiter im Auto oder für Fußgänger. Dabei wird im Wesentlichen der Ortskontext verwendet, um Dienstleistungen wie Navigationshilfen zu generieren. Abhängig vom Kontext werden auch die Ein-/Ausgabe-Modalitäten automatisch justiert. So ist im Auto die Sprache die Hauptkommunikationsform, wogegen in einem Raum mit vielen Personen die Kommunikation über das Display abläuft. In Abbildung 26 ist der Agent in einer kleinen Darstellung über der Karte zu sehen. Sprachausgabe kann sowohl über Textboxen als auch Sprachausgabe erfolgen.

Abbildung 26: Oberfläche für das mobile Szenario

In: Bühler 2002

3.5 Weitere life-like characters

In der folgenden Tabelle 8 sind zusätzlich die Agenten REA [Cassell 1999], Adele [Ganeshan 2000], Steve [Rickel 1999], Cosmo [Lester 1999], PPP-Persona [André 1999] und Greta [Pelachaud 2004] aufgelistet. Dabei wurden zuerst der Typ und der Einsatzbereich des Agenten betrachtet. Bis auf Greta sind alle Agenten aus dem Pädagogischen oder den nahe liegenden Bereichen der Relationalen- und Präsentations-agenten. Ziel dieser Untersuchung war es, Informationen über verwendete Werkzeuge, Technologien und Methoden zu erfahren. Auffallend ist, dass alle Systeme Sprachausgabe verwenden. Greta, REA und PPP-Persona verwenden das Festival-System als TTS-Engine. Keiner der Agenten läuft auf einem mobilen System. Adele und die PPP Persona sind als Java-Applets implementiert und werden über das Internet eingesetzt. Die anderen Systeme REA, Greta und Steve haben einen hohen technischen Aufwand (Spracherkennung, Virtuelle Realität) der auf einem mobilen System noch nicht zu realisieren ist. In den Projekten wurden mehrere frei erhältliche Werkzeuge verwendet. Für das dynamische Erstellen der Dialoge verwendet REA den Dialog-Planer SPUD. SOAR ist ein System für die Verhaltensplanung, das bei Steve eingesetzt wird. Greta basiert auf der Open Agent Architecture und verwendet das TrindiKit als Basis für das Dialog-System. Die Webadressen dieser Werkzeuge finden sich im Kapitel Links.

Tabelle 8: Übersicht life-like characters

	REA	Adele	Steve	Cosmo	PPP-Persona	Greta
Typ und Domain	Conversational/ Relational Agent, Immobilien-maklerin	Pädagogischer A-gent, medizinisches Umfeld	Pädagogischer Agent, Schiffsma-schinen	Pädagogischer Agent, Computer Netzwerke	Präsentations-Agent, Beliebig einsetzbar	Conversatio-nal-Agent, Beliebig einsetzbar
Interak-tion mit Benut-zer	Bidirektionaler multimodaler Dialog mit Gesten und Sprache	Benutzer verwendet Tastatur, Agent hat Sprachausgabe und Text	Spracherkennung, Sprachausgabe. Benutzer kann mit der 3D-Welt, in der Steve „lebt“, interagieren	Text-basierte Eingabe und Ausgabe. Benut-zer kann Objekte in der Welt von Cosmo verwenden.	Nur Präsentati-on	Spracherken-nung. Sprach-erzeugung
Ver-wendete Planer	Diskurs-Planer, hat „hardwired“ und „deliberative“ Reaktionen. Verwendet SPUD für die Satzerzeu-gung und BEAT für die Zuordnung der Animationen	Vorhersehbares Verhalten, sehr begrenzter Verhal-tensraum.	Kann dynamisch Pläne anhand von Zielen erstellen. Trennung in Wahrnehmung, Erkenntnis und Steuerung.	Behavior Sequencing Engine, ähnlich zu Herman-the-bug. Hat einen Deictic-Planer für die Synchronisati-on von Sprache und deictic-Gesten, Emo-tive- Planer für emoti-onale-Gesten	Rekursive Präsentations-planung nach vorgegeben Präsentations-zielen und zusätzlichen Parametern (Nutzerinteres-se).	Dialog Planer (Dipper) basierend auf dem TrindiKit.
Mögli-ches Verhal-ten	Hand-Gesten, Mimik, Blick Körperhaltung	Wenige Gesten und einige emotionale Gesichtsausdrücke	Gesten, Körper-haltung, Blick, In-teraktion mit Objekten in seiner Welt	Vordefinierte Verhal-tensweisen	Zeigegeste, Bewegung an beliebige Bildschirm-Position mög-lich, Mimik	Realistische Mimik und Mundbewe-gungen beim sprechen.
Darstel-lung	3D, Full-Body	2D-Comic, Oberkör-per	3D, Oberkörper	3D, Oberkörper	2D, Full-Body	3D, nur Kopf
Anima-tion	Key-frame-Animation, in-verse Kinematik. Verteilte Animati-onserstellung.	Geringe Anzahl von vordefinierten Ani-mationen	Bewegung im 3D Raum, 4 Handpo-sition (Ruhen, Zeigen, Drücken, Greifen), kann Objekte verwen-den.	Vordefinierte Ganz-körper- und Teilanim-ationen.	Animationsblö-cke (Springen, linker Schritt, etc.) vordefi-niert und kom-binierbar	Komplexe emotionalle Mimiken, Viseme, Blick, Lippen-synchrones Sprechen
Ver-wendete Techno-logien	SPUD: Satzplaner, BEAT, H-Anim VRML Modell, Festival TTS	Java-Applet	SOAR: Problem-lösungs und Schlußfolgerungs-system, VRML Modell	Microsoft Game Developers Kit	Java-Applet, Festival TTS	MPEG-4, Festival TTS, APML, Trin-diKit, Open Agent Archi-tecture
Kom-mentar	Ziel ist es, einen natürlichen Kom-munikations-partner zu ent-wickeln.	Unterstützung von Medizinstudenten. Darstellung und individuelle Erstell-ung von Kurs-materialien.	Benutzer wird in 3D-Welt über 3D-Brille und 3D-Mause integriert.	Soll den Benutzer durch Hilfe-Stellungen unterstützen. Hoher Aufwand, um Anima-tionen zu erstellen.	Automatisches Erstellen von Präsentationen	Magister-Prototyp. Fokus auf Mimik und Dialog

3.6 Agent-Frameworks

Es existieren eine Reihe von Frameworks, in denen Programme zur Erstellung von life-like characters zusammengestellt wurden. Eine einfache Übersicht ist in Tabelle 9 zusammengestellt. Die Gliederung beschreibt zuerst, ob das System eine eigene grafische Darstellung hat („Puppet“) und dann ob eine Planungskomponente („Planer“) enthalten ist. Die Zeile „Beschreibungssprache“ enthält Informationen darüber ob eine und welche Sprache verwendet wurde. Ein weiteres Kriterium ist ob das Projekt „Open-Source“ ist oder ob der Code zumindest frei verfügbar ist. Wichtig ist ob es zu dem Framework eine Community („Gemeinschaft“) gibt in der Informationen ausgetauscht werden. Eigenschaften der Frameworks, die nicht in diese Einteilung passen, sind unter „Bemerkung“ zusammengefasst.

Tabelle 9: Frameworks (Für WebAdressen siehe Kapitel „Links“)

	CU-Animate (CSLU Toolkit)	BEAT	XFace	Galatea	MS-Agent
Puppet	Animated faces, TTS, Speech-Recognition,	Nein	Talking Head mit TTS synchronisation	Talking Head basierend auf Photos, TTS, Speech Recognition	Verschiedene 2D Puppen erhältlich (z. B. „Büroklammer“), TTS
Planer	Dialog Management System	Verhaltensplanung abhängig von Texten	Nein	Dialog Manager	Nein, kann aber relativ einfach über Skripte festgelegt werden
Beschreibungs-Sprache	Nein	Kann verschiedene Charakterbeschreibungssprachen erzeugen	MPEG-4	Mehrere XML-basierte eigene Sprachen	MPML
Open-Source	Nein	Ja	Ja	Ja	Nein
Community	Ausführliche Dokumentation, Anwenderforum	Gute Dokumentation	Webseite, Entwickler Weblogs	Webseite, Wiki	Viele Webseiten, mit Skriptbeispielen.
Bemerkung	Festival für TTS, Fokus auf Sprachttraining	Nur für englische Sprache	Fokus auf Animation des Kopfes, Editor und Player getrennt erhältlich	Fokus auf Dialog-Management, nur japanisch	Schnell herstellbare Puppen. Einfache Integration in eine Anwendung über API.

Die in diesem Kapitel vorgestellten Systeme sind weitgehend aus universitären Forschungsprojekten. Daher war das Ziel häufig die Erforschung von spezifischen Eigenschaf-

ten, wie die Reaktion der Anwender auf den Agenten. Meistens wurde die gesamte Architektur auf den Agenten ausgerichtet. Dadurch ist zum Teil nicht mehr unterscheidbar, ob der Agent ein Teil des Systems oder das gesamte System ist.

Die vier ausführlicher vorgestellten Agenten haben alle für das MobiNaf Projekt interessante Eigenschaften.

Bei Laura wurde die Dialog-Komponente aufwendig gestaltet. Der Dialog-Server wird von Bickmore in seiner Doktorarbeit detailliert beschrieben. Außerdem ist die grafische Ausgabe der Figur sehr gut gelungen. Hier ist zu beachten, dass die Figur nur eine sehr geringe Bewegungsfreiheit hat. Dafür wurden die verwendeten Animationen umso aufwendiger gestaltet. Die Verwendung von Poser (s. Kapitel 2.5.1) für die Generierung könnte interessant sein, wenn es eine Möglichkeit gibt, die erstellten Animationen in das SVG-Format zu transformieren.

Gina stimmt in zwei Anforderungen mit dem hier zu erstellenden System überein. Zum einen basiert die grafische Darstellung auf SVG und zum anderen läuft das System auf einem PocketPC. Die verwendete Architektur und der verwendete SVG Viewer wurden bisher nicht dokumentiert. Das Interessante ist, dass der life-like character zuerst in Flash erstellt und dann erst nach SVG transformiert wurde. Weiterhin wird neben der Animation auch Sprachausgabe eingesetzt. Trotz der Verwendung von Vektorgrafik und dem Einsatz auf einem PDA werden keine Performance-Probleme erwähnt.

Für Herman-the-bug ist relativ ausführlich beschrieben, wie die dynamische Generierung der Lerninhalte für die Schüler funktioniert. Das Problem ist eine hohe Bindung der Animationen an die Lerninhalte. Änderungen sind relativ aufwendig, da als Konsequenz auch die Animationen neu erstellt werden müssen.

Im Gegensatz zu den anderen Systemen, hat SmartKom weniger eine pädagogische Zielsetzung. Hier ist insbesonders die Verwendung des Kontexts zur Anpassung der multimodalen Kommunikation interessant

Von den existierenden Frameworks können keine Komponenten wiederverwendet werden. Der Einsatzbereich der Frameworks liegt hauptsächlich in der schnellen Prototyp Entwicklung von PC-basierten Talking Heads („sprechenden Köpfen“). Von keinem der Frameworks wird eine PocketPC-Lösung angeboten.

Dadurch, dass kein frei verfügbares System existiert, das sowohl SVG-basiert ist als auch auf einem PocketPC-System läuft, muss ein neues System für das MobiNaf-Projekt erstellt werden. Die Architektur des Systems wird im folgenden Kapitel vorgestellt.

Kapitel 4: Systemkonzept

In diesem Kapitel soll das Konzept für einen pädagogischen Agenten vorgestellt werden. Im folgenden Abschnitt werden zuerst verschiedene Anwendungsfälle betrachtet. Das daraus entwickelte System ist ein Agent, der Kontext Information verwendet, um vom Autor beschriebene Präsentationen, Spiele und Dialoge zu verarbeiten. Das Ziel ist, dem Benutzer durch eine persönlichere Bindung an das System eine stärkere Umweltbindung zu vermitteln. Die dabei entstandene Architektur basiert auf Komponenten, die auf Änderungen von Kontextinformationen reagieren, die von einer zentralen Kontext Einheit verwaltet werden.

4.1 Use-Cases

Innerhalb einer Anwendungsfall-Analyse ist im Sinne einer Anforderungsspezifikation als erster Schritt herausgefunden worden, welche Aufgaben der Agent aus der Benutzer- und Autorensicht erfüllen soll. Das übergeordnete Ziel war dabei, sowohl für den Autor wie auch für den Endnutzer eine möglichst komfortable und funktionale Schnittstelle zu herauszuarbeiten. Hierbei ist besonders die Frage interessant, welche Rollen der Agent in Analogie zu einem menschlichen Führer innerhalb des Systems einnehmen soll. Jede dieser Rollen führt eventuell zu einem anderen Anwendungsfall-Szenario. Durch die Assoziation des Agenten mit einem menschlichen Naturführer ist eine große Anzahl von Funktionen denkbar. Viele dieser Funktionen sind aber sehr aufwendig und setzen ein mächtiges System voraus. Deswegen ist es hier besonders wichtig, dass bei der Analyse der Use-Cases bereits Überlegungen über den nötigen Realisierungs-Aufwand einfließen und mögliche Anwendungsszenarien vor der Implementierung in Bezug auf effektiven Nutzen und vertretbaren Implementierungsaufwand priorisiert werden.

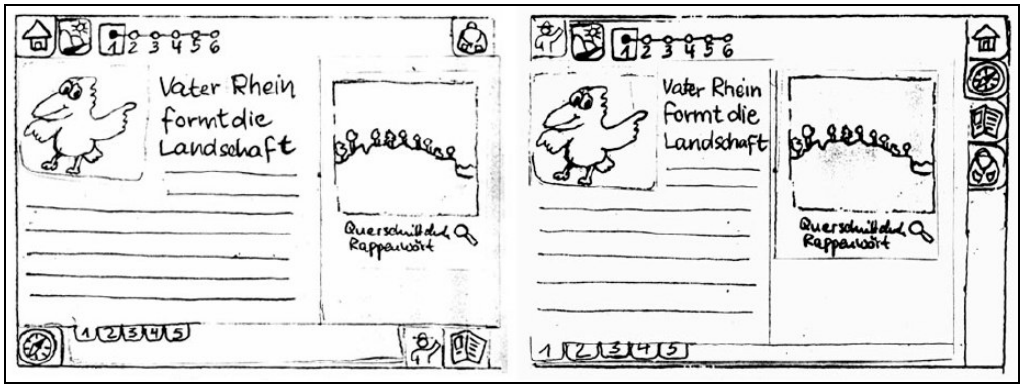

Abbildung 27: Rapid-Paper-Prototyping

In: Düpmeier 2004

Während der Analysephase wurden mehrere mögliche Anwendungsszenarien durchgespielt, die im Folgenden vorgestellt werden. Dabei hat sich besonders das Rapid-Paper-Prototyping[19]-Verfahren als ausgesprochen hilfreich erwiesen. Hierbei werden bereits konkrete grafische Benutzeroberflächen auf Papier gezeichnet, und in verschiedenen Anwendungs-Szenarios getestet. Die bereits in [Düpmeier 2004] erstellten Studien für die GUI eines mobilen Naturführers wurden dabei als Grundlage verwendet. In Abbildung 27 ist der Entwurf für das Layout mit dem Pädagogischen Agenten zu sehen. Dabei wurden zwei unterschiedliche Layouts mit Testpersonen ausprobiert.

Ein Aktor für Anwendungsfälle des Agenten ist der Besucher eines Naturschutzgebiets, der eine mobile Guide-Applikation als Informationsmedium verwendet. Der Autor ist ein weiterer Aktor, der das System als Medium benutzt, um Informationen über das Gebiet an den Besucher zu vermitteln. Der Autor muss hierzu über komfortable Möglichkeiten verfügen, die zu vermittelnden Informationen und ihren Bezug zum Agenten festzulegen. Folgende Anwendungsfälle aus der Sicht des Benutzers und des Autoren wurden dabei identifiziert.

[19] Rapid-Paper-Prototyping: Schnelle, meistens papierbasierte Erstellung der graphischen Benutzeroberfläche eines Systems.

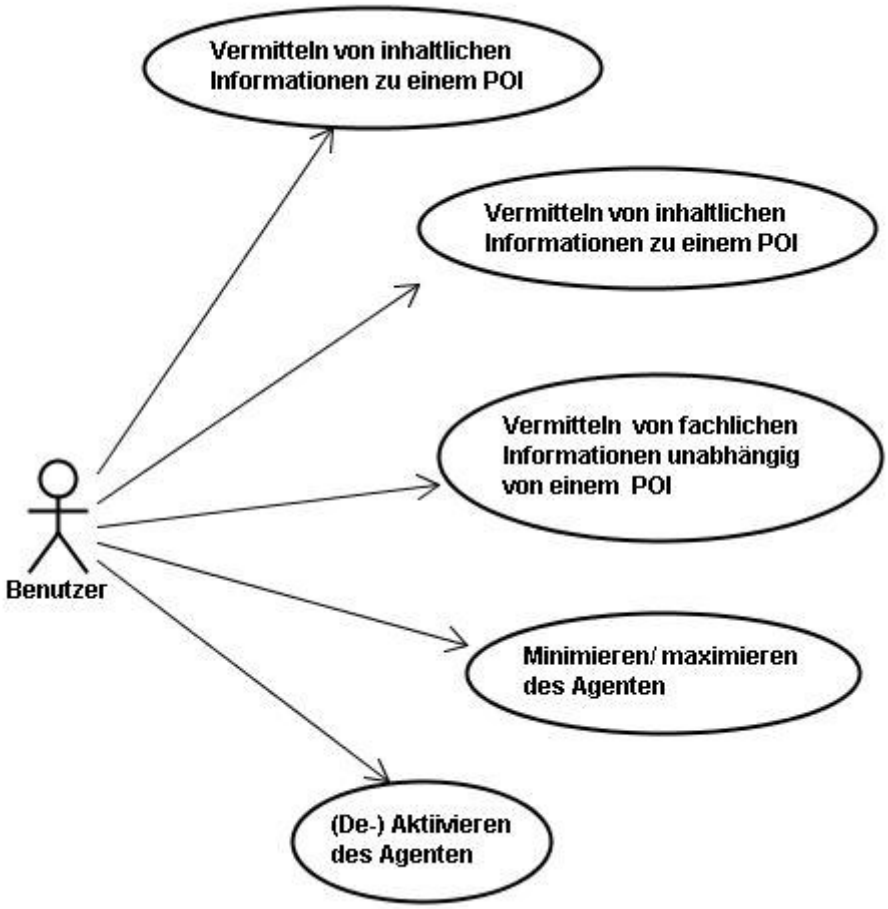

Abbildung 28: Benutzer Use-cases

Benutzer-Use-Cases:

Abbildung 28 zeigt typische Anwendungsfälle für einen Benutzer des Mobilen Naturführers. Hier sollen die einzelnen Anwendungsfälle im Folgenden kurz vorgestellt werden.

UC 1: Vermitteln von zusätzlichen fachlichen Informationen zu einem POI

Der Benutzer erreicht einen Point of Interests (POI). Dort wird ihm der Text zu diesem POI auf dem PDA angezeigt. Die darin enthaltenen Informationen sind nicht ausreichend oder zufriedenstellend. Deswegen möchte er Zusatzfragen an das System stellen. Denkbar sind Zusatzfragen zu einzelnen Begriffen oder zum gesamten Inhalt des Textes. Der Agent präsentiert dem Benutzer eine Anzahl von Zusatzfragen z. B. bei Klicken eines „Frage-Icons“, von denen der Benutzer eine auswählt. Diese Zusatzfragen und deren Antworten müssen vom Autor vorher geschrieben werden. Danach kann der Benutzer weitere Fragen stellen, bis er die Interaktion beendet hat. Dadurch werden die eigentlichen Texte auf die wesentlichen Informationen beschränkt. Weniger interessierte Leser müssen sich nicht durch für sie zu lange Texte durcharbeiten.

