Eine modulare Kontrollarchitektur für den Hol- und Bringdienst von Roboterassistenten

Von der Fakultät für Maschinenbau
der Universität Stuttgart zur Erlangung der Würde eines
Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

Vorgelegt von
Dipl.-Ing. Matthias Hans
aus Würzburg

Hauptberichter:	Univ.-Prof. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. E. Westkämper
Mitberichter:	Prof. Dr. rer. nat. T. Christaller
Tag der mündlichen Prüfung:	4. Februar 2005

Institut für Industrielle Fertigung und Fabrikbetrieb der Universität Stuttgart
2005

IPA-IAO Forschung und Praxis

Berichte aus dem
Fraunhofer-Institut für Produktionstechnik und
Automatisierung (IPA), Stuttgart,
Fraunhofer-Institut für Arbeitswirtschaft und
Organisation (IAO), Stuttgart,
Institut für Industrielle Fertigung und
Fabrikbetrieb (IFF), Universität Stuttgart
und Institut für Arbeitswissenschaft und
Technologiemanagement (IAT), Universität Stuttgart

Herausgeber:
Univ.-Prof. Dr.-Ing. Prof. e.h. Dr.-Ing. e.h. Dr. h.c. mult. Engelbert Westkämper
und
Univ.-Prof. Dr.-Ing. habil. Prof. e.h. Dr. h.c. Hans-Jörg Bullinger

Universität Stuttgart
Institut für Industrielle Fertigung und Fabrikbetrieb

Fraunhofer Institut Produktionstechnik und Automatisierung

Matthias Hans

Eine modulare Kontrollarchitektur für den Hol- und Bringdienst von Roboterassistenten

Nr. 412

JOST-JETTER VERLAG
Fachverlag · 71296 Heimsheim

Dr.-Ing. Matthias Hans
Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA), Stuttgart

Univ.-Prof. Dr.-Ing. Prof. e.h. Dr.-Ing. e.h. Dr. h.c. mult. Engelbert Westkämper
ord. Professor an der Universität Stuttgart
Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA), Stuttgart

Univ.-Prof. Dr.-Ing. habil. Prof. e.h. Dr. h.c. Hans-Jörg Bullinger
ord. Professor an der Universität Stuttgart
Präsident der Fraunhofer-Gesellschaft, München

D 93

ISBN 3-936947-51-1 Jost Jetter Verlag, Heimsheim

Druck: printsystem GmbH, Heimsheim

Geleitwort der Herausgeber

Über den Erfolg und das Bestehen von Unternehmen in einer marktwirtschaftlichen Ordnung entscheidet letztendlich der Absatzmarkt. Das bedeutet, möglichst frühzeitig absatzmarktorientierte Anforderungen sowie deren Veränderungen zu erkennen und darauf zu reagieren.

Neue Technologien und Werkstoffe ermöglichen neue Produkte und eröffnen neue Märkte. Die neuen Produktions- und Informationstechnologien verwandeln signifikant und nachhaltig unsere industrielle Arbeitswelt. Politische und gesellschaftliche Veränderungen signalisieren und begleiten dabei einen Wertewandel, der auch in unseren Industriebetrieben deutlichen Niederschlag findet.

Die Aufgaben des Produktionsmanagements sind vielfältiger und anspruchsvoller geworden. Die Integration des europäischen Marktes, die Globalisierung vieler Industrien, die zunehmende Innovationsgeschwindigkeit, die Entwicklung zur Freizeitgesellschaft und die übergreifenden ökologischen und sozialen Probleme, zu deren Lösung die Wirtschaft ihren Beitrag leisten muss, erfordern von den Führungskräften erweiterte Perspektiven und Antworten, die über den Fokus traditionellen Produktionsmanagements deutlich hinausgehen.

Neue Formen der Arbeitsorganisation im indirekten und direkten Bereich sind heute schon feste Bestandteile innovativer Unternehmen. Die Entkopplung der Arbeitszeit von der Betriebszeit, integrierte Planungsansätze sowie der Aufbau dezentraler Strukturen sind nur einige der Konzepte, welche die aktuellen Entwicklungsrichtungen kennzeichnen. Erfreulich ist der Trend, immer mehr den Menschen in den Mittelpunkt der Arbeitsgestaltung zu stellen - die traditionell eher technokratisch akzentuierten Ansätze weichen einer stärkeren Human- und Organisationsorientierung. Qualifizierungsprogramme, Training und andere Formen der Mitarbeiterentwicklung gewinnen als Differenzierungsmerkmal und als Zukunftsinvestition in *Human Resources* an strategischer Bedeutung.

Von wissenschaftlicher Seite muss dieses Bemühen durch die Entwicklung von Methoden und Vorgehensweisen zur systematischen Analyse und Verbesserung des Systems Produktionsbetrieb einschließlich der erforderlichen Dienstleistungsfunktionen unterstützt werden. Die Ingenieure sind hier gefordert, in enger Zusammenarbeit mit anderen Disziplinen, z. B. der Informatik, der Wirtschaftswissenschaften und der Arbeitswissenschaft, Lösungen zu erarbeiten, die den veränderten Randbedingungen Rechnung tragen.

Die von den Herausgebern langjährig geleiteten Institute, das

- Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA),

- Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO),

- Institut für Industrielle Fertigung und Fabrikbetrieb (IFF), Universität Stuttgart,

- Institut für Arbeitswissenschaft und Technologiemanagement (IAT), Universität Stuttgart

arbeiten in grundlegender und angewandter Forschung intensiv an den oben aufgezeigten Entwicklungen mit. Die Ausstattung der Labors und die Qualifikation der Mitarbeiter haben bereits in der Vergangenheit zu Forschungsergebnissen geführt, die für die Praxis von großem Wert waren. Zur Umsetzung gewonnener Erkenntnisse wird die Schriftenreihe „IPA-IAO - Forschung und Praxis“ herausgegeben. Der vorliegende Band setzt diese Reihe fort. Eine Übersicht über bisher erschienene Titel wird am Schluss dieses Buches gegeben.

Dem Verfasser sei für die geleistete Arbeit gedankt, dem Jost Jetter Verlag für die Aufnahme dieser Schriftenreihe in seine Angebotspalette und der Druckerei für saubere und zügige Ausführung. Möge das Buch von der Fachwelt gut aufgenommen werden.

Engelbert Westkämper Hans-Jörg Bullinger

Vorwort

Die vorliegende Arbeit entstand im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Fraunhofer-Institut für Produktionstechnik und Automatisierung (IPA) in Stuttgart. Sie wurde teilweise im Rahmen des Leitprojekts MORPHA mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 01 IL 902 G/9 gefördert.

Herrn Univ.-Prof. Dr.-Ing. Prof. E.h. Dr.-Ing. E.h. Dr. h.c. mult. Engelbert Westkämper danke ich für die wohlwollende Unterstützung und Förderung, die die Durchführung dieser Arbeit ermöglichte.

Herrn Prof. Dr. rer. nat. Thomas Christaller danke ich für die Übernahme des Mitberichts und die direkte, eingehende Durchsicht meiner Arbeit.

Besonders danke ich Herrn Prof. Dr.-Ing. Dr. h.c. mult. Rolf Dieter Schraft, Herrn Prof. Dr. rer. nat. Joachim Hertzberg, Herrn Dr.-Ing. Manfred Schweizer, Herrn Dr. rer. nat. Klaus Schmidt, Herrn Dr.-Ing. Arno Ritter und Herrn Dr.-Ing. Hendrik Rust für ihre kritischen Anregungen. Sie waren ein Ansporn zur Fertigstellung der Arbeit und große Hilfestellung.

Aus dem Kreise der Kollegen möchte ich vor allem Herrn Dipl.-Inform. Winfried Baum und Herrn Dipl.-Ing. (FH) MEng Jens Kubacki für die zahlreichen fachlichen Diskussionen und Frau Luzia Schuhmacher (M.A.) für die sorgfältige Durchsicht des Manuskriptes danken. Ferner gilt mein Dank dem ganzen MORPHA-Projektteam für die technische Unterstützung.

Herzlich bedanken möchte ich mich bei meiner Freundin Tanja Yvonne Richter. Sie stand mir stets mit großer Geduld während der Durchführung der Arbeit zur Seite.

Stuttgart, im Februar 2005 Matthias Hans

Inhaltsverzeichnis

Abkürzungsverzeichnis **12**

Namen und Abkürzungen 12

Formelzeichen 13

1 Einleitung **14**

1.1 Problemstellung 14

1.2 Zielsetzung und Vorgehensweise 15

2 Ausgangssituation **16**

2.1 Definitionen und Begriffe 16

2.2 Nutzenpotenziale von Roboterassistenten 18

2.3 Aufgaben von Roboterassistenten 19

2.3.1 Produktionsszenario 19

2.3.2 Haushaltsszenario 19

3 Analyse der Aufgabenstellung und Ableitung von Anforderungen **21**

3.1 Analyse der Szenarien 21

3.1.1 Mobilität 21

3.1.2 Manipulation 22

3.1.3 Umgebungserfassung 22

3.1.4 Kommunikation mit dem Menschen 22

3.1.5 Zielgerichtetes Verhalten 24

3.1.6 Robustes Agieren in menschlicher Umgebung 24

3.2 Analyse von Hol- und Bringdiensten 26

3.2.1 Zusammensetzung aus Handlungsschritten 26

3.2.2 Störquellen 29

3.2.3 Komplexität von kombinierten Hol- und Bringdiensten 30

3.3 Strukturierung der Anforderungen 31

4 Stand der Technik **32**

4.1 Roboterassistenten 32

4.2 Robotersteuerungen von Servicerobotern 33

4.3 Kontrollarchitekturen für mobile Agenten 34
4.3.1 Sequentielle Schichtenarchitektur 34
4.3.2 Verhaltensbasierte Architekturen 35
4.3.3 Mehrebenen-Architekturen 36
4.3.4 BDI-Theorie und PRS-Architekturen 38
4.3.5 Konnektionismus 39
4.3.6 Ableitung von Anforderungen 40
4.4 Kognitive Systeme 41
4.4.1 Problemlösen 41
4.4.2 Planen 42
4.4.3 Entscheiden 45
4.5 Wissensbasen 46

5 Lösungsansatz für die Kontrollarchitektur 49

5.1 Entwicklung des Gesamtkonzeptes 49
5.2 Konzeption des Ausführungsmoduls 51
5.2.1 Funktionen des Ausführungsmoduls 52
5.2.2 Eigenschaften von Aktionen 53
5.2.3 Behandlung von Störsituationen während der Aktionsausführung 55
5.3 Entwicklung des Ausführungsmoduls 56
5.3.1 Variantenbildung 56
5.3.2 Bewertung 57
5.3.3 Auswahl und Schlussfolgerungen 59
5.4 Konzeption des Planungsmoduls 59
5.4.1 Planungsproblem 59
5.4.2 Variantenbildung und Auswahl des Planungsmoduls 60
5.4.3 Schlussfolgerungen aus der Auswahl des Planungsmoduls 60
5.5 Konzeption der Wissensbasis 61
5.5.1 Entwicklung einer Ontologie zur symbolischen Umweltrepräsentation 62
5.5.2 Modellierung der Aktionen 66
5.6 Konzeption zur Wartung der Wissensbasis 67
5.7 Konzeption der Mensch-Maschine-Schnittstelle 68

5.8 Zusammenführung der Teilkonzepte zur Kontrollarchitektur 70
5.8.1 Gesamtsystem 70
5.8.2 Regelablauf 71
5.8.3 Erfüllung der Anforderungen 73

6 Realisierung der Kontrollarchitektur 74

6.1 Roboterassistent Care-O-bot® II 74
6.1.1 Elektromechanische Komponenten 75
6.1.2 Rechnerarchitektur 79
6.2 Realisierung des Ausführungsmoduls 80
6.2.1 Behandlung von Störsituationen 82
6.2.2 Anbindung an die Robotersteuerung 82
6.2.3 Aktionen für den Hol- und Bringdienst 83
6.3 Realisierung der weiteren Softwaremodule 90
6.3.1 Planungsmodul 90
6.3.2 Datenbank für die Wissensbasis 91
6.3.3 Mensch-Maschine-Schnittstelle 93

7 Validierung und Bewertung 94

7.1 Haushaltsszenario 94
7.2 Experimentelle Untersuchungen 95
7.2.1 Zielgerichtetes Verhalten 95
7.2.2 Robustes Agieren in menschlicher Umgebung 97
7.2.3 Kommunikation und Interaktion 99
7.3 Erweiterungsfähigkeit 99
7.4 Übertragung der Kontrollarchitektur auf andere Serviceroboter 100

8 Zusammenfassung und Ausblick 101

9 Summary 104

10 Literatur und Links 108

Abkürzungsverzeichnis

Namen und Abkürzungen

Zeichen	Bedeutung	Quelle
2-D / 3-D	Zweidimensional / Dreidimensional	
3T	Three interacting tiers	(Bonasso et al. 1997)
A i	Anforderung mit laufender Nummer i	
Ã *j*	Übergeordnete Anforderung mit laufender Nummer *j*	
ADL	Action Description Language	(Pednault 1989)
ATLANTIS	A Three-Layer Architecture for Navigating Through Intricate Situations	(Gat 1991)
BDI	Believe, Desire, Intention	
CCD	Charged Coupled Devices	
CORBA	Common Object Request Broker Architecture	
Cyc	Cyc® Knowledge Base	(Cycorp 2003)
GUI	Graphical User Interface (Grafische Benutzerschnittstelle)	
GUM	Generalized Upper Model	(Bateman et al. 1994)
JNI	Java Native Interface	
KI	Künstliche Intelligenz	
KM-Sensor	Kraft-Momenten-Sensor	
KR	Knowledge Representation (Wissensrepräsentation)	
LAN	Local Area Network	
MDP	Markov Decision Process	(Puterman 1994)
PC	Personal Computer	
PDDL	Problem Domain Definition Language	(Ghallab et al. 1998)
POMDP	Partially Observable Markov Decision Process	(Cassandra 1999)
PRS	Procedural Reasoning System	(Georgeff, Lansky 1987)
RAP	Reactive Action Packages	(Firby 1989)
SQL	Structured Query Language	
STRIPS	Stanford Research Institute Problem Solver	(Fikes, Nilsson 1971)
TCA	Task Control Architecture	(Simmons 1994)
TCP	Tool Center Point (Werkzeugmittelpunkt)	
TDL	Task Description Language	(Simmons, Apfelbaum 1998)
UML	Unified Modeling Language	(Oesterreich 1997)
XML	Extensible Markup Language	

Formelzeichen

Zeichen	Bedeutung
δ	Überführungsfunktion
κ	Kostenfunktion
ω	Ausgabefunktion
a	Aktion
A	Menge aller Aktionen
K	Anzahl von Kombinationen
K_{max}	maximale Anzahl von Kombinationen
l	Ort (englisch: location)
n	Zählvariable, z.B. Anzahl von Objekten
o	Operation
O	Menge aller Operationen
P_a	Wahrscheinlichkeit bei Ausführung von Aktion a
s	Situation
s_0	Ausgangssituation
s'	Nachfolgesituation
S	Situationsraum
S_z	Menge aller Zielsituationen
X	Vektor aller Eingangswerte
Y	Vektor aller Ausgangsgrößen
z	Zustand
z_0	Ausgangszustand
z'	Nachfolgezustand
Z	Zustandsraum
Z_z	Menge aller Zielzustände
$\mathcal{A}$	Menge aller Ablagepositionen
$\mathcal{A}_m$	Menge aller Ablagepositionen auf Möbel $m \in \mathcal{M}$
$\mathcal{B}$	Menge aller Bier-Objekte
$\mathcal{C}$	Menge aller Cola-Objekte
$\mathcal{L}$	Menge aller Orte (englisch: location)
$\mathcal{M}$	Menge aller Möbel
$\mathcal{O}$	Menge aller handhabbaren Objekte
$\mathcal{P}$	Menge aller Chips-Objekte (Pringles)
$\mathcal{R}$	Menge aller Räume

1 Einleitung

1.1 Problemstellung

Innovative Entwicklungen auf dem Gebiet der Sensor-, Steuerungs- und Antriebstechnik führen zu einem Anstieg der technischen Intelligenz in Maschinen (Westkämper 2001). Sie erschließen eine Reihe von neuen Einsatzmöglichkeiten auch außerhalb der industriellen Fertigung in leistungsfähigeren Produkten.

Nachdem in Deutschlands Industriehallen schon über 100.000 Industrieroboter den Fertigungsablauf unterstützen (UNECE 2003), sehen die ersten Roboterhersteller den Dienstleistungsbereich als künftiges Anwendungsfeld mit einem enormen Wachstumspotenzial (Schraft, Schmierer 1998) (KUKA 2003). Die so genannten Serviceroboter dienen anders als die bekannten Industrieroboter nicht der industriellen Erzeugung von Sachgütern, sondern der Verrichtung von Leistungen an Menschen und Einrichtungen (Schraft, Volz 1996).

Langfristig betrachtet werden in Zukunft die Serviceroboter den Menschen täglich bei der Arbeit unterstützen (UNECE 2003). Sie werden dabei nicht nur Fußböden reinigen oder Spielzeuge sein, sondern in einer fortgeschrittenen Entwicklungsstufe, als so genannte Roboterassistenten, werden sie Diener für den Menschen sein und ihn bei der Hausarbeit (Graefe, Bischoff 2003) oder in der Produktion unterstützen (Hägele et al. 2001). Diese Serviceroboter müssen im Alltag mit technischen Laien interagieren, sie müssen in menschlicher Umgebung arbeiten und sie müssen ein breites Spektrum an Tätigkeiten an wechselnden Orten erfüllen (Levi et al. 2003).

Für die intuitive Kooperation zwischen Mensch und Maschine muss es gelingen, dass die technischen Systeme, insbesondere die Roboterassistenten, einfach in der Bedienung werden (bdw 2004). Es wird deshalb von ihnen eine Intelligenz gefordert, die den Roboterassistenten ein autonomes Verhalten verleiht, so dass sie Tätigkeiten auf Kommando ausführen können (Christaller 2001). Sie müssen dafür fähig zur Entscheidungsfindung sein und mit zahlreichen Sensoren selbstständig die sie umgebende Umwelt wahrnehmen und interpretieren. Sie erstellen Handlungspläne zur Lösung komplexer Aufgabenstellungen, die sie selbstständig ausführen. Bei der Ausführung müssen sie auf unvorhersehbare Ereignisse reagieren und mit Störungen bzw. falschen Annahmen umgehen. Bei der Aktionsausführung sollen sie außerdem mit dem Menschen auf intelligente und natürlich empfundene Weise kommunizieren.

1.2 Zielsetzung und Vorgehensweise

Die wesentlichen Teilsysteme von Servicerobotern, wie z.B. mobile Plattformen, Antriebstechnik, Sensoren und Bediensysteme, sind modular aufgebaut und auf andere Einsatzfälle übertragbar (Schraft, Schmierer 1998). Dies soll auch für das geforderte interaktive und autonome Verhalten gelten. Ziel der vorliegenden Arbeit ist es, grundlegende Erkenntnisse zur Modularisierung der Verhaltenskontrolle von Roboterassistenten zu erarbeiten und durch eine systematische Vorgehensweise die Randbedingungen und Potenziale einer übertragbaren, modularen Kontrollarchitektur für Roboterassistenten aufzuzeigen.

Von dem breiten Spektrum an Tätigkeiten für Roboterassistenten wird der Hol- und Bringdienst als Datenbasis analysiert. Er repräsentiert grundlegend das Handlungsschema „Ortswechsel – Tätigkeit – Ortswechsel", wobei in diesem Falle die Tätigkeit das Greifen bzw. Abgeben von Objekten ist. Zur Verallgemeinerung der Ergebnisse kann eine andere Tätigkeit in dieses Schema eingesetzt werden. Aus den Ergebnissen der Analyse werden Anforderungen an die Gesamtarchitektur, an ihre Module und an ihr Zusammenwirken zur Verhaltenskontrolle abgeleitet.

Veröffentlichte Kontrollarchitekturen nach dem Stand der Technik werden anhand der erarbeiteten Anforderungen bewertet und Defizite identifiziert. Für jedes Modul erfolgt die Konzeption verschiedener Varianten und die Auswahl favorisierter Lösungskonzepte. Für das Teilsystem Verhaltenskontrolle wird ein neues Ausführungsmodul entwickelt, welches das Verhalten des Roboterassistenten an die aktuelle Situation adaptiert, Störungen in der Ausführung erkennt und Gegenmaßnahmen einleitet. Für das kognitive System wird auf den Stand der Technik zurückgegriffen. Mit einer neuen Ontologie für den Hol- und Bringdienst werden die Regeln und Fakten zum Planen und zur Ausführung definiert.

Anhand einer praktischen Realisierung der Kontrollarchitektur in einem Roboterassistenten für Haushalt und Pflege wird in einem Testszenario der Hol- und Bringdienst demonstriert und die Zielorientierung und Robustheit seines Verhaltens überprüft. Mit Betrachtungen zur Erweiterungsfähigkeit und Übertragung der modularen Kontrollarchitektur auf andere Serviceroboter wird die Allgemeingültigkeit des Ansatzes bestätigt.

2 Ausgangssituation

2.1 Definitionen und Begriffe

Agent

Ganz allgemein ist ein Agent jemand, der im Auftrag eines anderen handelt. Auf technische Systeme übertragen, werden Agenten als **autonome, kooperierende Entitäten** in zum Teil verteilten, dezentralen Systemen verstanden (Ritter 2003). Grundsätzlich sind **technische Agenten** von Software-Agenten zu unterscheiden. Im Unterschied zu reinen Software-Agenten verfügt ein technischer Agent über mechatronische Komponenten inklusive einer Steuerung (Ritter 2003). Der technische Agent nimmt seine Umgebung über Sensoren wahr und handelt in ihr mit Aktuatoren. Dabei wählt er selbstständig die Handlung, die für ihn aufgrund seines Wissens die erfolgversprechendste ist (Russel, Norvig 1995).

