# Realisierung nutzeradaptiven Interaktionsverhaltens für mobile Assistenzroboter

Dissertation
zur Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)

vorgelegt der
Fakultät für Informatik und Automatisierung
der Technischen Universität Ilmenau

von

**Dipl.-Inf. Steffen Müller**
geboren. am 14.06.1979 in Jena

Tag der Einreichung: 18.12.2015
Tag der wissenschaftlichen Aussprache: 27.06.2016

Gutachter 1) Prof. Dr. H.-M. Groß
2) Prof. Dr. G. Rigoll
3) Prof. Dr. H.-J. Böhme

urn:nbn:de:gbv:ilm1-2016000339

## Kurzfassung

Im Zentrum dieser Dissertation steht die soziale Assistenzrobotik. In den letzten Jahren hat die Bedeutung dieses Teilgebietes der mobilen Robotik stark zugenommen und zusammen mit der Diversifizierung robotischer Fähigkeiten hat sich die Nutzergruppe hin zur breiten Masse mit potentiellen technischen Laien gewandelt. Aus dieser Situation heraus erwachsen an die Interaktionsfähigkeiten sozialer Assistenzroboter umfangreiche Anforderungen. Insbesondere stehen in dieser Arbeit die Multimodalität der Interaktion und die Anpassungsfähigkeiten an den konkreten Nutzer im Vordergrund.
Am Beispiel eines Serviceroboters für die häusliche Gesundheitsassistenz, wie er in einem vom Autor mit bearbeiteten Forschungsprojekt realisiert wurde, wird zunächst der Analyse- und Entwurfsprozess für dessen Umsetzung geschildert. Im Anschluss daran wird gezeigt, wie sich aus der Systemspezifikation eine mehrschichtige Systemarchitektur ableiten lässt, welche auch auf andere Robotikanwendungen übertragbar ist.
Der Fokus liegt dabei auf der modularen Realisierung einer Ablauf- und Dialogsteuerung. Um dem System eine Persönlichkeit zu geben und ein im Langzeiteinsatz akzeptierbares Dialogverhalten zu generieren, wurde ein framebasierter Dialogmanager konzipiert und umgesetzt. Dabei wurden Aspekte wie Modularität durch ein App-Konzept, leichte Erweiterbarkeit und die Möglichkeit, nutzeradaptive Dialoge zu realisieren, berücksichtigt.
Im Kern des vorgestellten Dialogsystems kommt eine gänzlich neue Methode der probabilistischen online-Planung von Dialogsequenzen zum Einsatz. Ein eigens konzipiertes Realweltexperiment konnte zeigen, dass es mit diesem System möglich ist, anhand von systeminternen aber auch nutzergetriebenen Bewertungen, das Dialogverhalten im Rahmen von durch den Designer vorgegebenen Freiheiten zur Laufzeit zu optimieren.
Die Gestaltung des robotischen Gesundheitsassistenten wurde durch weitere Teilsysteme abgerundet. Unter diesen spielen verschiedene taktile Sensoriken und ein Emotionsmodell eine entscheidende Rolle für die Realisierung eines liebenswerten Begleiters. Letztendlich konnte in sehr erfolgreichen teils mehrtägigen Nutzerstudien mit Senioren die Praktikabilität des entwickelten Interaktionskonzepts und der Systemarchitektur nachgewiesen werden.

## Abstract

The central topic of this thesis concerns social service robotics. In recent years this branch of mobile robotics in general has seen increasing interest. Due to increasing capabilities and growing fields of application of such robots, the group of potential users has changed. Unexperienced users raise extensive requirements regarding the interaction capabilities of such robots. The multimodality of human-robot dialog and its adaptivity regarding user's preferences and needs are in the focus of this thesis.

First, the analysis and specification process for such a system is explained by means of an example, which is a service robot for health assistance in home environments, as it has been developed in a research project at which the author participated. Following this, it is shown how a multi-layer system architecture is derived from that specification, which is applicable to other robotic applications as well.

Though the main focus is on a modular realization for the control structures and the dialog handling. In order to enable a long term acceptability of such a system and to give it a personality, a frame-based dialog manager has been designed and is explained in detail. Aspects of interest there are modularity by means of an app-concept, extendablility, and adaptivity of the interaction skills regarding users' qirks and demands.

In the core of the presented dialog system, there is a unique planning mechanism based on probabilistic reasoning in a factor graph model of the dialog going on. In a real world experiment it could be shown that this online learning concept is able to optimize dialog behavior regarding system internal as well as user driven reward signals.

During the implementation of the health assistant robot further system compontents have been developed in order to realize a likeable companion. Among them, there are two kinds of tactile sensors and an emotion model, which are presented in this thesis as well.

Finally, very successful real world user trials of the health assistant robot involving 9 elderly people are described to show that the presented concepts for system architecture and dialog modelling are viable.

## Danksagung

Ich möchte allen herzlich danken, die mich während der langen Zeit der Bearbeitung dieser Dissertation unterstützt haben.
Besonderer Dank gilt dabei Prof. Dr. Horst-Michael Groß, der mir die Gelegenheit gegeben hat, eine interessante Thematik zu finden, bei der mir die Freiheit gegeben war, zahlreiche Ideen auszuprobieren und auch meinem Basteltrieb nachzugehen.
Ebenso dankbar bin ich meiner Familie, die teilweise unter meiner Laune leiden musste, wenn das Schreiben an der Arbeit wieder einmal wichtiger war als andere vielversprechende Hamster. Ihr habt mich immer wieder motiviert, es letztlich doch zu einem erfolgreichen Ende zu bringen.
Auch den Kollegen am Fachgebiet NI&KR möchte ich für die fruchtbare und freundschaftliche Atmosphäre und die Unterstützung bei der Umsetzung unseres Projekts danken. Ohne euch hätte es nur halb so viel Spaß gemacht und ohne eure eigenen Arbeiten am Roboter wäre meine Dissertation nicht möglich gewesen.
Zu guter Letzt möchte ich den vielen Studentinnen und Studenten danken, die mir mit ihren Arbeiten immer wieder Gelegenheit gaben, in andere Facetten der Robotik einzutauchen und die somit jeweils ihren eigenen Beitrag zum Gelingen dieser Arbeit haben. Besonders hervorheben möchte ich dabei Sören Kalesse, der mit seiner Akribie bei der Aufarbeitung der probabilistischen Inferenzalgorithmen erst die Grundlage für diese Arbeit gelegt hat und der mir persönlich zum Freund geworden ist.

# Inhaltsverzeichnis

# 1

# Einleitung

Nachdem, Ende des letzten Jahrhunderts, die Robotik in der Industrie Einzug gehalten hatte, um dort mit Fähigkeiten wie Präzision und Ausdauer den Menschen bei zum Teil gefährlichen Arbeiten zu ersetzen, werden seit Beginn des 21. Jahrhunderts Roboter auch für die Zusammenarbeit mit dem Menschen konzipiert. Der Roboter soll dabei zum einen mit dem Menschen an einer Aufgabe arbeiten und zum anderen auch Dienstleistungen am Menschen erbringen. Letzteres soll hier als **Assistenzrobotik** verstanden werden und steht im Fokus dieser Arbeit. Die reinen Manipulationsaufgaben für Industrieroboter werden dabei durch Mobilität der Systeme und nicht manipulative Informationsdienstleistungen ergänzt.

Mit dieser Entwicklung hat sich auch die Nutzergruppe gewandelt und erweitert. Es treten nicht mehr nur technisch versierte Nutzer auf, sondern mit dem Einsatz der Systeme im öffentlichen Raum auch technische Laien und nicht unterwiesene Nutzer. Daraus erwachsen auch vollkommen neue Anforderungen an die Nutzerschnittstelle, beispielsweise intuitive Bedienbarkeit und Verständlichkeit, welche zwar durch die gesellschaftliche Weiterentwicklung (allgegenwärtige Kommunikationstechnik) erleichtert werden, aber dennoch eine nicht zu unterschätzende Hürde für die Akzeptanz von Robotern im Alltag darstellen.

Eine zentrale Fähigkeit für solche Systeme ist daher die Kommunikation mit

dem Menschen. So erfordert eine Zusammenarbeit ein ständiges Abgleichen von internen Zuständen und aktuellen Zielen beider Interaktionspartner, was eben nicht mehr auf technischer Ebene realisiert werden kann, sondern die dem Menschen zur Verfügung stehenden Kommunikationsmittel nutzen muss. So werden neben klassischen Ein- und Ausgabegeräten wie Bildschirm, Tastaturen und Zeigegeräten auch die verbale Kommunikation mittels Sprachsynthese und Spracherkennung sowie haptische Eingabemöglichkeiten genutzt. Der Roboter tritt demnach mit dem Menschen förmlich in einen Dialog (altgriech. dialégomai sich unterhalten).

Ergänzt werden die Kommunikationsfähigkeiten der Roboter durch eine breite Palette von Wahrnehmungsleistungen, welche, für den Nutzer eher unbewusst, zur Erfassung der situativen Rahmenbedingungen dienen, was im Dialog von entscheidender Bedeutung ist, wenn dieser rationell und intuitiv gestaltet sein soll.

Durch die Mobilität und eben die erforderlichen natürlichen Kommunikationsformen werden, auch abhängig vom Kulturkreis, Roboter nicht mehr allein als Werkzeug wahrgenommen, sondern als ein verkörpertes Individuum, dem unter Umständen auch menschliche Eigenschaften und Fähigkeiten zugeschrieben werden. Man spricht daher im Allgemeinen auch von **Social Robotics**, wobei in [Breazeal, 2003] eine genauere Abstufung dieser Erscheinung vorgenommen wird.

Durch diese Antropomorphisierung der Robotertechnik und die damit einhergehende Projektion eines gewissen Grades an Intelligenz besteht auch gegenüber dem Interaktionsverhalten des Systems eine hohe Erwartungshaltung. Im Rahmen dieser Arbeit wird eine Systemarchitektur vorgeschlagen, welche es ermöglicht, das Interaktionsverhalten eines mobilen Roboters in weiten Zügen flexibel und intelligent zu gestalten. Ein wesentlicher Aspekt dabei ist die Möglichkeit des Systems, sich an den Interaktionspartner in vielfältiger Weise anzupassen, wie dies auch in zwischenmenschlichen Interaktionssituationen der Fall ist. Dadurch soll die Nützlichkeit und Akzeptanz von robotischen Assistenzsystemen gesteigert werden. Weiterhin können somit auch Interaktionsziele des Roboters durch eine gesteigerte Kooperation des Nutzers besser erfüllt werden.

Die Entwicklung der vorgeschlagenen Architektur und der Verfahren zur Ge-

nerierung des Interaktionsverhaltens basiert auf einem mehrjährigen iterativen Prozess, der die Arbeit an mehreren Robotikprojekten mit recht unterschiedlichen Einsatzszenarien und Roboterplattformen umfasst. Im nächsten Abschnitt sollen daher diese Projekte kurz umrissen werden. Später dient das im jüngsten Projekt SERROGA realisierte System als konkretes Beispiel für die Umsetzung der Systemarchitektur. Die verschiedenen Nutzungsvarianten für Serviceroboter sollen durch die Anzahl der Nutzer und deren Interaktionsdauer charakterisiert werden. Somit steht dem Einsatz in der Öffentlichkeit mit vielen verschiedenen Nutzern, welche nur kurzzeitig (**Kurzeitinteraktion**), also wenige Minuten, in Kontakt zum System sind, der Einsatz im privaten häuslichen Bereich gegenüber, bei dem es nur einen Nutzer gibt, welcher lange Zeit (**Langzeitinteraktion**), evtl. einmal mehrere Jahre, mit dem Roboter zusammenlebt. Eine Zwischenposition nimmt ein Assistenzroboter ein, der mit verschiedenen Nutzern über mehrere Tage hinweg eine **wiederkehrende Interaktion** aufnimmt, wobei jeweils an den vergangenen Verlauf angeknüpft werden kann, sofern die Reidentifikation des Nutzers gewährleistet ist.

## 1.1 Einsatzszenarios

### 1.1.1 Einkaufsassistent Toomas

Als Beispiel für Kurzzeitinteraktionen mit verschiedenen Nutzern im öffentlichen Raum sei der Einkaufsassistent Toomas (siehe Abb. 1.1) genannt, welcher im Rahmen mehrerer durch die toom Baumarkt GmbH und durch den Freistaat Thüringen geförderter Projekte von 1999 bis 2006 bis zur Marktreife entwickelt wurde. Die Aufgabe eines solchen Roboters ist es, Kunden die Artikelstandorte im Baumarkt zu zeigen und sie dorthin zu geleiten [Gross et al., 2009, Gross et al., 2008].

Im Rahmen dieses Projektes gab es erste Überlegungen [Martin et al., 2005] zur Softwarestrukturierung und somit wurde die Grundlage der vorgestellten Systemarchitektur gelegt. Ebenfalls in dieser länger zurückliegenden Phase wurde bereits über eine Differenzierung der Inhalte der Roboterkommunikation anhand von Nutzergruppen nachgedacht. Beispielsweise könnten unter Berücksichtigung von Alter und Geschlecht der Kundschaft im Baumarkt

unterschiedliche Produkte beworben werden. Denkbar wäre auch, den Weg entlang interessengruppenspezifischer Artikelstandorte zu führen.

### 1.1.2 CompanionAble

Die Mitarbeit am durch das 7. Rahmenprogramm der Europäischen Union geförderten Projekt CompanionAble (Integrated Cognitive Assistive & Domotic Companion Robotic Systems for Ability & Security) ermöglichte es dem Autor, die anspruchsvollere Langzeitinteraktion mit einem Nutzer zu bearbeiten. Das CompanionAble Projekt (2008 bis 2012) hatte die Demonstration eines synergetischen Gesamtsystems aus einer Smarthome-Installation und einem mobilen Serviceroboter als Kommunikationsfrontend zum Ziel, welche es kognitiv beeinträchtigten älteren Menschen ermöglichen sollte, länger in ihrer eigenen Wohnung zu verbleiben [Gross et al., 2011b, Gross et al., 2012, Schröter et al., 2013]. Erreicht wurde dies durch Services wie Daytime Management, Kommunikation mit Pflegepersonal und Angehörigen, Notfallhelfer und kognitiver Therapie. Die spezielle Zielgruppe und die Komplexität der Anwendung machte es notwendig, die Ablaufsteuerung und Dialoggestaltung weiterzuentwickeln. Hierbei kamen Anforderungsaspekte wie Modularität und Erweiterbarkeit der eigentlichen Servicefunktionalität hinzu. Außerdem bekam die Middleware als Grundlage für eine über mehrere Geräte verteilte Gesamtanwendung einen hohen Stellenwert.
Die Roboterplattform Scitos-G3 [Merten et al., 2012] (ähnlich der Abb. 1.3), welche für dieses Projekt konstruiert wurde, kam später auch in Folgeprojekt SERROGA mit ähnlichen Rahmenbedingungen zum Einsatz und wurde dabei weiterentwickelt.

### 1.1.3 SERROGA

Die Forschergruppe SERROGA (Service-Robotik für die Gesundheitsassistenz) greift ab 2012 die Thematik der häuslichen Assistenzrobotik auf, konzentriert sich allerdings nicht mehr auf den Einsatz bei kognitiv beeinträchtigten älteren Menschen. Vielmehr wird ein Demonstrator für Techniken entwickelt, welche den Nutzer länger fit und zufrieden halten sollen. Der Roboter (Abb. 1.3) nimmt dabei mehrere Rollen ein, in denen er verschiedene Grade der An-

tropomorphisierung aufweisen sollte. Zum einen dient er als Notfallhelfer und Vermittler (Informationen und Videotelefonie), wobei er eher einen Werkzeugcharakter hat. Andererseits soll er als Mitbewohner, Bewegungsmotivator und Spielpartner eher in Form eines sozialen Wesens auftreten.
Auch in diesem Projekt steht eine Langzeitinteraktion mit einem Nutzer im Zentrum. Die konzeptuellen Ansätze zur Systemarchitektur im CompanionAble Projekt wurden für SERROGA großenteils übernommen und weiterentwickelt. Im Rahmen der Prototypenentwicklung konnte die Software noch einmal von Grund auf neu gestaltet werden und integriert sich nun wesentlich besser in die Konzepte und Infrastruktur der verwendeten Middleware MIRA[1].

### 1.1.4 ROREAS

In einem parallel laufenden Robotikprojekt des Fachgebietes NI&KR der TU Ilmenau kommt die dritte Kategorie der Nutzungsvarianten zum Einsatz. Das von 2012 bis 2015 durch das BMBF geförderte Verbundprojekt ROREAS (Interaktiver RObotischer REha-ASsistent für das Lauf- und Orientierungstraining von Patienten nach Schlaganfällen) hat die Entwicklung eines Roboters für das Gang- und Orientierungstraining von Schlaganfallpatienten zum Ziel (siehe Abb. 1.2). Hierbei nutzen diese Patienten den Roboter mehrmals pro Woche, um in der Klinik ohne Mitwirkung von Krankenhauspersonal ihr Eigentraining zu absolvieren. Dabei ist es wichtig, dass der Roboter an die jeweiligen Vorleistungen der letzten Trainingssession anknüpft und somit den Patienten nicht mit einem starren Trainingsprogramm überfordert oder gar demotiviert.
Dieser spezielle Einsatzzweck des Roboters und die Fragestellungen im Projekt gehen leider mit zusätzlichen Anforderungen einher, die es nicht sinnvoll erscheinen lassen, die bereits in den Vorgängerprojekten entwickelten Dialogmechanismen zu nutzen. So soll beispielsweise eine große Vielfalt an Interaktionsverläufen frei durch den Therapeuten oder Entwickler definiert werden können, wobei das Spektrum der Services, die der Roboter anbieten soll, sehr beschränkt und fest vorgegeben ist. Es geht somit eher um die manuelle Exploration eines erfolgversprechenden Interaktionsmusters durch die Entwickler,

[1] Eine kurze Charakterisierung von MIRA ist in Abschn. 4.3 zu finden.


**Abbildung 1.1:** Roboter Toomas, Einkaufsassistent im Toom Baumarkt

**Abbildung 1.2:** Roboter ROREAS, Schlaganfallrehabilitation und Orientierungstraining

als um die Realisierung eines selbst-adaptiven Langzeitbegleiters mit einem variablen frei kombinierbaren Servicespektrum. Dennoch findet sich die vorgeschlagene Systemarchitektur in weiten Zügen in der Umsetzung des ROREAS wieder. Lediglich die Dialogsteuerung in den obersten Schichten wurde entsprechend durch eine Zustandsmaschine ersetzt.

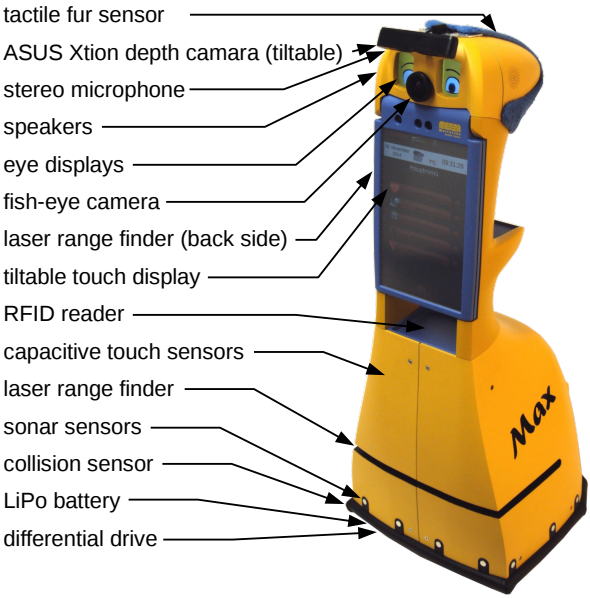



**Abbildung 1.3:** Roboter Max mit seinen Sensoren und Aktoren

## 1.2 Roboterplattform

Da der Hauptteil der in dieser Arbeit geschilderten Entwicklungen im Rahmen der Projekte SERROGA und CompanionAble stattfand, und später noch Beispiele aus diesem Bereich folgen, soll zunächst der bei diesen verwendete Roboter vorgestellt werden. Es handelt sich um einen Scitos-G3, der von der Firma MetraLabs GmbH im Rahmen der oben genannten Projekte entwickelt wurde. Diese Plattform zeichnet sich durch ihre für häusliche Anwendungen optimierte Größe und Manövrierfähigkeit aus. Mit 40 kg und 1,23 m Höhe ist der Roboter noch leicht zu handhaben und dennoch hinreichend stabil. Das Hauptinteraktionsmedium ist ein vertikal angebrachtes Touchdisplay, welches für die Nutzung im Stehen ausklappbar ist. Weitere Kommunikationsmittel sind ein angedeutetes Gesicht mit zwei TFT Displays als Augen, sowie Mikrofon und Lautsprecher. Zur Wahrnehmung seiner Umgebung ist der Roboter

zusätzlich mit einer Reihe von Sensoren ausgestattet. Diese umfassen einen SICK-Laser Scanner in einer horizontalen Ebene in 22 cm Höhe, ein Array aus 12 Ultraschallsensoren, eine Kinect-Tiefenkamera über dem Display, eine ASUS XtionPro (Tiefenkamera) an der Stirn, eine 180°-Fischaugenkamera in der Position der Nase, sowie einen schräg nach hinten gerichteten Laser Scanner, der unter dem Kopf angebracht ist. Zusätzlich sorgt ein rund um den Roboter verlaufender Gummipuffer für eine Notabschaltung der Antriebe bei versehentlicher Kollision mit einem Hindernis.
Für die Interaktion stehen weiterhin ein Streichelsensor am Kopf (siehe Abschn. 8.1.1), sowie kapazitive Touchsensoren unter der Plastikverkleidung (Abschn. 8.1.2) zur Verfügung. Um Gegenstände für den Nutzer aufbewahren zu können, besitzt der Roboter zwei Ablagefächer, welche mittels eines RFID-Lesegerätes überwacht werden. Der Roboter kann somit feststellen, ob und welche markierten Gegenstände im Fach liegen und dazu Auskunft erteilen.
Damit der Roboter autonom im Langzeiteinsatz verbleiben kann, verfügt er über eine Ladestation, die er mittels einer rückwärts gerichteten Kamera zum autonomen Andocken orten kann. Mit einer vollen Batterieladung kann der Roboter bis zu 8 h operieren.
Der Antrieb der Plattform besteht aus einem Differential Drive mit zwei angetriebenen Rädern mit 20 cm Durchmesser und einem Stützrad im hinteren Bereich. Dadurch steht der Roboter immer stabil und kann Schwellen bis 1cm Höhe überwinden.
Details zu den genutzten Verfahren für Lokalisation, Navigation und Bewegungssteuerung werden im Abschnitt 2.6 angeführt.

## 1.3 Inhaltliche Abgrenzung

Nachdem die Vielschichtigkeit von Robotikapplikationen deutlich wurde, soll dieser Abschnitt dazu dienen, die Inhalte und Beiträge dieser Arbeit herauszustellen und von den vielen beteiligten Modulen und Verfahren, die nicht vom Autor selbst bearbeitet wurden, abzugrenzen. Dies wird insbesondere notwendig, da mit der Schilderung der zugrundeliegenden Softwarearchitektur zwangsläufig auch diese Teile angesprochen werden, wobei auf die jeweilige weiterführende Literatur verwiesen wird.

Diese Arbeit will die grundlegende Struktur der Software begründen und konzentriert sich dabei auf die Ebene der Ablaufsteuerung der Dialogkontrolle und der eigentlichen Nutzanwendungen. Insbesondere stehen hier die Verfahren zur Dialogmodellierung und zur Anpassung des Verhaltens an den Nutzer im Zentrum. Dabei greifen die näher geschilderten Module auch auf technische Basisleistungen zu, die **nicht** Bestandteil dieser Arbeit sind. Insbesondere sind hier zu erwähnen: Lokalisation und Umgebungswahrnehmung (Mapping), autonome Navigation und Bewegungssteuerung, Personenwahrnehmung und -analyse sowie die reine Erfassung von Nutzerkommandos wie Kopfgesten oder Spracherkennung

## 1.4 Aufbau der Arbeit und zu erwartende Beiträge

Als Bogen über die einzelnen Beiträge der Arbeit dient das bereits genannte Szenario des Gesundheitsassistenten (SERROGA), wie es im Kapitel 2 genauer erläutert werden wird. In diesem zweiten Kapitel soll außerdem die Herangehensweise des Softwaredesigns geschildert werden, welche schrittweise von einer Beschreibung der Systemfunktion aus Nutzersicht zu einer Identifikation von Teilfunktionen und schließlich von robotischen Basisleistungen kommt. Diese hierarchische Verfeinerung wird sich auch in den Schichten der vorgeschlagenen Systemarchitektur wiederfinden. Mit dem Systemdesign wird auch die angestrebte Form der Interaktion mit dem Nutzer beschrieben. Als wesentlichen Aspekt wird dabei neben Multimodalität der Ein- und Ausgaben die Adaptivität des Systems gesehen, um den unterschiedlichen Anforderungen verschiedener Nutzer gerecht zu werden. Dazu wird in Kapitel 3 eine Analyse der möglichen Aspekte von Adaptivität und deren Möglichkeiten zur Realisierung erarbeitet werden.
Im Anschluss daran wird unter Berücksichtigung der Anforderungen zum Interaktionsdesign und Adaptivität in Kapitel 4 eine mehrschichtige Systemarchitektur vorgestellt und in den State of the Art eingeordnet werden. Diese kann den Rahmen für die Realisierung verschiedener modularer Servicerobotikanwendungen bilden und ist somit auch übertragbar auf andere Szenarien, beispielsweise die bereits genannten.
Im weiteren Verlauf der Arbeit wird schrittweise in Ausschnitte der Systemar-

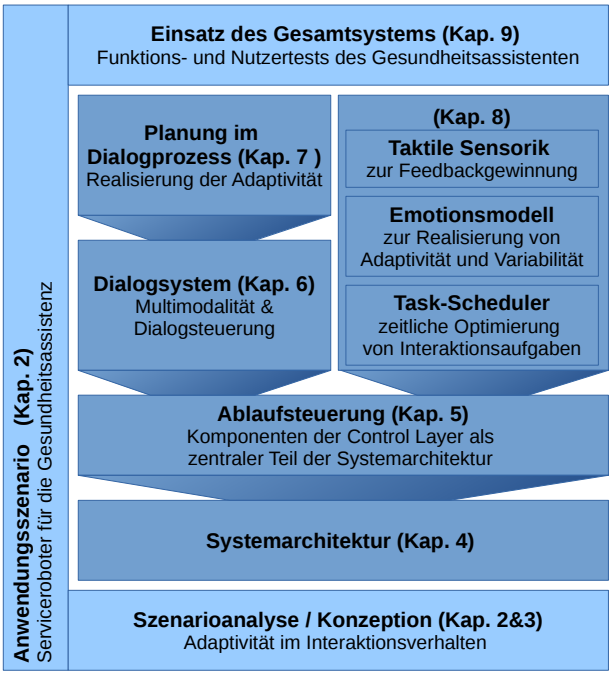



**Abbildung 1.4:** Aufbau der Arbeit

chitektur fokussiert. Abb. 1.4 zeigt dieses Vorgehen in der mittleren Spalte. Demzufolge wird Kapitel 5 zunächst einen Überblick zu den Modulen der Control Layer geben und es wird versucht, einen Überblick zu deren Zusammen- und Wechselwirken zu vermitteln. Die Control Layer ist das Herzstück der Architektur und nicht spezifisch für die Anwendung als Gesundheitsassistent.

Der zentrale Baustein der Architektur, das Dialogsystem, wird im Kapitel 6 beschrieben. Dabei soll zunächst der State of the Art zur Realisierung von Dialogsystemen dargestellt werden, um anschließend eine modulare Realisierung eines multimodalen Dialogsystems zu entwickeln, welche den Aspekten der Adaptivität gerecht wird. Das Konzept wird es dabei ermöglichen, entweder deterministisch zu agieren, wie für die Umsetzung des Beispielszenarios genutzt, oder die Dialogverläufe mittels probabilistischer Planung zu optimieren, wie es im Kapitel 3 vorgeschlagen werden wird.

Die im Dialogmanager integrierte probabilistische Aktionsplanung wird später im Kapitel 7 weiter vertieft, wobei der dafür entwickelte Modellierungsansatz und der eigens entworfene Inferenzalgorithmus am Beispiel eingeführt werden. Über den State of the Art hinaus geht dabei die Modellierung der probabilistischen Modelle mittels sample-basierter Darstellung hochdimensionaler Zustandsräume und den damit einhergehenden Operationen im Inferenzalgorithmus. Kapitel 7 schildert außerdem die konkrete Anwendung der Planung für den Dialogprozess an dem experimentellen Anwendungsszenario eines Büroassistenten. Da die planende Variante der Dialogsteuerung für die Umsetzung des Gesundheitsassistenten nicht zum Einsatz kam, wurde anhand dieses eigenständigen Versuchsszenarios die probabilistische Planung erprobt und deren korrekte Funktion nachgewiesen.
Neben der Dialogsteuerung werden im Kapitel 8 anschließend weitere Komponenten der Control Layer vorgestellt, welche das Gesamtsystem komplettieren. Hierbei wird ein Emotionsmodell für die Anpassung der Aktivitätszyklen an den Nutzungsrhythmus sowie die Grundidee eines Taskschedulers für die Optimierung der Ausführungszeitpunkte für Interaktionsaufgaben beschrieben werden. Außerdem geht Kapitel 8 auf die entwickelte taktile Sensorik ein, welche als Medium für die Erfassung von Nutzerfeedback dienen kann, was wiederum für die Optimierung eine wichtige Rolle spielt.
Den Abschluss der Arbeit bildet die Schilderung von realen Nutzertests im Kapitel 9, welche mit dem SERROGA Demonstrator durchgeführt wurden. Diese sollen zeigen, dass das vorgeschlagene System praxistauglich und funktionsfähig ist und schließen somit den Bogen zum geschilderten Entwurfsprozess.

## 1.5 Publikationen

Teile der in dieser Arbeit geschilderten Ideen wurden bereits auf internationalen Konferenzen publiziert.

[Müller et al., 2014b] Müller, St., Sprenger, S., Gross, H.-M.
**Online Adaptation of Dialog Strategies based on Probabilistic Planning**. in: Proc. 23rd IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN 2014), Edinburgh, UK, pp. 692-697, IEEE 2014

In diesem Beitrag wird die Idee der online Optimierung der Dialogabläufe mittels probabilistischer Planung beschrieben. Auch die samplebasierte Repräsentation von hochdimensionalen Wahrscheinlichkeitsverteilungen wird dabei vorgestellt.

[Müller et al., 2014a] Müller, St., Sprenger, S., Gross, H.-M.

**OfficeMate - a Study of an Online Learning Dialog System for Mobile Assistive Robots.** in: Proc. Sixth Int. Conference on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE 2014), Venice, Italy, pp. 104-110, 2014

Diese Veröffentlichung greift die Thematik der vorhergehenden auf und konzentriert sich auf die Beschreibung der Optimierung anhand des auch in Kap. 7 vorgestellten Versuchsszenarios eines Assistenzroboters in der Büroumgebung. Dieser Artikel erhielt den Best Paper Award der ADAPTIVE 2014.

[Müller et al., 2013] Müller, St., Schröter, Ch., Gross, H.-M.

**Low-Cost Whole-Body Touch Interaction for Manual Motion Control of a Mobile Service Robot.** in: Proc. Int. Conf. on Social Robotics (ICSR), ser. LNAI, vol. 8239, pp. 229–238, Springer, 2013

Dieses Paper stellt die kapazitive taktile Sensorik vor (vergleiche Kap. 8), welche unter den Gehäuseteilen des Roboters verbaut ist und durch eine Ansteuerung der Motoren bei Berührung ein assistiertes Schieben ermöglicht.

[Müller et al., 2015] Müller, St., Schröter, Ch., Gross, H.-M.

**Smart Fur Tactile Sensor for a Socially Assistive Mobile Robot.** in: Proc. Int. Conf. on Intelligent Robotics and Applications (ICIRA), UK, LNCS 9245, pp. 49-60, Springer 2015

Die zweite Form der taktilen Sensorik, ein Streichelsensor, wird in dieser Veröffentlichung zusammen mit der Methode zur Klassifikation der Berührungsmuster vorgestellt. In der vorliegenden Arbeit ist dieser Sensor in Kap. 8 beschrieben.

**Veröffentlichungen in Co-Autorschaft mit Bezug zu den Szenarien:**

[Böhme et al., 2006] Böhme, H.-J., Scheidig, A., Wilhelm, T., Schröter, Chr., Martin, Ch., König, A., Müller, St., Gross, H.-M.
**Progress in the Development of an Interactive Shopping-Assistant.** in: Proc. of the Joint Conf. on Robotics: International Symposium on Robotics (ISR) and German Conference on Robotics (Robotik), Munich, Germany, paper nr. 83, 20 pages, VDI, 2006

[Gross et al., 2008] Gross, H.-M., Böhme, H.-J., Schröter, Ch., Müller, St., König, A., Martin, Ch., Merten, M., Bley, A.
**ShopBot: Progress in Developing an Interactive Mobile Shopping Assistant for Everyday Use.** in: Proc. IEEE Int. Conf. on Systems, Man and Cybernetics (SMC), Singapore, pp. 3471-3478, IEEE 2008

[Gross et al., 2009] Gross, H.-M., Böhme, H.-J., Schröter, Ch., Müller, St., König, A., Einhorn, E., Martin, Ch., Merten, M., Bley, A.
**TOOMAS: Interactive Shopping Guide Robots in Everyday Use - Final Implementation and Experiences from Long-Term Field Trials.** in: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), St. Louis, USA, pp. 2005-2012, IEEE 2009

Diese Veröffentlichungen beschreiben die Entwicklung des Einkaufsassistenten TOOMAS (siehe Abschn. 1.1.1). Dabei werden auch erste Ansätze einer mehrschichtigen Softwarearchitektur vorgestellt, auf der die in dieser Arbeit entwickelte Architektur basiert. Eigene Beiträge dieser Veröffentlichungen beschränken sich auf das Personentracking und die probabilistische Modellierung der Nutzereigenschaften.

[Gross et al., 2011b] Gross, H.-M., Schröter, Ch., Müller, St., Volkhardt, M., Einhorn, E., Bley, A., Martin, Ch., Langner, T., Merten, M.
**Progress in Developing a Socially Assistive Mobile Home Robot Companion for the Elderly with Mild Cognitive Impairment.** in: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), San Francisco, USA, pp. 2430-2437, IEEE Omnipress 2011

[Gross et al., 2011a] Gross, H.-M., Schröter, Ch., Müller, St., Volkhardt, M., Einhorn, E., Bley, A., Martin, Ch., Langner, T., Merten, M.
**I'll Keep an Eye on You: Home Robot Companion for Elderly People with Cognitive Impairment.** in: IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), Anchorage, USA, pp. 2481-2488, IEEE 2011

[Gross et al., 2012] Gross, H.-M., Schröter, Ch., Müller, St., Volkhardt, M., Einhorn, E., Bley, A., Langner, T., Merten, M., Huijnen, C., v. d. Heuvel, H., v. Berlo, A. **Further Progress towards a Home Robot Companion for People with Mild Cognitive Impairment.** in: Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), Seoul, South Korea, pp. 637-644, IEEE 2012

[Schröter et al., 2013] Schröter, Ch., Müller, St., Volkhardt, M., Einhorn, E., Huijnen, C., van den Heuvel, H., van Berlo, A. , Bley, A., Gross, H.-M. **Realization and User Evaluation of a Companion Robot for People with Mild Cognitive Impairments.** in: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), Karlsruhe, Germany, pp. 1145-1151, IEEE 2013

Diese vier Publikationen stellen das CompanionAble Projekt vor (siehe Abschn. 1.1.2). Die eigenen Beiträge dabei umfassen Methoden zur Personendetektion anhand von online gelernten Farbmodellen, sowie die prinzipiellen Architekturmerkmale und die Komponenten der Dialogsteuerung, welche sich auch in der vorliegenden Fassung wiederfinden.

[Stricker et al., 2012b] Stricker, R., Müller, St., Einhorn, E., Schröter, C., Volkhardt, M., Debes, K., Gross, H.-M.
**Konrad and Suse, Two Robots Guiding Visitors in a University Building.** in: Autonomous Mobile Systems 2012 (AMS), Stuttgart, Germany, Informatik aktuell, pp. 49-58, Springer 2012

[Stricker et al., 2012a] Stricker, R., Müller, St., Einhorn, E., Schröter, C., Volkhardt, M., Debes, K., Gross, H.-M.
**Interactive Mobile Robots Guiding Visitors in a University Building.** in: Proc. IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN), Paris, France, pp. 695-700, IEEE 2012

Ein weiteres bislang ungenanntes Einsatzszenario der vorgestellten Architektur wird in den beiden letztgenannten Veröffentlichungen vorgestellt. Es

handelt sich um einen Lotsenroboter für das Fakultätsgebäude. Die entwickelte Architektur kommt prinzipiell auch hierbei zum Einsatz, wurde aber um eine Administrative Ebene erweitert, um eine web-basierte Fernwartung und Steuerung der Lotsenroboter zu ermöglichen. Der Dialogmanager, wie er in dieser Arbeit geschildert wird, kommt dabei allerdings nicht zum Einsatz, weshalb dieses Szenario hier nicht weiter vertieft wird.

[Gross et al., 2014] Gross, H.-M., Debes, K., Einhorn, E., Müller, St., Scheidig, A., Weinrich, Ch., Bley, A., Martin, Ch.
**Mobile Robotic Rehabilitation Assistant for Walking and Orientation Training of Stroke Patients: A Report on Work in Progress.** in: Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC 2014), San Diego, USA, pp. 1880-1887, IEEE 2014

Dieses Paper stellt die Fortschritte bei der Entwicklung des Schlaganfallrehabilitationsassistenten vor (siehe Abschn. 1.1.4). Die eigenen Beiträge dazu beschränken sich auf die Beschreibung der Systemarchitektur mit einer Abwandlung des Dialogsystems, welches dabei als Zustandsautomat umgesetzt wurde.

[Gross et al., 2015] Gross, H.-M., Müller, St., Schröter, Ch., Volkhardt, M., Scheidig, A., Debes, K., Richter, K., Döring, N.
**Robot Companion for Domestic Health Assistance: Implementation, Test and Case Study under Everyday Conditions in Private Apartments.** in: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Hamburg, Germany, pp. 5992-5999, 2015

Diese Publikation schildert die Ergebnisse der Forschergruppe SERROGA (siehe Abschn. 1.1.3). Der dabei beschriebene Gesundheitsassistent dient auch dieser Arbeit als Anwendungsbeispiel der eingeführten Methodiken. Neben der Darstellung der Softwarearchitektur umfassen die Beiträge dieser Publikation die Durchführung und Beschreibung der Funktions- und Nutzertests, sowie eine objektive Beschreibung der Einsatzumgebungen mittels verschiedener Komplexitätsmaße. Kapitel 9 stellt die Inhalte dieses Papers kurz da.

**Veröffentlichungen die nicht zum Kernthema dieser Dissertation gehören:**

[Müller et al., 2005] Müller, St., König, A., Gross, H.-M. **Neural Architecture for Concurrent Map Building and Localization using Adaptive Appearance Maps.** in: Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005 (ICANN), LNCS 3697, Part II, pp. 929-934, Springer 2005

[Müller et al., 2007] Müller, St., Schaffernicht, E., Scheidig, A., Böhme, H.-J., Gross, H.-M. **Are You Still Following Me?** in: Proc. European Conference on Mobile Robots (ECMR), Freiburg, pp. 211-216, Albert-Ludwigs-Universität Freiburg - Universitätsverlag 2007

[Müller et al., 2008] Müller, St., Hellbach, S., Schaffernicht, E., Ober, A., Scheidig, A., Gross, H.-M. **Whom to Talk to? Estimating User Interest from Movement Trajectories.** in: Proc. of the IEEE Int. Symposium on Robot and Human Interactive Communication, (RO-MAN), Munich, Germany, pp. 532-538, IEEE, 2008

Davon als Co-Autor:

[Gross et al., 2005] Gross, H.-M., König, A., Müller, St. **Omniview-Based Concurrent Map Building and Localization using Adaptive Appearance Maps.** in: Proc. IEEE Internat. Conf. on Systems, Man and Cybernetics (SMC), Hawaii, pp. 3510-3515, IEEE 2005

[König et al., 2005] König, A., Müller, St., Gross, H.-M. **Appearance-Based CML Approach for a Home Store Environment.** in: Proc. European Conference on Mobile Robots (ECMR), Ancona, Italy, pp. 206-211, stampalibri 2005 (ISBN 88-89177-187)

[Gross et al., 2006] Gross, H.-M., Richarz, J., Müller, St., Scheidig, A., Martin, Ch. **Probabilistic Multi-modal People Tracker and Monocular Pointing Pose Estimator for Visual Instruction of Mobile Robot Assistants.** in: Proc. IEEE World Congress on Computational Intelligence (WCCI), International Joint Conference on Neural Networks (IJCNN), Vancouver, Canada, pp. 8325-8333, IEEE 2006

[Scheidig et al., 2006] Scheidig, A., Müller, St., Martin, Ch., Gross, H.-M. **Generating Person's Movement Trajectories on a Mobile Robot.** in: Proc. IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN), Hatfield (UK), pp. 747-752, IEEE 2006

[Schröter et al., 2009] Schröter, Ch., Höchemer, M., Müller, St., Gross, H.-M. **Autonomous Robot Cameraman - Observation Pose Optimization for a Mobile Service Robot in Indoor Living Space.** in: Proc. IEEE Int. Conf. on Robotics and Automation (ICRA), Kobe, Japan, pp. 424-429, IEEE 2009

[Volkhardt et al., 2009] Volkhardt, M., Kalesse, S., Müller, St., Gross, H.-M. **Maximum a Posteriori Estimation of Dynamically Changing Distributions.** in: Proc. Annual Conference on Artificial Intelligence (KI), Paderborn, LNCS 5803, pp. 484–491, Springer Verlag 2009

[Volkhardt et al., 2011a] Volkhardt, M., Müller, St., Schröter, Ch., Gross, H.-M. **Detection of Lounging People with a Mobile Robot Companion.** in: Proc. Int. Conf. on Intelligent Robotics and Applications (ICIRA), Aachen, Germany, LNAI 7102, Part II, pp. 328-337, Springer 2011

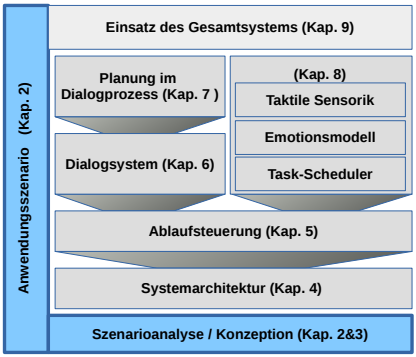
[Volkhardt et al., 2011b] Volkhardt, M., Müller, St., Schröter, Ch., Gross, H.-M. **Playing Hide and Seek with a Mobile Companion Robot.** in: Proc. IEEE-RAS Int. Conf. on Humanoid Robots (HUMANOIDS), Bled, Slovenia, pp. 40-46

[Stricker et al., 2014] Stricker, R., Müller, St., Gross, H.-M. **Non-contact Video-based Pulse Rate Measurement on a Mobile Service Robot.** in: Proc. 23rd IEEE Int. Symposium on Robot and Human Interactive Communication (RO-MAN 2014), Edinburgh, UK, pp. 1056-1062, IEEE 2014

[Stricker et al., 2015b] Stricker, R., Müller, St., Gross, H.-M. **Universal Usage of a Video Projector on a Mobile Guide Robot.** in: Proc. Int. Conf. on Intelligent Robotics and Applications (ICIRA), UK, LNCS 9246, part. III, pp. 25-36, Springer 2015

[Stricker et al., 2015a] Stricker, R., Müller, St., Gross, H.-M. **R2D2 Reloaded: Dynamic Video Projection on a Mobile Service Robot.** in: Proc. Europ. Conf. on Mobile Robotics (ECMR), Lincoln, UK, 6 pages, 2015

# 2




# Anwendungsszenario

Da die finale Version der in dieser Arbeit vorgestellten Konzepte hauptsächlich für die Umsetzung der Demonstratoren im Projekt SERROGA genutzt wurden, sollen in diesem Kapitel die Anwendungsszenarien dieses Projekts näher erläutert werden.

Hierbei wird anhand Abb. 2.1 die Herangehensweise beim Systementwurf verdeutlicht. Die Spezifikation geht dabei hierarchisch anhand von äußerlich beobachtbaren Verhaltensweisen des Roboters vor. Zunächst sollen die Einsatzbedingungen definiert werden, woran sich die konzeptionelle Sicht auf den Serviceroboter anschließt. Nachdem dabei verschiedene **Rollen** und Einsatzbereiche für den Roboter identifiziert wurden, können entsprechende **Services** beschrieben werden, wie sie vom Nutzer wahrgenommen werden sollen. Dabei wird sich herausstellen, dass gewisse Verhaltensweisen immer wieder vorkommen (**Verhaltensmuster**) und diese die Bausteine für eine modulare Systemrealisierung darstellen können. Im Anschluss daran lassen sich **technische Basisleistungen** identifizieren, welche diese Verhaltensweisen ermöglichen. Die Basisleistungen setzen dabei sehr nah auf der **Hardware** mit ihren Sensoren und Aktoren auf, deren Spezifikation sich somit indirekt aus den Anforderungen der erwünschten Services ergibt.

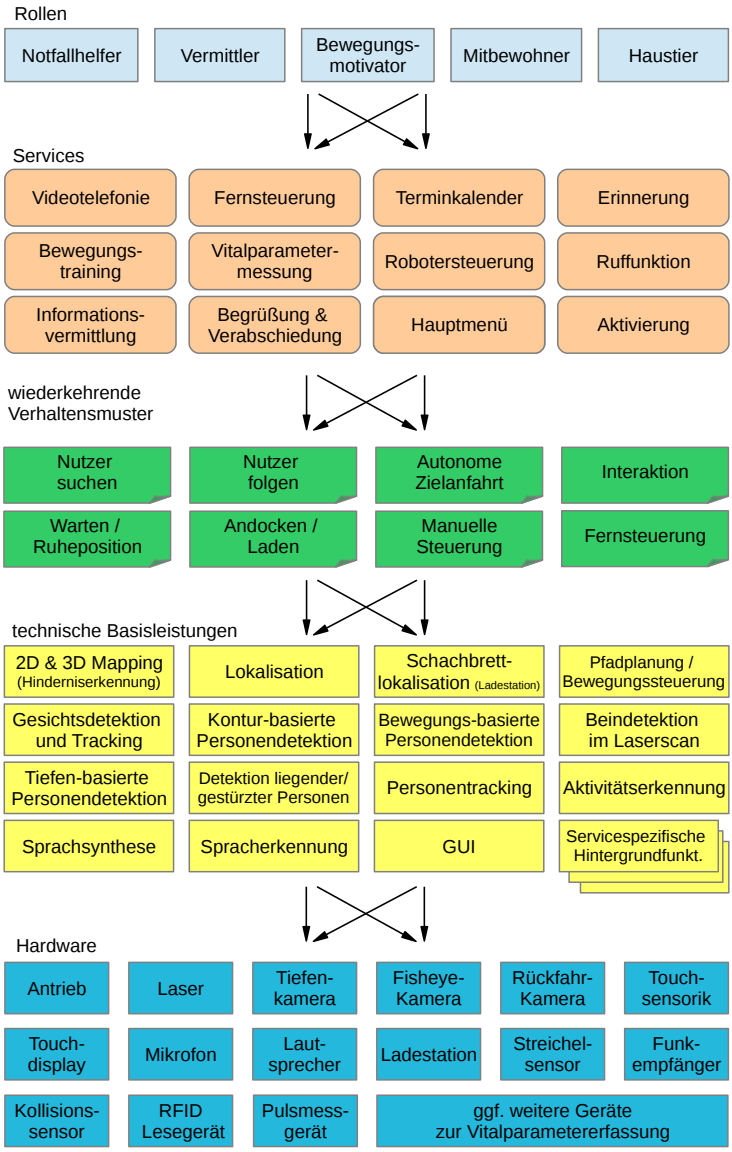



**Abbildung 2.1:** Übersicht zur Systemanalyse für den SERROGA Roboter

## 2.1 Rahmenbedingungen für einen Gesundheitsassistenzroboter

Die Idee eines Gesundheitsassistenten zielt im Wesentlichen auf die aufgrund der Bevölkerungsentwicklung immer größer werden Gruppe älterer Menschen ab, die durch Assistenzleistungen länger selbstbestimmt in ihrer eigenen Wohnung leben können. Damit steht als Einsatzfeld für einen Assistenzroboter die häusliche Umgebung fest, wobei möglichst wenig bis gar keine Anpassung der Umgebung für den Einsatz des Roboters erforderlich seien sollte. Das Robotersystem muss somit einen fortgeschrittenen Grad an Autonomie und Sicherheit für Umgebung und Benutzer aufweisen. Das motiviert zum gegenwärtigen Stand der Technik auch eine Einschränkung auf nicht manipulative Robotik also **soziale Assistenzrobotik**. Die Zusammenarbeit von Menschen und Manipulationsrobotern ist ein eigenes Forschungsfeld und wird daher von vornherein ausgeklammert.

## 2.2 Konzeptionelle Sicht (Rollen)

Im Rahmen der theoretischen Betrachtung einer Servicerobotik für Gesundheitsassistenz im Rahmen des SERROGA Projektes wurden verschiedene Möglichkeiten der Erscheinung eines robotischen Assistenzsystems erkannt, welche von einem unpersönlichen "Werkzeug" bis zu einem sozialen "Begleiter" reichen. Die möglichen Anwendungsgebiete eines Assistenzroboters ordnen sich daher entlang einer Achse von "werkzeughaft" bis "sozial" ein und sollen **Rollen** genannt werden, die mit jeweils einer eigenen Form der Erscheinung/Interaktion einhergehen. (siehe Abb. 2.2)

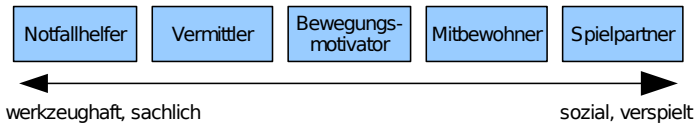



**Abbildung 2.2:** Rollen eines Assistenzroboters mit ihren zugehörigen Erscheinungsformen des Roboters

Ein Aufgabenfeld, welches durch das höhere Risiko für Sturz, Herzinfarkt

oder Schlaganfälle im Alter motiviert ist, stellt die Rolle des Notfallhelfers dar. Ein Assistenzroboter wäre prinzipiell als Erster vor Ort und könnte Hilfe rufen oder gar eine Telepräsenz der menschlichen Helfer ermöglichen. Der Roboter sollte hierbei vertrauenswürdig, vollkommen sachlich und vorhersehbar agieren.
Die Telepräsenz spielt bereits in die zweite Rolle hinein, welche die Aufrechterhaltung sozialer Kontakte zu Familie und Freunden zum Inhalt hat. Der Roboter könnte als Videotelefonieplatform fungieren oder wiederum als Avatar die Telepräsenz in der Wohnung der Senioren ermöglichen. Dabei ist es nicht zwingend notwendig, einen sachlichen Umgang mit dem Nutzer zu führen, dennoch sollte das Verhalten des Roboters deterministisch und vorhersehbar sein.
Eine dritte Rolle stellt die eines Bewegungsmotivators dar. Ältere Menschen können durch physisches Training ihre Mobilität erhalten und dies trägt zu Vorbeugung von Stürzen bei. Ein motivierender Begleiter oder Trainingspartner kann dabei helfen, die selbst gesteckten Ziele der Senioren für ein Bewegungsprogramm umzusetzen. Hierbei kommt der soziale Aspekt der Robotererscheinungsform zum Tragen. Eine Motivation geht weit über das bloße sachliche Erinnern an Trainingstermine hinaus und profitiert von der Verkörperung eines Trainingsbegleiters. Diese trägt letztendlich dazu bei, dass Hinweise des Systems ernster genommen werden und sich der Nutzer in gewisser Weise zum System sozial positioniert. Das heißt, dass er sich z.B. keine Blöße geben will oder seine Erwartungen an sich selbst auf das System als Kontrolleur projiziert [Geue, 2012][1].
Auch die Mitbewohner-Rolle erfordert eher soziale Formen der Interaktion. Hierbei steht das proaktive Kommunizieren des Roboters im Mittelpunkt. Einsamkeit und Depressionen sollten durch die Anwesenheit des Roboters gemindert werden. Auch Aufgaben eines Sekretärs (Terminverwaltung, E-mail-Korrespondenz, Erinnerungen, Informationsversorgung) können in der Rolle des Mitbewohners umgesetzt werden.
Die soziale Bindung zum Roboter wird noch weiter forciert in der Rolle eines Haustiers oder Spielgefährten. In dieser Sichtweise kann auch eine gegenseitige Abhängigkeit von Roboter und Nutzer plausibel untergebracht werden. Der

[1] Diese Arbeit wurde vom Autor betreut.

Roboter kann in Situationen geraten, in denen er Hilfe vom Nutzer braucht - hier profitiert er von einer emotionalen Bindung, die der Nutzer zu ihm aufgebaut hat. Adressiert wird diese gegenseitige Abhängigkeit ebenfalls im EU FP7 Projekt Hobbit [Bajones et al., 2015]. Der Nutzer akzeptiert solche Schwächen des Roboters unter Umständen besser, ohne den Nutzen des Systems in Frage zu stellen. Der Roboter kann in dieser Rolle weiterhin proaktiv den Kontakt zum Nutzer suchen, um diesen scheinbar grundlos zu beobachten[2], und bringt auch einen gewissen Unterhaltungswert durch Spiele mit, in denen er seine Körperlichkeit zum Einsatz bringt.

## 2.3 Services

Die Rollen des Assistenzroboters lassen sich weiter konkretisieren, indem einzelne Services identifiziert und in ihrer Erscheinung für den Nutzer beschrieben werden. Dies entspricht in etwa einer Use-Case Analyse.
Im Rahmen der Analyse des Gesundheitsassistenten wurden eine Vielzahl von Services vorgeschlagen, die ein sozialer Assistenzroboter für einen Menschen erbringen könnte. Aufgrund der begrenzten Ressourcen für die Entwicklung wurden im Rahmen der Forschergruppe SERROGA folgende Services für eine Implementierung ausgewählt, wobei weiterführende verbale Beschreibungen der Servicefunktionalitäten im Anhang A 2.3 zu finden sind.

- **Videotelefonie** zur Kommunikation mit Angehörigen, Freunden und Pflegepersonal
- **Fernsteuerung**, um den Roboter von außerhalb der Wohnung mittels eines Mobilgerätes zu steuern
- **Terminkalender** zum Verwalten und Teilen von Kalendern sowie zum Editieren von Terminen
- **Erinnerungen** an Termine im Kalender werden aktiv ausgeliefert
- **Aktivierung** durch Vorschläge für Aktivitäten und ToDos
- **Bewegungstraining** mit regelmäßigen Sessions und Langzeitzielen, welche individuell editiert und abgearbeitet werden können

[2] Zum Erfüllen der Notfallhelferfunktion und zur Ermittlung der Nutzeraktivitäten zwecks situationsgerechter Aktivierung von Interaktionstasks (siehe Abschn. 5.5) wäre es wünschenswert, den Nutzer permanent zu beobachten.

- **Vitalparametermessung** am Beispiel eines Pulsoximeters und einer bild-basierten Pulsmessmethode
- **Robotersteuerung** zur manuellen Aktivierung aller Roboternavigationsfunktionen
- **Ruffunktion** mittels eines Handfunksenders, um den Roboter zum Nutzer zu bringen
- **Wetterinfo** als Beispiel für weitere web-basierte Informationsservices
- **Begrüßung und Verabschiedung** zur Kommunikation der Anwesenheit des Nutzers und um dem Nutzer eine Privatsphäre zu ermöglichen, indem er den Roboter in einen Ruhemodus versetzt

#### Grundfunktionen als Services

Neben den eigentlichen Nutzservices werden für einen runden Ablauf noch weitere Grundfunktionen benötigt, welche ebenfalls als eigenständige Services realisiert sind. Hierbei ist das **Hauptmenü** von zentraler Bedeutung, welches den Zugriff auf die Services über die touchscreen-basierte Interaktion ermöglicht. Eine baumartige Strukturierung ermöglicht es, den Bildschirminhalt übersichtlich zu gestalten, geht aber durch die nur ausschnitthafte Sicht auf die volle Menge an Funktionen mit dem Nachteil einher, dass der Nutzer die Menüstruktur und die Gesamtheit der Funktionen kennen muss. Eine systemgetriebene Einführung in die Funktionen des Menüs sollte dem Nutzer diese vermitteln. (ähnlich wie in Abschn. 7.1 umgesetzt)
Weitere sinnvolle kleine Dialoge sollen den Nutzer informieren, wenn der Roboter autonom agieren will oder muss. So sollte nach längerer Untätigkeit an einer Position der Nutzer gefragt werden, ob die Dienste des Roboters noch benötigt werden und er demzufolge noch bleiben soll, oder ob er sich wieder zu seiner Ruheposition zurückziehen kann. (**Nutzerfeedback**, **Idle-Dialog**)
Ebenso sollte der Nutzer informiert werden, wenn die Interaktion abgebrochen werden muss, weil der Roboter auf seine Ladestation fahren muss, um seine Akkus aufzuladen.
Um seiner Rolle als „Haustier“ gerecht zu werden, und somit die emotionale Bindung zu stärken, sollte der Roboter auch von sich aus in Zeiten längerer Interaktionspausen die Wohnung explorieren und Kontakt mit dem Nutzer su-

chen. Hierbei kann das System auch den Status und die Aktivitäten des Nutzers observieren und somit seiner Notfallhelferrolle gerecht werden. Ebenso soll das Wissen über die Nutzeraktivitäten einer situationsangepassten Auslieferung von Erinnerungen und proaktiven Serviceerbringung dienen (siehe Abschn. 8.3). (**Emotionalisierungs-** und **Erkundungsservice**)

## 2.4 Interaktionsformen

Bevor die Basisverhaltensmuster aus den zu realisierenden Services abgeleitet werden, soll zunächst einmal ein generelles Bild von der angestrebten Interaktion zwischen Roboter und seinem Nutzer geschildert werden.
Die Interaktion mit dem SERROGA Roboter soll möglichst multimodal über ein grafisches Interface (GUI), aber auch über Kopfgesten und Spracheingaben geschehen. Dabei sollte der Roboter schrittweise verbal und mittels Text- und Sprachausgaben durch die Dialoge führen und die jeweiligen Antwortmöglichkeiten auf dem Display anbieten. Damit der Nutzer auch selbst die Initiative ergreifen kann, sollte über das Display jederzeit Zugriff auf das Auswahlmenü und somit auf alle Servicefunktionen möglich sein. Insbesondere die Robotersteuerung, und eine Stopp-Funktion (falls der Roboter autonom fährt) sollten prominent auf dem Display platziert werden. Durch die beiderseitige Möglichkeit, jederzeit neue Aufgaben in den Interaktionsfluss einzubringen, muss es möglich sein, laufende Services zu unterbrechen und diese nach der Unterbrechung fortzusetzen.
Der Roboter sollte auch in der Lage sein, die Position des Nutzers in Relation zu seiner eigenen Position wahrzunehmen. Dadurch kann er sich geeignet zum Interaktionspartner positionieren und die Interaktionsbereitschaft des Gegenübers abschätzen.
Da alle Beobachtungen des Roboters und somit auch Nutzereingaben immer zu einem gewissen Grad mit Unsicherheit behaftet sind, soll diese auch im Dialogmodell berücksichtigt werden. Insbesondere soll es möglich sein, auf unsichere Informationen mit Nachfragen zu reagieren oder diese implizit durch den Nutzer im weiteren Dialogverlauf bestätigen zu lassen.
Trotz einer Vielzahl von Rollen und Services soll der Roboter als ein Gesprächspartner erscheinen und nicht als Trägerplattform, auf der vielfältige

Services laufen. Ein Handy oder Tablet-PC würden eine solche Personifikation als soziales Individuum nicht leisten können.

## 2.5 Wiederkehrende Verhaltensmuster

Eine Betrachtung der zu realisierenden Funktionalitäten in den Services macht deutlich, dass sich das finale Interaktionsverhalten aus einer Mischung von verschiedenen wiederkehrenden Mustern sowie funktionsspezifischen Teilen zusammensetzt. Die wiederkehrenden Basisverhaltensweisen beziehen sich meist auf die Navigationsmöglichkeiten des Roboters. In der Umsetzung ist es wichtig, diese zu identifizieren und in der Softwarearchitektur geeignet abzubilden. Teilweise sind sie als eigenständige Softwaremodule implementiert, und teils bestehen sie lediglich aus einer geeigneten Ansteuerung des Navigators[3]. Folgende Verhaltensweisen konnten identifiziert werden:

- **Nutzersuche** zur Herstellung von Interaktionsbereitschaft. Die konkrete Umsetzung ist in [Volkhardt und Gross, 2013a, Volkhardt und Gross, 2013b] beschrieben und basiert auf einer Schätzung der Aufenthaltswahrscheinlichkeit und Optimierung der Suchfahrt anhand des zu erwartenden Informationsgewinns.
- **Folgen** zum Erhalten der Interaktionsbereitschaft, wenn der Nutzer sich in der Wohnung bewegt. Das Folgen wird über ein dynamisches Neuplanen des Bewegungspfades zur jeweils letzten bekannten Nutzerposition realisiert.
- **Autonome Zielanfahrt**, um den Roboter zu bestimmten Positionen zu schicken. Basiert vollständig auf der Navigationsbasisleistung.
- **Interaktion mit dem Nutzer**; Während der Interaktion muss der Roboter still halten. Ein alternatives Nachführen der Roboterorientierung zum Nutzer hat sich teilweise als störend herausgestellt.
- **Autonome Rückkehr zur Ruheposition** nach Beendigung einer Interaktion, um aus dem Weg zu gehen.
- **Andocken** an die Ladestation, um bei niedrigem Energievorrat die Akkus aufzuladen.

[3] Abschnitt 4.2.2 geht auf die Besonderheiten des genutzten Navigationsmoduls ein.

- **Manuelle Steuerung** zur lokalen Positionierung des Roboters, falls die autonome Ausrichtung zum Nutzer nicht die gewünschte Position erreicht. Ein Schieben des Roboters wird dabei aktiv durch die Motoren des Roboters unterstützt (siehe Abschn. 8.1.2).
- **Fernsteuerung** von außerhalb der Wohnung mit lokaler Hindernisvermeidung [Ebert, 2012][4].

Im Anhang A 2.5 finden sich für den interessierten Leser genauere Beschreibungen der Basisverhaltensweisen, die ein Bild vom realisierten Roboter vermitteln.

## 2.6 Technische Basisleistungen

Um die oben aufgeführten Verhaltensweisen zu realisieren und das gewünschte Interaktionsverhalten zu erreichen, werden verschiedene Erkennungsleistungen und Basisfähigkeiten benötigt. In der Praxis ging die Spezifikation auch teilweise von dieser Ebene aus, da letztendlich alle Funktionalitäten aus realisierbaren, also bekannten oder vorhandenen methodischen Basisleistungen zusammengesetzt werden müssen.
Die folgende Auflistung nennt die Teilleistungen und gibt Verweise auf die für die tatsächliche Umsetzung genutzten Algorithmen.

### Navigationsleistungen

- **2D & 3D Mapping** mittels 2D Occupancy Maps und NDT Maps als Basis für Hinderniswahrnehmung und Lokalisation; die konkrete Umsetzung ist in [Einhorn und Gross, 2015] beschrieben.
- **Pfadplanung** zu statischen und dynamischen Zielpunkten erfolgt mittels E* auf einer 2D Belegtheitskarte mit 3cm Auflösung, beschrieben in [Gross et al., 2014].
- **Lokalisation** in der Einsatzumgebung ist nach dem Adaptive Monte Carlo Lokalisation (AMCL) [Fox et al., 1999] Algorithmus implementiert und nutzt den 2D Laser Scan sowie die Odometrie und ein Gyrometer.
- **Visuelle Lokalisation** für den Andockvorgang. Da die Genauigkeit der AMCL nicht ausreicht, um 2cm genau auf die Ladestation zu fahren, er-

[4] Diese Arbeit wurde vom Autor betreut.

folgt die Lokalisation in der Nähe der Station anhand eines Kamerabildes und eines Schachbrettmusters an der Wand.

- **Bewegungssteuerung** erfolgt mit einer abgewandelten Implementierung nach dem Dynamic Window Approach (DWA) [Fox et al., 1997]. Die konkrete Realisierung ist in [Weinrich, 2015] und [Gross et al., 2014] zu finden.

#### Multimodale Personendetektion und Tracking

- **Personendetektion** erfolgt auf mehreren unabhängigen Wegen basierend auf verschiedenen Sensoren. Auf dem Weitwinkelkamerabild wird kontur-basiert [Weinrich et al., 2012], bewegungs-basiert (im Stand) und gesichts-basiert [Viola und Jones, 2004] nach Personen gesucht. Im Tiefenbild der Asus-Xtion an der Stirn des Roboters kann, wenn diese nach vorn gerichtet ist, die OpenNI Bibliothek zur Analyse der Nutzerpose genutzt werden. Ebenfalls auf der 3D Punktewolke erfolgt die Detektion gestürzter Personen [Schneemann, 2013]. Zusätzlich wird im 2D-Laser-Scan eine Erkennung von Beinpaaren [Weinrich et al., 2014] vorgenommen.
- **Personentracking** Die Fusion der vielfältigen Hypothesen und das Tracking der Nutzer über die Zeit wird mit einem 7D Kalman-Filter [Volkhardt et al., 2013] realisiert.

#### Nutzeranalyse

- **Kopfposentracking** mittels Regularized Landmark Mean-Shift (RLMS) [Saragih et al., 2011] im frontalen Kamerabild ermöglicht die Auswertung von Farbschwankungen im Gesicht zur Extraktion der Pulsrate [Stricker et al., 2014][5]. Die Kopfposeninformation könnte weiterhin für eine Schätzung der Blickrichtung zur Aufmerksamkeitsschätzung und einer Analyse von Kopfgesten genutzt werden.
- **Bewegungsanalyse während der Bewegungsübungen** erfolgt wie bereits erwähnt mittels des Posentrackers der OpenNI Bibliothek basierend auf den Tiefendaten der Asus Xtion.
- **Messgeräte für Vitalparameter**: Als Demonstrator für die Möglich-

[5] Der Autor ist Co-Autor dieses Papers.

keit, verschiedene medizinische Messgeräte vom Roboter mitführen zu lassen, wurde ein Fingerclip Pulsoximeter verbaut, wobei die Puls und Sauerstoffdaten direkt im Roboter erfasst werden.

#### Interaktion

- **Sprachsynthese** erfolgt offline mittels des Loquendo TTS Director$^{TM}$Tools.
- **Spracherkennung** wurde zu Versuchszwecken auf einem stationären PC im Roboterlabor, basierend auf dem open source Tool Julius, realisiert. Diese über WLAN mit dem Roboter kommunizierende Lösung wurde gewählt, um ein Beam-forming Mikrofon zur Verbesserung der Audiosignale nutzen zu können [Schnürer, 2015][6].

#### Service-spezifische Hintergrundfunktionalitäten

- **Fernsteuerinterface** als roboterseitige Gegenstelle zur Fernsteuerapp auf einem Mobilgerät.
- **Videotelefoniebackend** zur Realisierung des Verbindungsaufbaus und Streaming von Audio- und Videodaten.
- **Google Calendar Anbindung** für die Verwaltung und den Austausch von Termindaten.

## 2.7 Hardware

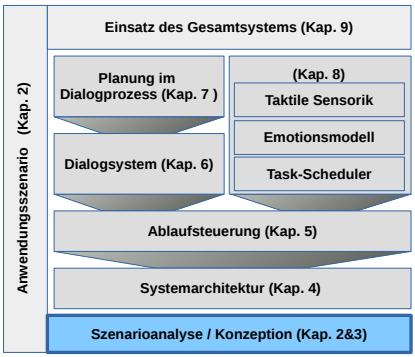
Die technischen Basisleistungen gehen eng mit der Hardwareausstattung der Roboterplattform einher. Die verfügbaren Komponenten waren größtenteils aus dem Vorgängerprojekt gegeben, konnten aber im Rahmen des SERROGA Projektes erweitert werden. Hierbei sind zum einen die Anbindung der Medizingeräte und zum anderen die sensorischen Erweiterungen zur Erkennung von Berührungen durch den Nutzer zu nennen (beschrieben in Kap 8). Außerdem wurden ein Funkempfänger und ein zusätzlicher PC für die Verarbeitung der Bilddaten integriert. Die resultierende Hardwarekonfiguration wurde bereits in Abschnitt 1.2 geschildert.

[6] Diese Arbeit wurde vom Autor betreut.

## 2.8 Fazit

Dieses Kapitel konnte zeigen, welche Zielstellung mit dem Projekt SERROGA verfolgt wurden, und wie die Anforderungsanalyse für ein interaktives Robotersystem durchgeführt wurde. Abb. 2.1 (Seite 20) fasst den Systemanalyseprozess für den SERROGA Roboter grafisch zusammen. Beginnend bei den Rollen, über die beschriebenen Services und die darin identifizierten Verhaltensmuster, stehen am Ende die benötigten technischen Basisleistungen und die Hardwarevoraussetzungen für die Realisierung eines solchen Systems. Die Übersicht macht deutlich, wie komplex das Gesamtsystem letztendlich aufgebaut ist. Dennoch blieb bei der Schilderung bis jetzt die Ablaufsteuerung, welche alle Teile koordiniert und schließlich die Dialogführung mit dem Nutzer übernimmt, unberücksichtigt. Lediglich das angestrebte Grundprinzip der Interaktion mit dem Roboter als Wechselspiel von Fragen und Antworten bzw. Nachrichten und Bestätigung über ein multimodales Interface wurde beschrieben. Der Roboter soll als eigenständiges Individuum und Dialogpartner auftreten. Im Kapitel 4 wird aus der eben beschriebenen Analyse und weiteren Anforderungen eine Softwarearchitektur entworfen, die letztendlich die Grundlage für die Implementierung des SERROGA Systems darstellt. Das vorgestellte Vorgehen bei der Spezifikation eines Serviceroboters lässt sich generalisieren. Für die Umsetzung des ROREAS Roboters (siehe Abschn. 1.1.4) wurde eine äquivalente hierarchische Analyse durchgeführt, welche durch die Identifikation von gemeinsam genutzten Funktionalitäten / Bestandteilen die Gesamtspezifikation schlank und effizient macht.

# 3




# Adaptivität im Interaktionsverhalten

Nachdem im vorangegangenen Kapitel eine eher konzeptionelle Sicht auf die Entwicklung eines robotischen Assistenzsystems zu einer Übersicht der benötigten Teilfunktionalitäten geführt hatte, soll in diesem Kapitel das Interaktionsverhalten eines Roboters näher beleuchtet werden. Dabei stehen die Ergonomie und Nutzerzufriedenheit im Mittelpunkt, welche durch Adaptivität und Personalisierung erreicht werden sollen.

In der gegenwärtigen Kommunikationstechnik aber auch in web-basierten Plattformen ist eine Personalisierung und Nutzeranpassung nicht mehr wegzudenken und wird oftmals nicht mehr bewusst wahrgenommen. Daher ist es auch für einen sozialen Assistenzroboter essenziell, ein Mindestmaß an Adaptivität in der Gestaltung des Nutzerinterfaces mitzubringen, um die potentiellen Nutzer nicht zu enttäuschen. Es entsteht sozusagen ein adaptives Nutzerinterface.

Langley [Langley, 1997] definiert ein adaptives Interface wie folgt:

**„An adaptive user interface is an interactive software system that improves its ability to interact with a user based on partial experience with that user.“**

Nutzeradaptivität bezieht sich also auf die Fähigkeit, aus Erfahrungen im Umgang mit dem Nutzer und aus Beobachtungen seines Verhaltens und anderer

Eigenschaften zu lernen und dadurch das Verhalten im späteren Verlauf der Interaktion besser zu gestalten. Darüber hinaus geht der Begriff der Personalisierung. Dieser umfasst nicht nur die systeminternen Eigenschaften, das Verhalten an einen Nutzer anzupassen, sondern auch die Möglichkeiten, die der Nutzer hat, aktiv Einfluss auf Verhalten zu nehmen und die äußere Erscheinung eines Artefaktes zu verändern. Bevor näher auf die Möglichkeiten für die Realisierung von Adaptivität und Personalisierung eingegangen wird, folgt in diesem Kapitel zunächst eine Motivation für die Bereitstellung dieser. Weiterhin wird die Einsatzsituation und speziell der Ablauf einer Langzeitinteraktion mit dem Roboter genauer analysiert, um Aspekte der Adaptivität zu identifizieren, welche in das im Kapitel 2 beschriebene System integriert werden sollten.

## 3.1 Gründe für Adaptivität im Interaktionsverhalten

Die Notwendigkeit von Adaptivität im Interaktionsprozess lässt sich zunächst durch die wachsende Variabilität der Nutzer begründen. Ein „Design for All“ Ansatz lässt sich aufgrund teilweise widersprüchlicher Vorlieben und Erwartungen bei einem sehr komplexen System kaum noch realisieren. Die Zielgruppe weist unter Umständen ein weites Spektrum unterschiedlicher Erfahrungen, unterschiedlichen Alters sowie kultureller Unterschiede auf, die alle im Design des Systems berücksichtigt werden müssten. Oftmals ist es zur Entwicklungszeit gar unmöglich oder zu aufwändig, die genauen Wünsche der Nutzer für alle möglichen Situationen vorherzusehen. Daher bietet es sich an, das System diese Anpassung zur Laufzeit erledigen zu lassen.
Ein weiteres Argument gegen ein statisches Interaktionsverhalten findet sich im unwillkürlichen Lern- und Entwicklungsprozess seitens der Nutzer, welcher während einer längeren Nutzung eines Systems eintritt. Der unterschiedliche Kenntnisstand im Umgang mit dem System erfordert eine parallele Weiterentwicklung der Umgangsform seitens des Roboters, um zum Beispiel anfänglich notwendige Hilfestellungen und explizite Einstellmöglichkeiten zugunsten besserer Effizienz nach und nach zu reduzieren.
Durch Berücksichtigung individueller Vorlieben, Eigenheiten und Fähigkeiten der Nutzer soll im wesentlichen die Akzeptanz für ein Assistenzsystem

gesteigert werden. Nutzer wollen oder können sich nicht in die Vielzahl von heutzutage vorhandenen Interfaces einarbeiten und an diese anpassen, vielmehr sind sie es mittlerweile gewohnt, dass sich diese an ihre Nutzungsform anpassen.

Der oben genannte Begriff der **Personalisierung** ist eher von Konsumerprodukten oder aus dem Web-Bereich bekannt. Er meint neben der Adaption auch das durch den Nutzer selbst vorgenommene Individualisieren eines Produktes und bezeichnet eine auftretende Spezialisierung, die die Erscheinung eines Produktes letztendlich für jeden Nutzer unterschiedlich ausfallen lässt.

Blom [Blom, 2000] identifiziert verschiedene Gründe für Personalisierung, die jene für Adaptivität noch erweitern:

- Anpassung an Arbeitsstil zur Optimierung von Workflows,
- Anpassung an körperliche Unterschiede (beispielsweise Seh- oder Hörbehinderungen),
- Anpassung eines Produktes, um bei sich selbst eine emotionale Reaktion hervorzurufen (Klingelton beim Handy)
- Als Ausdruck von Individualität

Nachdem so viele positive Wirkungen von Adaptivität in Nutzerinterfaces aufgelistet wurden, sollen auch die Nachteile nicht verschwiegen werden. Nach Jameson [Jameson, 2008] kann eine automatische Adaption auch Probleme bei **Usability, Vorhersagbarkeit und Verständnis des Systems** verursachen.

Nutzer adaptieren sich an mehrfach wiederholte Prozesse in der Handhabung eines Systems und können diese dann sogar unterbewusst und sehr effizient ausführen. Wenn das System diese Abläufe auch nur geringfügig abändert, wird dieser Automatismus gestört und erfordert seitens des Nutzers wieder mehr Aufmerksamkeit als ohne Adaption.

Da aufgrund der Verwendung der adaptiven Mechanismen ein System auch Entscheidungen vom Nutzer übernimmt, muss die Kompetenz des Systems gut vorhersagbar sein. Sonst besteht die Gefahr, dass sich der Nutzer auf Systemaussagen und Entscheidungen verlässt, die nicht so sicher sind wie sie scheinen. Ein Beispiel für diese Problematik ist eine Websuchmaschine, die aufgrund der vorhergehenden Suchanfragen und vieler weiterer undurch-

schaubarer Parameter die Ergebnisliste umsortiert und filtert.
Weiter führt Jameson eine **Einschränkung der Kontrollierbarkeit** und eine gewisse **Aufdringlichkeit** (obtrusiveness) an. Durch die autonomen Entscheidungen, sich in Zukunft anders zu verhalten, kann das System beim Nutzer eine gewisse Verunsicherung und damit verbunden ein Gefühl des Kontrollverlusts verursachen. Bei einer konkreten Implementierung von Adaptivität sollte dies daher geprüft werden, um nicht die positiven Effekte der Anpassung aufzuheben. Eine Möglichkeit, das Kontrollgefühl zu wahren, besteht beispielsweise darin, vor einer Änderung den Nutzer um Bestätigung zu bitten.
Ein letzter Punkt, der mit einer Anpassung verbunden ist, stellt eine mögliche **Einschränkung des genutzten Funktionsumfangs** und somit eine Beschränkung der Breite der Nutzungserfahrung dar. Es sollte daher beim Design vermieden werden, dass beispielsweise durch Menüs, die sich an die Häufigkeit der genutzten Funktionen anpassen, dem Nutzer bestimmte Möglichkeiten weniger bewusst, oder gar verwehrt werden. Durch die zu dominante Nutzung der eigenen Vorgeschichte besteht immer Gefahr, sich in einer dadurch begrenzten Subdomäne zu verfangen.

## 3.2 Grundlagen für eine Anpassung

Nachdem die Vor- und Nachteile einer Anpassung des Interaktionsverhaltens beleuchtet wurden, soll nun differenziert werden, welche Aspekte eines Verhaltens aufgrund welcher Einflussgrößen verändert werden können.
Es lassen sich folgende fünf Gruppen identifizieren:

**Vorgaben durch Beobachtungen der Interaktionen:** Aus Beobachtungen des Interaktionsverlaufs mit einem Nutzer lassen sich Aspekte wie Häufigkeiten der Nutzung von Services, Präferenzen in Auswahlsituationen oder auch Zeitpunkte und Situationsbezug von Interaktionsinhalten ermitteln. Mit längerer Beobachtungsdauer sollte dabei die Signifikanz dieser Informationen steigen und kann genutzt werden, um beispielsweise Auswahllisten entsprechend der Prioritäten umzusortieren, Unsicherheiten in der Inputinterpretation durch Nutzung der situationsbezogenen a-priori Informationen zu kom-

pensieren oder gar proaktiv situationsbezogene Inhalte und Services anzubieten. Bei der Personalisierung von grafischen Nutzerinterfaces existieren auch Arbeiten, die die Gestaltung des Screen Layouts an den Arbeitsfluss des Nutzers anpassen und z.B. Dialoge an die Reihenfolge und die Art der Bildschirmelemente anhand der Entropie der zu treffenden Entscheidung anpassen [Gajos et al., 2004].

**Explorierbare Verhaltensweisen:** Neben den durch Nutzervorgaben erlernten Verhaltensweisen sind auch Verhaltensvarianten unterscheidbar, die keinen direkten Bezug zu vom Nutzer ausgeführten Entscheidungen in der Vergangenheit besitzen. Dies betrifft beispielsweise Variationen im Ablauf von Dialogen, wenn redundante Dialogvarianten vorhanden sind. So kann beispielsweise in einem Begrüßungsdialog ein Smalltalk dazugehören oder nicht, oder für die Bestätigung von Eingaben besteht die Möglichkeit, alle einzeln bestätigen zu lassen oder diese implizit mit den weiteren Fragen im Dialog zu untermauern[1]. Weiterhin kann eine Variation des Auftretens eines Roboters (unterwürfig, höflich, bestimmt oder fordernd dominant) oder die Wahl von Argumentationsmöglichkeiten, wenn es darum geht den Nutzer zu einer Kooperation zu bewegen, zur Entscheidung stehen.
Diese Möglichkeiten sollten prinzipiell alle zielführend sein, können sich aber in ihrer Effizienz und der Zustimmung des Nutzers unterscheiden. Es bleibt demnach nur die Möglichkeit, die Varianten auszuprobieren und den Erfolg oder die Wirkung am Nutzer zu bestimmen. Dabei lässt sich dieses Kriterium meist auf einen skalaren Wert herunter brechen, der im Bereich des Reinforcement Learning als **Reward** bekannt ist. Durch Bestimmung des Rewards, welcher mittels aller alternativen Verhaltensmöglichkeiten erreichbar ist, kann anschließend die optimale Auswahl getroffen werden. Hierbei ist zu berücksichtigen, dass der Reward nicht immer unmittelbar nach der Aktion verfügbar ist (**immediate reward**), sondern unter Umständen erst am Ende einer längeren Interaktionssequenz in Erscheinung tritt (**delayed reward**).
Als Quellen für den Reward können dabei einerseits systeminterne Optimierungskriterien, wie beispielsweise die Dauer von Dialogen oder der Erfolg,

[1] In Anhang A 6.3 finden sich nähere Informationen zu diesem „Grounding“ genannten Prozess.

den Nutzer zu einer Zusammenarbeit zu bewegen, oder andererseits direkte Belohnungs- oder Bestrafungssignale vom Nutzer dienen. Beispielsweise kann es als Bestrafung gewertet werden, wenn ein Nutzer einen Dialog abbricht, da ihm der Inhalt oder die Form nicht zusagt und er darüber verärgert ist.
Der Roboter kann auch explizit nachfragen, um einen Reward zu bekommen oder aus Nutzerentscheidungen zu roboterseitigen Vorschlägen seine Schlüsse zu ziehen.
In allen Fällen ist eine Exploration der Varianten erforderlich. Insbesondere bei verzögertem Reward kann es dabei auch nötig sein, in verschiedenen Sequenzen ähnliche Verhalten mehrfach zu erproben, auch wenn diese bereits in anderen Situationen durch den Nutzer als negativ bewertet wurden. Dies kann zu einem Eindruck von Inkonsistenz führen und den Nutzer verwirren (siehe experimentelle Erkenntnisse in Kap. 7).