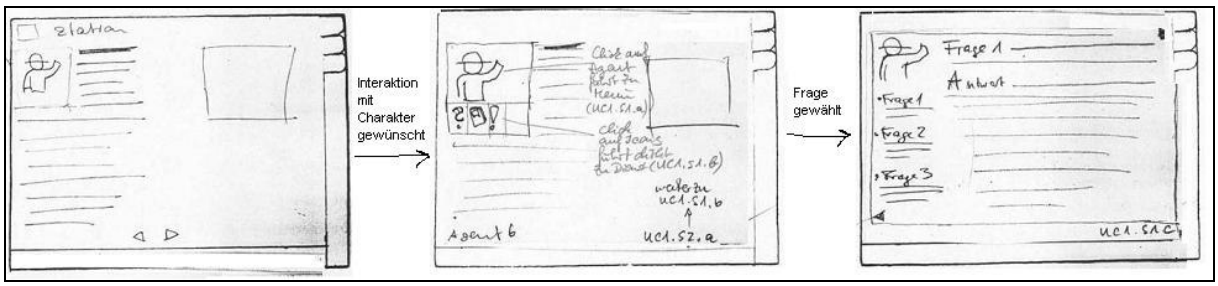

Abbildung 29: Benutzer hat Fragen an den Charakter

In Abbildung 29 ist diese Interaktion abgebildet. Auf dem linken Bild ist die Agentenfigur noch im Präsentationsmodus. Der Benutzer drückt auf die Figur und bekommt das „?"-Icon angezeigt (mittlerer Kasten). Nach Klicken auf das Fragezeichen werden Fragen über Themen abhängig von dem POI angezeigt (rechter Kasten). Wenn der Benutzer eine Frage auswählt, bekommt er die Antwort vom Agenten präsentiert.

Analyse:

Hierbei kann man bereits mehrere Komponenten des zu konzipierenden Agentensystems erkennen. Um Zusatzfragen zu beantworten, benötigt der Agent eine Art Dialogsystem. Zusätzlich ist eine vom Autor geschriebene Datenbasis mit Fragen und Antworten nötig. Bei den POI-spezifischen Fragen und Antworten muss es eine Beziehung zu dem zugehörigen POI geben.

UC 2: Vermitteln von fachlichen Informationen unabhängig von einem POI

Der Benutzer ist an irgendeinem Punkt im Naturschutzgebiet und hat eine fachliche Frage. (z. B. Welche Baumart ist das?). Er startet den Dialog mit dem Agenten. Der Agent versucht die Frage durch eine Frage-Antwort-Dialoghierarchie (d. h. Fragen zu immer spezifischeren Teilbereichen) zu beantworten. Danach kann der Benutzer weitere Zusatzfragen zu diesem Bereich stellen, bis er die Interaktion/den Dialog beendet.

Analyse:

Im Unterschied zu UC 1 ist hier eine allgemeine (von speziellen POIs unabhängige) Datenbasis nötig. Im einfachsten Fall ist hierbei eine Datenbasis mit konkreten Fragen und Antworten nötig. Ein komplexerer Lösungsansatz wäre der Versuch, die Fragen aus einer allgemeinen Domain-Knowledge-Faktendatenbank z. B. über Anwendung eines Regelsystems zu beantworten. Ein derartiges System wird von Ackerman beschrieben [Ackerman 1994].

UC 3: Minimieren/ Maximieren des Agenten

Der Agent wird in seiner maximalen Größe auf dem Bildschirm angezeigt. Das nimmt für den Benutzer zu viel Platz auf der Oberfläche weg. Deswegen möchte er den Agenten minimieren. Er klickt auf den Agenten und wählt aus dem Menu die Funktion zur Minimierung aus. Der Agent „schrumpft“ dadurch auf die Größe eines Icons. Die Animationen sind sehr eingeschränkt. Die volle Funktionalität ist vorhanden, eventuell maximiert sich der Agent automatisch bei bestimmten Aufgaben.

Der Agent ist minimiert und somit nur als kleines Objekt auf dem Bildschirm sichtbar. Der Benutzer hat eine Frage an den Agenten, die dieser nur im maximierten Zustand beantworten kann. Deswegen maximiert der Benutzer den Agenten. Dabei verschieben sich die dargestellten Bildschirmobjekte. Der Agent bietet im maximierten Zustand seine volle Funktionalität an.

Analyse:

Der Agent braucht eine interne Repräsentation für unterschiedliche Zustände (minimiert/maximiert). Abhängig davon wird auch eine unterschiedliche Funktionalität angeboten.

UC 4: Aktivieren, Deaktivieren des Agenten

Fall 1: Der Agent ist deaktiviert. Der Benutzer hat eine Frage an den Agenten, die dieser nur im maximierten Zustand beantworten kann. Deswegen maximiert der Benutzer den Agenten. Dabei verschieben sich die dargestellten Bildschirmobjekte. Der Agent bietet im maximierten Zustand seine volle Funktionalität an. Alternative: Der Agent legt sich als Overlay („Überlagerung“) über den Inhalt, dadurch würden Inhalte überdeckt.

Fall 2: Der Agent ist aktiv, entweder minimiert oder maximiert. Der Benutzer hat keine Verwendung für den Agenten oder fühlt sich sogar durch den Agent gestört. Deswegen deaktiviert der Benutzer den Agenten. Je nachdem, ob der Agent vorher im minimierten oder maximierten Zustand war, verschiebt sich danach die Bildschirmaufteilung.

Der Agent kann dadurch nur über expliziten Aufruf des Benutzers wieder aktiviert werden.

Analyse:

Entspricht UC 3

Abbildung 30 zeigt typische Anwendungsfälle aus der Sicht des Autors.

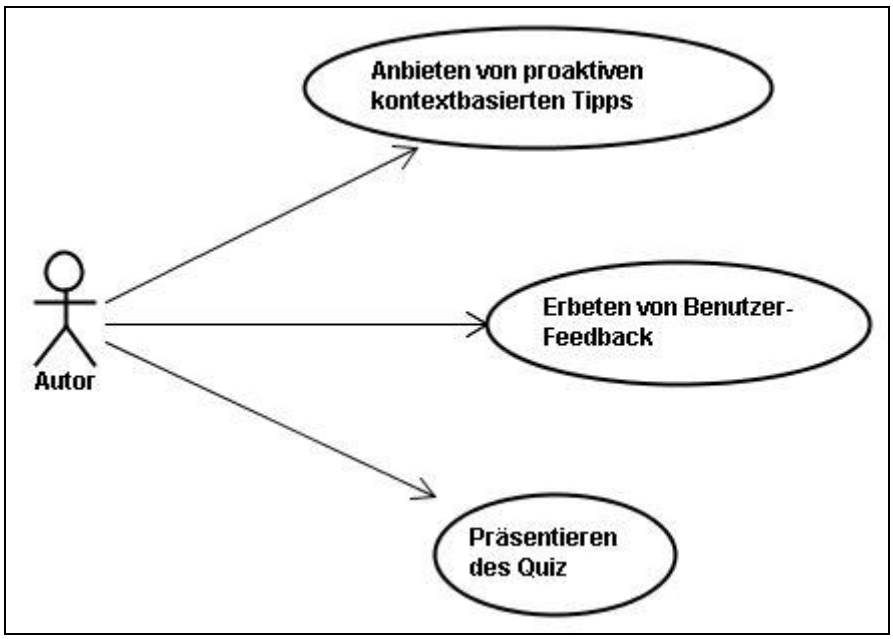

Abbildung 30: Autor-Use-cases

Die einzelnen Fälle sollen im Folgenden kurz diskutiert werden.

Autoren-Use-Cases:

UC 5: Erbeten von Benutzerfeedback

Der Benutzer hat eine Tour beendet und schließt die Applikation bzw. beendet die Tour, bevor er das Gerät zurückgibt. Der Autor der Tour möchte, dass der Benutzer ein Feedback über die Tour zurückgibt. Daher wird beim Beenden der Tour bzw. beim Schließen der Applikation der Agent eingeblendet, um eine Qualitäts-Rückmeldung vom Benutzer anzufordern. Dabei sollte der Benutzer auch die Möglichkeit haben, die Feedbackabfrage zu verweigern.

Analyse:

Hier ist eine Feedback-Komponente notwendig, die die Antworten des Benutzers speichert und später für den Autor verfügbar macht.

UC 6: Quiz präsentieren

Der Autor hat die Möglichkeit, Fragen und Antworten (Quiz) zu einem Themenbereich zu beschreiben. Dieses Quiz wird dem Benutzer am Ende einer POI-Interaktion gestellt, um das Benutzerverständnis bzw. den Lernfortschritt zu testen. (Denkbar wäre auch vor einem

POI oder am Anfang der Tour, um den Kenntnisstand des Benutzers zu erfahren). Der Benutzer wählt eine der Antworten aus, und das System ermittelt, ob die Antwort korrekt oder falsch ist. Die richtige Antwort wird dem Benutzer mitgeteilt. Danach geht es mit der nächsten Frage weiter, bis alle Fragen bearbeitet wurden oder der Benutzer die Interaktion beendet. Aus dem Gesamtergebnis wird der Benutzer in eine Kenntnisstand-Kategorie im Kontext eingeordnet.

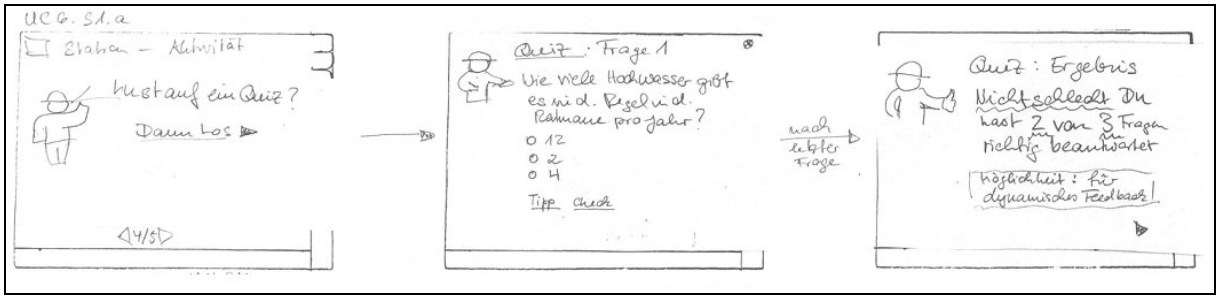

Abbildung 31: Quiz präsentieren

In Abbildung 31 sind mögliche Benutzeroberflächen für diesen Use-Case dargestellt. In dem linken Kasten bietet die Agentenfigur das Quiz an. Danach ist im mittleren Kasten eine Frage mit einer Liste von möglichen Antworten dargestellt. Diese Oberfläche kann sich in ähnlicher Form beliebig oft bis zum Ende des Quiz wiederholen. In dem rechten Kasten ist ein möglicher Ergebnisbildschirm gezeichnet worden.

Analyse:

Hier wird eine eigenständige Quiz-Komponente für die Steuerung von einem Quiz (oder allg. „Spiel“) vorgesehen. Hierbei kann man zwei Ziele für die Erstellung eines User Profiles unterscheiden. Einmal für die spätere Auswertung durch den Autor (z. B. Messung des Lernerfolgs), und zum Anderem, um Inhalte des Systems direkt an den Benutzer zu adaptieren [Brusilovsky 1996]. Dabei ist das Ziel, eine bessere Übereinstimmung der Benutzerinteressen mit den angebotenen Materialien zu erreichen. Ein Quiz ist nur eine Quelle für das Profil, z. B. lassen sich auch aus dem bereits abgelaufenen Weg in Verbindung mit der Aufenthaltsdauer an bestimmten POIs Rückschlüsse auf Interessen des Anwenders ziehen. Bei der Anpassung der Inhalte an den Anwender besteht nach [Cheverst 2001] das Risiko durch eine unpassende Adaption eher Frustration als einen besseren Anwendungskomfort zu erreichen. Eine Überlegung ist es, eventuell die Quiz-Funktionalität auch durch die Dialog Komponente darzustellen.

UC 7: Anbieten von proaktiven kontextbasierten Tipps

Der Benutzer ist an irgendeinem Punkt im Naturschutzgebiet. Das System meldet sich ohne Benutzeraktion, ausgelöst durch ein spezielles Ereignis in seinem Kontext. Ein Fall

wäre ein Positionshinweis in dem Fall, dass der Benutzer zu weit vom eigentlichen Weg einer Tour abgekommen ist. Das System informiert den Nutzer über die Situation und gibt ihm Anweisungen, wie er auf den Tourweg zurückfindet. Ein zweites gutes Beispiel im Zeit-Kontext wäre das folgende: der Benutzer befindet sich bereits 30 Minuten an einem POI, möchte aber die Tour in einer Stunde schaffen. Das System rät ihm weiterzugehen, da die Tour sonst nicht in der Zeit möglich ist. Eine Alternative wäre vorzuschlagen, die Tour abzukürzen.

Analyse:

Dieser Use-Case setzt die Existenz einer Kontext-Komponente voraus. Diese muss dem Agenten verschiedene Kontexte bereitstellen. Ebenso müssen vom Autor verschiedene Reaktionen (Aktionen) auf Kontext-Ereignisse definiert werden. Kontext-Ereignisse und Aktionen müssen dabei durch einen internen Mechanismus aufeinander abgebildet werden können. Weiter sollte es die Möglichkeit geben, neue Aktionen zu erstellen und diese möglichst leicht in das System zu integrieren.

Zum Schluss dieser Analyse von Anwendungsfällen sollen noch einige Randbedingungen an die Agentenimplementierung festgehalten werden, die sich zum Teil auch erst später aus dem Projekt entwickelt haben:

Unterschiedliche Ansichten des Agenten:

Die grafische Repräsentation des Agenten soll in drei unterschiedlichen Zoomstufen möglich sein. Der komplette Körper, der Oberkörper und nur der Kopf. Dadurch kann eine bessere Aufteilung der grafischen Benutzeroberfläche erreicht werden. Ebenfalls hat dies Einfluss auf die affektive Wahrnehmung des Agenten durch den Benutzer.

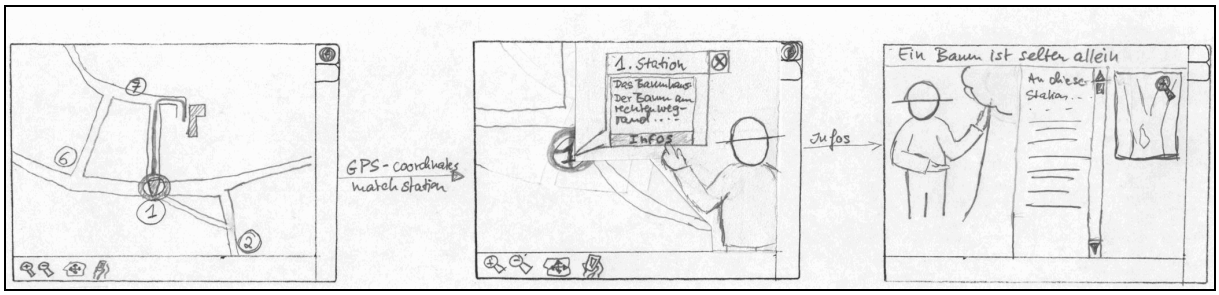

Abbildung 32: Präsentation von POI-Informationen

Präsentation der POI-Informationen:

Der Agent soll die durch den Autor erstellten Inhalte für die einzelnen POIs präsentieren. Sinnvoll wäre dabei eine Sprachausgabe, da gerade im Kontext einer Naturführung die Besucher während der Wahrnehmung der Texte auch die Natur betrachten sollen. Dies hätte mehrere Vorteile. Die geringe Displaygröße bedeutet, dass der Benutzer bei längeren Texten entweder häufig scrollen oder die Seite umblättern muss. Bei Sprachausgabe müsste der Benutzer nicht dauernd blättern. Weiter erscheint einem Benutzer ein Agent mit Sprachausgabe glaubwürdiger, da die Animationen mit der Sprache synchronisiert sind. Besonders bei Benutzergruppen, in der sich z. B. auch Personen befinden, die schlecht sehen können oder falls mehr als eine Person in der Gruppe sind, könnte die Sprachausgabe wichtig sein, damit die Person oder alle Personen überhaupt die Informationen mitbekommen. Die Präsentation von Informationen ist sowohl in einem Popup-Fenster mit Sicht auf die Karte oder in einem Vollbild-Fenster möglich. In Abbildung 32 ist eine Papierversion der Benutzeroberfläche dargestellt. Im linken Bild ist der Benutzer kurz vor dem Erreichen von Station 1. Sobald die Benutzerposition mit der Stationsposition übereinstimmt, präsentiert der Agent die Beschreibung der Station (mittlerer Kasten) und zeigt auf den Knopf für weitere Informationen. Wenn der Benutzer die weiteren Informationen auswählt, wechselt das System in den Vollbild-Präsentationsmodus mit dem Agenten, Bildern und dem Text (rechter Kasten) ohne Anzeige der Karte. Der Text wird dabei in jedem Szenario auch als Sprachausgabe ausgegeben.

Erweitern der Animationen:

Der Autor muss in der Lage sein, die existierenden Animationen der Figur um eigene Animationen zu erweitern, bzw. bestehende Animationen auszutauschen.

Die wichtigsten Erkenntnisse aus der bisherigen Use-Case-Analyse sind noch einmal in Tabelle 10 zusammengefasst.

Tabelle 10: Die ersten identifizierten Komponenten:

Komponente	Funktion
Dialogsystem	Führen von Dialogen mit dem Anwender. Am Ende des Dialogs kommt eine Aktion.
User-Profile-Komponente	Anpassen des Systems an den Benutzer. Informationen für den Autor.
Allg. und kontextabh. F&A Datenbasis	Beschreibung von möglichen Benutzerfragen und Antworten zu allgemeinen und kontextabhängigen Themen.
Dialog-Datenbasis	Beschreibung der Dialoge und resultierenden Aktionen durch den Autor.
Autor-Feedback-Komponente	Rückgabe von bestimmten Informationen an den Autor.
Kontext Komponente	Ermitteln von Kontexten. Benachrichtigung an andere Komponenten über Kontextveränderungen.
Quiz-Komponente	Darstellen von einem Quiz. Weitergeben des Resultats an die User-Profile Komponente.
Verschiedene Zustände des Agenten.	Komplett, Oberkörper, nur Kopf, minimiert/ maximiert, aktiviert/ deaktiviert.
Sprachausgabe	TTS für deutsche Sprache.
Erweiterbarkeit der Agentenanimationen.	Erweiterbare nicht verbale Ausdrucksformen für den Agenten.
Austauschbare grafische Repräsentationen.	Grafische Darstellung des Agenten soll auswechselbar sein.
Präsentations-Komponente	Präsentation der Inhalte durch den Agenten.

Nach der Anwendungsfall-Analyse wurden die wesentlichen Komponenten des Systems identifiziert und eine geeignete System-Architektur für das Agentensubsystem der Mobile Naturführer Applikation entworfen. Diese wird im folgenden Unterkapitel beschrieben.

4.2 System-Architektur

In diesem Abschnitt soll die Gesamtarchitektur eingeführt werden. Als erstes wird eine grobe Übersicht des Systems diskutiert. Um konkreter auf die Abläufe zwischen den Komponenten einzugehen, sollen mehrere Beispielszenarios aus den Use-Cases diskutiert werden.