Serviceroboter

Ein Serviceroboter ist eine frei programmierbare Bewegungseinrichtung, die teil- oder vollautomatisch **Dienstleistungen** verrichtet. Dienstleistungen sind dabei Tätigkeiten, die nicht der direkten industriellen Erzeugung von Sachgütern, sondern der Verrichtung von Leistungen an Menschen und Einrichtungen dienen (Schraft, Volz 1996).

Roboterassistent

Roboterassistenten können als weitere Evolutionsstufe von Servicerobotern verstanden werden. Sie sind **Helfer des Menschen**, charakterisiert durch ihre erweiterte, dem Menschen ähnliche Wahrnehmungs-, Interaktions- und Kommunikationsfähigkeit, und agieren in dessen natürlichen Umgebungen, wie zum Beispiel im Haushalt oder in Fertigungshallen (Hägele et al. 2001).

Roboterassistenten sind spezielle technischen Agenten, die primär nicht mit anderen Agenten, sondern mit dem Menschen interagieren und kooperieren. Die wichtigsten Teilsysteme von Roboterassistenten sind in Bild 2-1 dargestellt.

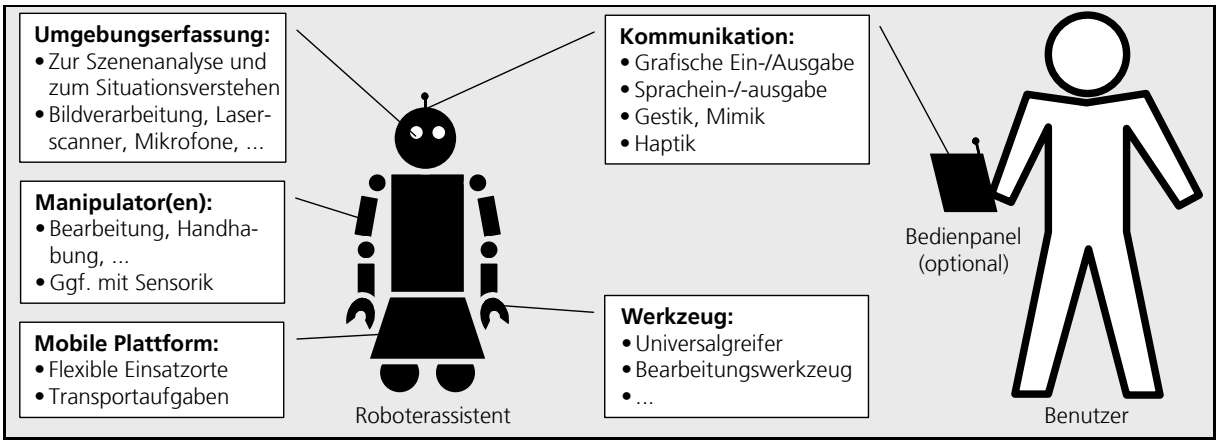


Bild 2-1: Teilsysteme eines Roboterassistenten.

Kontrollarchitektur

Die Kontrollarchitektur legt die Struktur und damit den während der Laufzeit im Allgemeinen unveränderlichen Anteil der Steuerung fest. Dabei wird zwischen zwei Aspekten unterschieden: Zum einen bestimmt die Architektur den **internen Aufbau aus funktionalen Bestandteilen** und das funktionelle Zusammenwirken zwischen diesen (Jung, Fischer 1998). Zum anderen dient die Architektur der **Verhaltenskontrolle**, indem sie die Prinzipien zur Steuerung des Verhaltens im Sinne des Einwirkens auf die Umgebung festlegt (Wooldridge 1999).

Planen und Handlungsplanung

Planen bedeutet das Erstellen einer Struktur von Aktionen, deren Ausführung ein gegebenes Problem löst. Im Forschungsgebiet der **Künstlichen Intelligenz** (KI) bedeutet Planen eindeutig die **Handlungsplanung**, womit die Lesart des Begriffs gegenüber seiner umgangssprachlichen Verwendung eingeschränkt wird (Biundo et al. 1995, S. 754). Die Handlungsplanung beschäftigt sich im weitesten Sinn mit der Erzeugung eines **zielgerichteten Verhaltens** für Agenten.

Die klassische Sicht des Planens wird durch den **Situationskalkül** geprägt. Grundelemente des Situationskalküls sind Situationen und Aktionen. Eine **Situation** ist der „Schnappschuss" des interessierenden Ausschnitts der Welt zu einem Zeitpunkt, beschrieben durch eine Menge logischer Formeln, die in dieser Situation gelten. Eine **Aktion** ist die formale Beschreibung einer Handlung in der Welt. Eine Aktion überführt eine gegebene Situation in eine andere Situation, die Nachfolgesituation. Der Situationskalkül ist eine formale Vorschrift, wie aus einer gegebenen Situation s und einer Aktion $a(s)$, die in s ausgeführt wird, die Nachfolgesituation s' errechnet wird (Biundo et al. 1995, S. 775).

Im engeren Sinne lässt sich Handlungsplanung damit folgendermaßen definieren: Gegeben sind eine Ausgangssituation, ein Ziel sowie eine Menge möglicher Aktionen. Gesucht ist ein Plan als eine Abfolge von Aktionen, die, wenn sie in der Ausgangssituation ausgeführt wird, eine Situation erreicht, in der das Ziel gilt (Köhler 2000).

Symbol System Hypothesis

Die *Symbol System Hypothesis* von Simon sagt aus, dass **Intelligenz** auf einem System aus Symbolen beruht (Simon 1969). Es wird impliziert, dass Wahrnehmungs- und Motorenschnittstellen jeweils mit Symbolen repräsentiert werden, mit denen die zentrale Intelligenz rechnet, d.h. das Planen basiert unabhängig von der Aufgabe und dem Umgebungskontext auf Symbolen. Jedes Symbol repräsentiert genau eine Entität in der realen Welt. Entitäten können individuelle Objekte sein, Eigenschaften, Konzepte, Erwartungen, Gefühle, Nationen, Farben, etc.; wichtig ist, dass jede Entität von einem Symbol repräsentiert wird.

Ontologie

Der Begriff Ontologie kommt aus der Philosophie (Duden 1997): Ontologie ist die Wissenschaft vom Sein, von den Ordnungs-, Begriffs- und Wesensbestimmungen des Seienden. In der Informatik wird Ontologie für die Wissensrepräsentation von schlussfolgernden Systemen (*reasoning systems*) gebraucht: Eine Ontologie ist die **explizite Spezifikation einer Konzeption** (Gruber 1993), d.h. sie ist eine Definition des Vokabulars für die Darstellung von Wissen.

In der Logik ist der Existenz-Quantifizierer ∃ eine Notation zur Aussage, dass etwas existiert. In der Logik selbst gibt es jedoch kein Vokabular zum Beschreiben der Dinge, die existieren. Diese Lücke füllen Ontologien: sie sind die Lehre der Existenz, von allem, was existiert – abstrakt oder konkret. Ontologien stellen die Prädikate der Prädikatenlogik zur Verfügung (Sowa 2000).

2.2 Nutzenpotenziale von Roboterassistenten

Der Einsatz von Roboterassistenten als Helfer des Menschen kann von vielseitigem Nutzen sein. Beispiele sind (Praßler et al. 1999):

- Die Verringerung körperlicher und mentaler Belastungen des Menschen
- Technische Hilfen für den Menschen in speziellen Situationen
- Erhöhung der Arbeitsleistung
- Produktionsmethoden, die den Fähigkeiten des Menschen besser angepasst sind

- Flexiblerer Einsatz von Maschinen
- Partielle Selbstüberwachung technischer Anlagen

Den genannten Beispielen ist gemeinsam, dass sie auf Seiten der Maschinen eine „Basisintelligenz" voraussetzen. Erst dadurch ist eine effiziente Kooperation möglich. Die Möglichkeit der auf die Benutzer angepassten Kooperation kann neben dem Einsatz in der Produktion zahlreiche Serviceaufgaben erschließen und es erlauben, Vorteile technischer Systeme wie z.B. deren Präzision nutzbar zu machen. Die Zielrichtung dieser Entwicklung ist somit eine verbesserte Ressourcennutzung in sehr allgemeinem Sinne.

2.3 Aufgaben von Roboterassistenten

Zwei beispielhafte Szenarien veranschaulichen den Einsatz von Roboterassistenten:

2.3.1 Produktionsszenario

Mobile Roboterassistenten, die Aufgaben in Kooperation mit dem Menschen erledigen, können eine neue Qualität in der Verbesserung von Produktionsprozessen hinsichtlich Produktivitätssteigerung und Humanisierung von Arbeitsplätzen erzielen, wenn sie den Menschen nicht ersetzen, sondern ihn unterstützen. Der Mensch übernimmt Kommandierungs-, Überwachungs- und Belehrungsfunktionen. Er greift in Fällen, in denen die Maschine nicht „weiter weiß", zur Führung und weiteren Belehrung ein und ist Partner im arbeitsteiligen Fertigungsprozess (Praßler et al. 1999).

In einem künftigen Produktionsszenario wird der Roboterassistent den Menschen bei Handlangeraufgaben, Transportaufgaben und Inspektionsaufgaben in einer typischen komplexen Fertigungsumgebung mit Werkzeugmaschinen, Förderbändern, Lagern, usw. unterstützen. Eine typische Aufgabe ist das Greifen unsortierter Teile, der Transport zu einer Werkzeugmaschine, zu einem Montagearbeitsplatz, das Assistieren bei der Montage oder der Transport zu einem Inspektions- bzw. Messplatz (Hägele et al. 2001).

2.3.2 Haushaltsszenario

Künftige technische Systeme bieten Unterstützung im Ablauf des täglichen Lebens. Visionen gehen davon aus, Roboterassistenten als Massenprodukte in unserem täglichen Umfeld (Haushalt) für eine Vielzahl anfallender Tätigkeiten zu nutzen. Diese Systeme mit multimedialer Schnittstelle, Handhabungsarm und Greifer erfüllen alle Anforderungen an ein technologisch

und wirtschaftlich anspruchsvolles Massenprodukt mit erheblichem mittel- bis langfristigem Marktpotenzial (UNECE 2002).

Der Roboterassistent im Haushalt und Pflegebereich soll es dem Benutzer ermöglichen, trotz möglicher Behinderung eine Reihe elementarer Tätigkeiten ohne fremde Hilfe durchzuführen. Hieraus ergeben sich für einen Roboterassistenten typischerweise folgende Funktionen (Schaeffer, May 1999):

- Persönliche Versorgung durch Ausführen einfacher Tätigkeiten, z.B. Bringen von Gegenständen wie Büchern, Fernbedienung, Arznei, etc.
- Gegenseitiges Anreichen von Gegenständen durch Mensch und Maschine
- Mobilitätshilfe
- Reinigen, z.B. von Böden, Waschbecken oder Wänden und Fenster
- Kommunikation insbesondere durch natürlich empfundene und intuitive Benutzerschnittstellen
- Haustechnik-Management (durch die Anbindung an Informationsnetze)

Der Roboterassistent wird im Haushalt überwiegend in bestehende Umgebungen mit geringen technischen Modifikationen eingesetzt und von ungeschulten, evtl. sogar behinderten Personen bedient oder benutzt werden. Ähnlich wie beim Roboterassistenten im Produktionsszenario ist ein Höchstmaß an Flexibilität der Ausführung unterschiedlichster Aufgaben in teilweise unbekannter Umgebung bei einem Höchstmaß an Bedienkomfort und Sicherheit gefordert.

3 Analyse der Aufgabenstellung und Ableitung von Anforderungen

3.1 Analyse der Szenarien

Die zu lösenden Aufgaben in den Szenarien (Kapitel 2.3.1 und 2.3.2) sind äquivalent bzw. ergänzen einander. Die Eigenschaften der Systeme sind im Detail unterschiedlich, bezüglich des Systemkonzeptes weisen sie jedoch folgende Gemeinsamkeiten auf (vgl. auch Bild 2-1):

- Das Gesamtsystem ist mobil; entweder zur Aufgabenausführung selbst oder zum flexiblen Wechsel des Arbeitsortes.
- Ein Aktuatorsystem (Manipulatoren) ist fähig, für den menschlichen Partner mit einem Werkzeug nutzbare mechanische Leistungen zu erbringen.
- Ein Sensorsystem ist in der Lage, die Umgebung zu erfassen und die Szene im Sinne einer Aufgabe zu deuten.
- Ein Kommunikationssystem dient der Interaktion mit den Menschen.

Darüber hinaus werden von einem Roboterassistenten folgende Eigenschaften erwartet:

- Zielgerichtetes Verhalten.
- Robustes Agieren in menschlicher Umgebung.

Welche Anforderungen die Kontrollarchitektur erfüllen muss, damit der Roboterassistent diesen Erwartungen gerecht wird, wird im Folgenden analysiert:

3.1.1 Mobilität

Der Roboterassistent muss in der Lage sein, selbstständig zu navigieren[1]. Er benötigt dafür eine geeignete Repräsentation der Umwelt sowie ein geeignetes Sensorsystem zur Ortung. Als Anforderung für die Kontrollarchitektur (Abkürzung A) wird formuliert:

A 1: Die Kontrollarchitektur muss Navigationssysteme unterstützen.

[1] Navigation ist die Zusammenfassung aller Tätigkeiten, die für die Bewegungsführung vom Ausgangspunkt zum Zielpunkt erforderlich sind. Dies umfasst insbesondere die Bahnplanung, Bahnführung und -regelung sowie Ortung (Müllerschön 1996).

3.1.2 Manipulation

Der Roboterassistent benötigt die Fähigkeit zu manipulieren. Darunter wird hier die Zusammenfassung aller Tätigkeiten zur zielgerichteten Bewegungsführung eines oder mehrerer Manipulatoren sowie die Benutzung eines Werkzeugs verstanden. Dies sind insbesondere die Bestimmung einer Bewegungstrajektorie und die Bahnregelung auf der Trajektorie.

Beim Hol- und Bringdienst besteht die Manipulation aus dem Greifen und Aufnehmen bzw. Abgeben eines Objektes. Als Werkzeug wird ein geeigneter Greifer benötigt, der geöffnet und geschlossen werden kann.

A 2: **Die Kontrollarchitektur muss die Ansteuerung eines Manipulatorsystems mit Werkzeug unterstützen.**

3.1.3 Umgebungserfassung

Damit der Roboterassistent an verschiedenen Orten arbeiten kann, muss er seine Handlungen auf wechselnde Situationen anpassen. Ein Sensorsystem dient ihm zur Erfassung der Umgebung und zur Analyse der Situation. Die Sensordaten müssen zum Lösen der Aufgabe geeignet interpretiert werden. Die Notwendigkeit der Umgebungserfassung wird in Abschnitt 3.1.6 tiefergehend analysiert.

3.1.4 Kommunikation mit dem Menschen

Ein mobiler Roboter ist umso nützlicher, je besser er mit anderen Agenten und Menschen kommunizieren kann (Konolige, Myers 1998).

A 3: **Die Kontrollarchitektur muss eine Mensch-Maschine-Schnittstelle zur Kommunikation unterstützen.**

Die Kommunikation dient der Interaktion und Kooperation. Diese können sowohl informatorisch als auch physisch ausgeprägt sein. Wichtig für alle Arten der Interaktion ist die Fähigkeit des Roboterassistenten, dass er seinen Interaktionspartner wieder erkennt. Das Verfahren, wie der Interaktionspartner erkannt wird, ist abhängig von der technischen Ausprägung der Sensorsysteme des Roboterassistenten.

Informatorische Interaktion

Bild 3-1 zeigt typische informatorische Kommunikationsarten zwischen Roboterassistenten und Menschen.

Kommunikationsrichtung	Kommunikationsinhalt	Erläuterung
Mensch ↓ Roboterassistent	Befehle	Erteilen von Aufträgen.
	Hilfestellung für den Roboterassistenten	In Situationen, in denen der Roboterassistent nicht weiter „weiß", soll ihm der Mensch helfen können.
	Betriebsmoduswechsel	Der Roboterassistent wird voraussichtlich verschiedene Betriebsmodi haben, zwischen denen der Mensch wählen kann.
Roboterassistent ↓ Mensch	Statusmeldungen	Ausgaben des Roboterassistenten über den Status der Befehlsausführung.
	Störungsmeldungen	Ausgaben des Roboterassistenten in Störsituationen.
	Rückfragen an den Menschen	In Situationen, in denen der Roboterassistent nicht weiter „weiß", soll er den Menschen um Hilfe bitten können.
	Informationsdienste	Auskünfte über interne Daten des Roboterassistenten oder z.B. auch Internetabfragen oder Bedienung der Haustechnik.

Bild 3-1: Informatorische Kommunikation zwischen Roboterassistent und Mensch.

Obwohl der Roboterassistent komplexe Aufgaben ausführen soll, wird eine **intuitive** und **einfache** Bedienung gefordert. Es ist dem Benutzer ist nicht zumutbar, eine Roboterprogrammiersprache erlernen zu müssen. Daraus folgt:

A 4: Der Roboterassistent muss einfache Befehle verstehen und selbstständig in (komplexe) Handlungen umsetzen können.

Bei der natürlichen Kommunikation identifiziert der Mensch Gegenstände in der Regel direkt anhand eines Begriffs. So weiß ein Mensch aufgrund seiner Erziehung, was z.B. eine Tasse ist, für den Roboter hat dieser Begriff zunächst keine Bedeutung. Für eine natürliche Kommunikation mit dem Mensch muss der Roboterassistent also ein Verständnis für dessen Begriffe erlangen. Die menschliche Sprache identifiziert dabei nicht nur individuelle Objekte (z.B. „meine rote Tasse"), sondern auch Klassen von Objekten (z.B. „Tasse").

A 5: Der Roboterassistent muss in der Lage sein, Objekte zu identifizieren und zwar nicht nur Individuen, sondern auch Klassen von Objekten.

Physische Interaktion

Eine physische Interaktion zwischen Mensch und Roboterassistent findet z.B. bei der kooperativen Bearbeitung von Werkstücken und ähnlichen Aufgaben statt. Beim Hol- und Bringdienst gibt es eine physische Interaktion während der gegenseitigen Übergabe von Objekten. Wenn sich Maschine und Mensch den gleichen Arbeitsraum teilen, muss sich der Roboterassistent sicher bewegen und darf den Menschen keiner Gefahr aussetzen. Der Roboterassistent braucht geeignete **Sicherheitssysteme** zur Vermeidung von Gefährdungen für den Menschen. Er muss dabei sehr schnell Bedrohungen erkennen und darauf reagieren.

A 6: Die Kontrollarchitektur muss auch zeitkritische Funktionen unterstützen.

3.1.5 Zielgerichtetes Verhalten

Das Verhalten des Roboterassistenten soll zielgerichtet sein. Stehen verschiedene Handlungsalternativen zur Auswahl, so muss der Roboterassistent die richtige Entscheidung treffen.

A 7: Die Kontrollarchitektur muss eine Komponente zur Entscheidungsfindung beinhalten.

3.1.6 Robustes Agieren in menschlicher Umgebung

Die typische, menschliche Umgebung, in der Roboterassistenten agieren sollen, wird in Bild 3-2 charakterisiert.

Kriterium (Strube 2000)	**Einordnung**	**Erläuterung**
(K 1) Beobachtbarkeit	Partiell zugänglich	In den meisten realistischen Fällen gibt es keine vollständige Zugänglichkeit, weil die Sensoren mit beschränkter Genauigkeit nur einen Ausschnitt der Umwelt wahrnehmen können.
(K 2) Dynamik	Dynamisch	Die Umwelt verändert sich; auch während der Entscheidungsfindung des Roboters.
(K 3) Deterministik	Deterministisch	Aktionen haben deterministische Effekte.
(K 4) Episodik	Episodisch	Die Qualität einer Aktion ist nur von der aktuellen Wahrnehmung-Aktions-Episode abhängig.
(K 5) Feindlichkeit	Nicht feindlich	Es gibt keinen Gegenspieler, der permanent die Umwelt bösartig verändert.
(K 6) Diskret / kontinuierlich	Diskret	Die Welt des Menschen wird mit diskreten Begriffen beschrieben. Der Roboterassistent soll mit den gleichen diskreten Begriffen arbeiten. Nach Möglichkeit werden diskrete Perzepte und Aktionen spezifiziert.

Bild 3-2: Charakterisierung von Einsatzumgebungen für Roboterassistenten.

Aus dieser Charakterisierung werden die folgenden Anforderungen abgeleitet. Aus (K 1) folgt:

A 8: Der Roboterassistent muss mit einer partiellen Umweltbeobachtungsfähigkeit handeln.

Benötigt der Roboterassistent Informationen über die Umwelt außerhalb seines Sensorerfassungsbereiches, so benötigt er dafür Modelle:

A 9: Die Kontrollarchitektur muss über eine interne Repräsentation der Umwelt verfügen.

Diese muss regelmäßig mit der realen Umgebung abgeglichen werden, da sie sich dynamisch verändert (K 2).

A 10: Die interne Repräsentation der Umwelt muss anhand von Sensordaten aktualisiert werden können.

Die Umwelt ist aufgrund ihrer Dynamik (K 2) nicht vorhersehbar. Wird ein Lösungsweg zur Aufgabenausführung geplant, so ist nicht gesichert, dass dieser direkt zum Ziel führt, weil sich

die Umwelt verändert haben kann. Diese Veränderungen können hinderlich sein, z.B. wenn ein Luftstoß eine Tür zuschlägt, oder sie können auch günstig sein, wenn z.B. ein zu holender Gegenstand schon näher auf dem Weg gefunden wird als ursprünglich angenommen. Allgemein wird gefolgert:

A 11: Der Roboterassistent muss in der Lage sein, seine Handlungen an eine sich verändernde Umwelt anzupassen.

Die Dynamik der Umwelt (K 2) ist nicht vorhersehbar; auch nicht die Geschwindigkeit der Veränderung. Dies stellt Anforderungen an den Zeitbedarf bei der Entscheidungsfindung und Ausführung von Aktionen.

A 12: Der Roboterassistent muss seine Handlungen in Echtzeit kontrollieren.