**Globale Einstellungen:** Neben den Varianten im Verhalten, die problemlos exploriert werden können, existieren auch Auswahlmöglichkeiten, deren Variation zur Laufzeit zu Inkonsistenzen in der Erscheinung des Roboters führen würden. Beispiele hierfür sind die Auswahl der Stimme für die Sprachsynthese oder die Form der Anrede (Du oder Sie). Diese Entscheidungen müssen durch den Nutzer explizit getroffen werden. Es bietet sich dafür an, diese Parameter in einem systemgetriggerten Einstellungsdialog am Anfang der Interaktion abzufragen.

**Verhalten bedingt durch Nutzereigenschaften:** Neben den Verhaltensentscheidungen, die auf Nutzerentscheidungen in der Vergangenheit basieren, existieren auch Unterschiede in den Vorlieben oder Anforderungen, die indirekt mit Nutzereigenschaften verbunden sind. Falls der Roboter diese Nutzereigenschaften bestimmen kann, kann dieses a-priori Wissen über bestimmte Nutzergruppen genutzt werden, um entsprechend angepasstes Verhalten zu generieren. Ein Beispiel hierfür ist eine Geschlechtsschätzung und die darauf basierende Auswahl von Stimme und Dialoginhalten in einem Einkaufsassistenten.
Eine weitere Nutzereigenschaft, die wesentlich für eine funktionierende Interaktion mit einem technischen System ist, ist der **Erfahrungsgrad** im Umgang mit diesem. Werden beispielsweise lange Verzögerungen zwischen den Einga-

ben des Nutzers erkannt, so kann auf eine Unsicherheit im Umgang geschlossen werden, und das Dialogverhalten sollte mehr Instruktionen zum Umgang geben oder gar von einem nutzergetriebenen Dialog zu einem systemgetriebenen Dialog wechseln.
Quellen für die Ermittlung der Nutzereigenschaften können also entweder direkte Erkennungsleistungen sein oder wiederum aus dem Interaktionsverhalten des Nutzers abgelesen werden. Letzteres spielt dabei in gewisser Weise in die erste Gruppe der auf Nutzerbeobachtungen basierenden Anpassung hinein.

**Zeitlich bedingte Veränderungen:** Wie bereits bei den Nutzereigenschaften erwähnt, ist das Wissen um den Hilfebedarf im Umgang mit dem System essentiell für einen reibungslosen Dialog. Der Grad an Hilfestellungen und auch andere Aspekte eines Dialogverhaltens unterliegen aber zusätzlich einer natürlichen Entwicklung während eines Langzeitdialogs und können daher unabhängig von der ständigen Beobachtung über die Zeit variiert werden. In Abschnitt 3.3.1 wird noch einmal näher auf die zeitliche Entwicklung von bestimmten Verhaltensparametern eingegangen.

### 3.2.1 Adaptivität vs. Adaptierbarkeit

Personalisierung eines Systems kann entweder durch den Nutzer oder durch das System selbst geschehen. Wenn ein System die manuelle Anpassung durch den Nutzer ermöglicht soll dieses **adaptierbar** genannt werden. Im Gegensatz wird von **adpativen** Systemen gesprochen, wenn diese von sich aus die Anpassung und Optimierung vornehmen. Adaptibilität und Adaptivität werden ausführlich in [Jameson, 2008] diskutiert.
In allen Szenarien, in denen der Roboter von nicht eingewiesenen Laien benutzt wird, kann man nicht erwarten, dass diese explizit am System Einstellungen vornehmen. Vielmehr muss durch den Interaktionsprozess eine Personalisierung des Systems auf den entsprechenden Nutzer stattfinden und somit eine Adaptivität vorhanden sein.
Wie aus den Diskussionen von Abschnitt 3.1 und 3.2 hervor geht, kann jedoch nicht immer auf eine explizite Mitbestimmung des Nutzers verzichtet werden, und es ist nötig, auch Aspekte der Adaptibilität beim Design zu berücksichtigen.

Bunt [Bunt et al., 2010] schlägt beispielsweise im Kontext von grafischen Nutzerinterfaces vor, die Adaption vom Nutzer triggern zu lassen, dann aber vom System die neuen Einstellungen vorschlagen und vom Nutzer bestätigen zu lassen, um die Irritationen durch sich unaufgefordert änderndes Verhalten zu minimieren. Für einen sozialen Assistenzroboter allerdings wird ein solches Verhalten vom Autor als nicht sinnvoll erachtet, da der Nutzer nicht weiß, was für alternative Verhaltensweisen existieren und wann eine Veränderung sinnvoll ist.
Laut [Heerink, 2011] bevorzugen ältere Nutzer, welche auch die Zielgruppe für den in dieser Arbeit als Beispiel genutzten Gesundheitsassistenten darstellen, Adaptivität gegenüber Adaptierbarkeit, wollen aber die Kontrolle nicht verlieren. Bei einer vergleichenden Untersuchung stellte Heerink allerdings fest, dass ein adaptiver Assistenzroboter, der bei Anpassungen des Verhaltens erst beim Nutzer um Bestätigung bittet, bezüglich der wahrgenommenen Kontrollierbarkeit nicht besser eingestuft wird als ein Roboter, der sich ohne Nachfragen adaptiert. Das Problem des potentiell wahrgenommenen Kontrollverlusts scheint sich bei einem sozialen Assistenzroboter, welchem evtl. auch ein gewisses Maß an Autonomie und Eigenintelligenz zugeschrieben wird, also zu relativieren. In der Entwicklung wird daher weiter eine direkte Adaptivität des Systems angestrebt.

### 3.2.2 Online vs. Offline

Adaptivität und Lernfähigkeit wird im Kontext eines Systems, welches im Langzeiteinsatz mit einem Nutzer ist, oft als online lernen wahrgenommen. Das heißt, dass sich das System zur Laufzeit verändert. Später wird noch ersichtlich, dass nicht jede Individualisierung eines Verhaltens auf Lernprozesse zur Laufzeit zurückzuführen ist. In der Literatur zu natürlichsprachlichen Dialogsystemen (siehe Anhang A 6.2) ist vielmals auch ein Entwicklungsprozess geschildert, welcher eine Prototypenphase vorsieht, in welcher Interaktionsdaten mit realen oder simulierten Nutzern gesammelt werden. Anschließend findet über diesen ein Optimierungsprozess statt, welcher die Interaktionspolicy einmalig vor dem finalen Einsatz lernt. Während des realen Einsatzes können weiter Interaktionsdaten gesammelt werden, um die Optimierung von

Zeit zu Zeit offline erneut auszuführen. Langley [Langley, 1997] spricht in diesem Fall von **gelernten** Systemen im Gegensatz zu **lernenden** Systemen, welche online zur Einsatzzeit aus den eben erfassten Daten Schlüsse ziehen und diese unmittelbar zur Veränderung ihrer Policy (Strategie) verwenden.
Beide Herangehensweisen haben ihre Berechtigung in Abhängigkeit vom Einsatzszenario des Systems. Im konkreten Fall des Gesundheitsassistenten für den kontinuierlichen Einsatz in einem Privathaushalt und zwecks schneller Reaktionen auf wenige zur Verfügung stehende Interaktionsbeispiele sollte eine life-long online Adaptivität möglich sein.

## 3.3 Adaptionsmöglichkeiten im Kurzzeit- und Langzeitdialog

Wie bereits im vorangegangenen Abschnitt erläutert bieten sich verschiedene Adaptionsformen für verschiedene Einsatzzwecke an. Hier ist im Allgemeinen die Dauer der Nutzung des Systems durch einen individuellen Nutzer und die damit zur Verfügung stehende Menge an Interaktionsdaten ausschlaggebend. Demzufolge können Einsatzszenarien in **Kurzzeitdialoge** und **Langzeitdialoge** unterteilt werden. Kurzzeitdialog käme beispielsweise bei einem Shoppingassistenten oder bei einem Museumsführer vor, während der Langzeitdialog im bereits mehrfach strapazierten häuslichen Gesundheitsassistentenbeispiel vorkäme.
Mit der Interaktionsdauer ist meist auch verbunden, ob mit einem einzelnen Nutzer oder mehreren wechselnden Nutzern zu rechnen ist. Im Fall von wechselnden Interaktionspartnern ist ebenfalls zu berücksichtigen, ob bei mehrmaligem Kontakt der Nutzer durch das System wiedererkannt werden kann und somit eine Fortsetzung der letzten Interaktionssession möglich ist. Damit wäre quasi ein Langzeitdialog mit Mehrnutzerbetrieb möglich. Beispiel dafür wäre ein Rehabilitationsassistent in einer öffentlichen Einsatzumgebung wie einer Klinik, der zu verschiedenen Zeiten wechselnde aber bekannte Patienten betreut (siehe ROREAS Abschn. 1.1.4). Auch im Experimentalsetup in Abschn. 7.1 kommt ein Mehrnutzerbetrieb vor.
Für einen **Kurzzeitdialog** steht innerhalb einer Session mit einem Nutzer meist nicht mehr als ein Zyklus für einen Dialog zur Verfügung. Daher ist es

eher ausgeschlossen, dass Erfahrungen daraus direkt mit diesem Nutzer noch für eine Verbesserung des Verhaltens genutzt werden können. Für Kurzzeitdialoge ist daher lediglich eine nutzerunspezifische Verbesserung des Verhaltens über alle Interaktionspartner hinweg denkbar. Dafür ist es nicht nötig diese online zu realisieren, sondern ein Batchlernen nach dem abgeschlossenen Dialog mit mehreren Nutzern ist aufgrund des Rechenaufwands zu empfehlen. Was im Sinne einer Personalisierung auch in dieser Situation möglich ist, ist die Adaption an bestimmte Nutzergruppen. Die Mittlung über die Interaktionen muss nicht notwendigerweise über alle Nutzer geschehen. Vielmehr ist es möglich und ratsam, die Nutzer anhand verschiedener Eigenschaften in Gruppen einzuteilen, um in diesen Gruppen evtl. vorherrschenden Eigenheiten gerecht zu werden. Ebenso kann versucht werden, den benötigten Grad an Hilfestellung während der Interaktion zu ermitteln und dieses Wissen direkt in der Interaktionsstrategie umzusetzen[2].
Inhalt der Adaption im Kurzzeitdialog könnten Anpassung an nicht nutzerbezogene Besonderheiten (geografische Eigenheiten, soziale Normen) und auch damit verbundene Optimierung des Workflows sein. Im Rahmen mehrerer Kurzzeitinteraktionen ist es auch möglich, die in Abschnitt 3.2 genannten globalen Einstellungen, die mit einem Nutzer nicht sinnvoll exploriert werden können, automatisch zu testen und zu optimieren. Bei verschiedenen Nutzern fällt durch die Variation dieser globalen Variablen kein Bruch in der Interaktion auf, solange sie für ein Individuum konstant gehalten werden.
Um die Adaptionsmöglichkeiten in einem **Langzeitdialog** zu erfassen, soll zunächst eine prototypische Phaseneinteilung den Entwicklungsprozess im Dialog charakterisieren. Die Phasen sind dabei nicht als diskrete Intervalle zu verstehen, sondern gehen gleitend ineinander über.

### 3.3.1 Phasen der Interaktion im Langzeitdialog

Abb. 3.1 zeigt den Verlauf eines Langzeitdialogs von der ersten Kontaktaufnahme zwischen Nutzer und System hin zu einer mehr oder weniger stabilen Phase, in der bei beiden Interaktionspartnern kaum noch nennenswerte Veränderungen im Umgang miteinander stattfinden. Evtl. kann sich durch äußere

[2] Dies entspricht der Klassifikation von Nutzergruppenzugehörigkeit anhand von Informationen, die erst während des Dialoges aus Inhalten oder beobachtetem Verhalten resultieren.

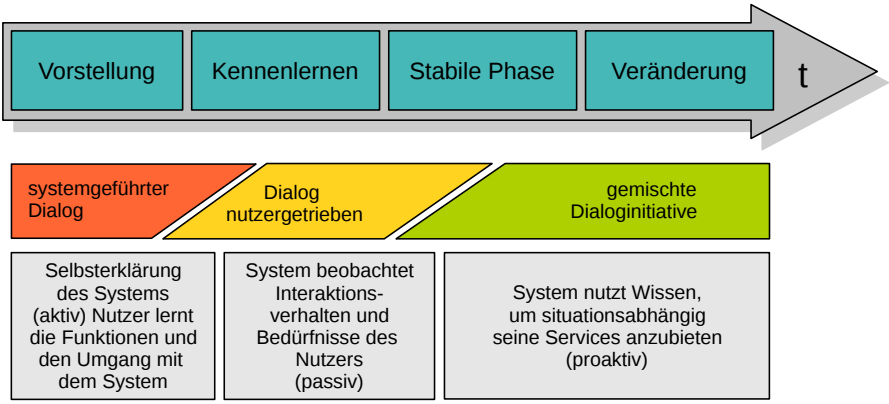



**Abbildung 3.1:** Phasen eines Langzeitdialogs, die mit jedem Nutzer durchlaufen werden sollten. Über die Zeit verlagert sich die Dialoginitiative mehrfach, und der Bedarf an Unterstützung nimmt ab.

Umstände schließlich dennoch das Nutzerverhalten verändern. Darauf sollte das System dann ebenfalls reagieren und sein Verhalten im Sinne des life-long-learnings nach regeln. Zu Beginn ist beim Nutzer ein großer Bedarf an Hilfestellungen und Erklärungen zu erwarten. Dadurch wird durch Unsicherheit auch wenig Initiative für eine Dialogführung vom Nutzer ausgehen. Das System sollte daher in dieser ersten Phase aktiv die Führung übernehmen und seine Funktionen, Services und den Umgang mit dem System vorstellen. Verschiedene Dienste könnten von Zeit zu Zeit situationsabhängig vorgeschlagen werden, damit der Nutzer über die zur Verfügung gestellten Assistenzfunktionen informiert wird. Anhand der Reaktionen des Nutzers kann der situationsbedingte Bedarf an Serviceleistungen dabei bestimmt werden. Dies führt bereits zur nächsten Phase, in der nun der Roboter in die Rolle des Beobachters schlüpft und ermittelt, wie der Nutzer mit dem System interagieren will und welche Funktionen wann benötigt werden. Damit die Nutzererfahrung dabei nicht beeinflusst wird, sollte die Dialoginitiative in dieser Phase vom System auf den Nutzer übergehen und das System sollte nun eher passiv agieren. Mit zunehmender Sicherheit über bestimmte Vorlieben und Bedürfnisse kann dann in der stabilen Phase das gesammelte Wissen genutzt werden,

um dem Nutzer proaktiv situationsbedingt Funktionen anzubieten oder Entscheidungen abzunehmen. Dadurch wird es auch möglich, dass sich Nutzer- und Systemintiative durchmischen und ein sogenannter mixed-initiative Dialog entsteht.

Auch im parallelen Mehrnutzerbetrieb oder beim Einsatz mehrerer Roboter können die unter dem Kurzzeitdialog beschriebenen Adaptionsmöglichkeiten basierend auf Gruppenzugehörigkeiten und beobachtbaren Nutzereigenschaften realisiert werden. Es ist aber eine Herausforderung, dieses Wissen in den online laufenden Adaptionsmechanismus für das Interaktionsverhalten mit dem einen Zielnutzer einfließen zu lassen.

## 3.4 Nutzeradaptivität im Rahmen der implementierten Architektur

Nachdem nun bereits die Notwendigkeit für Personalisierung und deren Erscheinungsformen betrachtet wurden, soll hier auf die konkreten Anpassungsleistungen eingegangen werden, die für die Umsetzung eines sozialen Serviceroboters als wichtig erachtet und im Rahmen der vorliegenden Arbeit in der konkreten Applikationsarchitektur berücksichtigt wurden.

Abb. 3.2 gibt einen Überblick zu den Teilleistungen, welche im Anschluss einzeln erläutert werden.

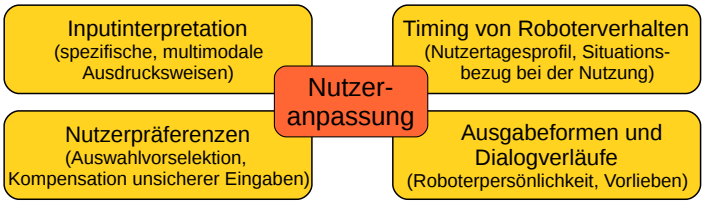



**Abbildung 3.2:** In dieser Arbeit behandelte Möglichkeiten zur Adaption des Verhaltens eines Assistenzroboters im Langzeitdialog mit einem Nutzer

### 3.4.1 Erlernen individueller Ausdrucksformen der Nutzer

Die Kommunikation mit einem mobilen Roboter ist am zweckmäßigsten, wenn sie auch auf Distanz erfolgen kann. Im angestrebten multimodalen Inputkonzept sind daher auch Gesten oder Spracheingaben vorgesehen. Leider sind diese Kanäle im Gegensatz zu einer Bedienung über einen Touchbildschirm mit zum Teil recht großen Unsicherheiten in ihrer Erkennungsgenauigkeit verbunden. Zu den Verlusten auf dem Übertragungskanal z.B. durch Störgeräusche kommt bei den genannten Modalitäten noch ein hoher Grad an interindividuellen Unterschieden hinzu. So bilden Dialekt und regionale oder kulturelle Eigenheiten sowohl bei der Sprache als auch Gestik eine Ursache für schwer vorherbestimmbare Bedeutungen mancher Äußerungen.
Im Rahmen der Inputfusion (siehe Abschnitt 6.9) lassen sich durch die gleichzeitige Analyse aller zur Verfügung stehenden Kanäle und der Fusion der Ergebnisse bereits Unsicherheiten reduzieren, aber im Sinne einer Adaption ist es auch möglich, durch diese Redundanz die personenspezifische Bedeutung von Äußerungen zu lernen. Im Verlauf der Langzeitinteraktion könnte sich damit auch die Fähigkeit entwickeln, bei Ausfall einzelner Kanäle die Eingaben anhand der anderen korrekt zu interpretieren.
Weitere Einzelheiten zur adaptiven Inputfusion finden sich in Abschn. 6.9.

### 3.4.2 Timing von proaktiven Serviceangeboten

Aus der Szenariobeschreibung in Kapitel 2 wird bereits ersichtlich, dass ein autonomer Assistenzroboter in seiner Rolle als Mitbewohner oder Butler meist im Hintergrund bleibt und sich zur Erbringung seiner Unterstützungsleistungen aktiv an den Nutzer wenden kann.
Hierbei sollte auf die aktuellen Tätigkeiten und Nutzerinteressen Rücksicht genommen werden. Beispielsweise sollte abhängig von der Wichtigkeit einer Interaktionsaufgabe der Nutzer nicht unbedingt bei einem Mittagsschlaf unterbrochen werden. Ebenso ist es nützlich, vorherzusehen, wann der Nutzer verfügbar sein wird und wann nicht, um beispielsweise Erinnerungen, die in einen Zeitraum der Abwesenheit fallen, rechtzeitig vor dem Verlassen der Wohnung ausliefern zu können.
Grundlage für ein solches adaptives Timing bildet ein Modell der Nutzerver-

fügbarkeit, mittels welchem eine Prädiktion möglich wird. Abschnitt 8.3 geht näher auf einen Lösungsansatz für diesen Aspekt ein.

### 3.4.3 Optimierung der Dialoggestaltung

Ein weiterer Aspekt, der eine Individualisierung aber auch im Mehrnutzerbetrieb eine allgemeine Optimierung ermöglichen soll, ist die Variation von Dialoginhalten, Dialogverlaufsvarianten und Ausdrucksweisen, um Ziele der Interaktion bestmöglich zu erreichen. Hier stehen vor allem die explorierbaren Verhaltensweisen aus Abschnitt 3.2 im Fokus. Durch ein online Lernen von Nutzerreaktionen auf bestimmte Dialogaktionen wird es möglich, für Aktionssequenzen vorherzusagen, mit welcher Wahrscheinlichkeit diese in einen gewünschten Zielzustand führen. Zusätzlich wird auch die Nutzerzufriedenheit durch direkte oder indirekte Rewardsignale vom Nutzer im System mit berücksichtigt und dadurch über die Zeit hoffentlich erhöht.
In existierenden Systemen kommen für diese Anpassungen Verfahren aus dem Bereich des Reinforcement Learning zum Einsatz. Ein Überblick dazu ist in [Cuayáhuitl et al., 2010] zu finden. Aufgrund der hohen Komplexität des Zustandsraumes während eines Dialoges ist eine Optimierung dabei nicht in Echtzeit möglich. In dieser Arbeit wird daher ein anderer Ansatz verfolgt. Eine explizite Planung einer Aktionssequenz ermöglicht es, online ein Modell zu lernen und in jedem Schritt bereits das gesamte Wissen der Vorgeschichte zu berücksichtigen. Dies wird im Kapitel 7 genauer erläutert.
Die Optimierung der globalen Einstellungen wie Stimme, Form der Anrede oder Persönlichkeit des Roboters ist mittels der vorgesehenen Planung im Mehrnutzer-Kurzzeitdialog ebenfalls möglich. Bei Interaktionsbeginn könnten die entsprechenden Einstellungen vorgenommen und die Erfolgsaussichten während des Dialoges protokolliert werden. In zukünftigen Interaktionen kann die Auswahl der Parameter dann im Planungsprozess berücksichtigt und optimiert werden.

### 3.4.4 Berücksichtigung von Nutzerpräferenzen

Eine letzte Art der Anpassung, welche in der vorliegenden Arbeit Anwendung finden wird, ist die Nutzung von beobachteten Nutzerpräferenzen. Die

Dialoge zwischen Nutzer und Roboter bestehen oft aus einer Abfrage von verschiedenen Optionen und Parametern, um eine Servicefunktion wunschgemäß ausführen zu können. Dabei kann die Eingabe erleichtert werden, wenn kontextabhängig die Entscheidungen aus der Vergangenheit berücksichtigt werden.
Die Behandlung dieser Adaptionsform wird im Rahmen dieser Arbeit mittels des für die probabilistische Planung erstellten Interaktionsmodells (siehe Kap. 7) realisiert. Hier werden alle Nutzereingaben mit dem Dialogzustand als Kontext hinterlegt. Die Auswertung dieses Modells liefert eine Wahrscheinlichkeitsverteilung über die auszuwählenden Optionen, welche als virtuelle Eingabe verarbeitet wird. Durch die probabilistische Modellierung von Nutzereingaben, wie sie auch für die Fusion der Inputmodalitäten benötigt wird, ist es dadurch auch möglich, die beschriebene Nutzung der Präferenzen für die Auflösung von missverständlichen Eingaben zu nutzen.

## 3.5 Möglichkeiten zur Realisierung von Adaptivität

Im vorangegangenen Abschnitt wurde erläutert, welche Adaptionsleistungen mit dem vorgeschlagenen System umgesetzt werden können. Bevor dieses allerdings näher vorgestellt wird, sollen zunächst einmal die prinzipiellen Möglichkeiten zur Realisierung von Adaptivität im Interaktionsprozess herausgearbeitet werden.

### 3.5.1 Der Interaktionszyklus

Um einen Dialog zwischen Mensch und Maschine für einen Algorithmus fassbar zu machen, muss zunächst ein Modell für den Ablauf angenommen werden. Da sich ein Dialog aus einem Wechselspiel von aktiven und passiven Phasen für beide Interaktionspartner zusammensetzt, ist es sinnvoll, ein so genanntes **Turn taking** Modell anzusetzen. Hierbei wechselt die Initiative in einzelnen Turns zwischen den Dialogpartnern und auf eine Äußerung oder Frage des einen wird eine Reaktion des anderen Partners erwartet. Abschnitt 6.4 und folgende gehen näher auf diese Modellannahme ein. Das Turn taking Modell lässt sich auf das aus der Domäne des Reinforcement Learning

bekannte Modell des **Perception Action Cycle** übertragen, welches dazu dient, die Schnittstelle zwischen einem autonomen Agenten und der Umwelt, in der dieser agiert, zu beschreiben. Abb. 3.3 stellt eine Grundform dieses Modells dar. Die beiden Interaktionspartner Roboter und Nutzer finden sich in dieser Sichtweise als der autonome Agent „Roboter“ und die Umwelt, in der er agiert wieder, wobei sich der menschliche Dialogpartner als Teil der Umwelt betrachten lässt. Ein Zeitschritt im Perception Action Cycle besteht nun aus der Wahrnehmung des Umweltzustands über die Sensoren des Agenten und dann aus der Auswahl einer Aktion, die der Agent in seiner Umwelt ausführt. Daraufhin ändert sich der Zustand der Umwelt und ggf. existiert noch eine Quelle für eine Bewertung der Aktionsauswahl, die entweder aus der Umwelt stammt oder im inneren des Agenten angesiedelt ist. Anschließend beginnt der Zyklus von Neuem auf dem geänderten Zustand. Im Dialogmodell besteht ein Zeitschritt aus der Abfolge von zwei Turns. Zunächst findet ein Turn für den Nutzer statt, in dem im Allgemeinen Eingaben an das System gerichtet werden. Diese sind Ausdruck des Umweltzustands, der auch die Nutzerziele und dessen Modell vom Wissen beider Interaktionspartner enthält. Die Eingaben sind allgemein als Beobachtungen zu sehen, die der Roboter in einem Dialogschritt aufnimmt und zum Update seines inneren Modells des Dialogzustandes nutzt. Der geänderte Zustand wird dann als Grundlage für die Auswahl einer adäquaten Reaktion in Form einer Systemausgabe genutzt. Dies entspricht einem Turn des Systems und führt zu einer Wirkung beim Nutzer, welche die Änderung des Umweltzustands darstellt. Im nächsten Zyklus beginnt das Ganze wieder von vorn, indem das System auf eine Änderung des Zustands durch Nutzereingaben oder andere Ereignisse wartet. Die Zyklen erfolgen daher nicht in zeitlich gleichen Abständen, sondern sind durch die Wechsel der Initiative und das Tempo der Nutzereingaben bestimmt.

Das Modell in Abb. 3.3 zeigt die einfachste Form eines Dialogsystems. Grundlage für die Entscheidung, was als nächstes vom System ausgegeben werden soll, bildet in allen Realisierungsformen eine Repräsentation des aktuellen Zustandes, in dem die bereits vergangenen Dialogschritte und die dabei gewonnenen Informationen über Dialoginhalte, Ziele des Nutzers und dessen kognitiven Zustand abgebildet werden. Da mit jedem Schritt der Zustand wechselt, kann dieser als flüchtig bezeichnet werden. Der Zustand soll für die theoreti-

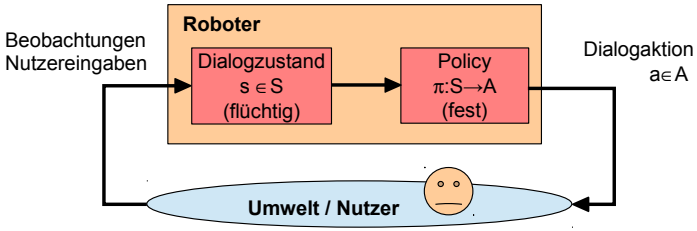



**Abbildung 3.3:** Interaktionszyklus in seiner einfachsten Form ohne Adaption an den Nutzer und sein Interaktionsverhalten. Der Dialogzustand bildet anhand der Eingaben und Beobachtungen die jeweilige Dialogsituation ab (ändert sich damit über die Zeit) und wird mittels einer fest definierten Abbildungsfunktion (Policy) auf eine Systemaktion abgebildet.

sche Betrachtung mit $s \in S$ bezeichnet werden. $S$ ist dabei der Wertebereich des Zustandsraums.
Der zweite grundlegende Bestandteil eines Dialogsystems ist eine Komponente zur Entscheidungsfindung, die im Allgemeinen eine Abbildung des Zustands auf eine Aktion $a$ aus einem endlichen Vorrat $A$ möglicher Aktionen realisiert. Diese Abbildung wird Policy $\pi : S \mapsto A$ genannt.

Es stellt sich nun die Frage: **Wo kommt Adaptivität in dieses Modell?**

Langley [Langley, 1999] gibt in seiner Definition eines adaptiven Nutzerinterfaces eine Hinweis darauf: **„An adaptive user interface is a software artifact that improves its ability to interact with a user by constructing a user model based on partial experience with that user.“**
Die Adaptivtät soll somit aus einem Modell vom Nutzer und seinem Verhalten herrühren.

### 3.5.2 Feste Policy aber Einsatz eines Nutzermodells

Die einfachste Möglichkeit, ein System für einen Nutzer als adaptiv erscheinen zu lassen, besteht also darin, ein Nutzermodell im Agenten anzulegen und bei der Entscheidungsfindung zu berücksichtigen, wie in Abb. 3.4 verdeutlicht. Im

Gegensatz zur Repräsentation des Dialogzustandes, welcher lediglich flüchtige Informationen enthält, die die aktuelle Interaktionssession betreffen, werden in einem Nutzermodell persistente Daten über Nutzer auf symbolischem Niveau gehalten. Diese können dann in späteren Sessions wiederverwendet werden. Die Daten in einem Nutzermodell können beispielsweise der Erfahrungslevel, die Altersklasse, das Geschlecht oder auch aus vorherigen Interaktionen resultierende Vorlieben sein. Im Schema wird der Nutzerzustand als $n \in N$ dargestellt. Und die Policy erweitert sich zu $\pi : S \times N \mapsto A$. Das bedeutet, dass der Designer des Systems für jede Nutzerklasse eine spezielle Variante der Policy vorsieht. Sobald in der Beobachtungsphase die Nutzerklasse festgestellt wurde, bewegt sich das System nur noch in einem nutzerspezifischen Unterraum und für verschiedene Nutzer stellt sich das Interaktionsverhalten des Systems verschieden (individualisiert) dar.

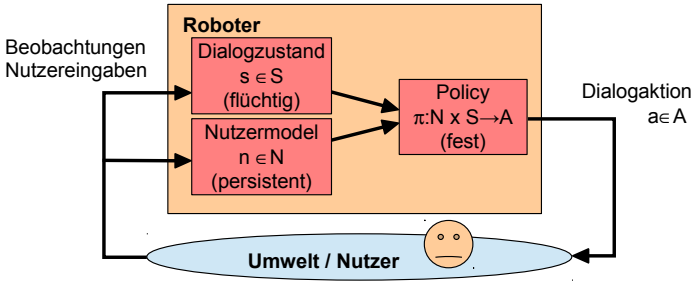



**Abbildung 3.4:** Einfache Adaptionsmöglichkeit durch Einführung eines persistenten Nutzermodells und Berücksichtigung dieses in der festen Policy der Aktionsauswahl

Diese Herangehensweise erfüllt noch nicht die Kriterien für ein **lernendes** System. Es werden lediglich direkte Beobachtungen gespeichert und es findet keine Inferenz oder Verarbeitung dieser Daten statt. Ein Nachteil dieser Variante ergibt sich aus der dadurch resultierenden Notwendigkeit, alle Varianten manuell durch den Designer festlegen zu müssen. Es kann damit nicht von Erfahrungen aus dem Umgang mit den Nutzern profitiert werden und die Optimalität bzgl. der Nutzerzufriedenheit oder anderer Erfolgskriterien ist nicht gesichert. Diese einfache Variante zur Personalisierung kann sowohl im Langzeitdialog mit einem Nutzer als auch im Mehrnutzerbetrieb und im

Kurzzeitdialog eingesetzt werden. Bei letzterem muss aber gewährleistet sein, dass die Nutzermerkmale relativ zügig innerhalb einer Session bestimmt werden können.

### 3.5.3 Variable Policy

Eine weitere Möglichkeit, eine echte Adaptivität ins System zu bringen, besteht darin, die Policy gemäß den Erfahrungen im Umgang mit dem Nutzer zu verändern. Diese Optimierung geschieht dabei auf Basis einer Zielfunktion, der sogenannten Rewardfunktion, welche sich über den erreichten Zuständen $S$ oder auch direkt vom Nutzer ableiten lässt. Durch diese Zielfunktion wird das ideale Verhalten lediglich implizit beschrieben und das nun **lernende** System versucht selbst, dieses zu erreichen. Zur Einsatzzeit werden dabei die nutzerspezifische Verteilung der Rewards über die Zustände, aber auch die typischen Übergänge zwischen den Zuständen in Abhängigkeit der gewählten Systemaktion berücksichtigt. Sie stellen in der resultierenden Policy somit ein **implizites Interaktionsmodell** dar (siehe Abb. 3.5).

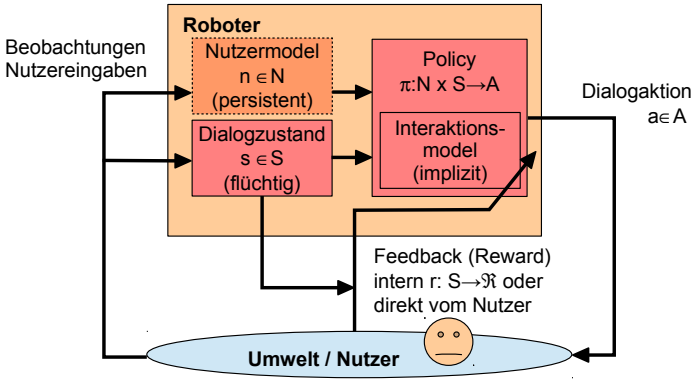



**Abbildung 3.5:** Adaptivität durch Variation der Policy mit implizitem Interaktionsmodell

Das Lernen der Policy kann entweder online durch eine schrittweise Anpassung oder offline im Batchmodus über einem aufgezeichneten Datensatz stattfinden.

Zusätzlich zur variablen Policy kann im Mehrnutzerbetrieb ein Nutzermodell hilfreich sein, um für verschiedene Nutzergruppen spezifische Verhalten auszubilden.

### 3.5.4 Planen zur Ausführungszeit

Eine letzte vom Autor identifizierte Möglichkeit, die Adaption des Verhaltens zu realisieren, besteht darin, die vorbestimmte Policy, welche im gelernten Fall einer impliziten Planung entspricht, durch eine explizite Planung zu ersetzen. Dabei bietet sich der Vorteil, dass nicht für jeden möglichen Zustand eine optimale Aktion berechnet werden muss, sondern lediglich für den aktuell vorliegenden. Dadurch ist es möglich, diese Optimierung zur Einsatzzeit online auszuführen und somit auch unmittelbar auf erlebte Nutzerreaktionen zu reagieren.

Die Grundlage für eine Planung bildet dabei ein Modell der erlebten Zustandsübergänge $T : S \times A \times S' \mapsto [0, 1]$ (hier als explizites Interaktionsmodell oder Transitionsmodell bezeichnet). Dieses modelliert die Wahrscheinlichkeit mit der eine Aktion $A$ in einem Zustand $S$ in einen Folgezustand $S'$ führt. Dieses lässt sich leicht online aufgrund der erlebten Übergänge erweitern. Neben den Zustandsübergangswahrscheinlichkeiten können während einer Planung auch Nutzerrewards berücksichtigt werden. Dadurch wird nicht nur ein möglichst effizienter weil kurzer Weg zu einem Dialogziel gesucht, sondern auch einer, der anderen Optimierungskriterien einer Rewardfunktion genügt. Insbesondere kann dabei auch die Nutzerzufriedenheit dargestellt werden. Neben dem Interaktionsmodell findet sich im Schema in Abb. 3.6 also auch ein Rewardmodell wieder, welches die Zustände $S$ auf einen reellwertigen Wert für ihre „Erwünschtheit“ aus Sicht der Nutzer abbildet.

Wie in den anderen Ansätzen kann auch bei der Planung eine explizite Modellierung von Nutzereigenschaften hilfreich sein, um in einem Mehrnutzerbetrieb spezialisiertes Verhalten zu generieren. Die Nutzerparameter $N$ würden dabei im Interaktionsmodell als Kontext mit berücksichtigt, welches in diesem Fall zu $T : N \times S \times A \times S \mapsto [0, 1]$ wird.

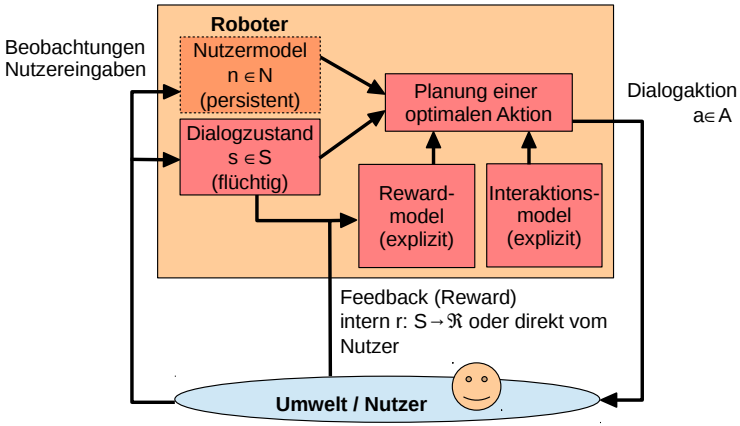



**Abbildung 3.6:** Adaption durch Planung zur Ausführungszeit über einem expliziten Modell der Interaktionsverläufe in der Vergangenheit. Ebenso kann ein Reward bei der Planung berücksichtigt werden, um bestimmte Zustände zu bevorzugen oder zu unterdrücken.

### 3.5.5 Exploration Exploitation Dilemma

Wie die lernende Policy, steht ein planendes System ebenfalls vor dem Explorationsproblem. Während der Planung kann lediglich das Wissen genutzt werden, welches schon einmal in einer Interaktion erprobt wurde. Die Konsistenz der generierten Aktionsfolgen kann, wie bei der lernenden Policy, durch eine geeignete Vorbelegung des Zustandsübergangsmodells gewährleistet werden. Für bestimmte Situationen werden dadurch unsinnige Aktionen von vorn herein ausgeschlossen und die Exploration bewegt sich nur in einem für den Nutzer plausiblen Unterraum.

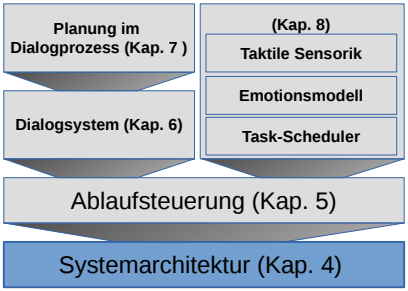
Für die vorliegende Implementierung eines planenden Dialogsystems wurde die aufwendige Vorbelegung der möglichen Zustandsübergänge durch eine manuelle Einschränkung der Policy umgangen. Der Designer von Dialogen kann eine Abbildung des Zustands auf eine Menge sinnvoller Systemaktionen angeben. Damit kann auch während der Exploration ein konsistenter Dialog gewährleistet werden. Dennoch bleibt der grundsätzliche Konflikt zwischen Exploration neuer Aktionsauswahlen und Ausnutzung des bereits erlernten

Wissens. Dieser kann durch eine wahrscheinlichkeitsgetriebene Auswahl mit sinkendem Anteil von Exploration und mittels einer optimistischen Initialisierung des Rewardmodells angegangen werden. Letztere sorgt für eine breitflächige Exploration, da neue Zustände immer potentiell ertragreicher sind als bereits beobachtete. Näheres zur Behandlung des Exploration Exploitation Dilemmas in der realisierten Dialogsteuerung findet sich in Abschn. 7.3 und Anhang A 7.4.

## 3.6 Fazit

Dieses Kapitel hat gezeigt, dass es notwendig ist, im Konzept für die Architektur eines sozialen Assistenzroboters Möglichkeiten für eine Anpassung an individuelle Vorlieben und Eigenheiten seiner Nutzer vorzusehen. Hierbei konnten neben Konfigurationsmöglichkeiten auch verschiedene Aspekte einer Adaption des Interaktionsverhaltens aufgezeigt werden, die auch online über lange Zeit hinweg ein Lernen ermöglichen. Es zeigte sich weiterhin, dass Adaptivität auch Probleme mit sich bringen kann, die beim Design der konkreten Nutzerschnittstelle zu berücksichtigen sind.
Zuletzt folgte eine theoretische Analyse, wie Adaptivität in einem Schema eines Dialogprozesses untergebracht werden könnte, und die präferierte Variante mittels einer online stattfindenden Planung wurde geschildert.
Eine detaillierte Beschreibung der implementierten Dialogsteuerung wird in Kapitel 6 gegeben, nachdem im folgenden Kapitel die Architektur eines Gesamtsystems vorgestellt werden soll, in welche sich das Dialogsystem einbettet.

# 4




# Systemarchitektur

In diesem Kapitel sollen die konzeptuellen Grundlagen für eine praktische Realisierung eines Serviceroboters gelegt werden. Es wird aufgezeigt, wo sich die vorgeschlagene Dialog- und Ablaufsteuerung (Kap. 5) in ein komplexes System aus verschiedensten funktionalen Teilsystemen einordnet, und wie diese strukturiert werden können, um für die Entwicklung der eigentlichen Services als definierte API (Application Programming Interface) zur Verfügung zu stehen.

Die Middleware, welche die Infrastruktur zur Kommunikation und Koordination der Softwaremodule bereitstellt, und auch die Applikationsarchitektur, welche die vorhandenen Module logisch organisiert und strukturiert, ergänzen somit die Betriebssystemfunktionalität, um die Realisierung eines äußerst komplexen Systems, wie einen autonomen Serviceroboter, überhaupt beherrschbar zu machen. Die weit verbreitete Middleware ROS (Robot Operating System) bringt bereits im Namen die Ähnlichkeit zu Betriebssystemfunktionen mit. ROS erfüllt hierbei als plattformunabhängiges Softwareframework im Wesentlichen klassische Middleware-Funktionen, wie Hardwareabstraktion (Gerätetreiber) und Kommunikationsinfrastruktur. Aber auch eine Paketverwaltung mit funktionalen Modulen zur Realisierung von robotischen Aufgabenstellungen gehören zu ROS. Die im Rahmen dieser Arbeit verwendete Middleware MIRA (MIddleware for Robotic Applications) [Einhorn et al., 2012] umfasst

einen vergleichbaren Aufgabenbereich.
Was die Middleware MIRA nicht umfasst, ist eine funktionale Abstraktionssicht, die zusammen mit einer konkreten Hardwarekonfiguration die robotischen Basisfunktionen kapselt und somit als Grundlage für die eigentliche Serviceentwicklung dient. Die Applikationsarchitektur, wie sie in dieser Arbeit verstanden werden soll, ist somit ein Vorschlag, wie die Möglichkeiten der Middleware genutzt werden können, um die benötigten Softwarekomponenten zu strukturieren. In Zusammenarbeit mit dem im Kapitel 6 vorgestellten Dialogsystem als Ablaufsteuerung entsteht dadurch eine Art Applikationsentwickler API, die in ihrer Aufgabe vergleichbar mit dem Android-SDK für die Entwicklung von Handy-Apps ist.

## 4.1 Anforderungen

Generell entstehen aus dem Ziel, variantenreiche Serviceroboter mit unterschiedlichsten Einsatzzwecken und Hardwarekonfigurationen zu entwickeln, grundlegende Anforderungen an die dafür benötigte Software. Dabei stehen soziale Serviceroboter mit dem Fokus auf die Interaktion mit Menschen im Mittelpunkt.
Im Folgenden sind die vom Autor identifizierten Anforderungen auf konzeptioneller und funktionaler Ebene aufgelistet.

### 4.1.1 Konzeptionelle Anforderungen

Zunächst sollen generelle Aspekte aus softwaretechnischer Sicht genannt werden, welche bereits bei der Wahl der genutzten Infrastruktur beachtet werden sollten, aber auch bei der weiteren Spezifikation von Details der Architektur eine Rolle spielen.

**Modularität der Software, Wiederverwendbarkeit der Softwaremodule und freie Kombinierbarkeit von Funktionen:** Diese Aspekte gehen aus der Komplexität des Gesamtsystems und der damit verbundenen Notwendigkeit, vielen Entwicklern ein gemeinsames Arbeiten zu ermöglichen, hervor. Außerdem werden einzelne Teilfunktionen unabhängig voneinander schrittweise weiterentwickelt und müssen trotzdem jederzeit in einem Gesamtsystem eingesetzt

werden können.

**Skalierbarkeit auf komplexere Systeme:** Neben der Entwicklung von Demonstratoren für einzelne Teilfunktionen, wie es immer wieder in der Forschung gehandhabt wird, sollte die Architektur auch übertragbar sein auf ein produktives Gesamtsystem mit weiterreichendem Funktionsumfang.

**Abstraktion von Hardwarespezifika:** Die Software soll nicht an eine bestimmte Roboterhardware gebunden sein. Der Austausch von bestimmten Sensoren und Aktoren muss transparent möglich sein und sich in den softwareseitigen Auswirkungen auf möglichst lokale Änderungen beschränken.

**Austauschbarkeit einzelner Teilfunktionalitäten, Schnittstellenabstraktion zu Basisfunktionen eines mobilen Roboters:** Um einzelnen Entwicklern das Gestalten von Anwendungen (Services) zu ermöglichen, ohne dass diese sich in die Details und inneren Abläufe der verschiedenen robotischen Basisfunktionalitäten einarbeiten müssen, sollte eine relativ schmale Schnittstelle zwischen den Modulen und besonders nach oben zur Anwendungsentwicklung hin realisiert werden. Wenn sich Implementierungsdetails der robotischen Funktionalität ändern, können Anwendungen (Services) trotzdem noch funktionieren.

**Schnelle Applikationsentwicklung, leichte Erlernbarkeit der API durch Übersichtlichkeit der Strukturen:** Im Zusammenhang mit der Schnittstellendefinition sollte demnach auch ein geeigneter Kompromiss zwischen Freiheit der Möglichkeiten, die eine Architektur offen lässt, und Einschränkungen durch die vorgegebenen Strukturen gegeben sein. Dies ermöglicht, dass Entwickler zügig den Umgang mit dem System erlernen können und somit auch die rationelle Umsetzung neuer robotischer Anwendungen.

**Erweiterbarkeit:** Trotz der strukturierenden Eigenschaften einer Softwarearchitektur muss es insbesondere im akademischen Bereich möglich bleiben, sowohl Services auf Anwendungsebene, als auch Basisfunktionalität flexibel hinzuzufügen. Die entsprechenden Schnittstellen sollten es daher zulassen, gegebenenfalls erweitert zu werden. Letztendlich sollten sogar die interne Ablaufsteuerung und das Dialogsystem nicht auf die vorgeschlagene Lösung eingeschränkt werden. Alternative Varianten sollten durch geeignete Interfaces auch dafür verwendet werden können.

### 4.1.2 Funktionale Anforderungen

Neben den konzeptionellen Randbedingungen lassen sich auch für die funktionalen Aspekte Anforderungen identifizieren.
Auf globaler Softwareebene gilt es folgende Aspekte zu berücksichtigen:

**Effizienz, parallele Ausführung vieler verschiedener Aufgaben, Einhaltung von Echtzeiteinschränkungen, Ressourcenverwaltung:** Für die Erfüllung der Aufgaben eines mobilen Serviceroboters ist es nötig, viele verschiedene Teilaufgaben gleichzeitig auszuführen. Beispielsweise muss die Bewegungssteuerung ständig auf die Umgebungssituation reagieren und währenddessen müssen Verfahren zur Personendetektion und Planungsaufgaben im Hintergrund laufen. Die Hindernisvermeidung unterliegt dabei beispielsweise harten Echtzeitbedingungen, wohingegen die Planung von Dialogschritten eher langsamer erfolgen kann. Für all diese Aufgaben ist es nötig, die begrenzten Ressourcen geschickt zu verteilen und gegebenenfalls nicht benötigte Teilmodule dynamisch abzuschalten.

**Verteilbarkeit der Rechenlast, Kommunikationsmöglichkeiten zwischen Softwarekomponenten:** Bereits die Middleware stellt diese Möglichkeiten zur Verfügung.

**Umgang mit unvollständigen und unsicheren Sensordaten:** Die Sensoren eines Roboters können im Allgemeinen nur einen Teilausschnitt der Umwelt erfassen und sind dabei zusätzlich mit Sensorrauschen beaufschlagt. Die eingesetzten Methoden für die Teilleistungen sollten dies berücksichtigen und in ihren Modellierungstechniken Unsicherheiten explizit repräsentieren.

**Sicherheit (gegen Manipulation von außen):** In vielen akademischen Anwendungen geht es meist nur darum, eine spezielle Teilleistung zu realisieren und zu demonstrieren. Für ein reales Produkt, welches ein Serviceroboter potentiell darstellt, gehört auch eine Absicherung gegen Manipulationen von außen zu den Anforderungen. Die heutigen Geräte sind meist vernetzt und nicht zuletzt daher gehört der Datenschutz auch in dieses Gebiet, da teilweise recht persönliche Daten wie Videobilder oder Audiodaten im Robotersystem verarbeitet und vorgehalten werden. Auch kann mit einem mobilen Roboter, der evtl. auch Manipulatoren aufweist, erheblicher Schaden angerichtet

werden, wenn von außen Kontrolle über das System ausgeübt wird. Die Sicherheit sollte daher von Grund auf im System verankert werden, da es sich als schwierig erweist, diese nachträglich nachzurüsten.

Auf Systemebene gilt es folgende Aspekte zu realisieren:

**Autonomie, Robustheit / Ausfallsicherheit, Fehlertoleranz und Fehlerbehandlung, Adaptivität, Kapselung der Sicherheit des Robotersystems (Selbsterhaltung), Gewährleistung der Sicherheit des Nutzers:** Ein Roboter wird heutzutage nicht mehr nur in für ihn präparierten Umgebungen eingesetzt, sondern auch in sehr dynamischen öffentlichen oder häuslichen Bereichen. Da nicht gewährleistet werden kann, dass nur sachkundige Nutzer mit dem Roboter arbeiten, wird es notwendig, dass der Roboter einen gewissen Grad an Autonomie besitzt. Ebenso muss jederzeit die Sicherheit der Personen im Umfeld gewährleistet werden, auch wenn Teile des Systems nicht wie vorgesehen funktionieren oder die Einsatzumgebung sich nicht wie vom Designer vorhergesehen verhält. In solchen Situationen sollte auf allen Ebenen eine Anpassung an neue Situationen (auch Fehlerzustände) und die Fähigkeit zur Introspektive möglich sein.
Die Serviceroboter, welche im Rahmen dieser Arbeit thematisiert werden, erfordern durch ihre Aufgaben und Einsatzumgebungen im Wesentlichen folgende Teilfunktionen:

**Wahrnehmungsfähigkeiten:** Für die Interaktion mit Nutzern muss der Roboter deren Anwesenheit und Position erfassen können. Außerdem ist es für eine intuitive Kommunikation hilfreich, bestimmte Nutzereigenschaften und Eingabemodalitäten wie Emotionen, Gesten oder Sprache zu erkennen. Neben den Personen ist auch die Geometrie der Einsatzumgebung zu erfassen, um sicher navigieren zu können.

**Navigationsfähigkeiten:** Wie der Name „mobiler Serviceroboter“ bereits sagt, steht die autonome Navigation des Roboters in seiner Einsatzumgebung im Zentrum seiner Funktionalität. Dabei sind die drei wesentlichen Aufgabengebiete **Mapping**, **Lokalisation** und **Bewegungssteuerung**.

**Interaktionsfähigkeiten:** Das Bild eines sozialen Assistenzroboters enthält als zentrales Element ebenfalls die Kooperation oder Interaktion mit Menschen, an denen bestimmte Assistenzleistungen erbracht werden sollen. Hierfür muss aufgrund der nicht näher geschulten Nutzergruppen besonders auf eine natürliche Kommunikation geachtet werden. Ebenso sollte sich der Nutzer stets über die nächsten Schritte des Systems im Klaren sein und bis zu einem gewissen Grad die internen Zustände erklärt bekommen.

**Ablaufsteuerung:** Um autonome Funktionen zur Selbsterhaltung und aufgabenorientierte Verhaltensweisen zu koordinieren, muss eine flexible Ablaufsteuerung im Zentrum der Architektur stehen. Zu dieser lässt sich auch die Dialogmodellierung und Dialogsteuerung zählen.

### 4.1.3 Anforderungen bezüglich der Interaktionsfähigkeiten

Von einem Serviceroboter wird im Rahmen dieser Arbeit eine spezielle Art und Weise der Interaktion mit dem Nutzer erwartet. Aus dieser heraus leiten sich folgende Anforderungen an die Interaktionsfähigkeiten ab:

**Multimodalität der Ein- und Ausgaben:** Da der Roboter möglichst intuitiv mit uneingewiesenen Nutzern interagieren soll, scheint es notwendig, mehrere Kommunikationskanäle bereitzustellen, um einen hohen Grad an gegenseitiger Verständlichkeit zu erreichen.

**Erscheinung des Roboters als Dialogpartner / Individuum:** Da es angestrebt wird, verschiedene Servicefunktionen modular im Sinne von Apps auf einem Roboter unterzubringen, ist es nötig zu erwähnen, dass trotz der Modularität der Roboter als eine eigenständige Persönlichkeit mit dem Nutzer in Konversation treten soll. Es gibt neben den in den servicebezogenen Teilmodulen verankerten Kommunikationsinhalten auch zentrale Aspekte, die bei jeder Kommunikation berücksichtigt werden müssen. Es soll serviceübergreifend ein Look-and-feel in der Bedienung realisiert werden und um soziale Roboter zu ermöglichen, kann auch ein zentraler emotionaler Zustand des Roboters kommuniziert werden.

**Intuitive Bedienbarkeit, variable Erscheinungsform:** Speziell für den Einsatz als Serviceroboter in der häuslichen Umgebung kann der Charakter der

Interaktion (Werkzeugcharakter oder sozialer Partner) anwendungsabhängig verschieden sein. Die Architektur sollte daher in diesem Aspekt Freiheiten bei der Umsetzung offen lassen.

**Adaptationsfähigkeit / Personalisierung:** Wenn Robotersysteme langfristig in ihrem Einsatzgebiet akzeptiert werden sollen, müssen bestimmte Lerneffekte und zeitlich variable Bedürfnisse der Nutzer berücksichtigt werden. Ebenso können die Erwartungen einer größeren und heterogeneren Nutzergruppe nur erfüllt werden, wenn in bestimmten Aspekten der Interaktion und des Verhaltens des Roboters im Allgemeinen eine Auswahlmöglichkeit vorgesehen wird.

Bevor die Randbedingungen für die konkrete Umsetzung einer Softwarearchitektur noch einmal zusammengefasst werden, soll im Anschluss zunächst der Stand der Technik in der Literatur gesichtet und der Entwicklungsprozess hin zu leistungsfähigen Architekturkonzepten geschildert werden.

## 4.2 State of the Art

Der Begriff **Architektur** kommt in der Literatur als Softwarearchitektur, Kontrollarchitektur, Systemarchitektur oder auch kognitive Architektur vor und hat in diesen Ausprägungen nicht immer einheitliche Bedeutungen. Eine Systemarchitektur kann dabei auf eine konkrete Realisierung von Hardware und Software, aber auch lediglich auf ein Muster, wie Komponenten arrangiert werden sollten, abzielen. Hier sollen eher die softwareseitigen und konzeptionellen Aspekte des Architekturbegriffs vertieft werden.
Es finden sich Architekturkonzepte für verschiedene Sichtweisen auf das Problem der Realisierung eines komplexen Gesamtsystems. Architektur geht dabei letztlich immer von einer Unterteilung des Ganzen aus. Dabei kommt der „Teile und Herrsche“ Gedanke zum Tragen.
Abbildung 4.1 zeigt auf, welche Spielarten von Architekturen unter den Gesichtspunkten **Softwaremodell** und **Organisationsform** unterschieden werden können.
Softwaretechnische Architekturparadigmen werden von Amoretti und Reggia-

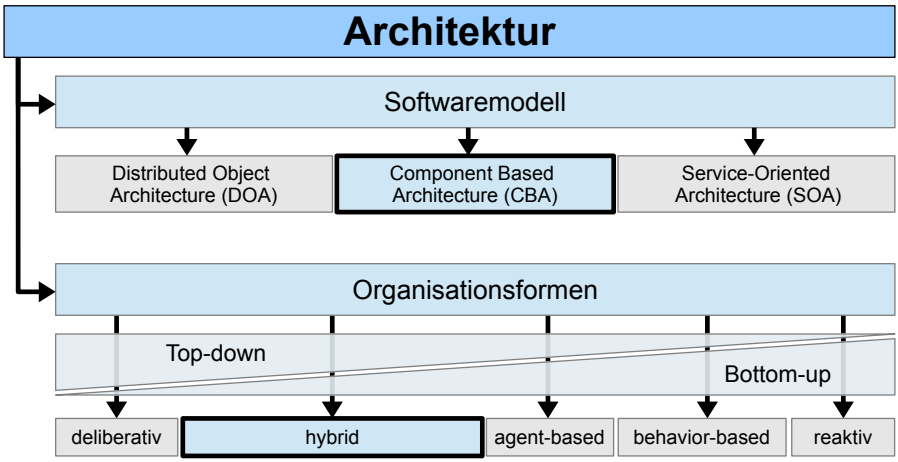



**Abbildung 4.1:** Aspekte und Ausprägungen von Architekturen zur Roboterkontrolle; die hervorgehobenen werden mit der entwickelten Architektur umgesetzt.

ni in [Amoretti und Reggiani, 2010] systematisiert. Sie identifizieren dabei drei Abstraktionsgrade, auf denen sie entsprechende Pattern oder Modelle ansiedeln. Die ersten beiden Abstraktionsgrade beschreiben hierbei, wie Module miteinander kommunizieren. Diese Aspekte werden meist für das verwendete Betriebssystem und die Middleware relevant. Die für diese Arbeit interessanten Paradigmen der dritten Abstraktionsschicht beziehen sich dagegen auf die strukturelle Sicht der Module:

- **Distributed Object Architecture** (DOA) überträgt die feingranuläre Untergliederung von Objekten aus der objektorientierten Programmierung mit ihren Operationen und Daten auf verteilte Systeme. Dabei entstehen sehr eng vermaschte Konstrukte mit sehr feingranulärer Unterteilung.
- **Component Based Architecture** (CBA) umfassen ähnliche Prinzipien wie DOA, beschreiben allerdings größere in sich abgeschlossene funktionale Einheiten, die unabhängig entwickelt und getestet werden können. Die Vernetzung ist weniger dicht und wird durch schlanke, wohldefinierte Interfaces transparent.

- **Service Oriented Architecture** (SOA) ist charakterisiert durch eine zustandslose Message-orientierte Interaktion zwischen den einzelnen Softwaremodulen (den Services). Gemeinsam genutzte Daten und Modelle werden als eigene Services abstrahiert und somit die Kopplung der Module noch weiter aufgelöst.

Aufgrund des hohen Grades an Overhead für die Kommunikation für SOA, der hohen Granularität in der DOA, welche das Design auf Systemebene erschwert, und den Anforderungen aus den in dieser Arbeit gegebenen Anwendungen, scheint die **Component Based Architecture** der beste Kompromiss für die Infrastruktur einer Robotikanwendung zu sein.
In der Praxis hat diese Betrachtungsweise zur Entwicklung von Robotik Middleware-Frameworks geführt, wobei diese von Canas et al. [Canas et al., 2007] in zwei Klassen unterteilt werden: die einen ohne Beachtung einer kognitiven Architektur und die, bei denen funktionale Aspekte bereits vorgegeben sind. Dies entspricht der Unterscheidung in domänenspezifische Architekturen und domänenunspezifische [Liu, 2005], wobei die spezifischen als effizienter, aber auch restriktiver für den Entwickler eingestuft werden, was hier explizit gewünscht wird. Canas et al. sehen die Vorteile einer Middleware mit funktionaler Strukturierung darin, dass der Designprozess für neue Systeme dadurch gelenkt und somit vereinfacht wird. **Daher wird mit dieser Arbeit auch versucht, in einer Middleware ohne strukturelle Einschränkungen (MIRA) durch Definition einer logischen Sicht den Rahmen für die Entwicklung von Gesamtanwendungen vorzugeben.**

Um eine **funktionale Strukturierung** zu begründen, sollen existierende Organisationsformen innerhalb von Steuerarchitekturen für mobile Roboter berücksichtigt werden (siehe Abb. 4.1). Nach Arkin [Arkin, 1995] lassen sich kognitive Architekturen in **Bottom-up** (Datengetriebene) und **Top-down** (Wissensgetriebene) unterscheiden. Außerdem stellt er **„Unified Field Theory Approaches“** und **„Mix-and-Match Approaches“** gegenüber. Erstere verfolgen dabei die Idee, das Gesamtsystem aus verschalteten Modulen ein und desselben Typs aufzubauen und die gewünschte kognitive Leistung als emergentes Verhalten zu erhalten. Die Philosophie des zweiten Ansatzes lautet dagegen sinngemäß: „Nimm funktionierende verschiedenartige Verfahren

und Methoden für Teilleistungen und bastele daraus eine Robotersteuerung“.

Diese zwei Klassen finden sich in einer feineren Unterteilung in reaktive, deliberative, hybride und behavior-based Paradigmen für kognitive Architekturen wieder [Canas et al., 2007, Matarić, 1997, Matarić, 2001]. Hinzu kommen noch die Agenten-basierten Architekturen.

Im Anhang A 4 findet sich eine genauere Erläuterung zu den Spielarten und der Entwicklungsgeschichte der kognitiven Architekturen.

Für den vorliegenden Entwurf einer Architektur wurde ein **hybrider Ansatz** gewählt, da er Schwächen der deliberativen (schlechte Skalierbarkeit und Echtzeitfähigkeit), sowie der reaktiven und behavior-based Ansätze (schlechte Erweiterbarkeit, Beschränkung auf Bewegungssteuerung) überwindet.

### 4.2.1 Hybride Architektur

Da sich hybride Ansätze sowohl reaktiver als auch deliberativer Ideen bedienen, resultieren daraus meist mehrschichtige Architekturen mit zwei oder drei Ebenen, wobei oben langsamere, planende und unten reaktive, echtzeitfähige Methoden eingesetzt werden.

Am bekanntesten sind dabei die **3T Architekture** [Peter Bonasso et al., 1997] (T steht dabei für Engl. tier = Schicht, Etage) und die **Servo, Subsumption, Symbolic Architektur** (SSS) [Connell, 1992]. Auch bei Göller [Göller et al., 2009, Göller, 2013] finden sich drei Schichten; „reaktive“ , „taktische“ und „strategische“ Layer bei der Realisierung einer Kontrollarchitektur für einen autonomen Einkaufswagen. Auch speziell für den Einsatz als kommunikativer Companion Roboter wurde eine hybride Drei-Ebenen-Architektur erfolgreich eingesetzt [Kleinehagenbrock, 2004]. Weiterhin stellen Duffy et al. [Duffy et al., 2005] die sogenannte **Social Robot Architecture** vor, welche ein Schichtenmodell umsetzt, das sich in physical, behavioral, deliberative und social level unterteilt. In der deliberativen Schicht wird dabei die Kontrolle in ein Multi-Agenten-System verteilt.

Es zeigt sich hiermit, dass hybride Architekturen in der Mix-and-Match-Philosophie recht flexibel sind und auch Raum bieten, um weiter gefasste Aufgaben, wie die Interaktion mit Menschen, realisieren zu können.

### 4.2.2 Navigationsarchitektur am Fachgebiet NI&KR

Die Hauptaufgabe der zuvor genannten Architekturen, zielt im Wesentlichen immer auf die Navigation und Bewegungssteuerung eines mobilen Roboters. Für die Umsetzung eines komplexen Assistenzroboters ist dies aber nur eine Teilaufgabe, welche für sich betrachtet bereits deliberative (Pfadplaner) und reaktive Schichten (z.B. Hindernisvermeidung) enthält. Um zu motivieren, warum in der vorgestellten Applikationsarchitektur die klassischen drei Schichten nicht direkt wiederzufinden sind, soll die Realisierung der Navigationsfähigkeiten an dieser Stelle kurz geschildert werden.

Im Laufe der Entwicklung von Verfahren zur Bewegungssteuerung hat sich unabhängig von der Architektursicht aus rein funktionalen Betrachtungen eine Methodik entwickelt, welche nur für die Auswahl des aktuellen Steuerkommandos verschiedene Einflussfaktoren, so genannte Objectives, berücksichtigt und im Raum der möglichen Bewegungskommandos eine Optimierung durchführt, um möglichst viele dieser Objectives zu erfüllen [Weinrich, 2015]. Da bei der Optimierung nur der wirklich erreichbare Ausschnitt aus dem Aktionsraum durchsucht wird, heißt dieses Verfahren **Dynamic Window Approach** (DWA) [Fox et al., 1997].

In den bearbeiteten Projekten CompanionAble, SERROGA und ROREAS kam eine Variante des DWA zum Einsatz, weshalb sich ein Großteil der Aufgaben einer klassischen Steuerarchitektur letztendlich in einem einzigen Softwaremodul dem **Navigator** wiederfinden [Gross et al., 2011a,Weinrich, 2015]. In diesem Modul finden sich auch wesentliche Gedanken der behavior-basierten Ansätze wieder. Wie bei diesen kann durch die geeignete Aktivierung und Deaktivierung verschiedener Kombinationen von Objectives (Realisierungen der Behaviors) durch übergeordnete Module unterschiedliches Navigationsverhalten realisiert werden.

Die Architektur dieser Arbeit konzentriert sich daher im Wesentlichen auf die Organisation der Ablaufsteuerung im Kontext des Einsatzszenarios, sowie auf die Gestaltung der Interaktion mit dem Nutzer. Navigationsaufgaben werden als funktional abgeschlossene Blöcke (Navigationsbehaviors) betrachtet (siehe Abschn. 4.4.3) und der **Navigator** findet sich als Basisfunktionalität in der Skill Layer wieder (siehe Abschn. 4.4.2).

### 4.2.3 Fazit zum State of the Art

Aus der Bestandsaufnahme zum State of the Art bezüglich der Ausgestaltung von Steuerarchitekturen für mobile Roboter kristallisierte sich heraus, dass ein hybrider Ansatz vielversprechend ist, wobei darauf zu achten ist, keine zentrale Umweltrepräsentation einzuführen (Flaschenhals). Vielmehr sollte, ähnlich den behavior-basierten Ansätzen, eine dezentrale Modellierung erfolgen. Ein Mix-and-Match-Ansatz scheint für die Realisierung eines solch komplexen Systems, wie es ein sozialer Assistenzroboter ist, aufgrund der Entwicklungspraxis am Fachgebiet NI&KR unerlässlich. Nur dadurch kann vom reichen Bestand an Methoden für Navigation und Wahrnehmungsleistungen profitiert werden.

## 4.3 Randbedingungen


Die konkrete Ausprägung der vorgeschlagenen Applikationsarchitektur geht eng mit der zugrundeliegenden Middleware, der eingesetzten Hardwareplattform sowie dem angestrebten Einsatzzweck einher. Diese Arbeit setzt auf der Nutzung der Middleware MIRA (MIddleware for Robotic Applications) [Einhorn et al., 2012] auf, welche im Laufe der Bearbeitung verschiedener Robotikprojekte am Fachgebiet NI&KR sowie von der Firma MetraLabs GmbH entwickelt wurde. Durch die Eigenentwicklung konnte dabei auf Erfahrungen im Umgang mit der zuvor genutzten Software eingegangen und die unter Abschn. 4.1 aufgeführten Anforderungen berücksichtigt werden. Außerdem wurde durch die Beteiligung der Firma MetraLabs auch auf Aspekte Wert gelegt, die für kommerzielle Robotiklösungen bedeutsam sind.
Das MIRA Framework unterstützt das Konzept einer Component Based Architecture (siehe Abschn. 4.2). Folgende Eigenschaften und Möglichkeiten sind durch die Verwendung von MIRA vorhanden:

- Plattform übergreifend (Linux, Windows)
- Transparente Kommunikation zwischen Software-Modulen mittels Publish / Subscribe Pattern über sogenannte Channels
- Remote Procedure Call (RPC) Mechanismus zum Austausch von Steu-

erkommandos zwischen den Modulen
- Serialisierungsframework (einfache Konfiguration der Komponenten)
- Realisierung von Prozess- und Threadentkopplung aller Module
- Konfiguration einer Gesamtapplikation und Kopplung der Module aus XML-Beschreibungen

Bezüglich der Hardware zielt die Applikationsarchitektur auf mobile Serviceroboter für die Interaktion mit Menschen ab. Obwohl als Demonstratoren nur Roboter ohne Manipulationsmöglichkeiten zur Verfügung stehen, ist das Konzept nicht auf solche beschränkt. An gegebener Stelle wird noch einmal darauf hingewiesen, wo Manipulationsfähigkeiten berücksichtigt werden können. Da in der Applikationsarchitektur explizit auch auf eine Ablaufsteuerung in Form eines Dialogmanagers abgezielt wird, ist als Einsatzzweck eindeutig die Interaktion mit und die Serviceerbringung am Menschen gegeben. Hierbei bleibt aber offen, ob ein oder mehrere Nutzer in längerer oder kürzerer Interaktion treten und ob eine private oder öffentliche Einsatzumgebung angestrebt wird.

## 4.4 Schichtweise Architektur

Um den vollen Funktionsumfang eines autonomen Serviceroboters zu gewährleisten, müssen zahlreiche Softwaremodule zusammenspielen. Die Middleware und das Softwareframework ermöglichen es dabei, die Kapselung von Funktionen in lose gekoppelte Module vorzunehmen, die allerdings alle einheitliche Methoden zur Kommunikation untereinander verwenden und äußerlich keine Unterscheidung bzgl. ihrer Rolle im Gesamtsystem aufweisen. Dennoch lassen sich diese Module, obwohl mit denselben Mitteln realisiert, nach Abstraktionsebene, Aufgabenbereich und Hardwarenähe logisch kategorisieren und gruppieren. In einer Implementierung spiegelt sich diese Aufteilung lediglich in der modularen Konfiguration einer Roboterapplikation wider, die entsprechende Unterteilungen vorsehen kann, um den Aspekten der Wiederverwendbarkeit gerecht zu werden. Beginnend bei der eigentlichen Hardwareansteuerung mittels Gerätetreibern werden daher verschiedene Schichten (Layers) identifiziert, die auch eine Art Kontrollhierarchie darstellen. Module höherer Schichten koordinieren oder steuern typischerweise Module niedrigerer Schichten. Abb. 4.2 gibt einen Überblick zu der vorgeschlagenen Einteilung. Im Folgenden werden

die Schichten näher erläutert.

### 4.4.1 Hardware Layer

Der Hardware Layer gehören einerseits die Module zur Erfassung der verschiedenen Sensorsignale (Laserscanner, Kameras, Ultraschallsensoren, Touchsensoren, RFID Antenne) und andererseits die Treiber für die Aktoren (Robotertreiber, Augendisplays, gegebenenfalls Manipulatoren) an. Diese Module laufen im Allgemeinen parallel und stellen die Sensordaten für die Weiterverarbeitung zur Verfügung bzw. werden von höheren Schichten per RPC[1] gesteuert und sorgen für die Realisierung der gewünschten Wirkung in der Umwelt.

### 4.4.2 Skill Layer

In der Skill Layer (gelb in Abb. 4.2) finden sich Module, die einzelne Erkennungsleistungen oder Bewegungssteuerung realisieren. Diese so genannten **Skills** arbeiten alle parallel und relativ unabhängig voneinander. Im Wesentlichen werden von Modulen der Skill Layer die Sensordaten aus der Hardware Layer verarbeitet und Steuerkommandos für diese generiert. Es kann auch zwischen den Modulen in der Skill Layer eine sequenzielle Abhängigkeit bestehen, indem z.B. ein Modul Zwischenergebnisse produziert, die von einem weiteren Modul der Skill Layer weiterverarbeitet werden. Auch servicespezifische Module, die im Hintergrund ihren Dienst tun und Sensordaten verarbeiten[2], zählen zur Skill Layer. Die Module der Skill Layer haben jeweils konfliktfreien Zugriff auf Ressourcen in der Hardware Layer und sollten keine weitere Koordination benötigen. Durch das Middleware-Framework können Datenströme von mehreren Senken gleichzeitig gelesen, aber nur von einem Modul geschrieben werden.

[1] Remote Procedure Call

[2] für einen Kalenderservice beispielsweise die Kalenderdatenbank oder für die Vitalparametermessung die Pulsratenextraktion aus dem Videobild

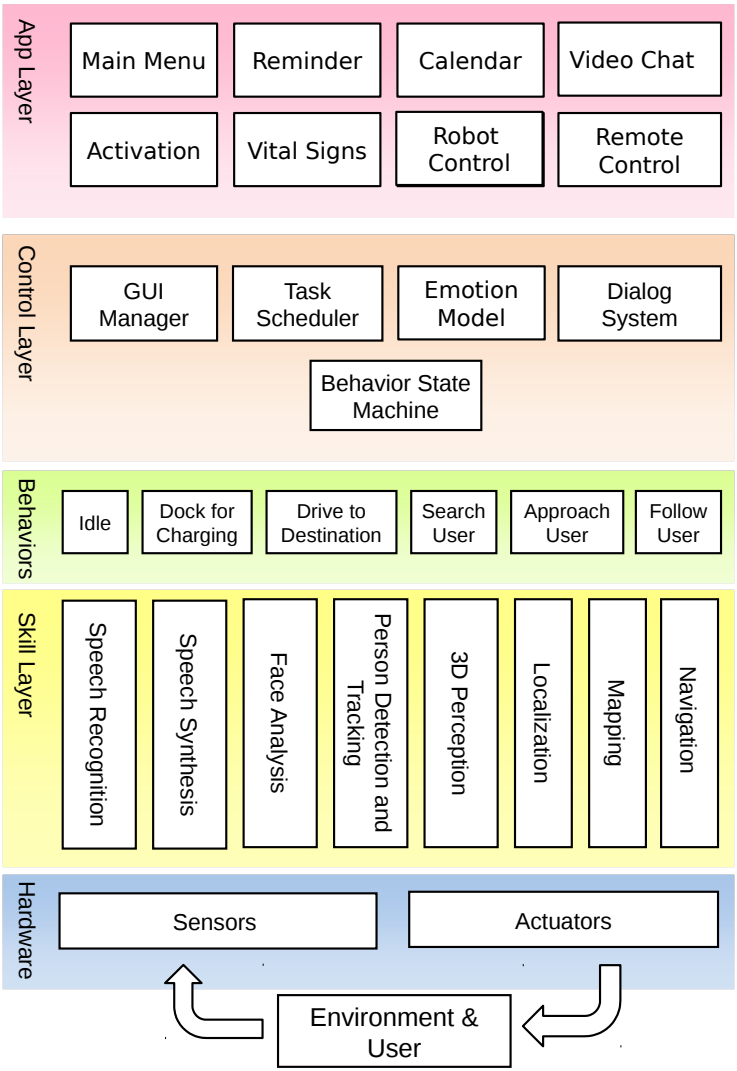



**Abbildung 4.2:** Schichtensicht der Applikationsarchitektur

### 4.4.3 Behavior Layer

In der Behavior Layer (grün in Abb. 4.2) finden sich Module, die grundlegende wiederkehrende Verhaltensweisen eines Roboters implementieren. Typischerweise werden für deren Ausführung Funktionalitäten der Skill Layer kombiniert.Dadurch kommt es letztendlich zu einem Ressourcenkonflikt, wenn verschiedene Module auf gleiche Funktionen wie beispielsweise die Navigationsfähigkeiten des Roboters zugreifen wollen. Die Module der Behavior Layer, kurz **Behaviors**, werden daher nur exklusiv aktiviert. Im Gegensatz zu den parallel arbeitenden Modulen der Skill Layer muss bei den Behaviors von übergeordneter Stelle gewährleistet werden, dass immer nur eines zur gleichen Zeit aktiv ist. Eine Überlagerung der Behaviors, wie in einer behavior-basierten Architektur, ist nicht vorgesehen.

Beispiele für Navigationsbehaviors wären das Aufladen auf der Ladestation, das Suchen eines Nutzers oder einfach das autonome Anfahren eines Zielpunktes.

Falls der Roboter neben der Fähigkeit zu fahren auch über Manipulatoren verfügt, welche potentiell unabhängig von der Bewegung steuerbar sind, so ist es empfehlenswert, dennoch Behaviors zu implementieren, in denen diese Ressourcen koordiniert angesteuert werden. Beispielsweise könnte das Aufnehmen eines Gegenstands ein Behavior bilden, wobei die Roboterposition und die Manipulatorkonfiguration gesteuert werden. Dieses Beispiel zeigt auch, dass Behaviors typischerweise als Zustandsautomaten zu verstehen sind, die in den entsprechenden Zuständen Skills ansteuern und auf Events aus der Skill Layer mit Zustandsübergängen reagieren.

Die Behaviors bilden für die darüberliegende Schicht auch eine gewisse Abstraktion von Basisfunktionalitäten. Durch ein relativ schmales Interface, welches im Allgemeinen eine **Start** und eine **Stop** Methode, sowie einige Events, die den internen Zustand der Behaviors beschreiben, umfasst, ist es möglich, die konkrete Realisierung der Funktionalität für höhere Schichten transparent auszutauschen. Beispielsweise kann ein Behavior zur Nutzersuche mit verschiedenen Personendetektionsverfahren funktionieren oder verschiedene Verfahren zur Bewegungsplanung einsetzen, ohne dass Änderungen an höheren Schichten nötig werden.

### 4.4.4 App Layer

Für den Fall, dass ein Roboter dynamisch um weitere Servicefunktionalitäten erweitert werden können soll, wie es im SERROGA Projekt der Fall ist, bietet es sich an, neben einer festen Infrastruktur für die Ablaufsteuerung, welche in der Control Layer angesiedelt ist, eine separate Schicht vorzusehen, in der die einzelnen Services implementiert sind (pink in Abb. 4.2 ganz oben). Diese greifen auf die Funktionalitäten der Control Layer zu, welche somit als eine API (Application Programming Interface) fungiert, und spezifizieren im Wesentlichen die Interaktionen mit dem Nutzer. Die einzelnen Services wären dann als eine Art App zu verstehen, die dynamisch zu einer laufenden Roboterapplikation hinzugefügt werden können. Im Weiteren wird daher für die Elemente dieser Schicht auch der Begriff **Serviceapplikation** oder **Service-App** genutzt werden. Die softwaretechnische Umsetzung der Serviceapplikationen kann dabei entweder als Bestandteil einer Gesamtapplikation oder als lose gekoppelte eigenständige Prozesse geschehen.

### 4.4.5 Control Layer/ Dialogsystem

Nachdem in den unteren Schichten bisher mehr oder weniger unabhängige Teilfunktionen nebeneinander existieren, finden sich in der Control Layer Elemente, die das Gesamtverhalten des Robotersystems bestimmen. Von hier aus werden die Behaviors koordiniert und die Interaktion mit dem Nutzer wird in dieser Schicht verwaltet. Weiterhin sollen hier auch die Selbsterhaltung und Sicherheitsaspekte Beachtung finden. Die Control Layer (siehe Abb. 4.2) implementiert mit einer nicht mehr unabhängigen Menge an Softwaremodulen die eigentliche Ablaufsteuerung. Die Module der Control Layer arbeiten eng miteinander zusammen, wobei verschiedene Möglichkeiten der Realisierung denkbar sind.
Im Rahmen des SERROGA Systems wurde die Umsetzung der Ablaufsteuerung zweigeteilt konzipiert. Alle sicherheitsrelevanten Aufgaben sowie die Gewährleistung der Selbsterhaltung werden in einem Zustandsautomaten gekapselt. Diese **Behavior State Machine** genannte Komponente ist daher beweisbar robust und übernimmt die Koordination der exklusiv zu aktivierenden Behaviors. In Abschn. 5.1 wird die Behavior State Machine näher erläutert.

Die Möglichkeiten zur Dialogführung mit dem Nutzer werden unabhängig von den konkreten Services durch eine Gruppe von Modulen, die das **Dialogsystem** bilden, bereitgestellt. Dieses wird in Kapitel 6 ausführlich behandelt. Wie durch die State Machine die exklusive Kontrolle der Roboterressourcen durch die Behaviors realisiert wird, so koordiniert das Dialogsystem den exklusiven Zugriff der Service-Apps auf das Nutzerinterface und in diesem Sinne auch auf den Nutzer. Durch das Dialogsystem wird die multimodale Ein- und Ausgabe auf die unabhängigen, parallel laufenden Service-Apps gemultiplext.
In der Control Layer lassen sich noch weitere zentrale Module implementieren, die zur Realisierung eines Charakters oder zur Koordination von längerfristigen Aufgaben des Roboters dienen. Beispielsweise könnten ein Emotionsmodell und ein Taskscheduler hinzugefügt werden, wie es in den Kapiteln 5 und 8 beschrieben ist.