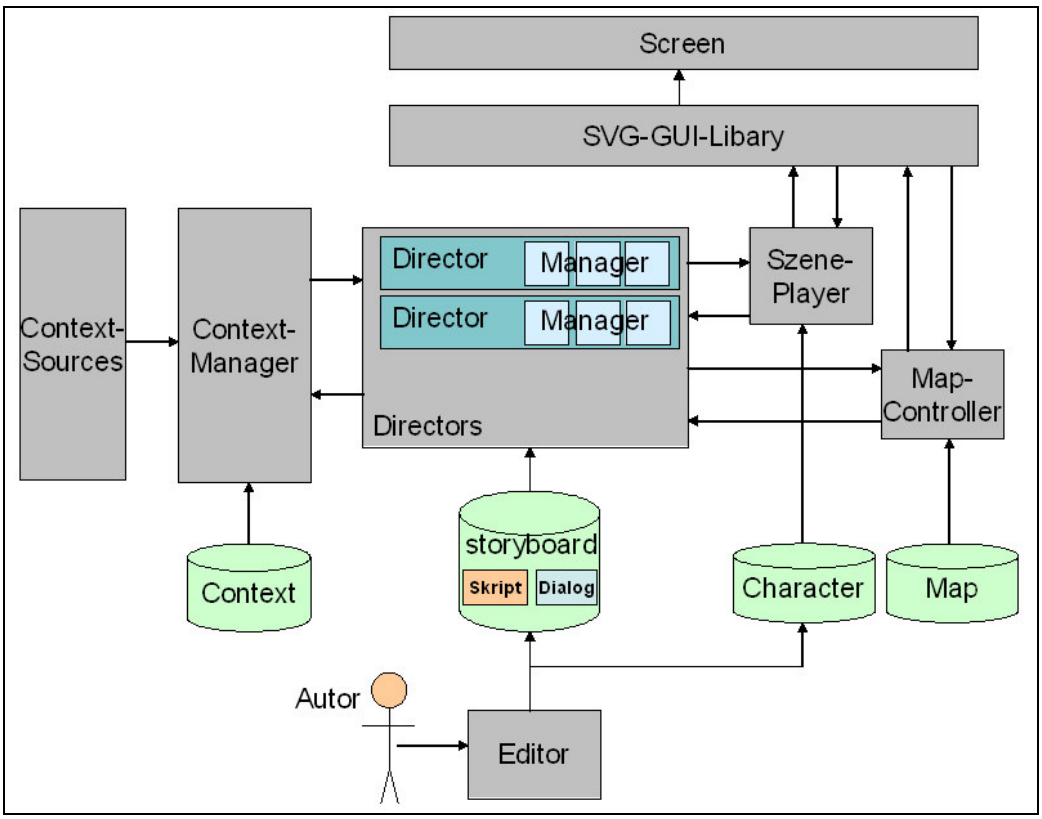

Abbildung 33: Gesamt-Architektur Übersicht

Die grundlegende Architektur ist in Abbildung 33 verdeutlicht. Für die Darstellung von Inhalten über SVG bzw. für das Empfangen von Interaktionen über das SVG-Event-Modell verwendet die Applikation eine **SVG-GUI-Libary**, die die Schnittstelle von SVG zur Applikation hin kapselt.

Das **Storyboard** enthält als Datenhaltungsschnittstelle alle vom Autor geschriebenen Informationen zu Szenarien, die durch den Agenten dargestellt werden sollen. Das Storyboard kann daher mit einem Drehbuch verglichen werden, das alle vom System darzustellenden Texte, Dialoge und Spielhandlungen in Form einzelner Beschreibungen enthält. Die Beschreibungen der Inhalte, die abhängig von einem Kontext- oder Benutzerereignis angezeigt werden sollen, ist in Form von Skripten („Drehbüchern“) im Storyboard eingetragen. Ein Skript beschreibt Aufbau, Steuerung und Ablauf der Darstellung von einzelnen Akten. Ein Akt kann ein bestimmter Darstellungsblock wie ein Spiel, eine Präsentation oder eine Warnungsmeldung sein. Wichtig ist dabei nur, dass die Akte in einem Zusammenhang stehen, wie etwa das sie alle zu einem bestimmten Ereignis wie das erreichen einer Station gehören. Das System kann mehrere Directors („Regisseure“) haben, die für die Koordination der Darstellung von Skripten verantwortlich sind. Zur Verarbeitung von Akten hat jeder einzelne Director spezialisierte Manager, die sich um die Darstellung der Akte kümmern. Je nach Typ eines Aktes (Dialog, Präsentation, Quiz) kümmert sich ein anderer, auf diesen Typ spezialisierte Manager um die Steuerung und Koordinierung des Ablaufs der einzelnen Szenen innerhalb

eines Aktes. Ein Akt wird von den Managern in Szenen transformiert, die von einem zugehörigen Szene-Player dargestellt werden. Der Unterschied ist, dass ein Akt noch etwas Allgemeines ist, wogegen eine Szene konkret auf den aktuellen Anwendungskontext angepasst wurde. Diese Struktur ist in Abbildung 34 dargestellt.

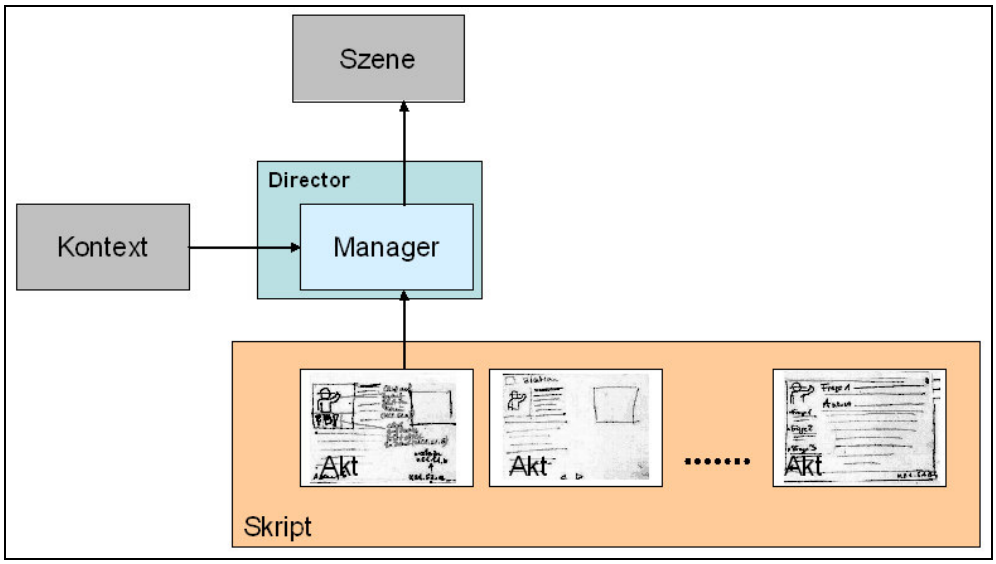

Abbildung 34: Skript-Akt-Szene Struktur

Das Storyboard, das die Informationen als Grundlage für die ablaufenden Skripte liefert, wird mit einer **Editor**-Komponente erstellt. Der Autor definiert dabei über den Editor die Informationstexte und Bilder, die als multimediale Informationsbeschreibung vom Szene-Player auf dem Bildschirm dargestellt werden. Die Texte können zusätzlich durch den Einsatz von multimodalen Kommunikationsformen unterstützt werden. Die akustische Ausgabe erfolgt durch eine Text to Speech (TTS)-Komponente. Gestik und Mimik werden von einer animierten Agentenfigur synchron zur Textaussprache dargestellt. Das vom Autor definierte Verhalten des animierten Charakters wird von der Editor-Komponente bereits vor der Laufzeit in eine SVG-Animation übersetzt. Dafür ist ein Animations-Compiler zuständig, der Teil des Editors ist. Das hat den Vorteil, dass während des Einsatzes des Systems Ressourcen gespart werden. Die SVG-Animation wird in die character-Datei des Szene-Players eingetragen. Analog wird im Editor die Sprachausgabe in einem Audio-Format erzeugt und gespeichert. Die SVG-Animationen zusammen mit dem Text, den Bildern und der Audio-Ausgabe bildet eine Szene. Es wird immer nur eine Szene von dem Szene-Player bearbeitet. Die verwendete Szenen-Beschreibungssprache wird in Abschnitt 4.6 erläutert. Für die gesamte Szene wird sowohl die grafische als auch die akustische Ausgabe vom Szene-Player synchronisiert.

Für jeden Manager der Director-Komponente müssen die passenden vom Autor vorbereiteten Informationen im Storyboard in zugehörigen Sektionen bereitstehen. So wird z. B. der Ablauf eines Dialogs in einer Dialog-Sektion des Storyboards beschrieben.

Über den **Context Manager** haben die einzelnen Directors Zugang zu dem augenblicklichen Kontext, in dem sich die Applikation befindet. Der Director meldet sich beim Context Manager für bestimmte Kontextänderungen an, über die sie benachrichtigt werden wollen. Basierend auf dem aktuellen Zustand der Darstellung, den Kontextinformationen und Ereignissen, die hereinkommen, kann und muss ein Director entscheiden, welcher Akt oder welcher Dialog als nächstes von dem Szene-Player abzuspielen ist.

Ereignisse, die von der grafischen Oberfläche (GUI) kommen, wie z. B. eine Interaktion des Benutzers mit dem Agenten werden zunächst an die Controller des jeweiligen Oberflächenbausteines (Widget) geschickt. So wird zum Beispiel bei einer Aktion des Benutzers mit dem Präsentations-Widget der ausgelöste Event an den Szene-Player weitergeleitet. Dieser bearbeitet das Ereignis selbstständig oder leitet es an den zugehörigen Director weiter.

Der grobe Ablauf soll einmal an mehreren Beispielszenarios erläutert werden. Diese Beispiele basieren alle auf einem Director (Stations-Director) speziell zur Anzeige von Stationsinhalten im Rahmen einer Tour durch ein Naturerlebnisgebiet.

Beispielszenario „Erreichen einer Station“

Als Vorraussetzung wurde vom Autor für das Ereignis „Erreichen der Station 4“ ein Skript erstellt. Die dazugehörigen Animationen, Bilder und Akustik-Ausgaben stehen im storyboard und in der character Datei.

Ein Ereignis wie das Erreichen einer neuen Station wird zunächst dem Context Manager als Kontextinformation („Station 4 erreicht“) gemeldet. Da sich der Stations-Director für solche Ereignisse im Context Manager registriert hat, wird diese Kontextinformation an den Stations-Director weitergeleitet.

Als Reaktion auf die Kontextänderung lädt der Director das Stationsskript für Station 4 aus dem Storyboard. Das Skript spezifiziert, dass an dieser Stelle ein Popup-Fenster erscheint, das auf die Station hinweist. Innerhalb des Popups soll der Agent integriert sein, der neben einer akustischen Ausgabe eines Satzes der Art „Sie haben Station 4 erreicht: Möchten Sie weitere Informationen zu dieser Station?“ eine entsprechende Geste auf dem Bildschirm produziert, die auf einen Button „Mehr Informationen“ weist. Im folgenden Codebeispiel ist die Beschreibung des Autors für das Erreichen von Station 4 beschrieben.

```
<MOUTH CMD="SMILE"/>Sie haben <GESTURE HAND="L" CMD="POINT_LEFT">
Station 4 </GESTURE> erreicht.
```

Der Director übergibt seinem Presentations-Manager den ersten Akt. Der Manager erzeugt daraus die entsprechende Szene für den Szene-Player, also eine logische Beschreibung

dessen, was gerade unter dem aktuellen Kontext angezeigt werden soll. Der Szene-Player lädt daraufhin die passende vorkompilierte Animation und das Audio-File aus der character-Datei. Der Text selbst wird in einem Popup-Fenster angezeigt. Zusätzlich wird die animierte Figur zusammen mit der synchronisierten Audioausgabe dargestellt (s. Abbildung 32). Eventuell möchte der Benutzer die Animation und Audioausgabe wiederholen. Dafür gibt es eine „replay"-Funktion, die der Szene-Player selbstständig durchführen kann. Als nächsten Schritt kann der Anwender sich zwischen zwei Aktionen entscheiden. Entweder er verlangt „mehr Informationen" zu der Station oder er schließt das Fenster. Beide Aktionen werden von dem Szene-Player zurück an den Director geleitet. Beim Schließen des Fensters benachrichtigt der Director den Kontext-Manager und dieser setzt daraufhin innerhalb des Tour-Kontexts die Station 4 auf „beendet". Gleichzeitig wird in einem User Profile, das Teil des Kontext-Managers ist, protokolliert, dass der Benutzer Station 4 beendet hat ohne die eigentliche Beschreibung gelesen zu haben.

Im nächsten Beispielszenario wird dieser Fall fortgesetzt, indem der Anwender „mehr Informationen" zu der Station ausgewählt hat.

Beispielszenario „Vermitteln von Informationen zu einer Station"

Der Anwender möchte mehr Informationen zu Station 4 haben. Dies wird dem Director durch den Szene-Player mitgeteilt. Daraufhin übergibt der Director den nächsten Akt an den Präsentations-Manager. Dieser kann eventuell an der Szene noch kontext-abhängige Änderungen vornehmen. Danach wird die erstellte Szene zur Darstellung an den Szene-Player übergeben. Der Text wird durch Sprache, Text und Animationen an den Besucher vermittelt. Am Ende der Präsentation kann der Benutzer den Text wiederholen („replay-Funktion") oder zur folgenden Szene weitergehen. Außerdem hat er die Möglichkeit, einen Dialog mit dem Charakter zu starten, um z. B. mehr Information zu einem Thema zu erfahren. Das soll im nächsten Szenario besprochen werden.

Beispielszenario „Benutzer möchte Fragen beantwortet haben"

Der Benutzer hat eine weitergehende Frage zu einem Thema aus der Präsentation (s. Kapitel 4.2 UC 6). Neben der animierten Figur gibt es ein „?"-Icon. Das Drücken dieses Icons empfängt der Szene-Player, der dieses Ereignis an den Director weiterleitet. Der Director aktiviert daraufhin den Dialog-Manager. Falls der Autor vorher in der Beschreibung für den Akt zusätzliche Informationen in Form von Fragen und Antworten bereitgestellt hat, werden die möglichen Fragen vom Dialog-Manager in einer „Frage-Auswahl"-Szene zusammengefasst und an den Szene-Player zur Präsentation übergeben. Der Benutzer hat die Auswahl zwischen den angebotenen Fragen oder dem Ende des Dialogs. Wenn er eine Frage auswählt, übergibt der Dialog-Manager dem Szene-Player die aktuelle Szene mit der Antwort zur Frage. Dies entspricht nur einer einstufigen

Beantwortung von Fragen im aktuellen Kontext. Ein Dialog zur Beantwortung von allgemeineren Fragen ist komplexer und ist nicht Teil dieses Szenarios.

Im Folgenden soll auf die Funktionsweise von einzelnen Komponenten des Gesamtsystems genauer eingegangen werden.

4.3 Director

Es kann mehrere Director für unterschiedliche Anwendungsfälle geben. Ein einzelner Director koordiniert die Darstellung der vom Autor bereitgestellten Informationselemente und Aktionen. Er reagiert auf Kontextänderungen und Ereignisse vom Nutzer und System und übergibt dem Szene-Player über seine Manager-Komponenten Anweisungen was für den Benutzer abgespielt werden soll. Er soll u. a. die folgende Funktionalität besitzen:

- Informationselemente passend zum Kontext auswählen.
- Darstellung der Texte koordinieren (Zuordnung der Texte) mit anderen Ausgaben, wie Sprache und Geste.
- Kontextabhängige Veränderung der abgespielten Szenen vornehmen.
- Spiele in Szenen darstellen und durch Puppet-Komponente abspielen lassen (im Fall eines Game Managers).
- Dialog mit Puppet kontrollieren (im Fall eines Dialog-Managers).

Eine zentrale Eigenschaft eines Director ist, dass er in Abhängigkeit von Änderungen im Kontext (sei es durch GUI-Ereignisse, Systemereignisse, Zustandsänderungen) oder anderen Kontextänderungen agiert. Dadurch, dass der Director eine eigenständige Komponente ist, die auf Ereignisse reagiert und abhängig vom Kontext eine Ausgabe erzeugt, erfüllt sie die Definition für einen Agenten (s. Kapitel 2.1 „Agenten“). Die Architektur basiert auf der „geschichteten Architektur“ („layered architecture“). Der Director erzeugt abhängig vom Kontext Anweisungen zum Abspielen bestimmter Szenen für die Puppet Komponente. So ist die Darstellung der Stationsbeschreibung als Szenerie genauso möglich wie die Generierung einer Szene zur Warnung, dass der Benutzer in eine naturgeschützte Zone eintritt oder das Naturschutzzentrum bald schließt. Das System kann dabei mehrere seperate Director haben, die auf verschiedene Kontexte reagieren.

Das Director-Konzept wird aus Gründen der einfachen Erweiterbarkeit über eine Plugin Architektur umgesetzt. Der Directors Plugin Container stellt Basisdienste wie den Zugang zu den Kontext Manager und Storyboard für die Komponenten bereit.

Jeder Director hat einen Szene-Player mit einer Puppet für eine anthropomorphe (s. Kapitel 2 „multimodale Kommunikation“) Kommunikation mit dem Benutzer. Der Autor hat bereits im Editor die von der Puppet darzustellenden Gesten und Mimiken (Anweisungen

an die Puppet) innerhalb des anzuzeigenden Textes eingetragen und mit dem Text synchronisiert.

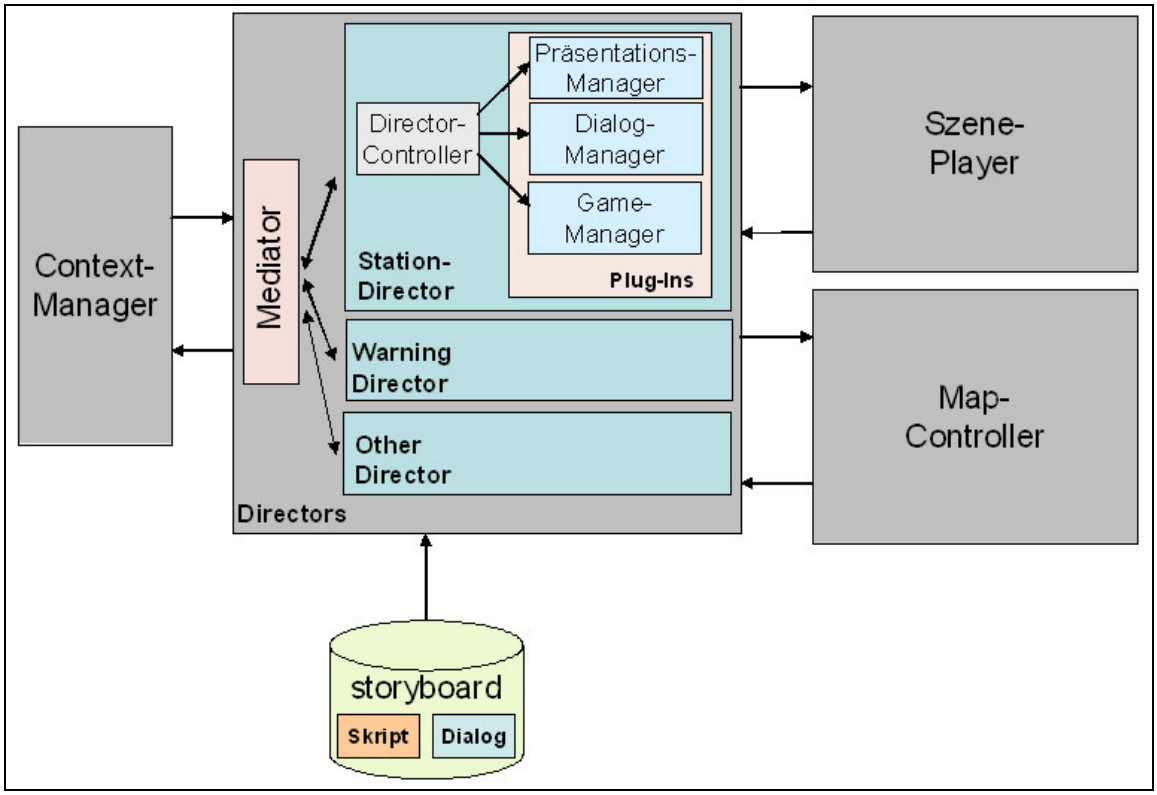

Abbildung 35: Basisarchitektur-Übersicht

In Abbildung 35 ist eine Übersicht der Architektur des Directors und die Verbindung zu anderen Komponenten gezeichnet.