Das ermöglicht ihm, auf plötzliche Veränderungen (Gefahren) zu reagieren und z.B. Hindernissen während einer Bewegung auszuweichen.

Die Annahme, dass es keinen Gegenspieler gibt, der die Umwelt bösartig verändert (K 5), und die Aussage, dass Aktionen deterministische Effekte haben (K 3), ermöglichen die Formulierung von diskreten Aktionen mit vorhersehbaren Ergebnissen. Um ein robustes Verhalten zu sichern, müssen die Aktionen dennoch überwacht werden:

A 13: Die Aktionen des Roboterassistenten müssen auf Erfolg überwacht werden.

Daraus wird weiter abgeleitet:

A 14: Jede Aktion muss ein definiertes Ende haben, wenn sie erfolgreich ausgeführt worden ist.

Ist eine Aktion nicht erfolgreich gewesen, so müssen Gegenmaßnahmen eingeleitet werden.

A 15: Die Kontrollarchitektur muss Methoden zur Störungsbehandlung unterstützen.

3.2 Analyse von Hol- und Bringdiensten

3.2.1 Zusammensetzung aus Handlungsschritten

Der Hol- und Bringdienst repräsentiert grundlegend die Tätigkeiten eines Roboterassistenten mit dem bereits genannten Schema „Ortswechsel – Tätigkeit – Ortswechsel“. Bei diesem speziellen

Dienst ist die Tätigkeit das Greifen und Aufnehmen bzw. das Abgeben eines Objektes. Die notwendigen Handlungsschritte und möglichen Störquellen werden im Folgenden analysiert. Bild 3-3 zeigt den einfachen Hol- und Bringdienst genau eines Objektes.

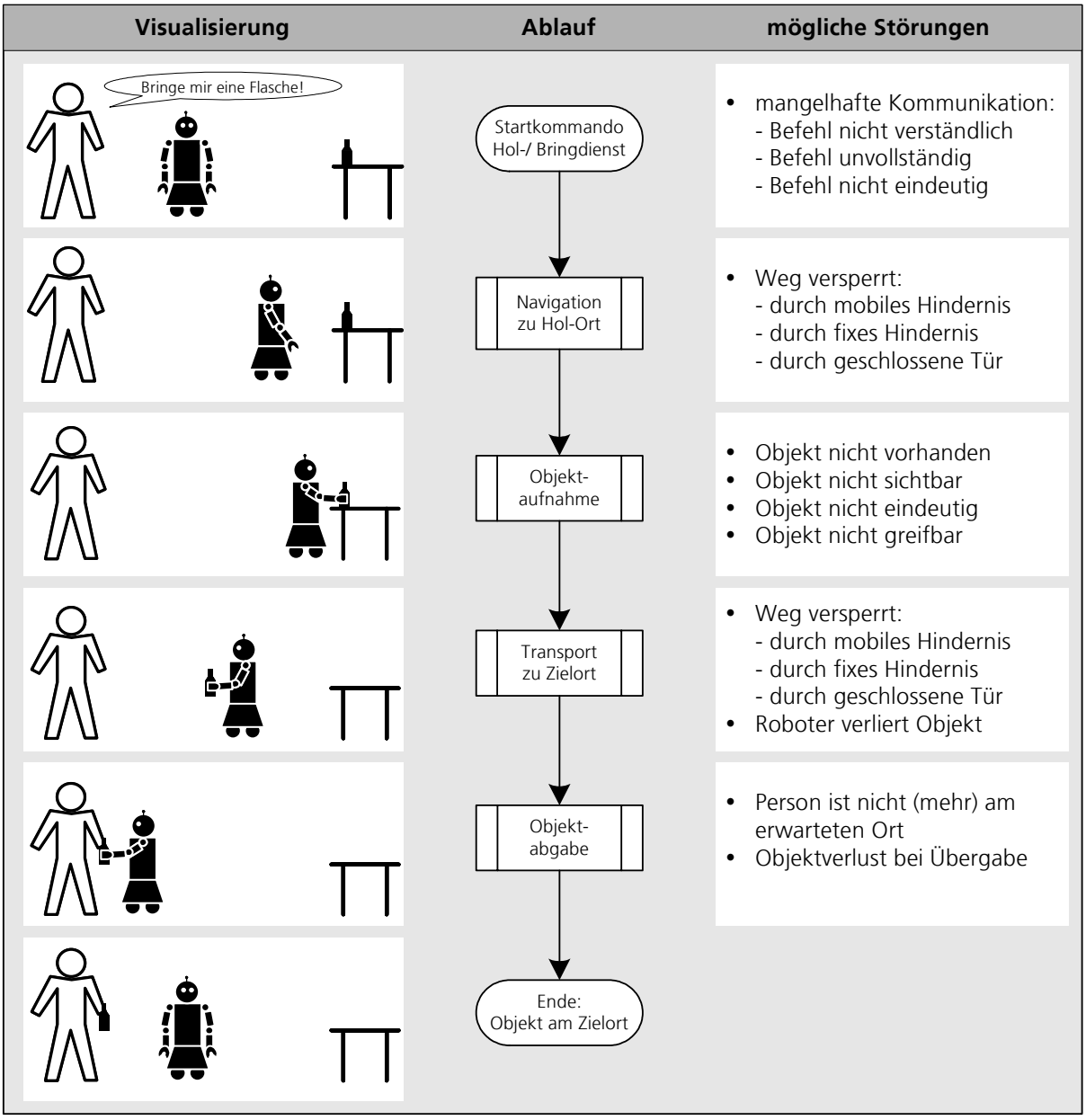


Bild 3-3: Teilschritte eines einfachen Hol- und Bringdienstes und mögliche Störungen.

Beim einfachen Hol- und Bringdienst führt der Roboterassistent gemäß Bild 3-3 vier Teilschritte aus: Navigation, Objektaufnahme, Transport und Objektabgabe. Diese werden in Bild 3-4 weiter detailliert.

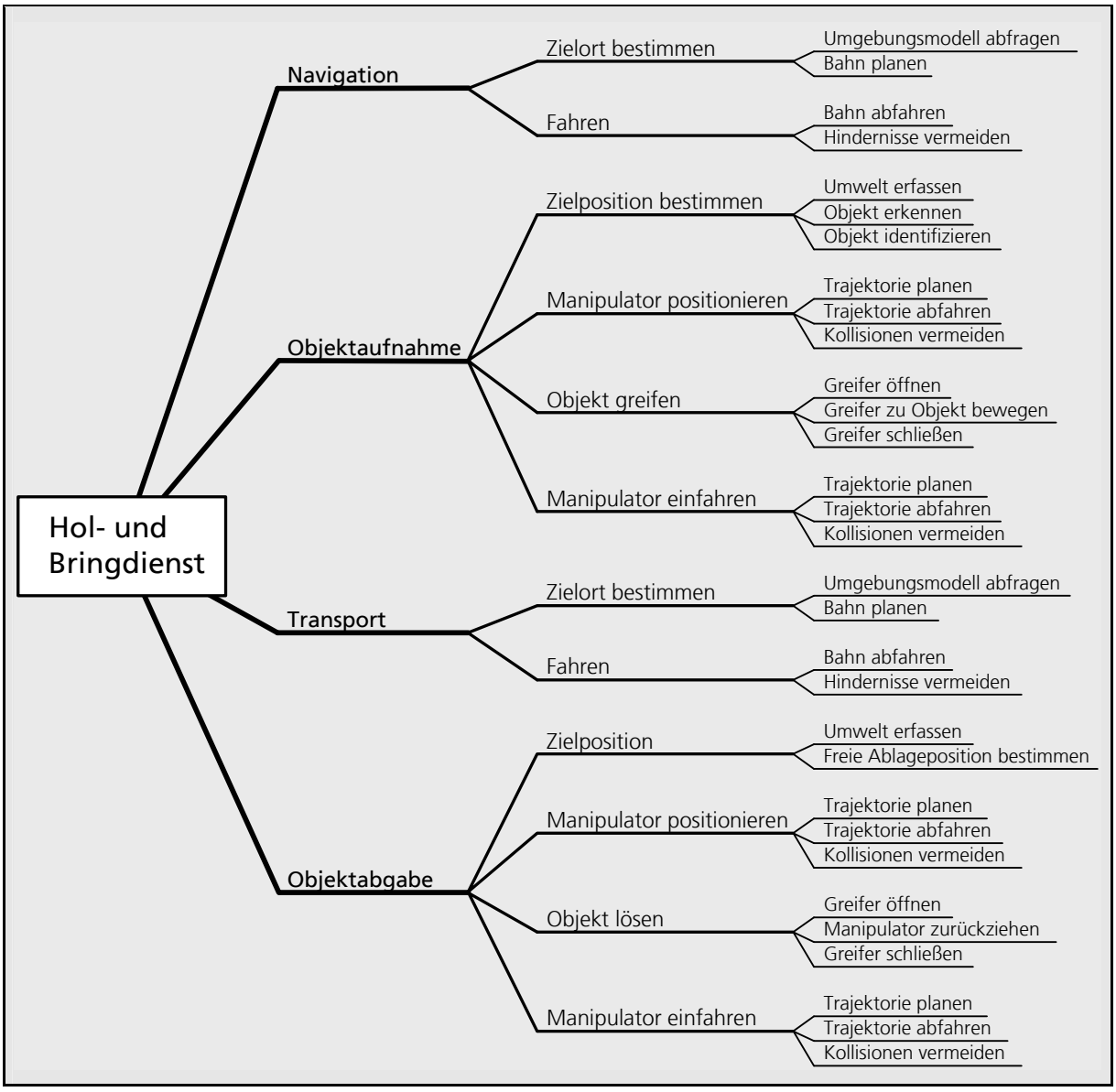


Bild 3-4: Funktionale Analyse der Teilschritte des einfachen Hol- und Bringdienstes.

Die rechte Spalte in Bild 3-4 enthält 31 Teilschritte, jedoch nur 15 verschiedene. Die Aufgabenausführung besteht also aus wiederkehrenden Teilschritten. Diese Eigenschaft soll im Konzept für die Kontrollarchitektur genutzt werden:

A 16: Die Handlungen sollen aus kombinierbaren Teilschritten gestaltet sein.

Für einen möglichst universellen Einsatz der Kontrollarchitektur soll es auch zukünftig möglich sein, das System zu erweitern:

A 17: Das Spektrum der Fähigkeiten soll erweiterbar sein.

Die Handlungsschritte für den Hol- und Bringdienst werden in Bild 3-5 weiter analysiert:

Handlungsschritt	Komponente	Effektor	Zeitverhalten	
Umgebungsmodell abfragen	Umgebungserfassung	Umgebungsmodell	SQ	E
Umwelt (in 3-D) erfassen	Umgebungserfassung	Umgebungssensorik	SQ	E
Objekt erkennen	Umgebungserfassung	Umgebungssensorik	SQ	E
Objekt identifizieren	Umgebungserfassung	Umgebungsmodell	SQ	E
Freie Ablageposition bestimmen	Umgebungserfassung	Umgebungsmodell	SQ	E
Bahn planen	Mobilität	Plattform	SQ	E
Bahn abfahren	Mobilität	Plattform	NL	E
Hindernisse vermeiden	Mobilität	Plattform	NL	K
Trajektorie planen	Manipulation	Manipulator	SQ	E
Trajektorie abfahren	Manipulation	Manipulator	NL	E
Kollisionen vermeiden	Manipulation	Manipulator	NL	K
Greifer um Objekt bewegen	Manipulation	Manipulator	NL	E
Manipulator zurückziehen	Manipulation	Manipulator	NL	E
Greifer öffnen	Manipulation	Greifer	NL	E
Greifer schließen	Manipulation	Greifer	NL	E
Legende: SQ: Sequentiell	NL: Nebenläufig	E: Endlich	K: Kontinuierlich	

Bild 3-5: Analyse der Handlungsschritte für den Hol- und Bringdienst.

Wichtig zu bemerken ist, dass einige der Handlungsschritte nebenläufig zueinander aktiv sein müssen, z.B. das „Kollisionen Vermeiden“ während des „Trajektorie Abfahrens“.

3.2.2 Störquellen

Während der Ausführung können verschiedene Störungen auftreten, so dass die gestellte Aufgabe vom Roboterassistenten nicht gelöst werden kann. Bild 3-3 zeigt in der rechten Spalte mögliche Störquellen. Dies bekräftigt Anforderung A 13, dass die einzelnen Handlungsschritte zum Erreichen eines robusten Verhaltens auf Erfolg überprüft und Routinen zur Störungsbehandlung eingeleitet werden müssen.

3.2.3 Komplexität von kombinierten Hol- und Bringdiensten

Der Roboterassistent soll nicht nur einfache Hol- und Bringdienste wie in Bild 3-3, sondern auch Verschachtelungen wie beim Holen *mehrerer* Gegenstände ausführen. Im Allgemeinen entspricht dies der gleichzeitigen Beauftragung mehrerer Aufgaben, die der Roboterassistent selbstständig plant und verschachtelt ausführt.

Ist n die Anzahl der Objekte und l die Anzahl der möglichen diskreten Orte und gilt $n < l$ und kann sich an einem Ort maximal ein Objekt befinden, so berechnet sich die Anzahl der möglichen Kombinationen K von Objekten an Orten zu:

$$K = \frac{l!}{(l-n)!} \qquad \textbf{(3-1)}$$

Soll beispielsweise der Tisch für sechs Personen zum Kaffee gedeckt werden, so müssen 34 Objekte transportiert werden, für die es 34 Positionen im Schrank, 34 Positionen auf dem Tisch und 34 Positionen in der Spülmaschine gibt. Die Anzahl der möglichen Kombinationen beträgt dann $K = \frac{102!}{(102-34)!} \approx 3{,}877 \cdot 10^{65}$. Holt der Roboterassistent jeweils nur einen Gegenstand, so muss er 34 mal in die Küche fahren. Tatsächlich ist es erstrebenswert, dass die Wege optimiert werden und der Roboter ähnlich wie der Mensch versucht, alle Teller zu stapeln und auf einmal zu transportieren, anschließend alle Tassen u.s.w. Folgende Anforderung lässt sich ableiten:

A 18: Die Kontrollarchitektur muss eine verschachtelte Aufgabenausführung ermöglichen.

Ist das Stapeln der Objekte erlaubt, d.h. können sich theoretisch an einem Ort alle Objekte auf einmal befinden, so gilt für die maximale Anzahl möglicher Kombinationen K_{max}:

$$K_{max} = l^n \qquad \textbf{(3-2)}$$

Im Beispiel ist $K_{max} = 102^{34} \approx 1{,}961 \cdot 10^{68}$ die oberste Schranke.

Diese große Zahl rührt daher, dass jedes einzelne Objekt als Individuum betrachtet wird. Im Beispiel mit dem Kaffeegedeck soll aber *ein* Teller an *einem* Platz liegen. Es ist unerheblich, welcher Teller an welchem Platz liegt. Es lässt sich folgende Anforderung ableiten:

A 19: Der Roboterassistent soll mit Klassen von Objekten planen können, ohne bestimmte Elemente identifizieren zu müssen.

3.3 Strukturierung der Anforderungen

In Bild 3-6 werden die gesammelten Anforderungen strukturiert und in übergeordnete Anforderungen Ã1 bis Ã5 zusammengefasst:

Ã1	Modularität und Erweiterungsfähigkeit.	
	Die Handlungen sollen aus kombinierbaren Teilschritten gestaltet sein.	(A 16)
	Das Spektrum der Fähigkeiten soll erweiterbar sein.	(A 17)
Ã2	Schnittstellen zur Robotersteuerung.	
	Die Kontrollarchitektur muss Navigationssysteme unterstützen.	(A 1)
	Die Kontrollarchitektur muss die Ansteuerung eines Manipulatorsystems mit Werkzeug unterstützen.	(A 2)
	Die Kontrollarchitektur muss eine Mensch-Maschine-Schnittstelle zur Kommunikation unterstützen.	(A 3)
	Die Kontrollarchitektur muss auch zeitkritische Funktionen unterstützen.	(A 6)
	Der Roboterassistent muss seine Handlungen in Echtzeit kontrollieren.	(A 12)
Ã3	Explizite Umweltrepräsentation.	
	Der Roboterassistent muss mit einer partiellen Umweltbeobachtungsfähigkeit handeln.	(A 8)
	Die Kontrollarchitektur muss über eine interne Repräsentation der Umwelt verfügen.	(A 9)
	Die interne Repräsentation der Umwelt muss anhand von Sensordaten aktualisiert werden können.	(A 10)
	Der Roboterassistent muss in der Lage sein, Objekte zu identifizieren und zwar nicht nur Individuen, sondern auch Klassen von Objekten.	(A 5)
Ã4	Adaptierende Verhaltenskontrolle.	
	Der Roboterassistent muss in der Lage sein, seine Handlungen an eine sich verändernde Umwelt anzupassen.	(A 11)
	Die Aktionen des Roboterassistenten müssen auf Erfolg überwacht werden.	(A 13)
	Jede Aktion muss ein definiertes Ende haben, wenn sie erfolgreich ausgeführt worden ist.	(A 14)
	Die Kontrollarchitektur muss Methoden zur Störungsbehandlung unterstützen.	(A 15)
Ã5	Planungskomponente.	
	Die Kontrollarchitektur muss eine Komponente zur Entscheidungsfindung beinhalten.	(A 7)
	Der Roboterassistent muss einfache Befehle verstehen und selbstständig in (komplexe) Handlungen umsetzen können.	(A 4)
	Der Roboterassistent soll mit Klassen von Objekten planen können, ohne bestimmte Elemente identifizieren zu müssen.	(A 19)
	Die Kontrollarchitektur muss eine verschachtelte Aufgabenausführung ermöglichen.	(A 18)

Bild 3-6: Anforderungen für das Gesamtsystem Kontrollarchitektur.

4 Stand der Technik

4.1 Roboterassistenten

Es gibt derzeit nur sehr wenige Serviceroboter, die der Definition für Roboterassistenten gerecht werden. Sie stammen aus einzelnen Forschungslabors und sind nicht kommerziell erhältlich. Bild 4-1 zeigt ausgewählte Beispiele. Ergänzend sind bekannte Versuchsplattformen ohne Manipulatoren abgebildet.

Versuchsplattformen mit Manipulator(en)		Versuchsplattformen ohne Manipulator
 HERMES (Graefe, Bischoff 2003)	 MobMan (Wichert et al. 2002)	 Care-O-bot® (Schaeffer, May 1999)
 MovAid (Dario et al. 1997)	 rob@work (Helms et al. 2002)	 Nursebot (Roy et al. 2000)

Bild 4-1: Ausgewählte Beispiele für Serviceroboter aus Forschungslabors.

Kommerzielle Systeme (Bild 4-2) verfügen bisher noch nicht über die Fähigkeit selbstständig zu manipulieren, sondern fokussieren auf Kommunikationsfunktionen; auch ASIMO und Wakamaru, die zwar beide Arme haben, können nicht selbstständig manipulieren.

Serviceroboter mit Armen		Serviceroboter ohne Arme
		
ASIMO (Honda 2004)	Wakamaru (Mitsubishi 2004)	PaPeRo (NEC 2004)

Bild 4-2: Ausgewählte Beispiele für kommerzielle Serviceroboter.

4.2 Robotersteuerungen von Servicerobotern

Die Robotersteuerung stellt die Basisfunktionalitäten des Roboterassistenten zur Verfügung. Sie ist die Schnittstelle zur Hardware. Die Funktionen einer Servicerobotersteuerung können in **Mobilitätsfunktionen**, welche die Mobilität des Roboters sicherstellen, und **Anwendungsfunktionen**, welche die Anwendungsmodule steuern, aufgeteilt werden (Müllerschön 1996). Zur Bereitstellung von Mensch-Maschine- und Maschine-Maschine-Schnittstellen (Schraft, Volz 1996) sowie für den internen Datenaustausch werden außerdem verschiedene **Kommunikationsfunktionen** verwendet. Eine entsprechende funktionale Gliederung für Steuerungen mobiler Serviceroboter ist in Bild 4-3 dargestellt.

Die verschiedenen Funktionen einer Servicerobotersteuerung sind teilweise nebenläufig und werden daher im allgemeinen Fall auf mehrere Threads[2] und Prozesse[3] aufgeteilt, welche die primären Subsysteme der Steuerungssoftware darstellen. Threads und Prozesse können entweder zyklisch, ereignisgesteuert oder durch eine Kombination beider Mechanismen aktiviert werden (Borelly et al. 1998).

[2] Ein Thread ist eine Folge von Anweisungen (Teilprogramm), die von einem Prozessor parallel zu anderen Threads und Prozessen abgearbeitet werden kann (Traub 2002).

[3] Ein Prozess ist ein selbstständig ausführbares Programm, das aus einem oder mehreren nebenläufigen Threads besteht (Traub 2002).