### 4.4.6 Diskussion

Bezugnehmend auf die oben aufgeführten Anforderungen werden die Aspekte Effizienz, Robustheit / Ausfallsicherheit und Modularität der Software weitestgehend durch die Middleware MIRA erfüllt. Die hier eingeführte logische Unterteilung der Module ermöglicht es, Teile der Gesamtapplikation in Konfigurationsdateien zu kapseln, sodass eine Abstraktion von Hardwarespezifika bzw. Schnittstellenabstraktion zu Basisfunktionen des Roboters realisiert werden kann. Ebenso bleiben Austauschbarkeit und Wiederverwendbarkeit der Softwaremodule in den unteren Schichten gegeben, wenn zur Control Layer durch die Behaviors ein schmales Interface besteht. Letztendlich lässt sich aus Service-Sicht und Control Layer-Sicht sogar der ganze Roboter substituieren, solange die benötigte Menge an Behaviors auf einer anderen Plattform oder gar in einem Simulator evtl. auch in komplett anderer Ausprägung zur Verfügung stehen. Die vorgeschlagene Trennung in exklusive und parallel laufende Teile erleichtert die Ressourcenverwaltung, welche im Detail in der Control Layer realisiert werden muss (siehe Abschn. 5.1). Ebenso hängt die schnelle Applikationsentwicklung und leichte Erweiterbarkeit von der konkreten Umsetzung der Control Layer ab, welche im folgenden Kapitel 5 genauer betrachtet wird.

## 4.5 Zusammenfassung

Ausgehend von einer Fortsetzung der Anforderungsanalyse in Kapitel 2 stellte sich in diesem Kapitel heraus, dass klassische Steuerarchitekturen für mobile Roboter nur einen geringen Teil der Anforderungen abdecken. Als Grundlage für einen geordneten Anwendungsentwicklungsprozess wurde daher eine Softwarearchitektur zur logischen Strukturierung von Softwarekomponenten vorgeschlagen, welche beispielhaft mit der Middleware MIRA umgesetzt wurde.

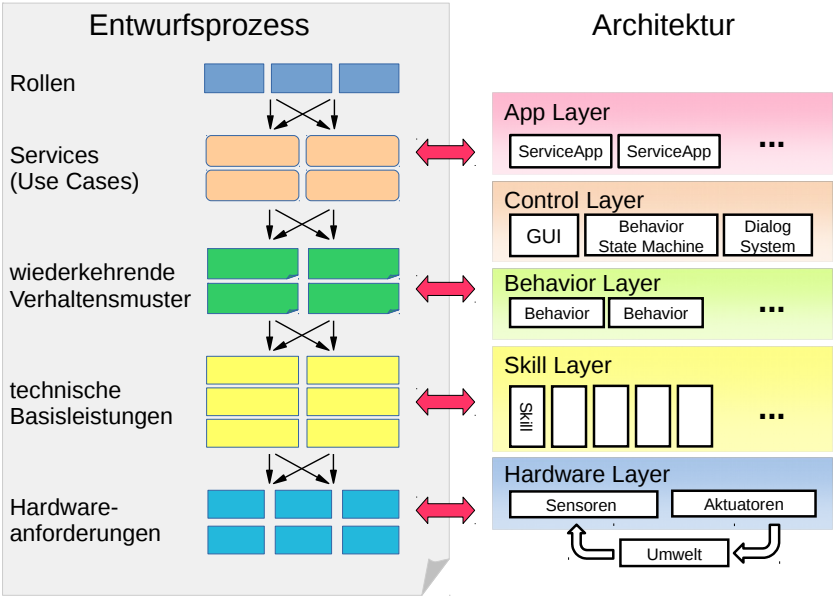


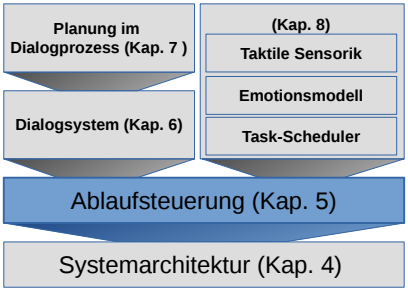

**Abbildung 4.3:** Korrespondenz der Analyseschritte im Entwurfsprozess zu den Elementen der Softwarearchitektur

Die Schritte des Analyseprozesses aus Abb. 2.1 finden sich in der Architektur in ähnlicher Form wieder (siehe Abb. 4.3).

Die technischen Basisfunktionalitäten sind größtenteils in der Skill Layer realisiert. Die meisten wiederkehrenden Verhaltensmuster ließen sich auf die exklusiv aktivierten Navigationsbehaviors in der Behavior Layer abbilden. Hinzu-

gekommen ist die Control Layer, in welcher die entsprechend weiter untersetzte Ablaufsteuerung die Koordination der Behaviors gemäß der Vorgaben der in der darüber liegenden App Layer realisierten Anwendungsfälle, vornimmt. Ebenfalls wird die Interaktion mit dem Nutzer zentral aus der Control Layer organisiert und lediglich durch die Service-Apps konfiguriert. Dadurch lässt sich sicherstellen, dass auch bei Hinzunahme weiterer Services der Gesamtablauf keinen Schaden nimmt und der Roboter in allen Anwendungsfällen eine einheitliche Erscheinungsform bzgl. der Interaktion aufweist.

# 5




# Komponenten der Ablaufsteuerung (Control Layer)

Nachdem der prinzipielle Gedanke hinter der Schichtenunterteilung der Applikationsarchitektur verdeutlicht wurde, folgt eine detailliertere Darstellung des Konzepts der entwickelten Methoden für die Umsetzung einer Ablaufsteuerung in der Control Layer. Bei der Darstellung der schrittweisen Entwicklung der Control Layer wird Bezug auf die Anforderungen aus Abschnitt 4.1 genommen und es wird versucht, das Zusammenwirken der in dieser Arbeit entwickelten Komponenten zu verdeutlichen. Der modulare Aufbau der Control Layer (siehe Abb. 5.1) ermöglicht es, verschiedene Aspekte der Interaktionsgestaltung individuell hinzuzufügen. Es lassen sich somit sehr einfache

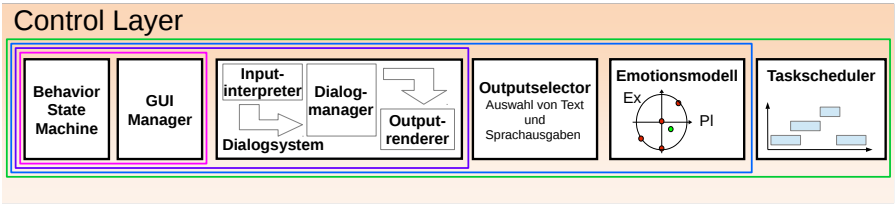



**Abbildung 5.1:** Komponenten der Control Layer, welche in verschiedenen Kombinationen zur Realisierung von Robotikanwendungen genutzt werden können. In rosa die minimale Ausbaustufe, über lila und blau umrahmt bis zur kompletten Ausstattung in grün.

GUI-basierte Anwendungen ohne zentrale Koordination der Nutzerkommunikation realisieren, welche sich durch Hinzunahme eines Dialogmanagers vereinheitlichen und koordinieren lassen. Dadurch wird es ermöglicht, einen geschlossenen Eindruck vom Roboter als Interaktionspartner zu erzielen. Durch weiteres Hinzufügen eines Emotionsmodells, welches die nun zentral gesteuerten Ein- und Ausgaben moduliert, wird es möglich, die Erscheinung des Roboters noch authentischer und abwechslungsreicher zu gestalten. Ebenso wird der Nutzungskomfort erhöht, wenn zusätzlich durch die Verwendung eines Taskschedulers die rein event- oder zeitgetriggerte Aktivität des Roboters an die Verfügbarkeit und Aktivität des Nutzers angepasst wird.
Im Rahmen des Szenarios Gesundheitsassistent wurden bis auf den Taskscheduler alle Module praktisch realisiert und während der Nutzertests[1] eingesetzt (Ausbaustufe blau). Der Taskscheduler wurde im Rahmen des CompanionAble Projekts (siehe Abschn. 1.1.2) realisiert, dabei allerdings ohne die Komponente Emotionsmodell eingesetzt.
Die einzelnen Module werden im Folgenden näher erläutert, wobei auf die Umsetzung des Dialogsystems nochmals gesondert in Kap. 6 eingegangen werden wird.

## 5.1 Behavior State Machine

Als zentrales Element der Ablaufsteuerung in der Control Layer findet sich die so genannte Behavior State Machine. Diese ist als Zustandsautomat implementiert (siehe Abb. 5.2) und gewährleistet die Selbsterhaltungsfunktion des Roboters sowie die sichere Navigation. Hierzu überwacht der Zustandsautomat den Ladezustand der Batterie und sorgt gegebenenfalls für ein autonomes Andocken an die Ladestation. Somit soll ein autonomer 24-Stunden Betrieb ermöglicht werden.
Weiterhin existiert für jedes Behavior[2] der Behavior Layer (siehe Abschn. 4.4.3) ein eigener Zustand, bei dessen Betreten das Behavior aktiviert bzw. beim Verlassen deaktiviert wird. Diese exklusive Zuordnung von Behaviors zu Zuständen realisiert den größten Teil der Ressourcenverwaltung bzgl. der Na-

[1] Details dazu in Kap. 9
[2] Implementierungen der in Abschn. 2.5 identifizierten Navigationsverhalten

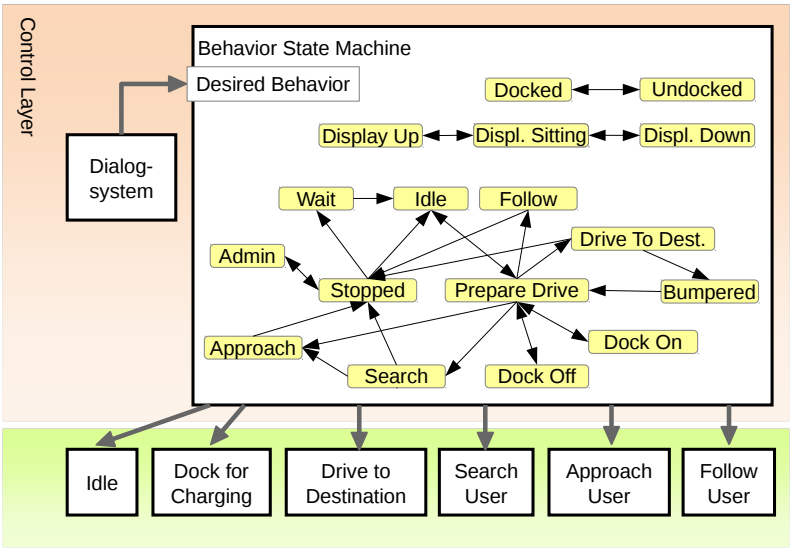



**Abbildung 5.2:** Koordination der Behaviors aus der Behavior State Machine. Für jedes Navigationsbehavior existiert ein Zustand, in welchem dieses exklusiv aktiviert ist. Der Zustandsautomat ist nur exemplarisch verschaltet.

vigationsfähigkeiten und sorgt für die exklusive Aktivierung von Navigationsbehaviors, wie es in der Architekturbeschreibung gefordert wurde (vergleiche 4.4.3).

Die Realisierung der Behavior State Machine als Zustandsautomat gewährleistet eine gewisse Nachvollziehbarkeit der Abläufe und bietet zudem die Möglichkeit eines formalen Tests auf Korrektheit der Funktion. Damit können auch die Sicherheitsaspekte gebührend berücksichtigt werden, da jegliche Form der physischen Bewegung des Roboters allein durch den Zustandsautomaten entschieden wird.

Da in der Behavior State Machine bereits eine gewisse Eigenintelligenz und Ablaufsteuerung vorhanden ist, und diese auch als zentrale Kontrollinstanz in der gesamten Architektur fungiert, wird eine Möglichkeit benötigt, das Verhalten auch entsprechend der Wünsche des Nutzers oder der Service-Apps zu beeinflussen. Hierfür wird als Interface das so genannte **Desired Beha-**

**vior** vorgesehen. Dieses wird von den Service-Apps situationsabhängig über das Dialogsystem vorgegeben[3]. Der Zustandsautomat ist so verschaltet, dass von jedem Zustand, in dem es möglich ist, versucht wird, in denjenigen zu gelangen, der durch das Desired Behavior vorgegeben wird. Falls das nicht möglich ist, wird dies lediglich durch Statusnachrichten signalisiert, um zu ermöglichen, dass die Situation entsprechend behandelt wird[4].

## 5.2 GUI-Manager

Nachdem die grundlegenden Navigationsfunktionen des Roboters durch die Behavior State Machine steuerbar sind, soll nun auch die Interaktion mit dem Nutzer ermöglicht werden. Bei den im Rahmen dieser Arbeit genutzten Robotern findet der Hauptteil der Interaktion über ein grafisches Benutzerinterface (GUI) statt, welches dem Interaktionspartner typischerweise über ein Touchdisplay präsentiert wird.

Eine Serviceapplikation der App Layer kann somit im einfachsten Falle neben der Implementation der eigentlichen Funktion ein oder mehrere GUI-Fenster (Widgets) enthalten, über welche Nutzerkommandos erfasst und Rückmeldungen des Systems ausgegeben werden. Auf diese Weise sind bereits einfache Roboterapplikationen realisierbar[5] (siehe Abb. 5.3).

Sobald der Roboter mehrere Servicefunktionen erfüllen soll, wird für diese der Bildschirm allerdings zu einer nicht exklusiv verfügbaren Ressource. Um den Nutzer nicht zu verwirren und um ihm ein der Dialogsituation angemessenes grafisches Interface zuzuführen, ist es daher nötig, die unabhängigen Fenster der Service-Apps prozessübergreifend zu koordinieren und somit Konflikte bei der Nutzung des Bildschirms aufzulösen. Die zur Verfügung stehende Bildschirmfläche kann dabei entweder aufgeteilt werden, um gleichzeitig die benötigten Fenster anzuzeigen, oder es wird zwischen den Fenstern der Serviceapplikationen hin- und hergeschaltet.

Da das Ziel darin besteht, den Roboter als eigenständigen, einheitlichen Dia-

[3] in der einfachsten Ausbaustufe auch direkt aus den Service-Apps

[4] Beispielsweise kann der Roboter nicht zu einem vom Nutzer vorgegebenen Ziel fahren, wenn der Akku leer ist und der Roboter auf der Ladestation steht.

[5] Die Behavior State Machine kann dazu direkt aus den Service-Apps oder deren GUI-Widgets gesteuert werden.

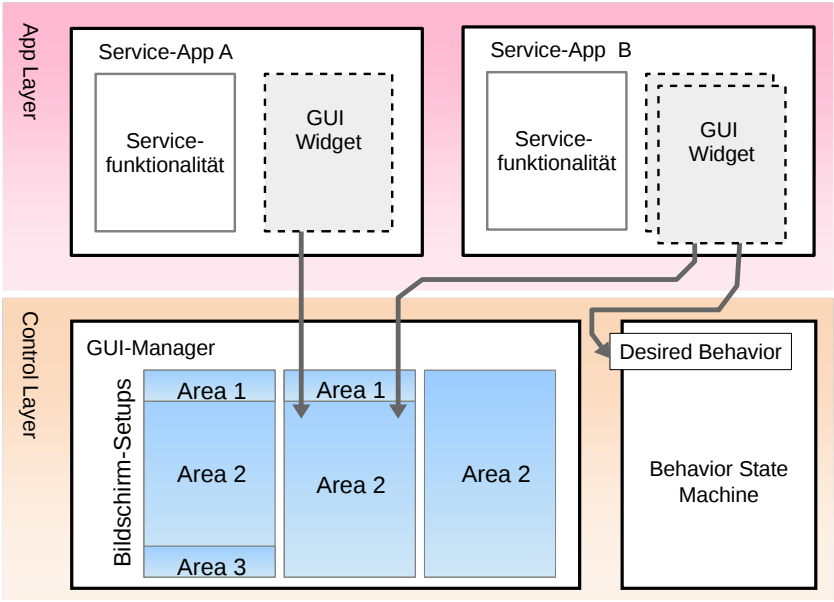



**Abbildung 5.3:** Organisation von Bildschirmausgaben durch den GUI-Manager, Aufteilung des Bildschirms in Areas (Bereiche), in denen exklusiv Widgets der Service-Apps dargestellt werden. Durch Umschalten der Widgets wird eine einfache Form der Koordination von Service-Apps ermöglicht. Die Navigationsbehaviors können im einfachsten Fall auch direkt aus der GUI Implementierung getriggert werden.

logpartner erscheinen zu lassen, scheint es nötig, dass jeweils nur die aktuell thematisierten Steuerelemente und Ausgaben sichtbar sind (Umschaltung zwischen den Fenstern der Service-Apps). Außerdem sollen alle Serviceapplikationen im gleichen Stil auf dem Bildschirm erscheinen, um die Grenzen zwischen den Services für den Nutzer unsichtbar zu machen. Im Sinne eines Dialogs, in dem die Initiative von beiden Partnern ausgehen kann, muss außerdem eine Eingabemöglichkeit vorgesehen werden, welche neben den roboter-initiierten Ausgaben und Eingabeaufforderungen immer sichtbar ist. Dies lässt sich sinnvollerweise über ein Menü realisieren, wie man es von grafischen Betriebssystemen auf PC oder Mobiltelefonen kennt.

Die Aufgaben eines GUI-Managers sind daher:

- Aufteilung des Bildschirmplatzes
- Umschaltung zwischen Fenstern
- Definition von Bildschirmposition und Fenstergröße
- Bereitstellung einheitlicher Bildschirmelemente und Designs

Mit Hilfe eines zentralen GUI-Managers wurde für den SERROGA Roboter konkret eine Bildschirmaufteilung in einen Header mit immer präsentem Zugang zu zentralen Funktionen, einen Statusanzeigebereich[6] und einen Hauptteil, in dem die Konversation der Service-Apps mit dem Nutzer stattfindet, realisiert. Hinzu kommt eine immer sichtbare Fußleiste, in der ein Menü- und ein Zurück-Button platziert wurden. Zusätzlich kann, wie von Mobiltelefonen gewohnt, eine virtuelle Tastatur eingeblendet werden, falls es für die Interaktion erforderlich ist.

## 5.3 Dialogsystem

Wie bereits erwähnt, ergänzt das Dialogsystem die Behavior State Machine bei der Ressourcenverwaltung auf Seite der Nutzerinterfaces. Wie der Zugriff auf die Bewegungsfähigkeiten des Roboters zwischen den Navigationsbehaviors umgeschaltet wird, so schaltet der Dialogmanager das Nutzerinterface zwischen den Service-Apps um. Zu diesem Zweck definieren die Service-Apps ihre Interaktionsmöglichkeiten mit dem Nutzer in Form von Teildialogen, so genannten Dialog-Frames (siehe Abb. 5.4 App Layer). Der Dialogmanager führt diese Teildialoge bei Bedarf Schritt für Schritt aus und kann auch zwischen Teildialogen verschiedener Service-Apps wechseln. Unterbrochene Dialoge können dabei bei Bedarf später fortgesetzt werden.
Das Dialogsystem soll es weiterhin ermöglichen, neben der GUI weitere Ein- und Aus-Modalitäten einzubinden, ohne dass dies explizit in jeder Service-App implementiert werden muss. Das Dialogsystem zentralisiert somit alle Kommunikation mit dem Nutzer und bildet daher auch eine zentrale Stelle für die Implementierung der Verfahren zur Nutzeradaption, die in Kap. 3 identifiziert wurden.

[6] zur Anzeige von Datum, Uhrzeit und Wettersituation, sowie weiteren Icons als Statusmeldung für die Service-Apps

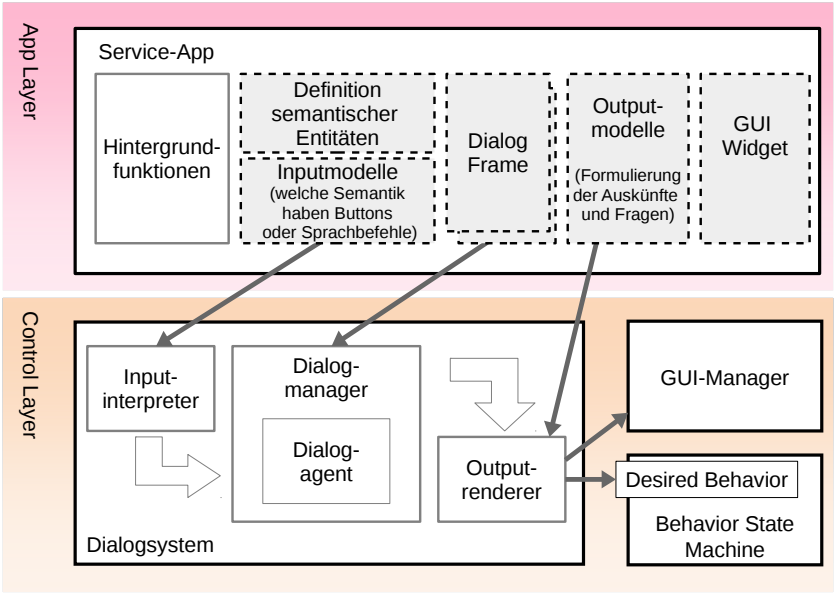



**Abbildung 5.4:** Dialogsystem als zentrale Verwaltung des Konversationszustands und Koordinator für die Interaktionen mit dem Nutzer. Serviceapplikationen definieren Teildialoge (Dialog-Frames), welche vom Dialogmanager abgearbeitet werden. Das Dialogsystem kann dabei zwischen verschiedenen Dialog-Frames koordiniert wechseln. Bildschirmausgaben und Navigationsbehaviors werden über die Aktionen der Dialog-Frames bestimmt.

Der Aufbau des Dialogsystems entspricht im Wesentlichen der klassischen Dreiteilung in Inputfusion (Inputinterpreter), Dialogsteuerung (Dialogmanager) und Outputrenderer für die Darstellung und Ausführung multimodaler Systemaktionen[7]. Dementsprechend finden sich Bestandteile der Definition eines Dialog-Frames für alle diese Komponenten in den Konfigurationen der Service-Apps, wie es in Abb. 5.4 zu sehen ist.
Kap. 6 wird näher auf die Einzelheiten der Realisierung des Dialogsystems eingehen.
Eine Zentralisierung der Interaktionsintelligenz und -ausführung ermöglicht

[7] Systemaktionen umfassen einerseits die an den Nutzer gerichtete Botschaft in Form von Text und Sprachausgaben, andererseits auch die Ansteuerung der GUI und der Navigationsbehaviors über die Behavior State Machine und weitere servicespezifische Aktivitäten.

es auch, dass die Ein- und Ausgaben zur Formung eines Charakters für den Roboter serviceübergreifend moduliert und vereinheitlicht werden können. Zu diesem Zweck wurden der Control Layer zwei weitere Komponenten, das Emotionsmodell und der Outputselector, hinzugefügt, welche eine Roboterapplikation optional ergänzen.

## 5.4 Emotionsmodell und Outputselector

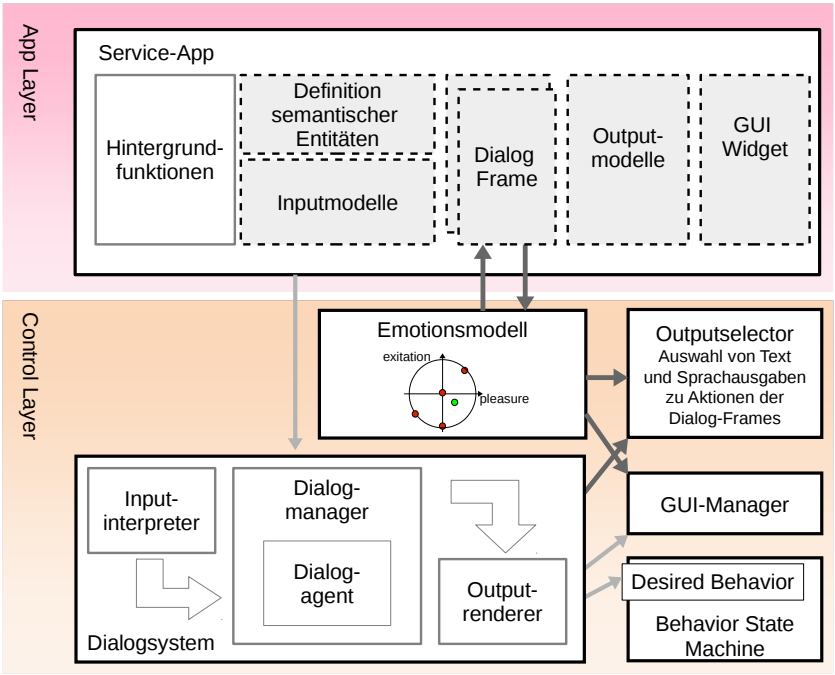



**Abbildung 5.5:** Der simulierte Emotionszustand kann bei der Definition von Ausgaben über GUI und Outputselector berücksichtigt werden. Beeinflusst wird der Zustand durch Ereignisse oder aus Aktionen der Dialog-Frames. Eine Berücksichtigung der Emotion im Dialogzustand als Kontext kann das Dialogverhalten verändern.

Mit den zuvor beschriebenen Komponenten ist es nun möglich, recht komplexe Roboterapplikationen mit zahlreichen Servicefunktionen zu realisieren,

wobei der Roboter durch die Möglichkeiten eines mixed-initiative Dialogs als Dialogpartner erscheint. Abhängig vom Einsatzzweck ist es manchmal wünschenswert, die Erscheinungsform zwischen sachlich (Werkzeugcharakter) und sozial emotional variieren zu können.
Speziell bei der Anwendung als Assistent in Langzeitinteraktion in der häuslichen Umgebung, wie beim SERROGA Projekt angestrebt, spielt die emotionale Bindung zum Roboter eine entscheidende Rolle für die Akzeptanz eines solchen Systems. Daher wird in der Architektur eine Komponente vorgesehen, welche es ermöglicht, serviceübergreifend einen emotionalen Zustand des Roboters zu modellieren. Dadurch können einerseits die Ausgaben des Systems variabler gestaltet und andererseits ein unterbewusster Weg zur Vermittlung roboterinterner Bedürfnisse (z.B. Müdigkeit bei schwachem Akku) zur Verfügung gestellt werden.
Das Emotionsmodell implementiert hierbei die Repräsentation und Modulation des robotereigenen Emotionszustands (siehe Abb. 5.5). Dieser sollte von den individuellen Serviceapplikationen in der Ausgabedefinition berücksichtigt werden. Für eine Dialogsituation können somit verschiedene Textausgaben für die unterschiedlichen Basisemotionen vorgesehen werden. Es bleibt aber auch möglich, z.B. für eine Notfall-Serviceapplikation, diese Option auszuschlagen und dadurch einen eher sachlichen Charakter umzusetzen.
Details zur Umsetzung des Emotionsmodells und eine weiterführende Diskussion werden in Abschn. 8.2 präsentiert. Hierbei wird auch erläutert, wie die Emotionszustände sich ändern und wie sie für die Variation von GUI und Dialogabläufen (siehe graue Pfeile am Emotionsmodell in Abb. 5.5) genutzt werden.
Der Outputselector dient der Umsetzung der situationsabhängigen Wahl von Text und Sprachausgaben und wurde zusammen mit dem Emotionsmodell eingeführt. Anstatt die Ausgaben direkt im Outputrenderer des Dialogsystems umzusetzen, kann die konkrete Generierung von Texten für einen bestimmten Kommunikationsinhalt an den Outputselector delegiert werden (siehe Pfeil in Abb.5.5). Dieser kann globale Zustandsvariablen, wie den Emotionsstatus, nutzen und aus einer vorgegebenen Menge an Phrasen für einen bestimmten Inhalt eine passende wählen. Außerdem kann durch Hinterlegung mehrerer Phrasen für eine Situation zusätzlich eine erhöhte Variabilität in der Erschei-

nungsform des Roboters erreicht werden, wodurch er für den Nutzer nicht so schnell langweilig und eintönig wird. Die Funktionsweise des Outputselectors wird in Abschn. 6.10 noch weiter vertieft.

## 5.5 Taskscheduler

Mit den bislang eingeführten Komponenten ist ein Roboter realisierbar, welcher alle Services für einen Gesundheitsassistenten erbringen kann. Das resultierende Verhalten ist dabei rein reaktiv und es gibt keine Vorausschau und Berücksichtigung der Nutzersituation, wenn Interaktionsaufgaben angestoßen werden müssen. Es wäre beispielsweise wünschenswert, auszuliefernde Erinnerung nicht rein zeitgesteuert zu triggern, sondern auch zu berücksichtigen, ob der Nutzer gerade beschäftigt ist. Ebenso ist es denkbar, vorausschauend Erinnerungen eher auszugeben, wenn zu erwarten ist, dass der Nutzer zum eigentlichen Erinnerungszeitpunkt nicht anwesend sein wird. Auch eine langfristige Planung von Aufgaben, die ein Roboter über den Tag wahrzunehmen hat, wäre wünschenswert, um diese optimal mit den Nutzerinteressen zu verbinden. Dadurch sollte auch die Akzeptanz eines häuslichen Assistenzroboters gesteigert werden, da dieser smarter agieren würde.
Im Rahmen des CompanionAble Projekts wurde den Komponenten der Ablaufsteuerung dazu eine weitere hinzugefügt, welche proaktiv ausgeführte Services anhand eines beobachteten Aktivitätsgrades des Nutzers und eines Prognosemodells für die Anwesenheit des Nutzers koordiniert.
Grundlage für eine situationsangepasste Auslieferung von Services ist die Beobachtung und Erfassung der aktuellen Tätigkeit des Nutzers. Hierbei kann einerseits auf robotereigene Beobachtungen aus der Skill Layer, auf den aktuellen Dialogzustand, aber auch beispielsweise auf Informationen von einer intelligenten Wohnumgebung, wie sie bei CompanionAble vorhanden war, zugegriffen werden.
Abb. 5.6 zeigt, wie die Aktivierung von Dialogen nicht mehr direkt aus den Service-App-Implementierungen geschieht, sondern durch eine Registrierung eines Interaktionstasks beim Taskscheduler ersetzt wird. Der Taskscheduler aktiviert daraufhin die entsprechenden Dialog-Frames beim Dialogmanager unter Berücksichtigung der Nutzerverfügbarkeit und Notwendigkeit der Aus-

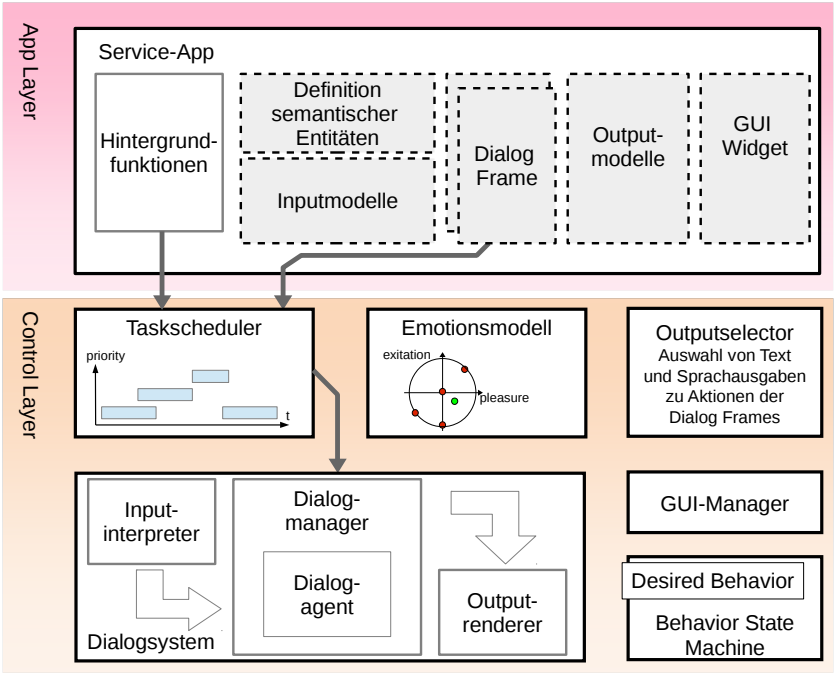



**Abbildung 5.6:** Beim Taskscheduler werden aus Dialogaktionen oder Service-App-funktionalitäten Tasks für die spätere Ausführung registriert. In geeigneten Situationen aktiviert der Taskscheduler über den Dialogmanager die dem Task entsprechenden Dialog-Frames.

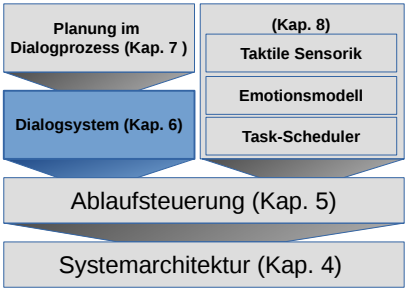
führung zu angepassten Zeitpunkten.
In Abschnitt 8.3 wird eine mögliche Realisierung, wie sie bei CompanionAble zum Einsatz kam, vorgestellt und kritisch diskutiert.

## 5.6 Fazit

In diesem Kapitel wurde ein Überblick über die modular gestaltete Ablauf und Dialogsteuerung in der vorgestellten Architektur gegeben. Es wurde gezeigt, wie schrittweise durch Hinzunahme neuer Komponenten in der Control Layer immer mehr Designaspekte umgesetzt wurden. Beginnend mit einer einfachen, robusten Steuerung der sicherheitskritischen Bewegungsverhalten

anhand eines Zustandsautomaten wurde unter Berücksichtigung der modularen Erweiterbarkeit um neue Service-Apps zunächst eine rein GUI-basierte Realisierung der Interaktion mit dem Nutzer eingeführt. Anschließend wurde Stück für Stück ein zentraler Dialogmanager hinzugenommen, welcher die Formung eines einheitlichen Interaktionspartners ermöglicht und viele Aspekte der Nutzeranpassung realisiert. Mit der Modellierung von Emotionen ist es schließlich möglich, einen Charakter zu formen und die Erscheinungsform des Systems gemäß der Rollen aus Kapitel 2 zu variieren. Einen smarter und rücksichtsvoller interagierenden Roboter erhält man durch einen Taskscheduler, welcher proaktives Verhalten des Roboters nicht mehr nur reaktiv und zeitgesteuert auslöst, sondern die Nutzersituation bzgl. Anwesenheit und Beschäftigungsgrad prognostiziert und berücksichtigt. Im weiteren Verlauf der Arbeit wird weiter auf die Details der verschiedenen Komponenten der Control Layer eingegangen. Insbesondere steht dabei im anschließenden Kapitel das Dialogsystem im Mittelpunkt. Emotionsmodell und Taskscheduler werden in Kapitel 8 behandelt werden.

# 6




# Dialogsystem

In der vorgeschlagenen Applikationsarchitektur stellt das Dialogsystem zusammen mit der Behavior State Machine die zentrale Entscheidungsinstanz dar, welche direkt das vom Nutzer erlebte Systemverhalten bestimmt. In diesem Kapitel soll das entwickelte Konzept des Dialogsystems näher erläutert werden. Die Grundidee der Dialogmodellierung basiert dabei zwar auf einem frame-basierten Ansatz, allerdings wird bei der Beschreibung der Dialogverläufe eine eigene Methodik zum Einsatz kommen, welche die angestrebte online Optimierung des Interaktionsverhaltens und die probabilistische Fusion von multimodalen Nutzereingaben ermöglicht. Es wird zunächst noch einmal resümiert werden, was ein Dialogsystem ausmacht und welche Randbedingungen für die Konzeption eines möglichst offenen, das heißt modularen und vielseitigen, Dialogsystems gegeben sind. Ein Überblick zu den Konzepten und Umsetzungen von Dialogsystemen in der Literatur wird eine Einordnung der in dieser Arbeit realisierten Lösung in den State of the Art ermöglichen.

Das vorgeschlagene Konzept umfasst eine losgelöste Verarbeitung und probabilistische Fusion von Nutzereingaben, welche in Abschn. 6.9 näher erörtert werden wird. Zuvor jedoch wird die Modellierung und Abarbeitung von Nutzerdialogen an einem Beispiel, angelehnt an die tatsächliche Implementierung des Systems im Rahmen des SERROGA Projekts, erläutert werden. Dieses Kapitel umfasst dabei die Komponenten des Kernsystems zur Realisierung

von einfachen Mensch-Roboter Dialogen, wie es im SERROGA System eingesetzt wurde (siehe Kap. 9). In Kapitel 7 erfolgt später die Schilderung der, zusätzlich zum Basissystem nutzbaren, Adaptivität und Optimierung mittels einer Planung von Dialogsequenzen.

## 6.1 Dialogmodelle

Vor der technischen Umsetzung eines Dialogpartners sollen zunächst einige theoretische Betrachtungen den Weg zur Modellierung ebnen. Unter einem Dialog im Sinne der Interaktion zwischen Mensch und Roboter soll in Anlehnung an die Mensch-Mensch Kommunikation der geordnete gegenseitige Austausch von Informationen und Anweisungen verstanden werden. Ein Dialog ist weiterhin das strukturierte Wechselspiel von aktiven und passiven Phasen der Dialogpartner. Diese einzelnen Phasen bilden für den aktiven Partner dabei meist abgeschlossene inhaltliche Einheiten, welche einer bestimmten Aufgabe dienen. Eine aktive Phase bis zum Wechsel der aktiven Rolle wird auch **Turn** genannt. Die Turns lassen sich anhand ihrer Intention in verschiedene Typen einteilen, wie z.B. Information, Frage, Antwort, Bestätigung oder Anweisung Im Kontext verbaler Dialoge wird dabei auch von **Sprechakten** gesprochen, was sich hier auf multimodale Dialoge übertragen lässt und in diesem Zusammenhang **Dialogakt** genannt werden soll.
Die Sprechakte geben der Turnfolge eine natürliche Struktur. Es lassen sich typische zusammengehörige Sprechakte identifizieren [Sacks et al., 1974]:

- question-answer
- propose-accept/reject/challenge
- offer-accept/decline
- compliment-refusal/thanks
- greeting-greeting

Diese Beobachtungen wurden bereits genutzt, um eine domänenunabhängige Steuerung für natürlichsprachliche Dialoge umzusetzen. Allerdings erfordert solch eine abstrakte Definition des Systemverhaltens eine sehr aufwändige Ausgabegenerierung und schränkt die Kontrollierbarkeit des Verhaltens im Einzelfall ein, weshalb in der vorliegenden Umsetzung darauf verzichtet wird.

Ähnlich einer zwischenmenschlichen Kommunikation wird für den Dialog zwischen Mensch und Roboter ein **turn-basiertes Regime** angestrebt, im Gegensatz zu einem reinen **user-initiative Interface**, wie beispielsweise ein Touchscreen am Handy oder ein Computer Desktop. Bei letzteren kann man eigentlich nicht von einem Dialog im Sinne eines gleichberechtigten Zwiegesprächs ausgehen, sondern es finden nur direkte Reaktionen auf die Nutzeranweisungen statt.

## 6.2 Klassifikation von Dialogsystemen

Computer-gestützte Dialogsysteme sind bereits seit den Anfängen der Rechentechnik im Interesse der Forschung. Dabei entwickelten sich recht bald **natürlichsprachliche Dialogsysteme**[1], welche mit den wachsenden technischen Möglichkeiten auch zu **multi-modalen Dialogsystemen** weiterentwickelt wurden.
Sprach-basierte Dialogsysteme gehen zurück auf die KI Forschung und Natural Language Processing (NLP) in den 1960er Jahren.
Es lassen sich zwei Haupteinsatzbereiche oder Domänen bei Dialogsystemen unterscheiden [McTear, 2004]. Dies sind zum einen theoretisch motivierte Modelle der Dialogführung basierend auf KI und NLP (**Task-orientiert**), und zum anderen die **Simulated Conversation** oder Human-Computer Conversation, welche mit Pattern Matching und datengetriebenen Methoden versucht, eine nicht notwendigerweise zielorientierte Konversation zu simulieren (siehe Abb. 6.1). Im Rahmen dieser Arbeit stehen die erstgenannten Ansätze im Vordergrund, da die Interaktion mit dem Roboter größtenteils zielgerichtet und auf festgelegte Inhalte beschränkt ist. In den letzten Jahren lässt sich beobachten, dass beide Welten zusammengeführt und datengetriebene Ansätze für die Realisierung taskorientierter Dialoge genutzt werden, was im Example-Based Dialog Modeling (EBDM) [Lee et al., 2009] mündet. Anhang A 6.2 geht näher auf die geschichtliche Entwicklung der Dialogsysteme ein. In der Literatur wurde weiterhin die Übertragung der klassischen Dialogsystemansätze auf die mobile Robotik behandelt. Anhang A 6.2.2 zeigt diese Überlegungen

[1] Anfangs basierend auf geschriebener Sprache [McTear, 2004] wurden ab 1980 erste Spracherkennungs und -syntheseansätze entwickelt.

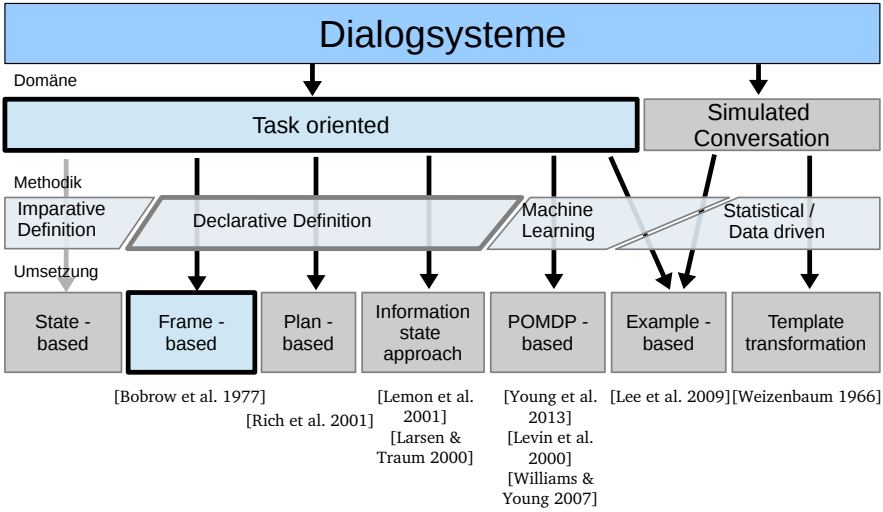



**Abbildung 6.1:** Kategorisierung von Dialogsystemansätzen nach Domäne, Methodik und konkreter Umsetzung. Die hervorgehobenen Teile kommen den in dieser Arbeit umgesetzten Konzepten am nächsten.

auf und motiviert letztendlich auch den in dieser Arbeit gewählten Integrationsansatz.
Zusätzlich zu den Domänen lassen sich nach Traum [Traum, 2008] unterscheidbare Herangehensweisen oder Sichtweisen für das Dialogmanagement identifizieren:

1. **Dialogsystem als Front-end für ein Back-end-system**[2]: Hier lautet die zentrale Frage: „Was soll das Dialogsystem zum Back-end-system oder zum Nutzer kommunizieren?“ (Vermittlerrolle)

2. **Dialogsystem als Agent**: Hierbei steht die zentrale Frage „Was als nächstes tun?“ im Mittelpunkt und das Dialogsystem versucht einen Ablauf zu generieren, um zusammen mit dem Nutzer ein Problem zu lösen.

Für die Realisierung eines Assistenzroboters, wie er in dieser Arbeit beschrieben ist, wird die Sicht als Front-end für ein Back-end-system, also das **Dia-**

[2] z.B. ein Nutzerinterface zum Abfragen einer Datenbank

**logsystem in der Vermittlerrolle**, präferiert, da der Roboter einen beschränkten Funktionsumfang (Back-end-system) besitzt, welcher lediglich für den Nutzer zugänglich gemacht werden muss. Die Aufgabenstruktur stellt sich eher als eine Vielzahl atomarer voneinander unabhängiger Tasks dar, weshalb Planungsschritte auf Task-Ebene, wie bei der Agentensicht auf das Dialogsystem, nur bedingt erforderlich sind.
Der Aussage von Allen et al. **„The Practical Dialogue Hypothesis: The conversational competence required for practical dialogues, although still complex, is significantly simpler to achieve than general human conversational competence.“** [Allen et al., 2001] folgend, wird hier davon ausgegangen, dass auch mit einem beherrschbaren Aufwand ein in vieler Hinsicht praktisches Dialogsystem realisiert werden kann.
Die akademische Entwicklung von Dialogsystemen brachte eine Vielzahl von Methoden hervor (siehe Abb. 6.1). Mit dem Einsatz von Machine Learning Techniken, speziell der Optimierung von Abläufen mit unsicheren Nutzerbeobachtungen mittels der Partially Observable Markov Decision Processes (POMDPs), wurde der Entwicklungsprozess zur Realisierung einer konkreten Anwendung jedoch immer komplizierter. Letztendlich werden mittels Prototypen der Systeme Interaktionsdaten mit realen oder simulierten Nutzern gesammelt, um anschließend die Optimierung der Policy (Dialogstrategie) durchzuführen.
Für die im Rahmen dieser Arbeit umgesetzte Methode stand jedoch ein einfacher, schneller Entwicklungsprozess im Mittelpunkt. Daher wird die eher einfache Modellierungstechnik der **frame-basierten Dialoge** aufgegriffen und um Möglichkeiten zur Adaption während der Laufzeit erweitert (wie in Kapitel 7 näher beschrieben).

## 6.3 Generischer Aufbau und Aufgaben eines Dialogsystems

Eine praktische Realisierung eines Dialogsystems folgt im Allgemeinen dem prinzipiellen Aufbau, wie er in Abb. 6.2 dargestellt ist. Dabei unterteilt sich das System aufgabenorientiert in eine Komponente zur Interpretation der Nutzereingaben (Inputinterpretation), den eigentlichen Dialogmanager (DM) und ein Modul zur Outputgenerierung.

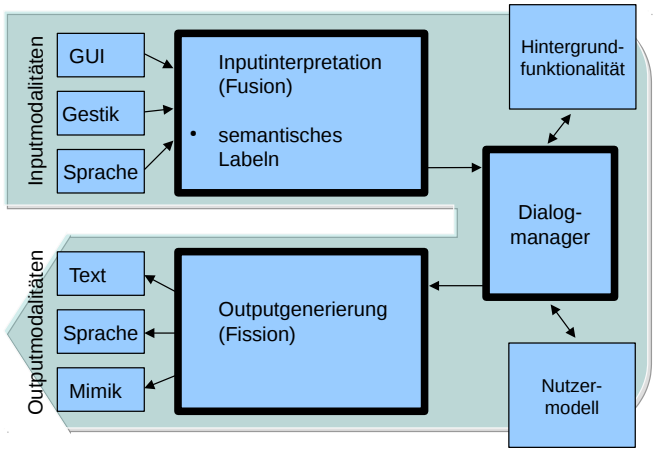



**Abbildung 6.2:** Prinzipieller Aufbau von Dialogsystemen

Mc Tear [McTear, 2002] sieht zwei Hauptaufgaben für Dialogsysteme: Die Verarbeitung der Nutzereingaben und die Behandlung von Fehlern (z.B. bei missverstandenen Eingaben). Hinzu kommt die Vermittlung von Hilfestellungen, wenn der Nutzer nicht genau weiß, was er wie mit dem System tun kann. Zur Behandlung von Fehlern gehört auch die Modellierung des kognitiven Zustands des Dialogpartners. Dessen Absichten, Ziele und Wissensstand zu kennen, ermöglicht erst ein korrektes Verhalten im Dialog. Dies führt den Gedanken weiter zum **Grounding**, womit der Abgleich beider Dialogpartner über den gegenseitigen Wissensstand zu einem bestimmten Fakt gemeint ist. Anhang A 6.3 gibt eine ausführlichere Definition zum Begriff Grounding.
Als Aufgaben für den Dialogmanager können somit zusätzlich „Grounding“ und “Repair“ formuliert werden.
Diese allgemein formulierten Aufgaben lassen sich den einzelnen Teilkomponenten eines Dialogsystems zuordnen und weiter konkretisieren:

### 6.3.1 Inputinterpretation

Ursprünglich ist die Aufgabe dieser Komponente die Überführung natürlichsprachlicher Äußerungen des Nutzers in eine für den Dialogmanager nutzbare

Form. Dabei gilt es, die Ausdrücke zu formalisieren und die Bedeutung, sprich die Semantik[3], zu extrahieren. Hierbei kommen verschieden komplexe Ansätze in Frage, angefangen von einfachen Schlagworterkennern, über Template Matching, bis hin zu Parsern, die eine angereicherte Grammatik nutzen, um Aktionen an erkannte Textbausteine zu binden (siehe 6.9.4). Ein Vergleich verschiedener Methoden zur Interpretation von natürlicher Sprache ist bei Knight et al. [Knight et al., 2001] zu finden. In multimodalen Systemen kommt die Fusion der Daten aus den unterschiedlichen Quellen hinzu.
Beispielsweise könnte ein Nutzer über Zeigegesten, Handzeichen oder Kopfgesten mit dem System kommunizieren. Weitere, vom Nutzer unbewusst geäußerte und vom Roboter erfassbare, Nutzermerkmale sind Mimik, Körpersprache oder aktuelle Tätigkeiten. Der Inputinterpreter kann auch vom Dialogmanager Informationen zum aktuellen Dialogverlauf heranziehen, um die Deutung von Eingabeereignissen der Situation anzupassen.

### 6.3.2 Dialogmanager

Dem Datenfluss folgend gehen die formalisierten Informationen über Nutzereingaben zum Dialogmanager (DM). Dieser ist im Gegensatz zu den anderen Komponenten zustandsbehaftet und sorgt für die Verwaltung des Diskurswissens. Weiterhin hat der DM auf dieser Basis eine Entscheidung darüber zu treffen, welche nächsten Dialogschritte getätigt werden sollen. Dazu hat der DM Zugriff auf das Hintergrundsystem (in unserem Fall auf den Roboter, dessen Zustand und Fähigkeiten), sowie auf ein evtl. vorhandenes Nutzermodell (siehe Abb. 6.2).
Nach Lemon et al. [Lemon et al., 2002] zerfällt das Dialogmanagement demnach in zwei Teilaufgaben: „dialogue modelling (representation), and dialogue control (algorithm)“.
Laut Mc Tear [McTear, 2002] umfasst das Dialogmodell alles Wissen, was zur Dialogausführung benötigt wird:

**Dialog History:** Ist eine Repräsentation des vergangenen Dialogverlaufs und speichert getroffene Entscheidungen und besprochene Themen. Für die

[3]Semantik (von griechisch sēmaínein, bezeichnen) nennt man die Theorie oder Wissenschaft von der Bedeutung der Zeichen.

Sprachverarbeitung kann dieses Wissen zum Auflösen von Anaphern und Polysemen genutzt werden.

**Task Record:** Dies ist eine Aufstellung aller für einen Task benötigten Informationen (form template, status graph), die definiert, welche Eingaben noch benötigt werden.

**World Knowledge Model:** Dies umfasst jegliches Wissen, das für eine korrekte Interpretation von Dialoginhalten im Sinne von Allgemeinwissen nötig ist.

**Domain Model:** Dies umfasst alle themenspezifischen Informationen und Daten.

**Generic model of conversational competence:** Das Modell umfasst das Grundverständnis vom Dialog, also das Wechselspiel der aktiven Rolle (turn taking) mit Timing und Berücksichtigung logischer Abhängigkeiten und sozialer Normen.

**User Model:** Das User Model umfasst zum einen alle statischen Informationen zum Nutzer, die wichtig für den Dialog sein können (Alter, Geschlecht, Präferenzen), und zum anderen auch dynamische Größen wie Dialogziele, Absichten und Vermutungen (im Sinne von Grounding), die ein Dialogpartner haben kann.
In der Umsetzung des Dialogmanagers und insbesondere der Definition der Policy für die zustandsabhängige Auswahl der Systemaktionen finden sich zahlreiche Spielarten unterschiedlich komplexer Ansätze (siehe Abb. 6.1 Umsetzung).

### 6.3.3 Outputgenerierung

Wie bereits bei der Inputinterpretation, war die ursprüngliche Aufgabe dieser Komponente rein auf die natürliche Sprache konzentriert. Dabei gilt es, aus den abstrakten Sprechakten, die der Dialogmanager als Reaktion auf die Nutzereingaben ausgewählt hat, eine natürlichsprachliche Ausgabe zu generieren, welche anschließend entweder als Text oder mit Hilfe eines Sprachsynthesesystems ausgegeben werden kann (Natural Language Generation - NLG).

Die Ansätze dafür decken wiederum ein reiches Spektrum unterschiedlich komplexer Methoden von festen Ausgaben (canned text), über template-basierte bis hin zu Grammatiken und statistischen Sprachmodellen ab, mit denen vollkommen freie Texte generiert werden können. Abschnitt 6.10 vertieft diese Thematik im weiteren Verlauf.
Im multimodalen Dialogsystem eines Roboters kommt zur NLG die Ansteuerung der weiteren Ausgabemodalitäten (grafische Nutzeroberfläche) und Effektoren des Systems hinzu. Hierzu kann im Fall eines Roboters die Ansteuerung der Mimik zum Ausdruck von Emotionen (siehe auch 8.2.6) oder die Nutzung der Manipulatoren zum Zeigen zählen.

## 6.4 Konzeption des Dialogsystems im Rahmen der vorgeschlagenen Applikationsarchitektur

Nachdem verschiedene Spielarten und Eigenschaften von Dialogsystemen erläutert wurden, soll nun der im Rahmen dieser Arbeit realisierte Ansatz begründet und dargestellt werden. Zunächst erfolgt eine Einordnung der für die Realisierung gewählten Konzepte.
Sprache als zentrales Kommunikationsmedium wird als wichtig erachtet, um eine Personifikation des Dialogpartners Roboter zu erreichen. Allerdings ist Spracherkennung auf einem mobilen Roboter ein schwieriges und bisher nur unzureichend gelöstes Problem. Im Konzept wird Spracheingabe mit ihrer Unsicherheit daher vorgesehen und beispielhaft implementiert (siehe Abschn. 6.9.4), wobei, bedingt durch den genutzten Spracherkenner (Julius [Lee et al., 2001]), Eingaben nur mittels eines Ansteckmikrofons robust erkannt werden, was für einen praktischen Einsatz jedoch ungeeignet ist. Die zentrale Rolle für die Nutzereingaben spielt daher der Touchbildschirm des Roboters. Für Ausgaben an den Nutzer hingegen wird die gesprochene Sprache als zentrales Medium beibehalten. Diese ermöglicht es, in der Wortwahl und in der Intonation zusätzliche unterbewusste Botschaften (z.B. Wertungen durch emotionale Darstellung, siehe Kap. 8) zu kommunizieren. Zusätzlich werden die gleichen Inhalte auch über den Bildschirm vermittelt.
Da der Roboter einen gewissen Grad an Autonomie besitzt und für bestimmte Aufgaben auch auf den Nutzer zugehen muss (z.B. wenn eine Erinnerung aus-

geliefert werden soll), wird eine **mixed-initiative** Dialogform angestrebt. Das Turn-Konzept des Dialogs als Wechselspiel von aktiven und passiven Phasen wird ebenfalls realisiert.

Auf eine Nutzung der Sprechakt-Theorie zur inhaltsunabhängigen Definition von Systemdialogakten (wie in Abschn. 6.1 kurz erwähnt) und einer entsprechenden Logik wird verzichtet, da damit im Allgemeinen hohe Anforderungen an die Ausgabegenerierung sowie die Inputinterpretation gestellt werden. Im multimodalen Dialog, in dem neben den Text- und Sprachausgaben auch Bildschirminhalte und weitere Steuerkommandos für den Roboter in einer deterministischen Weise zum Dialogakt dazugehören, ist eine domänenunabhängige Dialogkontrolle schwierig zu realisieren. Weiterhin ginge durch eine zentrale Definition der Logik im Dialogablauf die individuelle Kontrolle in den einzelnen servicebezogenen Teildialogen verloren.

Als Resultat dieser Analyse erfolgte eine Orientierung an den eher einfachen Ansätzen eines **frame-basierten Dialogmanagers** (siehe Abb. 6.1). Diese Entscheidung gewährleistet weiterhin eine gute Beherrschbarkeit der Aufgaben für den Entwickler von Service-Apps. Ein relativ einfacher Entwicklungsprozess ermöglicht es auch, schnell neue Services mit individuellen Teildialogen umzusetzen, ohne existierende Dialoge umgestalten zu müssen. Es wird eine **manuelle Definition einer Dialog-Policy** vorgesehen, um **ohne** das Sammeln von Dialogdaten, z.B. in einem Wizard-of-Oz Experiment, zu einem einsetzbaren System zu gelangen. Eine nachträgliche Optimierung der Abläufe ist dennoch entweder manuell oder auch durch Lernprozesse möglich. Die Fähigkeiten von Machine-Learning-Ansätzen, eine Optimierung der Dialogstrategie durchzuführen, wird ebenfalls berücksichtigt. Dabei wird neben systeminternen Optimierungszielen zusätzlich auf Nutzerinteressen Rücksicht genommen. Eine Optimierung findet daher zur Laufzeit und nicht in einer separaten Entwicklungsphase statt (näheres dazu in Kap.7). Dies wird durch den angestrebten Langzeitdialog notwendig. Auch im Mehrnutzer-Kurzzeitdialog kann das System eingesetzt werden und dabei entweder nutzergruppenspezifische[4] oder

[4]beispielsweise die Bestimmung der wirkungsvollsten Ausdrucksformen bei der Produktwerbung in einem Einkaufsassistenten in Abhängigkeit vom Geschlecht und Alter der Nutzer

generelle Dialogoptimierungen[5] erlernen. In beiden Dialogsituationen können unterschiedliche Erfahrungslevel der Nutzer, also ein unterschiedlicher Grad an Hilfestellungen und notwendigen Erklärungen, berücksichtigt werden, indem diese in den Zustandsraum der Teildialoge mit aufgenommen werden.

Neben dem effizienten Entwicklungsprozess spielt auch die Effizienz bei der Ausführung (limitierte Ressourcen auf einer mobilen Plattform) eine ganz wichtige Rolle. Durch einen Verzicht auf aufwändige Sprachinterpretation und Sprachgenerierung ist eine effiziente Ausführung möglich. Der gewählte Ansatz gibt dem Entwickler dadurch auch einen hohen Grad an Kontrolle über die Abläufe während der Interaktion. Dies ist insbesondere wichtig, wenn auch zustandsbehaftete Roboterfunktionalitäten wie die Auswahl des Navigationsbehaviors (siehe Abschn. 5.1) über den Dialog mitgesteuert werden müssen.

Neben den bis hier beschriebenen funktionalen Anforderungen an das entwickelte System sollen auch softwaretechnische Randbedingungen genannt werden, die im Konzept berücksichtigt wurden.

Eine modulare Aufteilung der Software in funktionale Blöcke macht Algorithmen austauschbar und ermöglicht ein unabhängiges Entwickeln und Testen der Komponenten. Sowohl experimentelle Verfahren als auch einfache praktische Lösungen für die Teilfunktionen sind realisierbar[6]. Dafür wurde die Kommunikation zwischen den Teilsystemen durch entsprechend flexible und schmale Interfaces realisiert. Eine verteilte Definition der Dialoginhalte im Rahmen der Service-Apps ermöglicht es, zum laufenden System auf Nutzerwunsch eine Dialogkomponente hinzuzunehmen oder auszutauschen. Die Aspekte der Adaption an den Nutzer gemäß Kapitel 3 wurde unabhängig von der Definition der einzelnen Serviceapplikationen ermöglicht, indem entsprechende Modelle und Algorithmen im Dialogmanager verwaltet werden. Die Deklaration der Teildialoge in den Service-Apps muss lediglich genügend Spielraum für eine Variation der Dialogverläufe vorsehen.

Die konzeptionelle Trennung von probabilistischer Inputfusion und Zustandsmodellierung im Dialogmanager ermöglicht es, eine Vielzahl auch determinis-

[5] z.B. die beste Reihenfolge, Fragen nach benötigten Informationen zu stellen. Bei manchen Fragen wird der Nutzer andere Daten mit angeben, wodurch eine weitere Nachfrage evtl. entfallen kann.

[6] beispielsweise eine deterministische Aktionsauswahl oder eine auf Basis der Planung zur Optimierung der Abläufe

tischer Informationsquellen im Dialog zu berücksichtigen und gegebenenfalls die Inputinterpretation auf andere Weise zu realisieren. Weiterhin sieht das Konzept vor, durch eine Rückpropagierung von Groundinginformationen zum Inputinterpreter eine Adaption der Modelle zum semantischen Labeling von Inputereignissen zu ermöglichen (siehe auch Abb. 6.3 Pfeil von Dialogmanager zu Inputinterpreter). Leider wurde diese Möglichkeit aus Zeitgründen bisher nur im Konzept vorgesehen, um den Aspekten aus Kap. 3 gerecht zu werden. Eine Implementierung der adaptiven Inputinterpretation steht noch aus.

## 6.5 Realisierung des Dialogsystems

Wie bereits erwähnt folgt die Umsetzung der klassischen modularen Strukturierung eines Dialogsystems in Inputinterpreter, Dialogmanager und Outputgenerierung. Alle drei Teile sind als eigenständige Softwaremodule implementiert, laufen parallel und sind nur lose über ein schmales RPC-Interface und wenige Datenkanäle gekoppelt[7] (siehe Abb. 6.3).
Die Verarbeitung der Nutzereingaben findet im Inputinterpreter statt. Wenn signifikante Eingaben erkannt wurden, so werden diese an den Dialogmanager übermittelt, damit dieser den Zustand der Konversation aktualisieren kann und auf diese Zustandsänderung entsprechend mit der Auswahl einer Systemaktion reagiert. Diese wird an den Outputrenderer übergeben, wo die Abarbeitung der multimodalen Aktionen stattfindet. Die reine NLG wird dabei noch einmal delegiert an den Outputselector.
Für die Kommunikation zwischen Inputinterpreter und Dialogmanager ist es notwendig, die auf vielfältige Weise kommunizierbaren Inhalte der Nutzereingaben zu formalisieren. Hierfür wird eine eigens entwickelte Ontologie genutzt, in welcher die Eingaben in semantische Klassen mit semantischen Werten als entsprechende Ausprägungen organisiert sind. Abschnitt 6.8 führt diese näher ein, da diese zum Verständnis der im Rahmen der Inputfusion realisierten Lösung benötigt werden. Für einen Robotersteuerungsdialog wären die semantische Klasse „Kommando“ mit den Werten „Stop“, „Folgen“ und „Fahre zu“ ,

[7] in der genutzten MIRA Middleware kommunizieren Module über Remote Procedure Calls (RPC) bei Punkt zu Punkt Verbindungen, und Daten mit unbestimmtem Empfänger werden auf s.g. Channels bereitgestellt und übertragen.

sowie die semantische Klasse „Zielposition“ mit den Werten „Couch“, „Küche“, „Wohnzimmer“ usw. relevant. Die eigentlichen Dialoginhalte sollen gemäß der

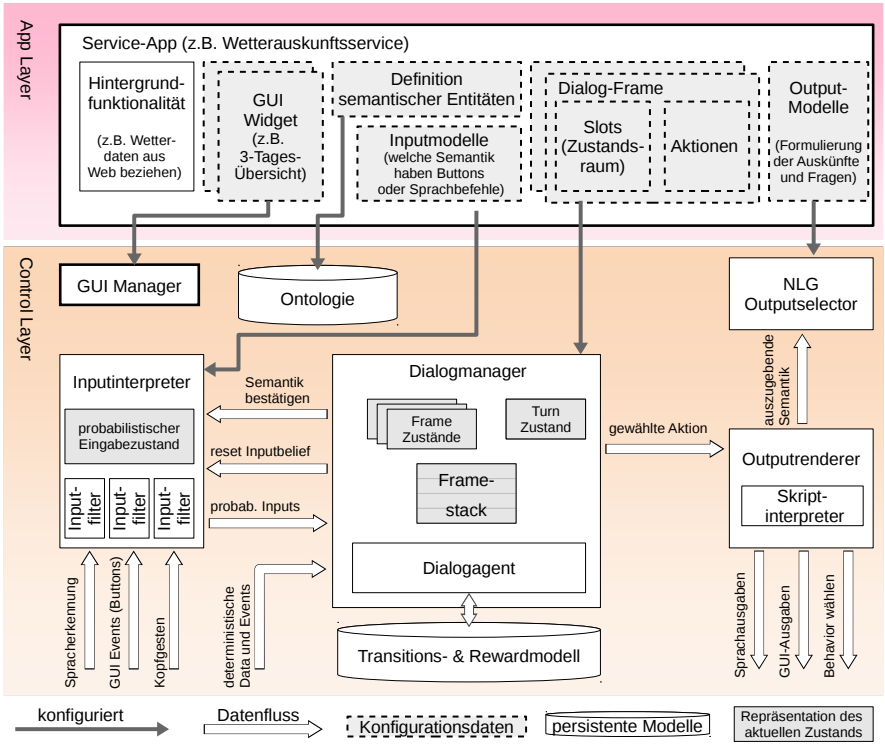



**Abbildung 6.3:** Komponenten des Dialogsystems, Kapselung der Konfigurationsbestandteile in einer Service-App der App Layer am Beispiel des Wetterauskunftsservice

Architektur aus Kap. 4 nicht in den zentralen Dialogsystemkomponenten der Control Layer implementiert, sondern unabhängig davon in den Service-Apps der App Layer konfiguriert werden können, damit eine Servicefunktionalität einfach als Modul dem System hinzugefügt werden kann.

Die verwendete Konfiguration des Dialogsystems aus der App Layer ist in Abb. 6.3 verdeutlicht und umfasst folgende Bestandteile einer Dialogkonfiguration:

- Semantische Entitäten (Klassen und deren Werte), welche in der ser-

vicespezifischen Domäne vorkommen können und den Wertebereich für Nutzereingaben und Systemausgaben aufspannen,

- Grammatiken und weitere Abbildungen von Inputmodalitäten auf die semantischen Werte zum Verstehen von multimodalen Eingaben (Inputmodelle),
- Definition der Inhalte von Teildialogen (Dialog-Frames), dabei sind der Zustandsraum (unter anderem Slots zur Aufnahme von Nutzereingaben oder Beobachtungen), die Aktionsmenge des Systems und die Policy für die Aktionsauswahl zu bestimmen,
- Outputmodelle für die Umwandlung von abstrakten Systemausgaben in reale Text und Sprachausgaben.

Bevor die vorgesehene Funktion der Teilkomponenten näher erläutert wird, soll zunächst der grundlegende Modellierungsansatz für die Dialoge vorgestellt werden.

## 6.6 Dialog-Modellierung

Mit den eben beschriebenen Konfigurationsdaten ergibt sich ein Bild der Dialogmodellierung gemäß Abb. 6.4, das am Beispiel eines Robotersteuerungsdialoges näher erläutert werden soll.

Alle vom System behandelbaren Dialogthemen werden in unabhängigen Teildialogen (auch Dialog-Frames genannt) definiert. Der Dialogmanager kennt diese Dialog-Frames und organisiert die Abarbeitung der Interaktion in eine lineare Sequenz von abwechselnden Turns. Dazu können einzelne Teildialoge aktiv werden, und somit auch die Dialogaktionen des Systems bestimmen. Im Dialogmanager wird ein Stack aktuell aktiver Teildialoge verwaltet. Dadurch wird es ermöglicht, dass ein Dialog zu einem Thema durch einen anderen Teildialog unterbrochen werden kann[8]. Der unterbrochene Dialog wird im Stack nach unten verdrängt, bis der unterbrechende Teildialog beendet ist. Danach wird mit dem alten Thema fortgesetzt.

Die Deklaration der Teildialoge muss es ermöglichen, eine Vorschrift zu definieren, welche Ausgaben und Aktionen vom System in verschiedenen Dialog-

[8] Beispielsweise kann das Eingeben eines Termins in den Kalender durch eine Nachfrage nach dem Wetterbericht für den entsprechenden Tag unterbrochen werden.

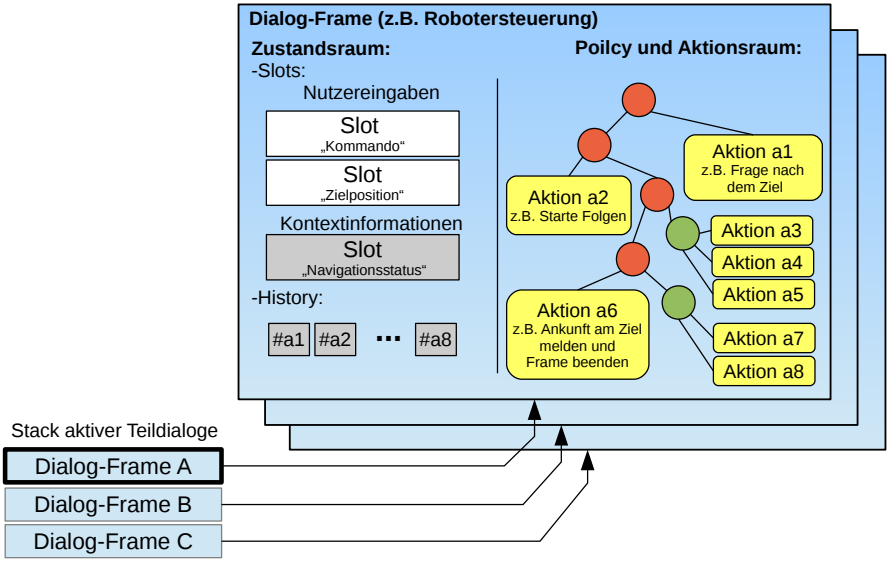



**Abbildung 6.4:** Modell des Dialogs; Im Dialogmanager wird ein Stack aktiver Teildialoge (Dialog-Frames) gehalten, wobei jeder Frame seinen eigenen Zustandsraum und Aktionsraum definiert.

situationen zu tätigen sind. Dazu wird zunächst ein Zustandsraum festgelegt, und anschließend erfolgt die Spezifikation der Aktionsmöglichkeiten, welche in Abhängigkeit vom Zustand des Teildialogs ausführbar sind (siehe Abb. 6.4). Der Gesamtzustand im Dialog wird somit verteilt repräsentiert, da die einzelnen Teildialoge voneinander unabhängig sind.

**Zustandsraum** Jeder Dialog-Frame besitzt einen eigenen Zustandsraum bestehend aus Variablen zur Repräsentation von Nutzereingaben. Diese werden **Slots** genannt. Neben den direkten Eingaben und Beobachtungen vom Nutzer können auch systeminterne Kontextinformationen in diesen Slots gespeichert werden und bestimmen somit das Verhalten mit. Für das Beispiel wären die zu berücksichtigenden Slots das „Kommando“ und die „Zielpositi-

on“, welche Werte gemäß der Definition in der Ontologie annehmen können[9]. Da die Interpretation von Nutzereingaben nicht immer eindeutig ist, wird auch im Dialog der probabilistische Gedanke der Inputinterpretation (siehe Abschn. 6.9) weiterverfolgt. Es müssen demnach nicht nur diskrete Werte in den Slots gespeichert werden, sondern Wahrscheinlichkeitsverteilungen über die semantischen Werte der entsprechenden Klassen. Für die Zielposition also eine diskrete Verteilung über „Couch“, „Küche“, „Wohnzimmer“, usw.. Aus Effizienzgründen kann diese Strategie leider nicht umgesetzt werden. Die Daten in den Slots sind jedoch als Paare $(W_s, C_s)$ von wahrscheinlichstem **Wert** $W$ und **Zuverlässigkeit** oder Konfidenz $C$ dieser Information repräsentiert[10]. Der Zuverlässigkeitswert der Slots dient somit außerdem der Realisierung von Grounding[11]. Nach einmaliger Eingabe ist die Konfidenz eines Fakts noch gering und mit jedem Dialogschritt, der den Slot-Wert direkt oder indirekt bestätigt, steigt die Konfidenz. Der Konfidenzwert zählt quasi die Bestätigungen (über verschiedene Inputkanäle oder über mehrere Turns hinweg).
Als Kontextinformationen für den Robotersteuerdialog wäre beispielsweise ein Slot für den Navigationsstatus nützlich, welcher darüber Auskunft gibt, ob der Roboter fährt, an einem Hindernis angestoßen oder am Ziel angekommen ist. Im Dialog sollen diese Ereignisse an den Nutzer kommuniziert werden.
Zum Zustand des Dialog-Frames kommt neben den Slots noch eine Repräsentation der Dialoghistorie hinzu (wie in Abschn. 6.3.2 gefordert). Dabei wird für jede Systemaktion $a_i$ gezählt, wie oft diese seit Aktivierung des Teildialogs bereits ausgeführt wurde.
Details zur Implementierung des Dialogzustandes finden sich in Anhang A 6.6.2.

**Aktionsraum** Die Menge der zur Verfügung stehenden Aktionen wird ebenfalls im Frame definiert, wobei gleichzeitig die Definition einer Policy[12] stattfindet, welche abhängig vom Zustand des Frames unterschiedliche Aktions-

[9] Beispiele für semantische Werte und semantische Klassen für den Robotersteuerungsdialog wurden in Abschn. 6.5 genannt.

[10] Diese Approximation des Zustandsraumes ist auch bei POMDP-basierten Dialogmanagern [Williams, 2003] zu finden, da dort die Dimensionalität des Zustandsraumes von entscheidender Bedeutung für die Lösbarkeit des Optimierungsproblems ist.

[11] Abgleich und Modellierung des gegenseitigen Verständnis der Dialogpartner über einen kommunizierten Fakt (siehe Anhang A 6.3)

[12] Entscheidungsfunktion

mengen zur Auswahl lässt. Dadurch kann gewährleistet werden, dass das System immer plausibel reagiert, auch wenn lernende Aktionsauswahlverfahren (wie später in Kapitel 7 beschrieben) zum Einsatz kommen, um innerhalb der Aktionsmengen die optimale zu finden. Die Policy wird in Form eines speziellen Entscheidungsbaumes über dem Dialogzustand festgelegt (in Abb. 6.4 rechts), wodurch Vollständigkeit über den möglichen Zuständen gewährleistet ist. Es ist demnach in jedem Zustand immer mindestens eine Aktion verfügbar.

Der Entscheidungsbaum enthält drei Typen von Knoten. Erstens die inneren **Entscheidungknoten**, welche abhängig vom Frame-Zustand (siehe voriger Abschn.) in ihre Kindknoten verzweigen. Neben den normalen Entscheidungsknoten existieren zweitens **Auswahl-** oder **Optionsknoten** (in Abb. 6.4 als grüne Kreise dargestellt), welche bei der Auswertung des Baums für die Weiterverfolgung aller Kindzweige sorgen. Dadurch kann erreicht werden, dass mehrere Aktionen in einem Zustand ausführbar werden. Als Blätter des Baumes dienen zu guter Letzt **Aktionsknoten**, in welchen die Definition der eigentlichen Aktionen als kleines Skript geschieht. Dabei wird eine Systemausgabe oder ein Systemverhalten definiert, welche entweder eine Nutzerinteraktion realisiert oder lediglich systemintern abläuft (siehe dazu auch Abb. 6.5).

Als Aktionen für das Beispiel wären neben vielen anderen die „Nachfrage nach einem Ziel“ bei erkanntem Kommando „Fahre zu“ oder die Ausgabe einer Bestätigung „Ich bin angekommen“ denkbar, welche ausgegeben wird, wenn der Navigationsstatus auf „Ziel erreicht“ wechselt. Dazu sind jeweils eine entsprechende GUI-Seite und eine Textnachricht auszuwählen und auszugeben.

Weitere Informationen zur Realisierung der Aktionen finden sich unter Abschn. A 6.6.3 im Anhang.

## 6.7 Steuerung der Abläufe durch den Dialogmanager

Der Dialogmanager ist das zentrale Element im System, in dem die oben geschilderten Zustände der Frames getrackt werden, und in welchem die Aktionsauswahl stattfindet. Der Dialogmanager steuert dazu auch das globale Wechselspiel der Turns, also die Zyklen von Systemaktionen und Nutzereinga-

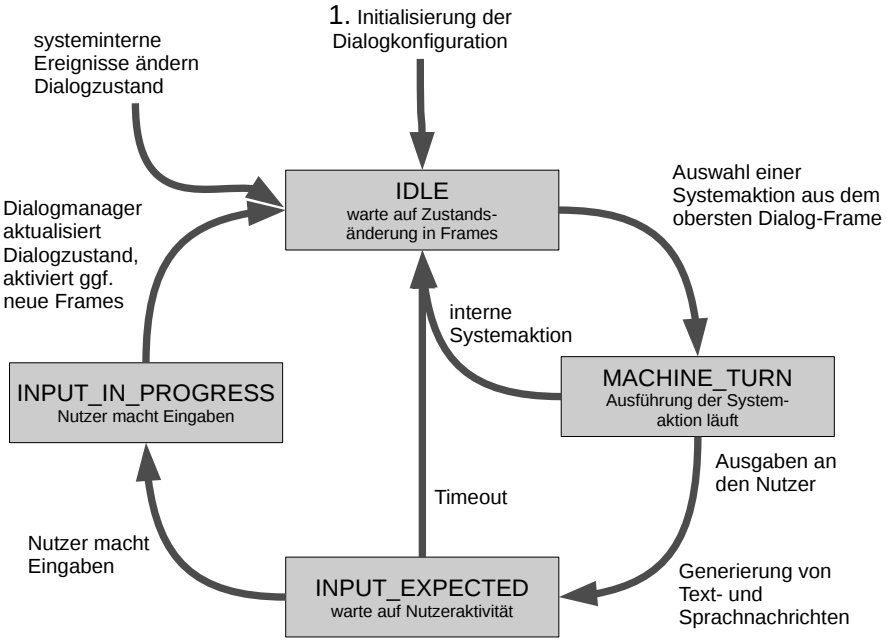



**Abbildung 6.5:** Zyklischer Ablauf im Dialog; die Kästchen verdeutlichen die verschiedenen Turnzustände

ben. Abb. 6.5 verdeutlicht diesen Ablauf, wobei das System die meiste Zeit bei Inaktivität im IDLE Zustand ist. Wenn sich durch Nutzereingaben oder interne Ereignisse der Zustand eines Frames ändert, kann ein MACHINE_TURN ausgelöst werden[13]. In diesem wird die Auswahl der auszuführenden Systemaktion des obersten Frames im Stack ausgeführt. Dazu wird die Policy (Abbildung von Dialogzuständen auf Aktionen) unter Berücksichtigung des aktuellen Systemzustands mittels eines Durchlaufs des Entscheidungsbaumes ausgewertet[14]. Dabei ergibt sich die Menge $\hat{A}$ der für die aktuelle Situation sinnvollen Aktionen, wie sie durch die Entwickler vorgesehen wurden.
Aus der resultierenden Aktionsmenge ist nun eine spezifische auszuwählen. Dies wird durch den eigentlichen **Dialogagenten** ausgeführt, wobei dieser

[13] Ob ein neuer Turn getriggert wird, wenn ein Slot seinen Wert ändert, kann bei der Definition der Slots mit spezifiziert werden.
[14] Algorithmus Baumtraversion im Anhang A 6.1

sowohl eine einfache deterministische Auswahl[15] treffen als auch nach dem in Kapitel 7 beschriebenen adaptiven Planungsmechanismus handeln kann. Entsprechend der ausgewählten Aktion setzt der Dialogmanager den neuen Turnzustand. Ausschlaggebend hierbei ist, ob die Aktion eine Nutzeraktion erwartet oder eben nicht. Falls der Nutzer nicht reagiert, wird mittels eines Timeout die weitere Funktion sichergestellt. Wenn vom Inputinterpreter eine beginnende Eingabe erkannt wird, so wechselt der Turnzustand auf INPUT_IN_PROGRESS, bis eine signifikante Änderung des Eingabezustands (siehe Abschn. 6.9) stattfindet und dem Dialogmanager übermittelt wird. Die Aktivierung neuer Dialog-Frames geschieht entweder system-getriggert durch den Taskscheduler (siehe Abschn. 5.5) bzw. aus der Implementierung der Servicehintergrundfunktionalität, aus Dialogaktionen heraus, oder nutzergetriggert durch das Füllen bestimmter Slots. Beendet werden Dialog-Frames aus den systeminternen Dialogaktionen.
Dieser Überblick zur Funktionsweise des Dialogmanagers soll an dieser Stelle für ein grobes Verständnis genügen. Weiterführende Details zum Ablauf im Dialogmanager sind im Anhang A 6.7.1 zu finden. Neben dem zentralen Dialogmanager sind für ein funktionierendes System noch weitere periphere Komponenten entscheidend. Diese werden im Anschluss näher erläutert.

## 6.8 Ontologie

Die unterschiedliche Ausprägung der vom Nutzer kommunizierten Inhalte auf den verschiedenen Kommunikationskanälen macht es notwendig, eine gemeinsame Sprache zu finden, in welche die Inhalte übersetzt werden können, um die Datenfusion zu betreiben und Dialoge unabhängig vom Kommunikationskanal definieren zu können.
Das einfachste Beispiel ist eine Ja/Nein-Entscheidung, bei der zwei semantisch unterschiedliche Werte für den Dialog relevant sind, eben das „Ja“ und das „Nein“. Aus Sicht der Nutzer wird diese Auswahl in Form einer Sprachausgabe, also eines Textstrings, einer Bildschirmeingabe durch Anklicken einer Schaltfläche oder einfach durch eine Kopfgeste (Nicken oder Schütteln) geäußert.