Da der Director sowohl den Kontext als auch den Inhalt kennt, ist er der natürliche Platz für eine Planungskomponente. Im Zusammenhang mit der reinen Präsentation von Inhalten (Präsentation Manager Plugin), kann man daher auch von einem Präsentationsplaner sprechen. Hierbei gibt es im Wesentlichen zwei Fragen: welchen Freiheitsgrad bekommt der Planer und wieweit sind die Texte bereits vorbearbeitet. Die Antwort darauf hängt vom Anwendungskontext ab. Präsentationsplanung ist aufwendig und verlangsamt dadurch die Reaktion des Systems. Somit sollten alle Texte, die nicht von einem aktuellen Kontext abhängen, bereits vorher geplant sein. Man muss auch zwischen der inhaltlichen und der Darstellungs-Planung unterscheiden. Eine inhaltliche Planung würde die Präsentation aus einer Wissensbasis nach einem definierten Ziel zusammenbauen (s. Kapitel 2.7.1). Der Autor müsste also Präsentationsbausteine erstellen und die Ziele dieser Bausteine im Rahmen einer Präsentation beschreiben und eine Wissensbasis bereitstellen. Das Ziel könnte wiederum von aktuellen Kontexten abhängen, wie z. B. dem Benutzer-Profil (Alter, Wissensstand, etc.). Eine inhaltliche Planungskomponente würde dann geeignete inhaltliche Bausteine nach vordefinierten Regeln in Abhängigkeit von Kontextinformationen

aus der Wissensbasis extrahieren und zu Präsentationen zusammenstellen. Diesen Präsentationen werden dann die Aktionen der Puppet zugeordnet (Gesten, Mimik, etc.). Im BEAT System [Cassell 2004] wird diese Zuordnung bereits vorher durchgeführt, da der Zeitaufwand sonst zu groß ist. Zur Optimierung der Performance ist es häufig sinnvoll bzw. notwendig, die Präsentation bereits weitgehend vor der Laufzeit vorzubereiten und nur bestimmte Teile während der Laufzeit an den aktuellen Kontext anzugleichen. Dadurch kann ein guter Ausgleich zwischen Glaubwürdigkeit (durch Einbezug des Kontexts) und Performance erreicht werden.

In dem Zusammenhang steht auch der Name „Director“ für diese Komponente. Der Regisseur (engl. „Director“) bekommt die Szene vorgegeben vom Drehbuchautor (hier der Tour-Autor, Designer oder ein Software-basierten Planer). Er hat die Freiheit die Szene an seine Darstellung anzupassen (zu interpretieren). Der Regisseur bringt die Szene auf die Bühne, wobei im Fall dieser Anwendung die Bühne der Bildschirm ist.

Da unterschiedliche Direktoren für ihren speziellen Anwendungskontext (bzw. ihre Kontextereignisse) verantwortlich sind, ist zusätzlich eine Koordinierungseinheit notwendig.

Mediator

Diese Koordinierungskomponente wird hier Mediator genannt, da sie zwischen den Direktoren vermittelt. Der Mediator entscheidet, ob ein Director aktiv sein kann. Ein aktiver Director kontrolliert die Bildschirmausgabe. Das Ziel ist eine einfachere Erweiterbarkeit und Austauschbarkeit von Direktoren. Beispielsweise könnte relativ schnell ein neuer Direktor hinzugefügt werden, der sich speziell um die Ausgabe von Warnungen kümmert (wie z. B. beim Betreten einer naturgeschützten Zone).

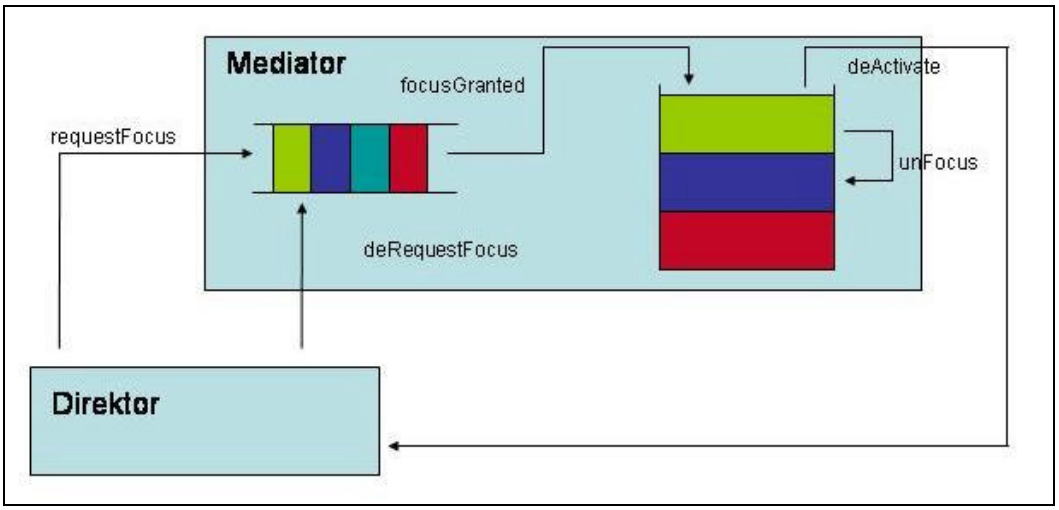

Abbildung 36: Schnittstelle zwischen Mediator und Director

Um die Koordination zu realisieren, bietet die Mediator-Komponente eine Schnittstelle, wie sie in Abbildung 36 verdeutlicht ist. Damit ein Director arbeiten kann, muss er zunächst den Fokus innerhalb der Applikation haben (d. h. als Director im Vordergrund arbeiten dürfen). Dies geschieht über eine requestFocus()-Anforderung an den Mediator.

Durch die Interaktion mit dem Mediator kann ein Direktor drei unterschiedliche Zustände einnehmen:

- Focused: Er kontrolliert die Bildschirmausgabe und reagiert auf Benutzeraktionen (z. B. der Benutzer klickt auf einen Button).
- Unfocused: Er kontrolliert die Bildschirmausgabe. Kann also im Hintergrund Updates durchführen. Reagiert aber nicht auf Aktionen des Benutzers.
- De-Aktiv: Der Director hat keine Kontrolle über die Bildschirmausgabe und reagiert nicht auf Benutzer-Aktionen.

Ein Director muss die Änderung seines Zustands auf „focused“ beim Mediator explizit anfordern. Dafür hat jeder Direktor eine Priorität. Ist die Priorität höher als beim fokussierten Direktor wird dieser auf „unfocused“ gesetzt und der den Fokus anfordernde Director auf „focused“ gesetzt. Alle Directoren, die gerade „focused“ oder „unfocused“ sind, sind in einen Stack eingetragen. Sollte ein Direktor seine Aufgabe beendet haben, setzt er sich selber auf De-Aktiv und wird damit aus dem Stack herausgenommen. Die Anträge für die Änderung des Zustands werden in einer Priority-Queue gesammelt. Sollte der Kontext, in dem der Antrag gestellt wurde, nicht mehr gültig sein und somit die Zustandsänderung keinen Sinn mehr ergeben, dann kann der Direktor seinen Antrag zurückziehen.

Reaktion auf Ereignisse

Der Direktor reagiert auf Ereignisse. Grob gesagt gibt es zwei verschiedene Ereignisse. Zu einem gibt es Kontextereignisse. Diese werden durch eine Kontextänderung wie das Erreichen einer bestimmten Position oder das Eintreten einer speziellen Uhrzeit, ausgelöst. Diese Ereignisse kommen von der Kontext-Komponente. Für das Empfangen von Kontextereignissen muss sich der Director bei dem KontextManager für spezifische Kontextereignisse anmelden (push-Mechanismus) oder einen Kontext direkt abfragen (pull-Mechanismus).

Die zweite Variante sind GUI-Events, die bei Aktionen des Benutzers mit der Benutzeroberfläche auftreten. Diese Ereignisse werden von GUI-Widgets direkt an den Direktor weitergeleitet.

Sobald ein Ereignis eintritt, durch das der Direktor aktiv wird, muss als erstes der Fokus beim Mediator beantragt werden. Hat der Direktor den Fokus oder anders ausgedrückt, sobald er „seine Bühne“ hat, kann er anfangen, die Szenen zu „inszenieren“.

Manager-Plugins

Diese Komponenten übernehmen die Erzeugung von Szenen. Der Director-Controller übergibt jeweils den nächsten relevanten Akt an den verantwortlichen Manager (z. B. Game-Manager). Der Akt wird von dem Manager auf die Szenen transformiert. Dabei kann ein Akt mehrere Szenen beinhalten, wie z. B. bei einem Quiz, das aus einem Frage- und Antwort-Teil bestehen kann.

Die Szenen werden von den Managern erstellt. Es wird immer eine Szene an den Szene-Player übergeben. Die Szenen beschreiben den Aufbau der Szene, z. B. welches Bild, welcher Text und welche Animation angezeigt wird. Nach der Darstellung der Szene meldet sich der Szene-Player beim Director. Der Director entscheidet, ob eine weitere Szene für den Player erzeugt werden soll. Der Manager entscheidet auch, welche Bühne verwendet bzw. welcher Szene-Player verwendet wird. So kann der Tour-Manager Szenen in Popup-Windows oberhalb der Karte oder in Fullscreen Multimedia-Darstellungen als Bühne abspielen.

Verbindung zur Map-Layer

Eine Besonderheit ist der Zusammenhang zwischen Director und Karte. Diese Verbindung ist für die Director notwendig, die auf POI-Ereignisse reagieren, sei es durch das Erreichen eines POIs oder einer Benutzeraktion.

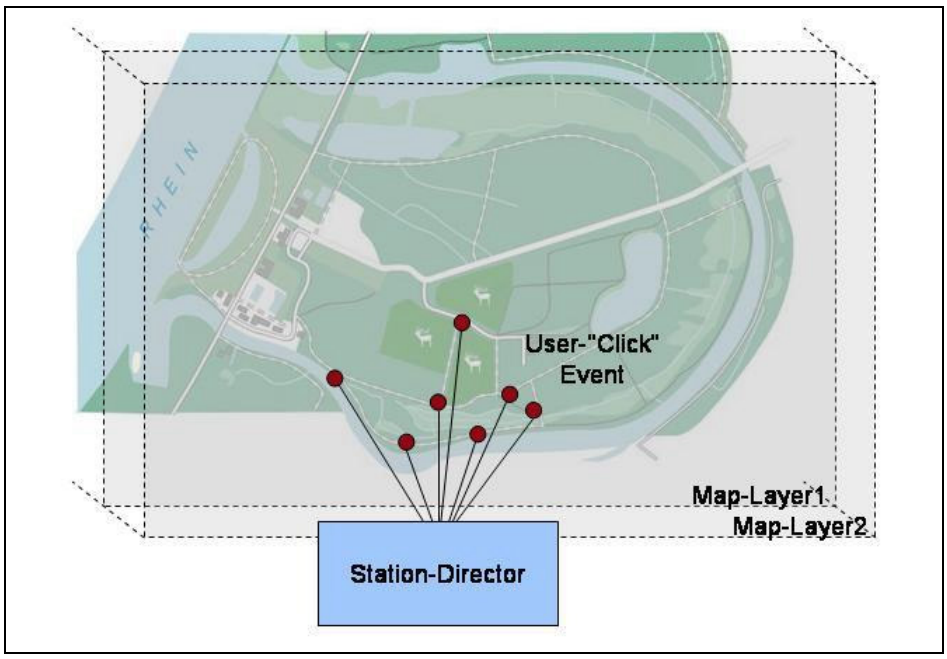

Abbildung 37: Schichten über der Karte

Um dem Benutzer die Möglichkeit zu geben, Informationen zu einem POI abzufragen, ohne dass der POI erreicht wurde, werden für die einzelnen POIs Widgets vom Director in

die Map-Layer eingetragen. Der Controller für die Widgets ist der verantwortliche Director. Er reagiert also direkt auf eine Benutzeraktion. Die Map-Layer sind Schichten oberhalb der eigentlichen Karte.

Der Vorteil der Map-Layer ist, dass jede Aktion auf der eigentlichen Karte (wie z. B. eine Vergrößerung) auch an die Map-Layer weitergeleitet wird, die Aktionen auf der eigentlichen Karte werden auch an die darüber liegenden Schichten weitergeleitet. Diese POI-Widgets sind kontext-sensitiv, d. h. sie sind ebenfalls bei dem Kontext-Manager eingetragen. Abhängig vom Kontext können sie ihre Gestaltung verändern, wie in dem Konzept von Patalaviciute vorgestellt [Patalaviciute 2005]. In Abbildung 37 ist das Prinzip grafisch dargestellt. Der Station-Director zeigt Informationen bei dem erreichen einer Station oder dem Klicken eines Benutzers auf eine Station an. Es ist also sinnvoll, wenn der Station-Director direkt von den Stationen über Benutzeraktionen benachrichtigt wird. In dem Bild sind auch die unterschiedlichen Schichten sichtbar. In einer weitern Schicht könnten andere POIs eingetragen sein, auf die ein anderer Director reagieren soll.

Im folgenden Unterkapitel wird kurz die Bedeutung der Editor-Komponente für das Konzept erläutert.

4.4 Editor und Vorverarbeitung der Animationserstellung

Der Editor ist eine Autorenkomponente, in der Touren für das gesamte System erstellt werden können. Aus Sicht des hier vorgestellten Konzepts sollte der Editor die folgende Funktionalitäten anbieten:

- Erstellen von Skripten zu Kontextereignissen und GUI-Ereignissen, wie das Klicken auf ein POI-Icon
- Kompilieren von Animationen aus den vom Autor um Verhaltensbeschreibungen erweiterten Texten

Zur Erzeugung der SVG-Ausgabe für die grafische Darstellung der animierten Figur werden die Basis-Animationen benötigt. In diesen sind die Animationen der animierten Figur gespeichert. Diese können direkt den Verhaltensbeschreibungen zugeordnet werden. Diese Vorverarbeitung hat den Vorteil, dass dadurch Ressourcen während der eigentlichen Laufzeit nicht verbraucht werden. Der Nachteil ist zu einem, dass bei jeder Änderung eines Textes der gesamte Ablauf wiederholt werden muss. Weiterhin hat dadurch der Director weniger Möglichkeiten, die Animationen dynamisch während der Laufzeit anzupassen. Hier muss also eine Abwägung stattfinden. Wenn dem System ausreichend Ressourcen zur Verfügung stehen, um Animationen dynamisch zu erstellen, sollte das gegenüber einer Vorverarbeitung bevorzugt werden.

Als nächstes soll die Erstellung der Animationssequenzen betrachtet werden. Ausgangstext ist dabei der folgende Codeabschnitt. Die verwendete Beschreibungssprache basiert auf der im BEAT System verwendeten Sprache (s. Kapitel 2.4)..

```
<GESTURE HAND="L" CMD="WINK">
 Hallo Besucher!
</GESTURE>
<MOUTH CMD="SMILE"/>
Willkommen
<GESTURE HAND="R" CMD="BEAT">
in
</GESTURE>
Rappenwörth
```

Für die Erstellung von Animationssequenzen sind folgende Schritte notwendig.

1. Zeiteinteilung („Scheduling“): Ziel ist, einen synchronen Ablauf von Animation und Sprachausgabe zu erreichen. Verwendet wird die Zeit, die für die Sprachausgabe der einzelnen Text-Abschnitte benötigt wird. Im Resultat steht also, wie lange die Ausgabe von „Hallo Besucher“ dauert. Dadurch ist bekannt, wann die einzelnen Gesten starten sollen.
2. Zuordnen der Animationen: Für die Verhaltensbeschreibungen wie die Winken-Handgeste gibt es Basis-Animationen. Diese Basis-Animationen bestehen entsprechend Kapitel 2.2.1 aus Preparation, Stroke und Retraction. Stroke, also die eigentliche Geste, kann entsprechend der Dauer der Sprachausgabe angepasst werden (wenn die Winken-Geste über den gesamten Text eingetragen wäre, dann würde der Charakter ununterbrochen winken). Nicht jedes Verhalten kann dementsprechend angepasst werden. Einige haben einen konstanten zeitlichen Ablauf.

In Abbildung 38 ist dieser Prozess noch einmal dargestellt. Der Autor schreibt den Text mit den Verhaltensbeschreibungen. Innerhalb des Editors werden aus diesen Beschreibungen von dem Animations-Compiler Animationssequenzen erstellt. Dafür werden Basisanimationen genutzt. Die fertige Animationssequenz wird in der character-Datei gespeichert. Letztendlich wird die Animationssequenz vom Szene-Player dargestellt.

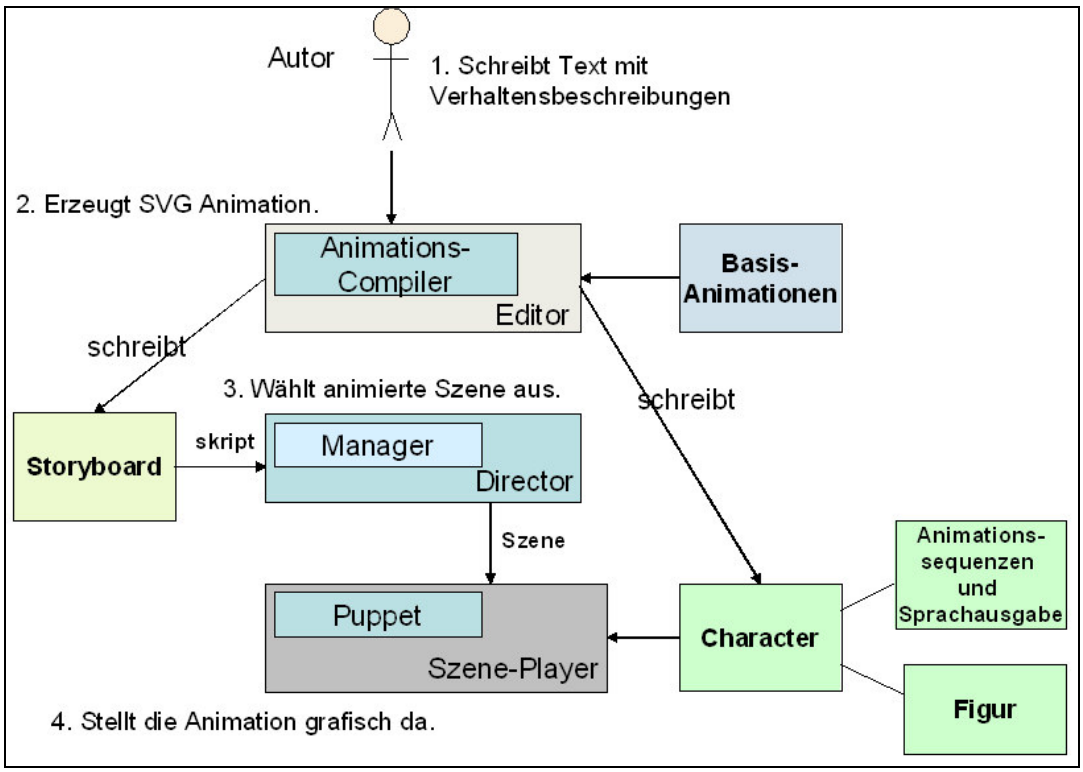

Abbildung 38: Prozess von der Erstellung bis zur Darstellung von Animationen

Analog können auch die Audiodateien erstellt werden. Es gibt zwei Möglichkeiten für die Sprachausgabe: Entweder wird die Sprachausgabe während der Laufzeit erzeugt oder mit der Animation vorbereitet. Das Audiofile wird zusammen mit den Animationen als Szenenbeschreibung in der Charakter-Beschreibung gespeichert.