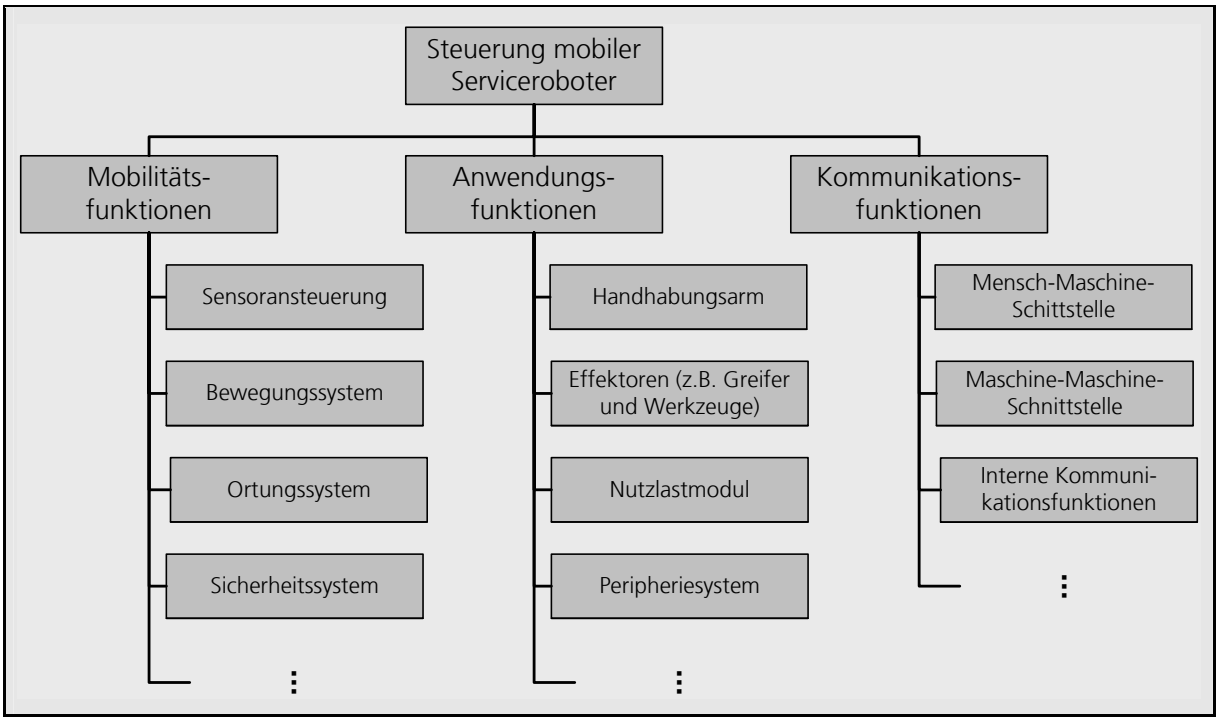


Bild 4-3: Funktionale Gliederung der Steuerung mobiler Serviceroboter (Traub 2002).

4.3 Kontrollarchitekturen für mobile Agenten

Ziel der Architekturentwicklung ist es, mit ihr eine **Intelligenz** im Sinne eines zielgerichteten, autonomen Verhaltens aufzubauen. Diese Intelligenz setzt sich aus einer Sammlung von Wissen und von Methoden zusammen. Es wird im Folgenden eine Übersicht über die größten Trends für Roboterarchitekturen in der Forschung gegeben: sequentielle Schichtenarchitektur, verhaltensbasierte (engl. *behavior-based*) Architekturen, Mehrebenen-Architekturen und die BDI-Theorie in ihrer Umsetzung PRS. Ergänzend wird der Ansatz des Konnektionismus erläutert.

4.3.1 Sequentielle Schichtenarchitektur

Bild 4-4 zeigt die traditionelle, sequentielle Schichtenarchitektur, die sowohl in der Psychologie als auch im Forschungsgebiet der Künstlichen Intelligenz verwendet wird. Die Intelligenz besteht bei diesem Ansatz darin, die wahrgenommenen Sensorwerte in ein Modell zu wandeln, mit einem Kognitionsprozess beabsichtigte Aufgaben zu formulieren und diese über eine Motoransteuerung auszuführen. Diese Vorgehensweise wird auch *sense-plan-act* genannt.

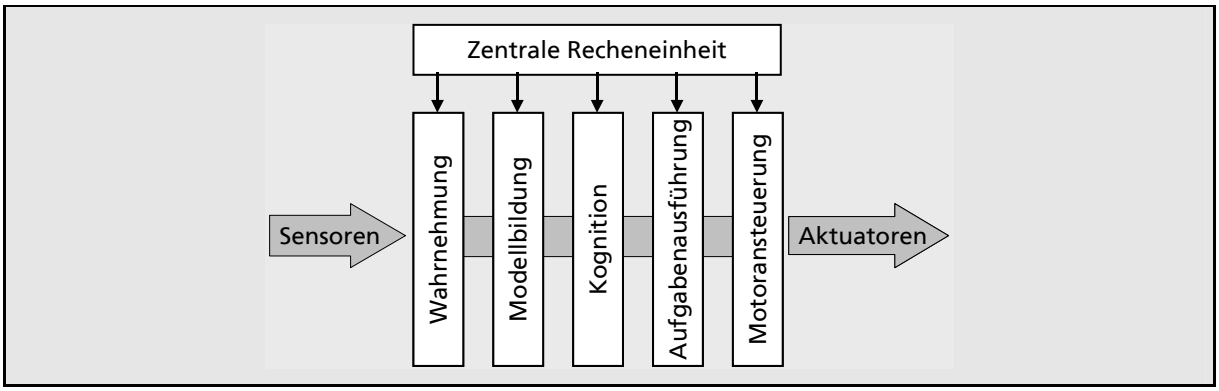


Bild 4-4: Sequentielle Schichtenarchitektur (*sense-plan-act*).

Dieser Ansatz ist auf den ersten Blick ausreichend generalistisch, enthält jedoch ein paar Annahmen, von denen man heute weiß, dass sie nicht korrekt sind: Es wird angenommen, dass Wahrnehmung und Ausführung vom Kognitionsprozess getrennt werden können. Dies ist nicht richtig, weil Wahrnehmung sehr häufig mit Erwartungen, was man wahrnehmen möchte, und mit Kontextwissen verbunden ist.

Das berühmteste Beispiel für diese Architektur war der Roboter *Shakey* (Nilsson 1984), der basierend auf den Sensordaten ein Modell der Umgebung mit einer logikbasierten Repräsentation aufbaute, damit symbolisch plante und die Pläne anschließend ausführte. Aufgrund der benötigten Rechenzeit stand der Roboter lange still, ehe er begann zu handeln.

4.3.2 Verhaltensbasierte Architekturen

Pionier der verhaltensbasierten Architekturen ist Rodney A. Brooks aufgrund seiner Entwicklung der so genannten **subsumption architecture** (Brooks 1986). Ihre Struktur ist in Bild 4-5 dargestellt. Brooks propagierte, einzelne, unabhängige Module, die jeweils ein spezifisches Verhalten erzeugen, zu schichten. Die einzelnen Verhaltensweisen sind nach ihrer Verhaltenskomplexität geschichtet, so dass mit steigender Schichthöhe die Fähigkeiten erweitert werden. Die jeweils höhere Verhaltensschicht kann die Ausgaben der darunter liegenden unterdrücken und Eingaben modifizieren. Dabei wird jedes Modul durch einen endlichen Automaten realisiert, der auf Sensoreingaben reagiert und Aktuatorsignale ausgibt. Gemäß Brooks' *Physical Grounding Hypothesis* (Brooks 1990) benötigt diese Architektur keine symbolische Repräsentation der Umwelt, sondern das Verhalten wird anhand der aktuell wahrgenommenen Sensordaten gesteuert.

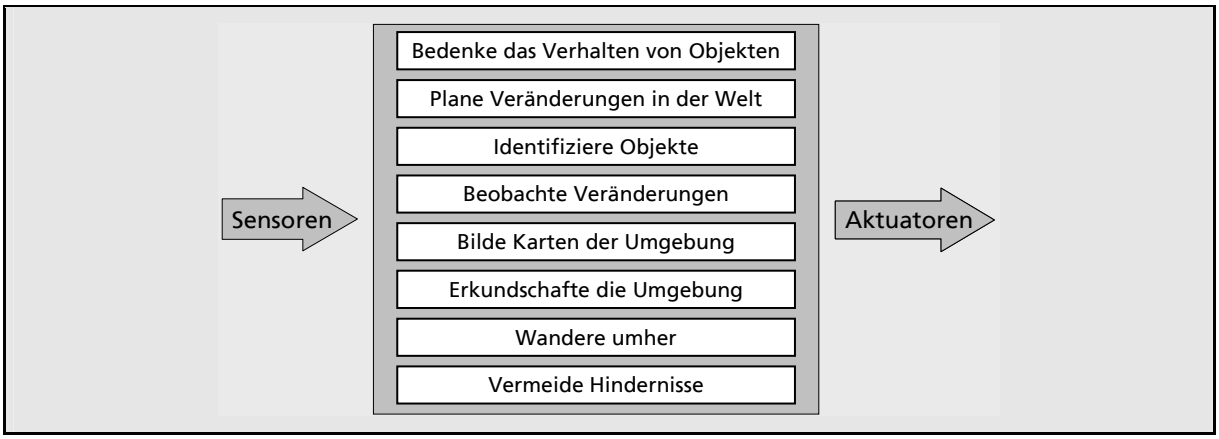


Bild 4-5: Subsumption-Architektur (Brooks 1986).

Roboter mit dieser Architektur agieren nach einem Verhaltensmuster auf der obersten Ebene und richten sich dabei reaktiv nach Sensordaten. Sie sind jedoch nicht in der Lage, komplexe Handlungen zu planen. Der Vorteil verhaltensbasierter Architekturen liegt in der Reaktivität und damit in der Möglichkeit opportunistisch auf Unvorhersehbares zu reagieren.

Beispiele von verhaltensbasierten Architekturen sind von Mataric (Mataric 1997) und von Arkin (Arkin 1998) beschrieben worden. Von den vielen durch die *subsumption*-Architektur inspirierten Entwicklungen sei Maes' **agent network architecture** (Maes 1991) hervorgehoben. Der größte Unterschied zwischen Maes' Architektur und Brooks' *subsumption*-Architektur ist ihre Forderung, dass Roboter mehrere, beeinflussbare Ziele verfolgen können müssen (Maes 1990). Mit ihrer Architektur werden die Ziele mit Hilfe der Gewichtung einzelner Aktionsmaße auch in einer vorgegebenen Reihenfolge erreicht. Tyrrell propagiert alternativ eine Methode zur aktiven Auswahl der Tätigkeit (*action selection*) (Tyrrell 1993). Blumberg greift die Prinzipien von Maes und Tyrrell auf: in seiner Lösung sind die Verhalten (*behavior*) hierarchisch angeordnet, können aber parallel aktiviert werden (Blumberg 1997). Auch in der **Hierarchical Abstract Behavior Architecture** (Nicolescu, Mataric 2002) werden die Verhaltensregeln hierarchisch angeordnet.

4.3.3 Mehrebenen-Architekturen

In so genannten hybriden Architekturen werden die Ansätze mehrerer Architekturtypen so verknüpft, dass sie sich gegenseitig ergänzen, um auf diese Weise deren Vorzüge miteinander zu kombinieren. Beispielsweise soll die Verbindung von reaktiven und deliberativen Methoden zielgerichtetes Verhalten flexibel und fehlertolerant auch in dynamischen Umgebungen ermöglichen. Die Integration der verschiedenen Ansätze ist jedoch nicht trivial, weil sie auf unterschiedlichen und zum Teil inkompatiblen Prinzipien beruhen (Sesseler 2002). Sie geschieht

meist auf Systemebene, indem auf verschiedenen Ansätzen basierende Teilsysteme kombiniert und untereinander koordiniert werden. Die Teilsysteme werden in Ebenen betrachtet.

Ein Überblick über eine Auswahl von veröffentlichten 2- und 3-Ebenen-Architekturen ist in (Kortenkamp et al. 1998) gegeben. Die derzeit wohl bekannteste 3-Ebenen-Architektur ist **3T** [4] (Bonasso et al. 1997) (Bild 4-6). Die Steuerung auf unterster Ebene besteht aus einer Menge von so genannten *Skills* (Basisfähigkeiten) mit kontinuierlichen Regelungsalgorithmen, die auf der mittleren Ebene über ein System aus so genannten *reactive action packages* (**RAP**) (Firby 1989) aktiviert werden. RAP ist eine in Lisp programmierte Architektur zur Integration reaktiver, flexibler, situationsabhängiger Programme, der *Skills*. 3T integriert in dieser Architektur einen symbolischen Planer auf oberster Ebene. Die Pläne werden in der mittleren Ebene vom RAP-Interpreter verfeinert, ausgeführt und überwacht.

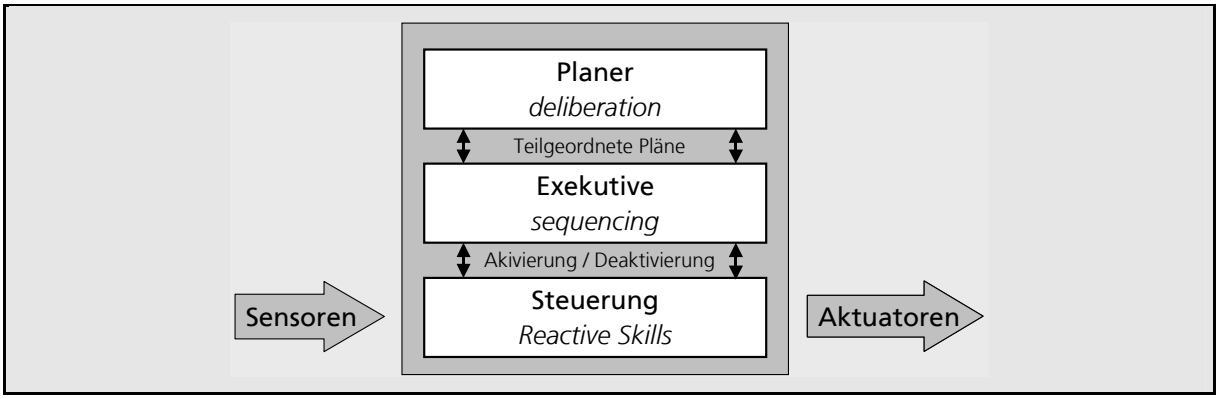


Bild 4-6: 3T-Architektur (Bonasso et al. 1997).

ATLANTIS (*A Three-Layer Architecture for Navigating Through Intricate Situations*) (Gat 1991) ist eine 3-Ebenen-Architektur, deren Steuerungsfokus im Gegensatz zu 3T in der mittleren Ebene liegt, d.h. der Roboter ist verhaltensorientiert und ruft die obere Planungsebene bei Bedarf auf. Typischer ist die Variante mit der obersten Ebene als Hauptsteuerung. Der Einsatz von ATLANTIS beschränkt sich auf die Navigation durch unbekanntes Terrain.

Die **Task Control Architecture** *(TCA)* von Simmons (Simmons 1994) entsprach ursprünglich einzig einer mittleren Ebene ohne eine klare Ebenen-Struktur in sich. Sie organisiert Prozessmodule um einen zentralen Controller, der die Interaktion zwischen den Modulen koordiniert. Mit der Erweiterung des Systems um die *Task Description Language (TDL)* (Simmons, Apfelbaum

[4] Die Bezeichnung 3T kommt von „*three interacting layers or tiers*" (Bonasso et al. 1997).

1998) wurde die 3-Ebenen-Architektur etabliert. Die TDL ermöglicht eine Zielbeschreibung mit Untergliederung in Teilziele, die von den einzelnen Modulen erreicht werden sollen.

4.3.4 BDI-Theorie und PRS-Architekturen

Nach der BDI-Theorie wird der Zustand eines Agenten mit den Einstellungen „Überzeugung" (*belief,* **B**), „Motivation" (*desire,* **D**) und „Absicht" (*intention,* **I**) beschrieben. Die Attraktivität dieses Ansatzes stammt in erster Linie aus der Formalisierung des Zusammenhangs zwischen diesen Einstellungen, mit deren Hilfe sich die Eigenschaften von Agenten definieren und analysieren lassen. Im Gegensatz zu klassischen Ansätzen von Wissen und Zielen in der KI werden in der BDI-Theorie die Überzeugungen als unvollständig und unsicher aufgefasst und bei den Motivationen vielfältige und auch inkompatible Zielsetzungen zugelassen.

Das **Procedural Reasoning System** (**PRS**) (Georgeff, Lansky 1987) orientiert sich konzeptionell an der BDI-Theorie. Dem BDI-Ansatz gemäß wird das Wissen eines Agenten in eine Datenbank mit Fakten (*beliefs*), in Ziele (*desires*), in Pläne (*knowledge area*, KA) und in Absichten (*intentions*) aufgeteilt (Bild 4-7). Ein zentraler Interpreter verwendet dieses Wissen, indem er anhand von Fakten und Zielen Pläne instantiiert und dann als Absichten aufstellt. Die Interaktion mit der Umwelt besteht in der Veränderung der Welt durch Ausführen von Absichten und in der Aufnahme von Informationen über die Welt in Form neuer Fakten.

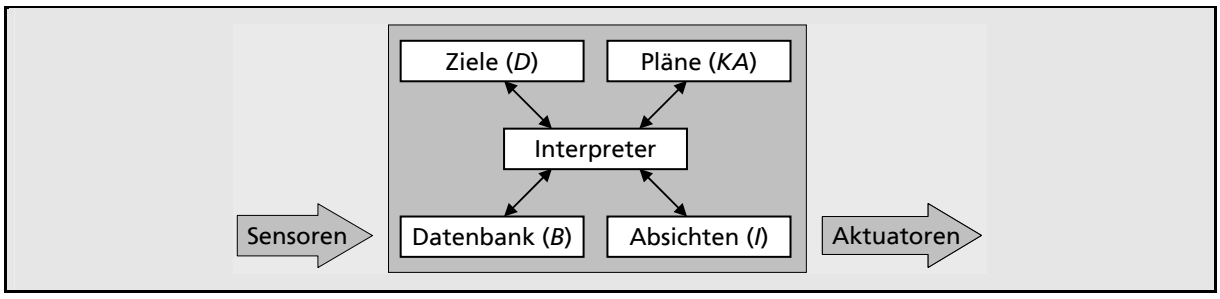


Bild 4-7: Kontrollarchitektur von PRS (Georgeff, Lansky 1987).

Alami et al. nutzen PRS in einer Mehrebenen-Architektur als mittlere Ebene (Alami et al. 1998). Diese Architektur umfasst die folgenden Komponenten und Ebenen: einen Planer *IxTeT*, den Abwickler *PRS* zur Verfeinerung und Überwachung der Aufgaben und eine funktionale Ebene *Kheops* für die reaktive Steuerung.

Aus PRS hat sich die **Saphira**-Architektur (Konolige, Myers 1998) entwickelt (Bild 4-8). Zentrales Element von Saphira ist der so genannte *local perceptual space (LPS)*, ein Weltmodell, das sowohl geometrische als auch symbolische Umgebungsinformationen ineinander vereint. Über die Interpretation der Sensorsignale mit steigendem Abstraktionsgrad wird dieses Weltmodell kontinuierlich aktualisiert. Verhaltensmodule erzeugen basierend auf diesem Modell die Steuersignale für die Aktuatoren.

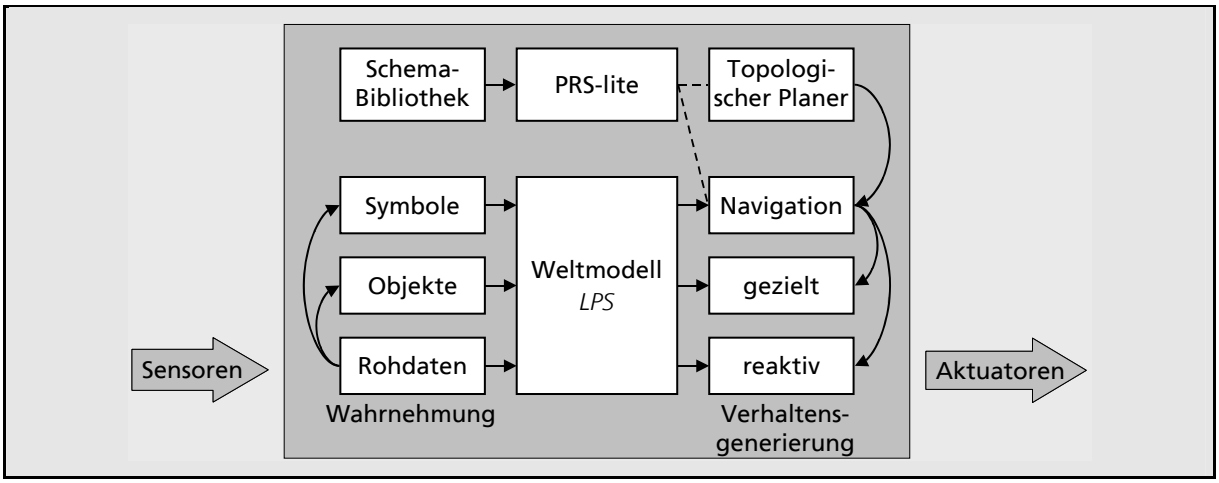


Bild 4-8: Saphira-Architektur (Konolige, Myers 1998).

Die Übergeordnete Aufgabenplanung wird von **PRS-lite** übernommen. Ein symbolischer Planer ist nicht vorgesehen. Heutzutage ist PRS-lite, das noch in Lisp programmiert war, durch **COLBERT** (Konolige 1997) ersetzt, das in C++ programmiert ist. Die Software wird kommerziell mit dem Roboter *Pioneer* (ActivMedia 2002) vertrieben.

JAM (Huber 1999) ist die bislang jüngste Implementierung von PRS und erfolgte in Java. Sie besitzt eine etwas von PRS abweichende Kontrollarchitektur, bei der die Komponenten für Ziele und Absichten in eine zusammengefasst wurden. Außerdem wurde die Plansprache um prozedurale Kontrollbefehle erweitert. Ferner sind Ziele und Pläne um eine Nutzenberechnungsfunktion ergänzt, über die der Interpreter an den Entscheidungspunkten die Auswahl vornehmen kann.

4.3.5 Konnektionismus

Konnektionistische Architekturen bestehen aus Netzstrukturen vieler gleichartiger Einheiten, die sehr einfach aufgebaut und im Gegensatz zu den zuvor beschriebenen, modularen Architek-

turen auf keine bestimmte Aufgabe spezialisiert sind. Die Funktionalität des Systems ergibt sich dabei aus der komplexen Struktur der Verbindungen zwischen diesen Einheiten. Die Repräsentationen von Wissen sind im Gegensatz zu symbolischen Systemen nicht klar abgegrenzt, sondern verteilt und einander überlagernd in den Verbindungen zwischen den Einheiten enthalten. Die Programmierung erfolgt nicht direkt, sondern durch das Lernen von Gewichtungen anhand von Beispielen.

Konnektionistische Ansätze und symbolische Ansätze sind nicht prinzipiell unvereinbar, werden jedoch weitgehend unabhängig voneinander erforscht und angewendet. Nur wenige Systeme kombinieren diese Verfahren, wobei sie jedoch meistens in getrennten Subsystemen ohne konzeptionelle Integration eingesetzt werden (Sesseler 2002).