[15] immer die Erste, was sinnvoll ist für einfache Dialoge in denen keine Variationsmöglichkeiten vorgesehen sind, wie in der SERROGA Implementierung

Für die Definition der Dialogabläufe sollte die Art der Kommunikationsmittel jedoch irrelevant sein.
Ähnlich anderer Ontologiebeschreibungssprachen wie OWL oder RDF werden zunächst Klassen von Entitäten definiert, auf welche sich die Definition der Dialoge beziehen kann. Diese sollen **semantische Klassen** genannt und mit $K_j$ bezeichnet werden. Semantische Klassen geben einer Menge $R(K_i)$ von **semantischen Werten** $v_j \in R(K_i)$ eine Bedeutung. Ein Beispiel könnte die Zielposition[16] für den Roboter sein. Ein Wert wie z.B. „Wohnzimmer“ bekommt erst eine Bedeutung, wenn er einer semantischen Klasse zugeordnet wird. Es könnte sich sonst entweder um die Zielposition oder eine Location für einen Termineintrag handeln. Noch deutlicher wird die Bedeutung der semantischen Klasse für Zahlenwerte oder Zeiten. 13:00 Uhr könnte eine Termin-Startzeit oder die Uhrzeit für die Anfrage einer Wettervorhersage sein.
In einer Dialogkonfiguration werden diese semantischen Klassen genutzt, um den Zustandsvariablen (Slots) eine Bedeutung zu geben und somit einzuschränken, welche erkannten Nutzereingaben diesen Slots zugewiesen werden. (Siehe Abschn. 6.6)
Die in der Ontologie definierten Klassen und Werte dienen dem Inputinterpreter als Grundlage für die Definition des möglichen Eingaberaumes und des Zustands der Eingabesituation (wie im Anschluss beschrieben). Außerdem können für einzelne semantische Werte Zusatzinformationen, wie eine Textdarstellung oder Icons in einem Lexikon zugeordnet werden, welche von der Ausgabegenerierung genutzt werden können, um generische Bildschirmseiten und Eingabemenüs zu generieren, wenn nach bestimmten Klassen gefragt werden soll. Implementierungsdetails sind in Anhang A 6.8 zu finden.

## 6.9 Probabilistische Inputfusion

Bevor die Verarbeitung der mannigfaltigen Beobachtungen und Eingaben beschrieben wird, soll zunächst ein Überblick zur Vielfalt der Aspekte der in einem multimodalen Robotersystem zu berücksichtigende Nutzereingaben ge-

[16] Wie bereits erwähnt wäre „Zielposition“ die semantische Klasse und „Couch“, „Küche“, „Wohnzimmer“ usw. die semantischen Werte.

geben werden. Abb. 6.6 zeigt eine Sammlung von Eigenschaften und Ausprägungen von Inputs. Im entwickelten Dialogsystem konnten diese größtenteils Berücksichtigung finden.

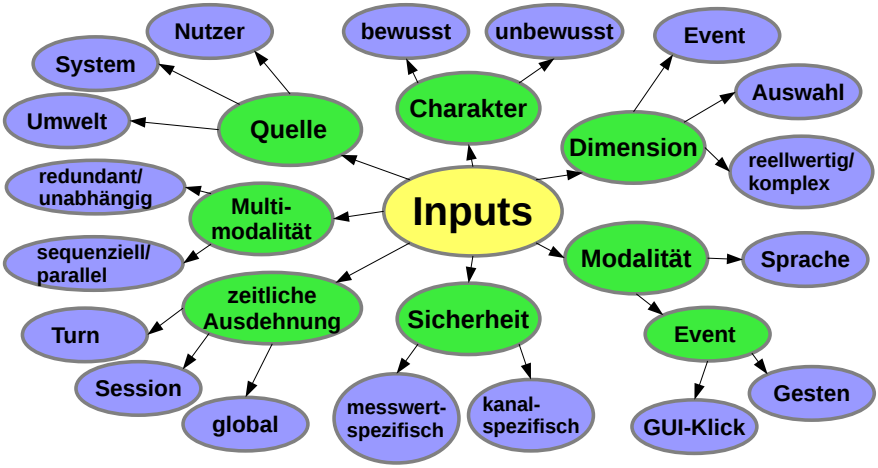



**Abbildung 6.6:** Übersicht zu Eigenschaften von zu berücksichtigenden Inputs

Neben den direkt vom Nutzer geäußerten Eingaben können auch die Umwelt oder das Robotersystem selbst als Quelle für eine dialogrelevante Information in Frage kommen. Neben den probabilistisch verarbeiteten Eingaben im Inputinterpreter können Informationen im Dialogmanager auch von beliebigen anderen deterministischen systeminternen Datenquellen[17] stammen. Dadurch wird eine große Flexibilität in der Umsetzung konkreter Dialoge erreicht.
Die Informationen, die den Dialogzustand ausmachen, können entweder bewusst oder unbewusst vom Nutzer kommuniziert werden. In letzterem Fall soll von Nutzerbeobachtungen gesprochen werden. Im Allgemeinen realisieren Module in der Skill Layer (siehe Abschn. 4.4.2) diese Nutzerbeobachtungen. Als Beispiel könnte eine Schätzung des Stresslevels des Dialogpartners anhand der Prosodie seiner Stimme dienen, wie sie in [Prüger, 2008][18] beschrieben ist.

[17] siehe Beispiel Navigationsstatus in Abschnitt 6.6 Seite 99
[18] Diese Arbeit wurde vom Autor betreut.

Bewusste Eingaben wären Kanäle wie Spracheingaben, Bildschirmeingaben oder auch Zeigegesten, deren Erkennung ebenfalls in der Skill Layer implementiert werden würden. Zusätzlich können Kopfgesten bewusst oder unbewusst eine Aussage ergänzen. All diese Kanäle besitzen eine ihnen eigene Zuverlässigkeit, welche bei der Interpretation für den Dialog und damit bei der Fusion der Modalitäten berücksichtigt werden kann. Außerdem können auch einzelne Beobachtungen eines Kanals mit unterschiedlicher Sicherheit aufgenommen worden sein, was ebenfalls eine probabilistische Auswertung erfordert, wie sie im Inputinterpreter realisiert wurde.
Eine weitere Eigenschaft der Inputs ist ihr Datenformat. Hier können einerseits Events und andererseits Auswahlen oder numerische Eingaben auftreten. Dies hat einen Einfluss auf die Modellierung der semantischen Werte, welche als eindeutige Vermittlerschicht eingeführt wurden (siehe Abschn. 6.8).
Erst im Dialogmanager wird die Geltungsdauer von Nutzereingaben behandelt. Eingaben können einerseits recht kurzzeitig gelten und sich nur auf die letzte Aussage beziehen, wie etwa eine Bestätigung, oder sie haben längere Geltungsdauer, entweder für die Ausführungsdauer eines Teildialogs (Session) oder noch länger (global). Dies macht es notwendig, dass der Dialogmanager zum Inputinterpreter kommuniziert, wann der Eingabezustand für eine bestimmte semantische Klasse zurückgesetzt werden soll[19], damit diese einen neuen Wert für den neuen Geltungsbereich annehmen kann.
Die Funktion der probabilistischen Schätzung des Eingabezustands wird in Abschn. 6.9.2 beschrieben. Zuvor jedoch sollen die verschiedenen Fälle von Eingaben innerhalb eines Turns auf mehreren Kanälen und die Ansatzmöglichkeiten für eine Datenfusion erörtert werden.

### 6.9.1 Datenfusion

Für die Fusion von Inputs sind drei Fälle von Zusammengehörigkeit der Einzelinformationen zu unterscheiden.
Im einfachsten Fall haben auf verschiedenen Kanälen kommunizierte Informationen nichts miteinander zu tun und stellen unabhängige Fakten dar. In

[19] Dieses Reset-Signal ist auch in Abb. 6.3 auf Seite 97 zwischen Inputinterpreter und Dialogmanager zu sehen.

diesem Fall hat die Fusionskomponente, in unserem Fall der Inputinterpreter, die Eingaben getrennt voneinander dem Dialogmanager zur Verfügung zu stellen.
Der zweite Fall beinhaltet Informationen, die den gleichen Fakt auf verschiedenen Kanälen in redundanter Weise beschreiben. Diese sollten in der Fusionskomponente genutzt werden, um Unsicherheiten in den Erkennungsleistungen und Übertragungskanälen zu reduzieren. Hierbei können Redundanzen sowohl zwischen Kanälen als auch zeitlich durch Wiederholungen vorkommen. In der konkreten Umsetzung werden gleichzeitig kommunizierte Fakten im Inputinterpreter fusioniert, und eine Aggregation der über mehrere Turns geäußerten Fakten findet auch statt, sofern der Dialogmanager nicht festlegt, dass die Schätzung zu einer bestimmten semantischen Klasse zurückgesetzt werden muss, weil der entsprechende Slot geleert wurde (z.B. beim Beenden eines Frames).
Der dritte und schwierigste Fall ist eine synergetische Bedeutung, die erst durch ein zeitliches Zusammentreffen von Eingaben auf verschiedenen Kanälen entsteht. Ein Beispiel hierfür wäre eine Zeigegeste, die zusammen mit einer Spracheingabe „dorthin“ einen Zielort bedeutet, in Kombination mit einer anderen Spracheingabe dagegen ein Objekt beschreibt, welches der Roboter bringen soll.
Synergetische Fusion wird aufgrund ihrer hohen Spezifität nicht als Basisbestandteil des Inputinterpreters implementiert. Synergetische Fusion findet entweder bereits auf Skill Ebene statt (Data- und Feature-Level; siehe folgender Abschnitt), oder wird direkt im Dialogmanager in der Dialogkonfiguration vorgesehen[20] (Decision-Level).
Dies zeigt, dass Fusion der Datenströme auf verschiedenen Ebenen geschehen kann. Im Allgemeinen unterscheidet man folgende Level:

1. **Data-Level**: Hierbei werden vor der Weiterverarbeitung Rohdaten kombiniert und als Einheit der Klassifikation zugeführt. Vorteile bietet diese Stufe der Fusion bei Kanälen, welche für sich allein gestellt nicht eindeutig wären. Die Ausprägungen der Daten sind also für ihre Bedeutung UND-verknüpft. Eine Berücksichtigung der Data-Level Fusion

[20] z.B. durch zwei getrennte Slots und eine Und-Verknüpfung in den Bedingungen des Entscheidungsbaumes für die Aktionsauswahl

kann nur in den Skills (technischen Basisleistungen in der Architektur) geschehen. Beispiel dafür wäre die Messung von Gelenkwinkeln eines Roboters. Ein Gesamtbild der Körperstellung ergibt sich nur wenn alle Winkel gleichzeitig bekannt sind.

2. **Feature-Level**: Diese setzt nach einer ersten Vorverarbeitung der Rohdaten verschiedener Kanäle an. Typischerweise bestehen die ersten Schritte in einer Verarbeitungskette aus einer Merkmalsextraktion, um danach eine Entscheidungsinstanz damit zu versorgen. Bei der Feature-Level-Fusion findet die Merkmalsextraktion in separat laufenden Einheiten statt und eine Kombination der gefundenen Merkmale wird vor der Klassifikation vorgenommen. Dies ist insbesondere sinnvoll, wenn sehr verschiedenartige Signale zunächst kompatibel gemacht werden müssen.

   Beispiel hierfür wäre die Kombination von Audiosignalen mit Videobildern der Lippenbewegung. Aus beiden können Featurevektoren extrahiert werden, welche anschließend in ihrer zeitlichen Abfolge synergetisch beispielsweise mittels eines HMMs klassifiziert werden.

   In der Architektur wäre diese Stufe ebenfalls in der untersten Schicht, der Skill Layer, zu finden und kann im Dialogsystem nicht mehr berücksichtigt werden.

3. **Decision-Level**: Diese Art der Fusion arbeitet auf Klassifikationsergebnissen vorgelagerter Stufen und ermöglicht bei unsicheren Klassifikationsausgaben verschiedener Kanäle, die Unsicherheiten zu reduzieren und damit die Messung zu verbessern. Die Decision-Level Fusion, oder auch Symbolic-Level-Fusion ist prädestiniert dafür, innerhalb des Dialogsystems stattzufinden.

Eine tiefer gehende Einführung mit einer weiter untergliederten Klassifikation der Fusionsansätze ist in [Lalanne et al., 2009] zu finden. Ebenso wird in [Jaimes und Sebe, 2007] aus der Sicht des maschinellen Sehens auf die Fusionsaspekte für eine multimodale Interaktion eingegangen.

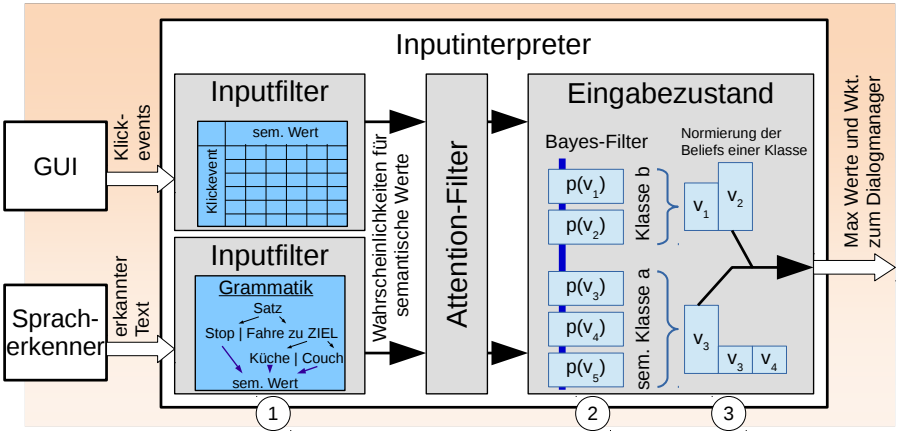



**Abbildung 6.7:** Datenfluss im Inputinterpreter mit den drei genannten Verarbeitungsschritten, vergleiche auch die Einbettung im Dialogsystem Abb. 6.3 (Seite 97)

### 6.9.2 Verarbeitung der Nutzereingaben

Voraussetzung für eine Fusion ist das Vorhandensein multimodaler unsicherer Beobachtungen zum Vorkommen semantischer Werte in den Eingaben. Der in dieser Arbeit verfolgte Ansatz zur Datenfusion beruht auf einer probabilistischen Modellierung des Eingabezustands. Dabei können einzelne Eingabeereignisse[21] als unabhängige Beobachtungen betrachtet und innerhalb eines begrenzten Zeitraums akkumuliert werden.
Die Verarbeitung der Eingaben erfolgt dabei unabhängig vom Dialogmanager und den definierten Dialog-Frames in drei Schritten:

1. Interpretation von Eingabeereignissen in den unabhängigen Kanälen,

2. Akkumulation der Eingabeereignisse im Inputinterpreter,

3. Normierung und Ausschluss von Widersprüchen unter Nutzung der Ontologie

[21] Eingabeereignisse sind die Rohdaten auf den verschiedenen Kommunikationskanälen, z.B. ein erkannter Textstring oder eine Kopfgeste

Abb. 6.7 gibt noch einmal einen detaillierten Überblick zu den Vorgängen im Inputinterpreter.
Im ersten Verarbeitungsschritt werden aus den Rohdaten der Eingabekanäle (Eingabeereignisse), jeweils unabhängig von den anderen Modalitäten, die Wahrscheinlichkeiten für das Vorhandensein semantischer Werte bestimmt. Dies geschieht in kanalspezifischen Plugins[22] für den Inputinterpreter, welche in Abb. 6.7 als Inputfilter bezeichnet sind. Abb. 6.3 (Seite 97) zeigt diese auch in Einbettung ins gesamte Dialogsystem. In der praktischen Umsetzung wurden ein Inputfilter für die GUI, sowie einer für die Interpretation von Spracherkennungsergebnissen implementiert. Ursprünglich vorgesehen war zusätzlich eine Interpretation von Kopfgesten, was allerdings aus Zeitgründen noch nicht umgesetzt werden konnte.

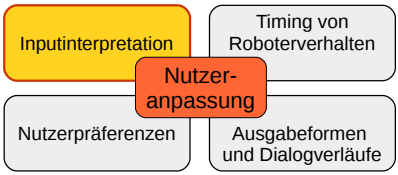



**Abbildung 6.8:** Behandlung des ersten Aspekts zu Adaptionsmöglichkeiten an den Nutzer.

Im Beispiel würde vom GUI-Inputfilter dem semantischen Wert „Fahre zu“ eine sehr hohe Auftrittswahrscheinlichkeit zugeordnet, wenn auf der GUI ein entsprechender Button gedrückt werden würde.
Neben solch einer recht deterministischen Zuordnung können die aus den Konfigurationen der Service-Apps erzeugten Inputmodelle (Abb. 6.7 blau in den Inputfiltern) auch probabilistischer Natur sein. Dann haben sie die Form einer bedingten Wahrscheinlichkeitstabelle für $p(Sem.Wert|Eingabeereignis)$. Diese Wahrscheinlichkeitstabellen bieten im Konzept prinzipiell die Möglichkeit für eine **Adaption der Inputinterpretation** wie sie in Kap. 3 vorgeschlagen wurde (siehe Abb. 6.8). Durch Rückpropagierung der Fusionsergebnisse und der Bestätigungen von semantischen Werten aus dem Dialogmanager könnten die Verbindungen zwischen Eingabeereignissen und semantischen Werten schrittweise verbessert werden, um sich den Eigenheiten der Nutzer anzupassen. Aufgrund der noch nicht erfolgten Integration der Modalität Kopfgesten konnte diese Adaptionsmöglichkeit leider noch nicht praktisch untersucht werden.

[22] Dadurch ist das System leicht um weitere Modalitäten erweiterbar.

Im Inputinterpreter, in dem die Fusion der erkannten semantischen Werte aus den Inputfiltern stattfindet, wird der so genannte **Eingabezustand** modelliert. Hierzu wird für jeden semantischen Wert aus der Ontologie eine Wahrscheinlichkeit dafür geschätzt, dass er vom Nutzer kommuniziert wurde. Dies geschieht mittels jeweils eines Bayes-Filters, welcher für ein Abklingen der Wahrscheinlichkeit über die Zeit sorgt, damit länger zurückliegende Eingaben durch neuere überschrieben werden können. Die Beobachtungsupdates dieser Bayes-Filter werden mit den Ergebnissen der Interpretation in den Inputfiltern vollzogen, was die Fusion dieser realisiert[23]. In Anhang A 6.9 wird dieses Vorgehen mathematisch näher beschrieben. Dieser Schritt der Modellierung findet sich in der Literatur auch beim Belief-Update [Young et al., 2013] im Rahmen einer POMDP-basierten Dialogmodellierung wieder.

Bis ein Eingabezyklus beendet ist, wird auf diese Weise die Fusion aller Beobachtungen vorgenommen. Sobald das Ende der Eingabe erkannt wurde, beispielsweise durch eine Pause in der Spracherkennung oder nach einer GUI Eingabe, wird der veränderte Eingabezustand an den Dialogmanager übermittelt.

Dazu umfasst der dritte Schritt anschließend die Berücksichtigung sich widersprechender semantischer Werte einer Klasse. Es könnte sein, dass mehrere der semantischen Werte einer Klasse innerhalb eines Turns erkannt wurden und sich demnach widersprechen. Beispielsweise kann in der Klasse „Zielposition“ nicht gleichzeitig die „Küche“ und die „Couch“ gemeint sein. Durch Normierung über die Werte einer semantischen Klasse kann eine valide Wahrscheinlichkeitsverteilung über diese Werte erzeugt werden[24] Anhang A 6.9 zeigt die Details dieser Normierung auf. Dieser Schritt ist gegenüber der herkömmlichen POMDP-Dialogmodellierung eine Neuerung, die nur durch die Einführung der Ontologie ermöglicht wurde und dadurch die Trennschärfe der Wahrscheinlichkeiten für die einzelnen Eingaben erhöht.

Nach der Normierung werden semantische Werte mit signifikanten Wahrscheinlichkeiten an den Dialogmanager übersendet, damit dieser die Slots der

[23] Sowohl zeitliche Aggregation als auch die Multiplikation der Ergebnisse der einzelnen Modalitäten findet in den Bayes-Filtern statt.

[24] Eine falsche Reaktion des Systems aufgrund zu hoher Konfidenzen in den Slots wird dadurch verhindert. Statt dessen wird eher noch einmal nachgefragt, um die Unklarheiten zu beseitigen.

Dialog-Frames damit aktualisieren kann. Die Übermittlung der Beobachtungen wird allerdings nur durchgeführt, sofern der Wahrscheinlichkeitswert gegenüber einer vorherigen Eingabe signifikant verändert wurde. Dadurch wird zu viel Datenverkehr und eine ungewollte Auslösung von Dialogturns verhindert.
Als letzter Schritt bleibt zu erwähnen, dass die Wahrscheinlichkeiten der semantischen Werte einer Klasse zurückgesetzt werden, wenn vom Dialogmanager ein entsprechendes Reset-Signal gesendet wird (siehe Abschn. 6.9), z.B. nach dem Beenden eines Teildialoges. Dies realisiert die Berücksichtigung der unterschiedlichen Geltungsdauer verschiedener Eingaben (siehe Abb. 6.6).

### 6.9.3 Aufmerksamkeitssteuerung

Die Abläufe, wie sie eben beschrieben wurden, machen leider entscheidende Annahmen. Erstens, dass der Interaktionspartner immer anwesend ist und zweitens, dass er seine Aufmerksamkeit auf die Interaktion mit dem System gerichtet hat. Für einen mobilen Serviceroboter als Langzeitbegleiter treffen beide Annahmen nicht immer zu. Daher ist es erforderlich, einerseits, falls die Initiative vom Roboter ausgeht, die Aufmerksamkeit des Nutzers zu gewinnen und gegebenenfalls den Nutzer zunächst in der Einsatzumgebung zu suchen. Hierzu soll durch die Dialogaktionen Gebrauch von den Navigationsverhalten (Behaviors) gemacht werden. Andererseits muss der Roboter auch entscheiden können, wann z.B. Spracheingaben an ihn gerichtet sind und wann nicht. Ebenso können auch Kopfgesten erkannt werden, wenn die Person gar nicht mit dem Roboter im Gespräch ist.
Um diese Problematik zu umgehen, wird für die multimodale Inputfusion eine Filterstufe vorgesehen, welche systemintern den **Focus of Attention** des Nutzers schätzt (siehe Abb. 6.7). Nur falls dieser auf den Roboter gerichtet ist, werden erkannte Eingaben auch weiterverarbeitet.
Für die Schätzung des Focus of Attention sind verschiedene anspruchsvolle Analysemethoden denkbar. Beispielsweise kann anhand eines Kurzzeithistogramms der Kopforientierung die Blickrichtung und damit auch der Aufmerksamkeitsfokus bestimmt werden [Hommel, 2009]. Um im Rahmen dieser Arbeit eine robuste und effiziente Lösung für dieses Problem zu finden, wird

die auf den Roboter gerichtete Aufmerksamkeit mittels eines Bayes-Filters anhand verschiedener Beobachtungen geschätzt. Ein periodisch angewendetes Prozessmodell zieht dabei den Zustand der Attention langsam gegen unbekannt. Beobachtungen zum Focus of Attention kommen einerseits vom Dialogmanager, wenn dieser eine Frage an den Nutzer richtet und somit erwartet wird, dass die nächsten Äußerungen des Nutzers ans System gerichtet sind, und andererseits von den jeweiligen Softwaremodulen, welche für die Auswertung jeweils einer Modalität zuständig sind. Konkret werden GUI-Eingaben mit Beobachtungen hoher Attention versehen und in der Spracherkennungsmodalität werden bestimmte Triggerwörter wie „Roboter“ oder der Name des Roboters erkannt, um die Attention zu erhöhen.
Die Mathematik dazu findet sich in Anhang A 6.9.3.

### 6.9.4 Inputfilter

Nachdem die prinzipiellen Abläufe im Inputinterpreter dargestellt wurden, soll an dieser Stelle auf die implementierten Modalitäten eingegangen werden. Diese werden jeweils von einem eigenen Plugin (Inputfilter) realisiert, wodurch eine leichte Erweiterbarkeit gewährleistet ist.

#### Bildschirmeingaben

Eingaben über den Touchbildschirm und die zugehörige (Bildschirm-)Tastatur können als sehr sicher angenommen werden. Ein Nutzer, der darüber Eingaben macht, ist sich dessen sicherlich bewusst und adressiert auch zwangsläufig den Roboter. Daher wird die Schwelle für die Aufmerksamkeitssteuerung für diesen Inputfilter sehr gering gewählt. Über die grafische Oberfläche lassen sich verschiedenste Eingaben machen. Einerseits kommen allgemeine Buttons zur Bestätigung oder Auswahl verschiedener semantischer Werte vor, und andererseits können auch Eingaben in Textform oder Interaktionen mit anderen Schaltflächen vorkommen. Die erstgenannten Button-Klicks werden im Allgemeinen über die Middleware MIRA an den Inputfilter signalisiert. Dieser kennt die Liste der Bedeutungen der Buttons, welche er aus den Konfigurationen der Services erhält, und leitet die entsprechenden semantischen Werte an den Inputinterpreter weiter.

### Spracheingaben

Für die Umsetzung einer Spracheingabemöglichkeit wurde im Rahmen dieser Arbeit der freie Spracherkenner Julius [Lee et al., 2001] genutzt. Dieser arbeitet auf Basis von HMMs[25] und setzt Gaussian Mixture Models für die Beschreibung der akustischen Modelle ein. Vom Spracherkenner wird für Sprachsignale nach Erkennen einer Pause eine Hypothese für den gesprochenen Text geliefert. Diese besteht aus mehreren Wörtern mit jeweils zugehörigen Konfidenzwerten, welche die Passfähigkeit der Audiosignale zum Modell im Spracherkenner widerspiegeln.

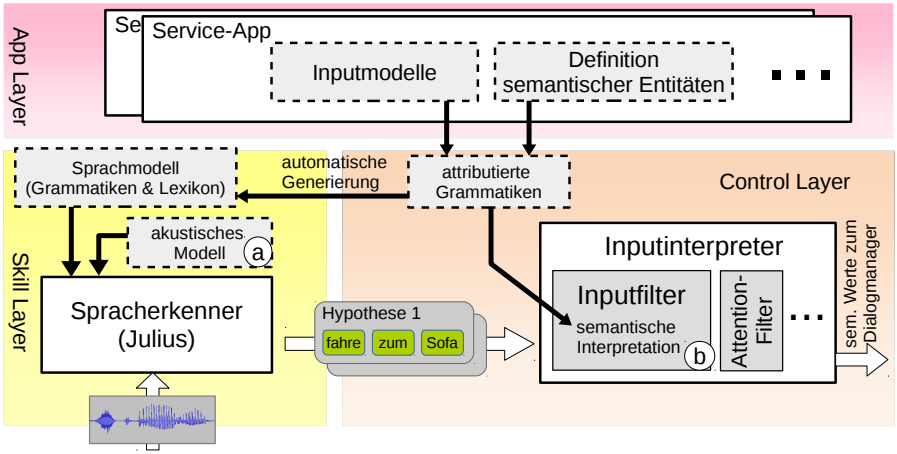



**Abbildung 6.9:** Datenfluss für die Spracherkennung und Konfigurationsprozess. Adaption an Nutzereigenheiten kann sowohl im Spracherkenner (a) in den akustischen Modellen als auch bei der semantischen Interpretation (b) geschehen.

[25] Hidden Markov Models

Abb. 6.9 verdeutlicht die notwendigen Zwischenschritte vom Audiosignal bis zum Zustandsupdate im Dialogmanager. Dabei ist der eigentliche Spracherkenner in der Skill-Layer angesiedelt. Im Inputinterpreter findet in einem speziell dafür vorgesehenen Plugin die semantische Interpretation statt.
Die Modalität Sprache ist gekennzeichnet durch große Unsicherheiten und eingehende Datenströme, auch wenn nicht der Roboter adressiert ist. Außerdem kann die natürliche Sprache sehr viele Varianten für die Übermittlung einer Bedeutung aufweisen, weshalb hier die Möglichkeit zum Lernen genutzt werden sollte. Einerseits bieten manche Spracherkenner die Möglichkeit, sich an die Stimme und Aussprache einzelner Sprecher anzupassen (a), wobei die akustischen Modelle angepasst werden, und andererseits kann auch im Rahmen der semantischen Interpretation versucht werden, die Bedeutung von einzelnen Wörtern für den Dialog zu lernen (b). Dies setzt allerdings voraus, dass der Spracherkenner domänenunabhängig zunächst beliebige Aussagen transkribieren kann. Bei dem genutzten Juliussystem wäre dies zwar potentiell möglich, erfordert aber erheblichen Trainingsaufwand und liefert weniger robuste Erkennungsergebnisse als die implementierte Nutzung domänenspezifischer Grammatiken. Es wäre für die Zukunft interessant auf State-of-the-Art Large-Vocabulary-Spracherkenner wie in [Weninger et al., 2014] beschrieben zurückgreifen zu können.
Für die semantische Interpretation wurde aufgrund der genannten Einschränkungen auf die Nutzung von attributierten Grammatiken (Semantic Grammars [McTear, 2004]) zurückgegriffen. Diese beschreiben die Struktur von Sprachkommandos an den Roboter in Form von Ersetzungsregeln[26]. Semantic Grammars sind weit verbreitet, beispielsweise verwendet auch der VoiceXML Standard diese Art der semantischen Interpretation von Sprache für die Beschreibung sprach-basierter Interfaces. Die Grammatiken werden zusammen mit den Dialog-Frames in den Service-Apps spezifiziert und können leicht genutzt werden, um die Grammatiken für den Spracherkenner daraus abzuleiten (siehe Abb. 6.9), welcher dadurch genau das erkennen kann, was auch von der semantischen Interpretation verstanden wird.
Der eingehende Textstring wird mittels der attributierten Grammatik geparst

[26]Im einfachsten Fall listet die Grammatik alle zu erkennenden Phrasen und die zugehörigen semantischen Werte auf.

und in einen Syntaxbaum überführt, in dessen Knoten die Regeln der Grammatik referenziert werden. Diese Regeln sind mit semantischen Werten attributiert, welche an den Inputinterpreter gesendet werden, sobald sie im Syntaxbaum vorkommen. Die Wahrscheinlichkeiten für das Vorkommen ergeben sich dabei direkt aus den vom Spracherkenner mitgelieferten Scores, welche die Erkennungssicherheit widerspiegeln.
Diese deterministische Variante der semantischen Interpretation von Sprache ist recht starr, und der Entwickler muss in der Grammatik alle interpretierbaren Sätze vorhersehen. Ein Lernen neuer Ausdrucksformen des Nutzers ist dabei nicht vorgesehen. Im Rahmen der Nutzung eines grammatik-basierten Spracherkenners wie Julius bietet die grammatik-basierte semantische Interpretation allerdings Vorteile, da ohnehin für die reine Erkennung der gesprochenen Texte eine Grammatik definiert werden muss.
Bevor die semantischen Werte weiterverarbeitet werden, muss die Aufmerksamkeitsschwelle überschritten sein. Damit unerwünschte Reaktionen des Systems auf nebensächliche Sprachsignale verhindert werden, ist die Schwelle für diese Modalität recht hoch gewählt. Um Sprachkommandos an das System zu richten, können in der attributierten Grammatik bestimmte Schlüsselwörter markiert werden, welche die Aufmerksamkeit erhöhen, bevor die Schwelle geprüft wird. Eine Anrede des Roboters ist somit jederzeit möglich.
Die Anbindung des Spracherkenners wurde im Rahmen einer Bachelorarbeit [Schnürer, 2015][27] durchgeführt und konnte bis zu einer demonstrierbaren Nutzung der Services für die Robotersteuerung, das Hauptmenü, die Wetterinformationen und den Kalender realisiert werden. Für die in Kap. 9 beschriebenen Nutzertests konnte die Spracheingabe allerdings noch nicht genutzt werden.

## 6.10 Outputgenerierung

Die letzte große Komponente im entwickelten Dialogsystem ist die Outputgenerierung. In klassischen sprach-basierten Dialogsystemen wird in dieser Komponente aus den Informationen des Dialogzustands und der Art der gewählten Aktion eine natürlichsprachliche Antwort generiert und anschließend

[27] Diese Arbeit wurde vom Autor betreut.

evtl. noch zu einer Sprachsynthesekomponente weitergeleitet. In einem multimodalen System, wie es hier umgesetzt wurde, spielen neben der reinen Textnachricht an den Nutzer noch viele weitere Aspekte eine Rolle. Die grafische Oberfläche muss entsprechend der Aktion umgeschaltet werden und die Hintergrundfunktionen der Service-Apps sind anzusteuern (beispielsweise eine Abfrage der Kalenderdatenbank nach Terminen). Außerdem können in verschiedenen Situationen unterschiedliche Navigationsverhalten des Roboters benötigt werden. Ein letzter wesentlicher Punkt ist die Kommunikation mit dem Dialogmanager selbst. In einzelnen Aktionen können neue Teildialoge aktiviert werden oder die Bearbeitung eines Teildialogs kann abgeschlossen und der entsprechende Dialog-Frame daher beendet werden.
Für all diese Teilaufgaben wurde ein erweiterbarer Interpreter umgesetzt, welcher die, in der Dialogkonfiguration definierten, Aktionsskripte ausführt. Die einzelnen Teilbereiche der Funktionen werden in unabhängigen, Outputfilter genannten, Plugins realisiert.

**Outputselektor** Die Aufgaben einer klassischen Ausgabegenerierung im sprach-basierten Dialogsystem, die Überführung von Botschaften in natürlichsprachlichen Text, übernimmt in diesem Dialogsystem eine eigene Komponente, der sogenannte Outputselektor (siehe Abb. 6.3 Seite 97). Die Generierung von Aussagen wird dabei aus den Aktionsskripten in den Framedefinitionen aufgerufen und dabei sowohl für die Anzeige auf dem Bildschirm als auch für die akustische Ausgabe genutzt.
Das Problem der „Natural Language Generation“ (NLG), sprich der Generierung von Text wird in der Literatur auf verschiedenen Wegen angegangen. Nach [McTear, 2004] reichen diese von einfachen, vordefinierten Texten (Canned Text) über template-basierte Ansätze hin zu anspruchsvollen akademischen Systemen, welche wirklich mehrstufig die Nachricht planen und letztendlich eine abstrakte Datenstruktur, die die Nachricht charakterisiert, schrittweise in einen Textstring umwandeln[28] [Reiter und Dale, 2000].
Aufgrund der vorliegenden Domäne der häuslichen Assistenzsysteme scheint eine effiziente Variante nach dem Prinzip der Canned Text Systeme ausreichend. Die Flexibilität des umgesetzten Systems ist außerdem durch die zur

[28] Dies wird eher für die Generierung längerer Texte eingesetzt.

Verfügung stehende, lediglich offline arbeitende, Sprachsynthese limitiert. Daher können die Sprachausgaben des Systems zur Zeit noch nicht dynamisch zur Laufzeit erstellt werden.
Um mit dem entwickelten System dennoch ein emotionales Auftreten des Roboters zu ermöglichen, wurde für die Umsetzung eine Variante gewählt, in welcher für jeden Typ einer Nachricht an den Nutzer eine Menge verschiedener Phrasen hinterlegt werden kann. Erstens kann dabei durch eine Zufallsauswahl eine größere Variabilität in den Systemausgaben erreicht werden, sodass die Nutzung des Roboters nicht zu schnell eintönig wird, und zweitens bietet die Beschränkung der nutzbaren Phrasen in Abhängigkeit von Zustandsvariablen des Systems die Möglichkeit, in den unterschiedlichen emotionalen Zuständen angepasste Sprachausgaben auszugeben, die auch prosodisch den gewünschten Emotionen entsprechen. Mehr dazu in Abschnitt 8.2.
Für eine Weiterentwicklung des Systems wäre die real-time Outputgenerierung ein interessanter Ansatzpunkt. Durch eine dynamische Synthese von Text und Sprache zur Laufzeit könnte die Flexibilität des Dialogsystems deutlich verbessert werden. Ein erster Schritt dazu ist die Erweiterung der Phrasen durch Platzhalter für Daten aus den Slots (Erweiterung zur template-basierten Generierung). Mittels des bereits implementierten Lexikons könnten die abstrakten semantischen Werte in den Slots in natürlichsprachliche Bezeichnungen übersetzt werden, welche anschließend in den vorgegebenen Textbaustein eingesetzt werden. Ohne eine Online-Sprachsynthese wäre dies allerdings nur für die geschriebenen Texte auf der grafischen Oberfläche möglich.

## 6.11 Zusammenfassung

Dieses Kapitel hat zunächst die Grundlagen für die Modellierung der Konversation zwischen Roboter und Nutzer als ein sequenzielles Wechselspiel von aktiven und passiven Phasen der Dialogpartner gelegt. Anhand der Entwicklungsgeschichte natürlichsprachlicher und multimodaler Dialogsysteme konnte eine Systemarchitektur für die Realisierung von frame-basierten Dialogen abgeleitet werden, welche sich an der klassischen dreiteiligen Sichtweise von Inputfusion, Dialogmanager und Outputgenerierung orientiert. Dabei wird diese gleichzeitig den speziellen Anforderungen bzgl. unabhängiger Definiti-

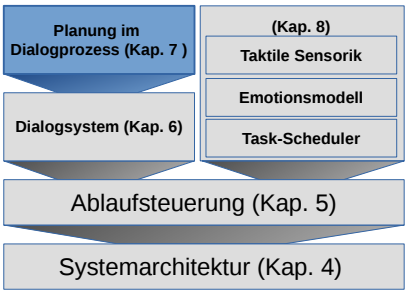
on der Dialoginhalte und der Bereitstellung von Interfaces zur Realisierung auch komplexerer, lernender Dialogstrategien gerecht. Eine probabilistische Fusion multimodaler, unsicherheitsbehafteter Nutzereingaben und Beobachtungen wurde im vorgestellten Inputinterpreter realisiert. Weiterhin wurde auf die Realisierungsdetails der Ablaufsteuerung im Dialog durch den Dialogmanager eingegangen und die multimodale Ausgabegenerierung des Systems vorgestellt.

Trotz der turn-basierten Dialogmodellierung liegt der Schwerpunkt der Lösung nicht auf sprach-basierten Dialogen. Sprache wurde lediglich als ein mögliches Eingabemedium genannt, wobei eine detaillierte Darstellung der Möglichkeiten zur Spracherkennung und zur semantischen Interpretation der Spracheingaben nicht im Fokus dieser Arbeit liegt. Gleiches gilt für die Modalität Sprache als Ausgabemedium des Roboters. Zur Bildung einer Persönlichkeit im Sinne der Verkörperung eines Dialogpartners wurden emotionale Sprachausgaben zwar als wichtig erachtet und genutzt, aufgrund der Komplexität dynamischer Textgenerierung kommt hier allerdings nur ein einfaches Verfahren mit vorsynthetisierten Texten zum Einsatz.

Die in diesem Kapitel vorgestellten Komponenten bilden das Kernsystem zum Führen von Mensch-Roboter Dialogen. Damit ist bereits eine Realisierung von nicht adaptiven Services im Rahmen der vorgeschlagenen Gesamtarchitektur eines Serviceroboters möglich. Die Realisierung des Demonstrators für den Gesundheitsassistenten (SERROGA), wie er letztendlich auch im Nutzertest eingesetzt wurde, erfolgte mit dem bis hierher beschriebenen Stand. Kap. 9 geht darauf noch einmal näher ein. Die Eingabemodalität Sprache konnte aufgrund der aufwändigen Infrastruktur mit externem Rechner und Ansteckmikrofon / Head-Set leider nicht mit den Senioren getestet werden.

Im Laufe der Entwicklung wurden neben den bis hier geschilderten grundlegenden Fähigkeiten jedoch weitere Teilfunktionen umgesetzt, welche in den folgenden Kapiteln dargestellt werden.

# 7




# Planung im Dialogprozess

Bis zu dieser Stelle wurde beschrieben, wie mittels des entwickelten Dialogsystems deterministische Dialoge realisiert werden können, wobei die Abläufe vollkommen durch den Entwickler vorbestimmt werden. Um den Ansprüchen der Arbeit, sprich der Realisierung von Nutzeradaptivität und Optimierung von Abläufen zur Laufzeit, gerecht zu werden, wird in diesem Kapitel erläutert, wie diese durch eine alternative Implementierung des Dialogagenten (siehe Abschn. 6.7 und Abb. 6.3 auf Seite 97) erreicht wurden. Im Dialog-Frame, wie er bei der Definition von Service-Apps vorgegeben wird, wurde bereits eine Policy durch einen Entscheidungsbaum festgelegt, welche für den gerade vorliegenden Zustand eine Menge zulässiger Aktionen bereitstellt. Der Dialogagent hat nun, in jedem Turn den der Dialogmanager startet, die Möglichkeit, frei aus dieser Aktionsmenge auszuwählen. An dieser Stelle setzt nun der Planungsprozess an, welcher in diesem Kapitel eingeführt werden wird. Eine Protokollierung der Auswirkungen der unterschiedlichen Aktionen in einem probabilistischen Zustandsübergangsmodell wird es ermöglichen, Aktionssequenzen bis zum Erreichen eines Zielzustandes zu Planen und dabei unter Zuhilfenahme eines Modells der Zufriedenheit des Nutzers (Rewardmodell) die Aktionsfolge dahingehend zu optimieren. Abb. 7.1 zeigt, wie sich diese Adaptionsmöglichkeit in die im Kapitel 3 eingeführte Zielstellung einfügt.

Das probabilistische Modell vom Dialogverlauf ermöglicht es weiterhin, Nut-

zereingaben situationsabhängig vorherzusagen. Dies kann nützlich sein, um langwierige, immer wiederkehrende Nachfragedialoge abzukürzen[1] und nach einer Lernphase gemäß dem Phasenmodell eines Langzeitdialogs (siehe Abschn. 3.3.1) ein proaktives Handeln des Serviceroboters zu ermöglichen.

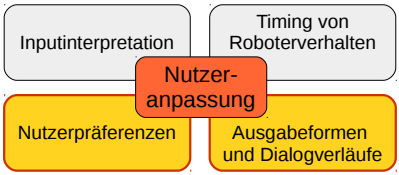



**Abbildung 7.1:** Behandlung der Aspekte Anpassung von Dialogverläufen und Ausgabeformen, sowie die Berücksichtigung von Nutzerpräferenzen

Die Modellierung des Dialogs mittels eines probabilistischen Modells und die darauf aufbauende Optimierung von Abläufen durch Planung zur Laufzeit wurde in [Müller et al., 2014b] und [Müller et al., 2014a] publiziert. Dabei erfolgte eine experimentelle Untersuchung im Rahmen einer studentischen Arbeit, welche die prinzipielle Funktionsfähigkeit des Dialogmodells und die Praktikabilität des Entwicklungsprozesses nachweisen sollte. Als Anwendungsszenario diente dabei aus verschiedenen Gründen nicht der SERROGA Gesundheitsassistent, sondern ein Helfer im Büro, genannt Office Mate. Im Rahmen dieses Kapitels wird daher dieser Office Mate als begleitendes Beispiel dienen. Am Ende dieses Kapitels, nach der Beschreibung der Funktion des planenden Dialogagenten, werden die Ergebnisse dieser Experimente kurz zusammengefasst und Schlussfolgerungen daraus gezogen (Abschn. 7.6).

## 7.1 Nutzeradaptive Dialogsteuerung im praktischen Test (Office Mate)

Die Implementierung eines multimodalen Dialogsystems, welches auf Basis eines probabilistischen Planungsprozesses online eine Dialogstrategie an das Nutzerverhalten adaptiert, wurde im Rahmen einer Masterarbeit [Sprenger, 2013][2] in einem Realweltszenario mit mehreren Nutzern getestet.
Aus Gründen der Verfügbarkeit der Roboterplattform (SCITOS-G3; vergleiche Abb.1.3) und damit gleichzeitig möglichst viele Nutzer am Langzeitexpe-

[1] Dies entspricht der Nutzung von Nutzerpräferenzen wie in Kapitel 3 geschildert.
[2] Diese Arbeit wurde vom Autor betreut.

riment teilnehmen konnten, wurde abweichend vom häuslichen Assistenzszenario eine Anwendung im Fachgebiet Neuroinformatik und kognitive Robotik der TU Ilmenau gewählt, wobei den Mitarbeitern täglich vom Roboter, genannt Office Mate, ein Besuch abgestattet wurde, bei dem er folgende Dienste anbot:

- Begrüßungsdialog
- Auswahldialog zu den weiteren Services (Hauptmenü)
- Präsentation verschiedener Nachrichtenportale aus dem Netz (Newspräsentation)
- Präsentation verschiedener Webcomics und Unterhaltungsseiten (Entertainment)
- Übersicht und Erinnerung fachgebietsweiter Termine
- Wetterauskunft
- Uhrzeit und Datumsauskunft
- Übersicht des aktuellen Speiseangebots der verschiedenen Mensen der TU Ilmenau
- Steuerung des Roboters
- Verabschiedung

Die Umsetzung der Applikation auf dem Roboter erfolgte in der genannten Arbeit unter Nutzung der bereits geschilderten Applikationsstruktur und des Dialogsystems innerhalb weniger Wochen, wodurch die Effizienz des hier vorgeschlagenen Entwicklungsprozesses sichtbar wurde. Die Durchführung der Nutzerexperimente mit dem Office Mate umfasste eine Eingangsbefragung mittels eines Fragebogens zur Erfassung von Vorkenntnissen und Präferenzen zur Roboterinteraktion, welche nicht sinnvoll durch Exploration zu bestimmen gewesen wären, ohne im Nutzereindruck einen Bruch in der Roboterpersönlichkeit zu erzeugen (Ansprache mit Du oder Sie, sowie männliche oder weibliche Roboterstimme). Außerdem wurden verschiedene Erwartungshaltungen an ein adaptives Robotersystem abgefragt. Die eigentlichen Interaktionen erfolgten über 2 Wochen (10 Arbeitstage) im Februar 2013, wobei 16 Probanden (4 weibliche, 12 männliche) an der Studie teilnahmen. Nach dem Robotereinsatz wurden die Probanden mittels eines zweiten Fragebogens nachbefragt, um Aussagen zur wahrgenommenen Adaptivität und Nützlichkeit der realisierten

Interaktionsform zu gewinnen (siehe Abschn. 7.7).

**Umsetzung des Testszenarios** Gemäß des Modells der Phasen im Langzeitdialog (Abschn. 3.3.1) wurde auch versucht, die Dialoge für das Testszenario anfänglich instruktiv zu gestalten. Insbesondere im Begrüßungsdialog kann der Roboter ein Tutorial zum Umgang mit der Oberfläche geben. Außerdem stehen die meisten Ausgaben des Systems in zwei Varianten zur Verfügung. Eine Funktionale sowie eine mit zusätzlichem Hilfetext (siehe Abb. 7.3). Im späteren Dialogverlauf soll durch den Lernprozess zu einem effektiveren Handling ohne Erklärungen und Auswahldialogen gewechselt werden.

Die Wandlung vom passiven zum proaktiven Verhalten wurde unter anderem im Rahmen des Hauptmenüdialoges[3] demonstriert und soll daher auch hier an diesem erläutert werden. Abb. 7.2 zeigt eine Übersicht zu den genutzten Variablen des Zustandsraumes und die Definition der Policy mittels des Entscheidungsbaumes. Aufgrund der Optionsknoten (grüne Kreise im Entscheidungsbaum) stehen für die Nutzerinteraktionen jeweils zwei Varianten zur Verfügung. Es gibt auch Fälle, in denen bis zu vier Aktionen verfügbar sind (die letzten vier gelben Aktionen in Abb. 7.2). Dabei kann durch den Roboter entweder eine Einführung bislang ungenutzter Servicefunktionen mittels eines Vorschlags geschehen (Aktion „Service vorschlagen“) oder die normale Auswahlfrage des als nächstes zu startenden Service wird dem Nutzer gestellt. Anfangs wird die Vorschlagaktion präferiert, bis diese vom Nutzer bestraft wird oder durch Ablehnung des Vorschlages ein unerwünschter Zustand auftritt.

In der Beobachtungsphase wird dem Nutzer die Initiative überlassen, aus dem Menü die gewünschten Services zu starten, wobei zunächst versucht wird, die Auswahl zu prädizieren (siehe Abschn. 7.5). Falls der gewählte Service bereits mit mindestens 50-prozentiger Sicherheit vorhergesagt werden kann, so wird nur eine binäre Bestätigung der Auswahl eingefordert (Aktion „Auswahl bestätigen lassen“). Bei über 80-prozentiger Sicherheit wird der entsprechende Service proaktiv gestartet. Dieses Vorgehen realisiert mit fortschreitender Menge an beobachteten Interaktionen den Übergang zu Phase 3 des Lang-

[3] Der Hauptmenüdialog wird im normalen Verlauf nach dem Begrüßungsdialog und zwischen den Nutzservices aktiviert.

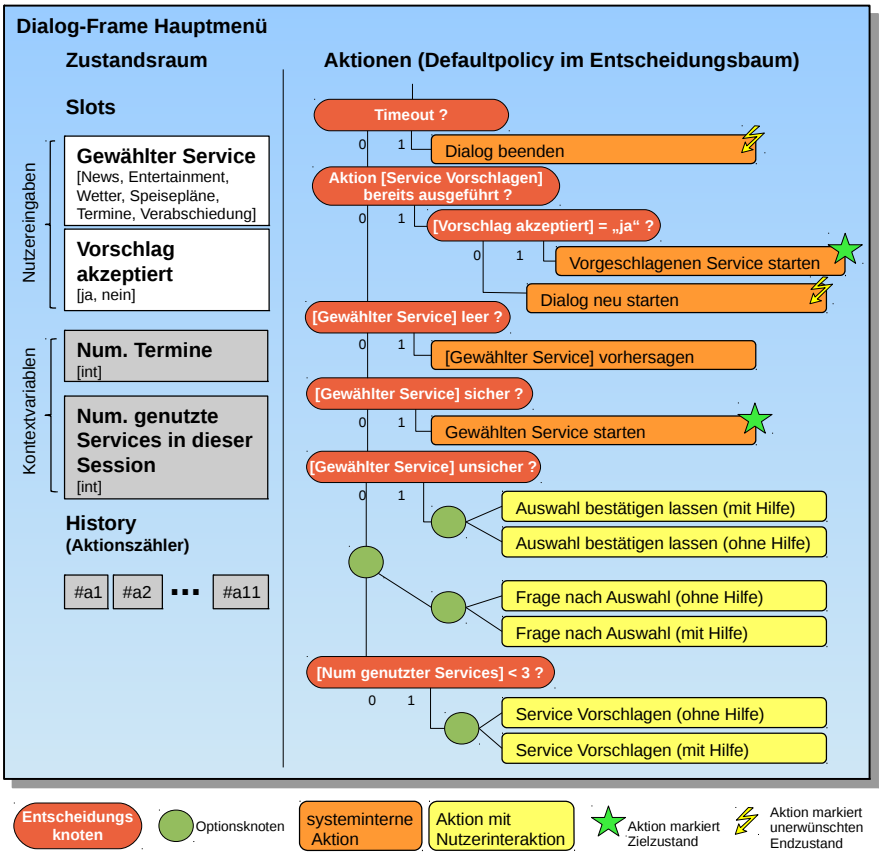



**Abbildung 7.2:** Realisierung des Hauptmenü Dialog-Frames in der Office Mate Applikation.

zeitdialogmodells (siehe Abschn. 3.3.1), in der das System die Entscheidungen vorhersieht und proaktiv Services anbietet.

Als Grundlage für die Entscheidung, welchen Service der Nutzer momentan bevorzugen könnte, dienen die Kontextvariablen im Zustandsraum des Dialog-Frames (Abb. 7.2 links). Da auf dem Hauptbildschirm die Anzahl der Termine bereits angezeigt wird, ist zu erwarten, dass der Nutzer abhängig davon den Kalenderservice aufrufen möchte oder nicht. Außerdem ermöglicht die Anzahl


**Abbildung 7.3:** Verschiedene Bildschirmseiten der Office Mate Applikation. Im oberen Bereich ist die Feedbackleiste sichtbar, über welche der Nutzer die Aktionen des Systems belohnen oder bestrafen kann. Die beiden letzten Seiten zeigen die Mensaspeisepläne einmal ohne und einmal mit eingeblendetem Hilfetext. Das Schließfeld des Hilfetextes wird als negatives Feedback gewertet. Bilder übernommen aus [Sprenger, 2013]

bereits genutzter Services das Erlernen einer Reihenfolge. Weitere Einflussgrößen wie Tageszeit (für den Speiseplanservice) wären außerdem denkbar,

wurden aber aufgrund der im Experiment begrenzten Anzahl möglicher Interaktionen weggelassen, da mit steigender Anzahl von Kontextvariablen auch die Exploration länger dauert.

Auf ähnliche Weise wurden die Auswahlprozesse in den Applikationen **Newspräsentation** und **Entertainment** realisiert. Hier wurde auch versucht, die gewählten Webseiten vorherzusagen oder anfänglich durch Vorschläge zu einer Erkundung der Möglichkeiten seitens des Nutzers anzuregen.

Der Begrüßungsdialog ist so gestaltet, dass er verschiedene Abfolgen (mit und ohne Smalltalk) zulässt, sowie optional das Tutorial vorgestellt wird. An diesem Beispiel sollte die Anpassung mittels Nutzerfeedback gezeigt werden. Dem Nutzer stehen dazu explizite Rewardvergabemechanismen zur Verfügung. Einerseits ist auf allen Bildschirmseiten am oberen Rand eine Skala von „gefällt mir“ bis „gefällt mir nicht“ sichtbar, über die eine Bewertung gegeben werden kann (siehe Abb. 7.3), und andererseits wurde der Roboter mit einem berührungssensitiven Streichelfell ausgestattet (siehe Abschn. 8.1.1), über das durch Streicheln positives Reward und durch einen Klaps negatives Reward vergeben werden konnte.

## 7.2 Probabilistische Modellierung des Dialogprozesses

Nachdem die angestrebten Fähigkeiten des planenden Systems geschildert wurden, soll nun auf deren Umsetzung näher eingegangen werden. Um im Allgemeinen Systeme zu beschreiben, deren Zusammenhänge inhärent unsicher sind, und über deren Zustände nur unsichere Messungen vorliegen, haben sich **probabilistische Modelle** bewährt. In diesen bilden die Systemgrößen Zufallsvariablen und die Zusammenhänge werden mit Hilfe von Wahrscheinlichkeitsverteilungen über diesen beschrieben. Anhang A 7.1 gibt eine nähere Einführung in die Grundlagen probabilistischer Modellierung, welche im Folgenden als bekannt vorausgesetzt werden.

### 7.2.1 Zustandsraum eines Dialog-Frames

Wie im Kapitel 6 zum Dialogsystem bereits geschildert, wird das Gesamtsystem in unabhängige Teildialoge heruntergebrochen. Diese werden auch unabhängig voneinander modelliert, wobei hierbei das Turnkonzept als Taktgeber fungiert. In jedem System-Turn liegt in den Dialog-Frames ein Zustand vor (definiert durch Inhalte der Slots und Historyzähler), welcher alle bisherigen Nutzereingaben und den groben Verlauf des Dialoges enthält. Nachdem das System seine diskrete Aktion ausgewählt und ausgeführt hat, wird nach möglichen, daraufhin stattfindenden, Nutzereingaben ein neuer Zustand erreicht.

Der **Zustand** des im Folgenden betrachteten Dialog-Frames sei mit $S$ bezeichnet und kann sehr hochdimensional werden (Zustandsvektor), da er alle Slotwerte $W$, die Konfidenzwerte $C$ der Slots, sowie die History in Form eines Zählers $Z$ für jede Aktion, die im entsprechenden Frame definiert ist, umfasst.

$$S = (W_1, \ldots, W_n, C_1, \ldots, C_n, Z_1, \ldots, Z_m) \tag{7.1}$$

Die Aktion $A$ entspricht in ihrem Wertebereich der Aktionsmenge des Frames, wobei in jedem Turn die tatsächlich erlaubte Menge $\hat{A}$ durch die Defaultpolicy im Entscheidungsbaum eingeschränkt ist. Im Beispiel des betrachteten Hauptmenü-Frames (Abb. 7.2 Seite 125) wäre der Zustandsvektor demnach 19-dimensional (4 Slots mit jeweils Wert und Konfidenz + 11 Zähler für die ausgeführten Aktionen).

Für die probabilistische Modellierung werden der Zustandsvektor $S$ und die Aktionen $A$ zu Zufallsvariablen mit ihren jeweiligen Realisierungen $S = s$ und $A = a$ genutzt.

### 7.2.2 Prozessmodell

Mit Hilfe der eben definierten Zufallsvariablen $S$ und $A$ kann ein probabilistisches Modell vom Dialogprozess unter Berücksichtigung der Markov-Annahme, dass Zustände nur vom unmittelbaren Vorgängerzustand und der Aktion abhängen, aufgestellt werden. Für $T$ Zeitschritte (Turns) soll mittels

dieser Unabhängigkeitsannahmen[4] gelten:

$$p(S_0, A_0, \ldots, S_{T-1}, A_{T-1}, S_T) = p(S_T|S_{T-1}, A_{T-1})p(A_{T-1}) \ldots p(S_1|S_0, A_0)p(A_0) \tag{7.2}$$

Eine bildliche Darstellung dieses Modells kann in Form eines Faktorgraphen [Jordan, 1998] gegeben werden, wie in Abb. 7.4 dargestellt. Faktorgraphen beschreiben alle Terme des probabilistischen Modells als Faktorknoten (rechteckig) und verbinden diese über Variablenknoten (Kreise), wobei ein Faktorknoten mit allen in seinem Term vorkommenden Zufallsvariablen verbunden wird. Eine kleine Einführung der Hintergründe dieser Modellierung ist in Anhang A 7.2 gegeben.
Die Faktoren $p(S_t|S_{t-1}, A_{t-1})$ bilden das sogenannte **Transitionsmodell**, welches die Statistik der Dialogverläufe enthält und im System während der Laufzeit anhand der erlebten Übergänge gelernt wird.

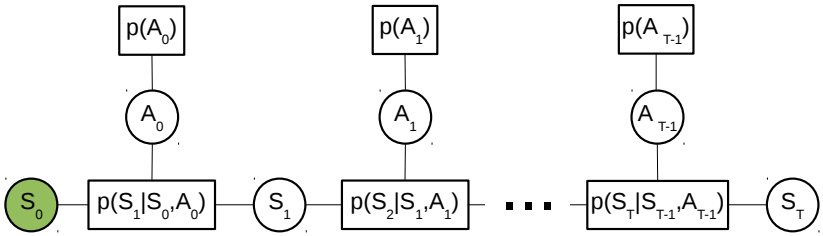



**Abbildung 7.4:** Einfaches probabilistisches Modell (Faktorgraph) des Dialogverlaufs mit T Turns; $S$ repräsentiert die Zustände des Dialog-Frames (Slotwerte und History) zu den verschiedenen Zeitschritten; $A$ steht für die jeweils gewählte Aktion; in grün der gegebene Ausgangszustand

Mit diesem Modell lassen sich bei einem gegebenen aktuellen Zustand $S_0$ bereits zukünftige Dialogverläufe vorhersagen. Für eine Planung zur Optimierung der Verläufe fehlt allerdings noch eine Zielvorgabe. Hierbei sollen einerseits im Sinne von Reinforcement-Lernen direkte Bewertungen der Verläufe durch den Nutzer (beispielsweise durch Streicheln belohnte Vorschläge

[4] mehr zum Unabhängigkeitsbegriff in Anhang A 7.2

für Services im Hauptmenüdialog) und andererseits systemintern definierte Zielzustände[5] berücksichtigt werden. Beispielsweise ist ein Zustand im Hauptmenübeispiel, in dem ein Service erfolgreich ausgewählt wurde und nun gestartet werden kann, ein Zielzustand (siehe Abb. 7.2). Ein Zustand, in dem ein Vorschlag abgelehnt wurde ist weniger erwünscht.

### 7.2.3 Rewardmodell

Die Bewertungen von erreichten Zuständen und damit auch der Dialogverläufe, welche in der History der Zustände kodiert ist, soll durch den Nutzer mittels eines bewusst abgegebenen Belohnungs- oder Bestrafungssignals, aber auch durch unbewusste im Robotersystem abgeleitete Nutzerbeobachtungen möglich sein. Hierzu kann beispielsweise taktile Interaktion wie Streicheln oder Klappsen des Roboters[6] genutzt werden. Auch ein Abbrechen der Interaktion seitens des Nutzers ist negativ zu werten (Timeout in Abb. 7.2). Weiterhin bietet es sich an, Verständnisschwierigkeiten und dadurch bedingte lange Reaktionszeiten des Nutzers als negatives Feedback zu werten, was allerdings noch nicht implementiert wurde.

Für die probabilistische Modellierung wird in dieser Arbeit der Reward $R$ in Abhängigkeit des Zustands $S$ mittels der bedingten Verteilung $p(R|S)$ modelliert (**Rewardmodell**). In das Modell werden positive und negative Rewardereignisse eingetragen, wodurch es ermöglicht wird, dass der Nutzer nicht in jedem Schritt einen Reward abgeben muss, da das System diesen gemäß dem Rewardmodell selbst bestimmen kann. Der Nutzer braucht somit nur Änderungen im Verhalten zu bewerten und das System gibt sich sozusagen bei zukünftigen Wiederholungen der Situation das Feedback selbst.

Bei Knox et al. [Knox und Stone, 2009],[Knox et al., 2013] kommt ebenfalls eine Art Modell des direkten Rewards vom Nutzer zum Einsatz. Allerdings wird in diesem Ansatz nicht, wie bei Reinforcement-Learning-Methoden üblich, eine Strategie gesucht, welche die zukünftig zu erwartenden Rewards maximiert, sondern die Autoren argumentieren, dass ein Feedback-Signal von

[5] Die systeminterne Wertung von Zuständen kann über die in diesen Zuständen ausführbaren Aktionen geschehen. Siehe Abb. 7.2 Zielzustand. Die Bedingungen für positive oder negative Zustandsbewertungen finden sich also im Entscheidungsbaum wieder.

[6] mehr zu eigenen Beiträgen zur taktilen Sensorik in Kapitel 8

einem Menschen bereits die Abschätzung der Erfolgsaussichten der Agentenstrategie enthält. Demnach wird in jedem Zeitschritt lediglich greedy die bestbewertete Aktion ausgewählt. Knox et al. sprechen daher von **Shaping** des Verhaltens, da mittels der Belohnungs- und Bestrafungssignale direkt das Agentenverhalten in der aktuellen Situation geändert wird.
Als Kritik an dieser Vorgehensweise bleibt die mangelnde Fähigkeit, auch andere komplexere Ziele zu verfolgen, die nicht durch den Menschen vorgegeben werden. Ein vorausschauendes Handeln wird damit nicht möglich sein. Im Gegensatz dazu bietet die in der vorliegenden Arbeit vorgeschlagene explizite Planung von Dialogaktionssequenzen auch die Möglichkeit, zielgerichtet über mehrere Schritte auf ein Belohnungssystem zuzuarbeiten und als Repräsentation der Erfahrungen dennoch nur ein zustandsbezogenes Modell des direkten Rewards zu verwenden.
Im Faktorgraph für die Dialogmodellierung wird demnach an jeden Zwischenzustand ein Faktor $p(R|S)$ angefügt (siehe Abb. 7.5).

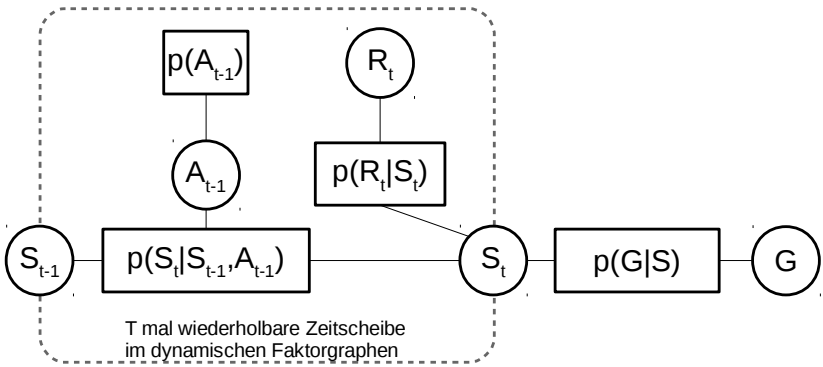



**Abbildung 7.5:** Dynamischer Faktorgraph für die Modellierung von zielgerichteten Dialogverläufen mit Berücksichtigung von Nutzerreward $R$ für die Zwischenzustände und Erwünschtheit $G$ des Endzustands ($G$ wie Goal).

### 7.2.4 Zielvorgaben

Neben der Erfüllung von Nutzerwünschen soll ein Assistenzroboter auch eigene Ziele verfolgen, bzw. das Handeln zusätzlich nach systeminternen Opti-

mierungskriterien verbessern. Beispielsweise könnte ein Ziel sein, den Nutzer zu einer Kooperation bei der Ausführung seiner Bewegungsübungen oder zur unbeliebten Messung von Vitalparametern zu bewegen. Ein weiteres zentrales Ziel ist außerdem die zügige Abarbeitung von Interaktionen, sprich das schnelle Erreichen eines Endzustands. Da diese Ziele sehr von den Inhalten der Service-Apps abhängen, müssen diese auch im Rahmen der Konfiguration der Dialog-Frames mit angegeben werden und können nicht zentral im System verankert werden. Es wird somit eine Möglichkeit vorgesehen, in den Aktionen des Roboters die aktuellen Zustände als wünschenswert oder unerwünscht zu markieren. Wird während der Interaktion ein solcher Zustand erreicht, so kann das System diesen in ein probabilistisches Modell der Ziele aufnehmen. Das System exploriert demnach, was wünschenswerte Zielzustände sind und welche vermieden werden sollen erst zur Laufzeit.
Analog zum Rewardmodell wird ebenfalls eine bedingte Wahrscheinlichkeitsverteilung genutzt, um dieses **Zielzustandsmodell** $p(G|S)$ darzustellen. Die Zufallsgröße $G$ steht hierbei für Goal, sprich wünschenswert oder nicht. Der Faktor $p(G|S)$ wird im Modell am Ende einer unbestimmt langen Kette von Aktionen und Zwischenzuständen stehen (siehe Abb. 7.5).

## 7.3 Auswahl optimierter Aktionen im Dialogagenten

Nachdem nun ein probabilistisches Modell des Dialogprozesses vorliegt, dessen Faktoren sich leicht anhand der im realen Einsatz erlebten Zustandsübergänge und Rewards lernen lassen, stellt sich nun die Frage, wie dies genutzt werden kann, um im Dialogagenten zu einer Entscheidung zu gelangen, welche Aktion am sinnvollsten auszuwählen ist.
Aufgrund der Tatsache, dass die Modelle anfangs unvollständig oder noch gänzlich unbekannt sind und sich auch im Laufe der Zeit ändern können, z.B. weil der Nutzer im Umgang mit dem Roboter dazulernt und nicht mehr auf Hilfestellungen angewiesen ist, besteht für das System das **Exploration-Exploitation Dilemma**. Sprich, einerseits soll versucht werden, gemäß dem gegenwärtigen Wissensstand eine optimale Aktion auszuführen, und andererseits wird es nötig, von Zeit zu Zeit alternative Aktionen zu erproben, um deren Erfolgsaussichten zu erkunden. Zusätzlich wäre es sinnvoll, das bekann-

te Modell der Entwicklung im Langzeitdialog explizit mit zu berücksichtigen. Es soll also einen Weg geben, die Auswahlwahrscheinlichkeiten für Aktionen mit Hilfestellungen im frühen Stadium und Aktionen, die proaktives Verhalten beinhalten, in einem späteren Stadium des Langzeitdialogs zu präferieren.

In der Umsetzung wird die finale Bewertung der verfügbaren Aktionen daher anhand dreier Faktoren vorgenommen. **Erstens** die aus dem probabilistischen Modell inferierbare Erfolgswahrscheinlichkeit[7], **zweitens** eine Statistik, wie oft eine Aktion bereits im gegenwärtigen Zustand gewählt wurde, um ein zielgerichtetes Explorieren zu ermöglichen, und **drittens** eine a-priori Verteilung über die Aktionen, welche die gegenwärtige Langzeitdialogphase berücksichtig.

Alle drei Faktoren werden mittels variabler Einflussfaktoren verrechnet (Details dazu in Anhang A 7.4). Aus der resultierenden Verteilung über den Aktionen wählt der Dialogagent letztendlich proportional zur Wahrscheinlichkeit zufällig aus und hat seine Aufgabe damit erfüllt.

## 7.4 Planung von Dialogsequenzen

Nachdem die Grundidee der Aktionsauswahl skizziert wurde, soll an dieser Stelle näher auf die Bestimmung der dabei benötigten Erfolgswahrscheinlichkeiten mittels Planung von optimalen Aktionssequenzen eingegangen werden. Abb. 7.6 zeigt, wie mittels des Modells eine beste Aktion und die Wahrscheinlichkeit dafür, dass diese letztendlich in einen Zielzustand führt, bestimmt werden kann. Als bekannt werden dabei der Startzustand $S_0$ sowie die gewünschte Verteilung über zu erwartende Rewards $R$ und die Tatsache, dass für einen Endzustand der Erwünschtheitswert hoch sein soll (Vorgabe einer Verteilung über $G$), angenommen. Unbekannt ist dagegen die Anzahl der Zwischenschritte sowie die dabei entstehenden Zwischenzustände und die jeweils zu wählende Aktion. Von Bedeutung für den Dialogagenten ist allerdings nur die unmittelbar als nächste auszuwählende Aktion (in Abb. 7.6 rot markiert). Prinzipiell besteht die Aufgabe des Systems nun darin, die bekannten Zufalls-

[7] Diese berücksichtigt auch die Nutzerzufriedenheit.

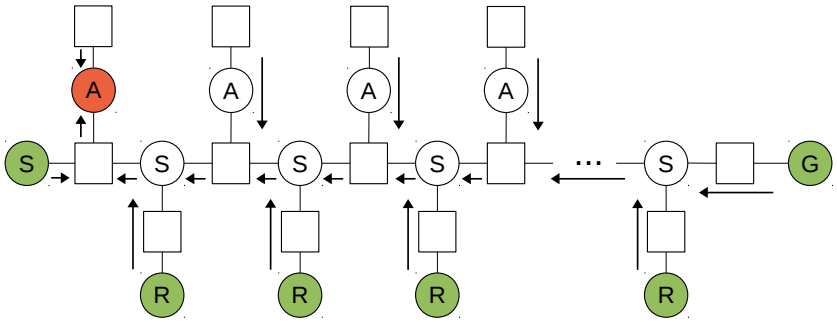



**Abbildung 7.6:** Ausgefalteter Faktorgraph für die Planung von Dialogverläufen. Grüne Größen sind bekannt und die rot markierte Aktion ist gesucht. Die Pfeile repräsentieren den bei der Berechnung anfallenden Informationsfluss. Beobachtung: auch die Pfeilrichtungen gleichen sich in den Zeitscheiben und zeigen an einem Ende hinein und am anderen Ende heraus.

größen in das Modell einzusetzen und anschließend

$$a_{max} = \underset{A_0}{\operatorname{argmax}}\, p(S_s = s_0, A_0, S_1, A_1, R_1 = r_1, ..., S_T, A_T, R_T = r_T, G_T = g_T) \tag{7.3}$$

zu berechnen. Da dies aufgrund der Komplexität der Modelle und aufgrund des unbekannten Zeithorizontes $T$ nicht ohne weiteres möglich ist, wurde im Rahmen dieser Arbeit eine spezialisierte Variante des Max-Product-Algorithmus (siehe Anhang A 7.3.2) entwickelt. Diese ist auf die Verwendung einer ebenfalls speziell für die Darstellung der hochdimensionalen Zustandsverteilungen entwickelten spärlichen Verteilungsrepräsentation optimiert. Der ursprüngliche Max-Product-Algorithmus [Bishop et al., 2006] dient eben zur Inferenz der maximalen Wahrscheinlichkeit in einem Faktorgraphen mittels des Prinzips des Messagepassing[8].

Da der Planungsalgorithmus und die spärliche Verteilungsrepräsentation einen zentralen Beitrag dieser Arbeit darstellen, wird im Folgenden versucht, zu motivieren, wie dabei vorgegangen wird. Eine mathematische Beschreibung

[8] Eine Ausprägung des Max-Product-Algorithmus im Rahmen von Hidden Markow Models (HMMs) ist unter dem Namen Viterbi-Algorithmus bekannt.

und der exakte Algorithmus sind für den interessierten Leser in Anhang A 7.4 zu finden.

### 7.4.1 Planungsalgorithmus

Wie bereits erwähnt, ist die Länge der Aktionssequenzen bis zu einem Zielzustand im Vorhinein nicht bekannt und somit auch die Länge des ausgefalteten Faktorgraphen. Die eben gefundene Beobachtung, dass Informationen nur von potentiellen Zielzuständen aus in Richtung der gesuchten Aktion $A_0$ fließen (siehe Abb. 7.6), legt es nahe, die Berechnungen der Planung in einer Schleife auszuführen und somit den Planungshorizont schrittweise in die Zukunft zu verschieben[9]. Währenddessen ergeben sich für die möglichen Aktionen jeweils Wahrscheinlichkeiten für das Erreichen eines Ziels, über welche lediglich das Maximum über der Planungstiefe ermittelt werden muss. Abb. 7.7 zeigt den Ablauf am Faktorgraphen.

Ausgangspunkt ist die Menge aller bekannten wünschenswerten Zielzustände, wie sie im Zielzustandsmodell $p(G|S)$ abgelegt sind. Die Planung läuft quasi vom Ziel zum aktuellen Zustand. Durch Einsetzen einer sigmoidalen Gewichtung für $G$ (siehe Abb. 7.7 ganz rechts), welche wünschenswerte Zustände bevorzugt, erhält man aus dem Zielzustandsmodell eine Verteilung über Zustände $S_t$, in der mögliche Endzustände eine hohe und unerwünschte Zustände eine geringe Wahrscheinlichkeit haben (in Abb. 7.7 Schritt (1) ). Mit dieser Verteilung kann der Planungsprozess beginnen. Ziel ist es nun, schrittweise zu fragen, aus welchen Zuständen man mit welchen Aktionen wie wahrscheinlich in diese Zielzustände gelangen kann. Dazu beginnt die Schleife mit Planungshorizont eins.

Als erstes werden mögliche Nutzerrewards einbezogen. Dazu wird für den Reward $R$ ebenfalls eine sigmoidale Gewichtung angesetzt und das Rewardmodell $p(R|S)$ mit dieser nach einer Verteilung über die Zustände befragt, welche vom Nutzer bevorzugt werden. In Abb. 7.7 ist dies als Schritt (2) zu sehen. Die im Variablenknoten $S_t$ vorhandene Verteilung über die Zustände wird nun mit der vom Rewardmodell kommenden multipliziert, wodurch die

[9] Würden die maximalen Wahrscheinlichkeiten aller Variablen von Interesse sein, so müssten Informationen in beide Richtungen fließen. Die Berechnung würde dann das Speichern aller Zwischenergebnisse der Zeitschritte erfordern.

Zustände entsprechend der Rewards gewichtet werden.
Der nächste Schritt (3) besteht im Anschluss darin, mittels des Transitionsmodells $p(S_t|S_{t-1}, A_{t-1})$ nachzuschlagen, welche Vorgängerzustände für die Zustände in $S_t$ in Frage kommen. Diese werden dann in Schritt (4) mit dem aktuellen Zustand $s_0$ verglichen und die Wahrscheinlichkeiten der zu den entsprechenden Transitionen von $s_0$ zu $S_t$ gehörenden Aktionen $A_0$ bestimmt. Hierbei kann auch eine a-priori Wahrscheinlichkeit für Aktionen berücksichtigt werden, um beispielsweise die von Dialogphasen abhängigen Prioritäten von Aktionen zu realisieren (vergleiche Abschn. 7.3).
Diese resultierenden Aktionswahrscheinlichkeiten werden für den zur Zeit betrachteten Planungshorizont gespeichert, um im Anschluss an die Schleife das Maximum über alle Planungstiefen bestimmen zu können.
Um den Planungshorizont zu vergrößern, wird gemäß Schritt (5) die A-Priori-Verteilung über die Aktionen mit der Zustandsverteilung $S_t$ und dem Transitionsmodell kombiniert. Dabei ergibt ich die Verteilung über die möglichen Vorgängerzustände $S_{t-1}$, welche wieder zum Knoten $S_t$ geschickt wird, wo sie als Ausgangspunkt für den nächsten Planungsschritt dient (Schritt (6) ). Dort beginnt erneut die Gewichtung mit den Nutzerrewards und die Bestimmung der Erreichbarkeit aus dem vorliegenden Zustand $s_0$, wie soeben geschildert. Die Schleife wird nach einer maximalen Planungstiefe abgebrochen und es resultiert die gesuchte Verteilung über der Aktionsmenge, welche die Wahrscheinlichkeiten ausdrückt, **mit Aktion $a$ von Zustand $S_0 = s_0$ nach einer Sequenz von weiteren Aktionen in einen Zielzustand zu gelangen und auf dem Weg dort hin den Nutzerreward zu maximieren**.
Diese Verteilung über $A_0$ wird im Dialogagenten wie in Abschn. 7.3 beschrieben zur Aktionsauswahl genutzt.
Der tatsächlich implementierte Algorithmus nutzt zur Ausführung der verbal beschriebenen Schritte die Rechenvorschriften des Max-Product-Algorithmus für die Bestimmung der maximalen Wahrscheinlichkeit in einem Faktorgraphen. Anhang A 7.4 beschreibt dazu die weiteren Details. Gegenüber einer herkömmlichen Inferenz in einem Faktorgraphen, ist am hier vorgestellten Verfahren die Nutzung komplexer Zustände und die Ausführung in einer Schleife zur Behandlung des unbekannten Zeithorizonts bis zum Ziel originär. Ermöglicht wird dieses Vorgehen allerdings erst durch die Verwendung einer besonders ef-

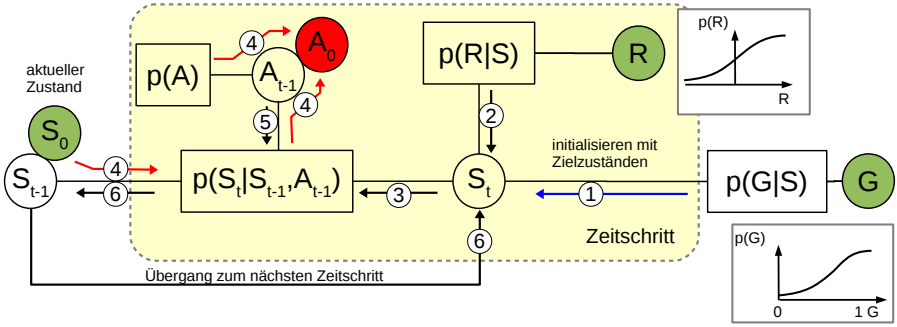



**Abbildung 7.7:** Loop-Trick für die Berechnung des Max-Product-Algorithmus bei unbekannter Planungstiefe. Die Verteilung des Zielzustandes wird anhand $p(G|S)$ initialisiert (1). Anschließend beginnt die Iteration mit den Schritten (2)-(6). Siehe Text für eine Erläuterung.

fizienten Repräsentationsform für die hochdimensionalen Wahrscheinlichkeitsverteilungen sowohl in den Modellen, als auch in den Zwischenergebnissen (Messages). Im folgenden Abschnitt soll daher diese Verteilungsmodellierung kurz erläutert werden.

### 7.4.2 Spärliche Verteilungsrepräsentation

Im Inferenzalgorithmus wurde abstrakt über Wahrscheinlichkeitsverteilungen von Zuständen gesprochen. Konkret bestehen die Zustände eines Dialog-Frames wie in Abschnitt 7.2.1 eingeführt aus mehreren Größen (Zustandsvektor) mit teilweise überabzählbarem und anfangs unbekanntem Wertebereich. Eine herkömmliche Repräsentation der Verteilung als Tabelle lässt sich daher nicht anwenden. Die Wahl fiel deshalb auf eine spärliche Repräsentation der Verteilungsdichte mittels Samples. Das heißt, die ohnehin endlich vielen diskreten Beispiele, welche der Roboter erlebt, werden als Liste von eben diesen Zustandsvektoren verwaltet, wobei jedem Sample ein Gewicht zugeordnet ist. Ein Update eines Modells ist daher recht einfach durch Hinzunahme eines neuen Samples möglich[10]. Für die Inferenz ist es jedoch aus Gründen der Generalisierungsfähigkeit wichtig, dass auch über noch nicht

[10] Bei Überschreiten einer Maximalanzahl von Samples lassen sich die Modelle leicht durch Zusammenfassen von Verteilungsmasse (Gewichten) ausdünnen.