4.5 Kontext-Manager

Der Kontext-Manager ist verantwortlich für die Verwaltung des Kontextes, in dem sich das System befindet. Die Idee ist, Kontextinformation an einer zentralen Stelle bereitzustellen und für unterschiedliche Komponenten des Systems verfügbar zu machen. Dabei wird die Trennung von Kontexterstellung und Kontextverwendung unterstützt. Düpmeier und Ruchter [Düpmeier 2004] haben relevante Kontexte für einen mobilen Naturführer in Abbildung 39 zusammengefasst. Dieses Kontextmodell soll als Grundlage für den Entwurf des Kontext-Managers dienen. In dem Modell werden drei Kontexte unterschieden, Umwelt („Environmental“), Geografisch („Geographical“) und Benutzer („User“). Dabei ist erkennbar, dass konkrete Kontextinformationen wie POI oder Jahreszeit („Season“) höherwertige Kontextinformationen sind, die aus Basiskontextinformationen (z. B. aktuelles Datum) und zusätzlichem Wissen („Sommer ist vom 21. Juni bis 22. September“) interpretiert sind. Die entwickelte Architektur verwendet die konzeptionelle Aufteilung in Integration und Aggregation ähnlich wie im Context Toolkit vorgestellt (s. Kapitel 2.6).

Der Kontext-Manager sollte also folgende Funktionalitäten anbieten:

- Interpretation von Kontexten zu höherwertigen Kontexten aus Basis-Kontexten und einer zusätzlichen Wissensbasis
- Aggregation von Kontexten: d. h., das Zusammenstellen von Kontexten zu thematisch zusammenhängenden Gruppen
- Polling-Schnittstelle zum Abfragen von Kontexten („pull"-Mechanismus)
- Callback-Mechanismus zur Benachrichtigung bei Kontextänderungen („push-Mechanismus).

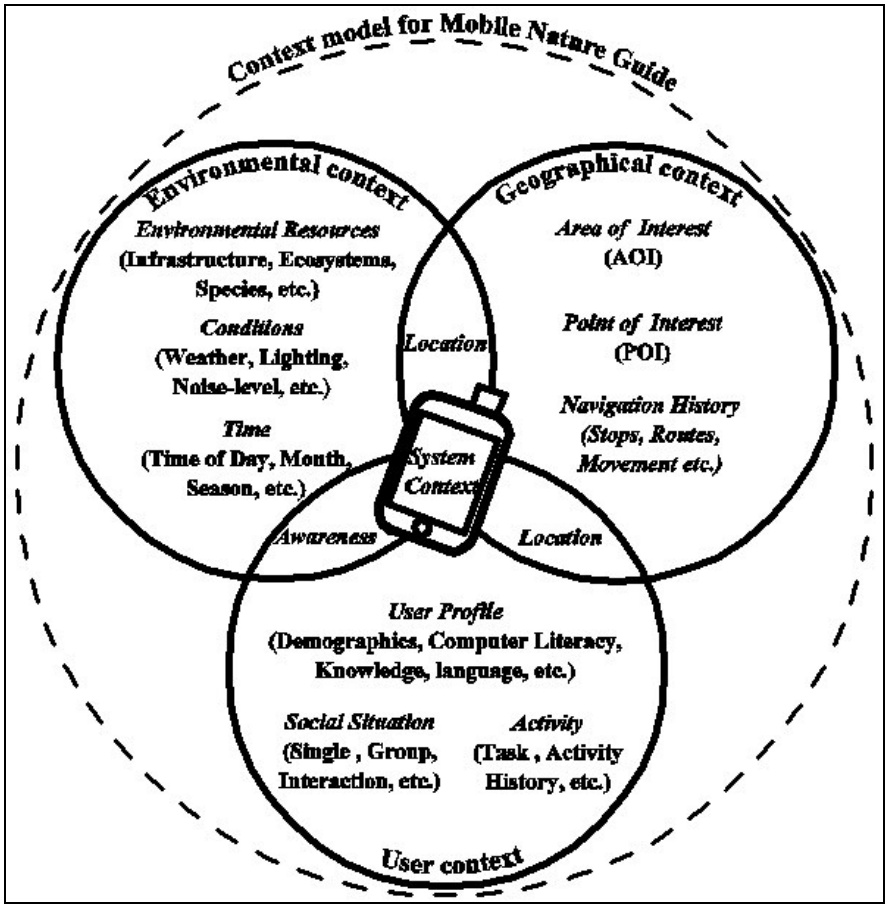

Abbildung 39: Context model für den Mobile Nature Guide

In: Düpmeier 2004

Das System soll so ausgelegt sein, dass einfach neue Kontexte hinzugefügt werden können. Die Idee der Kontext-Widgets bietet sich hierfür an. An dem Kontext interessierte Komponenten können sich an einem Widget für Kontextänderungen anmelden. Dadurch können sowohl Anwendungskomponenten als auch andere Kontext-Widgets bei Kontextänderungen benachrichtigt werden. So können Kontext-Hierarchien aufgebaut werden, die sukzessive höherwertige Kontextinformationen bilden. Neben der Interpretation bietet der Kontext-Manager die Aggregation von Kontextinformationen. Hierbei werden Kon-

textinformationen, die in einem bestimmten Zusammenhang stehen, zusammengefasst. Das kann z. B. entsprechend dem Kontext-Modell aus Abbildung 39 das User Profile sein, dass aus unterschiedlichen Benutzer-Kontexten wie Sprachkenntnisse („Languages“), Computervertrautheit („Computer Literacy“), usw. zusammengestellt ist.

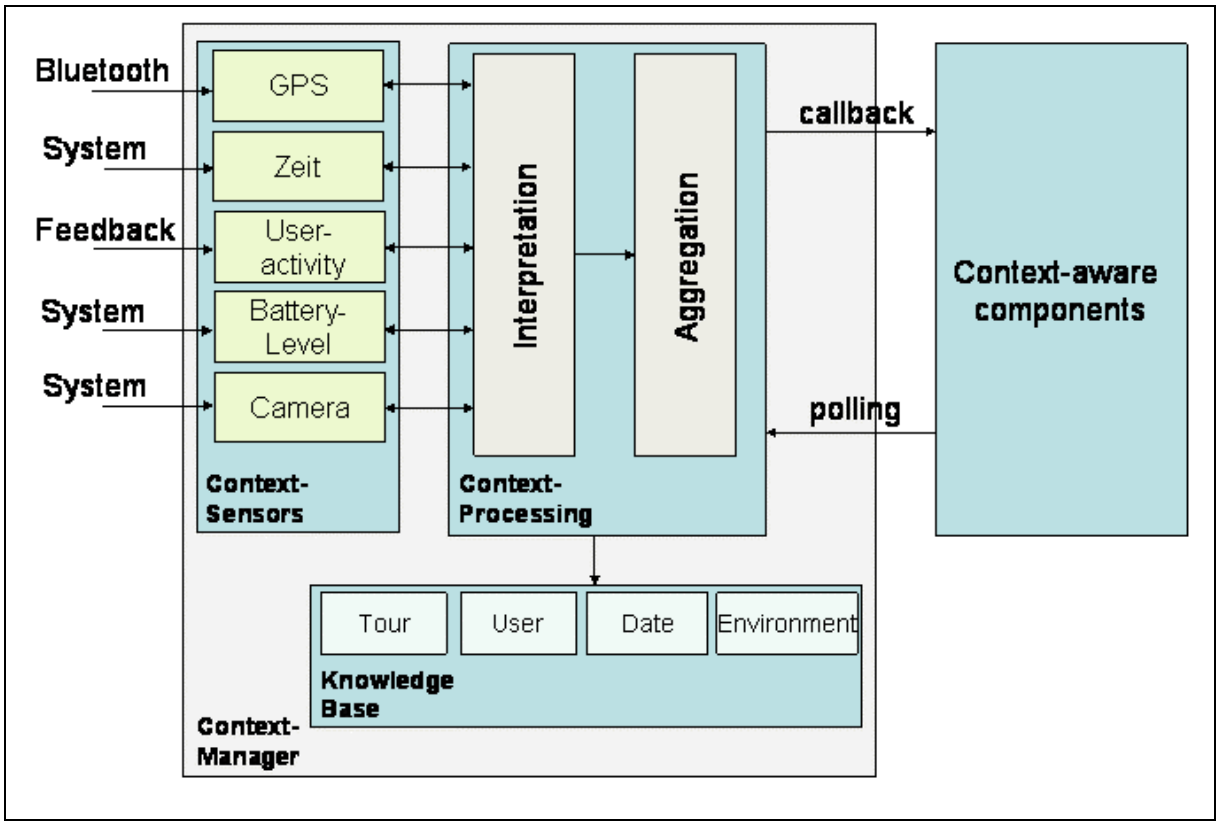

Abbildung 40: Kontext-Manager Architektur

In Abbildung 40 ist die erstellte Architektur dargestellt. Die grobe Aufteilung besteht aus Kontext-Sensoren, durch die die „Rohwerte“ in das System eingehen. Hier kommt z. B. eine geografische Koordinate über die Bluetooth-Schnittstelle des GPS-Empfängers an. Die Rohwerte werden innerhalb des Context-Processing („Kontext-Verarbeitung“) interpretiert. Als Beispiel wird ein Kontext-Widget für den Stations-Kontext beschrieben. Dieses Widget bekommt als Eingangswert die aktuelle GPS-Koordinate und verwendet zusätzlich aus der Wissensbasis („Knowledge Base“) die Tour-Informationen. Der Eingangswert wird dem Widget mitgeteilt, sobald der GPS-Sensor einen neuen Wert gelesen hat. Das Widget hat ein Attribut Station, in das die aktuelle Id der Station geschrieben wird. Wenn der Vergleich der GPS-Koordinate mit der Stationskoordinate aus dem Tour-Kontext ergibt, dass eine Station erreicht wurde, wird das Attribut dementsprechend geändert. Der Director, der abhängig von dem Stations-Kontext eine Szene anzeigen soll, hat sich bereits vorher bei dem Kontext-Widget für diese Kontextänderung angemeldet und wird über einen Callback-Mechanismus benachrichtigt.

Ebenso kann der Director einen Kontext abfragen. Wenn der Director die Szene darstellt, kann es sein, dass Kontextinformationen z. B. über die passenden Inhalte für den Benutzer nötig sind. Hierfür besteht die Möglichkeit das aggregierte Benutzerprofil abzufragen („polling“).

4.6 Szene-Player

Der Szene-Player stellt Szenen da, die von der Manager-Komponente innerhalb des Director erstellt wurden. Szenen bestehen aus Text, Animationen, Sprachausgabe und Interaktionselementen (z. B. Liste mit Auswahloptionen für ein Quiz). Szenen haben ein Template („Vorlage“), das den Aufbau auf dem Bildschirm beschreibt. Diese Szenen sollen synchronisiert mit der Sprachausgabe und den Animationen dargestellt werden. Dabei sollen folgende Funktionalitäten verfügbar sein:

- Play: Eine Szene abspielen
- Stop: Eine Szene anhalten und auf den Start zurücksetzen.
- Replay: Eine Szene wiederholen, entspricht eigentlich einem stop und play.
- Pause: Eine Szene anhalten und eventuell zu einem späteren Zeitpunkt wieder fortsetzen

Eine Szene wird entsprechend dem folgenden Codebeispiel als Szenebeschreibung an den Player-Controller übergeben. Für die Szene ist ein Template als Bühnenaufbau beschrieben, das den grafischen Aufbau der Oberfläche beschreibt. Wenn kein Template angegeben wurde, wird ein Default-Template verwendet. Weiterhin ist beschrieben, was die einzelnen Elemente darstellen sollen. Für die Darstellung werden GUI-Widgets aus der GUI-Library verwendet. Im folgenden Beispiel soll Text, Bild und Animation von der ersten Station der zweiten Tour angezeigt werden.

```
<SZENE template="station_content">
<TEXT id="tour2_station1_part2Text"/>
<IMAGE id="tour2_station1_imageRotBuche"/>
<PUPPET>
 <ANIMSEQ id= „tour2_station1_part2Anim”/>
 <SPEECH id=”tour2_station1_part2Speech”/>
</PUPPET>
```

Die hierbei interessanteste Komponente ist die Puppet, also die Figur für die Darstellung der vom Autor beschriebenen multimodalen Handlungen.

Puppet

Die Puppet repräsentiert die Agenten-Figur. Der Name bezieht sich auf die „Limited puppet“ von Shaw [Shaw 2004]. Sie fungiert als Darsteller für die Anweisungen aus der Szene-Beschreibung. Sollten keine Anweisungen vorliegen, kann die Komponente selbstständig Idle-Animationen anzeigen. Dies entspricht dem Konzept von André, dass das Verhalten der Persona (entspricht hier der Puppet) aus Anweisungen und eigenem Verhalten besteht (s. Kapitel 2.7 „Verhaltensplanung“). Auch wird die Trennung von Verhaltensskripterstellung und Darstellung eingehalten. Die Komponente erhält als Eingabe das Animationsskript vom Director. Die Puppet-Komponente sollte folgenden Funktionalitäten besitzen:

- Darstellen von Animations-Sequenzen
- Initialisieren des Charakters
- Übergänge zwischen Animationen ermöglichen
- Synchronisation mit der Audioausgabe

In Abbildung 41 sieht man den Zusammenhang der Puppet mit dem Szene-Player, Audio-Player und dem AnimationSeqPlayer. Der AnimationSeqPlayer ist für das Anzeigen der Animationen im Frametemplate verantwortlich. Der Szene-Player kann eine Animation starten, in dem er der Puppet über startAnimation die Anweisungen aus der SzeneBeschreibung übergibt. Die Puppet hat Zugriff auf die Animationen durch die AnimationLibrary. Damit kapselt die Puppet alle Funktionen zum Abspielen von Animationen und Audiofiles. Eine Animations-Sequenz besteht aus Basisanimationen, die bereits vor der Laufzeit von einem Designer zu einer Ablauf-Sequenz zusammengestellt wurden. Die Animations-Sequenzen sollten alle im gleichen neutralen Zustand der Figur anfangen und enden. Dadurch werden weiche Übergänge zwischen Animationen ermöglicht. Die Komponente hat nicht die Aufgabe die Übereinstimmung von Anfang- und Endpositionen zu überprüfen. Alle Animationen sind mit dem Charakter in einer Animationsbibliothek abgelegt. Der Charakter wird bei der Initialisierung der Komponente in die SVG-Grafik kopiert. Zu dem Charakter gehören auch die Idle-Animationen. Die sonstigen Animationen werden erst bei Bedarf in das Frametemplate kopiert.

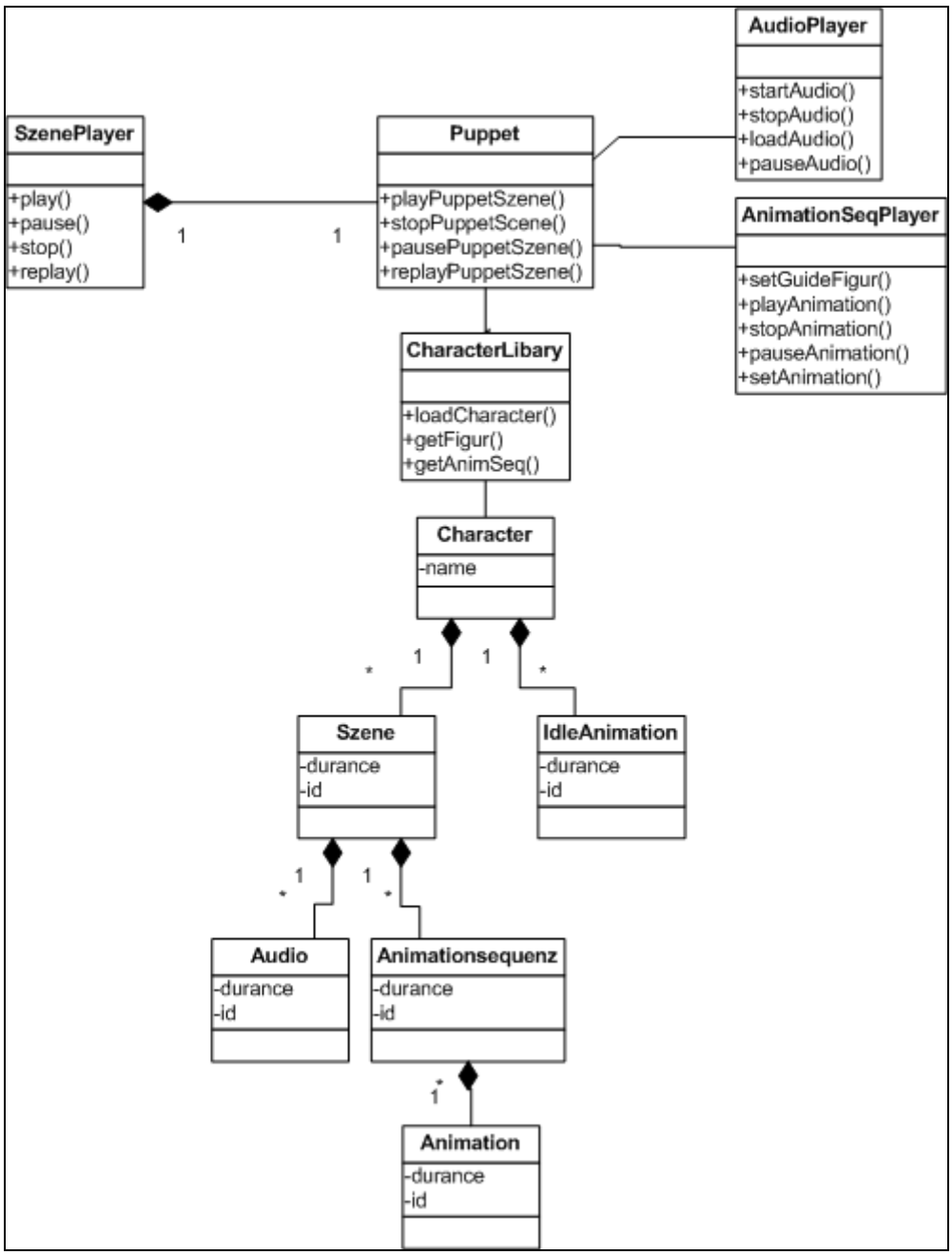

Abbildung 41: Klassendiagramm Puppet

Der Ablauf einer Animationsdarstellung ist in Abbildung 42 dargestellt. Die Animation wird möglichst parallel zum Audiofile gestartet. Beide sollten so ablaufen, dass sie synchron sind. Eine besondere Herausforderung ist das Unterbrechen und Fortsetzen einer Animation („pauseAnimation“). Dafür muss der aktuelle Zustand der Animation gespeichert werden. Ebenfalls wird von der Puppet die Sprachausgabe unterbrochen. Wenn keine Animation geladen ist, aber der AnimationSeqPlayer trotzdem über playAnimation aktiviert wird, dann zeigt er Idle-Animationen an, falls welche beim Laden des Charakters mit übergeben wurden. Wenn playAnimation() aufgerufen wird, ohne dass eine Animationssequenz geladen ist, dann sollen Idle-Animationen angezeigt werden, vorausgesetzt, dass für den Charakter Idle-Animationen definiert sind. Damit der Charakter

lebendiger erscheint wird, bei mehr als einer vorhanden Idle-Animation per Zufall zwischen den Animationen entschieden.