4.3.6 Ableitung von Anforderungen

Allen vorgestellten Roboterkontrollarchitekturen ist das Ziel gemeinsam, die **Komplexität** bei der Handlungskontrolle zu beherrschen. Dies wird durch eine **Modularisierung**, d.h. durch funktionales Gliedern in Teilsysteme und Definition einfacher Schnittstellen, erreicht.

A 20: Die Roboterkontrollarchitektur muss modular aufgebaut sein.

Dies ergänzt die Anforderung Ã1.

Ziel der verschiedenen Architekturen ist immer eine **Spezifikation von Verhalten**: Die Agenten sollen selbstständig komplexe Aufgaben bearbeiten. Die richtige Wahl der Kontrollarchitektur erlaubt einerseits die Spezifikation des Verhaltens auf einer hohen Abstraktionsebene zur Entwurfszeit und andererseits eine autonome und flexible Kontrolle zur Laufzeit. **Abstraktion** kann dabei ein leistungsfähiges Werkzeug sein, um Wege aus unvorhergesehenen Situationen zu planen (Konolige, Myers 1998).

Für die Spezifikation und Kontrolle des Verhaltens autonomer Roboterassistenten überwiegen die Vorteile des **Symbolansatzes** im Vergleich zum Konnektionismus: Für Roboterassistenten mit den Anforderungen aus Kapitel 3 ist es wichtig, das Verhalten möglichst klar und explizit spezifizieren zu können, um deren Komplexität beherrschbar zu machen. Dazu ist die Abstraktionsebene der formalen Logik des Symbolansatzes besonders geeignet. Die formale Logik bietet ferner die Möglichkeit durch neue Symbole das System später zu **erweitern** und **flexibel** anzupassen (vgl. Anforderung A 17). Deshalb werden im Folgenden zur Verhaltenssteuerung nur Techniken der Symbolmanipulation näher betrachtet.

4.4 Kognitive Systeme

Kognitive Prozesse sind die Grundlage für intelligentes Verhalten. Sie berücksichtigen gespeichertes Wissen und intervenieren zwischen Wahrnehmen und Handeln (Strube 1995, S. 299f). Kognitive Systeme zur Verhaltenssteuerung bedienen sich dabei Methoden aus dem Fachgebiet der Künstlichen Intelligenz. Drei Komponenten werden immer benötigt (Geffner 2000):

- **Mathematische Modelle**, um Probleme zu klassifizieren,
- eine **Repräsentationssprache**, um Probleme zu beschreiben, und
- **Algorithmen**, um Probleme zu lösen.

Dabei haben sich drei Methoden besonders hervorgehoben (Geffner 2000), die in den folgenden Unterkapiteln kurz dargestellt werden:

- **Problemlösen** mit Zustandsmodellen und Suchalgorithmen,
- **Planen** mit einfachen Aktionsmodellen und
- **Entscheiden** unter Berücksichtigung von Unsicherheiten und Feedback aus der Umwelt.

Darüber hinaus existieren noch spezielle, domänenabhängige Systeme zur Verhaltenssteuerung, die aufgrund ihrer Systemabhängigkeit hier nicht weiter betrachtet werden.

4.4.1 Problemlösen

Beim Problemlösen dienen **Zustandsmodelle** der mathematischen Modellierung. Diese bestehen aus einem endlichen, diskreten Zustandsraum Z, aus einem Ausgangszustand $z_0 \in Z$, aus einer Menge von zulässigen Zielzuständen $Z_z \subseteq Z$, aus Operationen $O(z) \subseteq O$, die in jedem Zustand $z \in Z$ anwendbar sind, aus einer Überführungsfunktion $\delta(o,z)$ für $z \in Z$ und $o \in O(z)$ und aus einer Kostenfunktion $\kappa(o,z) > 0$. Eine Lösung für dieses Modell ist eine Folge von anwendbaren Operationen o_i $(i = 0,\ldots,n)$, die den Ausgangszustand z_0 in einen Zielzustand $z_z \in Z_z$ überführt. Optimale Lösungen minimieren dabei die Kosten $\sum_{i=0}^{n} \kappa(o_i, z_i)$.

Suchalgorithmen explorieren jeden einzelnen Zustand und versuchen, einen (optimalen) Pfad von z_0 zu Z_z zu finden. Da der Zustandsraum sehr umfangreich sein kann, benötigt man Suchstrategien, die möglichst schnell eine oder die optimale Lösung finden. Eine blinde Suche ist sehr ineffektiv, da die Anzahl der Zustände mit der Suchtiefe exponentiell wächst. Es werden deshalb Algorithmen mit Heuristiken oder mit Methoden zur Reduzierung von Redundanzen

eingesetzt. Bei ganz speziellen Problemen konnten mit optimierten Algorithmen Lösungen in Zustandsräumen bis $|Z| = 10^{25}$ gefunden werden (Geffner 2000).

4.4.2 Planen

Die Forschung begann in den 60er Jahren mit der heute als *klassisch* bezeichneten Planung, die aber noch immer grundlegend ist. Klassisches Planen basiert auf dem **Situationskalkül** (vgl. S. 17). Die mathematische Modellierung entspricht dem **Zustandsmodell**. Im einfachsten Fall ist ein Zustand eine Situation und ein Übergang eine Aktion: Es gibt eine Menge möglicher Situationen *S*, eine Ausgangssituation $s_0 \in S$, eine Menge möglicher Zielsituationen $S_z \subseteq S$, in jeder Situation ausführbare Aktionen *A*(*s*) und eine Überführungsfunktion für $s' = \delta(a, s)$ mit $a \in A(s)$. Gibt es eine Aktionsfolge a_i ($i = 0, \ldots, n$), welche die Ausgangssituation s_0 in eine Zielsituation $s_z \in S_z$ überführt, dann ist diese Folge die Lösung des Planungsproblems: der Plan.

Der Schlüssel für die Leistungsfähigkeit von Planungssystemen ist die Wahl der Repräsentationssprache und der Algorithmen. Am Anfang stand 1972 der **STRIPS Formalismus** (*Stanford Research Institute Problem Solver*) (Fikes, Nilsson 1971), eine Beschreibungssprache, mit welcher der Roboter *Shakey* (Nilsson 1984) seine Welt modellieren und sinnvolle Aktionen ableiten sollte. Nach einigen Standardisierungen hat STRIPS heute das Aussehen von Bild 4-9.

Die Sprache **ADL** (*Action Description Language*) (Pednault 1989) erweitert STRIPS um Negation, Quantoren und Disjunktion. Insbesondere dürfen jetzt auch so genannte kontextsensitive Effekte auftreten, d.h. je nach Zusatzbedingungen, die ein Zustand erfüllt, kann eine Aktion nun eine ganz andere Zustandsänderung bewirken. Beispielsweise erlaubt die ADL Variante des `go`-Operators von Bild 4-9, den Ort aller sich im Besitz des Roboters befindlichen Objekte direkt abzuleiten, da der kontextsensitive Effekt besagt, dass sich diese immer mit dem Roboter mit bewegen:

```
                go(?loc1, ?loc2:location)
Vorbedingung:   in(?loc1)
Effekte:        ADD in(?loc2) DEL in(?loc1)
Kontexteffekte: ALL ?b$:object have(?b) ==> ADD in(?b,loc2) DEL in(?b,loc1)
```

(4-1)

Vorgehen	Beschreibung	Beispiel in STRIPS

Ausgangs-situation	Die Ausgangssituation („Wie ist die Welt momentan?“) wird durch eine Konjunktion von logischen Atomformeln (ohne Funktionssymbole) beschrieben. Es gilt meist die *closed-world assumption*, d.h. Fakten, die nicht vorkommen, gelten als falsch.	`origin(letter office1)`
		`in(office1)`
		`destination(letter office2)`

Ziel	Das Ziel des Planungsprozesses („Was soll erreicht werden?“) wird ebenfalls durch eine Konjunktion von Atomen beschrieben.	`delivered(letter)`

Mögliche Aktionen	Die möglichen Aktionen des Planungssystems („Welche Aktionen gibt es?“) werden durch Operatoren beschrieben. Charakteristisch sind die ADD- und DELETE-Listen in den Effekten der Operatoren. Sind alle Atomformeln in der Vorbedingung (Vorbed.) eines Operators in einem Zustand erfüllt, dann erhält man den Folgezustand, indem die Atome der ADD-Liste zur Zustandsbeschreibung hinzu genommen und die Atome der DELETE-Liste daraus entfernt werden.	`drop(?b:object, ?loc:location)`
		Vorbed.: `have(?b), dest(?b ?loc), in(?loc)`
		Effekte: `ADD delivered(?b) DELETE have(?b)`
		`get(?b:object, ?loc:location)`
		Vorbed.: `origin(?b ?loc), in(?loc)`
		Effekte: `ADD have(?b)`
		`go(?loc1, ?loc2:location)`
		Vorbed.: `in(?loc1)`
		Effekte: `ADD in(?loc2) DELETE in(?loc1)`

Plan	Das Ziel des Planungsprozesses („Was soll erreicht werden?“) wird ebenfalls durch eine Konjunktion von Atomen beschrieben.	`get(letter,office1)`
		`go(office1,office2)`
		`drop(letter,office2)`

Bild 4-9: STRIPS Formalismus (Köhler 2000).

ADL wurde später für Wettbewerbe für Planungssysteme in die Variante **PDDL** (*Problem Domain Definition Language*) (Ghallab et al. 1998) weiterentwickelt und standardisiert.

Zur Lösung der beschriebenen Planungsprobleme gibt es heute viele verschiedene **Algorithmen** und Systeme (Biundo et al. 1995), (Geffner 2000), (Köhler 2000), (Rintanen, Hoffmann 2001).

Das *klassische Planen* wird heute um Aufgabenstellungen *erweitert*, die daraus folgen, dass Aktionen eine numerisch anzugebende Dauer haben, sich überlappen können oder parallel ausgeführt werden müssen. Zur Beschreibung für diese Art von Problemen wurde die Repräsentationssprache PDDL angepasst. Bild 4-10 zeigt die Mächtigkeiten der Versionen **PDDL2.1** (Fox, Long 2002) für den Planungswettbewerb (IPC 2002) und **PDDL+** (Fox, Long 2001).

Stufe / Mächtigkeit				Beschreibung
PDDL+	PDDL2.1	Level 1	*STRIPS*	STRIPS und ADL des ursprünglichen PDDL (Ghallab et al. 1998)
		Level 2	*Numeric* *Hard Numeric*	Einfache numerische Effekte Numerische Effekte mit Abhängigkeiten von anderen Größen
		Level 3	*Simple Time*	Einfache, zeitbehaftete Aktionen ohne kontinuierliche Effekte
		Level 4	*Time*	Zeitbehaftete Aktionen mit sich kontinuierlich ändernden Werten
		Level 5	*Complex*	Gemischt diskrete/kontinuierliche Domänen mit kontinuierlicher Zeit

Bild 4-10: Mächtigkeit von PDDL+ (Fox, Long 2001) und PDDL2.1 (Fox, Long 2002).

Zum Vergleich der verschiedenen Algorithmen und Planungssysteme haben sich Benchmark-Probleme etabliert (IPC 2002). Eines davon, die so genannte *Depots*-Domäne, ist dem Hol- und Bringdienst, wie er in Kapitel 3.2 analysiert wurde, besonders ähnlich: die Kisten der *Depots*-Domäne entsprechen den zu transportierenden Objekten, anstelle der Lastwagen fährt der Roboterassistent und die Paletten entsprechen diskreten Plätzen, an denen die Objekte abgestellt werden. Anstelle von Winden nutzt der Roboterassistent seinen Manipulator. Bild 4-11 beschreibt die Domäne in den verschiedenen Schwierigkeitsstufen von PDDL2.1.

***Depots*-Domäne**

- *Level 1*: Lastwagen transportieren Kisten, die an ihrem Ziel auf Paletten gestapelt werden. Die Kisten werden dort mit Winden bewegt. Die Anzahl von Paletten ist limitiert.
- *Level 2*: Die Lastwagen haben eine beschränkte Ladekapazität und verbrauchen Energie während der Bewegung. Die Kisten haben Gewicht, so dass die Winden Energie zum Heben benötigen. Beim Lösen der Aufgabe soll ein minimaler Energieverbrauch erreicht werden.
- *Level 3*: Der Zeitkonsum variiert von 10 für Fahren bis 1 für Abladen einer Kiste auf einen Stapel. Es soll eine nebenläufige Benutzung der Lastwagen und Winden berücksichtigt werden.
- *Level 4*: Der Zeitkonsum ist abhängig von der gefahrenen Distanz und Geschwindigkeit zwischen Orten. Winden benötigen unterschiedlich viel Zeit zum Be- und Entladen der Kisten abhängig von ihrer Leistung und vom Gewicht der Kisten. Dies macht die Berücksichtigung der Nebenläufigkeit schwieriger, da schnellere Lastwagen verfügbar sind, aber an anderer Stelle als sie gebraucht werden. Ein Planungssystem muss die Koordinierung der Positionen und den Gebrauch der Winden berücksichtigen.

Bild 4-11: *Depots*-Domäne (IPC 2002).

Bild 4-12 zeigt eine Übersicht über die Leistungsfähigkeit verschiedener Planungssysteme für die *Depots*-Domäne:

Planungssystem	Mächtigkeit				Benchmark-Ergebnis		
	Level 1	*Level 2*	*Level 3*	*Level 4*	Ange-wandte Probleme	Gelöste Probleme	Erfolgs-quote
IxTeT	○	○	●	○	22	1	5%
LPG	●	●	●	●	264	231	88%
metric-FF	●	●	○	○	88	55	63%
MIPS	●	●	●	●	176	82	47%
Sapa	○	○	○	●	22	5	23%
SemSyn	●	●	○	○	44	3	7%
SHOP2	●	●	●	●	110	110	100%
Simplanner	●	○	○	○	22	22	100%
Stella	●	○	○	○	22	4	18%
TALPlanner	●	○	●	●	66	66	100%
TLPlan	●	●	●	●	88	88	100%
TP4	○	○	●	●	44	2	5%
VHPOP	●	○	●	○	44	6	14%
Legende: ● hat in dieser Kategorie teilgenommen ○ hat in dieser Kategorie nicht teilgenommen							

Bild 4-12: Benchmark-Ergebnis für die *Depots*-Domäne (IPC 2002).

4.4.3 Entscheiden

Die bisher betrachteten Methoden „Problemlösen" und „Planen" gehen von einer offenen Steuerung aus: nach der Problembeschreibung wird ein Plan erstellt, von dessen Ausführung erwartet wird, dass die Aktionen zum Ziel führen. Es ist jedoch keine Regelrückführung darin enthalten, falls eine der bei der Planung getroffenen Annahmen nicht korrekt war oder eine Aktion misslingt. Diese Lücke schließen **MDP** (*Markov decision processes*) (Puterman 1994):

Als mathematisches Modell wird von dem **Zustandsmodell** von Abschnitt 4.4.1 ausgegangen. Die Operationen werden um **stochastische Effekte** erweitert, indem die Überführungsfunktion nun durch die bedingte Wahrscheinlichkeit $P_o(z' \mid z)$ mit $z \in Z$ und $o \in O(z)$, dass ein bestimmter Nachfolgezustand z' erreicht wird, bestimmt wird. Für jede Aktion gibt es eine Kostenfunktion $\kappa(o,z) > 0$. Der Weg mit den geringsten erwarteten Kosten ist der optimale Plan.

Im Gegensatz zu MDP, bei denen vorausgesetzt wird, dass vollständig beobachtbar ist, in welchem Zustand sich das System befindet, berücksichtigen **POMDP**s (*particially observable MDP*) (Littman et al. 1997), (Cassandra 1999) zusätzlich eine Wahrscheinlichkeitsverteilung über die

Zustände, mit welcher das System glaubt, dass es sich gerade in z befindet, wenn dies nicht eindeutig beobachtbar ist. Es wird noch eine zusätzliche Beobachtungswahrscheinlichkeitsverteilung berücksichtigt, wenn auch die Beobachtung des aktuellen Zustands fehlerbehaftet ist.

Es gibt verschiedene **Algorithmen**, um die damit aufgespannten Problemräume zu lösen, z.B. heuristische Suche oder *Real-Time Dynamic Programming* (Barto et al. 1995). Als **Repräsentationssprache** hat sich noch keine Entwicklung durchgesetzt (Geffner 2000).

POMDPs scheinen für technische Agenten das adäquateste Modell zu sein, sind aber zu aufwändig: Die Wahrscheinlichkeitsverteilungen sind unter Berücksichtigung der Beobachtbarkeit mit mehr als 10 Situationen sehr schwierig zu berechnen (Strube 2000). Außerdem muss der Zustandsraum zur Verhaltenssteuerung im Voraus programmiert werden und lässt sich nicht automatisch generieren.

4.5 Wissensbasen

In einer Wissensbasis werden die Modelle der Umwelt gespeichert, auf deren Basis Handlungen geplant werden. Zur Strukturierung dienen **Ontologien**, die das Vokabular an Symbolen mit vorgegebener Bedeutung sowie Beziehungen zwischen diesen definieren (vgl. auch S.18). Sie werden verwendet, um Wissen auszutauschen, d.h. zur Definition einer gemeinsamen Sprache. In Roboterassistenten hoher Komplexität müssen die Untersysteme Informationen über das Weltmodell miteinander und auch mit dem Mensch als Programmierer und Benutzer austauschen. Die Ontologie dient zum Entwurf der technischen Lösung.

Umfangreiche Übersichten über den Stand der Technik von Ontologien sind in (Fridman, Hafner 1997) und (Gómez-Pérez 1999) gegeben. Bild 4-13 zeigt exemplarisch vier verschiedene Basiskonzepte in Ontologien.

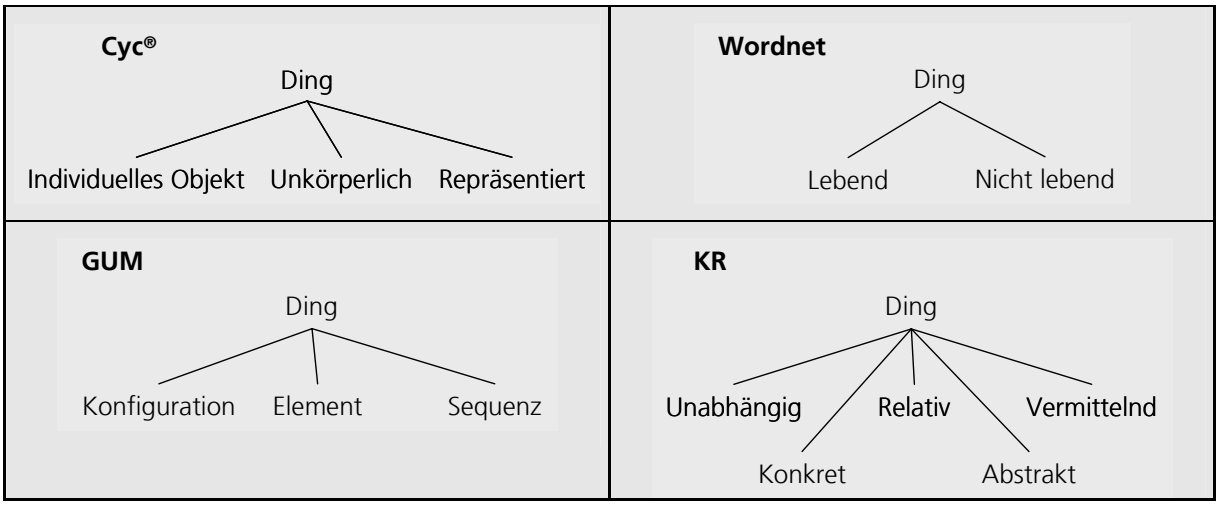


Bild 4-13: Basiskonzepte vier verschiedener Ontologien (Chandrasekaran et al. 1999).

Hervorzuheben ist die **KR-Ontologie**, mit der John F. Sowa versucht, philosophische Einsichten in einer generellen Ontologie zu erfassen (Sowa 2000). Sie ist als Basis für die Beschreibung von Weltdaten in technischen Systemen am besten geeignet, da sie technische Begriffe wie z.B. Objekt, Prozess, Relation, etc. unterscheidet. Die zwölf zentralen Kategorien und die Primitive, von denen sie abgeleitet werden, sind in der Matrix von Bild 4-14 dargestellt:

	Konkret		**Abstrakt**	
	Kontinuierlich	**Ereignend**	**Kontinuierlich**	**Ereignend**
Unabhängig	Objekt	Prozess	Schema	Ablauf
Relativ	Verbindung	Teilnahme	Beschreibung	Historie
Vermittelnd	Struktur	Situation	Begründung	Absicht

Bild 4-14: Matrix der zwölf zentralen Kategorien der KR-Ontologie (Sowa 2000).

Philosophen erstellen eine Ontologie nach dem top-down Ansatz: sie beginnen mit großen Konzepten über alles, das existiert, von Himmel und Erde (Sowa 2000, S. 52). Beim Programmieren bietet sich dagegen ein bottom-up Ansatz an, d.h. mit eingeschränkten Ontologien und Mikrowelten zu beginnen, die eine geringe Anzahl von maßgeschneiderten Konzepten für die spezielle Anwendung haben. Generell gilt für alle Ontologien (Chandrasekaran et al. 1999):

- In der Welt existieren **Objekte**.
- Objekte können **Teile** haben.

- Objekte haben **Eigenschaften** oder **Attribute**, die **Werte** annehmen können.
- Objekte können in verschiedenen **Relationen** zueinander stehen.
- Eigenschaften und Relationen können sich über die **Zeit** ändern.
- Es gibt **Ereignisse**, die zu verschiedenen **Zeitabschnitten** eintreten können.
- Es gibt **Prozesse**, an denen Objekte teilhaben und die über die Zeit geschehen.
- Die Welt und ihre Objekte kann sich in verschiedenen **Zuständen** befinden.
- Ereignisse können andere Ereignisse oder Zustandsänderungen als **Effekte verursachen**.