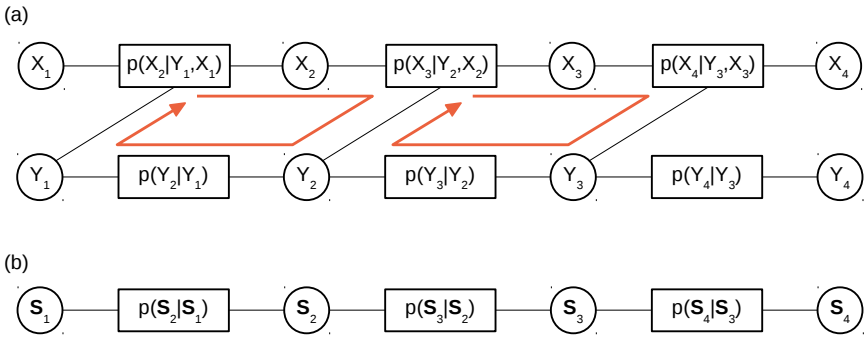



**Abbildung 7.8:** (a) nicht mehr kreisfreier (rote Pfeile) dynamischer Faktorgraph bei mehreren Variablen zur Zustandsrepräsentation ($X$ und $Y$ anstatt $S$). Inferenz ist nur noch mit Loopy Belief Propagation möglich. (b) Zusammenfassung der Variablen $X$ und $Y$ zu einem Zustandsvektor $\mathbf{S}$ verhindert die Entstehung von Kreisen im Faktograph. Inferenz ist durch deterministisches Propagieren der Informationen entlang der Kanten zur gesuchten Größe möglich (Message Passing).

direkt beobachtete Zustandsvektoren eine Wahrscheinlichkeitsaussage getroffen werden kann. Daher wird ähnlich einer Kernel-Density-Estimation mittels einer Ähnlichkeitsfunktion (diese übernimmt die Funktion der Kernelfunktion) auch für Zustandsvektoren, die bekannten Samples benachbart sind, eine Aussage zur Wahrscheinlichkeit ermöglicht. Anhang A 7.3.4 führt die Mathematik für diese Darstellung genauer ein, wobei insbesondere die für den Max-Product-Algorithmus benötigten **Operationen Marginalisierung, Konditionierung, Produkt und Maximierung** speziell für die Verwendung im Faktorgraphen optimiert wurden. Letztendlich werden fast alle Operationen durch eine Neuberechnung der Gewichte zu den festen Stützstellen (Samples) realisiert, wodurch eine hohe Effizienz erreicht wurde.

Originär an der gewählten Darstellung ist außerdem die Aufnahme von Variablen gemischter Typen (diskrete und reellwertige Variablen) und unbekannter Wertebereiche in einen Vektor, welcher bei der probabilistischen Modellierung als **eine Zufallsvariable** betrachtet wird. Dadurch wird im Faktorgraphen vermieden, dass zyklische Strukturen entstehen, und somit ist eine deterministische Inferenz erst möglich. Würden die einzelnen Größen des Zustandsvek-

tors als individuelle Zufallsvariablen modelliert, so könnten zwar weitere Unabhängigkeitsannahmen eingeführt werden, um das Modell zu vereinfachen[11] und auch die Generalisierungsfähigkeit zu verbessern, aber die Inferenz im so entstehenden, nicht mehr kreisfreien, Faktorgraphen könnte nur noch näherungsweise mittels iterativer Verfahren (Loopy Belief Propagation) erfolgen. Abb. 7.8 zeigt diesen Sachverhalt an einem vereinfachten Beispiel, wobei ein einfaches Transitionsmodell $p(S_t|S_{t-1})$ angenommen wird.
Mit dem bis zu dieser Stelle beschriebenen Modellierungsansatz und mittels des Planungsalgorithmus aus Abschn. 7.4.1 lassen sich nunmehr Dialogverläufe individualisieren. Abhängig von durch den Nutzer geäußerten Vorlieben, kann das hier entwickelte Dialogsystem bereits die in Abschn. 3.4 geforderte Anpassung der Ausgabeformen und Dialogverläufe vornehmen. Bleibt als weitere Adaptionsmöglichkeit die Umsetzung der Berücksichtigung von Nutzerpräferenzen, wie bereits im Beispiel angedeutet (anstatt nach einer Auswahl von Webseiten oder Services zu fragen, kann anhand der vormals getroffenen Entscheidungen die Auswahl vorhergesagt und damit abgekürzt werden).

## 7.5 Nutzerpräferenzen

Mit Hilfe des Zielzustandsmodells $p(G|S)$ aus Abschn. 7.2.4 besteht die Möglichkeit, Entscheidungen des Nutzers anhand der vergangenen Nutzereingaben vorwegzunehmen und somit gegebenenfalls effizientere Abläufe zu ermöglichen oder im Rahmen der probabilistischen Inputfusion unsichere Eingaben anhand der A-Priori-Verteilung der Antwortmöglichkeiten zu bestätigen. In einem erfolgreichen Endzustand eines Dialoges sollten die benötigten Eingaben vollständig und somit, neben den jeweils betrachteten Größen, auch die als Kontext benötigten Zustandsgrößen vorliegen (beispielsweise die bereits konsumierten Webseiten, um die als nächstes gewählte vorhersagen zu können).
In den Ablauf des Dialoges ordnet sich die Vorhersage von Eingaben in Form spezieller Dialogaktionen ohne Nutzerinteraktion ein. Im Beispiel des Hauptmenü-Dialoges aus dem Office Mate Szenario (siehe Abb. 7.2 Seite 125)

[11] In Abb. 7.8 hängt $Y_t$ nicht von $X_{t-1}$ ab, aber $X_t$ hängt sowohl von $X_{t-1}$, als auch von $Y_{t-1}$ ab.

ist dieses die „[Gewählter Service] vorhersagen“ Aktion, welche ausgeführt wird, wenn im Dialog die Beschaffung der Nutzereingaben zum Slot [Gewählter Service] ansteht.
Wenn diese Aktion ausgeführt wird, werden die gespeicherten Zielzustände im Zielzustandsmodell $p(G|S)$ mittels der Ähnlichkeit zum aktuellen Zustand $s_0$ neu gewichtet. Dabei werden nur die Zustandsdimensionen berücksichtigt, die den Kontext darstellen. Andere Dimensionen, wie die History, werden ignoriert, da sich diese sicherlich signifikant von $s_0$ unterscheiden werden. Anhand der so neu bewerteten Samples kann eine situationsabhängige Wahrscheinlichkeitsverteilung über die möglichen Eingaben für den benötigten Slot ermittelt werden. Bei hinreichender Signifikanz (Mindestanzahl von Beispielen) bekommt der Inputinterpreter diese Verteilung als Beobachtung und kann sie äquivalent zu direkten Nutzereingaben in die Schätzung des Eingabezustands übernehmen. Gegebenenfalls erfolgt daraufhin die Übermittlung eines semantischen Wertes mit Konfidenz an dem Dialogmanager, damit dieser den Slot füllen kann.
Das genaue Vorgehen bei der Vorhersage von potentiellen Eingaben wird in Anhang A 7.5 geschildert.

## 7.6 Ergebnisse der Office Mate Studie

Nachdem die Umsetzung des probabilistisch planenden Dialogsystems erläutert wurde, soll an dieser Stelle der Bogen zu dem am Anfang dieses Kapitels (Abschn. 7.1) geschilderten Experiment Office Mate geschlagen werden. Wie bereits erwähnt, wurden dabei die Probanden täglich vom Roboter besucht und konnten die Interaktion mit dem System erproben. Dabei waren die in diesem Kapitel geschilderten Adaptionsmechanismen aktiv.
Die Besuche des Roboters bei den Probanden wurden von einem Beobachter begleitet, der aufgetretene Probleme seitens des autonomen Roboters und die Nutzungsform durch die Probanden protokollierte. Daher muss davon ausgegangen werden, dass das Interaktionsverhalten der Probanden in mancher Hinsicht positiver ausgefallen ist, als es bei einer freien Nutzung der Fall gewesen wäre. Ebenso wurde der konkrete Funktionsumfang im Nachhinein als nicht ausreichend dafür erachtet, einen Roboter als nützliche Unterstützung

wahrzunehmen. Im Büro standen den Probanden die gleichen Services über ihren PC zur Verfügung. Für den Nachweis der korrekten Funktion des Adaptionsmechanismus sollte dies allerdings keine negativen Auswirkungen haben. Bezüglich des Bedienkonzepts (turn-basierter Dialog über eine grafische Benutzeroberfläche und multimodale Ausgaben als Text und Sprache) kamen alle Probanden damit zurecht, allerdings waren einige Details der Gestaltung in ihrer Funktion nicht intuitiv genug. Details dazu sind in [Sprenger, 2013] ausgeführt. 15 von 16 Probanden gaben an, dass der Dialog logisch aufgebaut und nachvollziehbar war. Ebenso bezeichneten 15 Probanden die Interaktion mit dem Roboter als intuitiv.

Bezüglich der Adaptivität werden zunächst die subjektiven Eindrücke der Nutzer reflektiert. Hierbei wurde mit Ausnahme eines Nutzers von allen bestätigt, dass eine Veränderung des Roboterverhaltens über die Zeit beobachtbar gewesen sei. Dabei ist zu berücksichtigen, dass drei der Probanden lediglich an 5 oder weniger Tagen Kontakt mit dem Roboter hatten. Ebenfalls wurde von 12 Kandidaten die Veränderung als positiv gewertet. Auch der Umfang der Anpassung sollte bewertet werden, wobei 9 Probanden angaben, mehr erwartet gehabt zu haben. Sechs gaben an, dass ihre Erwartungen bzgl. der Anpassungsfähigkeit erfüllt wurden. Neben den Erkenntnissen aus den Fragebögen fiel auf, dass mehrere Nutzer durch die inkonsistente, zufällige Verhaltensänderung während der Exploration von alternativen Verläufen irritiert wurden. Sie kritisierten, dass bereits einmal bestraftes Verhalten, welches danach vom System auch scheinbar bereits abgestellt wurde, nach mehreren Durchläufen dennoch erneut präsentiert wurde. Dies entspricht dem vorgesehenen Systemverhalten und ist dem Exploration-Exploitation-Dilemma geschuldet. Außerdem gaben mehrere Nutzer an, mit den expliziten Belohnungs- und Bestrafungsmöglichkeiten nicht zurecht zu kommen. Ihnen fehlte die Zuordnung des Feedbacksignals zu einzelnen Aspekten des Gesamtverhaltens des Roboters (Belohne ich die Anzeige, die Wortwahl oder die Lautstärke?). Außerdem fällt die Bewertung einer Aktion schwer, wenn die Handlungsalternativen nicht bekannt sind.

Aus objektiver Sicht (resultierend aus den Beobachtungen sowie aus der Auswertung der protokollierten Interaktionsverläufe) lässt sich die bestimmungsgemäße Funktion des Dialogsystems und der Anpassungsmechanismen nach-

vollziehen. Einerseits konnte durch Abbrechen des Tutorials sowie durch Bestrafung der Smalltalk-Aktionen der Verlauf des Begrüßungsdialoges an die Wünsche der Nutzer angepasst werden. Andererseits prägten sich 10 verschiedene Nutzungsmuster aus, welche die Reihenfolge der Services Wetter, Speiseplan, News und Terminkalender betreffen. Bei konsistenter Nutzung konnten vom System die Reihenfolge vorhergesagt und die Services proaktiv gestartet werden.
Es war zu beobachten, dass die Nutzungsdauer über die Zeit abnahm. Dies könnte einerseits auf schwindendes Interesse am Roboter, oder auch auf eine Effektivierung der Abläufe zurückzuführen sein. Als Haupteinflussfaktor stellten sich allerdings die tagesabhängigen Inhalte der betrachteten Webseiten heraus.

## 7.7 Schlussfolgerungen aus dem Experiment

Die Office Mate Studie hat gezeigt, dass der vorgeschlagene Modellierungsansatz der Dialoge prinzipiell funktioniert, für die Entwickler praktikabel und für den Einsatz auf einem mobilen Assistenzroboter angemessen ist. Er sollte demnach möglichst beibehalten werden. Die konkrete Gestaltung der Bildschirmoberflächen und der Form, in der zusätzliche Hinweise präsentiert werden, muss im realen Einsatz noch überarbeitet werden.
Aus den beobachteten Schwierigkeiten im Umgang mit explizit und bewusst zu gebendem Feedback sollte eine Konzentration auf implizites Feedback sowie auf unbewusste Kanäle abgeleitet werden. Der explizite Bewertungsbalken auf dem Bildschirm ist somit nicht zukunftsfähig. Alternativen zum bewussten Feedback wären in der Analyse von Mimik (Stirnrunzeln, Lächeln, Irritation) und in der Auswertung von Parametern des nutzerseitigen Interaktionsverhaltens zu suchen. Hier könnte die Reaktionsgeschwindigkeit bei Bildschirmeingaben oder auch die Schätzung von Stress und Emotionen anhand der Stimme (wie in [Prüger, 2008][12] demonstriert) bei Nutzung von Spracheingaben herangezogen werden. [Eyben et al., 2012] zeigen dass die Erkennung von positiven und negativen Emotionen aus Sprachsignalen auch unter schlechten Bedingungen möglich ist.

[12] Diese Arbeit wurde vom Autor betreut.

Auch die taktile Interaktion mit dem Roboter (z.B. über das Streichelfell [Müller et al., 2015] oder die berührungssensitive Außenhülle [Müller et al., 2013]) sollte nicht nur als direktes bewusstes Feedback, sondern über eine Auswertung der Statistik auch als unbewusstes Signal für generelle Zufriedenheit genutzt werden.

Schwerwiegender als die nicht intuitive Rewardvergabe ist die wahrnehmbare Inkonsistenz im Verhalten, wenn neue Aktionen exploriert werden. Das hängt mit der heuristischen Verrechnung der drei Einflussgrößen 1) geplante Aktion, 2) Statistik der Häufigkeiten für die Exploration und 3) dialogphasenabhängige a-priori Wahrscheinlichkeit für Aktionen zusammen, wie in Abschn. 7.3 geschildert. In Zukunft sollte diese Heuristik durch eine ganzheitliche Lösung, welche sich im Modell wiederfindet, ersetzt werden. Die Aktionsauswahl sollte dann nur noch der inferierten Verteilung folgen.

Das Explorationsproblem als erstes Teilproblem wäre dabei sinnvollerweise durch eine optimistische Intitialisierung anzugehen, wobei alternative, bislang nicht beobachtete Zustandsübergänge bereits von Beginn an eine gewisse Wahrscheinlichkeit besitzen müssten. Für ein bislang unbeobachtetes $S_t$ und $S_{t-1}$ würde somit eine Gleichverteilung über die Aktionen entstehen, aus welcher bei der Aktionsauswahl gezogen würde. Somit wäre eine Exploration gesteuert durch die Entropie im Modell. Wo viel Unsicherheit herrscht, wird viel exploriert und wo sichere Beobachtungen vorliegen, wird versucht, nach Plan zu agieren, um möglichst viel Reward zu bekommen. Durch ein zeitgesteuertes Vergessen von erlebten Dialogen könnten die lebenslange Flexibilität und gelegentliche erneute Exploration realisiert werden. Leider ist dies alles mit der sample-basierten Modellierung der Verteilungen, wie sie hier vorgestellt wurde, nur schwer umsetzbar. Eine denkbare Möglichkeit bestünde etwa in einer zufällige Mutation der Samples während der Planung, würde aber weitere Überlegungen erfordern.

Die zweite Komponente, die Berücksichtigung einer dialogphasenabhängigen a-priori Wahrscheinlichkeit für die Aktionen, könnte über einen veränderlichen Faktor $p(A)$ im Faktorgraphen realisiert werden (vergleiche Abb. 7.7). Bislang steht an dieser Stelle lediglich eine Gleichverteilung. $p(A)$ würde damit zu $p(A|D)$, wobei das gegebene $D$ die Langzeitdialogphase beschreiben müsste. Um dies zu realisieren, wäre eine Online-Schätzung des Erklärungs-

bedarfs bzw. des Fortschritts im Phasenmodell des Langzeitdialogs (vergleiche Abschn.3.3.1) erforderlich. Anhand der durch den Nutzer abgebrochenen Erklärungsmonologe des Systems und der Trefferquote von vorhergesagten Nutzereingaben (siehe 7.5) könnte auch dies versucht werden.

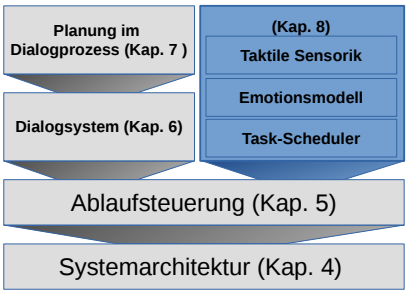
## 7.8 Fazit

In diesem Kernkapitel der Arbeit wurden einerseits die Grundlagen für eine nutzeradaptive Gestaltung von Dialogverläufen gelegt und andererseits anhand eines Realweltexperiments (Office Mate) mit einer praktischen Implementierung des vorgeschlagenen Planungsalgorithmus der Nachweis erbracht, dass dieser funktionsfähig und prinzipiell praktikabel ist.
Mittels des eigens entwickelten, in einer Schleife laufenden Planungsalgorithmus auf Basis des Max-Product-Algorithmus, konnte die explizite Ausfaltung des genutzten Faktorgraphen über die Zeitschritte des, von vorn hinein unbestimmten, Planungshorizontes überwunden werden, was einen praktischen Einsatz der Methode erst ermöglicht. Für die Berechnungen und für die Beschreibung der beteiligten Modelle (Wahrscheinlichkeitsverteilungen) wurde eine eigene auf Samples basierende Darstellungsform für Verteilungen vorgestellt. Durch die Samples wird dabei in einem potentiell sehr hochdimensionalen Raum ein diskreter Wertebereich erzeugt, über dem, mittels den Samples zugeordneter Gewichte, eine Verteilung definierbar ist. Die Berücksichtigung einer Ähnlichkeitsfunktion im hochdimensionalen Raum ermöglicht bei den Berechnungen eine gewisse Generalisierungsfähigkeit, selbst wenn der hochdimensionale Zustandsraum nur spärlich mit Samples besetzt ist.
Die vorgeschlagene Modellierung des Wissens als einfache Verbundverteilungen[13] ermöglicht es, zusammen mit der in jedem Dialogschritt ablaufenden Online-Planung, im Gegensatz zu anderen Reinforcement-Learning basierten Ansätzen, jederzeit sofort das vollständige Wissen aus der erlebten Vergangenheit zu nutzen. Dazu werden die Modelle lediglich um die neu erlebten Zustandsübergänge und Rewards ergänzt.
In Erweiterung zu den deterministischen Dialogen, welche mit dem in Kapitel

[13] Die Konditionierung zu $p(S_t|S_{t-1}, A)$ erfolgt erst beim Update und wird parallel zu $p(S_t, S_{t-1}, A)$ als zweites Set von Samplegewichten gehalten.

6 beschriebenen System realisierbar sind, konnten in diesem Kapitel die in Abschnitt 3.4 versprochene Anpassung der Ausgabeformen und Dialogverläufe sowie die Nutzung von Nutzerpräferenzen, welche direkt aus den genutzten Modellen ablesbar sind, realisiert werden.

# 8




# Weitere Komponenten zur Realisierung eines robotischen Assistenzsystems

In den Kapiteln 5 und 6 erfolgte die nähere Vorstellung der zentralen Module für die Ablaufsteuerung und das Führen des Mensch-Maschine-Dialoges. Für ein funktionales Gesamtsystem sind neben diesen noch weitere Komponenten nötig und im Rahmen der Projekte SERROGA und CompanionAble entwickelt worden. Zum einen zählen dazu Hardwarekomponenten, wie das bereits erwähnte Streichelfell und die kapazitive Touchsensorik. Zum anderen spielen Softwarekomponenten, wie ein Emotionsmodell, die grafische Nutzeroberfläche und ein Taskscheduler, eine Rolle bei der Gestaltung des äußeren Eindrucks des Roboters sowie seiner Erscheinung als smartes Individuum.

Dieses Kapitel stellt die genannten Komponenten kurz vor, um den Eindruck vom letztendlich realisierten Robotersystem etwas deutlicher zu formen. Dafür werden zuerst die Hardwarekomponenten behandelt und anschließend der Fokus noch einmal auf die Softwarekomponenten in der Control Layer gelegt. Dabei wird ein Verfahren zur Modellierung von Zeitverläufen mittels life-long erweiterbarer Mengen von Samples vorgestellt, welches für die Realisierung des Emotionsmodells und der Vorhersage von Nutzerverfügbarkeiten im Taskscheduler genutzt wurde.

## 8.1 Hardwarekomponenten

### 8.1.1 Streichelfell

Zur Verstärkung der emotionalen Beziehung zwischen Nutzer und Roboter kam im Rahmen der Anforderungsanalyse der Wunsch auf, dass der Roboter auf Berührungen wie Streicheln oder Kitzeln, aber auch auf einen Klaps reagieren sollte. Dazu ist zunächst die Erfassung dieser Berührungen nötig, wozu vom Autor ein spezieller taktiler Sensor entwickelt wurde, welcher in einem Plüschfell auf dem Kopf des Roboters verbaut ist (siehe App. 8.1).

**Abbildung 8.1:** Blauer Streichelfellsensor auf dem Hinterkopf des Serviceroboters (Ansicht von hinten)

Für die Erfassung der Berührungen bietet sich die Messung der entstehenden Druckkräfte an. Alternative Messprinzipien über Körperschall, welcher beim Streichen über die Oberfläche entsteht, oder auch die Nutzung kapazitiver Berührungsmessung wurde am Fachgebiet NI&KR ebenfalls erprobt. In entsprechenden studentischen Arbeiten dazu [Pahl, 2011] wurde kein zufriedenstellendes Ergebnis erreicht.

Die Messung einer Druckverteilung über der Fläche wurde letztendlich durch eine Matrix von Drucksensoren aus einem textilen Material realisiert, welche es ermöglicht, der geschwungenen Kopfform des Roboters zu folgen. In [Müller et al., 2015][1] sind die Details der Lösung beschrieben. An dieser Stelle sollen lediglich das Messprinzip und die resultierenden Daten erläutert werden, bevor die Auswertung der Signale weiter vertieft wird.

[1] Der Autor ist Co-Autor dieses Papers.

Abb. 8.2 zeigt den schichtweisen Aufbau des Sensors. Neben den Trägermaterialien wird die Funktionalität durch die zwei kreuzweise verlaufenden Elektroden aus kupferbeschichtetem Gewebe und einer dazwischen liegenden Schicht aus Velostat$^{TM}$, einem Material mit druckveränderlichen Widerstand, gebildet. Mittels eines ATMEGA16 Mikrocontrollers werden die Matrixzellen zyklisch ausgelesen und liefern somit 72 Widerstandswerte, welche mit 40Hz nach einer Tiefpassfilterung über USB an den PC des Roboters gesandt werden. In Abb. 8.3 kann die Verschaltung nachvollzogen werden. Zur Erhöhung der mit nur 6 ADC-Wandlern auswertbaren Anzahl von Zellen wird mittels der Dioden für jede der beiden Stromrichtungen ein anderer Teil der Spalten angesteuert.

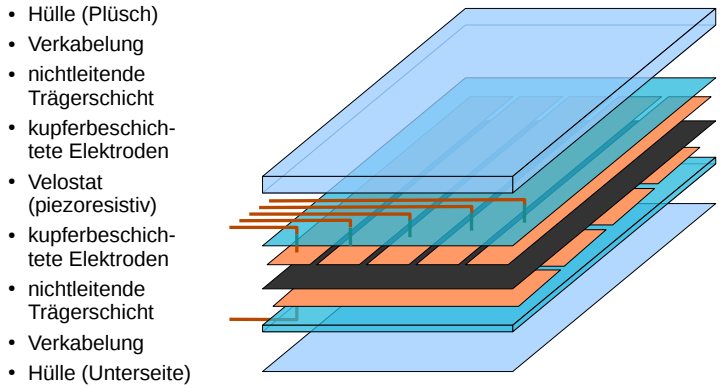



**Abbildung 8.2:** Aufbau des Fellsensors in abnehmbarer Plüschhülle

Der aufmerksame Leser wird sich fragen, wie die unabhängige Messung der Zellen in einer Matrix von Widerständen funktionieren kann. Die Antwort ist, dass die Messungen nicht unabhängig sind. Aufgrund der enormen Unterschiede von unberührt $40k\Omega$ zu $5\Omega$ bei Berührung ist das Übersprechen allerdings relativ gering. Für die Aufgabe, eine Geste zu erkennen, sind die Messwerte ausreichend lokal.

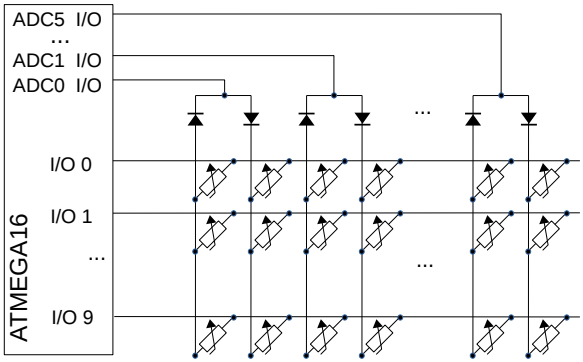



**Abbildung 8.3:** Elektrische Schaltung zum Auslesen der druckabhängigen Widerstände in den Matrixzellen

### Klassifikation der Berührungsmuster

Mit der Anwendung des Streichelfells als Quelle für Feedback und auch, um die emotionale Bindung durch Zoomorphisierung des Systems (tierische Reaktion auf taktile Interaktion nachahmen) zu realisieren, ergaben sich folgende Muster, die unterschieden werden sollten: **Klaps, Tätscheln, Streicheln, Kraulen, Kitzeln** und **Schieben** (statische Druckausübung). Die Klassifikation der Gesten erfolgt über einen einfachen Maximum-Likelihood-Klassifikator, wobei die Modelle der einzelnen Klassen als Gaußverteilung über einem 10-dimensionalen Merkmalsraum aus Beispieldaten gebildet wurden. Zur Extraktion der Merkmale wird der Signalstrom zunächst anhand einer Aktivitätsschwelle in Segmente von maximal 4 Sekunden Länge unterteilt. In diesen ergibt die Bestimmung von Fläche und Summe der Drücke sowie des Druckes des maximal aktivierten Sensors ein Zeitfenster von Rohwerten, über dem die statistischen Parameter Mittelwert und Varianz sowie das Maximum als Features bestimmt werden. Hinzu kommt noch die Varianz des Schwerpunktes der Berührung, um ortsfeste Gesten von Streicheln und Kraulen zu trennen. Eine Backward-Featureselection lieferte die letztendlich verwendete Teilmenge der Features (siehe [Müller et al., 2015]).
Die in Tabelle 8.1 angegebenen erreichten Klassifikationsergebnisse ermöglichen einen sinnvollen Einsatz des Systems auf dem Assistenzroboter. Hierbei

| | Klassifikationsergebnis | | | | | |
|---|---|---|---|---|---|---|
| wahre Klasse | Klaps | Tätscheln | Kraulen | Streicheln | Kitzeln | Schieben |
| Klaps | 55 | 0 | 0 | 3 | 1 | 0 |
| Tätscheln | 1 | 46 | 2 | 1 | 3 | 1 |
| Kraulen | 2 | 7 | 35 | 9 | 10 | 7 |
| Streicheln | 1 | 1 | 6 | 65 | 3 | 5 |
| Kitzeln | 1 | 2 | 2 | 2 | 60 | 1 |
| Schieben | 2 | 0 | 0 | 1 | 0 | 64 |

**Tabelle 8.1:** Klassifikationsergebnisse mit dem Fellsensor, Ergebnis einer Leave-One-Out Cross Validation über einem Datensatz mit 6 Probanden. True Positive Rate 81%

erfolgt für alle Gesten außer Schieben eine direkte akustische Reaktion des Systems durch ein Jaulen, Schnurren oder Lachen. Zeitweise kommt es zu spontanen Fehlerkennungen, wodurch der Roboter unvermittelt eines der Geräusche von sich gibt. Im Langzeitversuch mit Senioren (siehe Abschn. 9.3) wurde das allerdings nicht als negativ bewertet. Im Gegenteil verstärkte dieses Fehlverhalten noch den Eindruck eines gewissen Eigenlebens des Systems.

### 8.1.2 Berührungssensitive Oberfläche

Die Möglichkeit, über einfache Berührungen mit einem Roboter zu kommunizieren, führte zu einer weiteren Sensorik. Mittels kapazitiver Messung, wie sie von Touchscreens bekannt ist, wurde die gesamte Roboteroberfläche berührungssensitiv gestaltet. Dies hatte als geplante Anwendung ebenfalls die Generierung von robotischem Feedback auf Berührungen zum Ziel und sollte für Bewegungsspiele zur physischen Mobilisierung der Nutzer eingesetzt werden. Nachdem die Sensorik installiert war, bot sich eine zweite interessante Anwendungsmöglichkeit an, welche weiter vertieft wurde. Aufgrund der verbauten Getriebemotoren lässt sich der Roboter manuell nur mit viel Kraft und Geduld bewegen. Für eine praktikable Handhabung während der Interaktion muss der Roboter aber teilweise manuell feinpositioniert werden. Hierfür bieten die Berührungssensoren die Möglichkeit, manuelle Schiebebewegungen des Nutzers durch den Roboterantriebs zu unterstützen. Diese Idee wurde mit verschiedenen Entwicklungsstufen der Sensorik umgesetzt und in [Müller et al.,

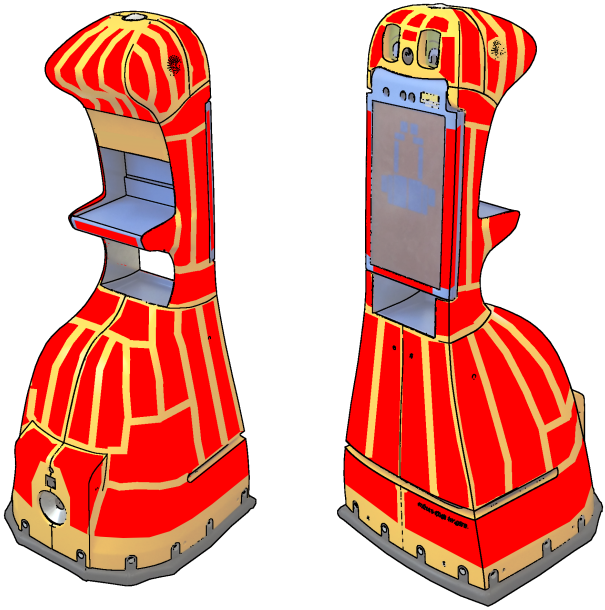


**Abbildung 8.4:** Positionen der kapazitiven Touchelektroden unter der Gehäuseoberfläche des Roboters (rot)

2013] publiziert.

Bevor die Kopplung der Berührungssensorik an die Bewegungssteuerung geschildert wird, soll zunächst die Entwicklung der Sensortechnik vorgestellt werden.

### Kapazitive Sensorhardware

Die Messmethode basiert auf der QTouch$^{TM}$-Technologie, welche einen Mikrocontroller nutzt, um die Kapazität von leitfähigen Elektrodenflächen auszumessen. Indem ein Referenzkondensator jeweils aufgeladen und anschließend auf die Messelektrode umgeladen wird, kann über die Anzahl der Zyklen gezählt werden, wie groß die Kapazität der Elektrode ist. Durch Berührung der Oberfläche ändert sich das Material in der Umgebung und damit die Dielektrizitätszahl, welche die Kapazität bestimmt. Die gemessene Anzahl von nötigen

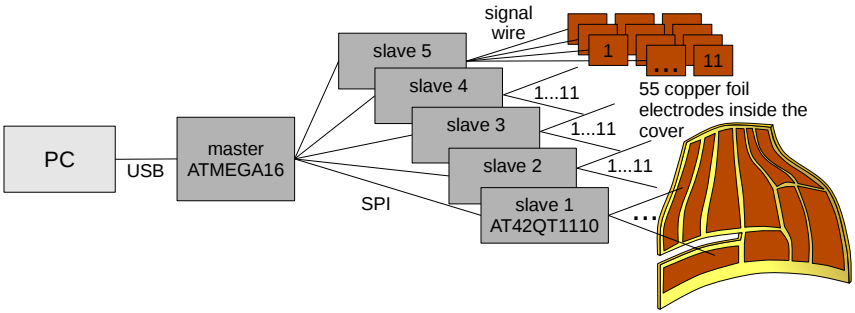



**Abbildung 8.5:** Master-Slave Sensorhardware. An den fünf größten Gehäuseteilen ist jeweils eine Slaveplatine mit 11 Kupferelektroden montiert, welche über SPI (Serial Peripheral Interface) mit der Masterplatine kommunizieren. Der Master sammelt die Daten und ist mittels USB am Roboter-PC angeschlossen.

Ladezyklen entspricht demnach der Stärke der Berührung.

In mehreren Entwicklungsstufen entstand eine Lösung, welche auch Aspekte wie Störungen durch lange Leitungen und Montierbarkeit der Gehäuseteile, Multitouch und große Anzahl unterscheidbarer Flächen berücksichtigt. Als Elektrodenmaterial wurden verschiedene selbstklebende Metallfolien (Aluminium und Kupfer) erprobt, aber auch leitfähiger Lack ist prinzipiell verwendbar. Bei letzterem gestaltet sich die Befestigung der Zuleitungen allerdings schwierig. Aus diesem Grund wird letztendlich eine Lösung basierend auf Kupferfolie präferiert, welche leicht verlötbar ist. Abb. 8.4 zeigt die Anordnung der Elektroden im Inneren der Plastikoberflächen der Gehäuseteile. Die Grenzen orientieren sich an den Orientierungsänderungen der Oberflächen, wodurch bei Druckausübung eine unterschiedliche Bewegungsreaktion des Roboters resultieren würde. Dadurch wird mit gegebener Anzahl von Sensorflächen eine optimale Unterscheidung der Schiebegesten ermöglicht.

Abb. 8.5 verdeutlicht die resultierende Master-Slave-Architektur des Sensorsystems. Jedes abnehmbare Gehäuseteil ist mit einer Slaveplatine versehen, auf welcher ein konventioneller Touch-Controller-Chip AT42QT1110 zum Einsatz kommt. Dieser kann elf unabhängige Kanäle erfassen und kommuniziert über eine SPI-Schnittstelle mit der Masterplatine. In dieser werden die Daten

akkumuliert und tiefpassgefiltert, bevor sie über eine USB-Verbindung zum PC gesandt werden.
Details zum Design der Sensorhardware können in [Lohfeld, 2014][2] nachgelesen werden.

### Unterstützte lokale Navigation

Die Idee der Bewegungssteuerung basiert auf einem einfachen Massemodell, welches durch an der Oberfläche wirkende Kräfte und Drehmomente beschleunigt und durch ein Reibungsmodell gebremst wird. Die resultierende Geschwindigkeit wird dann durch die robotereigenen Motoren umgesetzt. Leider wird von kapazitiven Sensoren keine Kraft erfasst. Die vereinfachende Annahme, dass Kräfte proportional zur gemessenen Berührungsstärke senkrecht zur Oberfläche wirken, ermöglicht jedoch trotzdem eine praktikable Anwendung des Modells. Die Berührungsstärke wird allerdings mehr durch die Auflagefläche der Finger oder Hände bestimmt, und bei natürlichen Greifgesten treten auch Kräfte tangential zur Oberfläche auf, welche bislang unberücksichtigt bleiben. Abb. 8.6 zeigt die modellierten Kräfte und Drehmomente, welche aus jeder einzelnen Sensorfläche resultieren und sich additiv überlagern.

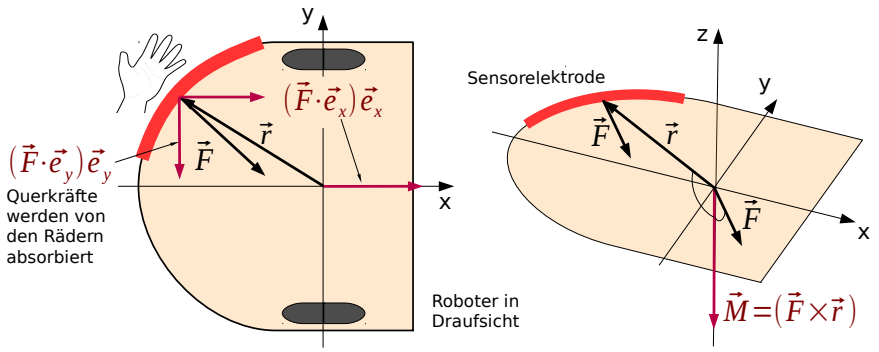



**Abbildung 8.6:** Anwendung der einwirkenden Kräfte auf das Massemodell des Roboters unter Berücksichtigung der beschränkten Freiheitsgrade durch den differentiellen Antrieb.

Die Ansteuerung der Motoren geschieht zwecks Integration einer Kollisions-

[2] Diese Arbeit wurde vom Autor betreut.

vermeidung über das zur Bewegungsteuerung genutzte Navigationssystem [Weinrich, 2015]. Hierbei wird für die berechnete und ähnliche Geschwindigkeiten mittels einer Gaußfunktion eine erhöhte Bewertung generiert. Das auf dem Dynamic Window Approch (DWA) [Fox et al., 1997] basierende Navigationssystem versucht somit, einen Kompromiss für eine Geschwindigkeitskombination zu finden, die auch den anderen Randbedingungen wie beispielsweise Hindernisabstand gerecht wird.
Weiterhin wird die aktuelle Geschwindigkeit $\mathbf{v}_x^0$ und $\mathbf{v}_{rot}^0$ in den Regelkreis integriert und dient jeweils als Grundlage, um mit der resultierenden Kraft $\mathbf{F}_x$ und dem Drehmoment $\mathbf{M}$ gemäß dem zeitlichen Takt $\Delta t$ ein Inkrement $\Delta\mathbf{v}_x$ und $\Delta\mathbf{v}_{rot}$ der Geschwindigkeit zu berechnen.

$$\Delta\mathbf{v}_x = \frac{\Delta t}{m}\left(\mathbf{F}_r(\mathbf{v}_x^0) + \sum_i \mathbf{F}_i \cdot \mathbf{e}_x\right) \tag{8.1}$$

Dabei ist $m$ eine virtuelle Masse des Roboters und $\mathbf{F}_i = b_i\mathbf{n}_i$ die ausgeübte Kraft mit Berührungssensorwert $b_i$ und Oberflächennormale $\mathbf{n}_i$ für Sensorfläche $i$. Der Term $\mathbf{F}_r(\mathbf{v}_x^0)$ steht für eine simulierte Reibungskraft, welche abhängig von der aktuellen Geschwindigkeit der Druckkraft entgegen wirkt und somit eine Bewegung dämpft.

$$\Delta\mathbf{v}_{rot} = \frac{\Delta t}{L}\left(\mathbf{M}_r(\mathbf{v}_{rot}^0) + \sum_i \mathbf{M}_i\right) \tag{8.2}$$

Analog zu $\Delta\mathbf{v}_x$ ergibt ein virtuelles Trägheitsmoment $L$ und ein bremsendes Drehmoment $\mathbf{M}_r(\mathbf{v}_{rot}^0)$ das Inkrement für die Rotationsgeschwindigkeit. Die pro Sensor resultierenden Drehmomente $\mathbf{M}_i$ ergeben sich aus der Position $\mathbf{r}_i$ und Normale $\mathbf{n}_i$ des Sensors $i$ zu $\mathbf{M}_i = \mathbf{r}_i \times b_i\mathbf{n}_i$.
Die Geschwindigkeitsvorgaben für den DWA-Navigator sind damit einfach die Summen $\mathbf{v}_x^0 + \Delta\mathbf{v}_x$ und $\mathbf{v}_{rot}^0 + \Delta\mathbf{v}_{rot}$.
Da aufgrund von Sensorrauschen mit diesem Ansatz auch nach Einführung einer Aktivitätsschwelle eine minimale Bewegung nicht ausgeschlossen ist, wurde die Bewegungssteuerung lediglich auf Kommando aktiviert. Dazu kann einerseits die grafische Oberfläche und andererseits eine längere Berührung an den Schläfen des Roboters genutzt werden.

Das beschriebene System zum unterstützten Schieben des Roboters ist in dieser Form originär und kam während der im folgenden Kapitel geschilderten Nutzertests zum Einsatz. Verständnisprobleme seitens der Senioren und anderer Testnutzer konnten dabei nicht beobachtet werden. Nach wenigen Versuchen waren alle Testpersonen in der Lage, die Berührungen so zu setzen, dass sie den Roboter zielgerichtet bewegen konnten.

## 8.2 Emotionsmodellierung

In diesem Abschnitt wird geschildert, wie der Roboter über den Tag hinweg seine Aktivität moduliert und sich dadurch den Gewohnheiten des Nutzers anpassen soll.

### 8.2.1 Zielstellung einer Emotionsmodellierung für den Roboter

Für den Begriff Emotion existiert keine einheitliche einfache Definition. Im Sinne wie Emotionen hier verstanden werden sollen, bezeichnen sie beim Menschen einen inneren Gefühlszustand, welcher einen wichtigen Teil der zwischenmenschlichen Kommunikation ausmacht. Emotionen lassen einen Kommunikationspartner die Bedeutung von Ereignissen für ein Individuum erkennen und haben mannigfaltige Wirkung auf Wahrnehmung, Verhalten und Expressivität eines Menschen. Sie dienen somit außerdem dem Grounding im zwischenmenschlichen Dialog. Neben Emotionen, welche eher an Ereignisse oder Objekte gebunden und daher zeitlich lokaler wirken, existiert auch der Begriff Stimmung (Mood). Stimmungen bestimmen Wahrnehmung und Verhalten auf zeitlich längeren Skalen und sind oft durch interne körperliche Bedürfnisse bestimmt.
Eine Übertragung dieser Mechanismen auf einen Roboter verfolgt die Absicht, implizit Bedürfnisse des Roboters an seinen Nutzer zu kommunizieren und die soziale Bindung an den Nutzer zu stärken. Durch Simulation von Stimmungen und emotionalen Reaktionen wird das System menschlicher oder tierähnlicher. Außerdem werden die immer wiederkehrenden Verhaltensweisen des Roboters abwechslungsreicher und dadurch evtl. nicht so schnell langweilig. Es ist weiterhin zu beobachten, dass sich Emotionen und Stimmungen

zwischen Menschen übertragen. So soll versucht werden, durch Formung eines Robotercharakters mit einer positiven Stimmung, den Nutzer ebenfalls positiv einzustellen und aus evtl. pessimistischen Phasen zu befreien.
Wie bereits erwähnt, bildet eine Stimmung auch interne Bedürfnisse ab, was die Möglichkeit bietet, den Tagesrhythmus des Roboters an den seines Nutzers anzupassen. Was beim Mensch Müdigkeit und Schlaf sind, findet sich als Ruhephasen auf der Ladestation beim Roboter wieder. Der Roboter soll den Menschen auch nicht ständig bedrängen und sollte auf Ruhephasen des Nutzers eingehen, indem er seine Bereitschafts- oder Beobachtungsposition entsprechend wählt. In aktiven Phasen kann er die Nähe zum Nutzer suchen, indem er im gleichen Raum bleibt (in der praktischen Umsetzung warten auf der Ruheposition im Wohnzimmer), und in Ruhephasen wird die Ladestation aufgesucht (möglichst außerhalb der Sichtweite).

### 8.2.2 Realisierung

Für die Gestaltung virtueller Avatare wurde bereits 2005 das ALMA (A Layered Model of Affect) [Gebhard, 2005] eingeführt. Dieses unterscheidet Emotionen, Stimmungen und Persönlichkeit anhand ihres Einflusses auf Verhalten und Wahrnehmung auf verschiedenen zeitlichen Skalen und stellt den aktuellen Zustand in einem dreidimensionalen Raum dar. Weiterhin werden recht komplizierte Zusammenhänge für die Änderung des Zustands definiert, welche in dieser Komplexität als nicht notwendig erachtet werden und eine Adaption an den Nutzer, wie sie hier gewünscht ist, erschweren.

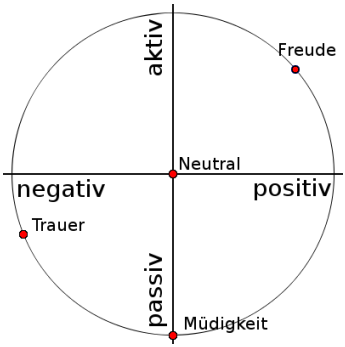



**Abbildung 8.7:** Zweidimensionaler Emotionsraum mit beispielhaften Basisemotionen (rote Punkte)

Für die Realisierung einer Emotions- / Stimmungssimulation wurde daher ein recht einfaches Modell gewählt, welches einen zweidimensionalen Zustandsraum nutzt. (siehe Abb. 8.7) Die Definition einer Positiv-Negativ-Achse (Pleasure) und einer Aktiv-Passiv-Achse (Excitation) entspricht in etwa dem Mo-

dell von Russel [Russell, 1980], in welchem ebenfalls verbal bezeichnete Emotionen in einem zweidimensionalen Raum auf einem Kreis angeordnet werden. Auch für die Roboterplattform SCITOS-G5 (siehe Abb. 1.1) mit ihrem mechanischen Kopf wurde in [Geiger et al., 2013] bereits ein zweidimensionaler Emotionsraum (arousal und valence) für die Beschreibung des Zustands genutzt.

In Emotionsraum, wie er in der vorliegenden Arbeit genutzt wurde, existiert ein Vektor, welcher den aktuellen Zustand $\mathbf{m} = (P, E)$ des Roboters angibt. Die Veränderung dieses Wertes wird durch zwei Einflussgrößen bestimmt. Zum Ersten erfolgt die kurzfristige Veränderung der Emotionen anhand natürlich interpretierbarer Ereignisse, die auch der menschliche Nutzer versteht. Beispielsweise löst Streicheln oder Fahren positive Emotionen aus, und Anstoßen, niedriger Batteriestand oder lange Untätigkeit führen zu negativen, passiven Reaktionen. Der zweite Faktor ist eine Attraktorkurve, welche den Tagesrhythmus vorgibt und durch eine Anpassung über die Nutzungszeit diesen Rhythmus an den des Nutzers anzugleichen vermag.

Konkret existiert eine Kurve für den Excitation-Wert und eine für den Pleasure-Wert. Die Verwendung unterschiedlicher Glättungsparameter macht es möglich, die zeitliche Skala für die Excitation kleiner zu wählen, dadurch kann das Modell sehr schnell den Aktiv-Passive-Phasen im Tagesrhythmus des Nutzers folgen. Die Änderung des Pleasure-Attraktorwertes sollte langsamer erfolgen, sodass nach einer ereignisbedingten Auslenkung der Zustand schnell wieder auf einen relativ stabilen Ruhewert konvergieren kann, welcher letztendlich den Charakter des Roboters bestimmt. In Abb.8.8 sind diese unterschiedlichen Zeitskalen anhand der Attraktorkurven (gestrichelte Linien) zu erkennen. Die grüne Kurve verläuft wesentlich glatter als die Rote.

Abb. 8.8 zeigt einen simulierten Verlauf des Emotionszustands über einen Tag. Die Sprünge im Verlauf entstehen durch beobachtete Ereignisse, welche die Excitation $E$ und den Pleasure-Wert $P$ mittels Gleichung 8.3 zu einem ereignisspezifischen Zielzustand $\mathbf{z}_i$ mit einer spezifischen Stärke $s_i \in [0, 1]$ auslenken.

$$\mathbf{m}^{neu}(t) = \mathbf{m}(t) + s_i(\mathbf{z}_i - \mathbf{m}(t)) \tag{8.3}$$

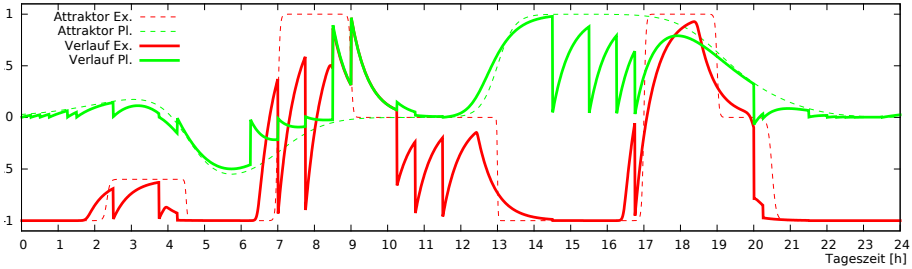



**Abbildung 8.8:** Beispiel für den Verlauf der Emotionsparameter Excitation und Pleasure über einen Tag. Die Werte versuchen, dem Attraktor zu folgen und werden durch konkrete Ereignisse immer wieder ausgelenkt.

Zusätzlich zu den Ereignissen wird periodisch im 5-Minuten-Takt die Annäherung des Zustands an den Attraktor-Wert $\mathbf{a}(t)$ ausgeführt.

$$\mathbf{m}(t + \Delta t) = \mathbf{m}(t) + 2^{-\Delta t/\tau}(\mathbf{a}(t + l) - \mathbf{m}(t)) \tag{8.4}$$

Hierbei ist $\tau$ eine Zeitkonstante, welche definiert, nach wieviel Zeit die Emotionsparameter sich bis auf die Hälfte zum Attraktor angenähert haben sollen. Die Konstante $l$ gibt eine Look-Ahead-Zeit an, welche das Nachlaufen der tatsächlichen Emotionswerte gegen den Attraktor ausgleichen soll. Aufgrund der schrittweisen Annäherung erreicht die Kurve die Zielwerte nur zeitversetzt und abgeschwächt.

### 8.2.3 Sample-basierte Modellierung von Zeitverläufen

Für die Modellierung der Attraktorkurven wird eine online durch Beobachtungen erweiterbare Repräsentation eines Funktionsverlaufs über der Zeit benötigt. Dabei soll nach Möglichkeit für mehrere Beobachtungen zum gleichen Zeitpunkt zwischen diesen interpoliert werden, und die Laufzeit für die Berechnung eines Funktionswertes darf nicht mit der Zeit (Anzahl der Beobachtungen) wachsen. Ziel dieser Anwendung ist Life-Long-Learning und dazu wäre eine Möglichkeit für exponentielles Vergessen lange zurückliegender Beobachtungen wünschenswert, ohne diese alle explizit speichern zu müssen. Der Kurvenverlauf soll außerdem stetig sein.

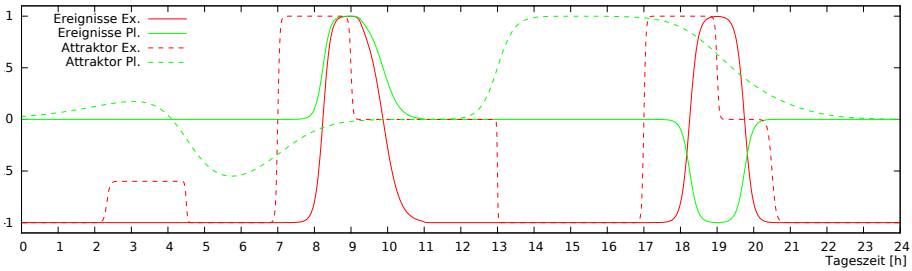



**Abbildung 8.9:** Anfänglicher Attraktor, welcher einen prototypischen Tagesverlauf vorgibt sowie Hüllkurven für simulierte emotionale Ereignisse. Anhand der Hüllkurven wird für die Simulation zufällig ca. ein Ereignis pro Stunde generiert.

Eine Möglichkeit für die Darstellung wäre eine Diskretisierung des Definitionsbereichs und die Interpolation der Funktionswerte zwischen diesen Stützstellen. Die Fragestellung lässt außerdem eine Anwendung von Gaussian Processes und Kernel-Regressionsmethoden zu. Bei diesen wird der Verlauf einer abhängigen Größe über Samples repräsentiert. Leider wächst das Sampleset unbeschränkt und die Berechnung von Funktionswerten kann dadurch sehr aufwändig werden. Für die praktische Umsetzung wurde trotzdem eine sample-basierte Darstellung und Regression mittels eines Nadaraya-Watson-Schätzers gewählt. Ergänzt wurde diese Methode um eine Gewichtung der Samples und einen Vereinfachungsalgorithmus der Samplemenge bei Überschreiten einer Maximalanzahl.

Die Funktion $y(x)$ wird demnach über eine gewichtete Menge an Samples $\{(w_i, x_i, y_i)|i = 1, \ldots, N\}$ dargestellt. Die Interpolation geschieht durch eine gewichtete Mittlung aller Samples, wobei der Einfluss eines Samples von seinem Abstand im Definitionsbereich und der Kernelfunktion $\Psi(x)$ abhängt. Über einen Bandbreitenparameter $\sigma$ kann der Einflussbereich eines Samples und damit die Glattheit der resultierenden Funktion bestimmt werden.

Der Funktionswert an Stelle $x_0$ lässt sich damit wie folgt berechnen:

$$y(x_0) = \frac{\sum_{i=1}^{N} w_i y_i K(\frac{x_i - x_0}{\sigma})}{\sum_{j=1}^{N} w_j K(\frac{x_j - x_0}{\sigma})} \tag{8.5}$$

Als Kernelfunktion kommt dabei ein Gausskernel $\Psi(x) = e^{-x^2}$ zum Einsatz.

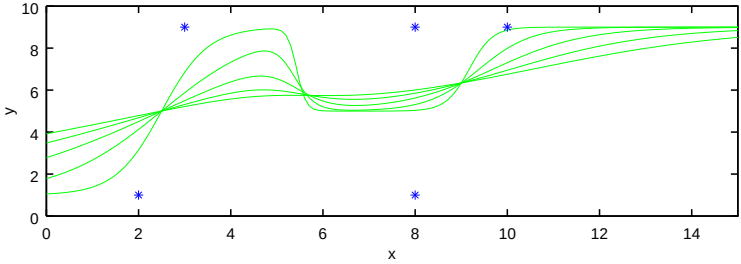



**Abbildung 8.10:** Beispiel für die Interpolation zu den als Sternchen gegebenen Stützstellen für verschiedene $\sigma$ Werte im Bereich von 1 bis 3.

Das Vergessen lange zurückliegender Beobachtungen lässt sich leicht realisieren, indem vor dem Einfügen neuer Samples die Gewichte aller existierenden mit einem Faktor $f = 2^{-\frac{1}{\tau}}$ multipliziert werden, wobei $\tau$ die Anzahl von Beobachtungsschritten angibt, bis zu welchem sich der Einfluss einer zurückliegenden Beobachtung halbiert haben soll.

Um ein unbegrenztes Ansteigen der Sampleanzahl zu unterbinden, wurde ein Vereinfachungsschritt definiert, der die Sampleanzahl reduziert. Dabei wird das Samplepaar $(k, l)$ bestimmt, welches im Definitionsbereich den geringsten Abstand besitzt. Diese werden nach Gleichung 8.6 durch ein neues Sample $m$ ersetzt, wobei sich das Gewicht addiert und die Position und Funktionswerte gemäß der Gewichte interpoliert werden. Falls die Samples auf der gleichen Stelle im Definitionsbereich der Funktion liegen, wird dabei kein Fehler begangen. Für eng nebeneinander liegende Samples ändert der Vereinfachungsschritt den Kurvenverlauf leicht, was aber wenig ins Gewicht fällt sofern der Bandbreitenparameter $\sigma$ größer ist als der Abstand zum nächsten Sample.

$$\begin{aligned} w_m &= w_l + w_k \\ x_m &= \tfrac{w_l}{w_m} x_l + \tfrac{w_k}{w_m} x_k \\ y_m &= \tfrac{w_l}{w_m} y_l + \tfrac{w_k}{w_m} y_k \end{aligned} \tag{8.6}$$

Abb. 8.10 zeigt ein Beispiel für die Interpolation der als Sternchen gegebenen Samples mit verschiedenen Bandbreitenparametern.

### 8.2.4 Adaption der Attraktorkurven

Die Attraktoren und somit auch der spontane Verlauf von Excitation und Pleasure nehmen die ereignisbedingten Werte vorweg und können somit dafür sorgen, dass der Roboter von sich aus rechtzeitig aktiv wird und auch die Ruhephasen korrekt voraussieht.

Nachdem der Verlauf des Emotionszustands über einen Tag bestimmt wurde, kann dieser genutzt werden, um den Attraktor an den tatsächlichen Verlauf, welcher auch das Nutzungsprofil enthält, anzupassen. Dadurch kann, unter Annahme von Periodizität, am nächsten Tag der Verlauf der Aktivität und der Grundstimmung vorweggenommen werden, ohne dass emotionale Ereignisse nötig sind. Die Anpassung erfolgt, indem zunächst die Gewichte der Samples im alten Attraktor mit einem Faktor $f < 1$ multipliziert werden, damit der Einfluss lange zurückliegender Profile abnimmt. Danach wird der tatsächliche Verlauf des erlebten Tages im 5-Minuten-Raster abgetastet und als neue Beobachtungen mit Gewicht 1 dem Attraktor hinzugefügt. Die Maximalanzahl der Samples ist auf 50 begrenzt (eines pro halbe Stunde), wodurch der Vereinfachungsschritt aus Abschn. 8.2.3 immer wieder angewendet und dadurch die Kurve geglättet wird.

Die bestimmungsgemäße Funktion der Modulation der Emotionsparameter sowie der Langzeitschätzung der Verläufe wurde in einer Simulation überprüft. Dabei wurde ein hypothetisches Nutzungsverhalten mit positiven und negativen Ereignissen für Excitation und Pleasure mittels einer Hüllkurve vorgegeben. Anschließend wurden der Tagesverlauf simuliert und zufällige Ereignisse (eines pro Stunde) generiert, wobei deren Zielwerte $\mathbf{z}_i$ der Hüllkurve entnommen wurden. Abb. 8.11 zeigt, wie sich der Attraktor ausgehend von einem hypothetischen Anfangsverlauf nach und nach dem tatsächlichen Nutzungsverhalten (simulierte Ereignis-Hüllkurve) anpasst. Erwartungsgemäß verschieben sich die Extrema an die Positionen, welche durch die Hüllkurven vorgegeben sind. Die gegenüber der Hüllkurve größere Glattheit der Attraktorkurven ist bedingt durch die abweichenden Bandbreitenparameter. Die niedrigeren Maximalamplituden lassen sich damit erklären, dass Ereignisse nur spärlich auftreten und die Werteverläufe dadurch niemals an die Hüllkurven heranreichen. Für eine erkennbare Variation des Systemverhaltens sind die resultierenden

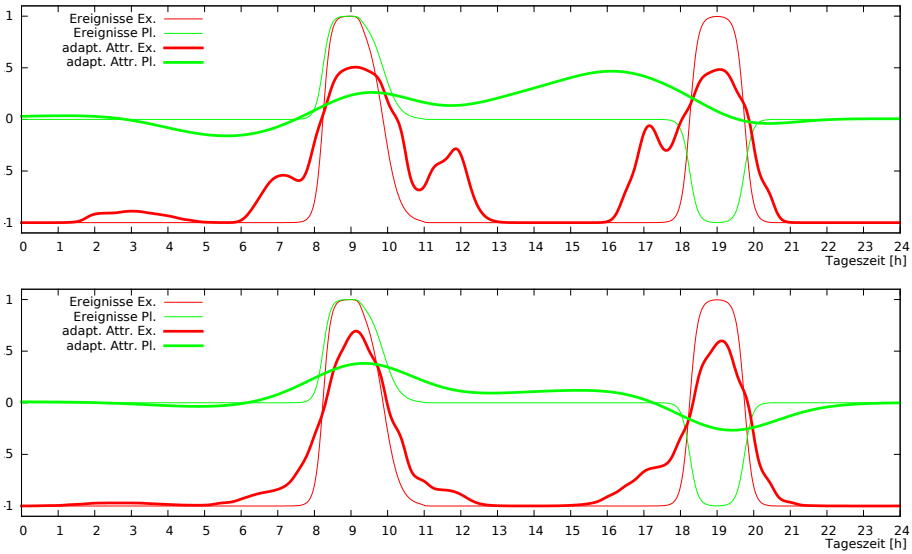



**Abbildung 8.11:** Adaptierter Attraktor nach einem Tag (oben) und nach sieben Tagen (unten) mit simulierten Ereignissen. Das Ergebnis nach einem Tag entspricht dem Fusionsergebnis der beiden Kurven (Attraktor + tatsächlicher Verlauf) aus Abb. 8.8

Auslenkungen allerdings ausreichend.

### 8.2.5 Emotionale Ereignisse

Wie bereits erwähnt müssen verschiedene Ereignisse definiert werden, welche eine emotionale Reaktion des Roboters auslösen und eine Kopplung zum Interaktionsverhalten des Nutzers ermöglichen.
In der konkreten Umsetzung des SERROGA Roboters wurden folgende Ereignisse vorgesehen:

- **Streicheln, Tätscheln, Kraulen, Kitzeln**: Mittels des Fellsensors (Abschnitt 8.1.1) werden positive Berührungsmuster mit einem positiven und aktiven Emotionsvektor verknüpft ($\mathbf{z} = (1, 1)$,$s = 0.5$). Dieser Zielzustand entspricht "Freude".
- **Klaps**: Ebenso ist ein negatives Berührungsmuster mit negativen Emo-

tionen verknüpft ($\mathbf{z} = (-1, -0.3)$,$s = 0.5$). Dies entspricht "Trauer". Es wurde absichtlich keine positive Excitation gewählt, da dies "Wut" oder "Ärger" ausdrücken würde, was vermieden werden soll, um einen positiven Grundcharakter des Systems zu gewährleisten.

- **Fahrbewegung**: Der Roboter soll "Freude" erfahren, wenn er sich bewegen darf. Daher wird das manuelle Starten eines Navigationsbehaviors (fahre zu, Nutzersuche, Folgen) mit $\mathbf{z} = (1, 1)$,$s = 0.5$ verknüpft.
- **Kollision**; Falls der Roboter an einem Hindernis anstößt, wird dies negative Emotionen auslösen ($\mathbf{z} = (-1, -0.3)$,$s = 0.75$).
- **Nutzerkontakt**: Der Roboter soll seine Aktivitätskurve an die des Menschen anpassen. Dazu werden Nutzerdetektionen im näheren Umkreis des Roboters mit Aktivität verknüpft $\mathbf{z} = (0, 1)$,$s = 0.2$. Die Prüfung dieser Bedingung wird dabei periodisch (5-Minuten-Takt) vollzogen, damit eine ständig präsente Person nicht zu starken Einfluss bekommt.
- **niedriger Batteriestand**: Um auszudrücken, dass der Roboter eine Aufladung der Akkus benötigt, wurde eine Basisemotion "Müdigkeit" eingeführt und im 5-Minuten-Takt bei Unterschreitung eines Ladezustands von 30% ein Ereignis $\mathbf{z} = (0, -1)$,$s = 0.2$ ausgelöst.
- **Untätigkeit**: Um Ruhezeiten zu erfassen, wurde viertelstündlich geprüft, ob innerhalb des letzten Intervalls eine Interaktion mit dem Nutzer stattgefunden hat. Falls dies nicht der Fall war, so wird der Roboter "müde" ($\mathbf{z} = (0, -1)$,$s = 0.5$).

Denkbare Erweiterungen für emotionale Ereignisse wären einerseits das erfolgreiche Abschließen eines Dialogs mit dem Nutzer, welches positiv bewertet werden könnte, andererseits die Spiegelung der Emotionen des Nutzers, falls diese durch den Roboter z.B. über eine Mimikanalyse hinreichend beobachtet werden könnten.

### 8.2.6 Ausdrucksmöglichkeiten des Emotionszustands

Das Emotionsmodell hat wenig Sinn ohne entsprechende Wirkung auf das Roboterverhalten oder sein Erscheinungsbild. Im Rahmen des SERROGA Demonstrators wurden drei systemweite Größen variiert (Bildschirmfarbe,

Sprachmelodie und Wortlaut sowie die Animation der Augen). Außerdem wurde eine spezielle Service-App entwickelt, welche, getriggert durch bestimmte Emotionszustände, explizit Aktivitäten zum Ausdruck der Emotionen ausführt, wenn der Roboter nichts zu tun hat. Bei Langeweile pfeift der Roboter vor sich hin, bei Freude sucht er den Nutzer und schnurrt, bei Müdigkeit geht der Roboter zu seiner Ladestation (auch wenn noch genügend Akkuladung vorhanden ist). Dadurch kann der Roboter den Tagesrhythmus des Nutzers nachahmen, ohne dass dieser sich immer an- und abmelden muss.

### Bildschirmfarbe

Durch die Verwendung des GUI-Managers und einer zentralen Definition der Gestaltungselemente für die Bedienoberflächen der Apps war es möglich, für alle Fenster auf dem Bildschirm eine zentral bestimmte Hintergrundfarbe zu realisieren.


**Abbildung 8.12:** Anpassung der Bildschirmhintergrundfarbe an die simulierte Emotion und die vorliegende Umgebungshelligkeit

Diese Farbe berücksichtigt zum einen die aktuelle Emotion im Roboter, zum anderen die vorliegende Beleuchtungssituation. Über die Weitwinkelkamera in der Nase des Roboters (siehe Abb. 1.3) ist es möglich, dass der Roboter Teile

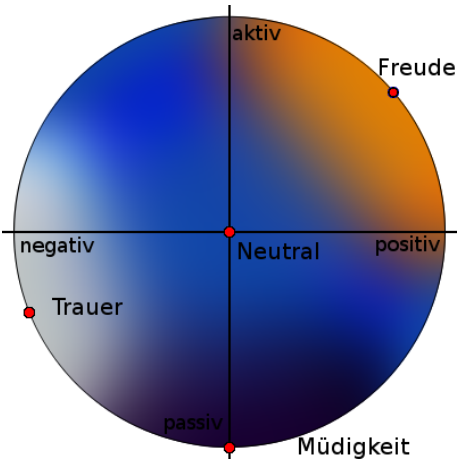



**Abbildung 8.13:** Positionen der Basisemotionen im Emotionsraum. Interpolierte Farbe für den Displayhintergrund in Abhängigkeit vom aktuellen Emotionszustand.

von sich selbst sieht. Darüber kann die Beleuchtungsstärke des Displayrahmens und somit die des Displays berechnet werden. Mit Hilfe dieser Daten erfolgt die Anpassung der Hintergrundhelligkeit des Displays, sodass es im Dunkeln nicht zu hell strahlt und im Hellen dennoch gut ausgeleuchtet ist (siehe Abb. 8.12). Der Farbton des Hintergrundes ergibt sich durch die Interpolation der vier Basisfarben, welche den vier Basisemotionen Neutral (0,0), Freude (0.7,0.7), Trauer (-0.8,-0.3) und Müdigkeit (0,-1) zugeordnet sind. Abb. 8.13 zeigt in etwa, wie sich die Farben aus dem Emotionsvektor ergeben. Die Auswahl der Farben orientierte sich dabei am äußeren Design des Roboters.

### Sprachausgaben

Die Modulation der Sprachausgaben gestaltet sich etwas aufwändiger, da sie bei der Definition jeder einzelnen Service-App berücksichtigt werden muss. Mit Hilfe des Outputselectors (siehe Abschn. 6.10) ist es möglich, in Abhängigkeit von Zustandsvariablen verschiedene Mengen von Phrasen für eine bestimmte Bedeutung zu definieren, aus denen bei Bedarf per Zufall ausgewählt

wird.
Zur Berücksichtigung des aktuellen Emotionsvektors muss dieser diskretisiert werden. Dafür finden die bereits für die Definition der Farbe festgelegten Basisemotionen Verwendung. Eine einfache Nearest-Neighbour-Suche zum aktuellen Emotionsparameter führt zur genutzten Klasse.

#### Augenanimationen

Der SCITOS-G3 Roboter, welcher auch für die Realisierung des SERROGA Demonstrators zum Einsatz kam, besitzt zwei LC-Displays als Augen. Auf diesen konnten ursprünglich lediglich zwei Bilder (offene und geschlossene Augen) angezeigt werden. Im Rahmen einer studentischen Arbeit [Funk, 2015][3] wurde eine Hardware entwickelt, mit der die beiden Displays beliebige Grafikausgaben eines Mini-PC wiedergeben können. Darauf aufbauend ist es nun möglich, eine für die verschiedenen Emotionszustände angepasste Augenmimik des Roboters zu realisieren. Eine weitere Verbesserung ergibt sich durch die Nutzung der variablen Augenbilder, um die Blickrichtung des Roboters zum Nutzer auszurichten. Anhand des Personentrackers (Skill) ist die relative 3D-Position des Kopfes des Interaktionspartners bekannt. Dies ermöglicht es, die Pupillen im Auge entsprechend zu positionieren.
Die Augenbilder werden, wie auch die Sprachausgaben, anhand der nächstgelegenen Basisemotion ausgewählt. Animierte Übergänge zwischen den Ausdrücken und geschlossene Augen zum Blinzeln lockern die Mimik zusätzlich auf.
Um weitere Absichten oder Zustände zu visualisieren, kann zusätzlich aus den Service-Apps manuell eine Vielzahl weiterer Mimiken abgerufen werden. Beispiele hierfür sind ein Zwinkern mit einem Auge oder Herzchen in den Augen. Abb. 8.14 gibt ein Beispiel für verschiedene Augendarstellungen.

### 8.2.7 Diskussion

Nachdem eine einfache Form einer Emotionalisierung des Roboters implementiert wurde, bleibt die Frage offen, ob diese vom Nutzer korrekt wahrgenom-

[3] Diese Arbeit wurde vom Autor betreut.

**Abbildung 8.14:** Beispielhafte Augenbilder: links - neutrale Mimik, Mitte - verfolgen eines Nutzers mit den Augen, rechts - trauriger Ausdruck.

men wird und ob dadurch in irgendeiner Form ein Vorteil für die Interaktion entsteht.
Leider lassen sich diese Fragen nicht ohne aufwändige Nutzertests klären. Es müssten vergleichende Studien mit und ohne Emotionsmodell ausgeführt werden, um die Nutzermeinung bzgl. Akzeptanzsteigerung, Wahrnehmung des Roboters als Individuum und nach langer Adaptionszeit auch bzgl. der Anpassungsfähigkeit zu erheben. Da dies im Rahmen der bearbeiteten Projekte nicht möglich war, wurde zur Überprüfung der Anpassungsfähigkeit die Simulation gewählt, mit der objektiv nachgewiesen werden konnte, dass sich die Verläufe der Emotionsparameter an die von außen eingebrachten Ereignisse anpassen.
Im Rahmen der Evaluation des SERROGA Demonstrators mit Senioren (siehe Abschn. 9.3) konnten Teile des Systems auch mit realen Nutzern erprobt werden. Allerdings war die Interaktionsdauer von maximal drei Tagen nicht ausreichend für eine Bewertung der Adaption, und eine Vergleichsgruppe ohne Emotionsmodell existierte bisher noch nicht. Bei den Nutzertests waren die Darstellung über Bildschirmfarbe und Sprachmodulation aktiv. Die zusätzlichen Ausdrucksmöglichkeiten durch die spezielle Service-App und die Augenanimationen konnten zum Zeitpunkt der Nutzertests noch nicht eingesetzt werden. Dennoch wurden die Nutzer in der Nachbesprechung befragt, ob sie eine Veränderung der Stimme während des Tests wahrgenommen haben, was allerdings von allen verneint wurde.
Dies zeigt auch die Schwierigkeit einer Evaluation. Emotionen und deren Darstellung sollen vom Kommunikationspartner (Nutzer) nur unbewusst wahrgenommen werden und können eigentlich nur über ihre Wirkung (Verbesserung der Akzeptanz, geringere Eintönigkeit des Roboterverhaltens) nachgewiesen werden. Dennoch wird in vielen Studien eine explizite Befragung

zur Wahrnehmung von dargestellten Emotionen vorgenommen. Dabei werden Ausdrucksformen für ein bestimmtes Set von Basisemotionen implementiert und anschließend eine Testgruppe gebeten, die vorgeführten Ausdrücke diesen Basisklassen zuzuordnen. Dieses Vorgehen wird vom Autor als kritisch erachtet, obwohl die Frage nach der Empfindung einer Ausdrucksform beim Nutzer durchaus legitim ist.
Eine bessere Vorgehensweise, welche auch am Fachgebiet NI&KR umgesetzt werden wird, ist eine Platzierung der Ausdrucksformen im Emotionsraum anhand der direkten Einschätzung durch Nutzer. Dadurch wird die Diskretisierung in verbal benannte und von jedem anders verstandene Emotionsklassen vermieden und es entsteht ein System, welches bestmöglich der Wahrnehmung durch die Nutzer entspricht. In einer Versuchsreihe werden dazu verschiedene (zufällige) Parameter für die Ausdrucksformen generiert und den Testpersonen das resultierende Verhalten präsentiert. Diese beschreiben die von ihnen wahrgenommene Emotion, indem sie im Emotionsraum (2D-Diagramm ähnlich Abb. 8.7) ein Kreuz setzen. Durch die Beschriftung mit weiteren Basisemotionen sollte auch Laien ein leichtes Verständnis der Darstellung ermöglicht werden. Im Anschluss kann mit diesen Daten eine ideale Abbildung des 2D-Emotionsraumes auf die Parameter für die Ausdrucksformen generiert werden. Bei hinreichend vielen Testbeobachtern mitteln sich dabei individuell abweichende Einschätzungen einzelner Personen heraus und auch gelegentlich auftretende Missverständnisse der Methodik fallen nicht zu sehr ins Gewicht. Als weitere wesentliche Ausdrucksmöglichkeit wird dabei zusätzlich zu den bereits beschriebenen die Bewegung des Roboters untersucht. Einerseits kann das Fahrverhalten bei zielgerichteten Bewegungen moduliert werden (Geschwindigkeit und Glattheit der Trajektorie) und andererseits können spezielle Bewegungsmuster definiert werden, die ausgeführt werden wenn keine direkte Aufgabe vorliegt.

## 8.3 Taskscheduler

Eine letzte nenneswerte Komponente in der Control Layer eines Serviceroboters ist ein Taskscheduler. Dieser ergänzt das bislang rein reaktive bzw. nutzer- oder zeitgesteuerte Verhalten des Roboters durch eine situationsab-

hängige Optimierung von Ausführungszeitpunkten verschiedener Interaktionstasks (siehe Abb. 8.15). Beispielsweise würde die bislang rein zeitgesteuerte Auslieferung von Erinnerungen die Anwesenheit des Nutzers und seine aktuelle Tätigkeit berücksichtigen und den Erinnerungszeitpunkt entweder vorverlegen, wenn zu erwarten ist, dass der Nutzer zum geplanten Zeitpunkt nicht verfügbar ist, oder die Erinnerung verzögern, bis eine wichtigere Aktivität des Nutzers beendet wurde. Weiterhin können bei parallel geplanten Tasks auch Prioritäten berücksichtigt und somit Konflikte sauber aufgelöst werden. Ohne einen Taskscheduler werden beim realisierten SERROGA System die Interaktionstasks allein durch den Stack im Dialogmanager organisiert, wonach der Task zuerst bearbeitet wird, welcher zuletzt aktiviert wurde. Eine Berücksichtigung von Prioritäten findet dabei nicht statt.

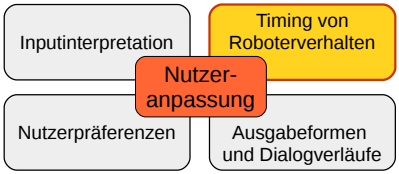



**Abbildung 8.15:** Behandlung des Aspekts des nutzerangepassten Timings von Interaktionsverhalten.

Die Überlegungen bezüglich eines Taskschedulers zur Optimierung von Ausführungszeitpunkten von Interaktionstasks wurden am Fachgebiet NI&KR bereits 2008 [Lapp, 2008][4] im Rahmen des HOROS (HOme RObot System) Projekts begonnen. Dabei Stand die Optimierung von Erinnerungszeitpunkten in einem Office-Szenario (als Ersatz für eine häusliche Testumgebung) anhand eines Vorhersagemodells der Interaktionsbereitschaft des Nutzers im Mittelpunkt. Die nötigen Wahrnehmungsleistungen des Roboters wurden dabei lediglich simuliert.

Eine Weiterentwicklung des Ansatzes fand im Rahmen des CompanionAble Projekts statt, wobei das benötigte Situationsverständnis durch eine Erkennung der Nutzeraktivitäten sowie durch Zusatzinformationen aus der intelligenten Wohnumgebung (der zweite Schwerpunkt im CompanionAble Projekt) realisiert wurde. Außerdem wurde in diesem Zug eine Priorisierung der unterschiedlichen Tasks eingeführt.

Beide Projekte brachten zwar Prototypen hervor, aber in der Praxis stellten diese sich für den Einsatz im angestrebten Szenario als nicht hinreichend

[4] Diese Arbeit wurde vom Autor betreut.

robust heraus. Beide Konzepte, sowohl die Modellierung der erwarteten Interaktionsbereitschaft, als auch die kontinuierliche unaufdringliche Erfassung der Nutzeraktivitäten, sind noch nicht ausgereift. Daher werden an dieser Stelle lediglich die Grundidee geschildert und die in den Experimenten festgestellten Kritikpunkte diskutiert.

Neben der beobachtungsgetriebenen Optimierung der Ausführungszeitpunkte verschiedener Tagesaufgaben könnte ein zentraler Taskscheduler auch die Dauer der Tasks und evtl. die Wegkosten zwischen den Tasks berücksichtigen, wie es im EU Projekt STRANDS realisiert wurde [Mudrova und Hawes, 2015]. Hierbei werden die Tasks durch einen globalen Optimierungsprozess über den Tag verteilt. Für das vorliegende Anwendungsszenario wird solch eine globale Planung nicht als notwendig erachtet, da die Dauer der Interaktionen im Vergleich zu ihrer Häufigkeit und der zur Verfügung stehenden Zeit vergleichsweise unbedeutend ist. Die Blockaden, die laufende Tasks für andere bedeuten, fallen für das gegebene Einsatzszenario als Gesundheitsassistent demnach nicht ins Gewicht, sodass jeder Task individuell gescheduled werden kann, ohne Abhängigkeiten zu anderen Tasks berücksichtigen zu müssen.

Abb. 8.16 zeigt das Zusammenspiel des Taskschedulers mit den anderen Komponenten der Systemarchitektur. Die einzelnen Service-Apps der App-Layer können beim Taskscheduler Interaktionstasks registrieren, wobei sie eine zeitliche Kurve definieren, welche die Notwendigkeit der Taskausführung zum jeweiligen Zeitpunkt beschreibt. Außerdem sollte den Tasks eine Priorität zugeordnet werden, damit diese bei der Entscheidung, wann der Roboter den Nutzer bei einer evtl. Tätigkeit unterbrechen soll, berücksichtigt werden kann. Durch das dynamische Hinzufügen und Löschen von Tasks können auch kausale Abhängigkeiten zwischen Tasks und Abhängigkeiten von beobachtbaren Ereignissen[5] in den Service-Apps realisiert werden. Der Taskscheduler entscheidet im weiteren Verlauf anhand eines Prognosemodells der Anwesenheit des Nutzers und seiner Interaktionsbereitschaft, wann ein Task gestartet werden kann und meldet dies zur Service-App oder dem Dialogmanager zurück. Dabei kann es auch passieren, dass der Zeitraum einer sinnvollen Ausfüh-

[5] Beispielsweise könnte eine Erinnerung an eine Medikamenteneinnahme eingefügt werden, sobald das Mittagessen beobachtet wurde. Falls die Einnahme durch den Nutzer spontan erfolgt und dies beobachtet werden kann, so kann der Eintrag auch wieder aus dem Plan entfernt werden.

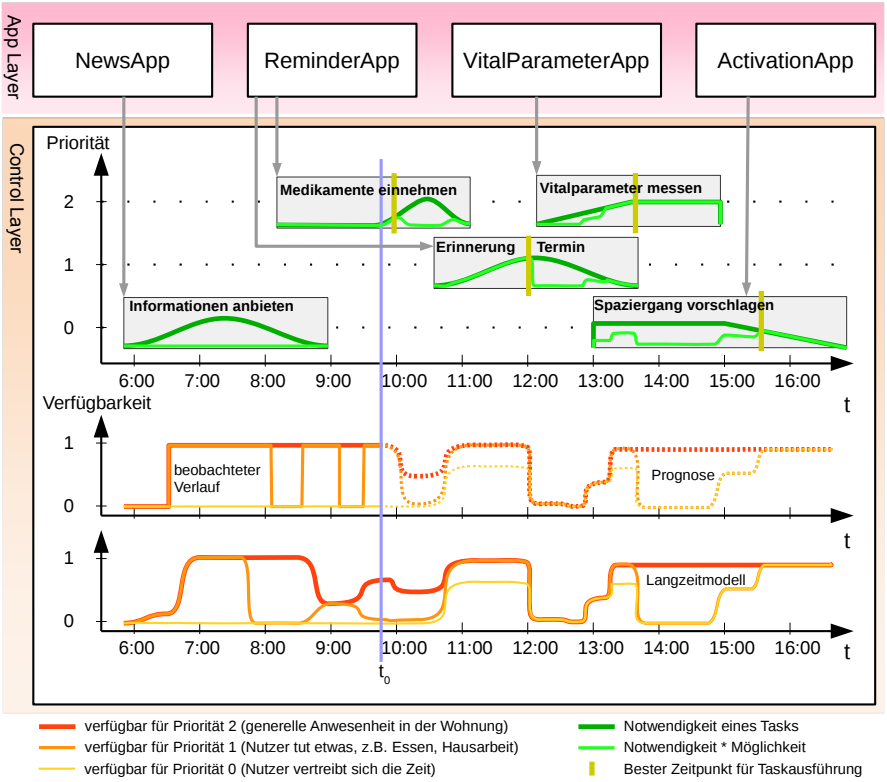



**Abbildung 8.16:** Übersicht zur Funktion des Taskschedulers: Service-Apps können zeitgesteuert oder eventgetriggert (z.B. nach dem Frühstück) Interaktionstasks in den Tagesplan eintragen. Die Kombination der Notwendigkeitskurve und der Möglichkeit für eine Interaktion mit entsprechender Priorität ergeben die hellgrünen Kurven, deren Maxima die idealen Ausführungszeitpunkte ergeben.

rung verstreicht, ohne dass sich eine Interaktiongelegenheit ergibt. In diesem Fall wird ebenfalls die Service-App informiert und kann entsprechend darauf reagieren.

Der Taskscheduler trifft seine Entscheidungen aufgrund einer Schätzung der Anwesenheit des Nutzers sowie einer Beobachtung dessen aktueller Aktivität.