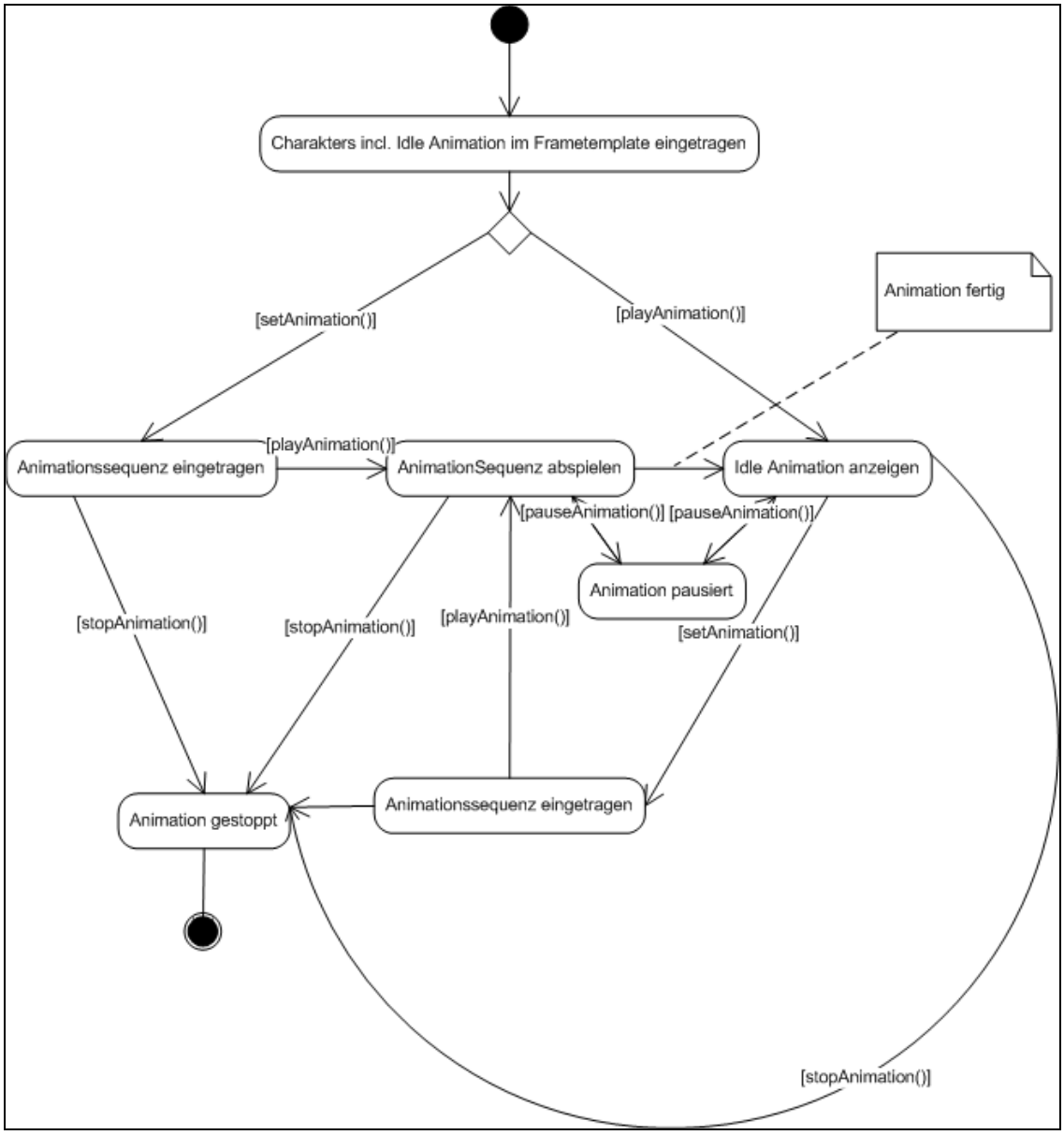

Abbildung 42: Zustandsübergangsdiagramm des Players

Für die Darstellung der Animationen wird ein Skelettmodell verwendet, das im folgenden Abschnitt mit den Animationen beschrieben wird.

Skelett-Modell

Das Skelett beschreibt den elementaren Aufbau des Charakters und die Bezeichnungen für die einzelnen Körperteile. Es gibt zwei wesentliche Vorteile eines solchen Ansatzes:

- Logische Zusammenhänge von Körperteilen. Untergeordnete Elemente bewegen sich mit.
- Texturen des Charakters können unabhängig von den Animationen ausgetauscht werden.

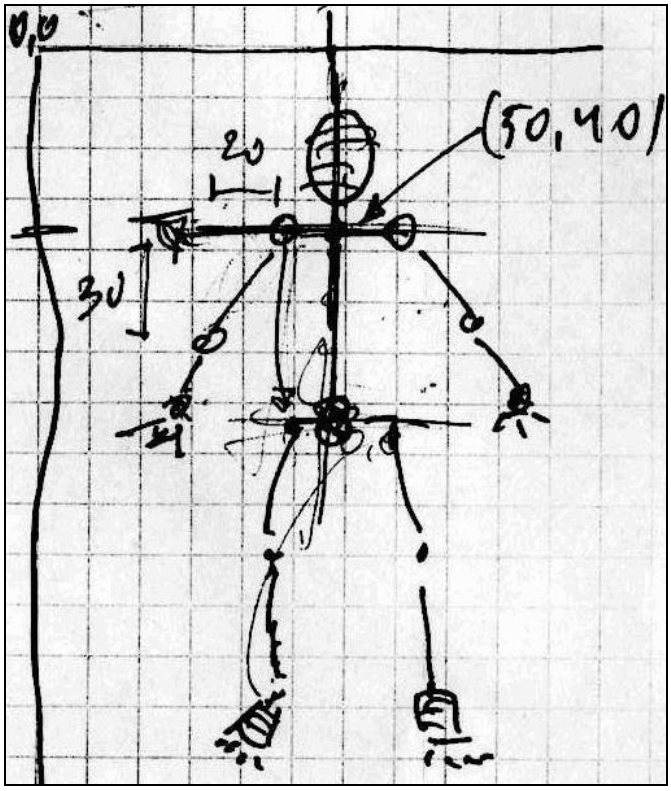

Abbildung 43: Konzept-Skizze für Skelett-Model

Der Körper hat, wie in Abbildung 43 dargestellt, hat einen hierarchischen Aufbau. Der Ursprung ist in der Hüfte, an der die Beine und der Oberkörper hängen. Am Oberkörper sind wiederum die Oberarme befestigt. Die Körperteile können von einer Animationsbeschreibung referenziert werden. Über die Körperteile werden Texturen für das passende Aussehen der animierten Figur gelegt. Dadurch passen die Animationen für unterschiedliche grafische Varianten.

Animationen

Animationen sind in Basisanimationen und Animationssequenzen aufgeteilt. Eine Animationssequenz besteht aus mehreren Basisanimationen. Die Basisanimationen beschreiben die Bewegungen von den im Skelett beschriebenen Körperteilen.

Animationssequenzen können nur am Stück abgespielt werden. Dies sind die Abschnitte, die vom AnimSeqPlayer dargestellt werden. Innerhalb einer Animationssequenz sind die Animationen Ereignis basiert („event-based“) gesteuert, d. h. sie werden nicht zu einem absoluten Zeitpunkt gestartet, sondern durch die Reaktion auf ein Ereignis (z. B. das Ende einer anderen Animation). Animationssequenzen sollten die Bedingung erfüllen, dass sie im gleichen Bild starten und enden. Dadurch sollten Übergänge zwischen Animationssequenzen nicht sichtbar sein. Neben den Animationssequenzen gibt es auch noch Idle-Animationen (z. B. das Wackeln mit dem Fuß). Diese werden immer dann abgespielt, wenn entweder die Animationsequenz zu Ende abgespielt wurde oder der Player eine Animation starten soll, ohne dass bereits eine Animation geladen wurde. Wenn mehr als eine Idle-Animation übergeben wurde, dann wird über ein Zufallsverfahren entschieden, welche Idle-Animation angezeigt wird.

In diesem Kapitel wurde eine Architektur vorgestellt, die es ermöglicht einen Agenten in Form eines Director in ein mobiles System einzubetten. Dieser Agent reagiert auf Veränderungen im Kontext des MobiNaf-Systems und arbeitet damit im Wesentlichen reaktiv. Als Beispiel wurde speziell auf einen Agenten eingegangen, der Stationsinformationen als Reaktion beim Erreichen einer Station anzeigt. Es sind auch weitere Directors denkbar, die auf andere Kontext-Ereignisse reagieren (z. B. das Ausgeben einer Warnung beim Betreten eines verbotenen Bereichs). Zur Verwaltung des Kontextes wurde eine Architektur für den Kontext-Manager vorgestellt, die es ermöglicht höherwertige Kontexte zu erzeugen. Diese Kontexte können dem restlichen System bereitgestellt werden. Abhängig vom Kontext (wie das Erreichen einer Station) werden vom Autor geschriebene Skripte dargestellt. Um eine hohe Bindung des Systems mit dem Anwender zu erreichen, ist eine grafische, Skelett-basierte animierte Figur als grafische Repräsentation eines Agenten erstellt worden.

Im folgenden Kapitel soll die Implementierung von ausgewählten Konzepten der Architektur erläutert werden.

Kapitel 5: Implementierung

Aufgrund der Komplexität der Gesamtapplikation konnten im Rahmen der Diplomarbeit nicht alle Komponenten implementiert werden. Die Aufgabe der Diplomarbeit war es, einige wesentliche Teile zur Überprüfung des Gesamtkonzeptes zu implementieren und in die bestehende MobiNaf-Implementierung zu integrieren. Da die Architektur der bestehenden MobiNaf-Applikation zur Integration des vollständigen Agenten Konzeptes, wie es in Kapitel 4 erarbeitet wurde, zunächst restrukturiert werden muss, sollte ein vereinfachtes Konzept für den Director in die bestehende MobiNaf-Implementierung integriert werden. Dies basiert im Wesentlichen auf der Ersetzung des in der MobiNaf-Applikation enthaltenen DescriptionViewer durch eine Mischung aus Director und Szene-Player. Weiterhin soll für die Realisierung eine Komponente implementiert werden, die eine SVG-implementierte Puppet darstellen kann, die auf dem Skelett-Konzept basiert. Mit diesen beiden Komponenten lässt sich dann das Gesamtkonzept eines Agenten in einer ersten Form innerhalb der bereits existierenden MobiNaf-Applikation verdeutlichen und testen.

5.1 Implementierungsumgebung

Das bisher bestehende System wurde von Sobek [Sobek 2004] implementiert. Die Implementierungsumgebung war aus dem bestehenden Projekt vorgegeben. Folgende Technologien wurden deshalb in dieser Arbeit für die Implementierung verwendet:

- SVG 1.1(„Scalable Vector Graphics“)-Format für die Darstellung der grafischen Benutzeroberfläche. Sowohl die Karte wie auch alle weiteren Oberflächenelemente wie Buttons, Text-Elemente, etc. werden in SVG erstellt.
- Intensis eSVG Viewer für die Darstellung der SVG Grafiken. Dieser SVG-Viewer ist als ActiveX-Komponente in die Applikation integriert.
- T-Mobil MDA II als PocketPC-.basierte Laufzeit Umgebung. Das Gerät kombiniert ein Mobiltelefon mit einem PDA.
- Visual embedded C++ 4.0 von Microsoft als Entwicklungsumgebung.
- MS-XML 4.0 XML-Parser von Microsoft.

Die grafische Gestaltung des Charakters wurde in SVG 1.1 realisiert. SVG ist eine XML Beschreibungssprache für 2D-Vektorgrafik.

Der Charakter besteht aus dem statischen Design und aus den Animationen. Hier soll kurz auf die SVG-Animationstechnologie eingegangen werden. Dabei werden auch andere zentrale Begriffe mit eingeführt. Die SVG-Animation basiert auf der SMIL-Animations-Spezifikation. Daraus kommen die Elemente:

- animate: Zuweisung von unterschiedlichen Werten über einen Zeitverlauf.
- set: Wie Animate nur für nicht numerische Werte, wie die Sichtbarkeit von Elementen.
- animateMotion: Bewegungen an einem Pfad ausrichten.
- animateColor: Farbzuordnungen ändern.

Zusätzlich wurde folgendes Animations-Element zum SVG Standard hinzugefügt:

- animateTransform: Translationen und Rotationen über einen Zeitverlauf.

Ein Beispiel für eine Animation ist hier abgebildet. In dem Beispiel wird ein Rechteck (das rect-Element) von den Koordinaten 300,100 nach 0,100 in 9 Sekunden verschoben.

```
<rect id="RectElement" x="300" y="100" width="300" height="100"
      fill="rgb(255,255,0)"  >
  <animate attributeName="x" attributeType="XML"
           begin="0s" dur="9s" fill="freeze" from="300" to="0" />
```

Die Animation in dem Beispiel ist an einer Zeitleiste ausgerichtet. Das gesamte Dokument hat genau eine Zeitleiste, die bei jedem Laden des Dokumentes bei 0 startet. Ein Vorteil von SVG ist die relativ intuitive Syntax. Deswegen werde ich die nötigen Begriffe anhand der Implementierungsbeispiele einführen.

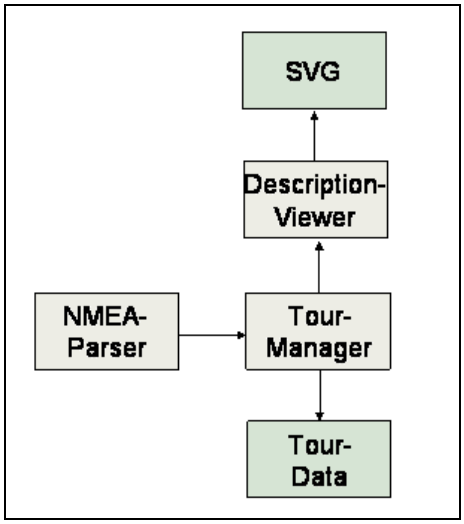

Abbildung 44: Bisherige Architektur zur Anzeige der Stationsinformationen

Die Architektur der Applikation wird in [Sobek 2004] beschrieben. An dieser Stelle wird nur der relevante Teilbereich betrachtet, in dem die Darstellung der Stationsbeschreibungen erfolgt. Eine grobe Übersicht der Architektur des Bereichs findet sich in Abbildung 44. Der TourManager ist die zentrale Komponente für die Verwaltung von Tourinformationen. Die Komponente kennt die Position der Stationen und vergleicht diese mit der aktuellen Position des Geräts. Die aktuelle Position wird von einer GPS-Komponente geliefert (NMEA-Parser). Wenn die Positionen übereinstimmen, wird die dazugehörige Stationsbeschreibung an den DescriptionViewer übergeben. Diese Komponente erzeugt dazu die passende SVG-Grafik.

5.2 Director

Abbildung 45 zeigt, wie der Director rückwärtskompatibel zur bestehenden Implementierung in die existierende MobiNaf-Applikation eingefügt werden kann. Hierbei leiten sich die alte DescriptionViewer Klasse und der Director von einer gemeinsamen Basisklasse BaseDescriptionViewer ab, die das vom Rest von MobiNaf verwendete Interface definiert. Der Director selber ist intern aus den Komponenten DialogManager, GameManager und Puppet Komponente zusammengesetzt. Der eigentliche SzeneManager, der für die Erzeugung der grafischen Darstellung verantwortlich ist, ist direkt in den Director integriert.

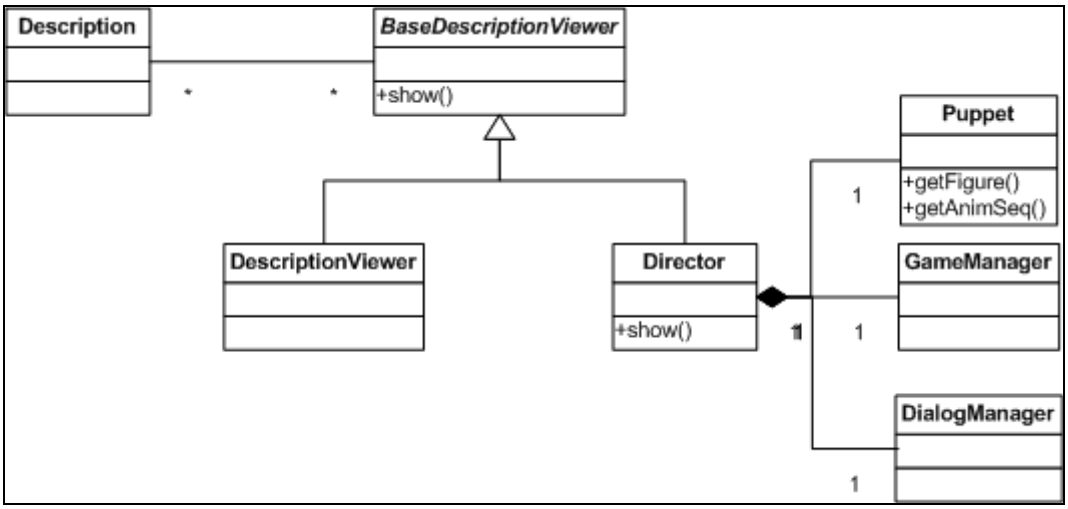

Abbildung 45: Klassendiagramm des Directors in der bestehenden Implementierung

Die Stationsbeschreibungen („Description“) mit den vom Autor geschriebenen Inhalten wurden um ein Attribut für die darzustellende AnimationsSequenz erweitert. Anhand der ID kann von der Puppet die anzuzeigende Sequenz geladen werden. Die Koordination übernimmt der Director. Die Animationssequenz wird an den SzeneManager übergeben.

Der SzeneManager besteht aus einer C++ und JavaScript Implementierung. Das Aufrufen von Funktionen in der Applikation vom SVG Dokument aus ist relativ aufwendig [Sobek 2004]. Daher hat es sich als sinnvoll erwiesen die Funktionen des Szene-Player, die direkt auf Benutzeraktionen reagieren sollen, wie z. B. das anhalten und wiederholen von Szenen, direkt in das SVG Dokument zu integrieren. Im folgenden Abschnitt wird der Aufbau der Puppet genauer betrachtet.

5.3 Puppet und Animation

Die Puppet, also die grafische Darstellung des Charakters, besteht während der Laufzeit aus der statischen Figur und den Animationssequenzen. Die Figur selbst ist zusammengesetzt aus dem Skelett und den Texturen. Bei der Initialisierung des DescriptionViewer wird die Figur über den SzenePlayer in das SVG-Dokument eingetragen. Als Test-Charakter wurde ein Ranger entworfen, da dieser gut in den Anwendungskontext als Naturführer passt. In einem ersten Schritt wurden Bezeichner für alle animierbaren Körperteile eingeführt.