5 Lösungsansatz für die Kontrollarchitektur

Für die Kontrollarchitektur und ihre Teilsysteme werden Lösungskonzepte erarbeitet und anhand der in den vorangegangenen Kapiteln abgeleiteten Anforderungen bewertet und ausgewählt.

5.1 Entwicklung des Gesamtkonzeptes

Aus den Anforderungen von Kapitel 3.3 und Kapitel 4.3.6 werden unmittelbar die folgenden Schlussfolgerungen gezogen:

- Für die Zielorientierung im Verhalten beinhaltet die Kontrollarchitektur eine Planungskomponente (Ã5), die vor der Ausführung die Situation analysiert und die Handlung plant.

- Die Kontrollarchitektur soll modular aufgebaut sein (Ã1). Innerhalb der Module haben die Prozesse unterschiedliche Zeitanforderungen; langsame Berechnungen, wie zum Beispiel Planen (Ã5), müssen nebenläufig zu schnellen Regelungsalgorithmen (Ã2, A 12) ausgeführt werden. Deshalb müssen die Prozesse in den Modulen **asynchron** laufen. Zwischen den Modulen werden Mechanismen zur Synchronisierung und zum Datenaustausch benötigt.

Im Stand der Technik wurden bewährte Kontrollarchitekturen von mobilen Agenten identifiziert (Kapitel 4.3). In Bild 5-1 werden diese den erarbeiteten Anforderungen für eine Kontrollarchitektur für Roboterassistenten gegenüber gestellt und bewertet:

Bewertungskriterium:	Modularität und Erweiterungsfähigkeit	Schnittstellen zur Roboterstеuerung	Explizite Umweltrepräsentation	Adaptierende Verhaltenskontrolle	Planungskomponente	Auswahl
Entspricht Anforderung:	Ã1	Ã2	Ã3	Ã4	Ã5	
(1) Sequentielle Schichtenarchitektur	○ Keine asynchronen Module.	○ Keine Entkopplung zeitkritischer Funktionen.	● Ja.	● Ja, in der Sequenz aus Planen und Handeln.	● Ja.	
(2) Verhaltensbasierte Architektur	● Ja, Verhaltensmodule.	○ Nein, direkte Verbindung.	○ Nein, nicht vorhanden.	● Ja.	○ Nein, nicht vorhanden.	
(3) Mehrebenen-Architektur	● Ja, Robotersteuerung, Verhaltenskontrolle, Planer.	● Ja.	● Ja.	● Ja.	● Ja.	←
(4) PRS-Architektur	● Ja.	○ Nein, zeitkritische Zieländerungen nur über Faktenanpassung möglich.	● Ja.	● Ja.	● Ja.	
(5) Konnektionismus	● Ja.	○ Nein, direkte Verbindung.	○ Nein, Wissen ist verteilt.	● Ja.	○ Nein, nicht vorhanden.	
Legende:	● geeignet	○ nicht / wenig geeignet				

Bild 5-1: Varianten, Bewertung und Auswahl für das Gesamtkonzept der Kontrollarchitektur für Roboterassistenten.

Variante (3) erfüllt alle Anforderungen Ã1 bis Ã5. Die Kontrollarchitektur wird als Mehrebenen-Architektur entwickelt. Diese verbindet ein Planungssystem mit einer reaktiven Robotersteuerung über eine adaptierende Verhaltenskontrolle. In der Recherche des Stands der Technik wurde kein System gefunden, das alle hier erarbeiteten Anforderungen an eine modulare, übertragbare Kontrollarchitektur für Roboterassistenten erfüllt. Die bekannten Systeme sind entweder für Serviceroboter ohne Manipulatoren oder für bestimmte Anwendungsszenarios oder bestimmte Roboter spezialisiert.

Die Auswahl der Mehrebenen-Architektur bedingt die Entwicklung der folgenden Teilsysteme:

- Die unterste Ebene der Mehrebenen-Architektur ist eine Robotersteuerung mit Schnittstellen zur Hardware. Sie steuert die Aktuatoren an, liest die Sensoren aus und stellt schnelle Regelungsprozesse zur Verfügung, die in Echtzeit reagieren können (Anforderung A 12). Es müssen Schnittstellen definiert werden, über die ein

übergeordnetes Ausführungsmodul die Ansteuerung der Effektoren aktivieren bzw. deaktivieren kann.

- Auf mittlerer Ebene der Mehrebenen-Architektur sorgt ein **Ausführungsmodul** für die Verhaltenskontrolle, indem es so genannte Aktionen in der Robotersteuerung aktiviert bzw. deaktiviert. Es überwacht die Aktionen auf Erfolg (A 13) und sorgt im Falle von Störungen für eine Störungsbehandlung (A 15). Die Aktionen zur Verhaltenskontrolle werden aus einer übergeordneten Ebene vorgegeben.
- Eine Liste von Aktionen, die von der aktuellen Situation zu einer geforderten Zielsituation führt, erstellt ein **Planungsmodul** auf oberster Ebene der Mehrebenen-Architektur. Zur Planung werden Informationen aus einer Umweltrepräsentation herangezogen. Das Planungsmodul muss so mächtig sein, dass auch eine verschachtelte Aufgabenausführung möglich ist (A 18).
- Alle Ebenen benötigen Informationen über die den Roboterassistenten umgebende Umwelt sowohl zum Planen als auch zur Verhaltenssteuerung und zur Aktionsausführung. Diese werden in einer **Wissensbasis** gespeichert und enthalten insbesondere Umweltinformationen, die nicht aktuell mit Sensoren erfasst werden können (A 8).
- Damit die Wissensbasis immer auf dem aktuellen Stand ist, muss es Methoden zu ihrer **Wartung** geben (A 10), welche die Sensordaten interpretieren und erkannte Veränderungen in der Umwelt eintragen.
- Die Aufträge an den Roboterassistenten werden über eine **Mensch-Maschine-Schnittstelle** kommuniziert (A 3). Diese dient im Allgemeinen zur Interaktion gemäß Bild 3-1.

Jedes der im Text hervorgehobenen Module wird abgegrenzt durch seine Funktionen und durch Schnittstellen entwickelt. Die Module sind damit austauschbar bzw. die Gesamtarchitektur ist dadurch modular und erweiterbar (Ã1). Die Entwicklung von Robotersteuerungen war Fokus anderer Arbeiten (vgl. Kapitel 4.2) und wird als gegeben genommen. Die anderen Teilsysteme werden in den folgenden Kapiteln neu konzipiert. In Kapitel 5.8 werden sie zu einem detaillierten Gesamtkonzept zusammengeführt.

5.2 Konzeption des Ausführungsmoduls

Das Ausführungsmodul ist innerhalb der Kontrollarchitektur für die Verhaltenskontrolle zuständig. Hier wird in Abhängigkeit von der aktuellen Situation und Aufgabe entschieden, wann der Roboterassistent welche Aktion ausführt. Intuitiv könnte man diese Abbildung mit den Gleichungen für die Ausgabefunktion ω und die Zustandsüberführungsfunktion δ

$$\begin{aligned} Y(n) &= \omega\left[Z(n), X(n)\right] \\ Z(n+1) &= \delta\left[Z(n), X(n)\right] \end{aligned} \qquad \textbf{(5-1)}$$

eines endlichen Automaten beschreiben. Die Eingangswerte $X(n)$ zum diskreten Zeitpunkt n sind Sensoreingänge, mit denen die aktuelle Umweltsituation erfasst wird, bzw. Umweltmodelle über den Bereich, der nicht aktuell von den Sensoren erfasst werden kann. Die zu lösende Aufgabe des Roboterassistenten ist in der Ausgabefunktion ω codiert, welche den Ausgang $Y(n)$ in Abhängigkeit vom Eingang $X(n)$ und vom inneren Zustand des Roboterassistenten $Z(n)$ setzt.

Wenn der Roboterassistent nun aber verschiedene Aufgaben erfüllen soll, so müsste dessen Automat im Ausführungsmodul in der Ausgabefunktion ω alle Aufgaben vereinen. Dies ist nicht praktikabel, da das Aufgabenspektrum nachträglich noch erweitert werden können soll bzw. nicht alle Aufgaben vorausgesagt werden können. Es wird deshalb ein anderer Lösungsansatz weiter verfolgt, damit das Ausführungsmodul in der Lage ist, unterschiedliche, außerhalb des Ausführungsmoduls spezifizierte Aufgaben abzuwickeln.

5.2.1 Funktionen des Ausführungsmoduls

Das Ausführungsmodul erhält als Eingang eine Aufgabe zur Abwicklung. Diese kann entweder eine Zielvorgabe sein, die über eine Mensch-Maschine-Schnittstelle vom Benutzer vorgegeben wird, oder ein Plan. Ein Plan ist eine partiell geordnete Liste von Aktionen, deren Ausführung von der aktuellen Situation zu einer Zielsituation führt. Der Plan kann entweder dynamisch von einem Planungsmodul berechnet werden, er kann aus einer endlichen Anzahl von vorprogrammierten Plänen vom Benutzer ausgewählt werden oder er wird starr vorgegeben; im letzten Fall wird der Roboterassistent immer die gleiche Handlung ausführen.

Ist noch kein Plan gegeben, muss das Ausführungsmodul die Erstellung eines Planes initiieren. Dafür benötigt es eine Zielvorgabe und eine Umweltbeschreibung, die zusammen an das Planungsmodul übergeben werden. Die Zielvorgabe wird mit der Aufgabe des Benutzers gegeben. Der Plan enthält Informationen *was* ausgeführt werden muss, aber nicht *wie*. Pläne sollen die Handlungen führen, aber nicht die Ausführung steuern.

Das Ausführungsmodul wickelt die Pläne Schritt für Schritt ab und zerlegt die geplanten Teilschritte in feinere Aktionen. Bei der Umsetzung, *wie* die Teilschritte ausgeführt werden, ist der Roboterassistent dadurch in der Lage, seine Handlungen an eine sich verändernde Umwelt anzupassen (A 11). Die Aktionen können auch verschachtelt werden (A 18).

Das Ausführungsmodul sichert den Erfolg der Handlung, indem es die Störungsbehandlung bei Scheitern einzelner Aktionen übernimmt. Nach der Ausführung einer Aktion findet eine Erfolgsprüfung statt (A 13). Misslingt sie, wird eine andere Aktion aufgerufen. Dies wird so lange wiederholt, bis der Erfolgstest positiv ist oder keine alternative Aktion mehr zur Verfügung steht. Dadurch wird das System robust, wenn eine Aktion nicht zum gewünschten Erfolg führt, beispielsweise weil sich die Umwelt während der Ausführung verändert hat.

Parallel zur regulären Aktionsausführung wird die Umwelt beobachtet, um auf Chancen und Risiken reagieren zu können. Zum Beispiel muss der Roboterassistent darauf achten, ob er schon auf seinem Weg auf den Gegenstand stößt, den er holen soll. Das Ausführungsmodul muss also gleichzeitig mehrere Aufgaben ausführen, um auf Chancen zu achten und schnell seine Pläne anzupassen, wenn sich günstige Gelegenheiten ergeben. Mittels spezieller Aktionen kann der Roboterassistent aktiv Umweltfakten akquirieren, die er nicht kennt.

Die Aktionen sind demnach kombinierbare Handlungsschritte gemäß Anforderung (A 16). Sie sind in der Robotersteuerung gekapselt. Das System ist modular und die Software wieder verwendbar. Außerdem muss der Programmierer des Ausführungsmoduls keine Kenntnis über die Realisierung der Aktionen in der Robotersteuerung haben.

5.2.2 Eigenschaften von Aktionen

Die wesentlichen Aktionen des Hol- und Bringdienstes sind in der Analyse identifiziert (siehe Kapitel 3.2.1). Sie lassen sich großenteils hierarchisch untergliedern, wie in Bild 3-4 dargestellt. Es gibt jedoch wichtige Ausnahmen, bei denen die Aktionen nicht hierarchisch, sondern überlappend angeordnet werden müssen, siehe Bild 5-2.

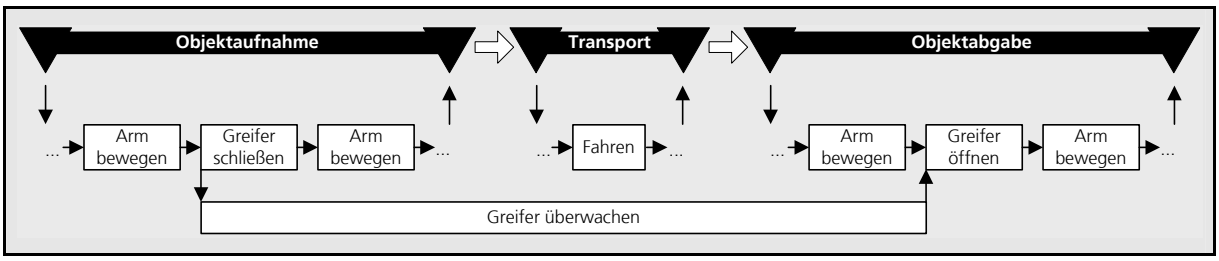


Bild 5-2: Beispiel für überlappende Aktionen.

Bild 5-2 zeigt, dass die Aktion „Greifer überwachen“ von dem Augenblick an aktiv sein muss, in dem ein Objekt aufgenommen wird, bis zu dem Augenblick, in dem es wieder abgestellt wird,

um sicherzustellen, dass das Objekt nicht verloren geht. Ein anderes Beispiel für überlappende Aktionen ist die Realisierung des feinfühligen Greifens über ein Tracking[5]: nachdem der Greifer grob positioniert worden ist, muss das Tracking so lange aktiv sein und die Regelgröße zum feinen Positionieren liefern, bis der Greifer geschlossen und das Objekt gegriffen ist.

Wichtig ist also die Fähigkeit des Ausführungsmoduls, nicht nur sequenzielle, sondern auch nebenläufige Aktionen zu unterstützen. Diese sind eine endliche Zeit lang oder kontinuierlich aktiv. Die möglichen Kombinationen von Ausführungsmodellen werden in Bild 5-3 strukturiert (siehe auch Analyse der Handlungsschritte für den Hol- und Bringdienst von Bild 3-5):

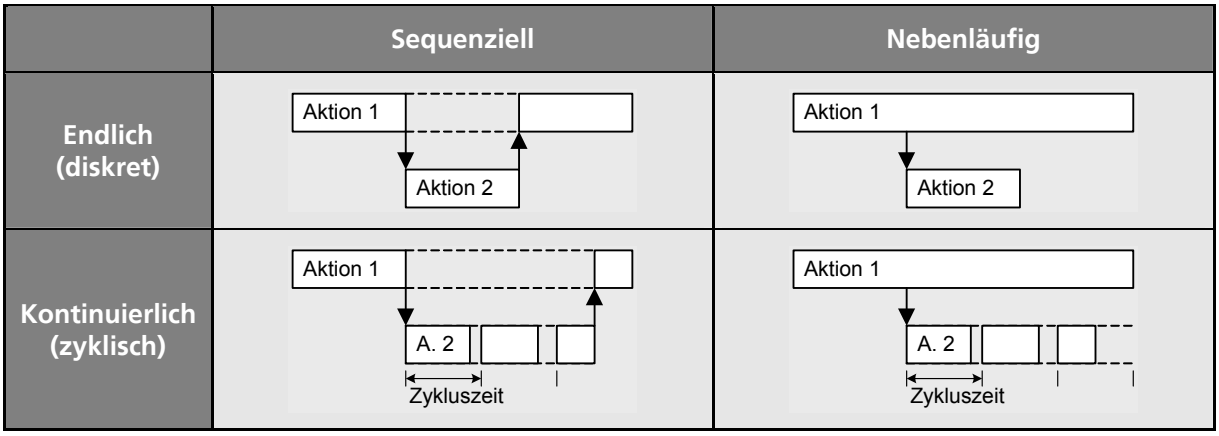


Bild 5-3: Ausführungsmodelle für Aktionen (Baum 2003), (Hans, Schraft 2003).

Aus Bild 5-2 lässt sich weiter ablesen, dass die überlappende Aktion „Greifer überwachen" von einer anderen Aktion beendet wird, als sie gestartet wurde. Um dies zu ermöglichen, wäre ein Konzept, das Aktivieren und Deaktivieren von einer hierarchisch übergeordneten Aktion aus zu koordinieren. Diese müsste in Bild 5-2 dann hierarchisch über den schwarz gekennzeichneten Aktionen „Objektaufnahme", „Transport" und „Objektabgabe" liegen. Sie wäre dann direkt von dem aktuellen Geschehen in der Umwelt betroffen und damit nicht mehr abstrakt und unabhängig. Außerdem ließen sich die Verknüpfungen zwischen den schwarz gekennzeichneten Aktionen nicht mehr automatisch planen. Dieses Konzept widerspricht also den Anforderungen nach Modularität (Ã1) und nach einer Planungskomponente (Ã5). Es ist also ein anderes Konzept notwendig, um eine abstrakte und modulare Struktur der Aktionen zu erhalten:

[5] Unter *Tracking* wird hier das Erkennen und Verfolgen eines Objektes mit kontinuierlicher Positionsbestimmung verstanden. Der Positionswert kann für eine Regelung verwendet werden.

Jede Aktion des Ausführungsmoduls ist ein eigenständiger Prozess, der aktiviert oder deaktiviert werden kann. Wenn eine Aktion aktiviert wird, wird ihr ein Satz von Parametern übergeben und der Prozess beginnt zu laufen. Alle laufenden Aktionen haben bei Bedarf Zugang zu aktuellen Sensordaten und Daten, die von anderen Aktionen generiert werden. Die Aktionen stehen in einer Bibliothek zur Verfügung.

Aktionen können andere Aktionen aufrufen. So entsteht eine hierarchische Ordnung und Verfeinerung, indem Aktionen aus der Bibliothek zu neuen Aktionen kombiniert werden. Eine derartige Hierarchie entspricht der Darstellung von Bild 3-4. Die Bibliothek kann zum einen Aktionen enthalten, die der Erfüllung von Teilzielen dienen, zum anderen auch primitive Aktionen, die Regelungsprozesse in der Robotersteuerung ein- und ausschalten.

Damit nun eine überlappende Aktion von einer anderen beendet werden kann, muss die übergeordnete Aktion in der Lage sein, die nebenläufigen Aktionen zu identifizieren, ihren Status abzufragen und sie dann ggf. zu beenden. Ohne diese Fähigkeiten können Aktionen nicht abstrakt und modular geschaffen werden und es besteht keine Möglichkeit, die höheren Abstraktionsebenen des Systems von den Details der Ausführung abzuschirmen.

Dieses Konzept erleichtert außerdem die Entwicklungsarbeit aus Engineering-Sicht, weil eigenständige Aktionen unabhängig voneinander entwickelt und getestet werden können.

5.2.3 Behandlung von Störsituationen während der Aktionsausführung

Robustes Handeln wird durch ein Konzept in zwei Schritten sichergestellt: zuerst müssen Störsituationen, d.h. Diskrepanzen zwischen einer korrekten Ausführung und der tatsächlichen Störung, erkannt werden. Danach wird eine geeignete Maßnahme zur Störungsbehandlung eingeleitet.

Für die Erkennung von Störsituationen überprüft jede Aktion ihre eigene Ausführung anhand von Plausibilitätsprüfungen und eigener, interner Störungsmeldungen. Zu jeder Störung gehört eine spezifische, erklärende Störungsmeldung. Auf diese Meldung kann noch in der gleichen Aktion eine andere Aktion zur Störungsbehandlung aufgerufen werden oder die Meldung wird an die hierarchisch übergeordnete Aktion weiter geleitet, welche die gestörte Aktion aufgerufen hatte. Sind mehrere Aktionen ineinander verschachtelt, so wird die Störungsmeldung so lange weiter gegeben, bis eine Aktion die Störsituation behandelt.

Für die Behandlung von Störsituationen sind in jeder Aktion spezifische Lösungen vorprogrammiert. Fällt z.B. beim Transport das zu transportierende Objekt herunter, so kann in der Aktion direkt eine Aktion „Aufheben" aktiviert werden. Erst wenn das Aufheben scheitert, muss hierarchisch übergeordnet entschieden werden, wie weiter gehandelt werden soll. Weitere Störungen des Hol- und Bringdienstes sind in Bild 3-3 analysiert.

Kann aufgrund der Störung ein Plan nicht zu Ende ausgeführt werden, wird er abgebrochen und das Planungsmodul aufgefordert, einen neuen Plan mit der neuen Situation zu erstellen. Wenn keine geeignete Maßnahme selbstständig vom Roboterassistenten gefunden wird, wird der Mensch interaktiv in die Behandlung einbezogen, indem die Meldung an ihn übertragen wird.

5.3 Entwicklung des Ausführungsmoduls

5.3.1 Variantenbildung

Zur Entwicklung des zuvor konzipierten Ausführungsmoduls stehen die folgenden vier Varianten zur Auswahl:

JAM Agents ist eine Weiterentwicklung des *Procedural Reasoning System (PRS)* (vgl. Kapitel 4.3.4). Nach Definition von Fakten (*beliefs*) und Zielen (*desires*) werden dynamisch während der Laufzeit Aktionen (*intentions*) basierend auf der Erfüllung der Vorbedingungen ausgewählt. Dieses Softwarepaket kann die Aktivierung und Erfolgskontrolle der Aktionen übernehmen. Die Reihenfolge der Aktionen muss von außen vorgegeben werden, sonst wählt das System immer die erste in der Programmierreihenfolge, deren Vorbedingungen erfüllt sind. In einer Simulation und auf dem Roboterassistenten für Produktion *rob@work* (Bild 4-1 unten Mitte) wurde der Hol- und Bringdienst erfolgreich demonstriert (Helms et al. 2002).

COLBERT ist eine kommerzielle, in C++ programmiert Software, die ebenfalls aus PRS entwickelt wurde (vgl. Kapitel 4.3.4). COLBERT ist Teil der Architektur Saphira, die speziell für den Roboter *Pioneer* angepasst wurde und mit diesem kommerziell vertrieben wird. Die Software bietet eine speziell für mobile Roboter entwickelte Skriptsprache, die sehr ähnlich wie C zu programmieren ist. Es werden so genannte *Activities* programmiert, die von der Architektur verwaltet und angesteuert werden.