Damit ergibt sich der Grad an Dringlichkeit von Interaktionsaufgaben, die den Nutzer bei seiner Aktivität unterbrechen dürfen ohne ihn zu frustrieren. Ein wesentlicher Bestandteil bei dieser Abschätzung ist eine Prognose des Verlaufs der Anwesenheit und Beschäftigung, um nicht nur rückwirkend zu wissen, dass eine Ausführung sinnvoll gewesen wäre.

Als den besten Zeitpunkt für eine Ausführung wird postuliert, dass an diesem Punkt das Produkt $U(t) = N(t)M(t)$ aus Notwendigkeit $N(t)$ und Möglichkeit zur Ausführung (Nutzerverfügbarkeit) $M(t)$ einen Maximalwert erreicht (siehe hellgrüne Kurvenverläufe in Abb. 8.16).

$$t_{opt} = \underset{t}{\operatorname{argmax}}\, N(t)M(t) \tag{8.7}$$

Dieser Zeitpunkt kann anfänglich anhand des Prognosemodells der Möglichkeit für die benötigte Prioritätsklasse bestimmt werden, muss aber während des Betriebs immer wieder an die konkret beobachtete Anwesenheit und Tätigkeit des Nutzers angepasst werden. Für die tatsächliche Entscheidung zur Ausführung können weiterhin auch Schwellwerte $U_{min}$ und $U_{max}$ für eine Mindestmöglichkeit bzw. Notwendigkeit, ab der man nicht mehr warten will, definiert werden.

In den Versuchen am Fachgebiet NI&KR wurde für die Prognose der Anwesenheit des Nutzers die unter Abschn. 8.2.3 eingeführte sample-basierte Modellierung von Zeitverläufen genutzt, wobei eine tägliche Periodizität der Anwesenheit angenommen wurde. Leider zeigte sich bei den Experimenten, dass sich dadurch lediglich die groben Eckdaten (Arbeitsbeginn, Mittagspause, Feierabend) erfassen lassen. Kleinere Unterbrechungen sind zu variabel, als dass solch ein einfaches Modell sie vorhersagen könnte. Für eine reale Anwendung sollte daher die Modellierung der Prognose weiter verbessert werden. Zunächst sollte dazu nicht nur ein prototypischer Verlauf modelliert, sondern eine Vielzahl realer Tagesverläufe gespeichert werden. Dadurch kann bei der Prognose auch der bereits beobachtete Kontext berücksichtigt werden, um die Prognose spezifischer zu gestalten. Außerdem sollten aus einer vorgehaltenen Datenbasis auch die Varianzen der Verläufe bestimmbar sein, die bei der Entscheidungsfindung ebenfalls zu berücksichtigen wären. Bei unsicher vorherzusehenden Verläufen der Interaktionsmöglichkeit wäre demnach die

Ausführung zu einem früheren Zeitpunkt vorzuziehen.

### 8.3.1 Aktivitätserkennung

Eine unaufdringliche und damit freudvolle Interaktion mit dem Nutzer, welche zur Erhöhung der Akzeptanz eines Roboters wichtig ist, sollte die aktuellen Aktivitäten des Nutzers berücksichtigen. Tasks niederer Priorität dürfen den Menschen nicht unterbrechen, wenn dessen aktuelle Tätigkeit höhere Priorität hat.

Die Wahrnehmung einer Nutzertätigkeit vom Standpunkt des Roboters aus ist ein sehr komplexes eigenes Themengebiet. In der Literatur werden dafür hauptsächlich visuelle oder auf 3D-Datenverarbeitung basierende Methoden vorgeschlagen, welche eine ständige Sichtbarkeit des Nutzers im Sensornetzwerk voraussetzen [Hu et al., 2004].

Um die daraus resultierenden Probleme zu umgehen, wurde am Fachgebiet NI&KR ein alternativer Ansatz entwickelt, welcher hauptsächlich auf der Position und Bewegungstrajektorie des Nutzers in der Einsatzumgebung basiert. Durch Labeling von Objekten und Räumen in der 2D-Umgebungskarte kann zusammen mit einer einfachen visuellen Bewegungsanalyse (Nutzer bewegt sich oder nicht) und Zusatzinformationen zum Zustand der intelligenten Wohnumgebung (Fernseher an/aus, Licht an/aus, ...) mittels einer manuell definierten Fuzzy-Regelbasis auf eine grobe Aktivitätsklasse geschlossen werden. [Tannert, 2014][6]

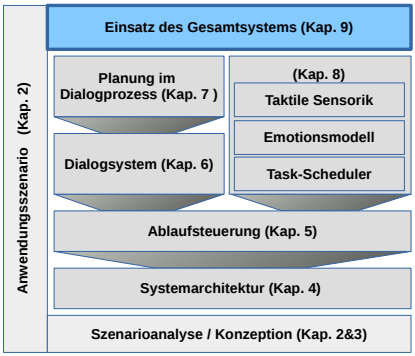
Die Beobachtung des Nutzers erfordert auch ein entsprechendes Navigationsbehavior, welches versucht, den Nutzer im Blick zu halten, ihm aber nicht aufdringlich zu nahe kommt oder im Weg herum steht. Bislang wurde dazu lediglich eine Ruheposition in der Nähe der hauptsächlichen Aufenthaltsorte des Nutzers definiert, an der der Roboter bei Nichtnutzung wartet und beobachtet. Da dieses Verhalten nicht ausreicht, um den Nutzer kontinuierlich zu beobachten, wird am Fachgebiet NI&KR gegenwärtig an einem aktiven Beobachtungsverhalten gearbeitet.

[6] Diese Arbeit wurde vom Autor betreut.

## 8.4 Fazit

Dieses Kapitel konnte zeigen, wie durch Hardwareerweiterungen neue Wahrnehmungsmöglichkeiten für Berührungen erschlossen wurden, und wie diese genutzt wurden, um das Interaktionsverhalten des Roboters noch intuitiver zu gestalten. Durch Einführung eines Emotionsmodells wurde gezeigt, wie sich der Roboter dem Nutzungsverhalten und Tagesrhythmus des Nutzers anpassen kann und somit möglichst unaufdringlich seine Services am Nutzer erbringt. Verschiedene Möglichkeiten zur Darstellung simulierter Emotionen wurden umgesetzt und geschildert, allerdings bleibt die Frage nach der Wirkung eines solchen Ansatzes auf die Beziehung zwischen Roboter und Mensch für sozialwissenschaftliche Studien offen. Es hat sich in den durchgeführten Nutzertests zumindest kein negativer Einfluss auf die Funktionsfähigkeit des Gesamtsystems gezeigt.
Nicht ganz soweit fortgeschritten stellte sich die Realisierung eines Taskschedulers dar, welcher die Ausführungszeitpunkte von Interaktionsaufgaben an die Verfügbarkeit des Nutzers anzupassen hilft. Mangels einer robusten Vorhersagemöglichkeit für die Interaktionsbereitschaft des Nutzers bringt das vorgestellte Konzept zum gegenwärtigen Zeitpunkt keine Vorteile gegenüber einer rein zeit- und eventgetriggerten Interaktion. Im SERROGA System wurde daher auf eine Realisierung eines Taskschedulers verzichtet, und dennoch konnten sehr positive Ergebnisse in realen Nutzertests erreicht werden, wie im folgenden Kapitel gezeigt werden wird.

# 9




# Einsatz des SERROGA Systems

Als Nachweis für die Tragfähigkeit des vorgestellten Architekturkonzeptes und des frame-basierten Dialogsystems sollen an dieser Stelle die sehr erfolgreichen Realweltexperimente im Rahmen des SERROGA Projekts angeführt werden.

Am Ende einer kontinuierlichen Reihe von Funktionstests und Benchmarks für die Entwicklung der robotischen Basisfähigkeiten, wie autonome Navigation und Lokalisation in der Wohnumgebung, Wahrnehmung von Personen, sowie der nutzerbezogenen Navigationsverhalten Suchen und Verfolgen einer Person, konnte das Gesamtsystem in verschiedenen Nutzertests seine Praxistauglichkeit unter Beweis stellen.

Dieses Kapitel wird kurz eine Skizze der vorbereitenden Funktionstests geben, bevor das Setup für die realen Nutzertests mit Senioren in deren eigenen Wohnungen und die dabei erzielten Ergebnisse geschildert werden.

Der praktische Einsatz des über mehrere Jahre und Vorgängerprojekte gereiften robotischen Gesundheitsassistenten, dessen zentrale Ablauf- und Dialogsteuerung in dieser Arbeit thematisiert wurde, wird die konzeptionellen und methodischen Beiträge der Dissertation abschließen.

| | Grund-fläche $A$ $[m^2]$ | Frei-fläche $F$ $[m^2]$ | befahrbare Fläche $B$ $[m^2]$ | $\frac{B}{A}$ | $\frac{U}{\sqrt{B}}$ | ges. Weg-länge $[m]$ | mittl. Wand-abstand $[m]$ | mittl. Durchfahrt-breite $[m]$ |
|---|---|---|---|---|---|---|---|---|
| Labor | 50.29 | 32.84 | 17.47 | 0.35 | 14.43 | 44.31 | 0.36 | 0.98 |
| Mitarbeiter 1 | 47.30 | 27.80 | 12.78 | 0.27 | 17.21 | 46.32 | 0.30 | 0.91 |
| Mitarbeiter 2 | 70.76 | 34.19 | 16.71 | 0.24 | 16.28 | 48.21 | 0.32 | 0.8 |
| Mitarbeiter 3 | 73.73 | 43.73 | 22.97 | 0.31 | 17.10 | 58.29 | 0.34 | 0.97 |
| AWO 1 | 35.50 | 18.48 | 9.54 | 0.27 | 11.32 | 27.12 | 0.33 | 0.86 |
| AWO 2 | 32.19 | 15.93 | 7.64 | 0.24 | 11.40 | 20.52 | 0.31 | 0.78 |
| AWO 3 | 29.98 | 16.24 | 6.88 | 0.23 | 13.60 | 27.09 | 0.28 | 0.75 |
| AWO 4 | 31.36 | 14.46 | 7.70 | 0.25 | 9.92 | 17.97 | 0.33 | 0.94 |
| ARTIS 1 | 24.62 | 13.84 | 6.12 | 0.25 | 9.95 | 15.63 | 0.31 | 0.80 |
| ARTIS 2 | 46.24 | 21.65 | 11.28 | 0.24 | 11.57 | 24.81 | 0.33 | 0.89 |
| ARTIS 3 | 29.60 | 12.16 | 6.26 | 0.21 | 9.14 | 14.07 | 0.33 | 0.95 |
| ARTIS 4 | 29.70 | 15.18 | 7.67 | 0.26 | 10.27 | 18.06 | 0.32 | 0.95 |
| ARTIS 5 | 21.59 | 11.45 | 4.75 | 0.22 | 12.09 | 15.96 | 0.26 | 0.81 |

**Tabelle 9.1:** Charakterisierung der Testumgebungen anhand verschiedener Kenngrößen, welche aus der Belegtheitskarte extrahiert wurden. $\frac{U}{\sqrt{B}}$ setzt den Umfang der befahrbaren Fläche ins Verhältnis zur Fläche selbst um einen Formfaktor zu beschreiben. Die gesamte Weglänge bezieht sich auf das Skelett der befahrbaren Fläche und gibt grob die Komplexität der Umgebung wieder.

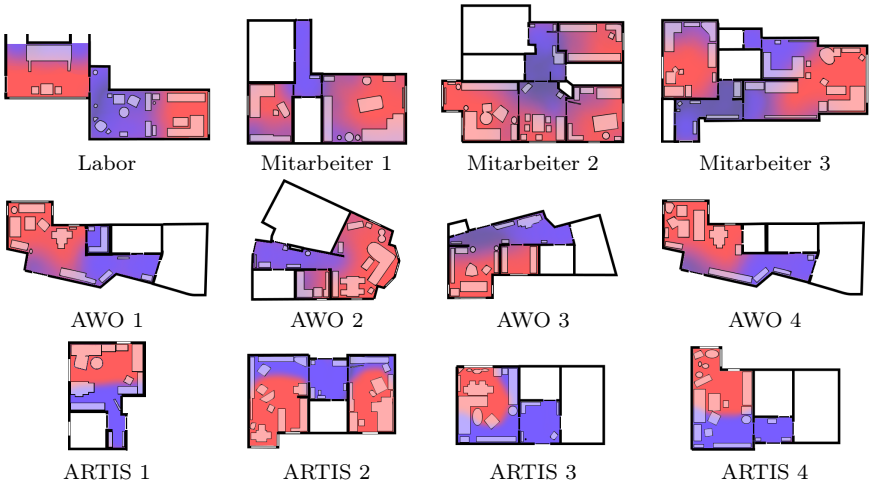



**Abbildung 9.1:** Grundrisse der Testwohnungen, eingefärbt die vom Roboter befahrenen Räume mit Einrichtungsgegenständen; rot helle und blau dunkle Bereiche, welche nur künstliche Beleuchtung erhalten

## 9.1 Einsatz- und Testumgebungen

Bevor die experimentellen Ergebnisse näher erläutert werden, soll zunächst ein Überblick über die genutzten Einsatzumgebungen gegeben werden. Neben dem Labor, welches wohnungsähnlich möbliert wurde, fanden die Funktionstests vornehmlich in drei Wohnungen von Mitarbeitern des Fachgebiets statt. Die Zieleinsatzumgebungen, in denen auch die anschließenden Nutzertests durchgeführt wurden, bilden die Wohnungen der teilnehmenden Senioren zweier Servicewohneinrichtungen in Erfurt. Einerseits die der AWO Wohnanlage am Krämpferufer und andererseits die der ARTIS Wohnanlage ebenfalls in Erfurt. Abb. 9.1 zeigt die Grundrisse der beteiligten Wohnungen.
Um zu überprüfen, inwieweit sich Ergebnisse der Funktionstests auf die, gegenüber dem Labor und den Mitarbeiterwohnungen doch recht unterschiedlichen, Wohnungen der Senioren übertragen lassen, wurden verschiedene Komplexitätsmaße anhand der Grundrisse der Wohnungen abgeleitet (siehe Tabelle 9.1). Dabei zeigte sich, dass die realen Wohnungen etwas dichter möbliert sind, als dies im Labor erreicht wurde. Die Wandabstände und Durchfahrts-

breiten, welche für die Navigationsfähigkeiten entscheidend sind, liegen jedoch in vergleichbaren Bereichen. Lediglich die Größe und damit auch die zu fahrenden Strecken sind in den Seniorenwohnungen wesentlich kleiner, wodurch sich die Suche nach dem Nutzer nur vereinfacht. In allen Umgebungen herrschen sehr wechselnde Beleuchtungsbedingungen, was durch große Fenster und teilweise vorhandene Räumlichkeiten ohne Tageslicht bedingt ist. Die Lichtsituation spielt für die Navigation eher keine Rolle, da diese über aktive Sensoren (Laser und IR-basierte Tiefenkameras) abgesichert ist, allerdings könnte sich die Helligkeit auf die teilweise kamera-basierte Personenwahrnehmung auswirken.

## 9.2 Vorbereitende Funktionstests

Es stellte sich als äußerst wichtig heraus, dass im Rahmen der Projektlaufzeit frühzeitig Funktionstests einzelner Basisleistungen in zahlreichen realen Wohnungen durchgeführt wurden. Nur durch die Entdeckung von Fehlverhalten, welches erst in immer neuen Situationen in den abwechslungsreichen Testumgebungen zu Tage trat, konnte am Ende ein robustes System entstehen, welches mehrere Tage autonom in vollkommen neuen Wohnungen agierte und einen reibungslosen Dauerbetrieb ermöglichte.
Zentrales Element für einen autonomen Betrieb stellen die Navigationsleistungen des Roboters dar. Ohne die Fähigkeit, robust eine vorgegebene Zielposition in seiner Einsatzumgebung anzufahren, wäre es auch nicht möglich, dass der Roboter sich selbst zum Aufladen seiner Akkus auf die Ladestation begibt. Somit wäre das System immer auf einen Betreuer angewiesen. In einer ersten Testreihe wurde daher zunächst im Labor und später auch in den Wohnungen von Mitarbeitern des Fachgebiets das Anfahren von Zielen evaluiert.
Insgesamt wurden dabei ca. 7,5 km in vier Umgebungen zurückgelegt und ca. 1350 Zielpunkte angefahren. Unter anderem wurde dabei eine Positionierungsgenauigkeit von 3-7 cm erreicht, was für die angestrebten Zwecke (Interaktion mit dem Nutzer) ausreichend erscheint. Es wurden auch 42 Ziele nicht erfolgreich erreicht, wobei in 16 Fällen eine kritische Situation eintrat, in der eine Kollision mit Gegenständen der Umgebung auftrat, die der Roboter aufgrund seiner Sensorkonfiguration nicht wahrnehmen konnte. Diese Erkenntnisse ha-

ben dabei geholfen, in den realen Nutzertests solche Situationen zu vermeiden und entsprechende Objekte aus dem Gefahrenbereich zu entfernen, bzw. dem Roboter diese Bereiche der Umgebung zu verbieten.

Als zweite wesentliche Navigationsleistung wurde das interaktive Navigationsbehavior **Verfolgen** einer Person getestet. Dazu wurden wiederum im Labor, in den drei Mitarbeiterwohnungen sowie in den Wohnungen der Senioren bei der Einrichtung des Systems die Verhaltensweise mit den realistischen Nutzern und zusätzlich mit einem erfahrenen Nutzer als Referenz erprobt. Der Tester versuchte dabei, den Roboter zwischen verschiedenen vordefinierten Punkten hin und her zu lotsen. Dabei wurde neben der Navigationsfähigkeit auch die Wahrnehmung von Personen unter verschiedensten Beleuchtungssituationen (von Dunkelheit im langen Flur bis zu Blendung durch direkte Sonneneinstrahlung durch die Fenster) auf die Probe gestellt.

Es zeigte sich, dass der Erfolg dabei stark von der Erfahrung der Testperson abhängt. Ein erfahrener Nutzer konnte den Roboter in allen Umgebungssituationen mit einer durchschnittlichen Geschwindigkeit von 0.22 m/s erfolgreich führen. Die Senioren hingegen hatten andere Vorstellungen von den Wahrnehmungsfähigkeiten des Roboters und kamen bei Verzögerungen teilweise auf ihn zu, wodurch die Navigation noch mehr behindert wurde. Im Ganzen konnten die uneingewiesenen Nutzer den Roboter mit 0.13 m/s lotsen, wobei insgesamt eine Erfolgsrate[1] von 90% erreicht wurde.

Die **Suche nach dem Nutzer** wurde als weiteres essentielles Navigationsverhalten erprobt. Dazu wurden wiederum in den Mitarbeiter- und Seniorenwohnungen verschiedene Suchfahrten durchgeführt. Die Testperson stellte oder setzte sich dazu an eine beliebige Position in der Einsatzumgebung und der Roboter begann von einer zufälligen Startposition aus die Suche. In den weitläufigeren Mitarbeiterwohnungen konnte dabei der Nutzer in 75% von 54 Versuchen und in den Seniorenwohnungen in 95% von 44 Versuchen gefunden werden. Die mittlere Suchdauer lag dabei zwischen 1:00 und 1:30 Minuten (maximal 3:53 min). In fehlgeschlagenen Versuchen kam der Roboter zu dem Schluss, dass keine Person in der Wohnung gewesen sei, oder er fuhr gar nicht erst los, da er den Nutzer vor sich vermutete.

[1] Erfolgreich war ein Lotsendurchgang, wenn er nicht abgebrochen werden musste, weil der Roboter offensichtlich zu einer anderen Position fuhr.

Die Herangehensweise und Ergebnisse der Benchmarks für die Basisleistungen sind in [Gross et al., 2015][2] zusammengefasst und mit weiterem Zahlenmaterial unterfüttert.

## 9.3 Nutzertests mit dem SERROGA System

Einen näheren Bezug zum Thema dieser Arbeit haben die im Anschluss an die Funktionstests durchgeführten Nutzertests. Hierbei wurde das Gesamtsystem mit unerfahrenen Nutzern konfrontiert, um die Praktikabilität der Navigationseigenschaften, aber auch der Nutzservices und damit die der Ablaufsteuerung und des Dialogsystems zu evaluieren.

### 9.3.1 Funktionsumfang des getesteten Demonstrators

Letztendlich konnte mit dem Demonstratorsystem ein Funktionsumfang realisiert werden, welcher Beispiele aus den verschiedenen Rollen (siehe Kap. 2) eines häuslichen Assistenzroboters abdeckt. Die angebotenen Services umfassten **Terminverwaltung mit Erinnerungsfunktion, Aktivierungsvorschläge**, Kommunikationsassistenz durch **Videotelefonie, Informationsservices** in Form von Wettervorhersage und -warnungen, **Vitalparametermessung** (mittels eines Fingerclip Pulsoximeters oder berührungslose Pulsmessung im Kamerabild [Stricker et al., 2014][3]) sowie die **Robotersteuerung**. Die bereitgestellten Navigationsfähigkeiten umfassten das Rufen des Roboters über eine Fernbedienung mittels der Personensuche, das Verfolgen des Nutzers, eine autonome Navigation zu Zielpunkten und das selbständige Andocken an die Ladestation. Weiterhin konnte sich der Nutzer beim Roboter an- und abmelden, um Privatsphäre und Ruhephasen zu gewährleisten.
Bezüglich der weiteren in dieser Arbeit beschriebenen Hardware- und Softwarefeatures waren bei den Nutzertests das Emotionsmodell (Abschn. 8.2) sowie das direkte akustische Feedback auf Berührungsmuster am Streichelsensor (Abschn. 8.1.1) integriert. Ebenso war das assistierte Schieben des Roboters mittels der Berührungssensorik im Einsatz (Abschn. 8.1.2). Die Darstellung emotionaler Augenausdrücke war leider noch nicht einsatzbereit und konnte

[2] Der Autor ist Co-Autor dieses Papers.
[3] Der Autor ist Co-Autor dieses Papers.

daher nicht mit getestet werden. Die Darstellung von Emotionen beschränkte sich somit auf die Färbung der GUI-Hintergründe und die Variation der Sprachausgaben und Texte.

Die Umsetzung der Service-Apps erfolgte mit dem in dieser Arbeit entwickelten Dialogsystem, wie es in Kapitel 6 geschildert wurde. Die multimodalen Ausgaben umfassten dabei GUI, Text- und Sprachausgaben. Die Nutzereingaben geschahen lediglich über die GUI bzw. eine Fernbedienung zum Rufen und Stoppen des Roboters. Die Spracheingabemodalität wurde aufgrund mangelnder Robustheit der Spracherkennung ohne Ansteckmikrofon **nicht** zum Einsatz gebracht.

Die Adaption der Dialogverläufe, wie in Kapitel 7 eingeführt, wurde im Rahmen dieser Tests ebenfalls **nicht** genutzt. Die mit 1-3 Tagen angesetzte Interaktionszeit wäre zu kurz gewesen, um eine merkliche Adaption zu erwarten und der Fokus im SERROGA Projekt lag nicht auf der Adaptivität des Interaktionsverhaltens. Außerdem würde die konsequente Umsetzung der Adaptivität einen Mehraufwand bei der Definition der Dialogverläufe in den Service-Apps erfordern.

Ebenso **nicht** im Einsatz war das situationsangepasste Timing der Interaktionstasks mittels eines Taskschedulers (Abschn. 8.3). Erstens sind die Voraussetzungen, die Nutzersituation zuverlässig zu schätzen und vorauszusagen, im gegebenen Szenario ohne die Integration von Smart-Home-Sensorik nicht gegeben gewesen, und zweitens bieten auch die Services nur einen geringen Bedarf für ein angepasstes Timing. Die Erinnerung an Termine geschah zeitgesteuert, was in der künstlichen Testsituation als angemessen zu erachten ist. Die zweite robotergetriggerte Interaktionssituation betrifft eingehende Telefonate. Dabei ist der Einsatz eines Taskschedulers nicht sinnvoll, da der Auslöser von Außen kommt. Der dritte Fall ist das Ausliefern von Aktivierungsvorschlägen. Hierbei wurde mangels wahrnehmbarer Triggerereignisse ebenfalls auf eine rein zeitgesteuerte Lösung zurückgegriffen, wobei ein Repertoire an Hinweisen, welche der Roboter geben sollte, in der Vorbesprechung mit den Senioren festgelegt wurde. Beispiele dafür wären Erinnerungen daran, etwas zu trinken, einen Spaziergang zu unternehmen oder sich auf den Gang zum Mittagstisch vorzubereiten.

### 9.3.2 Testdurchführung

Mit diesem Setup wurden im Frühjahr 2015 in zwei Einrichtungen für betreutes Wohnen in Erfurt (AWO Wohnanlage am Krämpferufer und Artis Service-Wohnen am Südpark) mit 9 Senioren unterschiedliche Nutzertests durchgeführt. Anfänglich fanden mit allen Teilnehmern einstündige Useability-Tests unter Beobachtung durch sozialwissenschaftliche Versuchsbegleiter statt, bei denen die einzelnen Funktionen separat getestet wurden. Ergebnisse dazu sind in [Döring et al., 2015] zu finden.

Mit zunehmendem Vertrauen der Entwickler ins Robotersystem und die Infrastruktur wurde die Versuchsdauer immer weiter verlängert und die Zielstellung verlagert bis zu dreitägigen Langzeittests, bei denen die Senioren ihr normales Tagesgeschehen unter Begleitung des Roboters absolvieren sollten. Technisches Personal und Beobachter waren dabei nicht mehr präsent, sondern standen nur auf Abruf bereit. Eine Protokollierung der Nutzungsweise wurde durch einen Logging-Mechanismus in der Robotersoftware realisiert. Außerdem sollten die Nutzer nach der ersten Nutzung jedes Services einen Fragebogen ausfüllen. Direkte Fragen und eine Einschätzung der Situation in den Wohnungen konnten über die gelegentliche Nutzung der Videotelefoniefunktion realisiert werden. Nur für absolute Notfälle war ein Versuchsbetreuer, der sich in der Nähe der Wohnung befand, telefonisch zu erreichen[4].

Die Versuche begannen mit der Einrichtung des Roboters vor Ort (Aufnahme einer Navigationskarte, festlegen von Navigationszielen, Ruheposition und Position für die Ladestation), woran sich eine Demonstration der Services anschloss. Den Nutzern wurde so Gelegenheit gegeben, die Funktionsweise zu erlernen und Fragen zum Umgang mit dem Roboter zu klären. Dies war auch notwendig, da eine erklärende Einführung der Services durch den Roboter selbst, wie es bei der Office Mate Studie implementiert war (siehe Abschn. 7.1), nicht vorgesehen war.

Die eigentlichen Nutzungstest begannen am darauf folgenden Tag, indem ein Mitarbeiter den Roboter in Betrieb nahm und ihn dem Nutzer für den Tag überließ. Anfangs wurde der Roboter allabendlich aus der Wohnung abgeholt.

[4]Dies war lediglich einmal erforderlich, als der Roboter ohne ausreichende Spannungsversorgung ausging.

Nach den ersten erfolgreichen Versuchen ohne größere Schwierigkeiten wurde dazu übergegangen, den Roboter auch über Nacht in der Wohnung der Testnutzer zu belassen (in Tab. 9.2 die mit aktiv übernachtet gekennzeichneten Versuche).

Auf die durchweg positiven Ergebnisse dieser Versuche wird in [Gross et al., 2015][5] eingegangen. An dieser Stelle sollen einige ausgewählte Ergebnisse der sozialwissenschaftlichen Auswertung der Experimente dargestellt werden, da sie Rückschlüsse über die erfolgreiche Umsetzung der Ziele dieser Arbeit ermöglichen.

## 9.4 Ergebnisse und Fazit

Die evaluativen Nutzertests wurden als Fallstudiendesign angelegt, da der Einsatz des Roboters einen hohen Grad an Komplexität aufweist und der Vorbereitungsaufwand für die Infrastruktur und Versuchsdurchführung enorm ist. Somit war es nicht möglich, eine größere Fallzahl zu realisieren, wie sie für eine quantitative Auswertung erforderlich gewesen wäre. Mit den neun Kandidaten konnte dennoch ein breites Spektrum an Nutzern verschiedenster Konditionen und Hintergründe abgedeckt werden (siehe Tab. 9.2), wobei die gezogenen Schlüsse versuchen, über alle Individuen zu mitteln.

Nach der Nutzung des Roboters wurde durch die Sozialwissenschaftler ein semi-strukturiertes Interview mit den Versuchsteilnehmern durchgeführt. Diese persönlichen Eindrücke, die Auswertung der Logdaten und der servicebezogenen Fragebögen dienten zur Analyse von Fragestellungen nach der Nützlichkeit und Akzeptanz eines robotischen Begleiters, nach dem Erfolg bei der Nutzung des Systems und nach der Absicht, den Roboter in Zukunft nutzen zu wollen. Weiterhin waren die sozio-emotionale Beziehung zum Roboter, das Sicherheitsgefühl und der Spaß bei der Nutzung von Interesse, worüber eine Analyse der Interviewdaten Aufschluss gewährte.

[5] Der Autor ist Co-Autor dieses Papers.

| Nutzer | Alter Geschlecht | Technik-affinität | Roboter bekannt | Versuchs-dauer | Nutzungs-zeit | Über-nachtung |
|---|---|---|---|---|---|---|
| AWO 1 | 68/m | hoch | teilweise | 1d (6:04 h) | 1:10 h | - |
| AWO 2 | 79/w | mittel | gut | 1d (5:09 h) | 1:33 h | - |
| AWO 3 | 86/w | hoch | gut | 1d (4:08 h) | 1:29 h | - |
| AWO 4 | 72/w | mittel | teilweise | 1d (5:44 h) | 1:53 h | - |
| ARTIS 1 | 92/w | hoch | nein | 2d (9:28 h) | 4:19 h | ausgeschaltet |
| ARTIS 2 | 75/m | hoch | nein | 3d (17:41 h) | 3:29 h | 1x aktiv |
| ARTIS 3 | 86/m | mittel | nein | 3d (28:17 h) | 4:15 h | ausgeschaltet |
| ARTIS 4 | 80/w | niedrig | nein | 3d (35:36 h) | 1:41 h | 1x aktiv |
| ARTIS 5 | 90/w | niedrig | nein | 1d (7:16 h) | 1:59 h | - |

**Tabelle 9.2:** Übersicht zu den Nutzertests bei der AWO und ARTIS, Versuchsdauer mit Nettobetriebszeit des Roboters

### Resultate

Die Akzeptanz eines Assistenzsystems basiert auf der robusten Funktionsfähigkeit aller angebotenen Services und dem realisierten Mehrwert gegenüber anderen Hilfsmitteln. Der Umgang mit dem Roboter erfordert etwas Übung und konnte von den Teilnehmern letztendlich beherrscht werden. Die erste Voraussetzung für Akzeptanz konnte also größtenteils erfüllt werden. Die Nützlichkeit des Assistenzroboters im getesteten Zustand wurde von allen Teilnehmern als eher gering eingestuft, alle waren sich aber bewusst, dass es nur ein Demonstrator war. Mit der Option auf mehr Servicefunktionen konnten sich 8 der 9 Testpersonen vorstellen, den Roboter in Zukunft zu nutzen und sahen einen Wert auch als Begleiter und Zeitvertreib.

In der kurzen Versuchslaufzeit wurde keine realistische Alltagssituation erreicht. Alle Teilnehmer empfanden den Roboter als willkommene Abwechslung und beschäftigten sich intensiv mit dem System. Bei den mehrtägigen Versuchen konnte aber ein Abfall des Interesses beobachtet werden. Ein Teilnehmer bemerkte, dass ihm am zweiten Tag die Ideen ausgingen, was er noch mit dem Roboter anstellen sollte. Ein gewisser Joy-of-Use war bei allen Senioren gegeben.

Für die Wahrnehmung des Roboters als sozialer Begleiter konnten folgende Beobachtungen gemacht werden: Alle Kandidaten fühlten sich sicher in Gegenwart des Roboters, zögerten aber, den Roboter allein in ihrer Wohnung zu lassen. Nur diejenigen Nutzer, die länger Gelegenheit hatten, den Roboter zu nutzen, gewannen letztendlich entsprechendes Vertrauen. Keiner der Senioren hatte Angst, den Roboter zu beschädigen.

Die Ergebnisse bezüglich der sozio-emotionalen Beziehung zum Roboter waren erstaunlich. Bis auf einen Fall behandelten die Senioren den Roboter wie ein soziales Wesen, wenn sie mit ihm interagierten oder über ihn sprachen. Sie gaben ihm individuelle Namen, sprachen zu ihm und kommentierten sein Verhalten obwohl sie wussten, dass er keine Spracheingabe versteht und keine Aufzeichnungen der Kommentare durchgeführt wurde. Sie begrüßten den Roboter, streichelten ihn und bedauerten sichtbar, wenn sie sich von ihm verabschieden mussten. Die Senioren reagierten auch emotional auf die Aktivitäten des Roboters. Sie lobten ihn, bedauerten Fehler und sorgten sich um

seinen Zustand. Gelegentlich fragten sie ihn nach seiner Meinung. Die Nutzer entwickelten dieses Verhalten obwohl sie sich bewusst waren, dass es sich nur um eine Maschine handelt. Trotzdem fühlten sich die Senioren mit dem Roboter nicht mehr allein und empfanden weniger Langeweile. Die Senioren meinten, je mehr proaktive Verhalten der Roboter habe, desto unterhaltsamer wäre er.

### Fazit

Für die Bewertung der Beiträge der vorliegenden Arbeit lassen sich aus den Versuchen folgende Schlüsse ziehen: Bei den absolut unvoreingenommenen Teilnehmern traten mit dem Interaktionsdesign keine größeren Verständnisprobleme auf. Die Multimodalität der Ausgaben und das turn-basierte Dialogregime sind für die menschlichen Interaktionspartner also intuitiv und gut nachvollziehbar. Weiterhin lässt sich aus den emotionalen Reaktionen folgern, dass es mit dem Interaktionsdesign und der Gestaltung des Roboters an sich gelungen ist, dem Roboter eine Persönlichkeit zu verleihen, wie es ein Ziel dieser Arbeit war. Nicht zuletzt hat der erfolgreiche autonome Langzeiteinsatz demonstriert, dass die zugrundeliegende Systemarchitektur und die Ablaufsteuerung, wie sie im ersten Teil der Arbeit vorgestellt wurden, tragfähig sind. Insbesondere erwiesen sich die Verteilung der Ablaufsteuerung auf einen Zustandsautomaten zur Realisierung von Sicherheit und Selbsterhaltung und das Dialogsystem zur Verwaltung aller nutzerbezogenen Aktivitäten als sehr vielversprechend.
Alle Aspekte der Adaptivität wie sie in Kapitel 7 angesprochen wurden und die Anteile der einzelnen Komponenten (Streichelfell, turn-basierter multimodaler Dialog, Emotionsmodell) am positiven Gesamteindruck bleiben noch ungeklärt und sollten in separaten Experimenten in Zukunft weiter untersucht werden.
Die positiven Reaktionen während der Tests mit realen nicht robotikaffinen Nutzern geben eine Bestätigung, dass Forschung an robotischen Assistenzsystemen für jedermann durchaus zukunftsträchtig und sinnvoll ist.

# 10

# Zusammenfassung

Dieses Kapitel resümiert noch einmal die vorgestellten Aspekte für die Realisierung eines nutzeradaptiven Interaktionsverhaltens für mobile Assistenzroboter.

Als reale Anwendung eines Assistenzroboters diente in der vorliegenden Arbeit der Demonstrator des SERROGA Projektes[1], welcher als sozialer Gesundheitsassistent in der häuslichen Einsatzumgebung dienen sollte (Kap. 2).

Als erstes wurde an diesem Beispiel die Herangehensweise für den Systementwurf geschildert und daraus Bezüge zu einer zugrundeliegenden schichtweisen Systemarchitektur abgeleitet. Beginnend bei abstrakten Rollen, die ein Roboter im Zusammenleben mit dem Mensch einnehmen kann, über konkrete Services, die er erbringen soll sowie die dafür benötigten Verhaltensweisen, konnten letztendlich konkrete robotische Basisfunktionalitäten und Hardwareanforderungen abgeleitet werden.

Die Beiträge der Arbeit umfassen weiterhin das Interaktionsdesign für einen sozialen Assistenzroboter mit dem Ziel, einen natürlichen, intuitiv verständlichen Dialog zu erreichen und dem Roboter eine Persönlichkeit zu geben, die eine intensive emotionale Bindung im Langzeiteinsatz ermöglichen soll. Praktisch umgesetzt und eingeführt wurde dieses Interaktionsdesign am Beispiel des SERROGA Demonstrators.

[1] SERvice RObotik für die GesundheitsAssistenz

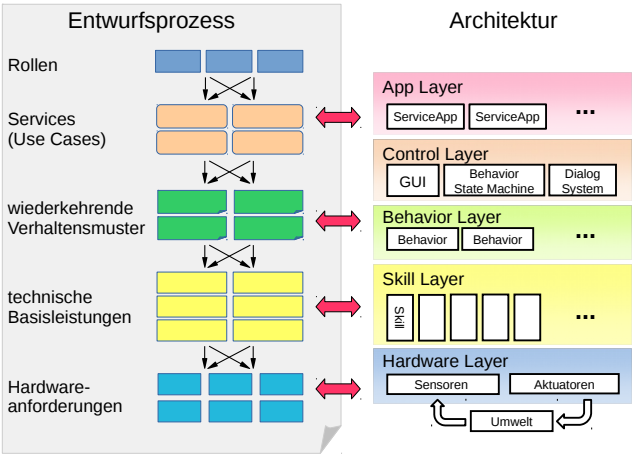



**Abbildung 10.1:** Entwurfsprozess eines Assistenzroboters und Korrespondenzen zur vorgeschlagenen Systemarchitektur

In Kap. 4 stellt die Arbeit ein Konzept für eine Softwarearchitektur vor, welche in ihren Schichten den Entwurfsprozess des Assistenzroboters widerspiegelt (siehe Abb. 10.1). Auf diesem über mehrere Robotikprojekte gereiften Konzept setzt die gesamte praktische Realisierung des Demonstrators auf.

Herzstück der Architektur und Schwerpunkt im Kap. 5 bildete das Konzept für eine modulare Ablaufsteuerung mit einer Trennung der Teile, die Sicherheit, Navigation und Selbsterhaltung realisieren, von allen die Nutzerinteraktion betreffenden.

Das multimodale Dialogsystem und weitere Komponenten der Control Layer (Kap. 5) wurden unter Berücksichtigung eines weiteren zentralen Aspektes dieser Arbeit, der Adaptionsfähigkeit an Vorlieben und Eigenheiten des Nutzers, designt. Eine theoretische Betrachtung der Möglichkeiten und Notwendigkeit von Nutzeradaption wurde in Kap. 3 gegeben. Dabei entstand der Anspruch, vier verschiedene Aspekte der Adaption zu realisieren, wie sie in Abb. 10.2 dargestellt sind.

Davon konnte für alle Aspekte eine konzeptionelle Berücksichtigung beim Entwurf des Dialogsystems erreicht werden. Bis auf die Anpassung der In-

putinterpretation (Abschn. 6.9.2) wurden alle vier Aspekte mit praktischen Umsetzungen erprobt. Sehr erfolgreich konnten die Nutzung von Nutzerpräferenzen und die Adaption der Dialogverläufe umgesetzt werden. Mehr dazu im Anschluss. Die Optimierung des Timings für die proaktiven Roboterverhalten, wie sie mittels eines Taskschedulers (Abschn. 8.3) vorgesehen ist, konnte bislang keine zufriedenstellenden Ergebnisse liefern. Der Grund dafür liegt in den unzureichend beobachtbaren Nutzeraktivitäten und der damit sehr beschränkten Vorhersagbarkeit von Zeiten, in denen der Nutzer verfügbar ist.

Im Konzept des Dialoges mit dem Roboter sollte eine intuitive und natürliche Kommunikation durch Multimodalität bei den Ein- und Ausgaben realisiert werden. Diese Ansprüche konnten durch die Entwicklung eines multimodalen Dialogsystems, welches modular durch unabhängige Service-Apps konfigurierbar ist, erfüllt werden. Hierbei wurde, aufsetzend auf dem klassischen Aufbau von Dialogsystemen bestehend aus Inputfusion, Dialogmanager und Outputgenerierung, eine eigene auf einer framebasierten Modellierung beruhende Umsetzung vorgestellt (Kap. 6). Durch die Implementierung einer probabilistischen Inputfusion und eine skript-basierte Ausgabegenerierung im Dialogsystem wurde die Multimodalität realisiert. Die Ausgaben über GUI-Anzeige und Sprachausgaben konnten aufgrund einer offline arbeitenden Sprachsynthese lediglich mit vordefinierten Phrasen geschehen. Das modulare Design erlaubt es aber, die Natural Language Generation leicht durch komplexere Methoden zu ersetzen. Auf Eingabeseite konnten eine exemplarische Anbindung einer Spracherkennung und die GUI-basierte Eingabe realisiert werden, wobei für die Sprachmodalität eigens eine Aufmerksamkeitssteuerung vorgesehen wurde. Diese dient dazu zu unterscheiden, ob erkannte Eingaben an den Roboter gerichtet sind oder nicht.

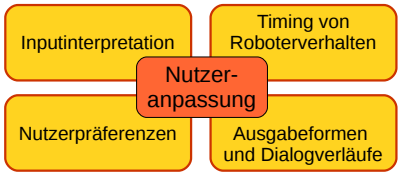



**Abbildung 10.2:** Erscheinungsformen für Adaption in der Mensch-Roboter Interaktion

Um das System abwechslungsreicher zu gestalten und den Eindruck des Roboters als soziales Wesens zu fördern, wurde ein Emotionsmodell entworfen (Abschn. 8.2), welches ebenfalls auf die multimodale Ausgabegenerierung ein-

wirkt. Neben den variablen Sprachausgaben und Texten wird vom emotionalen Zustand auch die Augenmimik gesteuert. Hierzu wurde im Rahmen der Arbeit für den Roboter eine Möglichkeit geschaffen, die als Displays ausgeführten Augen mit entsprechenden Animationen anzusteuern (Abschn. 8.2.6).
Den zentralen Beitrag für die Umsetzung der angestrebten Adaptivität stellt die probabilistisch planende Dialogsteuerung (Kap. 7) dar. Mit Hilfe dieser wird es dem vorgeschlagenen Dialogsystem ermöglicht, anhand online beobachteter Nutzerreaktionen das Verhalten des Roboters im Dialog zu optimieren und Nutzerpräferenzen zu berücksichtigen, wie in Abb. 10.2 genannt. Ermöglicht wurde dieser Ansatz durch die Einführung einer spärlichen Repräsentationsform zur Darstellung von Wahrscheinlichkeitsverteilungen über hochdimensionalen Dialogzuständen und deren Übergängen. Erst dadurch wurde es möglich, die aus der Literatur bekannten Algorithmen zur Inferenz in graphischen Modellen, konkret den Max-Product-Algorithmus für Faktorgraphen, auf die Problematik der Dialogplanung anzuwenden, was in der konkreten Art und Weise erstmalig in dieser Arbeit geschieht. Die erreichte Online-Planung von Dialogsequenzen stellt somit eine Alternative zu aus der Literatur bekannten, auf Reinforcement-Learning basierten Ansätzen für die Optimierung von Mensch-Maschine Dialogen dar. Die Tragfähigkeit dieser Lösung konnte anhand eines experimentellen Setups (Office Mate Abschn. 7.1) mit realen Nutzern verifiziert werden.
Das Interaktionsdesign hatte zum Ziel, dem Roboter eine Persönlichkeit zu geben und ihn für den Nutzer als ein soziales Wesen erscheinen zu lassen. Einen weiteren Beitrag dazu leistet die taktile Sensorik des genutzten Roboters. Es wurde in Kap. 8 zum einen ein Fellsensor zur Erfassung von Berührungsmustern wie Streicheln, Kitzeln oder Schlagen realisiert, welcher neben der direkten akustischen Reaktion auf die Berührungen auch als Quelle für das zur Optimierung genutzte Feedback der Dialogpartner dient. Zum anderen wurde eine kapazitive Touchsensorik für die gesamte Außenhülle des Roboters geschaffen, welche es dem Nutzer ermöglicht, den Roboter, durch seine eigenen Motoren unterstützt, zu schieben, um ihn lokal zu positionieren (Abschn. 8.1.2).
Zu guter Letzt wurde das vorgeschlagene System auch praktisch implementiert und konnte erfolgreich in umfangreichen Nutzertests demonstrieren, dass

die mit dem Design einhergehenden Ziele, einen natürlichen Kommunikationsablauf zu gewährleisten und einen sozialen Begleiter für den Langzeiteinsatz in der häuslichen Umgebung zu schaffen, erreicht werden konnten.
Offen geblieben ist ein Einsatz aller vorgestellten Teilfunktionen vereint in einem Versuchsszenario.

## Ausblick

Im Rahmen der Arbeit wurde zwar unter Mitwirkung vieler Kollegen ein komplettes, voll funktionstüchtiges Robotersystem entwickelt, aber dennoch besteht an vielen Stellen weiterer Forschungsbedarf. Manche der vorgestellten Teilfunktionen sind lediglich als Konzept vorgesehen und müssen noch detailliert ausgearbeitet, implementiert und experimentell evaluiert werden. Beispiele hierfür sind die Adaption der Inputinterpretation und die Vorhersage der Verfügbarkeiten im Taskscheduler.
Für die Implementierung einer Adaption der Interpretationsmodelle für die verschiedenen Inputmodalitäten müssten zunächst weitere Skills, wie beispielsweise die Kopfgestenerkennung oder eine Large-Vocabulary-Spracherkennung für die Transkription freier Texte, realisiert werden. Da bereits ein Gesichtstracker für die Auswertung des Pulssignals im System integriert ist, sollte die Erkennung von Kopfgesten leicht zu realisieren sein.
Auf Ausgabeseite sollte die auf Canned Text[2] basierende Lösung durch eine online Synthese ersetzt werden, welche größere Flexibilität in den Ausdrucksformen und das Rendering von variablen Inhalten wie Zahlen, Uhrzeiten und Vorlesen von Texten oder Nutzereingaben ermöglichen würde.
Wie bereits in Abschn. 7.7 geschildert, bestehen auch im Inferenz- und Planungsalgorithmus noch Möglichkeiten für Verbesserungen. Eine Berücksichtigung von Unabhängigkeitsannahmen bei der Rechnung mit dem komplexen Zustandsvektoren bietet sich dabei als nächster Schritt an. Beobachtete Zustände würden dadurch auch bei anderer Belegung der Kontextvariablen nützliche Informationen liefern und die Generalisierungsfähigkeit verbessern.
Einen weiteren Ansatzpunkt für eine Verbesserung bietet die Realisierung der Exploration und der Rewardgewinnung, wie bereits in der Diskussion (Ab-

[2] Nutzung vorgefertigter und synthetisierter Phrasen

schn. 7.7) an entsprechender Stelle erörtert. Insbesondere für die Rewardgewinnung sollten Langzeitexperimente durchgeführt werden, um Bewertungsmaße der Nutzerzufriedenheit ableiten zu können, welche zum Beispiel die Statistiken der taktilen Interaktionen nutzen.
Für folgende Projekte am interessantesten ist die Weiterentwicklung der emotionalen Ausdrucksmöglichkeiten und der Gesamterscheinung des Roboters. Konkret wird hierfür bereits versucht, das Bewegungsverhalten entsprechend der Emotionsparameter zu modulieren und den in Abschn. 7.7 geschilderten Versuchsablauf zu realisieren, um die Abbildung der Emotionsparameter auf die Darstellungsformen an die Wahrnehmung der Menschen anzupassen.

## Schlusswort

Die vorliegende Dissertation konnte hoffentlich einen Beitrag dazu leisten, die soziale Assistenzrobotik alltagstauglicher zu machen. Angesichts der in zahlreichen Projekten immer wieder angestrebten Zielgruppe der allein lebenden Senioren bleibt aber zu hoffen, dass Roboter, wie clever sie auch sein mögen, niemals menschliche Zuneigung ersetzen müssen.

# Anhang

Im Anhang dieser Arbeit finden zahlreiche Details zur Umsetzung der im Hauptteil angesprochenen Konzepte. Die einzelnen Anhänge sind zur besseren Nachvollziehbarkeit den Kapiteln im Hauptteil zugeordnet. Die Nummerierung bezieht sich somit auf die Kapitelnummerierung im Hauptteil und ist demzufolge nicht immer fortlaufend.

## Inhalte des Anhangs

Zunächst werden in A 1 die mathematischen Symbole für die im Anhang folgenden vertiefenden Beschreibungen gegeben. Im Anschluß daran, folgt in A 2 die Beschreibung der in der Spezifikation des SERROGA Roboters identifizierten Services und Basisverhaltensweisen. Im Sinnen von Use-Cases sollen dabei die vom Nutzer wahrnehmbaren Verhaltensweisen dargestellt werden. Dieses Anhangkapitel bezieht sich auf Kapitel 2 des Hauptteils der Arbeit.
In A 4 erfolgt eine Vertiefung des State of the Art zum Thema der Steuerarchitekturen für mobile Roboter in Ergänzung zu Kapitel 4. Hierbei wird versucht, den geschichtlichen Werdegang aufzuarbeiten.
Auf der Architektur aufsetzend wurde in der Arbeit ein modulares Dialogsystem entwickelt (Kap. 6). A 6.2 geht dabei zunächst auf den State of the Art zum Thema Dialogsysteme ein und ergänzt somit die in Kapitel 6 gegebene Einordnung. Im Anschluss daran folgt ab A 6.6 eine Vertiefung der Details zum implementierten Dialogsystem. Hierbei wird der Ablauf im Dialogmanager wiedergegeben und im weiteren Verlauf die Mathematik hinter der probabilistischen Inputfusion dargestellt.
Das eigentliche Herzstück des Dialogsystems ist der Dialogagent, welcher aus dem Vorrat von möglichen Systemaktionen situationsabhängig eine Auswahl treffen muss. In Kapitel 7 wurde dafür eine auf probabilistischer Planung basierende Umsetzung dieses Dialogagenten vorgestellt. A 7 gibt dafür zunächst die nötigen mathematischen Grundlagen und geht anschließend auf die Details des Planungsalgorithmus ein. Letztendlich erfolgt in A 7.5 eine vertiefende Darstellung der Auswertung von Nutzerpräferenzen zur Vorhersage von potentiellen Nutzereingaben wie sie in Abschn. 7.5 angesprochen wurde.

# A 1

# Mathematische Notation

Hauptsächlich im Anhang, in dem die mathematischen Hintergründe für die im vorderen Teil der Arbeit beschriebenen Konzepte gegeben werden, wird folgende Notation verwendet:

| | |
|---|---|
| Großbuchstaben $X$ | Zufallsgrößen |
| Kleinbuchstaben $x$ | konkretes Ereignis, Realisierung der Zufallsvariablen $X$ |
| $p(X)$ | Verteilungsdichtefunktion über der Zufallsvariable $X$ |
| $P(X = x)$ oder $P(x)$ | Wahrscheinlichkeit für das Eintreten von Ereignis $x$ |
| $K_i$ | semantische Klasse |
| $R(K_i)$ | Wertebereich der semantischen Klasse $K_i$ |
| $v_j \in R(K_i)$ | semantischer Wert aus Klasse $K_i$ |
| $e_k \in E_M$ | Eingabeereignis einer Inputmodalität $M$ |

| | |
|---|---|
| $P(K_i)$ | Wahrscheinlichkeitsverteilung über die Werte einer semantischen Klasse $K_i$ |
| $P(V_k) \in [0,1]$ | Wahrscheinlichkeit des Eingabeereignisses "semantischer Wert $v_k$ kam in der Eingabe vor" |
| $(W_i, C_i)$ | Slot $i$ im Dialog-Frame mit Wert $W_i$ und Konfidenz $C_i$ |
| $Z_j$ | Zähler für Aktion $j$, Teil der History des Dialog-Frames |
| $S'$ oder $S_t$ | Zufallsvariable für den Dialogzustand zu Zeitschritt $t$ |
| $S$ oder $S_{t-1}$ | Vorgängerzustand im Dialogmodell zu Zeitschritt $t-1$ |
| $s_0$ | aktueller Dialogzustand bestehend aus allen $(W_i, C_i)$ und $Z_j$ |
| $A$ | Aktionsmenge / Zufallsvariable im probabilistischen Modell |
| $\hat{A}$ | durch Entscheidungsbaum eingeschränkte Aktionsmenge |
| $a_j$ | konkrete Aktion |
| $G$ | binäre Zufallsgröße für Zielhaftigkeit (Goal) eines Zustands im Dialog |
| $R$ | Zufallsgröße für den Reward im Dialogmodell |
| $w_i^{(A,B,C)}$ | Gewicht eines Samples $i$ in einer Sample-Verteilung welche $p(A,B,C)$ darstellt |
| $s_i^{(A,B,C)}$ | Sample $i$ in einer Sample-Verteilung welche $p(A,B,C)$ darstellt |
| $\delta_Y(s_1, s_2)$ | Ähnlichkeitsfunktion zum Vergleich zweier Samples bezüglich der Dimensionen Y |
| $\mathbf{m} = (P, E)$ | Emotionszustand bestehend aus Pleasure und Excitation |

# A 2

# Der SERROGA Demonstrator (Spezifikation)

In Kap. 2 wurde der Entwurfsprozess zur Realisierung des robotischen Assistenzsystems beschrieben. Dieser Anhang ergänzt Kap. 2 um eine nähere Beschreibung der dort genannten Services und wiederkehrenden Verhaltensmuster, welche schließlich als Service-Apps (Abschn. 4.4.4) bzw. Navigationsbehaviors (Abschn. 4.4.3) implementiert wurden.

## A 2.3 Services

Für einen Gesundheitsassistenzroboter in seinen verschiedenen Rollen wurde folgende Auswahl von Services identifiziert, welche hier eine kurze verbale Charakterisierung im Sinne von Use-Cases erfahren sollen. Alle genannten Services wurden, falls nicht anders beschrieben, im SERROGA Demonstrator (siehe Abschn. 1.1.3) gemäß dieser Vorgaben umgesetzt.

### Videotelefonie

Der Roboter soll als Videotelefon fungieren. Das umfasst das Anrufen, aber auch das Entgegennehmen von Anrufen. Um einen Anruf zu initiieren, soll der Nutzer aus einer Liste von Kontakten wählen können, um den Gesprächspartner anzurufen. Dabei werden alle erreichbaren Kontakte hervorgehoben.

Während des Gesprächs soll der Roboter gesteuert werden können. Entweder manuell oder der Roboter soll dem Nutzer folgen. Auch die Displaystellung soll an die stehende oder sitzende Position des Nutzers angepasst werden können. Diese Funktionen werden durch die Nebenläufigkeit zur Robotersteuerungsapplikation realisiert.
Bei eingehenden Anrufen soll der Roboter dies akustisch ankündigen und den Nutzer in der Wohnung suchen, um ihn zu fragen, ob der Anruf angenommen oder abgelehnt werden soll. Die Annahme der Verbindung kann entweder mit Bild und Ton oder nur als Audioverbindung geschehen.
Hat der Nutzer sich beim Roboter außer Haus gemeldet, oder den „Bitte nicht Stören“ Modus gesetzt, so wird der Anruf nicht durchgestellt. Denkbar wäre in Ergänzung zur implementierten Version, dass der Anruf im Fall der Abwesenheit des Nutzers durch einen Anrufbeantworter entgegengenommen, und dies dem Nutzer beim nächsten Kontakt signalisiert wird.

#### Fernsteuerung

Der Roboter soll als Telepräsenzinterface für Angehörige, Notfallhelfer oder auch den Nutzer selbst dienen. Dabei soll eine Videoübertragung zum Fernsteuerungsgerät bestehen, welches ein Android-Smartphone oder Tablet ist. Außerdem soll der Roboter in verschiedenen Modi bewegt werden können, je nach Qualität und Verzögerung der Bildverbindung. Wenn der Anrufer eine Verbindung zum Roboter aufbauen will, soll der lokale Nutzer am Roboter um Erlaubnis gefragt werden. Dazu muss der Roboter wiederum den Nutzer suchen und die eingehende Anfrage signalisieren. Da sich der Remote-Nutzer identifizieren muss, ist es möglich, dass bestimmte Nutzer ohne Nachfrage automatisch die Roboterkontrolle übernehmen dürfen. Beispielsweise soll der Hauptnutzer selbst, aber auch Notfallhelfer automatisch die Kontrolle übertragen bekommen. Während der Fernsteuerung muss durch den Roboter jederzeit die Kollisionsvermeidung gewährleistet werden.
Für die Realisierung dieser Funktion wurde im Rahmen von mehreren studentischen Arbeiten [Ebert, 2012, Dahn, 2015][1] eine Android-App als Fernsteuergegenstelle realisiert.

[1] Diese Arbeiten wurden vom Autor betreut.

Die implementierten Fernsteuermodi umfassen dabei eine direkte Steuerung der Translations- und Rotationsgeschwindigkeit (Joystick-ähnlich), autonome Navigation zu Punkten, welche im Videobild angeklickt werden (bei langsamer Verbindung, da kein direktes visuelles Feedback notwendig) und eine autonome Navigation zu Punkten welche in der Übersichtskarte der Wohnung ausgewählt werden.
Auf Seite des Roboters kann die Fernbedienung jederzeit durch den Nutzer abgebrochen werden, wenn dieser zu einer anderen Service-App wechselt.
Ein zweite Ausbaustufe der Fernbedienungsfunktion vereint SIP[2] basierte Videotelefonie mit der Fernsteuerfunktion, sodass seitens des Anrufers während des Telefonats die Fernsteuerfunktion hinzu- und abgeschaltet werden kann.

### Terminkalender

Der Nutzer soll mit dem Roboter Zugriff auf seinen Google-Kalender haben. Es soll möglich sein, die Termine in verschiedenen Ansichten zu visualisieren und interaktiv zu editieren. Dabei werden die Details der Termine nacheinander abgefragt, um einerseits nicht an eine GUI-basierte Interaktion gebunden zu sein und andererseits die Nutzer nicht mit zu feingranulären Oberflächen zu konfrontieren. Der durch die Verwaltung eines privaten und eines öffentlichen Google-Kalenders soll es Angehörigen ermöglicht werden, Termine für den Roboterbesitzer zu erstellen und dessen freie Terminslots einzusehen.
In der Implementierung wurde im Rahmen der geforderten Funktionalität eine Wochenübersicht, eine Tagesübersicht und die Anzeige von Termindetails realisiert. Weiterhin existieren Dialoge zum schrittweisen Eingeben neuer Termine und zum Löschen von Einträgen.

### Erinnerungen

Der Kalender stellt die Basis für die Erinnerungsfunktion dar. Hierbei sollen für einen Termin Erinnerungszeitpunkte definiert werden können, zu denen der Roboter den Nutzer aufsucht und ihn an den Kalendereintrag erinnert. Die Erinnerung soll dann entweder bestätigt werden oder der Roboter erinnert zu einem späteren Zeitpunkt noch einmal. Während der Erinnerung sollen die

[2]Session Initiation Protocol (SIP) ein Standard für Videotelefonie

verschiedenen Ansichten des Kalenders verfügbar sein. Beim Erinnern soll die aktuelle Kommunikationssituation berücksichtigt werden. Beispielsweise kann eine Erinnerung auch kurze Zeit vor ihrer eigentlichen Fälligkeit ausgeliefert werden, wenn gerade ein Dialog mit dem Nutzer geführt wurde und dieser zu Ende ist. Ebenso sollte die Auslieferung der Erinnerung verzögert werden, wenn andere wichtige Aktivitäten wie z.B. ein Videotelefonat unterbrochen werden müssten.
Letzteres ist durch die passive Aktivierung des Erinnerungsdialoges realisiert. D.h. der Dialog-Frame wird zwar aktiviert, aber im Stack aktiver Teildialoge (siehe Abschn. 6.6) nicht über momentan laufende Dialoge gestellt.

#### Aktivierung

In Ergänzung zu den Erinnerungen an Termine im Kalender soll die Möglichkeit bestehen, dass der Roboter seinen Nutzer situationsgetriggert oder ebenfalls zeitgesteuert Vorschläge für Aktivitäten macht. Diese umfassen beispielsweise etwas zu trinken, einmal spazieren zu gehen oder an die Vorbereitung des Mittagessens zu denken und gegebenenfalls einkaufen zu gehen. Während der Aktivierung soll der Roboter den Nutzer aufsuchen und ihn fragen, ob er gerade Zeit hat oder ob der Roboter stört. Für den Fall, dass der Nutzer angibt nicht gestört zu werden, soll der entsprechende Vorschlag präsentiert und ein Feedback („Ja mache ich“, „Später vielleicht“ oder „Das gefällt mir nicht“) abgefragt werden.
Die Inhalte der Aktivierungsfunktion wurden bei der Durchführung der Nutzertests gemeinsam mit den Nutzern während der Inbetriebnahme und Vorbesprechung festgelegt.

#### Bewegungstraining

Das Bewegungstraining besteht aus mehreren Unterpunkten, welche in einen Ablauf eingebunden sind. Zunächst gliedert sich das Training in den Kalender ein. Der Nutzer soll sich ein Ziel für das wöchentliche Pensum an Bewegung definieren, für welches dann im Kalender entsprechende Übungstermine eingetragen werden. Weiterhin soll der Erfüllungsgrad der selbst gesteckten Ziele visualisiert und der Nutzer dadurch motiviert werden. Während des eigent-

lichen Trainings soll der Übende stehend oder aus Sicherheitsgründen auch sitzend vor dem Roboter turnen und bekommt von diesem Anweisungen verbal und mittels eines animierten Vorturners. Die Übungsausführung soll überwacht werden, um korrigierendes und lobendes Feedback zu geben. Während der Übung soll der Roboter selbständig zwischen der Interaktionsposition in der Nähe des Nutzers und der Beobachtungsposition ca. 2m vor dem Nutzer hin und her wechseln. Die Übungsstunde soll mit einer Rekapitulation der Erfolge und der Vereinbarung des nächsten Termins abgeschlossen werden.
In der Realisierung konnte diese Form des Bewegungstrainings nicht in den SERROGA Demonstrator übernommen werden. Das Konzept mit, Motivation durch selbst gesteckte Ziele und termingesteuerten Übungssessions, wurde auf einer anderen Roboterplattform in [Geue, 2012][3] als eigenständige Applikation erfolgreich mit einer Gruppe von Senioren im Langzeitversuch erprobt.
Weiterhin konnte eine maschinelle Überwachung der Übungsausführung und eine Generierung von Feedback an den Turner in einer zweiten Stand-Alone-Implementierung demonstriert werden [Gries, 2014, Arntz und Lewandowski, 2015][4].

### Vitalparametermessung

Unter anderem, um die Bewegungsübungen in einem gesunden Schwierigkeitsgrad zu halten und vorab die Übungsfähigkeit des Nutzers abzusichern, soll der Roboter verschiedene Vitalparameter wie Puls-Rate und Sauerstoffsättigung vor der Übungsstunde messen können. Dabei soll der Nutzer verbal und über eine visualisierte Anleitung über den Umgang mit dem Messgerät unterrichtet werden.
Die Pulsmessung ist auch berührungslos durch die Analyse des Videobilds vom Nutzer möglich. Dazu muss der Nutzer ca. 20 sec still sitzend in die Kamera des Roboters schauen. Er erhält auf dem Display ein visuelles Feedback zum Kameraausschnitt und zum Erfolg des Gesichtstrackings. Die Hintergründe zur videobild-basierten Pulsmessung sind in [Stricker et al., 2014][5] geschildert. Die Nutzung eines Pulsoximeters steht nur Beispielhaft für eine Integration

[3] Diese Arbeit wurde vom Autor betreut.
[4] Diese Arbeiten wurden vom Autor betreut.
[5] Der Autor ist Co-Autor dieses Papers.

beliebiger medizinischer Messgeräte auf dem Roboter. Beispielsweise wurde die selbständige Erfassung des Blutdrucks von Medizinern als sehr hilfreich erachtet.

### Robotersteuerung

Um seine Dienste auch an beliebigen Orten in der Einsatzumgebung anbieten zu können, soll der Roboter durch den Nutzer auf verschiedene Weisen instruiert werden können, bestimmte Positionen einzunehmen, aus dem Weg zu gehen oder dem Nutzer zu folgen. Dazu sollen eine Menge von Navigationszielen definiert werden, aus denen der Nutzer auswählen kann. Außerdem soll eine lokale Positionierung / Feinausrichtung möglich sein, damit immer eine angenehme Interaktionsposition eingenommen werden kann. Hierbei kann entweder eine tastenbasierte Steuerung über das Display oder ein unterstütztes Schieben genutzt werden, welches durch die touch-sensitive Oberfläche des Roboters ermöglicht wird (siehe Abschn. 8.1.2). Weiterhin soll die Stellung des Displays vom Nutzer angepasst werden können. Zu diesem Zweck besitzt der Roboter eine motorisierte Neigeeinheit für das Display. Zur Interaktion im Stehen kann das Display hochgeklappt werden. Wenn der Roboter fahren soll, muss das Display aus Sicherheitsgründen eingeklappt werden.

### Ruffunktion

Damit der Nutzer nicht immer zum Roboter gehen muss, wenn er eine der Funktionen nutzen will, soll es eine Möglichkeit geben, den Roboter zu sich zu rufen. Dies wird durch eine Funkfernbedienung realisiert, mittels welcher eine Nutzersuche getriggert werden kann. Im Idealfall sollte dadurch der Roboter beim Nutzer vorbei kommen. Es ist auch denkbar, über Spracheingaben oder mittels der beschränkten Anzahl von Kanälen der Fernbedienung zusätzliche Hinweise zur Aufenthaltsposition des Nut-


**Abbildung A 2.1:** Funkfernbedienung zum Rufen des Roboters

zers zu übermitteln, welche dann bei der Suche zu berücksichtigen sind, um die Suchdauer zu minimieren.

In der Umsetzung des Demonstrators wurden die zur Verfügung stehenden Tasten einer Funkfernbedienung (Abb. A 2.1) mit „Suche den Nutzer“, „Stop“, „Fahre zur Ruheposition“ belegt. Während der Nutzertests konnten die Probanden mittels der „Hinweis“-Taste Zeitpunkte markieren, an denen ihnen im Ablauf ein Problem aufgefallen ist.

#### Informationsservices

Als Beispiel für eine breite Palette an allgemeinen Informationsservices, die ein Assistenzroboter in der Rolle eines Sekretärs oder Mitbewohners erbringen kann, wurde im Rahmen der SERROGA Demonstratorentwicklung der Wetterbericht zur Realisierung ausgewählt.

Der Nutzer soll somit Zugriff auf die aktuellen Wetterdaten, sowie den Ausblick für die nächsten Tage haben. Weiterhin sollen proaktiv Wetterwarnungen in entsprechenden Situationen (während der Wetterdatenabfrage oder vor dem Verlassen der Wohnung) ausgegeben werden.

Realisiert wurde dieser Service mittels des Yahoo Wetter Webservice, über welchen die Wetterdaten abgerufen werden können.

#### Begrüßung und Verabschiedung

Der Nutzer soll die Möglichkeit haben, vor dem zu Bett gehen oder vor dem Verlassen der Wohnung dem System dies mitzuteilen. Dadurch weiß der Roboter, dass der Nutzer nicht durch Telefonate oder Erinnerungen gestört werden will und für andere Interaktionen nicht erreichbar ist. Ebenso kann durch den Roboter bei Heimkehr oder am Morgen eine Begrüßung stattfinden. Bei diesen Gelegenheiten sollen dem Nutzer noch wichtige Statusinformationen wie verpasste Anrufe, anstehende Erinnerungen oder Wetterwarnungen mitgeteilt werden. Das Wissen über die Erreichbarkeit des Nutzers kann auch für den Videotelefonieservice bereitgestellt werden, welcher dann den online-Status entsprechend kommunizieren kann.

## A 2.5 Basisverhalten (Wiederkehrende Verhaltensmuster)

Aus den Services lassen sich zum einen servicespezifische Verhaltenswiesen und zum anderen wiederkehrende, meist auf die Navigation des Roboters bezogene, Basisverhalten identifizieren. Diese Basisverhalten wurden im Hauptteil der Arbeit nur genannt und sollen hier genauer spezifiziert werden.

### Nutzersuche

Eines der häufigsten Verhaltensmuster, welches beispielsweise bei einem eingehenden Anruf oder aber beim Ausliefern einer Erinnerung zum Einsatz kommt, ist die Suche nach dem Nutzer. Der Roboter soll dabei möglichst effizient seine Einsatzumgebung absuchen und sich beim Nutzer bemerkbar machen, damit auch dieser aktiv Interaktionsbereitschaft herstellen kann. Da der Roboter nicht mit absoluter Sicherheit entscheiden kann, ob er eine Person wahrnimmt oder eine Fehlbeobachtung vorliegt, kann die Suche nicht selbständig mit der Entscheidung „Nutzer erreicht“ beendet werden. Vielmehr wird diese Entscheidung dem Nutzer überlassen, sodass der Roboter sich lediglich potentiellen Personenpositionen annähert und, falls eine Reaktion ausbleibt, seine Suche fortsetzt bis die gesamte Einsatzumgebung abgesucht wurde. Dadurch ist es im Falle von mehreren Personen in der Wohnung auch möglich, die Richtige aufzusuchen, auch ohne eine Personenwiedererkennung zu beherrschen. Beim Absuchen der Wohnung kommt, wie in [Volkhardt und Gross, 2013a, Volkhardt und Gross, 2013b] beschrieben, eine Karte mit Aufenthaltswahrscheinlichkeiten zum Einsatz, um vielversprechende Suchpositionen auszuwählen. Diese kann entweder bei der Einrichtung des Systems vorgegeben, oder über längere Zeit vom Roboter autonom gelernt werden. Weiterhin bietet es sich an, falls der Roboter in einer Smart-Home-Umgebung eingebunden ist, wie es im Rahmen des CompanionAble Projekts der Fall war, Daten von Bewegungsmeldern in diese Karte zu integrieren.

### Folgen

Ebenfalls auf die Detektion einer Person im Roboterumfeld angewiesen ist das Folgeverhalten des Roboters. Um von seiner Mobilität während der Serviceerbringung Gebrauch zu machen, soll der Nutzer dem Roboter vorweg gehen können, und dieser folgt ihm. Dieses Folgen wird durch ein kontinuierliches Neuplanen des Weges zu einer ringförmigen Zielregion um den detektierten Nutzer realisiert [Weinrich, 2015]. Da während der Zielanfahrt die Hindernisvermeidung aktiv ist, besteht das Problem, dass der laufende Nutzer als Hindernis wahrgenommen wird. Beim Folgen wird der Roboter daher einen gewissen Mindestabstand halten. Erst wenn der Nutzer stehen bleibt, kommt der Roboter näher heran.
Während des Folgens kann es passieren, dass der Nutzer nicht mehr wahrgenommen werden kann, oder dass der Nutzer für den Roboter nicht mehr erreichbar ist. Darauf muss dieser mittels entsprechender Ausgaben hingewiesen werden, was durch den Service Robotersteuerung übernommen wird.

### Autonome Zielanfahrt

Der Roboter soll selbständig Positionen in der Wohnung anfahren und dort warten können. Dabei kann es ebenso wie beim Folgen passieren, dass ein Zielort nicht erreichbar ist oder gar durch den Nutzer selbst der Pfad zum Ziel blockiert ist. Beide Fälle werden mit entsprechenden Hinweisen an den Nutzer behandelt. Wenn der Roboter am Zielpunkt angekommen ist, so soll er dort für einige Minuten verharren, damit der vom Nutzer vorgesehene Zweck der Zielanfahrt, erfüllt werden kann.

### Interaktion mit dem Nutzer

Während der Interaktion mit dem Nutzer soll der Roboter möglichst still halten und sich nicht autonom bewegen. Experimente mit einem automatischen Ausrichten zum Interaktionspartner ähnlich dem Folgen haben gezeigt, dass die unvorhersehbare Bewegung während einer möglichen Interaktion über das Touchdisplay irritierend wirken kann. Insbesondere durch die unsicher geschätzte Personenposition kommt es immer wieder vor, dass auch bei einem

still stehenden/sitzenden Nutzer ein vorher definierter Zielbereich für eine Interaktion anscheinend verlassen wird. Dadurch wäre bei aktiver Nachführung eine unnötige Korrekturbewegung des Roboters nicht zu vermeiden.

#### Autonome Rückkehr zur Ruheposition

Um für den Nutzer leicht erreichbar zu sein und damit der Roboter den Nutzer bei seinen Alltagsgeschäften im Blick hat, wird in der Wohnung eine Ruheposition definiert. An dieser soll der Roboter sich aufhalten, wenn gerade keine Interaktion mit dem Nutzer stattfindet. Falls für einen gewissen Zeitraum also keine Nutzereingaben mehr vorgenommen werden, so soll der Roboter selbstständig auf seine Ruheposition fahren. Weiterhin ist bei diesem autonomen Verhalten die Selbsterhaltung des Roboters berücksichtigt. Falls der Ladezustand der Akkus zu niedrig wird, soll der Roboter die Ruheposition eigenständig verlassen und an der Ladestation andocken, bis er wieder gerufen wird. Zwei Schwellen für den Ladezustand sollen es ermöglichen, dass der Nutzer möglichst selten während einer Interaktion zum Aufladen der Akkus verlassen werden muss. Eine Schwelle bei ca. 20% der Kapazität veranlasst dabei das Laden während der Ruhephasen auf der Ruheposition, während eine weitere Schwelle von ca. 5% bei Unterschreitung die Interaktion abbricht und zum Andocken führt. Wenn bekannt ist, dass der Nutzer die robotische Unterstützung längere Zeit nicht benötigt (z.B. nachdem er sich verabschiedet und abgemeldet hat), soll außerdem direkt zum Andocken übergegangen werden, anstatt auf der Ruheposition zu warten.

#### Andocken

Bei Bedarf muss der Roboter selbständig an seiner Ladestation andocken. Dazu ist die normale Lokalisationsgenauigkeit nicht ausreichend. Daher besitzt der Roboter eine nach hinten gerichtete Kamera, welche während des zweistufigen Andockvorgangs die Relativposition zur Ladestation anhand eines an dieser angebrachten Schachbrettmusters bestimmen kann. Die erste Phase des Andockens besteht in einer normalen Zielanfahrt mit einer Zielposition ca. 1m vor die Ladestation. Dort angekommen orientiert sich der Roboter mit der Aufnahme für den Andockstutzen zur Ladestation und beginnt eine geregelte

Rückwärtsfahrt. Dabei versucht er sich asymptotisch der Normalen durch den Mittelpunkt der Ladestation anzunähern. Wenn die Position aus einem Korridor um diese Normale abweicht, oder wenn nach erreichen der Dockposition kein Ladestrom fließt, dann beginnt der Roboter den Zyklus von vorn, bis letztendlich die elektrische Verbindung zustande kommt. Das Abdocken bei Interaktionsbedarf geschieht durch eine einfache Zielanfahrt an einen Punkt ca. 20cm vor der Dockposition. Dadurch ist gewährleistet, dass der Roboter senkrecht von der Ladestation losfährt.

#### Manuelle Steuerung

Ist der Roboter an einem Zielpunkt angekommen, soll er durch den Nutzer noch lokal positioniert werden können, um eine angenehme Interaktionsposition zu erreichen. Dazu kann der Roboter unterstützt geschoben werden (siehe Abschn. 8.1.2) oder ein Steuerkreuz auf dem Bildschirm dient als Kontrollmöglichkeit. Ein manuelles Schieben ist für Senioren als Zielgruppe nicht möglich, da die Getriebemotoren des Roboters einen erheblichen Widerstand leisten.

#### Fernsteuerung

Im Rahmen des Fernsteuerservice muss der Roboter Navigationsbefehle von Außerhalb entgegennehmen. Damit dabei die Sicherheit der Nutzer und der Umgebung gewährleistet bleibt, wird das Steuerkommando nicht direkt auf die Motorsteuerung des Roboters gegeben. Es wird vielmehr als Sollwert in einer entsprechenden Objective des DWA-basierten Navigators mit den Objectives für die Kollisionsvermeidung und das Vermeiden verbotener Bereiche kombiniert. Dadurch kann man mittels der Fernsteuerung den Roboter nicht gegen Objekte steuern.

# A 4

# State of the Art zu kognitiven Architekturen

In Ergänzung zu der in Abschnitt 4.2 eingeführten Übersicht zu Spielarten kognitiver Architekturen sollen diese hier noch einmal näher erläutert werden. Abb. A 4.1 zeigt die Spielarten noch einmal in einem Ausschnitt der Grafik 4.1.

### Deliberative Architektur

Die Ursprünge von deliberativen d.h. planenden Ansätzen liegen in der klassischen KI. Nach [Kleinehagenbrock, 2004] folgt man dabei der „Sense-Plan-Act"-Zerlegung, auch Perception Reasoning Execution Konzept genannt [Liu, 2005]. Die Modularisierung bildet dabei eine Hierarchie in welcher der Kon-

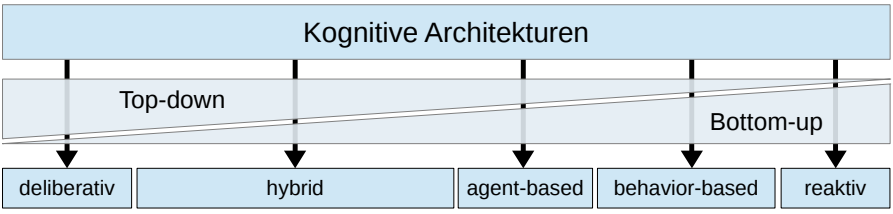



**Abbildung A 4.1:** Taxonomie kognitiver Architekturen

trollfluss eindeutig von oben nach unten abläuft. Module einer Ebene können zwar Daten austauschen, üben aber keine Kontrolle untereinander aus. Deliberative Architekturen konzentrieren alle Intelligenz in einem zentralen Modul in dem eine Art Umweltmodell aufgebaut wird, über dem die Inferenz nach zielführenden Aktionen stattfinden kann. Durch diesen Flaschenhals skalieren deliberative Architekturen schlecht und haben teilweise lange Reaktionszeiten bis zur Aktionsauswahl zum Nachteil.

### Reaktive Architektur

Um den Nachteilen zu begegnen wurde reaktive Architekturen vorgeschlagen, welche durch direkte Reaktionen auf Sensorwerte in den Modulen zu echtzeitfähigen Entscheidungen fähig sind. Diese Architekturen pflege im Allgemeinen kein zentrales Weltmodell und können dadurch robuster gegen Ausfall oder Fehlmessungen einzelner Sensoren sein, da die Gesamtheit aller Umweltinformationen über die reaktiven Einzelmodule verteilt ist. Die einzelnen Module implementieren dabei Basisverhalten, die allein durch die wahrgenommene Situation in der Umwelt aktiv werden und somit im Verbund ein emergentes Komplexverhalten ergeben. Die Fähigkeiten rein reaktiver Systeme sind leider begrenzt, da keine Planung über einer Gesamtsicht der Situation mehr stattfindet. Die von Brooks [Brooks, 1986] eingeführte Subsumption Architektur (engl. subsume, zusammenfassen) besteht aus reaktiven, asynchron arbeitenden Behavior-Modulen, die bereits in Layers mit unterschiedlichen Abstraktionsgraden unterteilt werden. Module der unteren Schichten realisieren Basisverhalten wie Hindernisvermeidung und Ausweichen, Module höherer Schichten realisieren Kombinationen dieser Basisverhalten, indem sie die Eingangs- und Ausgangssignale der Basisverhalten modulieren. Die einzelnen Module können dabei unabhängig voneinander entwickelt und getestet werden und sollen im Idealfall durch die übergeordneten Module in neue Aufgabenkontexte gebracht werden können.

Die Subsumption Architektur sieht auch vor, entgegen dem klassischen Ansatz keine zentrale Umweltrepräsentation aus allen Sensordaten aufzubauen. Vielmehr sollten lokal nur Teilmengen der Sensoren ausgewertet werden, um die entsprechenden Funktionen zu realisieren. Dies steigert nach Brooks auch

die Ausfallsicherheit und vermeidet das Datenfusionsproblem.
Hartley und Pipitone [Hartley und Pipitone, 1991] hatten im Rahmen einer Beispielimplementierung bereits einige Schwächen der Subsumption Architektur von Brooks identifiziert und motivieren die Notwendigkeit der direkten Kommunikation von Kontrollsignalen zwischen den Behaviormodulen. Ebenso zeigte sich, dass die ursprünglich angestrebte Modularität nicht gegeben ist. Bei Änderungen einzelner Basisbehaviors wird meist dennoch eine Anpassung aller übergeordneten Module und deren Vernetzung notwendig.
Dennoch wurde von Buttler et al. [Butler et al., 2001] die Subsumption Architektur mit softwaretechnischen Methodiken kombiniert um eine objektorientierte Implementierung dieser Steuerarchitektur zu realisieren.

### Hybride Architektur

In der weiteren Entwicklung wurden Konzepte ersonnen, welche sich der Vorzüge beider Paradigmen (deliberativ und reaktiv) bedienen und deren Nachteile beseitigen. Daraus resultieren mehrschichtige Architekturen mit meist zwei oder drei Ebenen, wobei oben langsamere planende und unten reaktive, echtzeitfähige Methoden eingesetzt werden.
Am bekanntesten sind dabei die **3T-Architekturen**, wobei T für engl. tier, Etage oder Schicht steht. Die drei Schichten sehen beispielsweise bei Bonasso et al. [Peter Bonasso et al., 1997] wie folgt aus:

- Die unterste Schicht bilden dynamisch verschaltbare „Skills“, welche von einem „Skill-Manager“ koordiniert werden. In dieser Schicht werden echtzeitkritische Entscheidungen getroffen.
- In der zweiten Schicht sorgt ein sogenannte „Sequencer“ dafür, dass die Skills in geeigneten Gruppen nacheinander aktiviert werden, um bestimmte Aufgaben (Tasks) zu realisieren.
- Die dritte Schicht beherbergt die planende Komponente, in der Systemziele, Ressourcen und Timing berücksichtigt werden, um Tasks in die richtige Reihenfolge zu bringen.