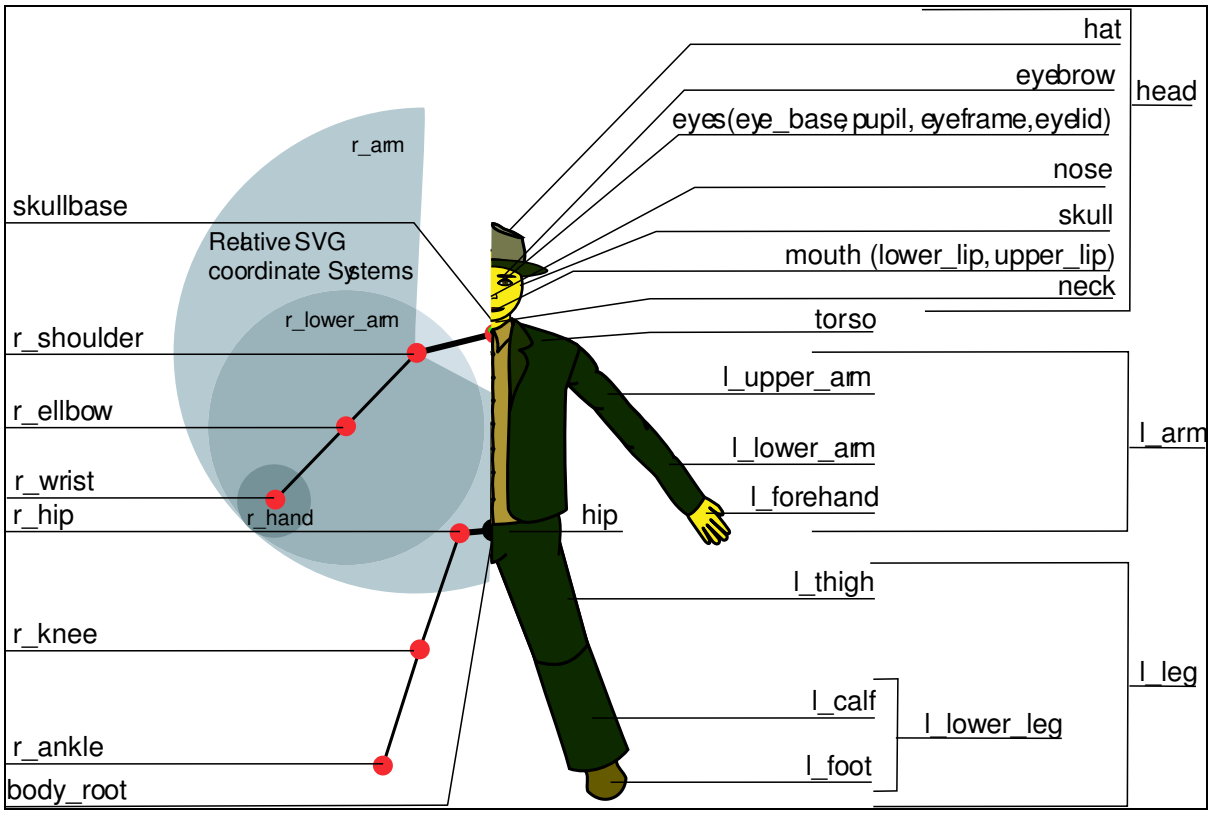

Abbildung 46: Ranger-Charakter

In Abbildung 46 ist der Charakter mit Textur und Skelett dargestellt. Hier sieht man, dass z. B. der rechte Arm („r_arm“) aus dem Unterarm und der Hand besteht. Diese Körperteile sind über Gelenke verbunden, z. B. das Armgelenk („r_ellbow“). Bei jedem Gelenk wird

ein eigenes lokales Koordinatensystem definiert. Der Vorteil dieser Methode ist, dass Animationen aus Rotationen und Translationen von Körperteilen beschrieben werden können. Ebenso kann eine Textur wie die Jacke des Rangers relativ leicht ausgetauscht werden, ohne dass die Animation neu gestaltet werden muss. Das Problem ist, dass hier nur 2D-Bewegungen beschrieben werden können. Dadurch kommt es zu Schwierigkeiten bei einfachen Aktionen wie dem Drehen der Hand oder Bewegungen des Arms hinter und vor den Körper. Als SVG Beschreibung sieht der Charakter aus wie im folgendem Beispiel. Hier ist der rechte Arm dargestellt. Durch das „g"-Element wird eine Gruppe von untergeordneten Elementen zusammengefasst. Das ermöglicht das ausführen von Transformationen auf die Gruppe. Der Unterarm („lower arm") ist Teil des Koordinatensystems des Arm. Die Animation referenziert z. B. die id „r_arm" um eine Rotation auf den gesamten Arm auszuführen. Die Pfade („path") beschreiben die Texturgrafik, so steht in der Gruppe „r_upper_arm" die Grafik für das obere Stück des Ärmels. Teile der Grafik wurden für eine bessere Lesbarkeit durch „…" ersetzt.

```
<!-- Right Arm -->
<g id="r_arm" transform="matrix(1,0,0,1,-38.9008,-97.6653)">
       <circle r="2" cx="0" cy="0" fill="rgb(0,0,255)"/>
       <polyline points="0,0 -13,17" stroke="rgb(0,0,0)"
fill="none"/>
       <!-- Right upperarm -->
       <g id="r_uppper_arm">
             <path style="fill:#454128;fill-opacity:1.0000000;…"/>
             <path d="M 96.829976,71.593236 C 99.903942,…"/>
             <path d="M 112.14492,84.438029…",id="path4826"/>
       </g>
       <!-- Right Lowerarm -->
       <g id="r_lower_arm" transform="matrix(1 0 0 1 -36.8908 …)">
             <circle r="2" cx="0" cy="0" fill="rgb(0,128,0)"/>
             <polyline points="0,0 -7,12" stroke="rgb(0,0,0)"/>
             <!-- Right Hand -->
                   <g id="r_hand" transform="matrix(1,0,0,…32,39)">
```

In einem SVG-Dokument darf keine ID zweimal vorkommen. Die Animationssequenzen sind allerdings aus den Basisanimationen zusammengebaut. Deswegen wurde ein Bezeichner-Schema eingeführt um die Eindeutigkeit der IDs zu gewährleisten. Das Schema setzt sich zusammen aus der Tournummer, der Stationsnummer, der DescriptionPart-Nummer, dem Bezeichner der Basisanimation und einer fortlaufenden Nummer, falls Basisanimationen mehrfach in einer Animationssequenz verwendet werden. Die Animationssequenzen verwenden die IDs der Körperteile zur Referenzierung. Die Animation wird also auf das zu bewegende Körperteil angewendet. Im folgenden Beispiel ist die Beschreibung einer Animation aus einer Animationssequenz für das Abwärts bewegen des linken Arms zu sehen.

```
<animateTransform id="tour1_station4_descPart2_arm_down_1"
   xlink:href="#l_arm" attributeName="transform" type="rotate"
   begin="tour1_station4_descPart2_head_left_6.end" by="2" from="0"
   to="32" dur="0.1s" calcMode="spline" keySplines="0.69 0.13 0.14
   0.71" fill="freeze" />
```

Bei diesem Beispiel sind mehrere interessante Eigenschaften zu sehen. Zu einem wird die Animation durch das Ende einer weiteren Animation gestartet („head_left_6.end“). Zum anderem sind am Ende keySplines angegeben. Mit keySplines können der Animationsbewegung während des Ablaufs unterschiedliche Geschwindigkeiten zugeordnet werden. Hier wird es verwendet um eine sogenannte „Ease In and Out“-Bewegung zu schaffen, das bedeutet, dass die Bewegung langsam startet dann beschleunigt und wieder langsamer wird. Dadurch wirken die Bewegungsabläufe natürlicher.

Eine Problematik, die vorher schon erwähnt wurde, ist die Animation von 3D-Bewegungen in einer 2D-Beschreibungssprache. Dieses Problem tritt z. B. bei der Darstellung von Handgesten auf. Es gibt verschiedene Handansichten, mal ist die Hand von der Seite, mal vorne und mal gespreizt für eine Wink-Bewegung zu sehen (Abbildung 47).

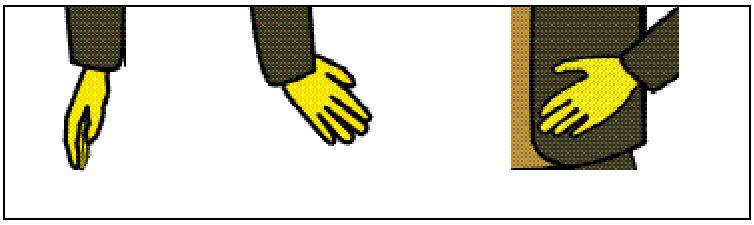

Abbildung 47: Verschiedene Handformen

Dafür werden verschiedene Ansichten des selben Körperteils vorbereitet, zwischen denen gewechselt werden kann, indem eine andere Ansicht angezeigt wird..

```
<set id=" tour1_station4_descPart2_l_forehand1"
   xlink:href="#l_forehand" attributeName="visibility" to="visible"
   begin="tour1_station4_descPart2_head_left_6.end" dur="1s"
   fill="freeze" />
```

Die Darstellung einer Stationsbeschreibung mit dem Charakter ist in Abbildung 48 zu sehen. Der Ranger führt gerade eine Zeigegeste auf das Bild einer Buche aus.

Abbildung 48: Der Ranger präsentiert die Stationsbeschreibung

5.4 Animations-Compiler

Für die Geste in Abbildung 48 hat der Autor folgende Beschreibung in seinen Text eingefügt.

```
Es ist <GESTURE CMD="point_middle" HAND="left"> eine Buche
</GESTURE> die hier …
```

Der Animations-Compiler verwendet eine Animationsbibliothek aus Basisanimationen, die zu Animationssequenzen kombiniert werden können. Dafür müssen zuerst innerhalb der Basisanimationen die Zeiten richtig gesetzt werden. Eine Basisanimation besteht im Normalfall aus einer Preperation, Stroke, Retraction (s. Kapitel 2.2.1 „Gesten“). Der Stroke kann angepasst werden. Der Stroke wird solange wiederholt, bis der GESTURE-Tag wieder geschlossen wird. Für die Zuordnung der Zeiten zu den Animationen muss die Länge der Sprachausgabe bekannt sein. Bisher wurde die TTS Funktionalität nur simuliert, da weder TTS oder Audioausgabe innerhalb der Applikation möglich ist. Dafür wurde pro Buchstabe eine konstante Zeit von 80 ms[20] pro Buchstabe verwendet. Anhand der damit ermittelten Zeiten konnten den Verhaltensbeschreibungen Anfangszeiten zugeordnet werden. Als nächster Schritt wird aus der Animationsbibliothek die passende Animation für die Geste ausgelesen. Ein Ausschnitt aus der Animationsbibliothek ist im folgenden Codebeispiel dargestellt. In dem Beispiel ist die Basis-Animation für eine linke Zeigegeste auf mittlere Höhe beschrieben. Hier muss jetzt für das „begin“ der preperation die dem

[20] Dies ist nur ein geschätzter Wert der sich als praktisch erwiessen hat. Der Wert basiert nicht auf konkreten sprachlichen Untersuchungen.

Verhalten zugeordnete Zeit eingetragen werden. Angenommen „Es ist“ hat 480 ms benötigt, dann würde in die erste Animation der preperation Gruppe begin=“480ms“ eingetragen werden. Außerdem muss die Wiederholung des stroke entsprechend der Zeit zwischen den GESTURE-Elementen gesetzt werden.

```
<animations>
 <gesture_l>
    <animation name="point_gesture_1">
       <preperation>
          <animateTransform id="l_arm_up_point1" xlink:href="#l_arm"
attributeName="transform" type="rotate" by="2" from="0" to="-65"
dur="1.5s" calcMode="spline" keySplines="0.69 0.13 0.14 0.71"
fill="freeze"/>
          <set id="r_hand_point1" xlink:href="#l_hand_point"
attributeName="visibility" to="visible"
begin="l_lowerarm_up1.begin+0.25s" dur="1s" fill="freeze"/>
       </preperation>
       <stroke repeat=”open”>
          <animateTransform id="l_hand_wave1" xlink:href="#l_hand"
attributeName="transform" type="rotate" begin="l_lowerarm_up1.end" …
repeatCount="4"/>
          <animateTransform id="l_hand_wave2" xlink:href="#l_hand"
attributeName="transform" type="rotate"
begin="l_hand_wave1.end+0.1s" …" keySplines="0.69 0.13 0.14 0.71"
fill="freeze"/>
       </stroke>
       <prestroke>
          <animateTransform id="l_lowerarm_down2"
xlink:href="#l_lower_arm" … from="0" to="60" dur="0.5s"
calcMode="spline" keySplines="0.69 0.13 0.14 0.71" fill="freeze"/>
          <animateTransform id="l_arm_down3" xlink:href="#l_arm"
attributeName="transform" type="rotate"
begin="l_lowerarm_down2.begin+0.1s" … " keySplines="0.69 0.13 0.14
0.71" fill="freeze"/>
       </prestroke>
    </animation>
 </gesture_l>
…
```

Nach dem alle Verhaltenbeschreibungen nach diesem Verfahren bearbeitet wurden ergibt sich die Animationssequenz für den Beschreibungsteil. Diese Animationssequenz wird in der characters-Datei für die Puppet-Komponente gespeichert.

Im folgenden Abschnitt werden einige Probleme beschrieben, die in Verbindung mit dem Einsatz des eSVG-Players und SVG allgemein aufgetreten sind.

5.5 Implementierungsprobleme

Während der Implementierung hat sich herausgestellt, dass die Kommunikation zwischen der Applikation und dem SVG-Dokument die Reaktionsfähigkeit des Systems sehr einschränkt. Daher hat es sich als sinnvoll erwiesen die Aufrufe von Funktionen im SVG-Dokument zu reduzieren. Im eSVG Konzept existiert bereits ein so genanntes „Exchange Objekt“, das von der Komponente für den Datentransfer zwischen Applikation und SVG-Dokument verwendet wird. Über dieses Exchange-Objekt können Texte von der Applikation in das Dokument und umgekehrt übertragen werden. Dadurch dass ein XML Objekt über die Microsoft XML-Funktion get_xml() in Text umgeformt werden kann und dieser Text über die Javascript Funktion parseXML() wieder in ein XML-Objekt zurückgewandelt werden kann, lassen sich ganze SVG-Strukturen dynamisch in das Dokument integrieren. Über dieses Verfahren sollen die Figur und die Animationen in das Dokument eingetragen werden. Das Problem dabei ist, dass jegliche Skripte die innerhalb der Animationen verwendet wurden, nicht mehr funktionieren. Daher war das Ereignis basierte Auslösen von Animationen nicht möglich. Um trotzdem die Animationssequenzen zu verwenden, wurden diese bereits vor der eigentlichen Laufzeit in das SVG-Dokument eingetragen.

Ein weiterer Nachteil ist die fehlende Unterstützung von Pfad-Animationen durch die eSVG-engine. Dies wäre gut geeignet für Mundbewegungen, bei dem der Mund als Pfad definiert ist und entsprechend dem Viseme (s. Kapitel 2.2.3 „Sprache“) animiert wird.

Leider ist die Darstellung der Animationen (besonders der Rotationen) auf dem PDA sehr langsam. Dadurch gibt es zum Teil große Sprünge innerhalb von Animationen.

Allgemein zu SVG bietet SVG 1.1 wenig Kontrolle über die Zeitleiste. Im SVG 1.2 working draft ist vorgesehen, dass innerhalb eines Dokuments unterschiedliche Zeitleisten existieren können. Für jede Zeitleiste lassen sich einzelne Geschwindigkeiten angeben. Außerdem können die Zeitleisten pausiert werden. Bisher ist das Anhalten und Fortsetzen von Animationen nur im Adobe SVG-Viewer möglich.

Nachdem in diesem Kapitel interessante Aspekte der Implementierung hervorgehoben wurden, soll im abschließenden Kapitel 6 ein Resümee der Arbeit und ein Ausblick auf zukünftige Entwicklungen gegeben werden.

Kapitel 6: Fazit und Ausblick

6.1 Fazit

Gemäß der Aufgabenstellung wurde in Kapitel 2 zunächst eine Übersicht zu relevanten Themen aus dem Bereich der life-like characters gegeben. Dabei wurden wichtige Begriffe, wie multimodale Kommunikation, Kontext, Verhaltensplanung und Emotionen erläutert. Das dabei gewonnene Verständnis der Gesamtthematik hat sich besonders für das Design der Architektur als hilfreich erwiesen. Hier konnten Konzepte von anderen Projekten übernommen oder adaptiert werden. Wegen der großen Bandbreite an Forschungsgebieten, die in diesem Bereich relevant sind, ist es sehr vorteilhaft, eine Übersicht der wichtigsten Konzepte dieser Gebiete zu haben. Diese Übersicht wird durch Kapitel 2 bereitgestellt und kann auch in zukünftigen Projekten verwendet werden.

Anhand der Grundlagen aus Kapitel 2 wurden in Kapitel 3 verschiedene Systeme betrachtet und verglichen. Dabei wurde auch eine umfangreiche Tabelle über bisherige und aktuelle Entwicklungen von life-like characters zusammengestellt.

Das Hauptziel dieser Arbeit war die Erstellung eines Konzeptes für einen emotionalen pädagogischen Agenten in dem Szenario eines mobilen Naturführers. Eine kontext-basierte Architektur wurde in Kapitel 4 dargestellt. Der Agent ist in der Lage, abhängig vom aktuellen Kontext dem Benutzer angepasste Tourinhalte zu präsentieren. Dafür stehen verschiedene multimodale Kommunikationsformen zur Verfügung, sei es über Sprache, Gestik, Mimik oder eher traditionell über Text und Bilder. Die Architektur ist flexibel genug, um neben dem Agenten auch noch weitere kontext-basierte Services zu integrieren.

Auf den Konzepten aufbauend wurde auch eine Implementierung von ausgewählten Komponenten durchgeführt. Dabei ist besonders die Implementierung einer grafischen Darstellung des animierten Agenten unter SVG wichtig gewesen, da diese die Basis für die übrigen Konzepte darstellt. Es wurde wie in der Aufgabenstellung gefordert SVG und C++ für die Implementierung verwendet. Weiterhin wurde in Kapitel 5 beschrieben, wie bereits in die existierende MobiNaf-Applikation Teilfunktionalitäten der in Kapitel 4 vorgestellten Konzepte integriert werden können.

6.2 Ausblick

Aufgrund der großen Bandbreite der Aufgabe konnten einzelne Bereiche nicht im Detail betrachtet werden. In einer zukünftigen Arbeit sollte deswegen die Planungskomponente des Agenten vertieft betrachtet werden. Dafür müssten den Stationsinhalten Meta-Informationen hinzugefügt werden, die dem Agenten Kriterien bieten, um Texte für den Benutzer individuell zusammenzustellen. Hier sind Werkzeuge wie CLIPS und SOAR (s.Kapitel „Links“) interessant.

Die Dialog-Führung durch den Agenten und das Durchführen und Anzeigen von Spielen sind zwei weitere Gebiete die vertiefte eigenständige Betrachtungen erfordern.

In diesem Zusammenhang ist auch die Generierung von User-Profilen innerhalb der Kontext-Komponente wichtig. Anhand eines User-Profils kann das System besser an den Benutzer angepasst werden.

Als Teil der Animationen wäre es interessant auch Frame-basierte Animationen für den Agenten zu testen. Der Vorteil könnte in einer weniger CPU-Ressourcen verbrauchenden Darstellung liegen. Momentan sind die Animationen nur sehr eingeschränkt möglich. Hier sind besonders die Agenten Laura und Gina zu betrachten, da für deren Animation eine Frame-basierte Vektorgrafik Lösung verwendet wurde.

Wegen den Performance-Problemen, die bereits in den Arbeiten von Sobek (2004) und Patalaviciute (2005) beschrieben sind, sollten weiterhin neue SVG-Viewer getestet werden. Besonders interessant könnte hier der für Ende 2005 angekündigte Renesis Viewer[21] sein, der auch als PocketPC-Version erscheinen soll.

Ein Defizit der momentanen Implementierung ist die nur sehr rudimentäre GUI-Bibliothek. Hier gibt es neuerdings das SPARK SVG-GUI Framework[22]. Es wäre interessant zu wissen ,wie dieses Framework auf dem PocketPC läuft. Insbesondere da in der Architektur aus Kapitel 4 eine GUI-Bibliothek verwendet wird, könnte SPARK den Implementierungsaufwand wesentlich reduzieren.