Python (Python 2003) ist eine frei verfügbare, interpretierte, interaktive, objektorientierte Programmiersprache. Python ist sehr mächtig und hat eine einfache Syntax. Es hat Module, Klassen, Ausnahmebehandlungsroutinen, high-level Datentypen und dynamische Typen. Es gibt

Interfaces zu den meisten Betriebssystemen, Bibliotheken und Programmiersprachen. Python wird häufig als Erweiterungssprache für Anwendungen genutzt, die ein programmierbares Interface benötigen. Die Aktionen des Ausführungsmoduls werden als Funktionen in Python programmiert und können ggf. parallel aufgerufen werden. Ein externes Planungsmodul ist einfach einzubinden. Weil Python eine Skriptsprache ist und zur Laufzeit interpretiert wird, bietet sie einerseits den Vorteil einer interaktiven Kommandozeileneingabe, ist andererseits aber auch langsam.

Als vierte Variante besteht die Möglichkeit, eine **eigene Lösung in C++** zu entwickeln. Die Aktionen sind Klassen und die Liste möglicher Aktionen ein .cpp-file. Mittels des selbst entwickelten Ausführungsmoduls müssen die Klassen in der richtigen Reihenfolge zur richtigen Zeit ausgeführt werden. Die Reihenfolge wird durch ein Planungsmodul vorgegeben. Diese Variante hat den Vorteil, dass die komplette Software-Kontrollarchitektur in C++ ist. Es müssen jedoch alle Mechanismen selbst spezifiziert und entwickelt werden, so dass das Risiko besteht, dass Probleme erst während der Entwicklung erkannt werden, die in verfügbaren Softwarepaketen bereits gelöst sind.

5.3.2 Bewertung

Im Folgenden werden die vier Varianten für die Realisierung eines Ausführungsmoduls bewertet. Die Kriterien und die Bewertung sind in Bild 5-4 dargestellt. Neben den wissenschaftlichen Kriterien einer idealen Lösung sind auch Engineering-Aspekte von sehr großer Bedeutung: So ist die Entwicklung und Fehlersuche mit einer interaktiven Skriptsprache sehr viel einfacher als mit einer Programmiersprache, bei der nach jeder Änderung das Projekt neu kompiliert werden muss. Außerdem muss bei einer Programmiersprache zum Testen jeder Funktion und jedes Moduls immer ein eigenes Testprogramm geschrieben werden. In interaktiven Skriptsprachen können dagegen Funktionen direkt aus einer Kommandozeile aufgerufen werden.

Aus Engineering-Sicht muss die Realisierung transparent, einfach, leicht verständlich und modifizierbar sein. Der Ablaufprogrammierer soll eine Hochsprache verwenden können und soll nicht über Kenntnisse über die Robotersteuerung verfügen müssen.

Kriterium	JAM Agents	COLBERT	Python	C++ Eigenentwicklung
Art	BDI-Architektur (für Roboter entwickelt)	Skriptsprache; speziell für Roboter entwickelt	Objektorientierte Skriptsprache	Objektorientierte Programmiersprache
Progr.-Sprache	JAM	Teilmenge von C bzw. C-ähnlich	Python	C++
Lösung für ein Ausführungsmodul	Aktionen in JAM-Files; JAM Agents übernehmen Sequencing Anbindung über JNI	Aktionen entsprechen Activities in COLBERT; Saphira übernimmt Sequencing	Aktionen als Funktionen mit Anbindung an untere C++-Ebene der Steuerung; Sequencer in Python programmieren; Python bietet API für high-level Roboteranwendungen	Aktionen als C-Klassen; Liste möglicher Aktionen als .cpp-file; Sequencer in C++ programmieren
Aufwand zum Implementieren von Aktionen	○ hoch; Interrupts erfordern aufwändige Konstrukte	● einfach zu programmieren; viele Kommunikationsstrukturen zwischen Activities	● einfach zu lernen; verschiedene Sprachkonstrukte vereinigt; einfache, integrierte Entwicklungsumgebung	◑ mittel
Interaktivität	○ keine Interaktivität	● Interaktiv: online programmierbar und bedienbar	● Interaktiv: online programmierbar und bedienbar	○ nicht interaktiv: jede Änderung neu kompilieren
Transparenz	○ nicht transparent; schwer verständlich	● transparent; beliebig modifizierbar	● transparent; beliebig modifizierbar	● transparent; beliebig modifizierbar
Debugging	○ keine Möglichkeit	◑ keine Angabe	● Debugger	● Debugger
Integration in bestehende Steuerung	● Integrationsaufwand relativ gering aufgrund Erfahrung aus Simulation und Implementierung auf rob@work	○ tendenziell hoher Integrationsaufwand; z. Zt. nur als Binär-Code verfügbar; Schnittstellen unklar	● Integrationsaufwand tendenziell gering; Einbindung Python in C++-Steuerung einfach; Einbindung für MySQL-Datenbanken verfügbar	○ Spezifikation sehr aufwändig; reiner Implementierungsaufwand mittel; Risiko noch nicht sichtbarer Probleme
Einbindung eines Planungsmoduls	○ Modifikation der Planersoftware	● keine Modifikation notwendig	● keine Modifikation notwendig; Einbindung externer Prozesse sehr einfach	● keine Modifikation notwendig; Einbindung externer Prozesse möglich
Kommunikationsmechanismen	○ Daten über 3 Sprachen: JAM, Java, C++ mittels CORBA und JNI	◑ keine Angabe	● Interprozess- und Netzwerkkommunikationsmechanismen vorhanden	◑ keine Angabe
Mächtigkeit	○ nur einfache Datentypen; Rechnen kaum möglich	● Mächtigkeit von C	● mächtige Datenstrukturen und Sprachkonstrukte einer modularen, universellen Programmiersprache	◑ Mächtigkeit von C++; Nutzen von der Implementierung abhängig
Kapselung	● ja	● ja	● ja	○ keine
Reifegrad	○ kleiner Nutzerkreis; jüngste Version Nov'01	● großer Nutzerkreis; hoher Reifegrad	● große Community; ausgereift	○ Eigenentwicklung; keine Reife
Erfahrung	● Simulation erfolgreich; Test mit rob@work	◑ Inbetriebnahme Pioneer-Roboter	○ keine	● Fortgeschrittene C++ Programmierkenntnisse
Verfügbarkeit	● Freeware; open source	○ Bestandteil von Saphira Vertrieb mit Pioneer-Roboter	● Freeware; open source	● Entwicklungsumgebung vorhanden
allgemeine Vorteile	◑ keine Angabe	● wissenschaftlich ausgefeilt	● Produktiv; es gibt viele Libraries, z.B. für Matrizen, XML, GUI,…	● Einheitlich in C++; kein Programmiersprachenwechsel; kurze Einarbeitung
allgemeine Nachteile	○ Java als zusätzliche Programmiersprache	◑ keine Angabe	○ weitere Programmiersprache zu erlernen; i. a. langsam	◑ keine Angabe
Legende:	● positive Bewertung	◑ neutrale Bewertung	○ negative Bewertung	

Bild 5-4: Bewertung der Varianten für eine Realisierung des Ausführungsmoduls.

5.3.3 Auswahl und Schlussfolgerungen

Die Tabelle von Bild 5-4 enthält für die Skriptsprache Python die meisten positiven Bewertungen. Trotz der dagegen sprechenden Argumente der fehlenden Erfahrung und des Risikos, eine neue Sprache zu erlernen und erst im Nachhinein die tatsächlichen Probleme und Nachteile zu erkennen, wird für die weitere Entwicklung des Ausführungsmoduls die Skriptsprache Python verwendet.

Die Aktionen des Ausführungsmoduls werden als Funktionen programmiert. Zur Entwicklung unabhängiger Aktionen nach den Modellen von Bild 5-3 wird das Multithreading[6] des Betriebssystems genutzt.

5.4 Konzeption des Planungsmoduls

Das Planungsmodul fällt innerhalb der Kontrollarchitektur die Entscheidungen über die einzelnen Handlungsschritte zum Erfüllen vorgegebener Aufgaben. Es erstellt dynamisch die Pläne, welche das Ausführungsmodul abwickelt.

5.4.1 Planungsproblem

Übereinstimmend mit dem Konzept für das Ausführungsmodul (Kapitel 5.2) muss das Planungsmodul grobe Pläne ohne Wissen über deren detaillierte Ausführung erstellen. Die Pläne werden dann innerhalb der Aktionen verfeinert. Bei der Ausführung nutzen die Aktionen aktuelle Sensordaten und passen sich der aktuellen Umgebungssituation an.

Die Pläne beschreiben die kontinuierlichen Veränderungen in der Welt mit diskreten Situationsübergängen. Die Beschreibung ist symbolisch und unabhängig von realen Details der Umgebung. Das Planungsproblem muss deshalb umgebungsunabhängig sein. Für die symbolische Beschreibung des Hol- und Bringdienstes ist dabei nur von Interesse, welches Objekt in welcher Reihenfolge von wo nach wo transportiert wird. Es ist nicht von Belang, wie beispielsweise einem Hindernis ausgewichen wird. Ein derartiger Detaillierungsgrad ist in Bild 3-4 links mit den Schritten auf oberster Ebene gegeben: „Navigation“, „Objektaufnahme“, „Transport“ und „Objektabgabe“.

---

[6] Multithreading ist die Fähigkeit des Betriebssystems, mit nur einem Prozessor mehrere nebenläufige Threads und Prozesse (vgl. S.33) auszuführen.

5.4.2 Variantenbildung und Auswahl des Planungsmoduls

Zur Lösung des Planungsproblems sind in Kapitel 4.4 drei bewährte Methoden vorgestellt: „Problemlösen", „Planen" und „Entscheiden". Im Folgenden werden diese Varianten auf ihre Eignung in einem Planungsmodul für die Kontrollarchitektur von Roboterassistenten geprüft:

- **Problemlösen**
  Das Planungsproblem wird in einem Zustandsmodell mathematisch modelliert und mittels Suchalgorithmen wird ein optimaler Pfad im Zustandsraum bestimmt. Der Zustandsraum kann dabei sehr groß werden, wie das Beispiel in der Analyse (Kapitel 3.2.3) zeigt, und die Lösungsfindung sehr verlangsamen oder sogar unmöglich machen (vgl. Kapitel 4.4.1).

- **Planen**
  Das Planungsproblem wird in Regeln und Fakten mittels einer Repräsentationssprache beschrieben. Zur Lösung des Planungsproblems existiert eine Vielzahl von Algorithmen und Systemen (siehe Kapitel 4.4.2). Der Hol- und Bringdienst kann auf das Benchmark-Problem *Depots* (siehe Bild 4-11) abgebildet werden. In der *Depots*-Domäne werden ebenfalls Gegenstände aufgenommen, transportiert und wieder abgegeben.

- **Entscheiden**
  Das Planungsproblem wird wie beim Problemlösen über ein mathematisches Zustandsmodell beschrieben, das zusätzlich um Wahrscheinlichkeitsfunktionen in den Zustandsübergängen und um Beobachtungswahrscheinlichkeiten der Zustände erweitert wird. Nicht-deterministische Effekte bei der Aktionsausführung werden mit modelliert. Diese Methode ist jedoch aufgrund des Rechenaufwandes nicht praktikabel (vgl. Kapitel 4.4.3).

Mit diesen Argumenten ist das **Planen** die am besten geeignete Methode, um für das in dieser Arbeit entwickelte Konzept die geforderte Liste von Handlungsschritten zu generieren. Die Strukturierung des Kontrollflusses in Regeln, was der Roboterassistent tun kann und was deren Anwendung bewirkt, und in Fakten über den Weltzustand fördert außerdem die Wiederverwendbarkeit, d.h. die Modularität und Erweiterungsfähigkeit (Anforderung Ã1). Die Entscheidung, welche Regel wann angewendet wird, fällt das Planungssystem.

5.4.3 Schlussfolgerungen aus der Auswahl des Planungsmoduls

Es werden nur deterministische Effekte geplant. Dies entspricht den Annahmen über die Einsatzumgebung von Roboterassistenten (siehe Bild 3-2), jedoch können unter realen Umgebungsbe-

dingungen Störungen zu nicht-deterministischen Effekten führen. Für diese Fälle ist ein Feedback notwendig. Eine Diskrepanz zwischen einer korrekten Ausführung und der tatsächlichen Ausführung wird vom Ausführungsmodul erkannt und behandelt (siehe Kapitel 5.2.3).

Besteht Unsicherheit über den aktuellen Zustand – z.B. ob jemand in seinem Büro ist –, dann werden die wahrscheinlichsten Zustände modelliert: Leute sind normalerweise im Büro anwesend, falls nicht Mittagszeit ist. Wenn der Roboterassistent im Büro angekommen ist, stellt er fest, ob die Annahme verletzt ist. Gegebenenfalls triggert die Störungsbehandlung des Ausführungsmoduls eine Neuplanung an.

Planungssysteme arbeiten mit einer expliziten Umweltrepräsentation, d.h. diese Wahl für das Planungsmodul erfüllt Anforderung Ã3. Zur Erfüllung von Anforderung Ã5 müssen bei der Modellierung der Umwelt nicht nur einzelne Objekte, sondern auch Klassen von Objekten beschrieben werden (Anforderung A 19).

Das klassische Planen ist symbolisch mit diskreten Werten. Daraus folgt, dass auch die Umweltrepräsentation, welche die Planungsdomäne beschreibt, symbolisch und diskret sein muss. Dies deckt sich mit der Analyse der Einsatzumgebung von Roboterassistenten gemäß Bild 3-2.

5.5 Konzeption der Wissensbasis

Die technische Umsetzung der Umweltrepräsentation geschieht durch die Verwaltung von Fakten in einer Wissensbasis. Dazu werden effiziente Methoden zur Speicherung von Fakten und für den Zugriff auf diese benötigt. Außerdem sollen die enthaltenen Fakten immer konsistent sein sowie möglichst korrekt und so vollständig wie nötig den tatsächlichen Zustand der Welt widerspiegeln.

Fakten über die Umwelt benötigt sowohl das Planungsmodul als auch das Ausführungsmodul. Das Planungsmodul benötigt eine symbolische Weltbeschreibung, z.B. ob ein Objekt auf dem Tisch steht. Eine Ontologie für den Hol- und Bringdienst wird im folgenden Abschnitt 5.5.1 entwickelt. Außerdem benötigt das Planungsmodul eine prädikatenlogische Beschreibung der möglichen Aktionen. Diese wird in Abschnitt 5.5.2 entwickelt. Die Daten zur Effektoransteuerung durch das Ausführungsmodul, z.B. Objekteigenschaften, mit denen das Objekt zum Greifen identifiziert werden kann, sind spezifisch für die Realisierung des Roboterassistenten. Sie werden in Kapitel 6.3.2 behandelt.

5.5.1 Entwicklung einer Ontologie zur symbolischen Umweltrepräsentation

Die Ontologie für den Hol- und Bringdienst wird bottom-up entwickelt (vgl. Kapitel 4.5). Sie spezifiziert die Konzepte für die Modellierung der Umwelt aus Sicht des Roboters und ist damit die Basis für die Domänenbeschreibung des Planungsmoduls. Bei der Entwicklung wird die menschliche Welt und der menschliche Sprachgebrauch auf die Umweltmodellierung abgebildet, so dass die Kommandierung für den Menschen verständlich und intuitiv bleibt. Generell erhöht ein höherer Detaillierungsgrad der Modellierung die Handlungskompetenz. Er erhöht aber auch den Planungsaufwand. Deshalb wird die Ontologie so einfach wie möglich und nur so detailliert wie nötig gehalten. Von der KR-Ontologie aus Bild 4-14 genügen die konkreten Kategorien.

Roboterassistenten sind mobil und manipulieren Objekte (vgl. Bild 2-1). Es sind daher in erster Linie zwei Kriterien von essentiellem Interesse: **Mobilität** und **Manipulierbarkeit**. Diese Kriterien werden für die Kategorisierung der höchsten Ebene verwendet. Eine Entität x kann auf diese beiden Eigenschaften über die Prädikate

$$\texttt{mobil}(x) \qquad \textbf{(5-2)}$$

$$\texttt{manipulierbar}(x) \qquad \textbf{(5-3)}$$

geprüft werden. Daraus werden die vier in Bild 5-5 dargestellten Kategorien abgeleitet. Es wird im Bild das Symbol für Diskriminatoren von der UML (*Unified Modeling Language*) verwendet.

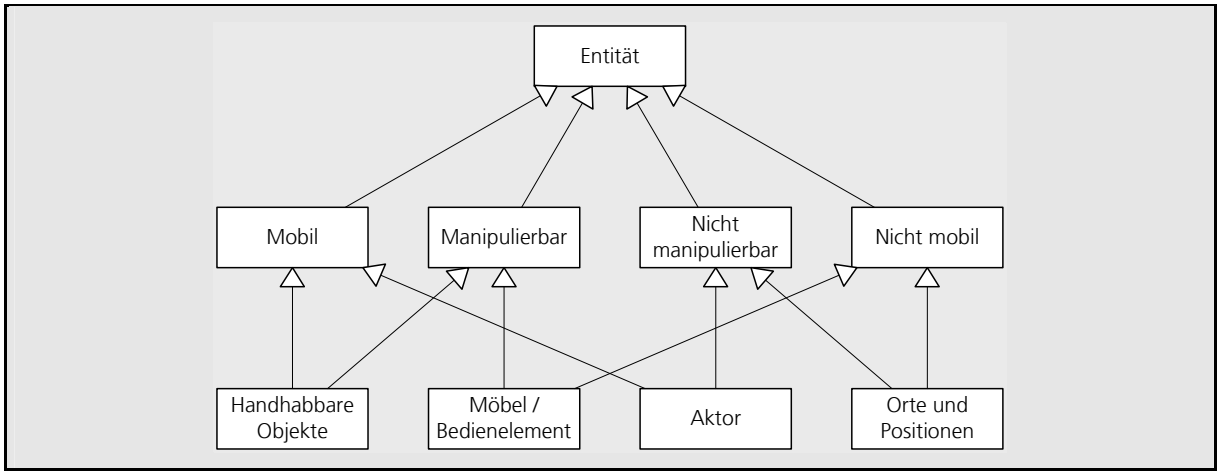


Bild 5-5: Hierarchie der obersten Kategorien in der Ontologie für den Hol- und Bringdienst von Roboterassistenten.

Handhabbare Objekte sind alles, was mobil ist, d.h. sich vom Roboterassistenten von Ort zu Ort bewegen lässt, und sich manipulieren lässt. Sie werden weiter unten in Bild 5-6 detailliert. Die Menge aller handhabbaren Objekte wird mit $\mathcal{O}$ bezeichnet.

Möbel und Bedienelemente sind nicht mobil, aber manipulierbar. Dies können z.B. Schalter sein oder Taster, um einen Aufzug zu rufen, es können (Schrank-)Türen sein (diese sind zwar beweglich, bleiben aber an ihrem Ort). Auch Wasserhähne fallen in diese Kategorie. Die Menge aller Möbel und Bedienelemente wird mit $\mathcal{M}$ bezeichnet.

Aktoren sind nicht manipulierbar, bewegen sich aber von Ort zu Ort. Diese sind entweder **Roboterassistenten** oder **Menschen**. Tiere würden zwar auch in diese Kategorie fallen, müssen aber nicht für den Roboter modelliert werden, da sie in der Regel keine Objekte manipulieren.

Die letzte Kategorie beschreibt **Orte und Positionen**, d.h. die räumlichen Zusammenhänge. Sie werden weiter unten in Bild 5-7 detailliert.

Die handhabbaren Objekte werden wie in Bild 5-6 dargestellt weiter kategorisiert:

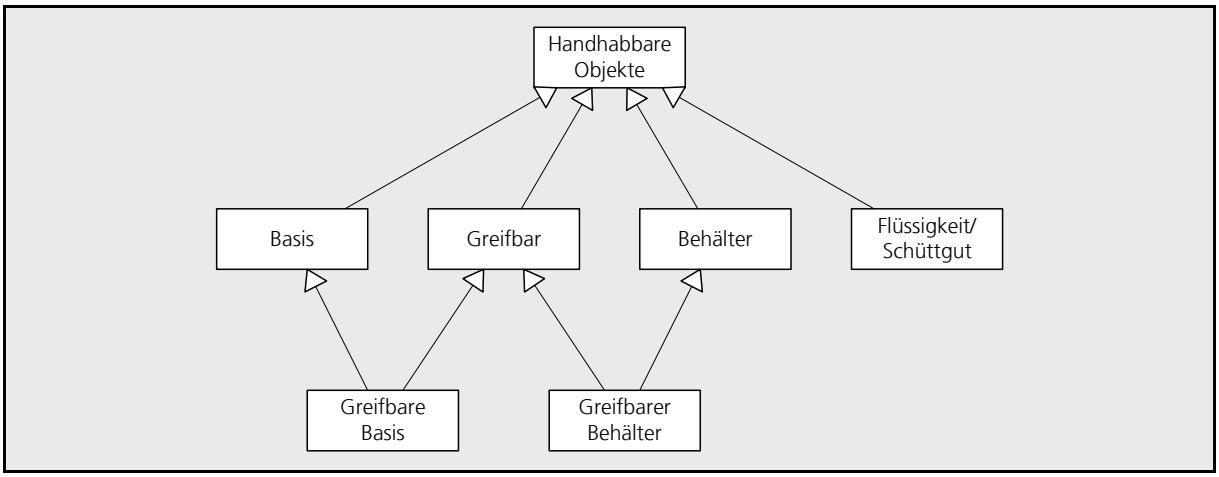


Bild 5-6: Unter-Kategorien von handhabbaren Objekten.