3T ist etwas weiter verbreitet als andere Architekturen, weil frühzeitig entsprechende Entwicklungswerkzeuge bereitgestellt wurden.
Eine weitere Variante der Kombination von reaktiven subsumption Ansätzen

und den Stärken der symbolischen Informationsverarbeitung der klassischen KI ist „Servo, Subsumption, Symbolic“ (SSS) [Connell, 1992]. Auch hier wird ermöglicht, dass auf den verschiedenen Ebenen die spezialisierten Methodiken ihre Stärken ausspielen können.

Bei Goeller [Göller et al., 2009, Göller, 2013], welcher eine Kontrollarchitektur für einen autonomen Einkaufswagen vorstellt. Heißen die drei Schichten „reaktive“, „taktische“ und „strategische“ Layer. Eine echtzeitfähige Reaktion wird durch seinen Ansatz auch bei komplexen Planungsaufgaben gewährleistet. In den unteren Schichten dieser Architektur kommen behavior-basierte Koordinationsmethoden zum Einsatz (siehe nächsten Abschnitt) um die geforderte Robustheit und Fehlertoleranz zu erzielen. Verschiedene Behavior-Module liefern einen kontinuierlichen Strom an Kontrollkommandos, wobei der Einfluss der einzelnen unabhängigen Module auf die Gesamtentscheidung dynamisch moduliert wird, um verschiedene Funktionen zu realisieren. Die Architektur sieht ein zentrales Modell für das Weltwissen vor, um allen Modulen Zugang zu diesem zu ermöglichen.

Die bisher betrachteten Architekturprinzipien beziehen sich meist auf die reine Bewegungssteuerung von Robotern. Sobald Interaktion und Dialog mit Menschen ins Aufgabenspektrum des Roboters rückt, kommt eine neue Dimension hinzu. Nun gibt es neben internen Zielen, die ein Roboter zu erfüllen hat, auch dynamisch vom Nutzer vorgegebene Aufgaben und die Notwendigkeit zur Konversation, welche die Fähigkeiten der bisherigen Architekturen überfordert. [Kleinehagenbrock, 2004] stellt in seiner Dissertation eine hybride drei Ebenen Architektur für einen Companion Roboter vor, der explizit für die Zusammenarbeit mit dem Menschen konzipiert wurde. Diese Architektur sieht auf Ebene der Skills auch eine Aufmerksamkeitssteuerung für Objekte und Personen vor. Die Koordination dieser nimmt auf der mittleren Ebene ein Execution-Supervisor wahr, der auf ein zentrales Szenenmodell zugreifen kann. Eine getrennte Komponente zur Dialogsteuerung findet sich in der deliberativen obersten Ebenen.

Mit dem Ziel, robotisches Verhalten für die Menschen in seinem Umfeld besser verständlich zu machen und dadurch die Kooperation zu verbessern, wurde von Stoytchev und Arkin [Stoytchev und Arkin, 2001] über der rationalen Ebene eine weitere emotionale Ebene eingeführt, mit welcher Motivationen

für die Ziele des robotischen Verhaltens modelliert werden können.
Duffy et al. [Duffy et al., 2005] stellen die sogenannte „Social-Robot-Architecture“ vor, welche ein Schichtenmodell umsetzt, dass sich in Physical-, Behavioral-, Deliberative- und Social-Level unterteilt. In dem Physical-Level werden hardwarespezifische Eigenheiten der Zielplattform abstrahiert. Darauf setzt dann im Behavioral- Level eine Menge vernetzter, parallel laufender Module auf, welche reaktives Verhalten gemäß der Sensordaten und Vorgaben von dem darüber liegenden Deliberative-Level realisieren. In diesem Deliberative-Level wird die Kontrolle in ein Multi-Agenten-System verteilt, wobei die einzelnen Softwareagenten der Belief-Desire-Intention-Theorie [Rao et al., 1995] folgen. Auf dem Social-Level werden letztlich zwischen Individuen Nachrichten ausgetauscht, was mittels einer Agent-Communication-Language (ACL) geschieht.
Es zeigt sich also, dass hybride Architekturen recht flexibel sind und auch Raum bieten, um weiter gefasste Aufgaben wie die Interaktion mit Menschen realisieren zu können.

#### Behavior-based Architekturen

Auch in der diametralen Denkweise der **Unified-Field-Theory**[1] wurden die reaktiven Architekturen weiterentwickelt und bilden einen eigenen Zweig der Behaviour-Based-Systems (BBS). Biologisch inspiriert ähneln sie den reaktiven Systemen im Sinne der Zerlegung aus Verhaltenssicht. BBS besteht aus einzelnen Behaviors die jeweils eine primitive Aufgabe erfüllen und zusammen „emergent behavior“ erzeugen. Ähnlich den Schichtenmodellen erlauben sie hierarchische oder zentrale Koordination und können auch im Gegensatz zu rein reaktiven Ansätzen Weltwissen lernen und modellieren. Dies wird allerdings verteilt gehalten und verarbeitet. Der Behavior-based Ansatz behandelt oft nur sehr roboternahe Aspekte wie die Navigationsleistungen. Für High-

[1] Dabei wird versucht das Gesamtverhalten eines Systems aus prinzipiell gleichen Funktionsmodulen zusammenzusetzen.

Level Aufgaben, welche oft mit Zustandsautomaten modellierbar sind, wurde von Armbrust et al. [Armbrust et al., 2012] daher auch eine Möglichkeit vorgeschlagen, Zustandsautomaten automatisch in Behaviornetze zu überführen. Trotz der implizit definierten Gesamtfunktion eines BBS soll aufgrund der Modularität eine unabhängige Entwicklung der Einzelteile möglich sein [Armbrust et al., 2012].

#### Agenten-basierte Architektur

Mit der Entwicklung hin zu multi-Roboter Anwendungen kam neben den bisher genannten Architekturen auch eine agenten-basierte Sicht auf. Die interagierenden, zusammenarbeitenden Akteure werden als Agenten bezeichnet. Diese sehr lose gekoppelten Entitäten wurden anschließend auch zur Beschreibung der inneren Abläufe in einem Robotersystem herangezogen. Die Agenten verfolgen dabei individuelle Ziele und können über das Verhalten der anderen Akteure im Gesamtsystem folgern und diese Erkenntnisse bei der eigenen Verhaltensauswahl berücksichtigen.

Knopp et al. [Knoop et al., 2004] stellen eine auf CORBA[2] basierende Kontrollarchitektur vor, in welcher die reaktiven Module als Agents bezeichnet werden und von einem Agent-Manager verwaltet werden. Diese Agents sind jeweils für eine spezielle Hardware zuständig. Die deliberativen Module werden als „Flexible-Programs“ bezeichnet und von einem zentralen Flexible-Program-Manager verwaltet. Alle Instanzen können auf ein zentrales Umweltmodell zugreifen. In diesem Ansatz steht das Lernen von Verhaltensweisen durch Vormachen im Vordergrund. Flexible-Programs können zur Laufzeit modifiziert und generiert werden.

[2]CORBA: Common Object Request Broker Architecture ist eine plattformübergreifende, objektorientierte Middleware, deren Kern ein so genannter Object Request Broker bildet.

Bei Kawamura et al. [Kawamura et al., 2000] wird sogar die Repräsentation des menschlichen Kommunikationspartners intern als „Human-Agent“ modelliert, mit dem der „Self-Agent“ des Roboters kommuniziert. Wie bei hybriden Architekturen werden bei Kawamura einzelne Softwaremodule in verschiedene Klassen eingeteilt.

- Hardware-Agents bedienen in Echtzeit die Sensoren und Aktoren im System,
- Behavior-/Skill-Agents implementieren Basisverhalten,
- Environment-Agents verkörpern eine Art Umweltmodell mit den Fähigkeiten der repräsentierten Objekte,
- Sequencer-Agents bilden eine Hierarchie sich gegenseitig aufrufender Agenten, welche durch Koordination der low-level Agenten bestimmte Tasks erfüllen,
- Multitype-Agents weichen die Struktur in gewissem Maße auf, um aus Effizientzgründen Funktionalitäten verschiedener Agentenklassen in einem Modul zu realisieren.

Die Agentensicht hat sich neben der Robotersteuerung auch in anderen Bereichen durchgesetzt. Speziell für Spoken-Dialog-Systems gibt es die Open-Agent-Architecture (OAA). Die einzelnen Agenten behandeln dabei jeweils einen Teildialog oder Teilbereich des Interessensgebietes des Systems und können in beliebigen Programmiersprachen erstellt werden, solange sie die von OAA definierten Schnittstellen implementieren. Agenten-basierte Dialogsysteme werden in A 6.2 noch einmal genauer charakterisiert.

# A 6

# Ergänzungen zum Kapitel Dialogsystem

## A 6.2 Entwicklungsgeschichte von Dialogsytemen

Kapitel 6 hat im Hauptteil bereits eine grobe Einordnung von Dialogsystemen im Bezug auf die in dieser Arbeit angestrebte Aufgabenstellung gegeben. An dieser Stelle sollen die historischen Entwicklungen im Bereich der sprachbasierten Dialogsystem und später der mutlimodalen Dialogsysteme für den Einsatz in der Robotik geschildert werden.

### A 6.2.1 Natürlichsprachliche Dialogsysteme (Historie)

Das Interesse an natürlicher Interaktion mit Computersystemen ist so alt wie die Geschichte der elektronischen Rechentechnik selbst. Ziel dabei ist es, auch Laien die Nutzung der Technik zu ermöglichen. Einen Durchbruch in das öffentliche Einsatzfeld erfuhren natürlichsprachliche Dialogsysteme allerdings erst mit dem Einsatz automatischer Auskunfts- und Buchungssysteme in den 1990er Jahren. Diese hatten hochgradig strukturierte Aufgaben und die Entwickler konnten ein sehr eingeschränktes Domänenwissen direkt umsetzen. **State-based Ansätze** (siehe Abb. A 6.1) lagen dabei einem Großteil der Umsetzungen zu Grunde und waren für die Realisierung eines funktio-

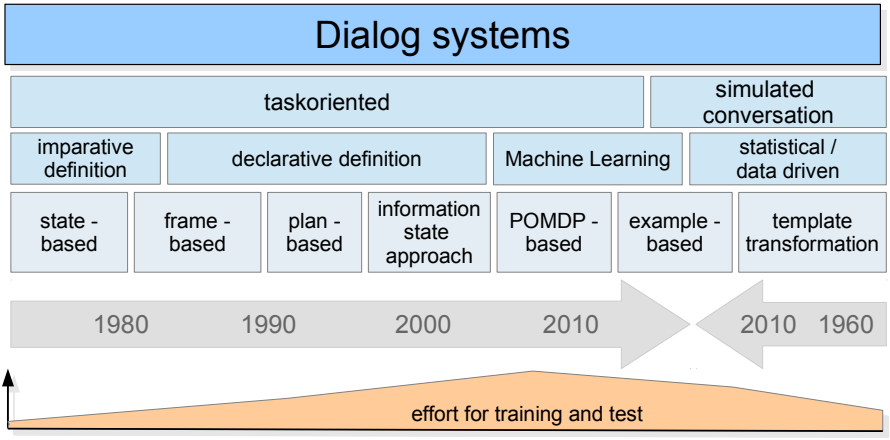



**Abbildung A 6.1:** Entwicklung der Dialogsystemansätze, Ausgehend von zwei Grundprinzipien taskorientierter Dialoge und der simulierten Konversation (Chat-Bot) wurden die Ansätze immer mächtiger und erfordern auch wachsenden Aufwand für die einsatzfähige Ausgestaltung. In neuster Zeit und in Zukunft werden beide Domänen mehr und mehr verschmelzen.

nalen Interfaces für uneingewiesene Nutzer im Sinne einer systemgeführten Abfrage ausreichend[1]. Das Grundprinzip dabei ist, dass in jedem Zustand eine spezifische Aktion (Frage nach Information, Bestätigung oder Ausgabe einer Antwort) ausgeführt und abhängig von der folgenden Nutzereingabe in einen nächsten diskreten Zustand übergegangen wird.

Bereits Bobrow et al. [Bobrow et al., 1977] entwickelten einen **frame-based Ansatz** zur Dialogmodellierung, um die Abläufe zu organisieren und die Konversation benutzerfreundlicher zu gestalten.

Der frame-based Ansatz gibt dabei ein Template für die benötigten Informationen vor und definiert Fragen oder Äußerungen, die an Bedingungen über den Informationen im Template geknüpft sind. Dadurch ist der Nutzer nicht mehr an eine bestimmte Abfolge bei der Übermittlung der relevanten Fakten gebunden, und der Dialogablauf ist gegenüber der state-based Varianten

[1] beispielsweise Telefonservices mit DTMF Steuerung (Mehrtonwahlverfahren über die Telefontastatur)

etwas flexibler. Wenn mehrere Aktionen zum aktuellen Zustand im Template passen, gibt es Prioritäten, die zur Auflösung des Konflikts herangezogen werden. Frame-based Systeme können deklarativ spezifiziert werden und sind nicht auf eine imperative Definition angewiesen wie finite state-based Systeme. Das erleichtert das Reagieren auf unvorhergesehene Verläufe und ist dadurch flexibler und in diesem Sinne einfacher für den Entwickler. Nachteilig könnte sein, dass durch die implizite Definition des Verhaltens, dieses bei komplexeren Systemen nicht mehr immer klar vorherbestimmt oder während des Betriebs nachvollzogen werden kann.

Um Dialogsysteme auch in Einsatzgebiete mit steigender Komplexität der Aufgaben einsetzen zu können, wurden die Dialogmodelle weiterentwickelt zu **plan-based-systems** Ein Beispiel hierfür ist das Collagen System [Rich et al., 2001]. In plan-basierten Systemen treten die Methoden der KI mehr und mehr in den Vordergrund und der Dialogmanager wird modularisiert im Sinne der Aufgabenstruktur. Den Kern bildet nun eine Agenda mit anstehenden Sprechakten, die von unabhängigen Softwareagenten manipuliert wird. Ebenso werden Nutzereingaben genutzt, um das Zustandsmodell zu aktualisieren und die Agenda anzupassen. Im Turn des Systems wird dann die Agenda abgearbeitet. Diese Herangehensweise ermöglicht eine noch freiere Gestaltung des Dialogablaufs und realisiert eine tatsächliche mixed-Initative, da beide Dialogpartner jederzeit vollkommen neue Aspekte auf die Agenda bringen können. Aufgrund der verteilten Softwarearchitektur und der jeweils spezifische Ziele verfolgenden Teilmodule (Agenten), wird bei diesen System auch von **agent-based dialog systems** gesprochen.

Mit diesen Entwicklungen entstand eine wachsende Diskrepanz zwischen den in Realweltanwendungen umgesetzten und den in der Forschung entwickelten immer anspruchsvolleren Methoden. Daher wurde im Jahr 2000 ein Software-Toolkit (TRINDI-Kit) für die Entwicklung von natürlichsprachlichen Dialogsystemen vorgestellt, welches aufgrund seiner Modularität das Entwickeln von Realweltanwendungen mit den althergebrachten Methoden ermöglicht, aber gleichzeitig den punktuellen Austausch von Algorithmen erlaubt, um mit in den Anwendungen gesammelten Daten auch wissenschaftlich weiterarbeiten zu können. Es handelt sich hierbei ebenfalls um einen KI basierten Ansatz der **Information-State-Approach** [Lemon et al., 2001] [Larsson und Traum,

2000] genannt wird. Die Grundidee dabei besteht darin, den aktuellen Dialogkontext in einer Datenstruktur (information state) zu repräsentieren und mittels einer Regelbasis (dialog move engine) über diesem Zustand eine Entscheidung für einen nächsten Dialogschritt zu erzeugen. Hierbei kann die dialog move engine einerseits generische domänenunabhängige Dialoglogik, und andererseits domänenspezifische Regeln enthalten die Aktionen des Systems definieren. Die Trennung von domänenunabhängiger Dialoglogik (Abfolge der Sprechakte) und der eigentlichen Inhalte des Dialogs sollte die einfacher Übertragung von Systemen auf neue Anwendungsgebiete ermöglichen. Verschiedene Module für Inputinterpretation, aber auch die Ausführung von Systemaktionen manipulieren den information state entsprechend und sorgen damit für das Voranschreiten im Dialog. Der Information-State-Approach zählt nicht zu den plan-based Systemen, da keine vorbestimmte Abarbeitungsreihenfolge der Dialogschritte in einer expliziten Agenda vorgegeben oder geplant wird. Die Generierung der Aktionen findet eher reaktiv statt.

Neben dem Information-State-Approach, wurden Anfang der 2000er noch weitere Ansätze entwickelt welche generische Dialogmodelle nutzen um domänenunabhängiger zu werden.

Beispielsweise wird von Bui und Rajman [Bui et al., 2004] ein generischer Prozess zur Erstellung von Dialogsystemen vorgeschlagen, welcher eine Zerlegung eines komplexen Tasks in generische Teildialoge zur Abfrage benötigter Informationen vorsieht. Weiterhin definieren Bui und Rajman einen Rapid-Prototyping-Prozess, welcher zielsicher zu einer praktikablen Lösung für eine Dialoganwendung führen soll. Dabei spielt auch das Erproben der Dialoge in einem Wizard-of-Oz Experiment eine Rolle.

In Ravenclaw [Bohus und Rudnicky, 2003], einem weiteren Beispiel, findet eine Trennung von domänenspezifischer Logik im Dialogmanager von domänenunabhängiger Steuerung von timing, turn-taking, grounding und allgemeinen Verhaltensweisen wie Hilfestellung, Wiederholung und Unterbrechungen statt. Zentrales Element hierbei ist eine domänenspezifische Taskbeschreibung in Form einer Hierarchie in einem Baum. Diese modulare Struktur stellt somit auch Wiederverwendbarkeit und Erweiterbarkeit sicher. Der Task-Baum wird mit Hilfe eines Stacks durchlaufen, auf dem jeweils die auszuführenden Aktionen abgelegt werden, welche entweder primitive Eingabe- oder Ausga-

beaktionen sein können oder weitere Makroaktionen (innere Knoten des Baumes), welche wiederum Aktionen auf den Stack legen. Dabei können verschiedene Strategien für die Auflösung der Planungsaufgabe für die Reihenfolge und Notwendigkeit der Bestandteile einer komplexen Aktion (Teildialog) angewendet werden. Leider ist das Übertragen eines solchen Systems auf eine neue Einsatzaufgabe noch immer recht aufwändig, da die Wissensbasis mit ihren hierarchischen Taskmodellen weiterhin händisch vom Entwickler designt und in aufwändigen Realwelttests angepasst und optimiert werden muss.

Mitte der 2000er wurden festgestellt, dass die Forschung im Bereich der Dialogmodellierung in ihren Beispielanwendungen in der Komplexität weit hinter den bis dahin vorhandenen realen Anwendungen zurück lag. Daher wurde 2007 Olympus als open-source Framework für die Erstellung von natürlichsprachlichen Dialogsystemen [Bohus et al., 2007] vorgestellt. Olympus wurde eingeführt, um die Lücke zwischen akademischen Dialogbeispielen und Realweltanwendungen zu überbrücken. Durch ein transparentes Softwaredesign sollten Unternehmen ermutigt werden, dieses System für ihre Anwendungen zu nutzen und damit neue Forschungsaspekte zu eröffnen, die erst aus dem Einsatz im großen Maßstab entstehen. Die Olympus Architektur folgt dem typischen modularen Aufbau eines Dialogsystems. Eine Komponente sorgt für die Interpretation der natürlichsprachlichen Eingaben, der zentrale Dialogmanager entscheidet über die nächsten Aktionen welche als semantische Einheiten an die Sprachgenerierung übergeben werden, wo sie in sinnvolle verbale Antworten des Systems umgewandelt werden. Der Dialogmanager in Olympus ist das eben beschriebene Ravenclaw [Bohus und Rudnicky, 2003] System.

Mit der Möglichkeit umfangreiche Datensätze von Dialogverläufen zu erfassen kam die Idee auf, die domänenspezifischen Teile eines Dialoges aus Beispieldaten zu lernen. Dies führte zur Entwicklung von Reinforcement-Learning Ansätzen welche mit Markov-Decision-Processes (MDP) oder Partially-Observable-Markov-Decision-Processes (POMDPs) arbeiten [Levin et al., 2000],[Williams und Young, 2007].

**POMDP-based dialog management** behandelt dabei auch einen wesentlichen Einflussfaktor auf die Ergonomie eines maschinell geführten Dialoges. Es findet dabei eine explizite Berücksichtigung von Unsicherheiten und Mehrdeu-

tigkeiten in der Interpretation von gesprochener Sprache statt. Die machine-learning-basierten Verfahren machten es möglich, auch auf der Planerebene mit Unsicherheiten umzugehen.
[Young et al., 2013] gibt einen fundierten Überblick zu den POMDP-basierten Dialogsystemen.
Die Einführung des POMDP-Ansatzes geschah vor allem in Verbindung mit einem Slot-filling Modell des Dialogs. Die benötigten Informationen, die durch den Nutzer für die Problemlösung spezifiziert werden müssen, werden demzufolge in Variablen, den so genannten Slots gespeichert. Hierbei werden nicht nur einzelne Werte sondern Wahrscheinlichkeitsverteilungen über den möglichen Werten genutzt, welche inkrementell anhand der beobachteten Spracheingaben aktualisiert werden. Daraus ergibt sich der Zustandsraum aus dem Kreuzprodukt aller möglichen Belegung aller Slots und ist demzufolge exponentiell in der Anzahl der Slots.
In jedem Dialogschritt besteht nun die Auswahl aus einer Menge aller möglichen Systemausgaben, welche im Anschluss bewertet wird. Ziel der POMDP-Modellierung ist es nun, diese Bewertungen für einen Dialogzyklus zu maximieren. Dadurch ist es zum Beispiel möglich, die Anzahl der Korrekturen für falsch verstandene Eingaben zu minimieren, indem das System Aktionen zum Grounding geschickt in den Verlauf des Dialoges einbaut. Für Telefonservices ist die Minimierung der Dialogdauer ein oft genutztes Optimierungskriterium.
Ein wesentlicher Nachteil der Machine-Learning Ansätze ist die Notwendigkeit, einen repräsentativen Trainingsdatensatz von Dialogverläufen zur Verfügung zu haben. Neben Wizard-of-Oz Versuchen, in denen der maschinelle Dialogpartner durch menschliche Akteure simuliert wird, kommen auch Simulationen zu Einsatz, in denen aus wenigen Interaktionsbeispielen simulierte Nutzer generiert werden, welche dann wiederum für das Training der eigentlichen Dialogmanager herangezogen werden. Nutzersimulation hat sich in ein eigenes Forschungsgebiet entwickelt. Schatzmann et al. [Schatzmann et al., 2006] geben dazu einen Überblick.
Ein weiterer Nachteil der POMDP-Modellierung liegt in der Komplexität der Aufgabe, eine Lösung für die optimale Strategie zu finden. Im Allgemeinen ist die direkte Lösung des zu einer einfachen Dialogaufgabe gehörigen POMDP nicht mehr beherrschbar, weshalb verschiedene Verfahren zur Approximation

der Problemstellung vorgeschlagen wurden.
Beispielsweise wird in [Kim et al., 2008] versucht die Anzahl der States zu beherrschen indem eine Hierarchie von teilausgefüllten Frames gebildet wird. Sicheres Wissen wird somit nicht mehr probabilistisch modelliert. Bei einem gegebenen Belief (Belegung der Wahrscheinlichkeiten in den Slots) braucht somit nur ein kleinerer Teil der möglichen Folgebeliefs durchsucht werden.
T. Paek [Paek, 2006] zeigt kritisch die Grenzen und Nachteile der Entwicklungsrichtung der POMDP-Dialogmanager auf. Neben der Komplexität nennt er ebenfalls den Kontrollverlust über den Dialogverlauf durch den Entwickler, sowie eine begrenzte Erweiterbarkeit einmal gelernter Systeme.
Mit **example-based dialog modeling** (EBDM) [Lee et al., 2009] wurde in jüngerer Vergangenheit die Grenze zwischen task-oriented dialog und simulated conversation verwischt. Diese neue Form der Dialogkontrolle basiert auf Pattern-Matching von Eingaben mit einer Datenbasis von Aufgezeichneten Dialogen. Im Gegensatz zu den Chat-Bot Systemen wird bei EBDM versucht die domänenspezifischen Inhalte vom allgemeinen Ablauf zu trennen, wodurch die benötigte Datenbasis nicht so groß sein muss. Dabei werden zusätzlich frame-artige Datenstrukturen genutzt, um den Dialogzustand zu repräsentieren.

### A 6.2.2 Dialogsysteme für interaktive Roboter

Mit Aufkommen der mobilen interaktiven Servicerobotik lag es nahe, die bekannten Ansätze aus der maschinellen Dialogführung direkt zu übertragen auf diese neue Einsatzsituation. Dies wurde auch mehr oder weniger erfolgreich gemacht, aber leider kamen mit der neuen Umgebung auch zusätzliche Anforderungen hinzu.
Daher stellen Peltason und Wrede fest:

> "Such approaches [spoken dialog systems from information seeking domain] are not directly transferable to the robotics domain which presents new challenges for dialog system engineering as it requires mixed-initiative, multi-modal, asynchronous, multi-tasking and open-ended interaction in dynamic environments" [Peltason und Wrede, 2010a]

Bereits 1999 nennen Asoh et al. folgende Hauptprobleme für den Einsatz von Dialogsystemen auf mobilen Robotern:

> "realization of robust speech recognition for noisy environments, realization of real-time responses with limited computational resources, and realization of flexible dialog control covering a wide variety of behaviors of the robots" [Asoh et al., 1999].

Davon sollte die Limitierung der Ressourcen heutzutage nicht mehr der entscheidende Faktor sein, allerdings stellt eine robuste Spracherkennung über mittlere Entfernungen von 1-3 m in nicht geräuschfreien Umgebungen noch immer eine ungelöste Aufgabe dar.
Neben der verbalen Kommunikation, wie sie von den etablierten Systemen bereits behandelt wird, müssen in der Robotik Domäne die Nutzer beobachtet werden während sie auch nicht-verbal mit System und Umwelt interagieren können. Der Einsatz von Dialogsystemen auf Robotern macht daher auch multimodale Eingabemöglichkeiten notwendig.
Außerdem kommt die Frage auf, ob das Dialogsystem die zentrale Entscheidungsinstanz sein kann, wenn noch viele andere autonome Aufgaben in einem mobilen Roboter erfüllt werden müssen. Die oben genannte direkte Übertragung der sprach-basierten Dialogsysteme würde zunächst einmal die zentrale Entscheidungsinstanz darstellen. Toptsis [Toptsis et al., 2004] dagegen sieht den Dialogmanager nur als Interface zum Nutzer, wobei die zentrale Kontrollinstanz eine andere ist. Dialogmanager und zentrale Ablaufsteuerung kommunizieren bei ihm bidirektional und bilden gleichwertige Partner bei der Entscheidungsfindung.
Auch in der in dieser Arbeit vorgeschlagenen Architektur bilden das Dialogsystem und die Behavior State Machine (siehe Kap. 5) ein sich ergänzendes Paar. Beide können von sich aus aktiv werden und bestimmen nur zusammen das Verhalten des Roboters.
Bei Toptsis et al. [Toptsis et al., 2004] wird für den Roboter Biron der Universität Bielefeld eine Kombination von Zustandsautomaten mit Dialog-Frames genutzt, welche die Informationen vom Nutzer und zusätzlich Systemzustände speichern. Der Zustandsautomat versucht dann alle Slots zu füllen, um in einen Zielzustand zu kommen, in dem ein gegebener Task erfüllt wird.

Generell wird die Behandlung der neuen Komponente Roboter durch Integration des Roboterzustands in die Dialogmodelle gehandhabt. Dadurch soll der Dialogmanager sich jederzeit über den generellen Roboterzustand "bewusst" sein und diesen bei seinen Entscheidungen berücksichtigen.
Peltason und Wrede [Peltason und Wrede, 2010b] stellen ein Konzept vor, welches explizit die Anforderungen aus der Robotik (Tasks) mit denen der Dialogführung vereint. Er definiert in seinem System PAMINI (Pattern-based mixed-initiative) Interaction Patterns, welche letztendlich auch jeweils einen Zustandsautomaten bilden. Dieser wird durch Nutzereingaben oder systeminterne Ereignisse im parallel arbeitenden Taskplaner geschaltet. Die Interaction Patterns werden als prototypische Abläufe von Dialog Acts identifiziert und mit einem Task des Systems assoziiert. PAMINI wird auch von einem der führenden RobotCup@home Teams (ToBI aus Bielefeld) eingesetzt [Wachsmuth et al., 2012]. Bei RobotCup@home lässt sich auch in der Aufgabenstellung eine gewisse Ähnlichkeit zur hier vorliegenden Zielapplikation feststellen.

## A 6.3 Was ist Grounding?

Grounding bezieht sich auf die Wissenssituation von Dialogpartnern. Der Begriff geht zurück auf den Artikel [Clark und Wilkes-Gibbs, 1986] in dem von "common ground" gesprochen wird, was soviel bedeutet, wie eine gemeinsame Wissensbasis, gemeinsame Annahmen bezüglich eines Faktes zu besitzen.
Kommunikation ist nicht immer eindeutig und die Inhalte nicht vollständig. Daher können zwei Dialogpartner einen gemeinsamen Wissensstand über einen Fakt erst nach mehreren Dialogakten erreichen. Es müssen einerseits die Inhalte kommuniziert werden und andererseits die Bestätigung des Empfängers über das von ihm Verstandene in die Gegenrichtung.
Beide Dialogpartner haben Grounding in Bezug auf einen Fakt, wenn sie beide den Fakt kennen und wenn beide wissen, dass der jeweils andere das weiß (inklusive des Wissens des Anderen über Wissen des ersten). Diese rekursive Definition führt zu verschiedenen Graden von "Groundedness" welche in einem Dialogsystem berücksichtigt werden können. Beispiele wären: unbekannt, angenommen, unbestätigt, bekannt, bestätigt oder missverstanden.
Während des Dialoges sollte daher ein fortwährender gegenseitiger Abgleich

von dem was der jeweilige Kommunikationspartner als Ziele und Inhalte der Konversation verstanden hat passieren.
[Clark und Brennan, 1991] geben Beispiele und charakterisieren näher, welche Schritte zum Erreichen des Zustands eines gemeinsamen Groundings möglich sind. Hier werden auch verschiedene Kosten für die Möglichkeiten eingeführt, wodurch sich für die Dialogsteuerung ein Optimierungsproblem ableiten lässt, welches die zu wählende Dialogstrategie (Schritte des Groundings) in Abhängigkeit vom Stand des gegenseitigen Wissens betrifft.

## A 6.6 Details zur konkreten Realisierung des Dialogsystems

### A 6.6.1 Dialog Modellierung

Im Abschn. 6.6 wurde bereits ein Überblick zur Modellierung des Dialoges im Rahmen des vorgeschlagenen Dialogsystems gegeben. In Ergänzung dazu folgen an dieser Stelle weitere Details zur Implementierung, welche ein tieferes Verständnis der Abläufe im System ermöglichen sollen. Abb. A 6.2 gibt dazu einen detaillierteren Überblick.

### A 6.6.2 Zustandsraum

Abschn. 6.6 hat bereits den Zustandsraum eines Dialog-Frames als Menge von Slots (mit Wert und Konfidenz) und Zählern für die bereits ausgeführten Aktionen eingeführt. Diese können die Nutzereingaben, deren Bestätigung (Grounding) und die History aufnehmen. In der Umsetzung sind dabei noch weitere Aspekte bedacht.
Da die Inputinterpretation und die restlichen Komponenten des Systems parallel arbeiten, können jederzeit neue Inhalte für die Slots entstehen. Slots bekommen in ihrer Definition daher Zusatzinformationen, unter welchen Umständen sie Daten aufnehmen können und wie das System auf diesen neuen Zustand reagieren soll. Slots können entweder nur während eines aktiven Dialoges oder immer empfänglich für neue Daten sein. Auch die gezielte Befüllung nach einer Frage des Systems ist möglich. Dadurch können z.B. mehrere Slots gleich lautende Antworten (Ja/Nein) aufnehmen, auch wenn sie für verschiedene Fragen stehen. Um solche Fragen nach Slots zu Beschreiben, werden den

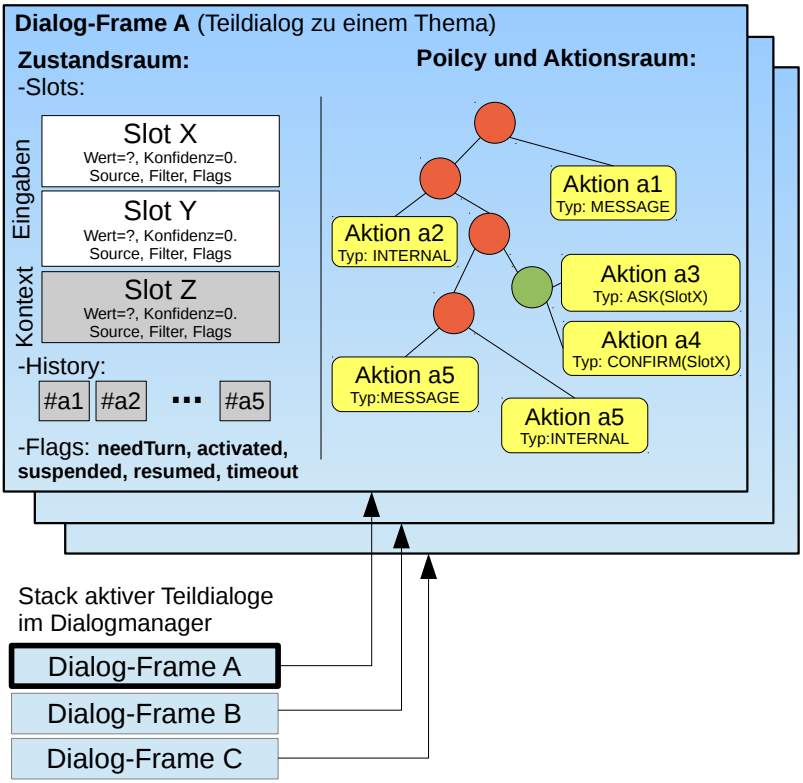



**Abbildung A 6.2:** Ergänzung zum Modell des Dialogs (vergleiche Abb. 6.4); Der Zustandsraum der Dialog-Frames enthält zusätzliche binäre Flags und die Aktionen sind mit einem Typ versehen, der das Verhalten des Dialogmanagers bestimmt.

zugehörigen Aktionen entsprechenden Angaben in Form von „Typen“ zugeordnet (siehe A 6.6.3). Nachdem ein Slot neue Nutzereingaben aufgenommen hat, muss auch die Reaktion des Systems darauf spezifiziert werden. Entweder kann ein neuer Turn getriggert werden, oder ein Dialog-Frame wird aktiv und kommt auf den Stack aktiver Dialoge. Diese Zusatzinformationen zu den Slots und ihrem Verhalten gehören nicht zu dem veränderlichen Zustand des Dialog-Frames, sondern sind Teil der statischen Konfiguration und werden in Slot-Flags gespeichert.

Neben den direkten Eingaben und Beobachtungen vom Nutzer können auch systeminterne Kontextinformationen in den Slots gespeichert werden und bestimmen somit das Verhalten mit. Beispielsweise kann der Emotionszustand des Roboters oder die Anwesenheit des Nutzers als Kontext Verwendung finden. Die Deklaration eines Slots bestimmt demnach zunächst eine Datenquelle (in Form eines Channels im MIRA-System). Für Nutzereingaben ist dabei im Allgemeinen der Inputinterpreter die Quelle (siehe Abb. 6.3 auf Seite 97), für Kontextinformationen können aber auch beliebige andere deterministischen Datenquellen, z.B. aus der Skill Layer, genutzt werden.

Damit Slots selektiv auf einzelne Ereignisse oder bestimmte Nutzereingaben gemacht werden können, wird jedem ein Filter in Form eines booleschen Ausdrucks zugeordnet. Dadurch kann ein Slot beispielsweise nur für Eingaben bestimmter semantischer Klassen empfänglich gemacht werden (siehe Abschn. 6.8).

Bei der Deklaration der Slots wird, wie bereits erwähnt, mit einer Reihe von Flags festgelegt, unter welchen Umständen Daten in den Slot aufgenommen werden können sowie, wann diese wieder gelöscht werden. Die möglichen Flags eines Slots sind die folgenden:

- EXPECTED_ALWAYS, EXPECTED_IF_ACTIVE, EXPECTED_IF_TOP: Erlaubt das Befüllen des Slots immer, wenn ein Frame aktiv ist, bzw. wenn der Frame auch im Stack ganz oben ist.
- EXPECTED_IF_ASKED: Erlaubt das Befüllen des Slots nur wenn im Dialog durch das System danach gefragt wird. Dadurch kann eine bestimmte Eingabe situationsabhängig verschiedenen Teildialogen oder Slots zugeordnet werden. Beispiel kann eine Ja/Nein Entscheidung sein, welche als Antwort auf verschiedene Fragen vorkommen kann.
- TRIGGER_TURN: Durch dieses Flag wird veranlasst, dass ein Systemturn ausgelöst wird, sobald sich der Slotinhalt ändert.
- ACTIVATES_FRAME, RAISES_FRAME: Diese Flags bewirken dass bei Änderung des Slotwerts der Frame aktiviert, bzw. auf die Spitze des Stacks aktiver Frames geschoben wird.
- CLEAR_ON_ACTIVATE, CLEAR_ON_COMPLETE: Definiert wann ein Slot geleert werden soll, entweder beim Start oder beim

Beenden eines Frames.

- READ_ON_ACTIVATE: Da Slots auch statische Kontextvariablen enthalten können, welche sich u.U. nicht regelmäßig ändern, kann durch dieses Flag veranlasst werden, dass diese Daten aktiv in den Slot des Frames übernommen werden, sobald der Teildialog aktiv wird.

Neben Slots und History gehören in der Praxis zum veränderlichen Zustandsvektor eines Dialog-Frames noch einige Zustandsbits (Flags), welche vom Dialogmanager gesetzt werden, wenn der Frame aktiviert, im Stack verschoben, oder wenn durch Nutzereingaben oder Zustandsänderung ein Systemturn erforderlich wird, in welchem eine systemseitige Aktion ausgeführt werden soll. Die Flags im Framezustand sind demnach diese:

- WANT_TURN: Zeigt an, dass durch Aktivierung oder Befüllung entsprechender Slots eine Systemaktion notwendig ist. Der Dialogmanager wird dadurch sobald wie möglich eine Systemaktion für diesen Frame ausführen.
- ACTIVATED: Zeigt an, dass der Frame gerade aktiviert wurde und noch keine Systemaktion für diesen Frame ausgeführt wurde.
- RAISED: Zeigt an, dass der Frame seit Ausführung der letzten Systemaktion dieses Frames im Stak ganz nach oben geschoben wurde.
- SUSPEND: Falls ein anderer Frame aktiviert wird und diesen verdrängt, so wird dieses Flag gesetzt und der unterbrochene Frame bekommt noch einen Turn, d.h. er kann noch eine Systemaktion ausführen, um auf die Unterbrechung zu reagieren.
- RESUMED: Nach einer Unterbrechung ist während der Auswahl einer adäquaten Systemaktion dieses Flag gesetzt. Dadurch kann ein Teildialog sinnvoll fortgesetzt werden, indem der Nutzer zunächst wieder in den entsprechenden Kontext gesetzt wird.
- TIMEDOUT: Falls eine Nutzerreaktion ausbleibt und eine Wartezeit abgelaufen ist, so wird noch einmal ein Turn fürs System aktiviert, in dem bei gesetztem TIMEDOUT Flag entsprechend reagiert werden kann (z.B. Fragen wiederholen oder den Dialog abbrechen).

Im Entscheidungsbaum der Aktionen des Dialog-Frames (siehe auch Abschn. 6.6) können in den Bedingungen der Knoten diese Flags mit berücksichtigt

werden. Es wird dadurch ermöglicht, dass beispielsweise nach einer Unterbrechung durch einen andern Teildialog der Nutzer zunächst wieder in den ursprünglichen Gesprächskontext gebracht wird. In diesem Fall wäre das RESUMED Flag gesetzt und der Roboter könnte eine Aktion auswählen, welche evtl. ausgibt: "Bevor wir nach dem Wetterbericht für morgen geschaut haben, waren wir gerade dabei einen Termin einzutragen..."
Ebenso kann mittels des SUSPEND Flags für den zu unterbrechenden Dialog noch ein sicherer Zustand hergestellt werden. Beispielsweise kann der Roboter gestoppt werden, wenn der Robotersteuerungsdialog unterbrochen wird.
Auch der Sonderfall, dass der Nutzer nicht mehr reagiert und es zu einem TIMEDOUT kommt kann abgefangen und evtl. mit einer internen Markierung als negativer Endzustand versehen werden, wie es in Abschn. 7.2.4 beschrieben ist.

### A 6.6.3 Aktionsraum

In Abschn. 6.6 wurde geschildert, wie die möglichen Aktionen und Ausgaben des Systems sowie eine Policy für deren Zuordnung zu verschiedenen Zuständen definiert werden. In Ergänzung dazu soll hier noch erwähnt werden, dass Aktionen angelehnt an die Sprech-Akt-Theorie (siehe Abschn. 6.1) einen Typ (Frage, Bestätigung oder keine Nutzerinteraktion) besitzen. Dieser wird bei der Definition der Aktionen mit spezifiziert und bestimmt, welches Verhalten des Systems nach der Abarbeitung des Turns ausführt. Dies entspricht in gewisser Weise dem domänenunabhängigen Diskurswissen aus Abschn. 6.1.
Folgende fünf **Aktionstypen** sind vorgesehen, welche teilweise auch in Kombination vorkommen können:

- INTERNAL: Dabei wird sofort nach der Abarbeitung wieder in den IDLE Turn-Zustand gewechselt (siehe Ablaufdiagramm Abb. 6.5 auf Seite 102), sodass keine Wartezeit für den Nutzer entsteht und vom Dialogmanager gegebenenfalls sofort die nächste Aktion ausgewählt werden kann. Dieser Typ tritt meist beim Beenden von Teildialogen oder beim rekursiven Aufrufen von anderen Teildialogen auf.
- MESSAGE: Dieser Typ deutet an, dass dem Nutzer etwas mitgeteilt wurde. Da dieser Zeit braucht um die Nachricht aufzunehmen, wird

in den Zustand ACKNOWLEDGEMENT_EXPECTED übergegangen, woraufhin auf bestätigende Eingaben des Nutzers oder auf das Erreichen des Endes einer Wartezeit gewartet wird. Danach wechselt der Turn-Zustand wieder auf IDLE und der Dialog kann fortgesetzt werden.

- ASK(Slot): Dieser Typ indiziert, dass mittels dieser Aktion nach einer bestimmten Information gefragt wird. Daraufhin wird der Turn-Zustand auf INPUT_EXPECTED gesetzt und der entsprechende Slot als „erfragt“ markiert, damit dieser die Informationen aufnehmen kann. Durch diesen Machanismus ist es möglich semantische Werte wie z.B. „Ja“ und „Nein“ entsprechenden Teildialogen zuzuordnen, auch wenn diese mehrfach vorkommen. Eine Systemaktion kann auch mit mehreren ASK(slot) Einträgen verknüpft sein und somit gleich mehrere Informationen erfragen. Außerdem wird auch im INPUT_EXPECTED Zustand ein Timer gestartet, der nach Ablauf den Nutzerturn beendet, auch wenn keine Eingaben vorgenommen werden. Dadurch bleibt das System nicht stecken und kann eventuell noch einmal gezielter nachfragen oder Hilfeleistungen bieten.
- CONFIRM(Slot): Diese ebenfalls in Kombination zu gebrauchende Markierung einer Aktion zeigt an, dass mit der Aussage im Text eine vom Nutzer gegebene Information bestätigt wird. Falls im nächsten Nutzerturn keine Korrektur dieses Fakts vorgenommen wird, kann der Dialogmanager davon ausgehen, dass die erkannte Eingabe korrekt war und die Sicherheit dieser Information erhöhen. Dies ist wichtig für den Prozess des Groundings und ermöglicht eine implizite Bestätigung von Fakten.
- EXPLICIT_CONFIRM(Slot): Dieser Typ markiert explizite Fragen des Systems ob eine bestimmte Information korrekt ist. Daraufhin wird ebenfalls der Turn-Zustand INPUT_EXPECTED gesetzt und neben den Eingaben für den spezifizierten Slot werden ebenfalls „Ja“, „Nein“ und „Bestätigung“ ausgewertet, um die Sicherheit des erfragten Fakts entsprechend zu verändern.

## A 6.7 Ablaufsteuerung, Datenfluss und Aktionstypen

In Ergänzung zu Abschn. 6.7 wird hier noch einmal ein Überblick zu den möglichen Turn-Zuständen im Dialogsystem gegeben. Abb. 6.5 auf Seite 102 visualisiert die prinzipiellen periodischen Abläufe im Dialog, bei denen zwischen den Turn-Zuständen ereignisabhängig gewechselt wird.

Mögliche Werte für den Turn-Zustand sind die folgenden:

- IDLE: Dieser Zustand zeigt an, dass das System gerade bereit ist die Initiative zu übernehmen und einen neuen Systemturn auszulösen, sofern der Dialogzustand (WANT_TURN Flag des obersten Dialog-Frames auf dem Stack) dies erfordert. Spontane Eingaben des Nutzers werden auch verarbeitet und aktualisieren den Dialogzustand.
- INPUT_EXPECTED: Dieser Wert liegt vor nachdem von der Outputgenerierung eine Frage an den Nutzer ausgegeben wurde und eine Interaktion erforderlich ist. Der Inputinterpreter wird in diesem Zustand besonders auf Nutzereingaben achten und die geschätzte Attention des Nutzers wird erhöht (siehe dazu Abschn. 6.9.3).
- ACKNOWLEDGEMENT_EXPECTED: Dieser in Abb. 6.5 nicht erwähnte Zustand verhält sich ähnlich zu dem vorherigen Wert, allerdings wird keine konkrete Eingabe sondern lediglich eine Reaktion zur Bestätigung der Kenntnisnahme einer Nachricht erwartet (nach Aktionen vom Typ MESSAGE).
- INPUT_IN_PROGRESS: Sobald vom Inputinterpreter erkannt wird, dass der Nutzer Eingaben vornimmt, wird dieser Zustand gesetzt bis die Eingabe Abgeschlossen ist. Dies verhindert, dass das System vorzeitig noch während einer Eingabe aktiv wird und unter Umständen auf eine unvollständige Eingabe falsch reagiert.
- MACHINE_TURN: Sobald der Dialogmanager entscheidet die Initiative zu übernehmen und eine Dialogaktion auszuwählen, setzt er diesen Turn-Zustand und verhindert dadurch eine weitere Aktionsauswahl, bis die Outputgenerierung den Zustand nach abgeschlossener Ausgabe zurücksetzt auf IDLE, INPUT_EXPECTED oder ACKNOWLEDGEMENT_EXPECTED.

### A 6.7.1 Abläufe im Dialogmanager

In Abschn. 6.7 erfolgte die Darstellung der Verarbeitung von Nutzereingaben und eine grobe Erläuterung der Abläufe der eigentlichen Dialogführung im Dialogmanager. Hier sollen diese Abläufe noch etwas genauer beschrieben werden.

Der Dialogmanager hat zwei parallele Aufgaben zu erledigen. Erstens ist er verantwortlich den Dialogzustand zu verwalten, sprich die Belegung aller Slots und den Frame-Stack zu aktualisieren und zweitens sorgt er für das Turnmanagement und führt die Auswahl der auszuführenden Systemaktion aus. Außerdem bietet der Dialogmanager ein Interface zum direkten manipulieren der Dialog-Frames, wenn beispielsweise eine Service-App einen Frame aktivieren, oder bestimmte Slotwerte abfragen möchte.

#### Tracking des Dialogzustandes

Wie bereits erwähnt, besteht der Gesamtzustand des Dialogs aus den Zuständen der einzelnen Dialog-Frames, sowie des globalen Stacks, auf dem aktive Teildialoge vorgehalten werden (siehe Abschn. A 6.6.1). Der Dialogmanager überwacht alle Datenquelle für die Slots der Dialogkonfigurationen. Bei Änderung der Werte prüft der Dialogmanager zunächst, ob die Filterbedingung der Slots erfüllt ist, und ob die zugehörigen Slots bereit sind, Daten aufzunehmen. Dazu werden die Slot-Flags (siehe Abschnitt A 6.6.2) berücksichtigt und gegebenenfalls wird überprüft, ob eine der letzten Aktionen des Systems den Typ ASK(Slot) besaß und somit der entsprechende Slot durch das System aktiv erfragt wurde. Falls die Vorbedingungen zutreffen wird der Wert der Datenquelle (MIRA-Channel) im Slot abgelegt und die zugehörige Konfidenz angepasst. Bei semantischen Werten vom Inputinterpreter wird dazu die dort berechnete Wahrscheinlichkeit genutzt, bei deterministischen Datenquellen wird der Sicherheitswert hart auf bekannt (1,0) gesetzt. Falls ein Slot auf diese Weise gefüllt wurde, ist vom Dialogmanager ebenfalls zu prüfen, ob gemäß der Slot-Flags eine Aktivierung des Frames angestoßen werden muss, ob ein neuer Systemturn ausgelöst werden soll und ob der Frame im Stack aktiver Frames nach oben verschoben werden soll (siehe Abb. A 6.3).

#### Manipulation von Dialogframes

Neben der Aktualisierung der Slotwerte gehört zum Dialogzustand auch die Reihenfolge der Dialog-Frames auf dem Stack aktiver Frames. Frames können aktiviert, im Stack nach oben verschoben (raised), deaktiviert sowie unterbrochen und fortgesetzt werden. Unterbrechung und Fortsetzung werden behandelt, indem für den betroffenen Dialog-Frame ein zusätzlicher Turn angestoßen wird.
Damit innerhalb der Bearbeitung eines Systemturns (alle Module arbeiten ja parallel) der Stackzustand konsistent bleibt, ist für die Aktivierung und Deaktivierung von Frames eine Warteschlange für diese Befehle vorgesehen. Während eines Turns werden eingehende Aktivierungs- und Deaktivierungsbefehle, in diese Warteschlange eingereiht. Dabei können sich korrespondierende Befehle für den gleichen Frame überschreiben (ein deaktivate löscht ein activate usw.), sodass letztendlich der zuletzt eingegangene Befehl ausgeführt wird (siehe Abb. A 6.3). Abgearbeitet wird die Warteschlange wenn der Turn-Zustand IDLE ist.
Der Dialogmanager besitzt einen eigenen Thread, welcher periodisch den Turn-Zustand überprüft und gegebenenfalls Die Abarbeitung der Befehlswarteschlange ausführt. Außerdem wird periodisch geprüft, ob ein neuer Systemturn ausgelöst werden muss, wobei die Aktionsauswahl gemäß dem Entscheidungsbaum im obersten Dialog-Frame auf dem Stack vorgenommen wird. Abbildung A 6.3 zeigt die Vorgänge noch einmal als Ablaufplan.

### A 6.7.2 Aktionsauswahl und Grounding

Am Start eines neuen Systemturns steht nach Abbildung A 6.3 die Auswahl der auszuführenden Systemaktion. Dies geschieht unter Berücksichtigung des aktuellen Systemzustands anhand der, in der Dialogkonfiguration des obersten Frames vorgesehenen Policy. Mittels eines Baumdurchlaufs nach dem rekursiven Algorithmus A 6.1 Baumtraversion wird dabei die Menge der sinnvollen Aktionen für die aktuelle Situation ermittelt.
Aus der resultierenden Aktionsmenge ist nun eine spezifische auszuwählen. Dies wird von dem im Anschluss geschilderten Dialogagenten ausgeführt. Entsprechend der ausgewählten Aktion setzt der Dialogmanager den neuen

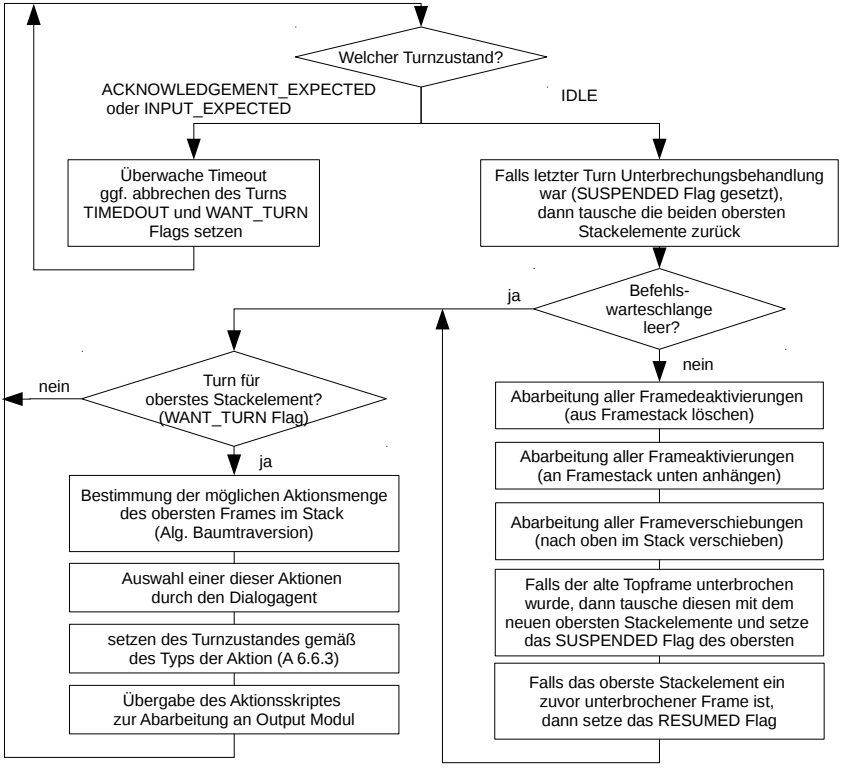



**Abbildung A 6.3:** Zyklischer Ablauf im Dialogmanger

Turn-Zustand. Ausschlaggebend hierbei ist der Typ der Aktion. Falls dieser ein ASK(Slot) enthält, so ist zu erwarten, dass der Nutzer nach einer bestimmten Information gefragt wird und der Turn-Zustand wechselt zu INPUT_EXPECTED. Im Fall einer Nachricht an den Nutzer, beispielsweise eine Antwort auf eine Frage oder eine einfache Bestätigung, welche durch den Aktionstyp MESSAGE gekennzeichnet ist, wird der Turn-Zustand auf ACKNOWLEDGEMENT_EXPECTED gesetzt. Den letzten Fall bilden Aktionen des Typs INTERNAL, welche keine Nutzerinteraktion vorsehen und daher auch keinen Nutzerturn auslösen. Bei diesen wechselt der Turn-Zustand direkt zurück auf IDLE, damit kann sofort die nächste Aktion ausgelöst werden. Bei Ausführung von Aktionen des Typs ASK(Slot) markiert der Dialogmanager zusätzlich die entsprechenden Slots als erwartet, und falls diese die Eigenschaft EXPECTED_IF_ASKED besitzen können sie im nächsten Nutzerturn Informationen aufnehmen. Die ausgeführte Aktion kann auch den Typ CONFIRM(Slot) oder EXPLICIT_CONFIRM(Slot) aufweisen. In diesen Fällen wird für die Verarbeitung von neuen Nutzereingaben die Interpretation der „Ja“ und „Nein“ Semantik variiert. Bei EXPLICIT_CONFIRM(Slot) Aktionen sollte der Nutzer explizit gefragt werden, ob eine bestimmte Information in einem Slot korrekt ist. Eine zustimmende Antwort darauf führt zu einer Erhöhung der Konfidenz des entsprechenden Slots. Ebenso führt eine Verneinung zu einer Reduktion der Konfidenz (das entspricht dem Reset dieses Slots) und damit zu einer erneuten Nachfrage der benötigten Information. Den Typ CONFIRM(Slot) können Aktionen als Zusatzattribut bekommen, wenn neben ihrer eigentlichen Aussage oder Frage die Information im genannten Slot mit der Systemaktion implizit bestätigt wird. Dies ist wichtig für ein effizientes Grounding. Ein Beispiel könnte in einem Dialog zum Erstellen eines Kalendereintrags vorkommen. Der Nutzer hat hierbei ein Datum zu wählen und wird anschließend gefragt, um welche Uhrzeit ein Termin eingetragen werden soll. Eine Aktion des Systems vom Typ ASK(Time) würde in der Ausgabe der Frage ,,`Wann soll der Termin eingetragen werden?`‘‘ lauten. Bei unsicher erkanntem Datum könnte die Aktion vom Typ ASK(Time), CONFIRM(Date) lauten: ,,`Für welche Uhrzeit soll ich am 24.12.2015 den Termin eintragen?`‘‘ Damit bekommt der Nutzer ein Feedback welches Datum verstanden wurde und hat die Möglichkeit dieses zu korrigieren.

Nach Aktionen mit Typ CONFIRM(Slot) wird der Dialogmanager beim erfolgreichen Füllen weiterer Slots die Sicherheit des referenzierten Slots weiter erhöhen. Die Sicherheit der Slotwerte in der Zustandsrepräsentation widerspiegelt somit auch den Grad des erfolgten Groundings bezüglich dieser Fakten. In der Policy des Dialog-Frames kann daher anhand der Sicherheit verschiedener Slotinhalte entschieden werden, wie vom System weiter zu verfahren ist. Dies kann entweder ein Nachfragen der Fakten, ein implizit oder explizit bestätigen lassen der Fakten oder das Fortfahren mit den bisherigen Informationen bedeuten.

**Algorithmus A 6.1:** Baumtraversion ( N )

**Eingaben**
`N` // Wurzelknoten des betrachteten Entscheidungsbaums
**Initialisierung**
$\hat{A} \leftarrow \varnothing$ // erlaubte Aktionsmenge
**Algorithmus**
```
IF N ist Optionsknoten, THEN
    FOR alle Kindknoten k von N DO
        Â = Â ∪ Baumtraversion(k)   // Vereinigung aller optionalen Zweige
IF N ist Entscheidungsknoten, THEN
    IF Bedingung in N erfüllt, THEN          // Abstieg in die Zweige
        Â = Baumtraversion( True-Teilbaum )
    ELSE
        Â = Baumtraversion( False-Teilbaum )
IF N ist Aktionsknoten, THEN
    Â = {Aktion a_N}
```
**Rückgabe**
$\hat{A}$

### A 6.7.3 Dialogagent

Um flexibel im Einsatz der konkreten Dialogsteuerungsstrategie zu bleiben wurde die Kernfunktionalität der Auswahl von Aktionen in das separate Softwaremodul Dialogagent gekapselt. Der entworfene Dialogmanager instantiiert gemäß seiner Konfiguration einen bestimmten Dialogagent und wird dadurch in seinem für den Nutzer des Systems wahrnehmbaren Verhalten definiert. Bislang wurden zwei Varianten des Dialogagenten realisiert, wobei eine triviale Lösung der Auswahlaufgabe im **Random-Agent** implementiert ist. Dieser

wählt aus der übergebenen Menge der zulässigen Aktionen $\hat{A}$ rein zufällig aus und kommt auch bei der Nutzung vollständig deterministischer Dialog zum Einsatz, da dabei die Auswahl aus einer einelementigen Menge auch zufällig geschehen kann. Die für die Nutzertests implementierten Service-Apps des SERROGA Demonstrators (siehe Kap. 9) nutzt diesen Random-Agent. Der zweite **Learning-Agent** genannte Dialogagent dient zur Umsetzung des im Kapitel 7 vorgestellten online lernenden Auswahlverfahrens, welches in der Lage ist das Dialogverhalten gemäß den Überlegungen aus Kapitel 3 an die Nutzervorlieben anzupassen und die Abläufe bezüglich eines systeminternen und auch durch den Nutzer bestimmten Rewardsignals zu optimieren.
Neben der jeweiligen Aktionsmenge benötigt der Dialogagent für diese komplexeren Strategien noch weitere Zustandsinformationen. Dafür wird er informiert, wenn Dialog-Frames aktiviert oder beendet werden, sowie wenn sich deren Reihenfolge auf dem Stack verändert. Das Beenden eines Frames ist dabei für die interne Bewertung mit einem Erfolgswert verbunden. Dieser kann entweder SUCCESSFULL, CANCELED_BY_USER oder CANCELED_BY_SYSTEM sein und dient zur Verteilung der Bewertung in Ketten sich gegenseitig aktivierender Teildialoge.
Für die Details zur Umsetzung des Learning-agents wird auf Anhang A 7 verwiesen.

## A 6.8 Ontologie

Die in Abschn. 6.8 eingeführten semantischen Klassen und semantischen Werte sollen an dieser Stelle noch einmal formal bezeichnet werden, um im späteren Verlauf die probabilistische Inputfusion mathematisch beschreiben zu können. Semantische Klassen seien demnach mit $K_j$ bezeichnet und geben einer Menge $R(K_j)$ (Range) von semantischen Werten eine eindeutige Bedeutung. Semantische Werte allgemein, beispielsweise Zahlen oder der Bezeichner "Küche", können in mehreren Klassen vorkommen und somit jeweils eine kontextabhängige Bedeutung besitzen. "Küche" in der Klasse "Zielposition" meint etwas anderes als "Küche" in der Klasse "Nutzerposition", welche den gegenwärtigen Aufenthaltsort des Nutzers angibt und evtl. bei einer Nutzersuche relevant ist.

Mit $v_i \in R(K_j)$ werden die möglichen semantischen Werte der Klasse $K_j$ bezeichnet. Eine semantische Klasse kann dabei einen der folgenden Typen besitzen:

- EVENT: In diesem Fall umfasst der Wertebereich der Klasse lediglich ein Element. Dieses bezeichnet ein eindeutiges Event wie beispielsweise das Kommando “Stop”.
- VALUE: Auswahl mehrerer diskreter semantischer Werte, welche im Sinne einer Entscheidung zu einer Bedeutungsklasse gehören. Beispiele hierfür sind die Klasse ”Zielposition”, welche die Werte ”Küche”, ”Wohnzimmer”, usw. besitzt.
- INT, FLOAT: Eine Klasse dieser Typen bezeichnet die Bedeutung einer Ganz- oder Gleitkommazahl. Beispiele wären die Anzahl von Terminen oder die Geschwindigkeit, die der Roboter fahren darf. In diesem Fall ist die Menge der Werte gegebenenfalls nicht endlich.
- TIME, DATE: Die Werte dieser Klassen sind Zeit oder Datumsangaben. Relevant sind diese beispielsweise bei Anfangs- und Endzeit für einen Termin im Kontext eines Kalenderservice.

Im Gegensatz zu richtigen Ontologien ist für die Realisierung von einfachen Dialogen keine Vererbung unter den Klassen vorgesehen. Eine semantische Entität im Dialogsystem wird immer als Kombination von Klasse und Wert beschrieben.

## A 6.9 Probabilistische Inputfusion

Die in Abschn. 6.9.2 beschriebene Inputverarbeitung mit Interpretation der Eingabeereignisse und anschließender Fusion wird in diesem Abschnitt noch einmal mit seiner mathematischen Umsetzung beschrieben.
Der erste Verarbeitungsschritt der Nutzereingaben ist das Übersetzen von multimodalen Eingabeereignissen in semantische Werte. Die Rohdaten oder Eingabeereignisse werden mit $e_j$ bezeichnet und stellen je nach Eingabekanal entweder eine erkannte Phrase der Spracherkennung, einen Klick auf einen GUI-Button oder eine erkannte Gestenklasse dar. Bei unsicheren Kanälen können auch Verteilungen $p(E)$ über der Zufallsvariablen $E$ (Eingaben) mit dem Wertebereich $\{e_j\}$ als Beobachtungen auftreten.

In den Inputfiltern (siehe Abschn. 6.9.2) wird für die in der Ontologie definierten semantischen Werte $v_i$ die Wahrscheinlichkeit für ihr Vorhandensein in der Eingabe abgeleitet. Dazu kann diese Abbildung deterministisch über feste Regeln (wie in der implementierten semantischen Analyse der Spracheingaben), oder über gelernte Wahrscheinlichkeiten $p(V_i|E)$ für alle möglichen $V_i$ geschehen. Dabei ist $V_i$ das binäre Eingabeereignis „Nutzer kommuniziert semantischen Wert $v_i$“. Das Modell der Bedeutung verschiedener Eingabeereignisse $p(V_i|E)$ kann nutzerspezifisch unterschiedlich ausfallen und anhand der Fusionsergebnisse und Rückmeldungen vom Dialogmanager gelernt werden.

Die Berechnung der Wahrscheinlichkeit $p_{Obs}(V_i)$, welche an den Inputinterpreter für die Fusion übermittelt werden, geschieht durch Multiplikation von $p(V_i|E)$ mit der beobachteten Verteilung über die Eingabeereignisse $p(E)$ und entsprechender Marginalisierung nach $V_i$ (siehe Gl. A 6.1)

$$p_{Obs}(V_i) = \sum_E p(V_i|E)p(E) \tag{A 6.1}$$

Die $p_{Obs}(V_i)$ geben dabei die Wahrscheinlichkeit dafür an, dass in dem Eingabekanal mit der letzten Beobachtung der semantische Wert $v_i$ vom Nutzer kommuniziert wurde. Eine Nachricht der Inputfilter an den Inputinterpreter über das Auftreten eines Ereignisses besteht somit aus einer Liste aus Paaren von semantischem Wert und einer Wahrscheinlichkeit aus $[0; 1]$, wobei $0,5$ bedeutet, dass die Eingabe zu dem speziellen semantischen Wert keine Information enthalten hat. Werte $< 0,5$ signalisieren, dass bestimmte Eingaben nicht gemeint waren, z.B. wenn ein „Ja“ erkannt wurde ist dies ein Indiz dafür, dass „Nein“ nicht gemeint wurde. Werte $> 0,5$ deuten auf ein Vorhandensein der Semantik in den Eingaben hin. Da die Menge der semantischen Werte sehr groß ist, bzw. bei Zahlen und Zeiten sogar unendlich groß ist, werden nur Werte kommuniziert und gespeichert, welche von $0,5$ verschiedene Wahrscheinlichkeiten aufweisen.

Der zweite Schritt der Verarbeitung besteht in der Akkumulation der Beobachtungen der verschiedenen Modalitäten im Inputinterpreter. Dies ist die eigentliche Fusion. Dazu wird der Eingabezustand, sprich das Vorhandensein bestimmter semantischer Werte in den Eingaben mit jeweils einem Bayes-

Filter[2] geschätzt.
$P(V_i)$ stellt dabei den Belief für den semantischen Wert $v_i$ dar (aktuelle Zustandsschätzung). Der Bayes-Filter besteht aus zwei Teilen, in denen zunächst ein Zustandsübergangsmodell angewendet wird, welches für ein zeit-basiertes Vergessen sorgt (ältere Eingaben sollen von jüngeren überschrieben werden können). Dazu bestimmt sich der prädizierte Belief zu:

$$\hat{P}(V_i) = 0,5 + 2^{-\frac{\Delta t}{\tau}}(P(V_i) - 0,5) \tag{A 6.2}$$

Hierbei stellt $\tau$ die Vergessensrate dar, konkret ist dies die Zeitspanne, innerhalb derer sich die Abweichung der Wahrscheinlichkeit von 0,5 halbiert. $\Delta t$ entspricht der Zeitdauer seit der letzten Beobachtung. Der prädizierte Belief wird anschließend mit der neuen Beobachtung multipliziert und ergibt so den neuen Belief. Da binäre Zustände geschätzt werden ergibt sich die Berechnung zu:

$$P(V_i) = \frac{\hat{P}(V_i)\ P_{Obs}(V_i)}{\hat{P}(V_i)\ P_{Obs}(V_i) + (1 - \hat{P}(V_i))\ (1 - P_{Obs}(V_i))} \tag{A 6.3}$$

Damit der Aufwand bei sehr vielen möglichen semantischen Werten überschaubar bleibt, wurde eine spärliche Repräsentation des Gesamtbeliefs implementiert. In dieser Liste wird für jeden beobachteten semantischen Wert ein Bayes-Filter angelegt und sobald der Belief $P(V_i)$ sich vom Wert $0,5$ nicht mehr signifikant unterscheidet auch wieder entfernt. Bis ein Eingabezyklus beendet ist, wird auf diese Weise die Fusion aller Beobachtungen vorgenommen. Sobald das Ende der Eingabe erkannt wurde, beispielsweise durch eine Pause in der Spracherkennung oder nach einer GUI Eingabe, wird der veränderte Eingabezustand normalisiert[3] und an den Dialogmanager übermittelt.
Dazu umfasst der dritte Schritt anschließend die Berücksichtigung sich widersprechender semantischer Werte, wie bereits am „Ja/Nein“ Beispiel eingeführt. Selbst wenn in den Inputfiltern nicht immer zu jedem Wert einer semantischen Klasse alle alternativen Werte abgewichtet werden, kann durch eine Normierung über die bekannten Werte eine valide Wahrscheinlichkeits-

[2] optimaler Zustandsschätzer basierend auf dem Bayes Theorem
[3] Widersprüche über Werte einer semantischen Klasse werden beseitigt

verteilung erzeugt werden.
Für die Werte $v_1$ bis $v_N$ einer semantischen Klasse $K$ werden die geschätzten Auftretenswahrscheinlichkeiten $P(V_i)$ jeweils als unabhängige Beobachtungen betrachtet, die zur Verteilung über die möglichen semantischen Werte $p(K)$ der Klasse $K$ jeweils einen Wahrscheinlichkeitswert gemäß Gl. A 6.4 beitragen.

$$P(K = v_j) = \begin{cases} \frac{1-P(V_i)}{N-1}, falls\ i \neq j \\ P(V_i), falls\ i = j \end{cases} \tag{A 6.4}$$

Dabei wird angenommen, dass sich die Wahrscheinlichkeitsmasse der Gegenereignisse $\bar{V_i}$ auf die anderen Werte $v_j$ mit $i \neq j$ gleichmäßig verteilen. Das Produkt der $P(K = v_i)$ für alle i ergibt demnach die Verteilung der semantischen Werte der betrachteten Klasse im Inputzustand:

$$P_{norm}(V_j) \propto P(V_j) \prod_{i \in \{1,\ldots,N\}/j} \frac{1 - P(V_i)}{N - 1} \tag{A 6.5}$$

Diese Verteilung ist anschließend noch auf Summe 1 zu normieren. Danach kann das maximale Element gesucht und an den Dialogmanager übermittelt werden. Die Slotinhalte werden durch den Dialogmanager bei eintreffenden Beobachtungen hart überschrieben, falls a) der bereits im Slot gespeicherte Wert unterschiedlich zum neu beobachteten ist oder b) die Wahrscheinlichkeit der neuen Beobachtung größer ist als der bereits gespeicherte Konfidenzwert. Eine Verringerung der Konfidenzwerte in den Slots kann also nicht geschehen. Dadurch wird verhindert, dass innerhalb des Dialogmanagers gesammelte Bestätigungen nicht wieder gelöscht werden. Ein probabilistisches Update der Slotwerte wird nur über den Inputinterpreter und den dort persistenten Eingabezustand realisiert.

### A 6.9.1 Lernen der Bedeutung von Nutzereingaben

Das Konzept des Dialogsystems erlaubt es, im Inputinterpreter die Bedeutung von Nutzereingaben und -beobachtungen online während des Betriebs zu erlernen bzw. zu adaptieren. Dazu können die Interpretationsmodelle $p(V_i|E)$ (wie wahrscheinlich ist der semantische Wert $v_i$ gemeint bei Eingabeereignis

$E = e$) angepasst werden, sobald vom Dialogmanager ein Bestätigungssignal für eine bestimmte semantische Klasse generiert wird.
Im Rahmen dieser Arbeit konnte diese Möglichkeit leider nur theoretisch betrachtet werden. Für eine praktische Umsetzung fehlen die nötigen Inputmodalitäten (beispielsweise eine Kopfgestenerkennung).
Als Grundlage für eine Adaption würde ein Kurzzeitspeicher benötigt werden, welcher die unmittelbar zurückliegenden Eingabeereignisse $e_j$ vorhält. Nach eingehender Bestätigung für einen semantischen Wert $v_i$ könnte dann die Verteilung $P(V_i|E)$ mittels einer MAP Schätzung (Maximum-a-posteriori Schätzung) adaptiert werden. Eine weitere Möglichkeit zur Adaption der $P(V_i|E)$ bietet sich auch ohne die Bestätigungsrückmeldung vom Dialogmanager, indem innerhalb eines Turns, bei gleichzeitig eintreffenden Eingabeereignissen, durch die Fusion eine höhere Sicherheit für einen bestimmten Wert resultiert, als durch die einzelnen Modalitäten erreicht würde. Es wäre demnach möglich, die Wahrscheinlichkeit der Fusionsergebnisse nach der Normalisierung über die Werte einer semantischen Klasse zu nutzen, um die $P(V_i|E = e_j)$ der beitragenden Eingabeereignisse $e_j$ zu verändern. Hierbei kann sowohl eine Bestätigung, als auch eine Verschlechterung der Zuordnung eintreten.
Bei einer Implementierung dieser Möglichkeiten bleibt zu testen, ob ein solches System langzeitstabil bleibt. Um dies zu forcieren, sollte neben den adaptiven Modellen für unsichere Eingabekanäle auch immer ein oder mehrere sichere Kanäle wie die GUI-Eingaben beibehalten werden, für welche sich die Modelle nicht ändern können.

### A 6.9.3 Aufmerksamkeitssteuerung

In Abschn. 6.9.3 wurde motiviert, warum erkannte Eingaben nicht ungehindert zum Dialogmanager durchgeleitet werden sollten. Eine Reaktion des Systems macht nur Sinn, wenn auch sichergestellt ist, dass die Eingaben an den Roboter gerichtet sind. Der **Focus of Attention** des Nutzers wird daher systemintern geschätzt.
Für diese Schätzung wird ebenfalls das bewährte Bayes-Filter eingesetzt, wobei der zu trackende Zustand $P(A)$ ist. Dabei bedeutet $A$, „der Nutzer ist auf das System fokussiert“. Um das zeitliche Verhalten zu modellieren, wird

periodisch ein Prozessmodell angewendet, welches die Wahrscheinlichkeit gegen 0,5 also unbekannt zieht. Die aktualisierte Wahrscheinlichkeit $\hat{P}(A)$ ergibt sich nach:

$$\hat{P}(A) = 0,5 + 2^{-\frac{\Delta t}{\tau}} \left(P(A) - 0,5\right) \tag{A 6.6}$$

Hierbei ist $\Delta t$ die Zeitspanne seit dem letzten Update und $\tau$ ein Parameter, welcher die Abklinggeschwindigkeit bestimmt. Auch bei ausbleibenden Beobachtungen wird dieses Update durchgeführt, wenn der Attention-Wert abgefragt werden soll.
Um Beobachtungsupdates der Aufmerksamkeit zu machen, wird $P(A)$ gemäß dem Produkt zweier Verteilungen über einer binären Zufallsvariablen aktualisiert:

$$P(A) = \frac{\hat{P}(A)\ P(O)}{\hat{P}(A)\ P(O) + (1 - \hat{P}(A))\ (1 - P(O))} \tag{A 6.7}$$

Hierbei ist $P(O)$ die beobachtete Aufmerksamkeit, welche ebenfalls mit einer Unsicherheit behaftet sein kann und daher auch als Wahrscheinlichkeitsverteilung modelliert wird. Es kann sowohl Aufmerksamkeit ($P(O) > 0,5$), als auch Ablenkung ($P(O) < 0,5$) beobachtet werden. Konkret wird für jede erkannte und weiterverarbeitete Eingabe eine Beobachtung mit $P(O) > 0,5$ generiert, um den Aufmerksamkeitslevel während eines Dialoges aufrecht zu erhalten. Bevor jedoch eine Eingabe verarbeitet wird, muss entweder ein eindeutig an den Roboter gerichtetes Ereignis, wie beispielsweise bestimmte Triggerwörter, erkannt werden oder die geschätzte Aufmerksamkeit $P(A)$ ist bereits über einer modalitätsspezifischen Schwelle. Für Eingaben über das grafische Nutzerinterface ist diese Schwelle sehr gering ($< 0,5$) und kann damit auch zur Aktivierung der Aufmerksamkeit dienen.

# A 7

# Probabilistische Modellierung und Dialogplanung

Dieses Anhangkapitel führt die mathematischen Werkzeuge und Konzepte ein, welche für die Umsetzung eines planenden adaptiven Dialogs genutzt werden. Basis bildet hierbei eine probabilistische Sichtweise der Problemstellung. In den letzten Jahren hat sich gezeigt, dass insbesondere im Feld der mobilen Robotik eine Berücksichtigung von Zufall und Unsicherheiten in der Modellbildung als Basis für eine robuste Entscheidungsfindung und Inferenz nicht mehr wegzudenken sind [Thrun et al., 2005]. Insbesondere eine beschränkte Wahrnehmungsfähigkeit von technischen Systemen führt zu unvollständigen Informationen, die einem System dennoch als Entscheidungsgrundlage ausreichen müssen. In diesem Sinne stellt auch die Dialogdomäne keine Ausnahme dar. Einerseits werden die Zustandsinformationen zu Intentionen und Absichten der Dialogpartner nur nach und nach kommuniziert, was zu unvollständigem Wissen während des Dialoges führt, und andererseits sind auch die kommunizierten Informationen aufgrund unsicherer Kanäle, wie Sprache oder Gestik, nicht frei von Fehlern.

## A 7.1 Grundlagen probabilistischer Modellierung

Ein Ansatz für die Behandlung von unvollständigem Wissen im System bietet dessen explizite Berücksichtigung in der Modellierung und Inferenz. Werden die Variablen im System durch Zufallsvariablen (Darstellung durch Großbuchstaben beispielsweise $A$ oder $S$) und deren Zusammenhänge durch Wahrscheinlichkeitsverteilungen beschrieben, so nennt man dies ein **probabilistisches Modell**. Wahrscheinlichkeitsdichtefunktionen sollen hier mittels $p(A, S)$ bezeichnet sein und die Wahrscheinlichkeit eines konkreten Ereignisses als $P(A = a_n, S = s_m)$. Mittels eines solchen Modells können anschließend Schlussfolgerungen gezogen werden. Beispielsweise können beobachtete Variablen in das Modell eingesetzt und nach resultierenden Verteilungen der abhängigen Variablen gefragt werden ($p(S|A = a_n)$ zum Beispiel). Dieser Prozess nennt sich Inferenz. Die Grundlagen der probabilistischen Mathematik können in der einschlägigen Literatur [Bishop et al., 2006] nachgeschlagen werden. An diesem Stelle werden lediglich die unmittelbar benötigten Algorithmen eingeführt.

Grundlage für die Inferenz mittels eines probabilistischen Modells bildet die Regel der bedingten Wahrscheinlichkeit:

$$P(A|B) = \frac{P(A, B)}{P(B)} \tag{A 7.1}$$

Und daraus abgeleitet die Bayes-Regel für die Umkehrung einer bedingten Wahrscheinlichkeit:

$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)} \tag{A 7.2}$$

Für ein gegebenes probabilistisches Modell und eine Teilmenge bekannter Modellgrößen kann mit Hilfe dieser Regeln die Verteilung so umgeformt werden, dass alle bekannten Größen auf der Bedingungsseite und die unbekannten auf der Seite der abhängigen Variablen stehen ($p(abhaengig|beobachtet)$). Dadurch erhält man eine Verteilung über die unbekannten Größen, welche dem Wissen im Modell und allen bekannten/beobachteten Größen entspricht und somit als Grundlage für eine optimale Entscheidung dienen kann.

## A 7.2 Faktorgraphen

Die probabilistische Modellierung von komplexen Systemen führt schnell zu ebenso komplexen und damit kaum noch effizient berechenbaren probabilistischen Modellen. Mit jeder zusätzlichen Zufallsvariablen im System (z.B. auch durch Hinzunahme eines neuen Zeitschrittes) vergrößert sich der Definitionsbereich der benötigten Verteilung um einen Faktor, der dem Wertebereich der neuen Zufallsvariablen, entspricht. Glücklicherweise bestehen oftmals nicht zwischen allen Zufallsgrößen im System Abhängigkeiten, so dass sich Vereinfachungen für die Darstellung ergeben.

Zunächst lässt sich feststellen, dass ein beliebiges probabilistisches Modell $p(A, B|C)$ gemäß Gleichung A 7.1 als Produkt verschiedener Faktoren schreiben lässt. $p(A, B|C) = p(A|B, C)p(B|C)$ - hierbei spricht man auch von Faktorisierung einer Verteilung. Wenn die Verteilung der Größe $A$ nicht von der Belegung von $B$ abhängt, so heißt $A$ unter $C$ bedingt unabhängig von $B$ (kurz $A \perp B|C$) und es gilt: $p(A|B, C) = p(A|C)$

Dies zeigt schon am einfachsten Beispiel mit drei Variablen, dass für die Darstellung eines Modells nicht notwendigerweise ein dreidimensionaler Raum benötigt wird, sondern dass zwei zweidimensionale Verteilungsfunktionen ausreichen, sofern die entsprechende Unabhängigkeitsannahmen gelten.

$$p(A, B|C) = p(A|C)p(B|C) \ , falls \ A \perp B|C \tag{A 7.3}$$

Zur bildlichen Darstellung solcher Abhängigkeiten / Unabhängigkeiten wurden so genannte graphische Modelle [Jordan, 1998] eingeführt, wobei dies ein Oberbegriff für verschiedene Ausprägungen ist.