Es wurde als Teil dieser Diplomarbeit für das MobiNaf-Projekt eine umfangreiche Animations-Bibliothek für den Ranger-Charakter erstellt. Dabei war ein besonderer Nachteil, dass es zu dem Zeitpunkt der Implementierung keinen brauchbaren SVG-Editor für die Erstellung von Animationen gab. Daher wurden viele Animationen mit einem einfachen Texteditor erstellt. Dies ist zu einem Teil ein Vorteil von SVG, aber für einen praktischen Einsatz ist diese Methode viel zu umständlich. Daher sollten in Zukunft weitere SVG Animationseditoren auf ihre Brauchbarkeit getestet werden.

Momentan wird innerhalb des MobiNaf-Projektes die Entwicklung einer Editor Komponente für einen Autor vorangetrieben, eine Integration des Animations-Compilers wäre hier eine sinnvolle Erweiterung. Die Verwendung des BEAT-Systems für das automatische Hinzufügen von Charakterbeschreibungen wäre ebenso als Teil des Editors interessant. Dafür müsste getestet werden, inwiefern das System für deutsche Texte einsetzbar ist bzw. welche Änderungen gemacht werden müssten.

Verschiedene Systeme für die Sprachausgabe auf dem PDA wurden in Kapitel 2 vorgestellt. Sprachausgabe ist zum einen für den Agenten wichtig und ermöglicht zusätzliche neue Anwendungsszenarien. Im Vergleich der unterschiedlichen TTS Systeme sind besonders das System von SVOX als kommerzielle Lösung und IMS als open-source

[21] Renesis SVG Viewer: http://www.gosvg.net/
[22] SPARK (SVG Programmers' Application Resource Kit): http://spark.sourceforge.net/

System interessant. Eine Integration in eine zukünftige MobiNaf-Version sollte getestet werden.

In dieser Arbeit wurde ein Ansatz vorgestellt, der eine zentrale Kontext-Komponente verwendet. Kontext ist ein wichtiges Konzept für die hier vorgestellte Architektur und für mobile Geräte allgemein. Eine Implementierung dieser Komponente macht eine Vielzahl von weiteren Nutzungsszenarien denkbar. Daher sollte im geplanten Re-Design der MobiNaf-Applikation der Kontext-Manager eine der zentralen System-Komponenten sein.

Wenn die angesprochenen Probleme, wie die nicht ausreichende Performance des SVG-Viewers und die Integration eines TTS-Systems sowie die Implementierung des Kontext-Managers erfolgreich gelöst sind, wird das in der Einleitung gestellte Szenario realisiert werden können.

Literaturverzeichnis

Ackerman, M., Augmenting the Organizational Memory: A Field, Study of Answer Garden. University of California, Irvine, 1994

Arafa, Y., Kamyab, K., Mamdani, E., Character Animation Scripting Languages: A comparison, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, Melbourne, Australia, July, 14-18, 2003, ACM Press, New York, N.Y., pp. 920 - 921

Arafa, Y., et al. Two approaches to Scripting Character Animation, Proceedings First Int Conf Autonomous, Agents & Multi-Agent Systems, Proceedings of the Workshop on Embodied Conversational Agents, Bologna, Italy, July, 14-18, 2002, ACM Press

Arafa, Y., Mamdani, E., Scripting embodied agents behaviour with CML : Character Markup Language, Proceedings of the 8th International Conference on Intelligent User Interfaces (IUI03), Miami, Florida, January, 12–15, 2003, pp. 313-316

Arafa, Y. Kamyab, K, Mamdani, E, Toward a unified Scripting Language: Lessons Learned from Developing CML and AML, In: Prendinger, H., Ishizuka, M.(Eds.) Life-Like Characters, Springer, 2004, pp. 39 - 63

Argyle, M., Bodily Communication, Methuen & Co. Ltd (New York), 1988

Aschenberner, B., Weiss, C., Phoneme-Viseme Mapping for German Video-Realistic, IKP - Working Paper NF 11, 2005

Bartneck, C., How convincing is Mr. Data's smile?, User Modeling and User-Adapted Interaction, 11, 2001, pp. 279-295

Bartneck, C., Integrating the OCC Model of Emotions in Embodied Characters, Workshop on Virtual Conversational Characters: Applications, Methods, and Research Challenges, Melbourne, Nov, 29, 2002

Benzmüller, R., Grice, M., Trainingsmaterialien zur Etikettierung deutscher Intonation mit GTOBI, 1997, http://www.uni-koeln.de/phil-fak/phonetik/gtobi/index.html

Bickmore, T., Relational Agents: Effecting Change through Human-Computer Relationships, PhD Thesis, Media Arts & Sciences, Massachusetts Institute of Technology, Cambridge, 2003

Bickmore, T., Cassell, J., Social Dialogue with Embodied Conversational Agents, J. van Kuppevelt, L. Dybkjaer, and N. Bernsen (eds.), Natural, Intelligent and Effective Interaction with Multimodal Dialogue Systems, Kluwer Academic (New York)

Bos, J., Clark, R., Laaksolahti, J., Matheson, C., Oka, T., Steedman. M., Magicster, Language and Speech Architecture for Prototype 3, D2.5, 2004

Brna, P.; Cooper, B; Razmerita, L., Marching to the Wrong Distance Drum: Pedagogic Agents, Emotion and Student Modelling, Workshop on Attitude, Personality and Emotions in User-Adapted Interaction in conjunction with User Modelling, Sonthofen, July, 2001

Brusilovsky, P., Methods and techniques of adaptive hypermedia, User Modeling and User-Adapted Interaction 6, 2 - 3, 1996, pp. 87-129

Bühler, D., Häussler, J., Krüger, S., Minker, W., Flexible Multimodal Human-Machine Interaction in Mobile Environments, ECAI 2002 Workshop on Artificial Intelligence in Mobile System (AIMS), Lyon , France, 2002

Cassell, J., Vilhjalmsson, H.H., Bickmore, T., 2004. BEAT: the Behavior Expression Animation Toolkit, H. Prendinger and M. Ishizuka (eds.), Life-Like Characters. Tools, Affective Functions, and Applications, Springer, 2004, p. 163-185

Cassell, J., Sullivan, J., Prevost, S., Churchill, E., Embodied Conversational Agents. MIT Press (Cambridge), 2001

Cassell, J., A Framework For Gesture Generation And Interpretation, Cipolla, R. and Pentland, A. (eds.), Computer Vision in Human-Machine Interaction, Cambridge University Press (New York), 1998, pp. 191-215

Cassell, J., Bickmore, T., Billinghurst, M., Campbell, L., Chang, K., Vilhjálmsson, H., Yan, H., Embodiment in Conversational Interfaces: Rea, ACM CHI 99 Conference Proceedings, Pittsburgh, USA, May 15-20, 1999, ACM Press, pp. 520-527

Chen, G., Kotz, D., A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381, November 2000

Düpmeier, C., Ruchter, M., User Interface Architecture of a Mobile Guide for exploring the Wild, 6th International Conference on Human Computer Interaction with Mobile Devices and Services, 3rd International Workshop on "HCI in Mobile Guides", Glasgow, GB, September, 13-16, 2004

Ekman, P., Friesen, W. V., & Ellsworth, P., Emotion in the Human Face: Guidelines for Research and an Integration of Findings, Pergamon Press (New York), 1972

Ekman, P., An argument for basic emotions, Cognition and Emotion 6 (3-4), (1992), pp. 169 - 200,

Etcoff, N.L., Magee, J.J., Categorical perception of facial expressions, Cognition 44, (1992), pp. 227 – 240

Franklin, S., Graesser, A., Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents, Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, Budapest, Hungary, August, 12-13, 1996, Springer Verlag,, London, 1996, pp. 21-35

Fleischman, M. and Hovy, E, Emotional Variations in Speech-Based Natural Language Generation, International Natural Language Generation Conference, Arden House, USA, July, 1-3, 2002

Ganeshan, R., Johnson, W.L., Shaw, E., and Wood, B.P. Tutoring Diagnostic Problem Solving, Proceedings of the Fifth Conference on Intelligent Tutoring Systems, Montréal, Canada, June, 19-23, 2000, Springer Verlag, London, pp. 33-42

Graesser, A.C., Lu, S., Jackson, G.T., Mitchell, H., Ventura, M., Olney, A., Louwerse, M.M., AutoTutor: A tutor with dialogue in natural language, Behavioral Research Methods, Instruments, and Computers, 36, (2004), pp. 180-193

Ham, S.H., Environmental Interpretation – A Practical Guide for People with Big Ideas and Small Budgets, Golden, Colorado, USA, Fulcrum/ North American Press, 1992

Ishizuka, M., Tsutsui, T., Saeyor, S., MPML: A multi-modal presentation markup language with character agent control functions, Proc.(CD-ROM), WebNet 2000 World Conf. on the WWW and Internet, San Antonio, Texas, USA, October 30 – November 4, 2000

Johnson, W. L., Rickel, J., Animated Pedagogical Agents: Face-to-Face Interaction in Interactive Learning Environments, International Journal of Artificial Intelligence in Education, Vol. 11, (2000), pp. 47-78

Johnson, W. L., LaBore, Catherine, Chiu, Yuan-Chun, A Pedagogical Agent for Psychosocial Intervention on a Handheld Computer, AAAI Fall Symposium on Dialogue Systems for Health Communication, October 21-24, 2004, Arlington, Virginia, AAAI Press, Menlo Park, USA, 2004, pp. 64-70

Kendon, A., Gesticulation and speech: two aspects of the process of utterances, M.R. Key (Ed.), TheRelation Between Verbal and Nonverbal Communication, Mouton, 1980, pp. 150-168

Knopp, S , Synthese und Koordination von Sprache und Gestik für virtuelle multimodale Agenten, Verlagsgesellschaft Aka GmbH (Berlin), 2003

Kovar, L., Gleicher, M., Pighin, F., Motion Graphs, Proceedings of the 29th annual conference on Computer graphics and interactive techniques, San Antonio Texas, ACM Press, 2002, pp. 473-482

Lester, J., Voerman, S., Towns, G., Callaway, C. Cosmo, A life-like animated paedagogical agent with deictic believability, André, E. (Ed.), Proc. of the IJCAI-97 Workshop on Animated Agents: Making them Intelligent, Nagaoya, Japan, August, 25, 1997, Taylor & Francis, 1999, pp. 61-69

Lester, J., Animated agents and problem-solving effectiveness: A large-scale empirical evaluation, Artificial Intelligence in Education, pp. 23-30, 1999

Li, Q., Nakano, Y., Okamoto, M., Nishida, T., Highlighting Multimodal Synchronization for Embodied Conversational Agent, Proceedings of the 2nd International Conference on Information Technology for Application, Harbin, China, January 8-11, 2004, Macquarie Scientific Publishing

McNeill, D., Hand and mind: What gestures reveal about thought, University of, Chicago Press (Chicago), 1992

Newell, A.F., Murray, I.R., Arnott, J.L., Dye, R., Cruickshank, G., Carter, K.E.P., Human Factors Studies of Speech-Driven Word Processors: Experiments Using a Simulated Natural Rate Large Vocabulary Continuous Speech Recognition System, Ainsworth, W.A. (Ed.), Advances in Speech, Hearing and Language Processing, Vol. 2, J.A.I. Press, London, 1992

Ortony, A., Clore, G.L, Collins, A., The Cognitive Structure of Emotions, Cambridge Press (Cambridge), 1988

Patalaviciute, V., SVG basierte Karten für mobile Guidesysteme: Erstellung, Gestaltung und Interaktionsmechanismen unter besonderer Berücksichtigung von Karten für den Naturtourismus, Master Thesis, FH Karlsruhe, 2005

Pelachaud, C., Bevacqua, E., Mancini, M., Speaking with Emotions, AISB 2004, Convention: Motion, Emotion and Cognition, University of Leeds, UK, 2004

Prendinger, H., Ishizuka, M.(Eds.) Life-Like Characters, Springer, 2004

Rickel J., Johnson W. L., Animated Agents for Procedural Training in Virtual Reality: Perception, Cognition, and Motor Control. In Applied Artificial Intelligence 13, pp. 343-382, 1999

Ruchter, M., Düpmeier, C., Projektskizze MobiNaf, unveröffentlichter Bericht, 2003

Russell, J.A., A circumplex model of affect. Journal of Personality and Social Psychology, 39, (1980), pp. 1161-1178

Schultz, P. W., Emphasizing with Nature: The effects of Perspective Taking on Concern for Environmental Issues, Journal of Social Issues, Vol. 56, No. 3, (2000), S.391-406

Shaw, E., LaBore, C., Chiu, Y.-C., & Johnson, W.L., Animating 2D digital puppets with limited autonomy, Proceedings of the International Symposium on Smart Graphics, Banff, Alberta, May 23-25, 2004, Springer Verlag, pp. 1-10

Sobek, R., unveröffentlichter Bericht, 2004

Stone, B., Lester, J., Dynamically Sequencing an Animated Pedagogical Agent, Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon, August, 4-8, 1996, AAAI Press / The MIT Press, pp. 424-431

Van Mulken, S., Andre, E., Mueller, J, The Persona Effect: How Substantial Is It, Proceedings of HCI '98, Sheffield, England, September, 1998, Springer, 1998, pp. 53-56

Wachsmuth, I., Sowa, T., Gesture and Sign Language in Human-Computer Interaction, International Gesture Workshop, GW 2001, London, UK, April 18-20, 2001, Springer, 2002

Wahlster, W., SmartKom: Symmetric Multimodality in an Adaptive and Reusable Dialogue Shell, Proceedings of the Human Computer Interaction Status Conference, Berlin, June 3-4, 2003, DLR, pp. 47 – 62

Wooldridge, M., Intelligent Agents, Multiagent Systems, A Modern Approach to Distributed Artificial Intelligence, Gerhard Weiss (Ed.), The MIT Press, 2000, pp. 27-78

Abkürzungen

2D:	zweidimensional
3D:	dreidimensional
eSVG:	Embedded SVG
Flash:	Probrietäres Vektorgrafikformat von Macromedia.
FPS:	Franes per Second
GIF:	Graphics Interchange File
GPS:	Global Positioning System
GUI:	Graphical User Interface
MobiNaf:	Mobiler Naturführer
MPEG-4:	Motion Picture Expert Group
PC:	Personal Computer
PDA:	Personal Digital Assistant
POI:	Point of Interest
SMIL:	Synchronized Multimedia Integration Language
SVG	Scalable Vector Graphics
SVGZ:	Komprimiertes SVG
TTS:	Text-To-Speech, Erzeugung von Sprachausgabe von Text
VRML:	Virtual Reality Modeling Language
XML:	Extended Mark-Up Language

Links

Zu life-like characters:

Forschungsgruppen:
Center of Advanced Reseach in Technology for Education (CARTE): http://www.isi.edu/isd/carte (Stand: 08/05)
Deutsche Forschungszentrum für Künstliche Intelligenz (DFKI): http://www.dfki.de (Stand: 08/05)
University of Bari, Research Group on Intelligent Interfaces: http://www.di.uniba.it/intint/index.html (Stand: 08/05)
Center of Research of Innovative Technologies for Learning (RITL): http://ritl.fsu.edu/index.html (Stand: 08/05)
Magicster: http://www.ltg.ed.ac.uk/magicster (Stand: 08/05)
The IntelliMedia Initiative, North Carolina State: http://research.csc.ncsu.edu/intellimedia/index.htm (Stand: 08/05)

Firmen:
Veepers Pulse 3D Puppets: http://www.pulse3d.com (Stand: 08/05)
Oddcast: http://www.oddcast.com/home (Stand: 08/05)
Haptek: http://www.haptek.com (Stand: 08/05)

Frameworks:
CU-Animate (CSLU Toolkit): http://cslu.cse.ogi.edu/toolkit/index.html (Stand: 08/05)
BEAT (Behavior Expression Animation Toolkit): http://www.isi.edu/~hannes/beat/ ,(Stand: 08/05)
X-FACE: http://xface.itc.it/index.html (Stand: 08/05)
Galatea: http://slab.hil.t.u-tokyo.ac.jp/~galatea/ (Stand: 08/05)
MS-Agent: http://www.microsoft.com/msagent/default.asp (Stand: 08/05)

Personen:
Erin Shaw (CARTE): http://www.isi.edu/~shaw/ (Stand: 08/05)
Hannes Högni Vilhjálmsson (CARTE): http://www.isi.edu/~hannes/index.html (Stand: 08/05)
Fiorella de Rosis (Univ. Bari): http://www.di.uniba.it/intint/people/fior.html (Stand: 08/05)
Zsófia Ruttkay (Univ. Twente): http://wwwhome.cs.utwente.nl/~zsofi (Stand: 08/05)
Amy L. Baylor (Florida State): http://garnet.acns.fsu.edu/%7Eabaylor (Stand: 08/05)
Elisabeth André (Univ. Augsburg, DFKI): http://www.dfki.uni-sb.de/~andre (Stand: 08/05)
Thomas Rist (DFKI): http://www.dfki.uni-sb.de/~rist (Stand: 08/05)
Timothy W. Bickmore (Northeastern Uni): http://www.ccs.neu.edu/home/bickmore (Stand: 08/05)
W. Lewis Johnson (CARTE): http://www.isi.edu/isd/johnson.html (Stand: 08/05)
Helmut Prendinger (Japan National Institute of Informatics): http://research.nii.ac.jp/~prendinger (Stand: 08/05)
Justine Cassell (Northwestern Uni): http://www.soc.northwestern.edu/justine (Stand

Natural language generation:
SPUD: http://www.cs.rutgers.edu/~mdstone/nlg.html#overview (Stand: 08/05)

Problem solving framework:
SOAR: http://sitemaker.umich.edu/soar (Stand: 08/05)

Expert Systems:
Clips: http://www.ghg.net/clips/CLIPS.html (Stand: 08/05)

Dialog Planung:
TrindiKit: http://www.ling.gu.se/projekt/trindi/trindikit (Stand: 08/05)

Agenten Technologien:
Open Agent Architecture (OOA): http://www.ai.sri.com/~oaa (Stand: 08/05)

Zu SVG:

Mailing-Liste:
SVG-Developers Maillingliste: http://groups.yahoo.com/group/SVG-developers (Stand: 08/05)

Tutorials:
SVG Tutorial1: http://svg.tutorial.aptico.de/index.php (Stand: 08/05)
SVG Tutorial2: http://www.fh-wedel.de/~si/praktika/MultimediaProjekte/SVG/SVG_Tutorial_mi3794/index.htm (Stand: 08/05)
SVG Leaning by Coding: http://svglbc.datenverdrahten.de/ (Stand: 08/05)
Scale a vector: http://www.scale-a-vector.de/index.htm (Stand: 08/05)
SVG Animation: http://srufaculty.sru.edu/david.dailey/svg/SVGAnimations.htm (Stand: 08/05)
Carto.Net: http://www.carto.net/papers/ (Stand: 08/05)

Demos:
Dotus comus: http://www.dotuscomus.com/svg/#demos (Stand: 08/05)
SVG-Whiz: http://svg-whiz.com/samples.html (Stand: 08/05)
Kevin Lindsey samples: http://www.kevlindev.com/samples/index.htm (Stand: 08/05)
Carto.Net: http://www.carto.net/papers/svg/samples/ (Stand: 08/05)

SVG GUI Frameworks:
SPARK: http://spark.sourceforge.net (Stand: 08/05)
CGUI: http://homepage.usask.ca/~ctl271/cgui/index.shtml (Stand: 08/05)

Allgemein:
SVG.org: http://svg.org (Stand: 08/05)
W3C SVG : http://www.w3.org/Graphics/SVG (Stand: 08/05)