Abb. A 7.1 zeigt links die Notation des Beispiels als Bayessches Netz, wobei Abhängigkeiten zwischen den Variablen durch Pfeile dargestellt werden. Die Faktoren des Modells lassen sich hierbei indirekt ablesen, indem für jeden Knoten alle eingehenden Pfeile eine Bedingungsvariable im Faktor ergeben. Alternativ bietet sich die Darstellung als Faktorgraph (Abb. A 7.1 rechts) an, wobei die Faktoren direkt als Knoten des Graphen angegeben werden und Kanten zu allen Knoten existieren, welche die Variablen des Faktors repräsentieren. Der Faktorgraph ist demnach ein bipartiter Graph aus den Kno-

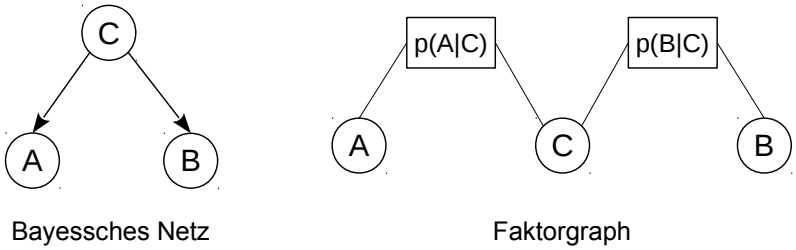



**Abbildung A 7.1:** Verschiedene graphische Modelle für ein probabilistisches Modell $p(A|C)p(B|C)$.

tenmengen für Faktoren (eckig) und Variablen (rund).
Faktorgraphen bieten sich an, anhand ihrer Struktur effiziente Inferenzalgorithmen einzuführen, wie beispielsweise den Sum-Product und den Max-Product Algorithmus [Frey et al., 1997, Kschischang et al., 2001, Murphy, 2002]. Hierbei wird die Unabhängigkeitsstruktur genutzt, um Zwischenergebnisse bzgl. ihrer Komplexität ebenfalls minimal zu halten.
In dynamischen Systemen kann ebenfalls eine graphische Modellierung vorgenommen werden. Dabei treten die Zufallsvariablen in Abhängigkeit von der Zeit mehrfach auf und werden mit diskreten Zeitindizes versehen $(A_t)$. Falls das Systemverhalten, sprich die Zusammenhänge der Zufallsgrößen zwischen Zeitpunkt $t$ und $t+1$, konstant über die Zeit sind, so kann man einen dynamischen Faktorgraph aufstellen, der lediglich eine Zeitscheibe mit den Variablen der Zeitpunkte $t$ und $t+1$ beschreibt. Dieser lässt sich anschließend für die Inferenz beliebig weit ausfalten. Diese Notation wurde in Abschn. 7.2.2 und Abb. 7.5 für die Beschreibung des Dialogverlaufs über der Zeit genutzt.

## A 7.3 Inferenz und Planung in Faktorgraphen

Wie bereits erwähnt werden die Faktorgraphen genutzt, um Algorithmen für Inferenz und Planung zu definieren. Da dabei lokale Operationen in den Graphknoten ausgeführt und anschließend die Zwischenergebnisse entlang der Kanten im Graph propagiert werden, spricht man von **Message-Passing-Algorithmen**. Eine Message $\mu_{X_i \to f_j}(X_i)$ ist dabei eine Funktion über einer Zufallsvariablen $X_i$ und wird zwischen den Knoten $X_i$ und

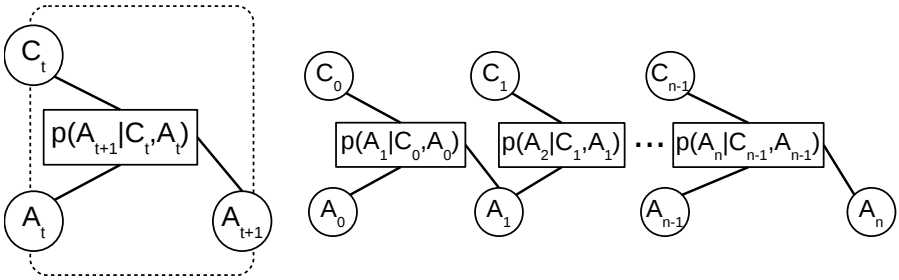



**Abbildung A 7.2:** links: dynamischer Faktorgraph, rechts: ausgefaltet für n Zeitschritte

$f_j = p(X_1, \ldots, X_l | X_{l+1}, \ldots, X_m)$ versandt. Abb. A 7.3 zeigt den Messageversand, welcher üblicherweise in zwei Phasen (Distribute und Collect Phase) geschieht, damit letztendlich in jedem Konten die Informationen von allen anderen berücksichtigt werden. Zu bemerken ist dabei, dass die Messages, welche über eine Kante aus einem Knoten heraus gehen, nicht von den eingehenden Messages aus dieser Kante abhängen.

### A 7.3.1 Sum-Product-Algorithmus

Konkret kann mittels des Sum-Product-Algorithmus bei einer gegebenen Menge von beobachteten Variablen die marginale Verteilung aller anderen Zufallsgrößen bestimmt werden. Hierzu sollte der Graph kreisfrei sein und somit eine Baumstruktur aufweisen (siehe Abb. A 7.3). Einen beliebigen Variablenknoten betrachtend (o.B.d.A. Wurzel des Baumes) werden gemäß des Message-Schemas (Collect- und Distribute-Phase) zunächst Messages von den Blättern zur Wurzel gesandt und anschließend von dieser zurück in die Blätter. Ein Knoten kann dazu eine Nachricht an einen adjazenten Knoten schicken, sobald er Nachrichten von allen anderen Nachbarn erhalten hat, was durch die zwei Phasen und die Baumstruktur gewährleistet ist. Die Nachrichten selbst berechnen sich, wie der Name Sum-Product schon sagt, wie folgt:

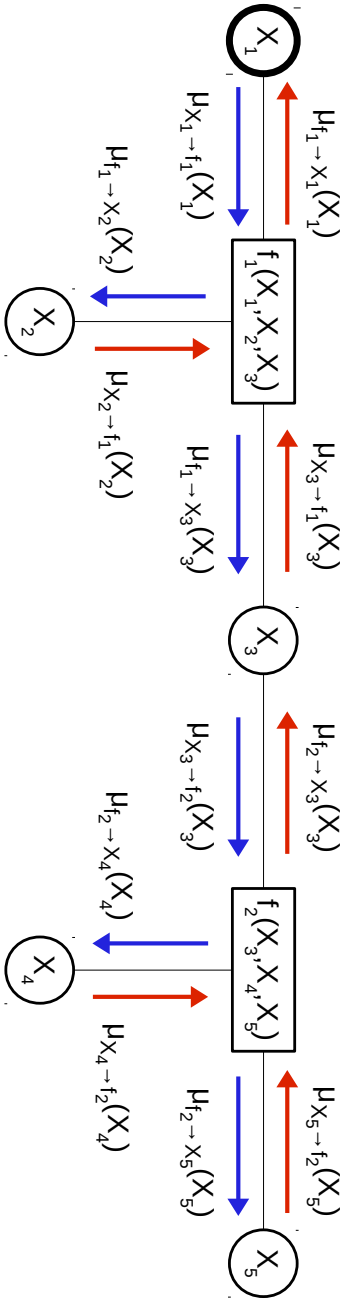



**Abbildung A 7.3:** Versand der Messages in einem Faktorgraphen, rot: Collect-Phase, alle Messages werden zum Knoten $X_1$ gesandt; blau: Distribute-Phase, Messages werden ausgehend von $X_1$ bis in alle Blätter des Baumes geschickt.

Messages von Variablenknoten $X_i$ zu Faktorknoten $f_j$ :

$$\mu_{X_i \to f_j}(X_i) = \prod_{l \in ne(X_i) \setminus f_j} \mu_{f_l \to X_i}(X_i) \tag{A 7.4}$$

wobei $ne(n)$ alle zu $n$ benachbarten Knoten im Graph bezeichnet. Im Variablenknoten werden demnach alle eingehenden Nachrichten multipliziert und an den Nachbarknoten geschickt.

Messages von Faktorknoten $f_i$ zu Variablenknoten $X_j$:

$$\mu_{f_i \to X_j}(X_j) = \sum_{X_1} \dots \sum_{X_m \neq X_j} f_i(X_1, \dots, X_m) \prod_{n \in ne(f_i) \setminus X_j} \mu_{X_n \to f_i}(X_n) \tag{A 7.5}$$

Im Faktorknoten werden ebenfalls alle eingehenden Messageverteilungen auf das Faktorpotential $f_i(X_1, \dots, X_m) = p(X_1, \dots, X_l | X_{l+1}, \dots, X_m)$ multipliziert und anschließend alle für den Zielknoten nicht relevanten Variablen durch Summation eliminiert.

Diese Vorgehensweise resultiert aus der Betrachtung der trivialen Inferenz, bei der zunächst alle Faktoren multipliziert und anschließend über alle Variablen marginalisiert (summiert) wird, welche gerade nicht betrachtet werden. Dabei fällt auf, dass einzelne Faktoren aus Teilen der Summen ausgeklammert werden können, was zu den lokalen Rechenvorschriften für die Knoten des Graphen führt. Eine anschauliche Herleitung ist in [Bishop et al., 2006] zu finden.

### A 7.3.2 Max-Product-Algorithmus

Neben der Bestimmung von Marginalverteilungen existiert auch ein Algorithmus zur Bestimmung der wahrscheinlichsten Belegung von Variablen, bzw. einer Belegung, welche die Wahrscheinlichkeit des Modells maximiert. Dieser kommt für die Planung der vielversprechendsten Aktion im Dialog zum Einsatz (Abschn. 7.4). Für ein Modell $p(X)$ findet der Max-Product-Algorithmus

die Belegung $x_{max}$ für $X$ mit

$$x_{max} = \underset{x}{\operatorname{argmax}}\, p(X = x) \tag{A 7.6}$$

Eine Ausprägung des Max-Product-Algorithmus im Rahmen von Hidden-Markow-Models (HMMs) ist unter dem Namen Viterbi-Algorithmus bekannt. Dieser versucht die wahrscheinlichste Abfolge von Zuständen im HMM bei gegebenen Beobachtungen zu finden.

Die Rechenvorschriften sind äquivalent zum Sum-Product-Algorithmus, mit der Ausnahme, dass im Faktorknoten die Summierung über die nicht benötigten Variablen durch eine Maximum-Operation entlang dieser Dimensionen ersetzt wird. Messages vom Faktorknoten $f_i$ zum Variablenknoten $X_j$ berechnen sich daher wie folgt:

$$\mu_{f_i \to X_j}(X_j) = \max_{X_1} \dots \max_{X_m \neq X_j} f_i(X_1, \dots, X_m) \prod_{n \in ne(f_i) \setminus X_j} \mu_{X_n \to f_i}(X_n) \tag{A 7.7}$$

Da durch die Maximum-Operation die Gesamtwahrscheinlichkeiten nicht wieder akkumuliert werden und dadurch numerische Probleme durch die sehr kleinen Zahlen entstehen können, bietet es sich an, die Berechnungen in der log-Domäne auszuführen. Die Produkte werden dabei zu Summen, wodurch sich die alternative Bezeichnung Max-Sum-Algorithmus ergibt, unter welcher er auch in [Bishop et al., 2006] zu finden ist. Wenn, wie in der Anwendung für diese Arbeit, lediglich die beste Belegung interessiert, nicht aber der maximale Wahrscheinlichkeitswert, so kann alternativ durch Normierung der Messages den numerischen Problemen begegnet werden.

### A 7.3.3 Inferenzframework

Im Rahmen der Entwicklungen in dieser Arbeit wurde in Zusammenarbeit mit Soeren Kalesse ein Inferenzframework [Kalesse, 2008][1] implementiert, wel-

[1] Diese Arbeit wurde vom Autor betreut.

ches grundlegende Algorithmen und Datenstrukturen für die angesprochene Modellierung als Softwarebibliothek bereitstellt. Von zentralem Interesse sind dabei die Repräsentationsformen von Wahrscheinlichkeitsverteilungen und die damit auszuführenden Basisoperationen, welche in den komplexeren Inferenzalgorithmen zur Anwendung kommen.
Die Verteilungsrepräsentationen hängen stark von der Natur und dem Wertebereich der Systemvariablen ab. Für diskrete Zustände mit einem endlichen Wertebereich kommt meist eine **diskrete Verteilungstabelle** zum Einsatz. Hierbei wird für jede mögliche Belegung der Zufallsvariablen der Wahrscheinlichkeitswert gespeichert, was mit entsprechenden Kosten für Speicherplatz und Verarbeitungszeit einhergeht. Eine effizientere Form der Modellierung stellen parametrische Beschreibungen der Verteilung dar (**parametrische Verteilung**). Hierbei soll die **Normalverteilung** (auch **Gaussverteilung**) explizit genannt werden, da diese häufig auch für die approximative Beschreibung von reellwertigen Zufallsgrößen zum Einsatz kommt. Für nichtnormalverteilte Größen können **Mixture-of-Gaussians** (MoG) genutzt werden, wobei die Form der Wahrscheinlichkeitsdichtefunktion als gewichtete Summe von lokalen Gaussverteilungen dargestellt wird. Der Aufwand für die Repräsentation und Verarbeitung steigt dabei in Abhängigkeit von der Approximationsgüte (Anzahl der Komponenten). In konsequenter Weiterverfolgung der Erhöhung der Komponentenanzahl resultiert als weitere Form der Repräsentation die **Partikel-Verteilung**. Hierbei wird lediglich eine Menge von Datenpunkten im Wertebereich der Zufallsvariablen als Beschreibung der Dichtefunktion genutzt. Eine komponentenweise Formbeschreibung wird dabei nicht mehr durchgeführt. Eine für alle Partikel geltende Kernelfunktion kann genutzt werden, um die inhärente Unsicherheit der Partikel zu berücksichtigen und die Dichtefunktion trotz der punktuellen Definition stetig zu machen. Dies führt letztendlich zur **Kernel-Density Schätzung**.

### A 7.3.4 Sample-Verteilung

Wie bereits erwähnt (Abschn. 7.4.2) wird für die Planung in der Dialogdomäne ein sehr hochdimensionaler Zustandsraum vorkommen, weshalb für die Repräsentation von Verteilungen über diesem Zustandsvektor, aber auch für die

Darstellung von Verbundverteilungen und bedingten Verteilungen in den Faktoren eine eigene sample-basierte Variante entwickelt wurde. Diese zeichnet sich dadurch aus, dass sowohl reellwertige als auch diskrete Dimensionen im Definitionsbereich vorkommen können. In Anlehnung an eine Kernel-Density-Darstellung wird eine Ähnlichkeitsfunktion $\delta_S(s_1, s_2)$ für zwei Samples $s_1$ und $s_2$ eingeführt, welche den Kernel für die Dichteberechnung ersetzt. Diese ergibt sich durch Multiplikation der Ähnlichkeiten in den im tiefgestellten Index angegebenen Dimensionen (siehe Gl. A 7.8)

$$\delta_D(s_1, s_2) = \prod_{d \in D} \delta_d(s_1, s_2) \tag{A 7.8}$$

Dabei finden für die Einzeldimensionen je nach Typ folgende Ähnlichkeitsfunktionen Anwendung:

- Reellwertige Größen werden mittels einer gauss-förmigen Glockenfunktion beschrieben, wobei der Parameter $\sigma$ die Trennschärfe zu den benachbarten Werten angibt.

$$\delta_{reell}(s_1, s_2) = e^{\frac{-(s_1 - s_2)^2}{\sigma^2}} \tag{A 7.9}$$

- Diskrete Größen ohne Ordnung sind sich alle gleich unähnlich.

$$\delta_{diskret}(s_1, s_2) = \begin{cases} \epsilon \ll 1, falls \;\; s_1 \neq s_2 \\ 1, sonst \end{cases} \tag{A 7.10}$$

  Dabei sorgt $\epsilon > 0$ dafür, dass im Produkt die anderen Dimensionen nicht eliminiert werden.
- Diskrete Größen mit Ordnung können entweder ebenfalls durch die Gauss-Funktion zu naheliegenden Elementen ähnlicher sein, als zu weiter entfernten (so in der vorliegenden Implementierung), oder es wird eine Rampenfunktion über dem Abstand genutzt.

Mittels der Ähnlichkeitsfunktion (Gl. A 7.8) sollen Verteilungen als Mengen gewichteter Samples $p(S) := \{(w_i, s_i)|i = 1, \ldots, N\}$ dargestellt werden, wobei das Integral von 1 nur durch einen konstanten Faktor $\eta$ gewährleistet wird, welcher allerdings für die konkreten Berechnungen nicht benötigt wird.

$$P(S = s) = \eta \sum_{i=1}^{N} w_i \delta_S(s, s_i) \tag{A 7.11}$$

Die Gewichte $w_i$ der Samples vermeiden dabei die Dopplung von Samples und ermöglichen eine effiziente online Schätzung der Verteilung durch einfache Hinzunahme eines Samples, bzw. durch inkrementieren des Gewichtes, falls das Sample bereits bekannt ist. Eine Reduktion der Gewichte kann auch zum Vergessen lange zurück liegender Beobachtungen genutzt werden. Da die Samplepositionen $s_i$ sich während der Berechnungen nicht ändern, kann die Matrix mit den Ähnlichkeiten zwischen den Samples vorberechnet und später wiederverwendet werden.
Der Faktor $\eta$ lässt sich unter Annahme der Stationarität der Ähnlichkeitsfunktionen über das Integral der Ähnlichkeitsfunktion über die Wertebereiche aller Dimensionen ($R(D)$) berechnen:

$$\eta = \frac{1}{N} \int_{s \in R(D)} \delta_D(0, s)\ ds \tag{A 7.12}$$

#### Operationen mit der Sample-Verteilung

Für die oben angeführten Message-Passing-Algorithmen werden verschiedene Grundoperationen mit Verteilungen benötigt. Hierbei sollen sowohl die Faktorpotentiale als auch die Messages und Randverteilungen in den Variablenknoten berücksichtigt werden.
Die einfachste Operation über einer mehrdimensionalen Sample-Verteilung $p(A, B, \ldots, X)$ ist die **Marginalisierung**. Um die Dichtefunktion entlang einzelner Dimensionen (beispielsweise über $A$) zu integrieren, kann diese Dimension bei den vorhandenen Samples einfach weggelassen werden. Aus $\{(w_i, (a, b, \ldots, x)_i)|i = 1, \ldots, N\}$ wird $\{(w_i, (b, \ldots, x)_i)|i = 1, \ldots, N\}$. Alter-

nativ kann für die Rechnung anstatt der vollständigen Ähnlichkeitsfunktion $\delta_{A,B,\ldots,X}(s_i, s_j)$ eine reduzierte Funktion $\delta_{B,\ldots,X}(s_i, s_j)$ genutzt werden, welche nur die übrigen Dimensionen umfasst. Die Laufzeitkomplexität dieser Operation ist demnach $O(n)$ mit $n$ als Anzahl der Samples, bzw. sogar eine Einsparung von Rechenzeit, wenn bei weiteren Operationen die vereinfachte Ähnlichkeitsfunktion genutzt wird.

Im Rahmen der Planung für die Dialogführung werden die Faktorpotentiale als Verbundverteilungen gespeichert, was ein einfaches Hinzufügen neuer Beobachtungen erlaubt. Für die Nutzung im Faktorgraph werden allerdings bedingte Wahrscheinlichkeiten benötigt. Diese zu erzeugen ist Aufgabe der **Konditionierungsoperation**. Dabei muss bei gegebener Verteilung $p(A, B)$ durch die Randverteilung $p(B)$ geteilt werden, um die bedingte Verteilung $P(A|B)$ zu erhalten. In der Sample-Repräsentation lässt sich dies recht einfach realisieren, indem die Gewichte der Samples angepasst werden, da nicht zu erwarten ist, dass an anderen Stützstellen durch die Division Wahrscheinlichkeitsdichte entsteht. Mit der oben definierten Marginalisierung ergibt sich für die neuen Gewichte:

$$w_i^{(A|B)} = \frac{w_i^{(A,B)}}{\sum_j w_j^{(A,B)} \delta_B(s_i, s_j)} \tag{A 7.13}$$

Aus der Gleichung ist ersichtlich, dass diese Operation mit $O(n^2)$ recht teuer ist. Für die Planung mit einem dynamischen Faktorgraph wird diese Operation daher vor dem Message-Passing beim Update der Modelle in den Faktoren durchgeführt. Dabei kann das Einfügen eines einzelnen neuen Samples in $O(n)$ realisiert werden.

Die wichtigste Operation für die Inferenz ist die **Multiplikation**. Hierbei können sowohl Verteilungen mit gleichem Definitionsbereich als auch Faktoren unterschiedlichen oder sich teilweise überlappenden Definitionsbereichs Multipliziert werden. Abb. A 7.4 zeigt eine Beobachtung, die für den praktischen Fall im Faktorgraph sehr hilfreich ist. Da im Faktorgraph die Domäne eines

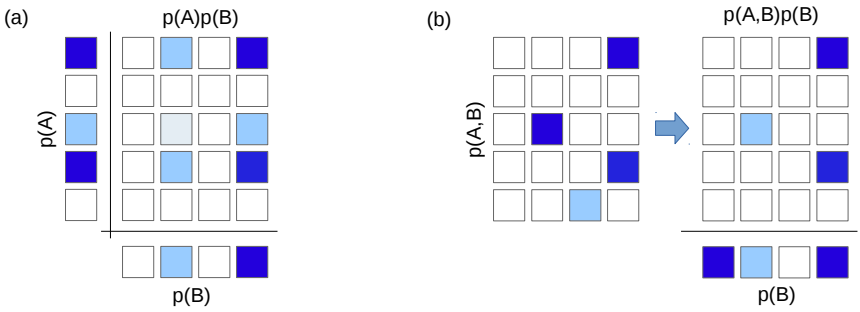



**Abbildung A 7.4:** (a) Kreuzprodukt Verteilungen disjunkter Definitionsbereiche, Anzahl der Samples vergrößert sich auf $|A||B|$. (b) Definitionsbereich eines Faktors $p(B)$ ist Teilmenge der des Faktors $p(A, B)$, Die Anzahl der Samples kann nicht über die von Faktor $p(A, B)$ wachsen (unter der Annahme, das die Ähnlichkeitsfunktion zwischen ungleichen Samples der Faktoren hinreichend gering ist).

Operanden immer eine Teilmenge der Domäne des zweiten Operanden ist[2], wird die Anzahl der benötigten Samples nicht wachsen. Im Gegensatz dazu können beim Kreuzprodukt zweier verschiedener Verteilungen mit $n$ Samples bis zu $n^2$ Stützstellen benötigt werden.

Die Berechnung eines Produktes zweier Sample-Verteilungen läuft somit unter Vernachlässigung einer Normierung auf eine Neuberechnug der Gewichte der Samples im Operanden mit der größeren Domäne hinaus[3].

$$w_i^{(A,B\cdot B)} = w_i^{(A,B)} \sum_j w_j^{(B)} \delta_B(s_i, s_j) \tag{A 7.14}$$

Dies lässt sich in $O(nm)$ ausführen wenn $n$ und $m$ die Anzahlen der Samples in den Operanden ist.

Eine letzte Operation, welche für die Planung eine Rolle spielt ist die **Maximierung** über einer Teilmenge der Dimensionen. Da hierfür ebenfalls $O(n^2)$

[2] Aufgrund der Definition der Faktorgraphen sind die Wertebereiche der Messages zwischen adjazenten Knoten immer Teilmengen der Wertebereiche der Faktoren

[3] Bei gleichem Wertebereich können die Samplestützstellen auch vereinigt werden.

viele Ähnlichkeitsberechnungen nötig wären, wurde bei der konkreten Implementierung ein Kompromiss eingegangen. Die Samples wurden in allen Dimensionen diskretisiert, auch wenn reellwertige Größen vorlagen. Dadurch ist es möglich, für die Samples einen Hashwert zu bestimmen und so kann die Menge identischer Samples ermittelt werden, ohne die Abstandsberechnung quadratischer Komplexität ausführen zu müssen. Dies ist einerseits notwendig, um doppelte Samples, wie sie nach einer Marginalisierung entstehen können, zu vereinen, sodass das Gewicht der resultierenden Samples der Höhe der Wahrscheinlichkeitsdichte entspricht und sich diese nicht auf mehrere Samples aufteilt. Andererseits erfolgt nach diesem Prinzip die Realisierung der Maximierungsoperation durch eine Iteration über alle Samples, wobei Samples anhand der nicht zu maximierenden Dimensionen gehasht und in eine Map mit ihrem Gewicht eingetragen werden. Dabei werden Samples überschrieben, sofern das neue Sample ein höheres Gewicht besitzt. Diese Auslese lässt sich somit quasi in $O(n)$ ausführen.

Für die Berechnungen im Faktorgraph werden in den Faktoren über die Zeit wachsende Sample-Verteilungen als Modell genutzt. Um ein unbeschränktes Wachstum der Samplemenge zu verhindern, wird beim Überschreiten einer Maximalanzahl das Modell ausgedünnt, indem die Samples mit geringstem Gewicht entfernt werden.

Diese Vereinfachung wird auch für die Zwischenergebnisse (Messages) angewendet. In der Praxis wurden für die Messages 20 und für die Modelle 1000 Samples erlaubt, wobei in den Modellen aufgrund der beschränkten Dauer der Experimente (siehe Abschn. 7.6) die Maximalanzahl nicht erreicht wurde. Dadurch konnte die Berechnung der Dialogaktion nahezu verzögerungsfrei umgesetzt werden.

Um ein lebenslanges Lernen der Modelle (Transitionsmodell $p(S_t|S_{t-1}, A_{t-1})$, Rewardmodell $p(R|S)$ und Zielzustandsmodell $p(G|S)$, siehe Abschn. 7.2.2) zu ermöglichen kann in Betracht gezogen werden, ein exponentielles Vergessen alter Beobachtungen zu implementieren. Dazu müssen lediglich vor dem Einfügen neuer Samples die Gewichte aller bereits Vorhandenen mit einem Faktor $< 1$ multipliziert werden. Wird dies nicht genutzt, so ist zu erwarten, dass nach Erreichen der Maximalanzahl der Samples immer das neue gelöscht wird, falls es einer unbekannten Beobachtung entspricht.

## A 7.4 Planung der Erfolgswahrscheinlichkeiten verschiedener Aktionen

Mit der in Abschn. 7.2 beschriebenen Modellbildung für den Dialogprozess ergibt sich der Faktorgraph aus Abb. 7.5 Seite 131. Um im Rahmen der Aktionsauswahl im Dialogagenten eine Bewertung für die zur Verfügung stehenden Aktionen zu erhalten, wird in einer Schleife der Max-Product Algorithmus (siehe A 7.3.2) angewendet. In Ergänzung zu Abschn. 7.4.1, in welchem die Abläufe verbal geschildert wurden, soll hier noch einmal die zu Grunde liegende Mathematik gegeben werden.

Wie bereits erwähnt ist die Länge der Aktionssequenzen bis zu einem Zielzustand von Vorhinein nicht bekannt und somit auch die Länge des ausgefalteten Faktorgraphs. Da die Zeitschritte prinzipiell identisch sind, läuft die Planung in einer Schleife bis zur maximalen Planungstiefe. Währenddessen ergeben sich für die Aktionen jeweils die Belegung, welche die Wahrscheinlichkeiten im Modell maximieren. Als Randbedingungen werden dabei die erwarteten Rewards in den erreichten Zuständen, die Erwünschtheit des Endzustands und der gegebene Ausgangszustand $S_0$ als beobachtete Variablenknoten ins Modell eingesetzt. Abb. A 7.5 zeigt den Faktorgraphen ergänzt um die Messages, welche im Laufe eines Zyklus versandt werden. Abweichend von Abschnitt 7.4.1 werden hier zur Vereinfachung der Notation die Zeitindizes der Zustandsvariablen $S_t$ und $S_{t-1}$ weggelassen. $S_t$ wird zur Unterscheidung von $S_{t-1}$ mit $S'$ bezeichnet.

Ausgangspunkt bildet die Verteilung über die Zielzustände. Diese wird als Sample-Verteilung (siehe A 7.3.4) $p(G, S') := \{(w_i^{(G,S')}, (g, s')_i) | i = 1, \ldots, K\}$ (Zielzustandsmodell) modelliert und lässt sich einfach durch neue Bobachtungen zur Laufzeit ergänzen. Für die korrekte Berechnung im probabilistischen Modell muss diese Verteilung konditioniert werden. Hierzu werden zunächst Samples für gleiche Zustände $s'$ mit unterschiedlichen $g$ Werten zusammengefasst und das $g \in [0, 1]$ (Erwünschtheit des Zustandes als Ziel) gemittelt. Dies ersetzt die Konditionierung und kann bereits beim Aufnehmen der Beobachtungen geschehen.

Zur Bestimmung der initialen Message $\mu_{p(G|S')\to S'}(S')$ (blau in Abb. A 7.5),

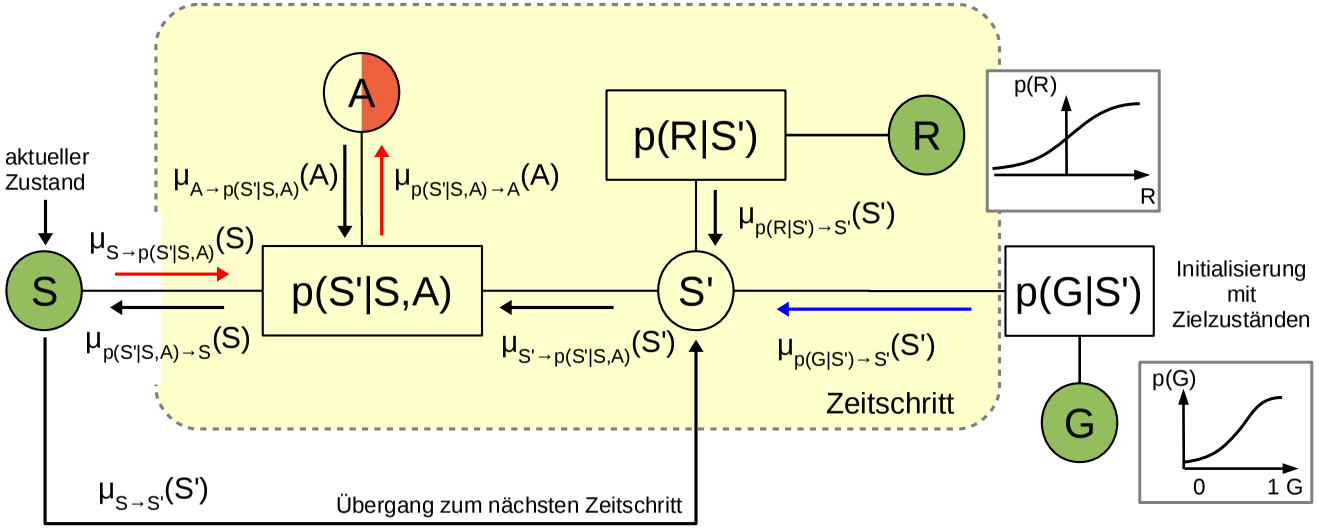



**Abbildung A 7.5:** Messageversand für die Berechnung des Max-Product-Algorithmus bei unbekannter Planungstiefe. Die Verteilung des Zielzustandes wird anhand $p(G|S')$ initialisiert. Anschließend beginnt die Iteration mit Neubewertung der $S'$ Samples anhand des Rewardmodells und Verrechnung der Messages von $S_0$ im Faktor $p(S'|S, A)$.

wird entsprechend der gewünschten Vorgabe, ein Ziel zu erreichen, für $G$ eine sigmoidale Gewichtung angenommen und genutzt, um Samples für die Zustände im Zielzustandsmodell neu zu gewichten. Dies entspricht dem Einsetzen der Beobachtung für $G$.

$$w_i^{(\mu_{p(G|S')\to S'})} = \frac{1}{1+e^{(-g_i+0.5)\sigma}} w_i^{(G,S')} \tag{A 7.15}$$

Mittels Gleichung A 7.15 ergeben sich die gesuchten Gewichte der Samples $i$, welche als Message $\mu_{p(G|S')\to S'}(S')$ an den Variablenknoten $S'$ als Ausgangshypothese gesandt werden. Dazu wird die Message zuvor noch durch weglassen der $g$ Komponenten der Samples marginalisiert auf die Variable $S'$. Der Skalierungsfaktor $\sigma = 10$ der Sigmoid-Funktion wird so gewählt, dass die Kurve bei 0 bzw. 1 in Sättigung geht (siehe Abb.A 7.5 rechts).

Um die Komplexität der nachfolgenden Berechnungen zu beschränken, erfolgt vor dem Betreten der Schleife noch ein Ausdünnen der Verteilung (in der genutzten Implementierung auf 20 Samples). Dies geschieht durch Weglassen der Samples mit den geringsten Gewichten.

Für jeden Planungsschritt der nun beginnenden Schleife ist zunächst die Beobachtung für den erwarteten Reward $R$ ins Rewardmodell $p(R,S') := \{(w_i^{(R,S')}, (r,s')_i) | i = 1,\ldots,L\}$ einzusetzen. Wie bei der Zielzustandsverteilung geschieht die Konditionierung auf $S'$ ebenfalls durch Mittlung der Rewardwerte $r_i$ für gleiche Zustände $s'_i$. Auch der erwartete Reward $p(R)$ wird mit einer sigmoidalen Verteilungsdichtefunktion modelliert, wie sie in Abb. A 7.5 zu sehen ist. Samples die hohe $r$ Werte besitzen, werden äquivalent zu den $(g,s')$ Samples aufgewertet, wohingegen jene mit kleinem $r$ eine Abwertung erfahren.

Dem Max-Produkt-Algorithmus folgend müsste nun eine Message $\mu_{p(R|S')\to S'}(S')$ gebildet und im Variablenknoten $S'$ auf die Message $\mu_{p(G|S')\to S'}(S')$ multipliziert werden, um die Message $\mu_{S'\to p(S'|S,A)}(S')$ zu berechnen. Laut Gl. A 7.14 würden dazu lediglich die in den Messages vorhandenen Samples neu gewichtet werden. Um durch Ausdünnen der Messages einen Informationsverlust zu vermeiden, wird an Stelle der Message vom Rewardmodell im Variablenknoten $S'$ das vollständige konditionierte

Rewardmodell genutzt.

Die im Variablenknoten $S'$ vorhandene Verteilung (anfangs von $p(G|S')$ später von Variablenknoten $S$) soll zur Bestimmung der Message $\mu_{S' \to p(S'|S,A)}(S')$ neu gewichtet werden. Dies entspricht dem Produkt mit $\mu_{p(R|S') \to S'}(S')$.

In einem ersten Schritt werden dazu für die vorhandenen Samples $(w_j, s'_j)$ die zu erwartenden Rewards $r_j$ aus dem Rewardmodell bestimmt. Dies geschieht durch einen Vergleich der Samples mit der Ähnlichkeitsfunktion und anschließender Mittlung.

$$r_j = \frac{\sum_{i=1}^{L} r_i \ \delta_{S'}\big(s_i^{(p(R|S'))}, s_j\big)}{\epsilon + \sum_{i=1}^{L} \delta_{S'}\big(s_i^{(p(R|S'))}, s_j\big)} \tag{A 7.16}$$

$\epsilon > 0$ sorgt dabei dafür, dass für im Rewardmodell unbekannte Samples ein Reward $r = 0$ resultiert.

Die neuen Gewichte ergeben sich dann durch Einsetzen der Rewards in die Sigmoid-Funktion wie folgt:

$$w_j^{(\mu_{S' \to p(S'|S,A)}(S))} = w_j^{(S')} \ \frac{1}{1 + e^{(-r_j \ \sigma)}} \tag{A 7.17}$$

Nach der Beschreibung in Abb. 7.7 folgt als nächster Schritt die Berechnung der Message vom Transitionsmodell (Faktor $p(S'|S, A)$) zum gesuchten Aktions Variablenknoten. Dazu wird angenommen die Sequenz der Dialogschritte beginne mit $S_0$.

Der Max-Produkt-Algorithmus sieht dafür folgende Schritte vor: 1) Multiplikation des Faktorpotentials $p(S'|S, A)$ mit den eingehenden Messages von $S$ und $S'$ und 2) die Maximierung über alle $S$ und $S'$.

Gemäß Gl. A 7.14 werden zunächst für die Samples im Transitionsmodell neue Gewichte $\hat{w}_i^{(p(S'|S,A))}$ anhand der Message $\mu_{S' \to p(S'|S,A)}(S')$ bestimmt. Diese wird später weiterverwendet. Die Multiplikation mit $S = s_0$ geschieht durch eine Neuberechnung der Gewichte der Samples im Transitionsmodell gemäß

der Ähnlichkeit zu $s_0$.

$$\check{w}_i = \hat{w}_i^{(p(S'|S,A))} \; \delta_S(s_0, s_i^{(p(S'|S,A))}) \tag{A 7.18}$$

Mit der resultierenden Sample-Verteilung $\{(\check{w}_i, (s', s, a)_i) | i = 1, \ldots, M\}$ wird gemäß A 7.3.4 die Maximierung vorgenommen und es entsteht eine Sample-Verteilung über $A$, welche als Message $\mu_{p(S'|S,A)\to A}(A)$ an den gesuchten Variablenknoten $A$ gesandt wird. Falls im Modell eine a-priori Wahrscheinlichkeit für die Aktionen vorgesehen ist, könnte diese nun damit multipliziert werden. In der konkreten Umsetzung wurde allerdings eine Gleichverteilung über die Aktionen angenommen. Demzufolge ist die Inferenz der Verteilung über $A$ für diese Planungstiefe abgeschlossen. Es wird an dieser Stelle für die zulässigen Aktionen aus $\hat{A}$ geprüft, ob die berechnete Wahrscheinlichkeit höher ist, als die aus vorangegangenen Planungsschritten und diese ggf. aktualisiert. Dies führt zur gesuchten Bewertung $p_{plan}(\hat{A})$.

Um den Planungshorizont zu erweitern, wird angenommen, der Faktorgraph setzte sich nach links periodisch fort, und der Variablenknoten $S$ sei nicht beobachtet.

Anhand der bereits berechneten $\hat{w}_i$, welche bereits das Produkt aus $p(S'|S, A)$ und $\mu_{S'\to p(S'|S,A)}(S')$ darstellen, kann nun die Message $\mu_{p(S'|S,A)\to S}(S)$ berechnet werden. Dazu muss laut Max-Product-Algorithmus zunächst mit der vom Knoten $A$ kommenden Message multipliziert werden, was bei der angenommen Gleichverteilung der $A$ entfallen kann, und anschließend über $S'$ und $A$ maximiert werden. Dies geschieht wiederum durch Hashing der $s$ Teile der Samples und Suche der maximalen Gewichte $w_i$ in den resultierenden Hash-Klassen (siehe A 7.3.4). Im Ergebnis entsteht eine Sample-Verteilung über der Zufallsvariablen $S$, welche zum Knoten $S$ gesendet wird, nachdem sie auf eine Maximalanzahl Samples (in der Implementierung 20) reduziert wurde, um die Komplexität der folgenden Berechnungen zu reduzieren. Da die beteiligten Verteilungen fortwährend multipliziert werden, ist zu erwarten, dass die Gewichte der Samples nicht numerisch stabil bleiben. Um diesem Umstand vorzubeugen, wird die Verteilung $\mu_{p(S'|S,A)\to S}(S)$ an dieser Stelle normalisiert. Dies ist nicht schädlich, da letztendlich lediglich ein Verhältnis der Bewertungen der möglichen Aktionen gesucht wird. Selbst, wenn die

korrekten resultierenden Wahrscheinlichkeiten aller Aktionen nahe null wären, muss vom System letztendlich eine ausgewählt werden. Dafür reicht eine Betrachtung des Verhältnisses.

Durch Übernahme der Verteilung $\mu_{p(S'|S,A)\to S}(S)$ in den Knoten $S'$ wird der Übergang zum nächsten Zeitschritt vollzogen und die Schleife beginnt von vorn.

Der bis hier geschilderte Ablauf lässt sich noch einmal, wie in Algorithmus A 7.1 dargestellt, zusammenfassen.

**Algorithmus A 7.1:** Planung( $s_0$, $p(S'|S,A)$, $p(R|S')$, $p(G|S')$, $\hat{A}$ )

**Eingaben**

$s_0$ // Ausgangszustand

$p(S'|S,A)$ // Transitionsmodell konditioniert

$p(R|S')$ // Rewardmodell konditioniert

$p(G|S')$ // Zielzustandsmodell konditioniert

$\hat{A}$ // zulässige Aktionsteilmenge

**Initialisierung**

$p_{plan}(\hat{A}) \leftarrow 0$ // Erfolgswkt. für alle zulässigen Aktionen = 0

$p(S') \leftarrow \sum_G p(G|S')prior(G)$ // Zielzustandsverteilung nach Gl.A 7.15

**Algorithmus**

`FOR t=1 bis T DO` // T Planungsschritte

$p(S') = \sum_R p(S')p(R|S')prior(R)$ // Multiplikation mit $\mu_{p(R|S')\to S'}(S')$

// Berechnung der Aktionsbelegung mit max Wkt.

$\mu_{p(S'|S,A)\to A}(A) = max_{S',S}\left(p(S'|S,A)p(S')p(S=s_0)\right)$

`FOREACH` $a \in \hat{A}$ `DO`

$p_{plan}(\hat{A}=a) = max\left\{p_{plan}(\hat{A}=a), \mu_{p(S'|S,A)\to A}(A=a)\right\}$

$\mu_{p(S'|S,A)\to S}(S) = max_{S',A}\left(p(S'|S,A)p(S')\right)$ // wahrscheinlichster

// Vorgängerzustand

// von $S'$? Annahme p(A) gleichverteilt

$p(S') = \mu_{p(S'|S,A)\to S}(S)$ // Übergang nächster Zeitschritt

`normalisiere` $p(S')$ // zur Vermeidung numerischer Probleme

`normalisiere` $p_{plan}(\hat{A})$

**Rückgabe**

$p_{plan}(\hat{A})$

## Aktionsauswahl

Im vorherigen Abschnitt wurde erläutert, wie die Erfolgswahrscheinlichkeiten $p_{plan}(\hat{A})$ der zur Verfügung stehenden Aktionen bestimmt werden können.

Leider berücksichtigt die Planung nur Wissen über Aktionsauswirkungen, die bereits einmal vom System erlebt wurden. Daher ergibt sich, wie in Abschn. 7.3 geschildert, die Notwendigkeit, die geplante Aktion zwecks Exploration und Berücksichtigung der Prioritäten abzuwandeln.
Dazu wird eine Statistik $p_{count}(\hat{A})$ der in Zustand $S$ bereits ausgeführten Aktionen herangezogen. Während einer Exploration sollen möglichst noch nicht so oft genutzte Aktionen bevorzugt werden. Die Verteilung $p_{prior}(\hat{A})$ dient der dialogphasenabhängigen Gewichtung der unterschiedlichen Aktionsmöglichkeiten (viel Erklärung am Anfang und rationelle proaktive Aktionen in späteren Phasen).
Konkret ergibt sich die Verteilung $p_{count}(\hat{A})$ aus den absoluten Häufigkeiten $H$ der Aktionsanwendungen im aktuellen Zustand nach Gl. A 7.19

$$p_{count}(\hat{A} = a) = \frac{H_a^{-1}}{\sum_{b \in \hat{A}} H_b^{-1}} \tag{A 7.19}$$

**Exploration** Da während eines längeren Dialoges durch explorative Auswahl die geplanten Verläufe quasi nie so umgesetzt werden, wie bei der Planung vorgesehen, ist es nötig, die Entscheidung zwischen Exploration oder Exploitation jeweils für eine komplette Dialogsequenz (von Aktivierung bis Beendigung eines Dialog-Frames) zufällig festzulegen. In Exploitation-Sessions wird die Aktionsauswahl allein anhand der $p_{plan}(\hat{A})$ Verteilung durch Sampling aus dieser Verteilung vorgenommen. Die sich dadurch ergebenden Aktionssequenzen sollten weitgehend den Erwartungen entsprechen, sofern der Nutzer erwartungskonforme Eingaben macht.
Im Fall von Explorations-Sessions werden die Verteilungen abhängig von der Anzahl $d$ der Ausführungen des aktuellen Dialog-Frames miteinander kombiniert, wobei die Einflussfaktoren sich mit unterschiedlichen Zeitkonstanten ändern. Der Einfluss der Reihenfolge $p_{prior}(\hat{A})$ dominiert anfangs und nimmt langsam über die Zeit ab. Ebenso ist auch $p_{count}(\hat{A})$ anfangs höher gewichtet, um eine breite Exploration zu gewährleisten. Der Einfluss von $p_{plan}(\hat{A})$ ist zu Beginn gering, da die Planung ohnehin anfangs noch keine sinnvol-

len Modelle zur Verfügung hat und nimmt mit steigender Anzahl geführter Dialog zu. Gl.A 7.20 zeigt, wie die multiplikative Kombination zu einer Verteilung $p_{draw}(\hat{A})$ erfolgt, aus der anschließend die auszuführende Aktion gezogen wird. Die Verteilungen werden dabei mittels der zeitabhängigen Gewichtungsfaktoren $e^{-d^2/\sigma}$ gegen eine Gleichverteilung gezogen um die Einflussstärke zu variieren.

$$\begin{aligned} p_{draw}(\hat{A}) = \quad & \left(p_{plan}(\hat{A}) + (1 - p_{plan}(\hat{A}))e^{-d^2/\sigma_{plan}}\right) \\ & \left(1 + (p_{count}(\hat{A}) - 1)e^{-d^2/\sigma_{count}}\right) \\ & \left(1 + (p_{prior}(\hat{A}) - 1)e^{-d^2/\sigma_{prior}}\right) \end{aligned} \tag{A 7.20}$$

## A 7.5 Nutzerpräferenzen

In Abschn. 7.5 wurde geschildert, wie mit Hilfe des Zielzustandsmodells $p(G, S')$ Entscheidungen des Nutzers anhand der vergangenen Nutzereingaben vorhergesagt werden können. Hier soll noch einmal auf die konkrete Implementierung dieser Möglichkeit eingegangen werden.

Die Vorhersage eines Slotwertes $V_x$ und Zuverlässigkeit $C_x$ geschieht indem alle Samples der zuvor erlebten Zielzustände ausgewertet werden, bei denen der $V_x$ Wert mit einer hohen Sicherheit $C_x$ bekannt ist. Abbruchzustände oder alternative Zielzustände, in denen die gesuchte Information nicht benötigt wurde, werden ignoriert.

Damit dabei eine gewisse Kontextsensitivität realisiert wird, werden die Endzustände zusätzlich mit dem aktuellen Zustand $s_0$ verglichen, wobei nur die Komponenten des Zustandsvektors genutzt werden, welche in $s_0$ nicht leer sind, sprich $C_i > 0$ haben. Außerdem werden für den Vergleich die Aktionszähler $Z_j$ ignoriert, da die ausgeführten Aktionen im Endzustand nicht zum aktuellen Zustand passen würden. Zunächst wird daher, wie eben erklärt, die Indexmenge $Y$ der zu vergleichenden Zustandsdimensionen bestimmt.

Mit Hilfe der Ähnlichkeitsfunktion $\delta_Y(s_0, s_i)$ lässt sich eine neue Wichtung der

Samples in $p(G|S') = \{(w_i, (g, s)_i)|i = 1, \ldots, N\}$ vornehmen und anschließend bilden die auf $V_x$ marginalisierten Samples mit ihren Gewichten eine Verteilung über den gesuchten Slotwerten. Die jeweiligen Zuverlässigkeiten $C_{xi} \subset s_i$ des gesuchten Slotwertes und die Zielhaftigkeit $g_i$ des Zustands im Sample werden ebenfalls auf die Gewichte multipliziert.

$$w_i^{(V_x)} = w_i^{(G|S)} \; \delta_Y(s_0, s_i) \; C_{xi} \; g_i \tag{A 7.21}$$

Für die vorkommenden Werte des Slots $V_x$ kann nun anhand der gewichteten Samples eine Statistik erhoben werden, wobei eine Normierung zur gesuchten Verteilung führt, wie sie anschließend im Inputinterpreter als virtuelle Eingabe verarbeitet wird.
Durch die systeminterne Veränderung der Slotinhalte könnte zu befürchten sein, dass das System seine eigenen Vorhersagen als Vorgeschichte interpretiert und somit ein Eigenleben entwickelt. Da aber nur Zielzustände und nicht alle beobachteten Zustände des Transitionsmodells genutzt werden, ist anzunehmen, dass der Nutzer jeweils mit der Entscheidung einverstanden war. Andererseits hätte er entweder eine Korrektur des Wertes vorgenommen oder den Dialog abgebrochen, was in einem nicht wünschenswerten Zustand geendet hätte, der mit $g_i = 0$ im Zielzustandsmodell verbucht würde und somit bei zukünftigen Vorhersagen keine Berücksichtigung fände.

# Abbildungsverzeichnis

# Literaturverzeichnis

[Allen et al., 2001] Allen, James F, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu und A. Stent (2001). Toward conversational human-computer interaction. AI magazine, 22(4):27.

[Amoretti und Reggiani, 2010] Amoretti, Michele und M. Reggiani (2010). Architectural paradigms for robotics applications. Advanced Engineering Informatics, 24(1):4–13.

[Arkin, 1995] Arkin, Ronald C (1995). Just what is a robot architecture anyway? Turing equivalency versus organizing principles. In: AAAI Spring Symposium: Lessons Learned from Implemented Software Architectures for Physical Agents.

[Armbrust et al., 2012] Armbrust, Christopher, D. Schmidt und K. Berns (2012). Generating behaviour networks from finite-state machines. In: Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on, S. 1–6. VDE.

[Arntz und Lewandowski, 2015] Arntz, Marlene und B. Lewandowski (2015). Weiterentwicklung eines Demonstrators für überwachte und angeleitete Bewegungsübungen. Projektseminar, TU Ilmenau.

[Asoh et al., 1999] Asoh, Hideki, T. Matsui, J. Fry, F. Asano und S. Hayamizu (1999). A spoken dialog system for a mobile office robot. In: Eurospeech.

[Bajones et al., 2015] Bajones, M., M. Zillich und M. Vincze (2015). Can you help me here please? Enabling robust robotics through human-robot behaviour coordination. In: HRI.

[Bishop et al., 2006] Bishop, Christopher M et al. (2006). Pattern recognition and machine learning, Bd. 4. springer New York.

[Blom, 2000] Blom, Jan (2000). Personalization: a taxonomy. In: CHI'00 extended abstracts on Human factors in computing systems, S. 313–314. ACM.

[Bobrow et al., 1977] Bobrow, Daniel G, R. M. Kaplan, M. Kay, D. A. Norman, H. Thompson und T. Winograd (1977). GUS, a frame-driven dialog system. Artificial intelligence, 8(2):155–173.

[Bohus et al., 2007] Bohus, Dan, A. Raux, T. K. Harris, M. Eskenazi und A. I. Rudnicky (2007). Olympus: an open-source framework for conversational spoken

language interface research. In: Proceedings of the workshop on bridging the gap: Academic and industrial research in dialog technologies, S. 32–39. Association for Computational Linguistics.

[Bohus und Rudnicky, 2003] Bohus, Dan und A. I. Rudnicky (2003). RavenClaw: Dialog Management Using Hierarchical Task Decomposition and an Expectation Agenda. In: Eighth European Conference on Speech Communication and Technology.

[Breazeal, 2003] Breazeal, Cynthia (2003). Toward sociable robots. Robotics and Autonomous Systems, 42(3):167–175.

[Brooks, 1986] Brooks, Rodney A (1986). A robust layered control system for a mobile robot. Robotics and Automation, IEEE Journal of, 2(1):14–23.

[Bui et al., 2004] Bui, Trung H, M. Rajman und M. Melichar (2004). Rapid dialogue prototyping methodology. In: Text, Speech and Dialogue, S. 579–586. Springer.

[Bunt et al., 2010] Bunt, Andrea, C. Conati und J. McGrenere (2010). Mixed-initiative interface personalization as a case study in usable AI. AI Magazine, 30(4):58.

[Butler et al., 2001] Butler, Greg, A. Gantchev und P. Grogono (2001). Object-oriented design of the subsumption architecture. Software: Practice and Experience, 31(9):911–923.

[Böhme et al., 2006] Böhme, Hans-Joachim, A. Scheidig, T. Wilhelm, Ch. Schröter, Ch. Martin, A. König, St. Müller und H.-M. Gross (2006). Progress in the Development of an Interactive Shopping-Assistant. In: Joint Conf. on Robotics: Int. Symp. on Robotics (ISR) and German Conference on Robotics (ROBOTIK). VDI.

[Canas et al., 2007] Canas, JM, D. Lobato und P. Barrera (2007). Jde+: an open-source schema-based framework for robotic applications. In: IEEE ICRA 2007 Workshop on Software Development and Integration in Robotics, SDIR-II, IEEE Robotics and Automation Society.

[Clark und Brennan, 1991] Clark, Herbert H und S. E. Brennan (1991). Grounding in communication. Perspectives on socially shared cognition, 13(1991):127–149.

[Clark und Wilkes-Gibbs, 1986] Clark, Herbert H und D. Wilkes-Gibbs (1986). Referring as a collaborative process. Cognition, 22(1):1–39.

[Connell, 1992] Connell, Jonathan H (1992). SSS: A hybrid architecture applied to robot navigation. In: Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on, S. 2719–2724. IEEE.

[Cuayáhuitl et al., 2010] Cuayáhuitl, Heriberto, S. Renals, O. Lemon und H. Shimodaira (2010). Evaluation of a hierarchical reinforcement learning spoken dialogue system. Computer Speech and Language, 24:395–429.

[Dahn, 2015] Dahn, Nikolas (2015). Integration von SIP basierter Videokonferenz und Fernsteuerung eines mobilen Serviceroboters in einer Android App. Design-Projekt, TU Ilmenau.

[Duffy et al., 2005] Duffy, Brian R, M. Dragone und G. M. O'Hare (2005). Social robot architecture: A framework for explicit social interaction. In: Android Science: Towards Social Mechanisms, CogSci 2005 Workshop, Stresa, Italy.

[Döring et al., 2015] Döring, N., K. Richter, H.-M. Gross, C. Schröter, S. Müller, M. Volkhardt, A. Scheidig und K. Debes (2015). Robotic Companions for Older People: A Case Study in the Wild. Annual Review of CyberTherapy and Telemedicine.

[Ebert, 2012] Ebert, Christoph (2012). Entwicklung einer Smartphone-App als Interface zur Roboterinteraktion und Fernsteuerung. Masterarbeit, TU Ilemenau.

[Einhorn und Gross, 2015] Einhorn, Erik und H.-M. Gross (2015). Generic NDT Mapping in Dynamic Environments and its Application for Lifelong SLAM. Robotics and Autonomous Systems, 69:28–39.

[Einhorn et al., 2012] Einhorn, Erik, T. Langner, R. Stricker, Ch. Martin und H.-M. Gross (2012). MIRA – Middleware for Robotic Applications. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), S. 2591–2598. IEEE.

[Eyben et al., 2012] Eyben, F., B. Schuller und G. Rigoll (2012). Improving Generalisation and Robustness of Acoustic Affect Recognition. In: Proceedings of the 14th ACM International Conference on Multimodal Interaction, ICMI, S. 517–522.

[Fox et al., 1999] Fox, Dieter, W. Burgard, F. Dellaert und S. Thrun (1999). Monte carlo localization: Efficient position estimation for mobile robots. Bd. 1999, S. 343–349.

[Fox et al., 1997] Fox, Dieter, W. Burgard und S. Thrun (1997). The dynamic window approach to collision avoidance. IEEE Robotics & Automation Magazine, 4(1):23–33.

[Frey et al., 1997] Frey, Brendan J, F. R. Kschischang, H.-A. Loeliger und N. Wiberg (1997). Factor graphs and algorithms. In: Proceedings of the Annual Allerton Conference on Communication Control and Computing, Bd. 35, S. 666–680. University of Illinois.

[Funk, 2015] Funk, Harald (2015). Ansteuerung von OLED-Displays zur Darstellung von emotionalen Augen eines Assistenzroboters. Masterarbeit, TU Ilmenau.

[Gajos et al., 2004] Gajos, Krzysztof, R. Hoffmann und D. S. Weld (2004). Improving user interface personalization. Supplementary Proceedings of UIST, Vol. 4.

[Gebhard, 2005] Gebhard, Patrick (2005). ALMA: a layered model of affect. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, S. 29–36. ACM.

[Geiger et al., 2013] Geiger, J., I. Yenin, T. Rehrl, F. Wallhoff und G. Rigoll (2013). Display of Emotions with the Robotic Platform ALIAS. In: Tagungsband des 6. Deutschen Ambient Assisted Living (AAL 2013) Kongresses, Berlin, S. 248–253. VDE Verlag.

[Geue, 2012] Geue, Paul-Oliver (2012). Vorbereitung und Begleitung des Langzeiteinsatzes des robotischen Mobilisierungsassistenten HOROS in einer Seniorenwohnanlage. Diplomarbeit, TU Ilmenau.

[Gries, 2014] Gries, Vanessa (2014). Bewegungsübungen mit Feedback für Senioren auf einem Serviceroboter. Bachelorarbeit, TU Ilmenau.

[Gross et al., 2008] Gross, H-M, H.-J. Böhme, C. Schröter, S. Müller, A. König, C. Martin, M. Merten und A. Bley (2008). Shopbot: Progress in developing an interactive mobile shopping assistant for everyday use. In: Systems, Man and Cybernetics, 2008. SMC 2008. IEEE International Conference on, S. 3471–3478. IEEE.

[Gross et al., 2009] Gross, H-M, H. Böhme, C. Schröter, S. Müller, A. König, E. Einhorn, C. Martin, M. Merten und A. Bley (2009). TOOMAS: interactive shopping guide robots in everyday use-final implementation and experiences from long-term field trials. In: Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on, S. 2005–2012. IEEE.

[Gross et al., 2015] Gross, H.-M., S. Müller, C. Schröter, M. Volkhardt, A. Scheidig, K. Debes, K. Richter und N. Döring (2015). Robot Companion for Domestic Health Assistance: Implementation, Test and Case Study under Everyday Conditions in Private Apartments.. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS).

[Gross et al., 2011a] Gross, H.-M., C. Schröter, S. Müller, M. Volkhardt, E. Einhorn, A. Bley, T. Langner, C. Martin und M. Merten (2011a). I'll keep an eye on you:

Home robot companion for elderly people with cognitive impairment. In: Systems, Man, and Cybernetics (SMC), 2011 IEEE International Conference on, S. 2481–2488. IEEE.

[Gross et al., 2011b] Gross, H.-M., C. Schröter, S. Müller, M. Volkhardt, E. Einhorn, A. Bley, M. Ch., T. Langner und M. Merten (2011b). Progress in Developing a Socially Assistive Mobile Home Robot Companion for the Elderly with Mild Cognitive Impairment. In: Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2011), S. 2430–2437.

[Gross et al., 2012] Gross, H.-M., C. Schröter, S. Müller, M. Volkhardt, E. Einhorn, A. Bley, T. Langner, M. Merten, C. Huijnen, H. van den Heuvel und A. van Berlo (2012). Further Progress towards a Home Robot Companion for People with Mild Cognitive Impairment. In: Proc. IEEE Int. Conf. on Systems, Man, and Cybernetics (IEEE-SMC 2012), S. 637–644.

[Gross et al., 2014] Gross, Horst-Michael, K. Debes, E. Einhorn, St. Müller, A. Scheidig, Ch. Weinrich, A. Bley und Ch. Martin (2014). Mobile Robotic Rehabilitation Assistant for Walking and Orientation Training of Stroke Patients: A Report on Work in Progress. In: IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), S. 1880–1887. IEEE.

[Gross et al., 2005] Gross, Horst-Michael, A. König und St. Müller (2005). Omniview-Based Concurrent Map Building and Localization Using Adaptive Appearance Maps. In: IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), S. 3510–3515. IEEE.

[Gross et al., 2006] Gross, Horst-Michael, J. Richarz, St. Müller, A. Scheidig und Ch. Martin (2006). Probabilistic Multi-Modal People Tracker and Monocular Pointing Pose Estimator for Visual Instruction of Mobile Robot Assistants. In: IEEE-INNS Int. Joint Conf. on Neural Networks (IJCNN) and IEEE World Congress on Computational Intelligence (WCCI), S. 8325–8333. IEEE.

[Göller, 2013] Göller, M. (2013). Behavior-based Control for Service Robots inspired by Human Motion Patterns A robotic shopping assistant. Doktorarbeit, Karlsruher Instituts für Technologie (KIT) Universität des Landes Baden-Württemberg.

[Göller et al., 2009] Göller, M, T. Kerscher, J. Zollner, R. Dillmann, M. Devy, T. Germa und F. Lerasle (2009). Setup and control architecture for an interactive shopping cart in human all day environments. In: Advanced Robotics, 2009. ICAR 2009. International Conference on, S. 1–6. IEEE.

[Hartley und Pipitone, 1991] Hartley, Ralph und F. Pipitone (1991). Experiments with the subsumption architecture. In: Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on, S. 1652–1658. IEEE.

[Heerink, 2011] Heerink, Marcel (2011). How elderly users of a socially interactive robot experience adaptiveness, adaptability and user control. In: Computational Intelligence and Informatics (CINTI), 2011 IEEE 12th International Symposium on, S. 79–84. IEEE.

[Hommel, 2009] Hommel, Sebastian (2009). Zeitliche Analyse von Emotionen auf Basis von Acitve Appearance Modellen. Diplomarbeit, Technische Universität Ilmenau, Fak IA, FG NIKR.

[Hu et al., 2004] Hu, Weiming, T. Tan, L. Wang und S. Maybank (2004). A survey on visual surveillance of object motion and behaviors. Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on, 34(3):334–352.

[Jaimes und Sebe, 2007] Jaimes, Alejandro und N. Sebe (2007). Multimodal human–computer interaction: A survey. Computer vision and image understanding, 108(1):116–134.

[Jameson, 2008] Jameson, Anthony (2008). Adaptive Interfaces and Agents. In: Sears, Andrew und J. A. Jacko, Hrsg.: The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications, S. 433–458. CRC Press, Boca Raton, FL, 2nd Aufl.

[Jordan, 1998] Jordan, Michael Irwin (1998). Learning in Graphical Models. Springer Science & Business Media.

[Kalesse, 2008] Kalesse, Sören (2008). Entwicklung eines Framework zum effizienten Inferieren, Lernen und Planen in probabilistischen Modellen. Diplomarbeit, TU Ilmenau.

[Kawamura et al., 2000] Kawamura, Kazuhiko, R. A. Peters, D. M. Wilkes, W. A. Alford und T. E. Rogers (2000). ISAC: Foundations in human-humanoid interaction. IEEE Intelligent Systems, 15(4):38–45.

[Kim et al., 2008] Kim, Kyungduk, C. Lee, S. Jung und G. G. Lee (2008). A frame-based probabilistic framework for spoken dialog management using dialog examples. In: Proceedings of the 9th SIGdial Workshop on Discourse and Dialogue, S. 120–127. Association for Computational Linguistics.

[Kleinehagenbrock, 2004] Kleinehagenbrock, Marcus (2004). Interaktive Verhaltenssteuerung für robot companions. Doktorarbeit, Universität Bielefeld.

[Knight et al., 2001] Knight, Sylvia, G. Gorrell, M. Rayner, D. Milward, R. Koeling und I. Lewin (2001). Comparing grammar-based and robust approaches to speech understanding: a case study.. In: INTERSPEECH, S. 1779–1782.

[Knoop et al., 2004] Knoop, Steffen, S. Vacek, R. Zöllner, C. Au und R. Dillmann (2004). A CORBA-based distributed software architecture for control of service robots. In: IROS, S. 3656–3661.

[Knox und Stone, 2009] Knox, W Bradley und P. Stone (2009). Interactively shaping agents via human reinforcement: The TAMER framework. In: Proceedings of the fifth international conference on Knowledge capture, S. 9–16. ACM.

[Knox et al., 2013] Knox, W Bradley, P. Stone und C. Breazeal (2013). Training a robot via human feedback: A case study. In: Social Robotics, S. 460–470. Springer.

[Kschischang et al., 2001] Kschischang, Frank R, B. J. Frey und H.-A. Loeliger (2001). Factor graphs and the sum-product algorithm. Information Theory, IEEE Transactions on, 47(2):498–519.

[König et al., 2005] König, Alexander, St. Müller und H.-M. Gross (2005). Appearance-Based CML Approach for a Home Store Environment. In: Europ. Conf. on Mobile Robots (ECMR), S. 206–211. stampalibri.

[Lalanne et al., 2009] Lalanne, Denis, L. Nigay, P. Robinson, J. Vanderdonckt, J.-F. Ladry et al. (2009). Fusion engines for multimodal input: a survey. In: Proceedings of the 2009 international conference on Multimodal interfaces, S. 153–160. ACM.

[Langley, 1997] Langley, Pat (1997). Machine learning for adaptive user interfaces. In: KI-97: Advances in artificial intelligence, S. 53–62. Springer.

[Langley, 1999] Langley, Pat (1999). User Modeling in Adaptive Interfaces. In: Proceedings of the seventh international conference on user modeling, S. 357–370.

[Lapp, 2008] Lapp, Hans-Christian (2008). Modellierung eines nutzerangepassten Langzeit-Dialoges. Diplomarbeit, TU-Ilmenau.

[Larsson und Traum, 2000] Larsson, Staffan und D. R. Traum (2000). Information state and dialogue management in the TRINDI dialogue move engine toolkit. Natural language engineering, 6(3&4):323–340.

[Lee et al., 2001] Lee, A., T. Kawahara und K. Shikano (2001). Task-based dialog management using an agenda. In: European Conf. on Speech Communication and Technology, S. 1691–1694.

[Lee et al., 2009] Lee, Cheongjae, S. Jung, S. Kim und G. G. Lee (2009). Example-based dialog modeling for practical multi-domain dialog system. Speech Communication, 51(5):466–484.

[Lemon et al., 2001] Lemon, Oliver, A. Bracy, A. Grünstein und S. Peters (2001). Information states in a multimodal dialogue system for human-robot conversation. In: 5th Workshop on Formal Semantics and Pragmatics of Dialogue (Bi-Dialog 2001), S. 57–67.

[Lemon et al., 2002] Lemon, Oliver, A. Grünstein, A. Battle und S. Peters (2002). Multi-tasking and collaborative activities in dialogue systems. In: Proceedings of the 3rd SIGdial workshop on Discourse and dialogue-Volume 2, S. 113–124. Association for Computational Linguistics.

[Levin et al., 2000] Levin, Esther, R. Pieraccini und W. Eckert (2000). A stochastic model of human-machine interaction for learning dialog strategies. Speech and Audio Processing, IEEE Transactions on, 8(1):11–23.

[Liu, 2005] Liu, Xiao-Wen Terry (2005). An intuitive and flexible architecture for intelligent mobile robots. Doktorarbeit, The University of Manitoba.

[Lohfeld, 2014] Lohfeld, Marc (2014). Haptische Mensch-Roboter Interaktion zur lokalen Navigation eines mobilen Assistenzroboters. Bachelorarbeit, TU-Ilmenau.

[Martin et al., 2005] Martin, C., A. Scheidig, T. Wilhelm, C. Schröter, H.-J. Böhme und H.-M. Gross (2005). A new Control Architecture for Mobile Interaction Robots. In: Proc. of the 2nd European Conference on Mobile Robots (ECMR 2005), S. 224–229.

[Matarić, 1997] Matarić, Maja J (1997). Behaviour-based control: Examples from navigation, learning, and group behaviour. Journal of Experimental & Theoretical Artificial Intelligence, 9(2-3):323–336.

[Matarić, 2001] Matarić, Maja J (2001). Learning in behavior-based multi-robot systems: Policies, models, and other agents. Cognitive Systems Research, 2(1):81–93.

[McTear, 2004] McTear, Michael (2004). Spoken dialogue technology: Towards the conversational user interface. Springer.

[McTear, 2002] McTear, Michael F (2002). Spoken dialogue technology: enabling the conversational user interface. ACM Computing Surveys (CSUR), 34(1):90–169.

[Merten et al., 2012] Merten, M., A. Bley, C. Schröter und H.-M. Gross (2012). A mobile robot platform for socially assistive home-care applications. In: Proc. 7th German Conference on Robotics, S. 233–238.

[Mudrova und Hawes, 2015] Mudrova, Lenka und N. Hawes (2015). Task Scheduling for Mobile Robots Using Interval Algebra. In: 2015 IEEE International Conference on Robotics and Automation, Seattle, Washington, USA.

[Müller et al., 2015] Müller, S, C. Schröter und H.-M. Gross (2015). Smart Fur Tactile Sensor for a Socially Assistive Mobile Robot. In: Intelligent Robotics and Applications, S. 49–60. Springer.

[Müller et al., 2008] Müller, Steffen, S. Hellbach, E. Schaffernicht, A. Ober, A. Scheidig und H.-M. Gross (2008). Whom to Talk to? Estimating User Interest from Movement Trajectories. In: IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), S. 532–538. IEEE.

[Müller et al., 2005] Müller, Steffen, A. König und H.-M. Gross (2005). Neural Architecture for Concurrent Map Building and Localization Using Adaptive Appearance Maps. In: Int. Conf. on Artificial Neural Networks (ICANN), Bd. 3697 d. Reihe LNCS, S. 929–934. Springer.

[Müller et al., 2007] Müller, Steffen, E. Schaffernicht, A. Scheidig, H.-J. Böhme und H.-M. Gross (2007). Are You Still Following Me?. In: Europ. Conf. on Mobile Robots (ECMR), S. 211–216. Albert-Ludwigs-Universität Freiburg – Universitätsverlag.

[Müller et al., 2013] Müller, Steffen, Ch. Schröter und H.-M. Gross (2013). Low-Cost Whole-Body Touch Interaction for Manual Motion Control of a Mobile Service Robot. In: Int. Conf. on Social Robotics (ICSR), Bd. 8239 d. Reihe LNCS, S. 229–238. Springer.

[Müller et al., 2014a] Müller, Steffen, S. Sprenger und H.-M. Gross (2014a). OfficeMate – a Study of an Online Learning Dialog System for Mobile Assistive Robots. In: Int. Conf. on Adaptive and Self-Adaptive Systems and Applications (ADAPTIVE), S. 104–110.

[Müller et al., 2014b] Müller, Steffen, S. Sprenger und H.-M. Gross (2014b). Online Adaptation of Dialog Strategies based on Probabilistic Planning. In: IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), S. 692–697. IEEE.

[Murphy, 2002] Murphy, Kevin Patrick (2002). Dynamic bayesian networks: representation, inference and learning. Doktorarbeit, University of California, Berkeley.

[Paek, 2006] Paek, Tim (2006). Reinforcement learning for spoken dialogue systems: Comparing strengths and weaknesses for practical deployment. In: Proc. Dialog-on-Dialog Workshop, Interspeech. Citeseer.

[Pahl, 2011] Pahl, Christina (2011). Sensibilisierung eines Roboterkopfes mittels taktiler Sensoren. Belegarbeit im Fach Mensch-Maschine Kommunikation, TU Ilmenau.

[Peltason und Wrede, 2010a] Peltason, Julia und B. Wrede (2010a). Modeling Human-Robot Interaction Based on Generic Interaction Patterns.. In: AAAI Fall Symposium: Dialog with Robots.

[Peltason und Wrede, 2010b] Peltason, Julia und B. Wrede (2010b). Pamini: A framework for assembling mixed-initiative human-robot interaction from generic interaction patterns. In: Proceedings of the 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue, S. 229–232. Association for Computational Linguistics.

[Peter Bonasso et al., 1997] Peter Bonasso, R, R. James Firby, E. Gat, D. Kortenkamp, D. P. Miller und M. G. Slack (1997). Experiences with an architecture for intelligent, reactive agents. Journal of Experimental & Theoretical Artificial Intelligence, 9(2-3):237–256.

[Prüger, 2008] Prüger, Tobias (2008). Audiobasierte Merkmale zur multimodalen Nutzermodellierung. Diplomarbeit, Technische Universität Ilmenau.

[Rao et al., 1995] Rao, Anand S, M. P. Georgeff et al. (1995). BDI Agents: From Theory to Practice. In: ICMAS, Bd. 95, S. 312–319.

[Reiter und Dale, 2000] Reiter, E. und R. Dale (2000). Building Natural Language Generation Systems. Cambridge University Press.

[Rich et al., 2001] Rich, Charlie, C. L. Sidner und N. Lesh (2001). Collagen: Applying Collaborative Discourse Theory to Human-Computer Interaction. AI magazine, 22(4).

[Russell, 1980] Russell, James A (1980). A circumplex model of affect. Journal of personality and social psychology, 39(6):1161.

[Sacks et al., 1974] Sacks, Harvey, E. A. Schegloff und G. Jefferson (1974). A simplest systematics for the organization of turn-taking for conversation. Language, S. 696–735.

[Saragih et al., 2011] Saragih, Jason M, S. Lucey und J. F. Cohn (2011). Deformable model fitting by regularized landmark mean-shift. International Journal of Computer Vision, 91(2):200–215.

[Schatzmann et al., 2006] Schatzmann, Jost, K. Weilhammer, M. Stuttle und S. Young (2006). A survey of statistical user simulation techniques for reinforcement-learning of dialogue management strategies. The Knowledge Engineering Review, 21(2):97–126.

[Scheidig et al., 2006] Scheidig, Andrea, St. Müller, S.ller, Ch. Martin und H.-M. Gross (2006). Generating Person's Movement Trajectories on a Mobile Robot. In: IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), S. 747–752. IEEE.

[Schneemann, 2013] Schneemann, Friederike (2013). Recherche und Evaluation von Features zur Detektion von gestürzten Personen in häuslichen Umgebungen. Masterarbeit, TU Ilmenau.

[Schnürer, 2015] Schnürer, Thomas (2015). Integration von Spracheingabe in einen multimodalen Dialogmanager. Bachelorarbeit, TU Ilmenau, Fachgebiet NI&KR.

[Schröter et al., 2009] Schröter, Christof, M. Höchemer, St. Müller und H.-M. Gross (2009). Autonomous Robot Cameraman – Observation Pose Optimization for a Mobile Service Robot in Indoor Living Space. In: IEEE Int. Conf. on Robotics and Automation (ICRA), S. 424–429. IEEE.

[Schröter et al., 2013] Schröter, Christof, St. Müller, M. Volkhardt, E. Einhorn, C. Huijnen, H. van den Heuvel, A. van Berlo, A. Bley und H.-M. Gross (2013). Realization and User Evaluation of a Companion Robot for People with Mild Cognitive Impairments. In: IEEE Int. Conf. on Robotics and Automation (ICRA), S. 1145–1151. IEEE.

[Sprenger, 2013] Sprenger, Sina (2013). Akzeptanzerhöhung durch nutzerspezifische Dialogstrategien. Masterarbeit, TU Ilmenau.

[Stoytchev und Arkin, 2001] Stoytchev, Alexander und R. C. Arkin (2001). Combining deliberation, reactivity, and motivation in the context of a behavior-based robot architecture. In: Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on, S. 290–295. IEEE.

[Stricker et al., 2015a] Stricker, R., S. Müller und H.-M. Gross (2015a). R2D2 Reloaded: Dynamic Video Projection on a Mobile Service Robot.. In: Proc. 2015 Europ. Conf. on Mobile Robotics (ECMR), Lincoln,UK.

[Stricker et al., 2012a] Stricker, Ronny, St. Müller, E. Einhorn, Ch. Schröter, M. Volkhardt, K. Debes und H.-M. Gross (2012a). Interactive Mobile Robots Guiding Visitors in a University Building. In: IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), S. 695–700. IEEE.

[Stricker et al., 2012b] Stricker, Ronny, St. Müller, E. Einhorn, Ch. Schröter, M. Volkhardt, K. Debes und H.-M. Gross (2012b). Konrad and Suse, Two Robots Guiding Visitors in a University Building. In: Autonomous Mobile Systems 2012 (AMS), Informatik aktuell, S. 49–58. Springer.

[Stricker et al., 2014] Stricker, Ronny, St. Müller und H.-M. Gross (2014). Non-contact Video-based Pulse Rate Measurement on a Mobile Service Robot. In: IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), S. 1056–1062. IEEE.

[Stricker et al., 2015b] Stricker, Ronny, S. Müller und H.-M. Gross (2015b). Universal Usage of a Video Projector on a Mobile Guide Robot. In: Intelligent Robotics and Applications, S. 25–36. Springer.

[Tannert, 2014] Tannert, Marco (2014). Fuzzy-basierte Kategorisierung von körperlichen Aktivitäten im Haushalt. Diplomarbeit, TU Ilmenau.

[Thrun et al., 2005] Thrun, Sebastian, W. Burgard und D. Fox (2005). Probabilistic robotics. MIT Press.

[Toptsis et al., 2004] Toptsis, Ioannis, S. Li, B. Wrede und G. A. Fink (2004). A multi-modal dialog system for a mobile robot. In: INTERSPEECH.

[Traum, 2008] Traum, David (2008). Approaches to Dialogue Systems and Dialogue Management - Lecture Notes and Bibliography. http://people.ict.usc.edu/~traum/ESSLLI08/ Abrufdatum: 10.12.2015.

[Viola und Jones, 2004] Viola, Paul und M. J. Jones (2004). Robust real-time face detection. International journal of computer vision, 57(2):137–154.

[Volkhardt und Gross, 2013a] Volkhardt, Michael und H.-M. Gross (2013a). Finding people in home environments with a mobile robot. In: Mobile Robots (ECMR), 2013 European Conference on, S. 282–287. IEEE.

[Volkhardt und Gross, 2013b] Volkhardt, Michael und H.-M. Gross (2013b). Finding People in Apartments with a Mobile Robot. In: IEEE Int. Conf. on Systems, Man, and Cybernetics (SMC), S. 4348–4353. IEEE.

[Volkhardt et al., 2009] Volkhardt, Michael, S. Kalesse, St. Müller und H.-M. Gross (2009). Maximum a Posteriori Estimation of Dynamically Changing Distributions. In: German Conf. on Artificial Intelligence (KI), Bd. 5803 d. Reihe LNAI, S. 484–491. Springer.

[Volkhardt et al., 2011a] Volkhardt, Michael, St. Müller, Ch. Schröter und H.-M. Gross (2011a). Detection of Lounging People with a Mobile Robot Companion.

In: Int. Conf. on Intelligent Robotics and Applications (ICIRA), Bd. 7102 d. Reihe LNCS, S. 328–337. Springer.

[Volkhardt et al., 2011b] Volkhardt, Michael, St. Müller, Ch. Schröter und H.-M. Gross (2011b). Playing Hide and Seek with a Mobile Companion Robot. In: IEEE-RAS Int. Conf. on Humanoid Robots (HUMANOIDS), S. 40–46. IEEE.

[Volkhardt et al., 2013] Volkhardt, Michael, C. Weinrich und H.-M. Gross (2013). Multi-modal people tracking on a mobile companion robot. In: Mobile Robots (ECMR), 2013 European Conference on, S. 288–293. IEEE.

[Wachsmuth et al., 2012] Wachsmuth, Sven, F. Siepmann, L. Ziegler, F. Lier und M. Schöpfer (2012). ToBI-Team of Bielefeld: The Human-Robot Interaction System for RoboCup@ Home 2012.

[Weinrich, 2015] Weinrich, Christoph (2015). Personenwahrnehmung für eine sozialverträgliche und nutzerzentrierte Roboternavigation in öffentlichen Einsatzumgebungen. Doktorarbeit, TU Ilmenau.

[Weinrich et al., 2012] Weinrich, Christoph, Ch. Vollmer und H.-M. Gross (2012). Estimation of Human Upper Body Orientation for Mobile Robotics using an SVM Decision Tree on Monocular Images. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), S. 2147–2152. IEEE.

[Weinrich et al., 2014] Weinrich, Christoph, T. Wengefeld, Ch. Schroeter und H.-M. Gross (2014). People Detection and Distinction of their Walking Aids in 2D Laser Range Data based on Generic Distance-Invariant Features. In: IEEE Int. Symp. on Robot and Human Interactive Communication (RO-MAN), S. 767–773. IEEE.

[Weninger et al., 2014] Weninger, Felix, B. Schuller, F. Eyben, M. Wöllmer und G. Rigoll (2014). A Broadcast News Corpus for Evaluation and Tuning of German LVCSR Systems. arXiv preprint arXiv:1412.4616.

[Williams, 2003] Williams, J. (2003). A probabilistic model of human/computer dia- logue with application to a partially observable marcov decisioning process. Technischer Bericht, Machine Intelligence Laboratory Department of Engineering University of Cambridge.

[Williams und Young, 2007] Williams, Jason D und S. Young (2007). Partially observable Markov decision processes for spoken dialog systems. Computer Speech & Language, 21(2):393–422.

[Young et al., 2013] Young, Steve, M. Gasic, B. Thomson und J. D. Williams (2013). POMDP-based statistical spoken dialog systems: A review. Proceedings of the IEEE, 101(5):1160–1179.