Handhabbare Objekte, die Roboterassistenten holen und bringen können sollen, sind:

- **Basis**, d.h. können als Unterlage dienen, worauf andere Objekte abgestellt werden können, z.B. Tische.
  Prädikat: `basis(`*x*`)` **(5-4)**
- **greifbar**, d.h. Objekte, die aufgenommen werden können, z.B. ein Stift.
  Prädikat: `greifbar(`*x*`)` **(5-5)**
- **greifbare Basis**, d.h. Objekte, die sowohl aufgenommen werden können als auch Unterlage sein können, wie z.B. Bücher oder Teller. Dies ist wichtig, um Objekte stapeln zu können.
  Prädikat: `greifbareBasis(`*x*`)` **(5-6)**
- **Behälter**, d.h. Objekte, in die etwas eingefüllt werden kann.
  Prädikat: `behaelter(`*x*`)` **(5-7)**
- **greifbare Behälter**, d.h. Behälter, die vom Roboterassistenten aufgenommen werden können, wie z.B. Tassen, Gläser, Flaschen,…
  Prädikat: `greifbarerBehaelter(`*x*`)` **(5-8)**
- **Flüssigkeit/Schüttgut**, d.h. jede Art von Flüssigkeit, Pulver und anderen Schüttgütern, die in Behälter gefüllt werden kann.
  Prädikat: `fluessig(`*x*`)` **(5-9)**

Die Modellierung der räumlichen Zusammenhänge in der Umwelt von Roboterassistenten wird in Bild 5-7 detailliert. Es werden die Symbole für die Aggregation und für die gerichtete Assoziation von der UML verwendet:

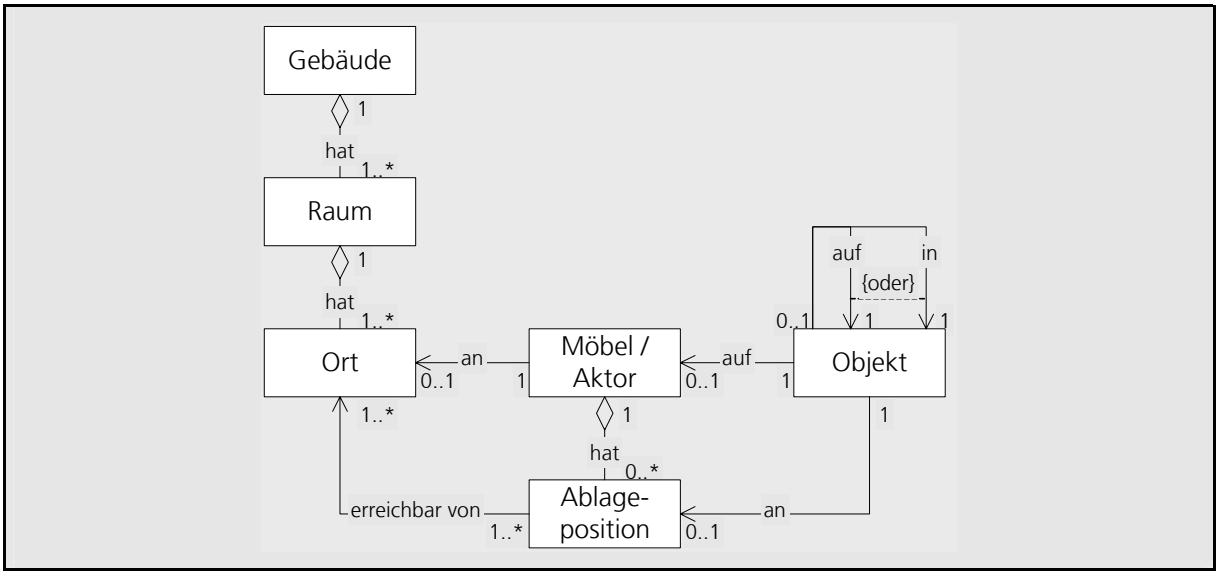


Bild 5-7: Ontologie der räumlichen Zusammenhänge für den Hol- und Bringdienst.

Der Roboterassistent soll sich in **Gebäuden** bewegen. Ein Gebäude besteht aus mindestens einem **Raum**. Die Menge aller Räume sei $\mathcal{R}$.

Innerhalb der Räume ist mindestens ein **Ort** definiert. Die Menge aller Orte sei $\mathcal{L}$ (wie englisch *location*). Die Orte definieren die topologischen Zusammenhänge innerhalb der Räume. Ein Ort kann entweder frei sein oder es kann sich dort ein **Möbel** oder ein **Aktor** befinden. In letzterem Fall gilt der Ort $x \in \mathcal{L}$ als besetzt, d.h. das Prädikat

`besetzt(`x`)` **(5-10)**

ist wahr. Befindet sich der Roboterassistent am Ort $x \in \mathcal{L}$, so ist

`robPos(`x`)` **(5-11)**

wahr.

Möbel haben verschiedene **Ablagepositionen**, auf die Objekte abgelegt werden können. Sie werden für jedes Möbel über das Prädikat

`hat-position(m,a)` **(5-12)**

definiert, wobei `a` Element der Menge der Ablagepositionen $\mathcal{A}_m$ auf dem Möbel $m \in \mathcal{M}$ ist. Die Menge aller Ablagepositionen wird mit $\mathcal{A}$ definiert:

$$\mathcal{A} = \bigcup_{m \in \mathcal{M}} \mathcal{A}_m \quad \textbf{(5-13)}$$

Jede Ablageposition ist von mindestens einem Orten aus **erreichbar**. Das Prädikat

`erreichbar(`x`,`l`)` **(5-14)**

ist wahr, wenn die Ablageposition $x \in \mathcal{A}$ von dem Ort $l \in \mathcal{L}$ aus erreichbar ist.

Objekte haben immer eine Position, **an** der sie sich befinden; entweder **auf** einem Möbel oder auf einem Aktor, z.B. **im Greifer** des Roboters oder in der Hand eines Menschen. Der Greifer bzw. die Hand sind Spezialfälle einer Ablageposition. Aktoren sind logisch betrachtet bewegliche Möbel, die ihren Ort wechseln können. Sie können gleichsam wie Möbel Objekte aufnehmen. Es werden die folgenden Prädikate definiert:

`an(`x`,`a`)` mit $a \in \mathcal{A}$ **(5-15)**

$\texttt{auf}(x,b)$ mit $\texttt{basis}(b)$ **(5-16)**

$\texttt{imGreifer}(x)$ **(5-17)**

Objekte können gestapelt werden, wenn das untere eine Basis ist. In diesem Fall befindet sich ein Objekt **auf** einem anderen Objekt. Für Flüssigkeiten gilt, dass sich ein Objekt (die Flüssigkeit) **in** einem anderen Objekt (Behälter) befindet.

$\texttt{in}(x,o)$ mit $\texttt{behaelter}(o)$ **(5-18)**

Nachbarschaftsbeziehungen (neben, vor, hinter, …) werden für Objekte nicht modelliert. Sie ergeben sich aus den Positionen und sind abhängig von der Perspektive. Zeitliche Relationen werden ebenfalls nicht modelliert, weil immer der augenblickliche Ist-Zustand von Interesse ist. Eine zeitliche Veränderung wird beim Planen berücksichtigt.

5.5.2 Modellierung der Aktionen

Die beim Hol- und Bringdienst zu planenden Aktionen sind in Kapitel 5.4.1 hergeleitet: es sind die „Navigation" bzw. der „Transport", die sich aus „Fahren" (englisch: *move*) ergeben, die „Objektaufnahme" (englisch: *grasp*) und die „Objektabgabe" (englisch: *drop*). Sie werden in Bild 5-8 prädikatenlogisch modelliert:

Name:	$\texttt{move(y)}$
Variablen:	$x,y \in \mathcal{L}$
Vorbedingung:	$\texttt{robPos}(x) \wedge \neg\texttt{besetzt}(y) \wedge (x \in \mathcal{L}) \wedge (y \in \mathcal{L})$
Effekt:	$\neg\texttt{robPos}(x) \wedge \texttt{robPos}(y)$
Name:	$\texttt{grasp(x)}$
Variablen:	$\texttt{x},z_1,z_2 \in \mathcal{O}, \texttt{y} \in \{\tilde{y} \mid \texttt{basis}(\tilde{y})\}, \texttt{l} \in \mathcal{L}, \texttt{a} \in \mathcal{A}$
Vorbedingung:	$\texttt{robPos(l)} \wedge \texttt{erreichbar(a,l)} \wedge \texttt{auf(x,y)} \wedge \texttt{an(x,a)} \wedge (\neg\exists z_1 : \texttt{imGreifer}(z_1)) \wedge (\neg\exists z_2 : \texttt{auf}(z_2,\texttt{x}))$
Effekt:	$\neg\texttt{auf(x,y)} \wedge \neg\texttt{an(x,a)} \wedge \texttt{imGreifer(x)}$
Name:	$\texttt{drop(x)}$
Variablen:	$\texttt{x,z} \in \mathcal{O}, \texttt{y} \in \{\tilde{y} \mid \texttt{basis}(\tilde{y})\}, \texttt{l} \in \mathcal{L}, \texttt{a} \in \mathcal{A}$
Vorbedingung:	$\texttt{robPos(l)} \wedge \texttt{erreichbar(a,l)} \wedge (\neg\exists \texttt{z} : \texttt{an(z,a)}) \wedge \texttt{imGreifer(x)}$
Effekt:	$\texttt{auf(x,y)} \wedge \texttt{an(x,a)} \wedge \neg\texttt{imGreifer(x)}$

Bild 5-8: Modellierung der Aktionen für den Hol- und Bringdienst.

Bei der Objektaufnahme bzw. -abgabe kann unterschieden werden, ob das Objekt auf ein anderes Objekt in der Umwelt abgelegt wird, oder ob es auf den Roboter selbst abgelegt wird. Der Unterschied besteht darin, dass sich die Objekte auf dem Roboter mit bewegen, wohingegen alle anderen Objekte als stationär betrachtet werden.

Bei dieser Modellierung der Fahr-Aktion wird davon ausgegangen, dass jeder Ort y von jedem Ort x aus erreichbar ist. Die Navigation soll nicht symbolisch geplant werden, sondern von der Robotersteuerung übernommen werden. Beim Fahren können Türen den Weg versperren. Türen und das Öffnen bzw. Schließen könnten ebenfalls in die Ontologie aufgenommen werden. Tatsächlich ändert sich der Türzustand aber, ohne ihn überwachen zu können, z.B. kann ein Windstoß die Tür schließen oder ein Mensch sie öffnen. Der Roboter muss also in jedem Fall prüfen, ob seine Informationen aktuell sind. Deshalb wird hier die Variante verfolgt, Türen nicht zu modellieren. Wird während der Fahrt der geplante Pfad zwischen Räumen durch eine Tür versperrt, tritt eine Störsituation in der Fahr-Aktion auf. An dieser Stelle werden reaktiv die Aktionen „Tür prüfen“ und gegebenenfalls „Tür öffnen“ aufgerufen.

5.6 Konzeption zur Wartung der Wissensbasis

Der Roboterassistent kann nur dann seine Handlungen korrekt planen, wenn sein symbolisches Modell von der Umwelt aktuell ist. Eine Methode zur Wartung und Pflege der Wissensbasis ist deshalb wichtiger Bestandteil der Kontrollarchitektur für Roboterassistenten.

Die Aufgabe der Wartung besteht also darin, die Wissensbasis mit Symbolen zu versorgen. Diese Aufgabe teilt sich in zwei Herausforderungen bei der Realisierung: zum einen Symbole kontinuierlich zu sammeln, d.h. aus Sensordaten und aus eigenen Handlungen zu generieren, und zum anderen sie richtig, d.h. widerspruchsfrei und abhängig vom Aufgabenkontext, zuzuordnen.

Ein zentrales Modul zur Wartung ist ungeeignet, da Informationen von der Umwelt über verschiedene Kanäle fließen. Es werden deshalb folgende Konzepte gleichzeitig verfolgt:

Die Effekte des Handelns werden automatisch in die Wissensbasis eingetragen. Bei der Realisierung der Aktionen im Ausführungsmodul trägt jede Aktion beim Beenden ihren Effekt ein; bei einem erfolgreichen, regulären Beenden den regulären Effekt, bei einem Abbruch aufgrund einer Störsituation trägt die Störungsbehandlung den individuellen Effekt der Störung ein. Wird beispielsweise ein Objekt x vom Tisch aufgenommen, so wird der Platz auf dem Tisch frei und das Prädikat `imGreifer(`x`)` wahr gesetzt. Fällt das Objekt dabei herunter, so wird das Prädikat `imGreifer(`x`)` wieder falsch.

Aktionen zur aktiven Wahrnehmung stützen die Wartung der Wissensbasis. Sie werden aktiviert, um die symbolischen Fakten mit der Realität vor einer Handlung zu überprüfen. Dabei wird vorhandenes Kontextwissen genutzt. Zum Beispiel werden die Algorithmen zur Objekterkennung darauf spezialisiert[7], ein Objekt x zu erkennen, wenn der Roboterassistent als nächstes das Objekt x greifen soll. Es ist dann nicht von Belang, ob noch andere Objekte $\overline{x}$ im Sensorbereich sind. Wird x nicht erkannt, so wird eine Störungsmeldung an die aufrufende Aktion gegeben.

Neue Fakten können in direktem Widerspruch zu den bestehenden Annahmen stehen oder Inkonsistenzen ergeben sich indirekt relativ zu vorhandenen Schlussregeln. Deshalb müssen die Fakten nicht nur gespeichert, sondern aktiv geprüft werden, um Inkonsistenzen und ungültige Schlüsse zu erkennen und gegebenenfalls zu beheben. Eine einfache Methode zur Beseitigung von direkten Widersprüchen ist beispielsweise die Faustregel, dass aktuellere Informationen verlässlicher sind, da sich die Welt mit der Zeit ändert. Andere Inkonsistenzen lassen sich auflösen, indem Annahmen eine Gewissheit zugeordnet wird, so dass weniger sichere Annahmen bei einem Widerspruch aufgegeben werden. In diese Gewissheit können viele verschiedene Faktoren mit einbezogen werden, wie die Aktualität, die Rechtfertigung durch andere Annahmen oder auch die Beständigkeit von Tatsachen eines bestimmten Typs zusammen mit der Zeit der letzten Bestätigung.

Zur Wartung der Umweltinformationen kann außerdem der Benutzer die Wissensbasis manuell editieren. Dies ist notwendig, wenn z.B. neue Objekte in die Umwelt eingebracht werden oder wenn der Benutzer die Umwelt selbst verändert hat. Das manuelle Editieren hilft dann Störungen zu vermeiden, wenn der Roboterassistent beispielsweise Objekte an Orten vermutet, wo keine mehr sind.

5.7 Konzeption der Mensch-Maschine-Schnittstelle

Über eine Mensch-Maschine-Schnittstelle der Kontrollarchitektur interagieren Roboterassistent und Mensch informatorisch miteinander. Die Kommunikation findet sowohl bei der kooperativen Aufgabenausführung als auch bei der Kommandierung sowie beim Wechsel von Betriebsmodi und beim Editieren der Wissensbasis statt. Bild 5-9 zeigt verschiedene Varianten für die Ausprägung der Kommunikationskanäle mit einer Bewertung ihrer Eignung.

[7] Ein generelles Situationsverstehen ist mit heutigen Mitteln nicht möglich. Eine dem Menschen ähnliche Sehfähigkeit wird noch 30 Jahre Forschungsarbeit erfordern (Brooks 2002, S.74) (Levi et al. 2003, S.297).

Kommunikationsrichtung	Kommunikationsinhalt	Sprachein- / -ausgabe	Touchscreen	Gestik
Mensch ↓ Roboterassistent	Befehle	●	●	◑
	Wissensbasis editieren	●	●	◑
	Hilfestellung für den Roboterassistenten	◑	●	◑
	Betriebsmoduswechsel	●	●	○
Roboterassistent ↓ Mensch	Statusmeldungen	●	●	○
	Störungsmeldungen	◑	●	○
	Rückfragen an den Menschen	●	●	◑
	Informationsdienste	●	●	○

Legende: ● gut geeignet ◑ weniger gut geeignet ○ nicht geeignet

Bild 5-9: Eignung verschiedener Kommunikationskanäle für die Mensch-Maschine-Schnittstelle.

Ein Touchscreen eignet sich universell für alle Eingaben am und Ausgaben vom Roboterassistent. Sprachein- und -ausgaben sind zwar sehr intuitiv und menschlich, haben jedoch den Nachteil, dass das gesprochene Wort nach dem Aussprechen zeitlich vergangen ist. Informationen, die länger abrufbar sein müssen, wie z.B. Störungsmeldungen, müssen deshalb visualisiert werden.

Natürliche Gesten sind für die Kommandierung sehr interessant, ihr Einsatzgebiet ist jedoch sehr eingeschränkt. Zum Beispiel lässt sich über eine Kombination aus Sprachkommando und Geste ein Objekt identifizieren („diese Tasse“) oder ein Kommando geben („fahre dort hin“). Über reine Zeigegesten kann jedoch nicht kommuniziert werden[8].

[8] Eine Ausnahme bildet die Taubstummensprache, die hier jedoch nicht betrachtet wird.

5.8 Zusammenführung der Teilkonzepte zur Kontrollarchitektur

5.8.1 Gesamtsystem

Die Teilkonzepte aus den Abschnitten 5.2 bis 5.7 werden im Folgenden zusammengeführt. Kern der Architektur ist das Ausführungsmodul. Um dieses werden die anderen Module sternförmig angeordnet. Das heißt, es gibt eine zentrale Verhaltenskontrolle, die sich der Dienste anderer Teilsysteme bedient. Jedes Teilsystem ist über eine einfache Schnittstelle angebunden. Es gibt keine weiteren Schnittstellen zwischen den Teilsystemen. Bild 5-10 zeigt die Kontrollarchitektur. Sie ist dort als Drei-Ebenen-Architektur dargestellt.

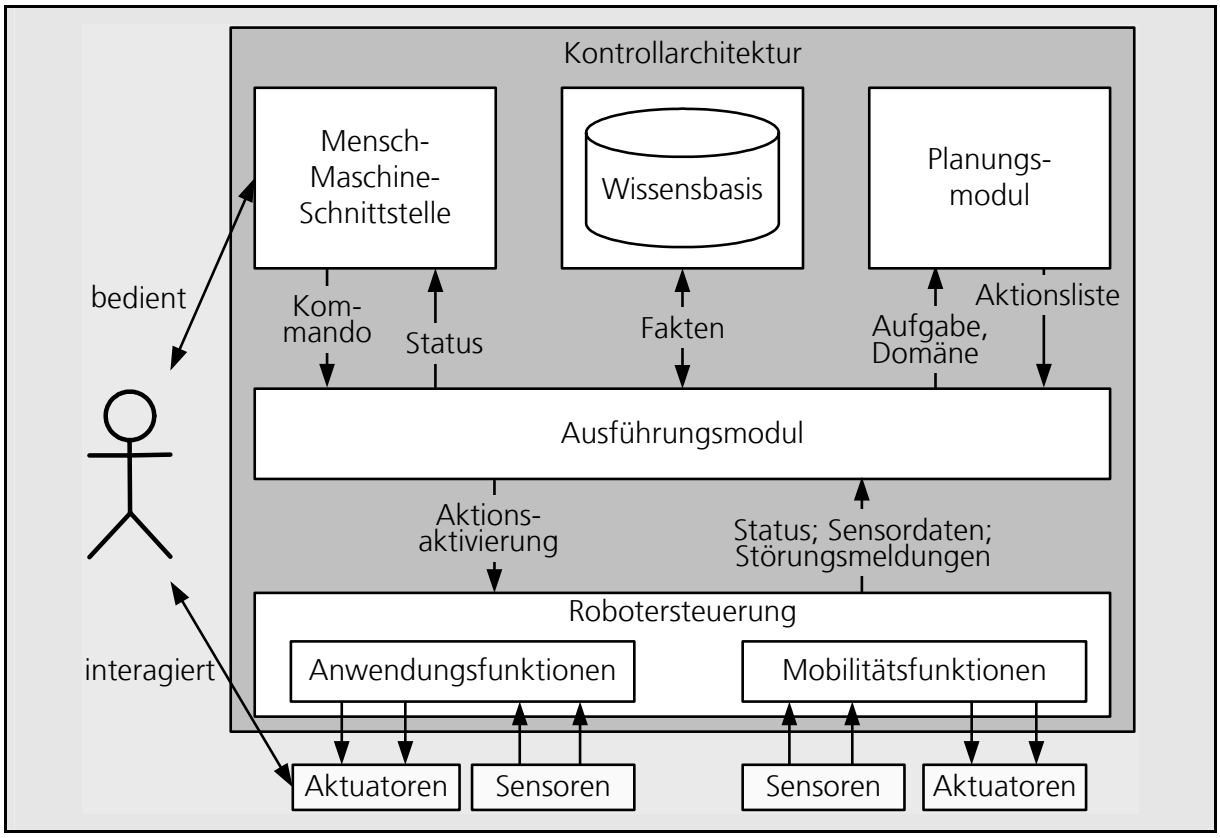


Bild 5-10: Kontrollarchitektur für Roboterassistenten (Hans, Schraft 2003).

Der Mensch bedient den Roboterassistenten über die Mensch-Maschine-Schnittstelle. Die Kommandos werden dort vorverarbeitet und an das Ausführungsmodul weitergegeben.

Es gibt keine direkte Verbindung zwischen Mensch-Maschine-Schnittstelle und Wissensbasis zum Editieren. Dies hat den Vorteil, dass über das Ausführungsmodul noch weitere, zur internen

Weiterverarbeitung benötigte Daten zu den Eingaben des Benutzers hinzugefügt werden können. Außerdem ist die Wissensbasis so gekapselt, dass sie durch eine andere technische Realisierung ausgetauscht werden kann, ohne die Mensch-Maschine-Schnittstelle anpassen zu müssen.

Das gleiche gilt für das Planungsmodul: Es hat weder eine Verbindung zur Mensch-Maschine-Schnittstelle noch zur Wissensbasis. Planungsaufgaben werden nur vom Ausführungsmodul erteilt und die Fakten aus der Wissensbasis werden im Ausführungsmodul zu einer Domänenbeschreibung zusammengefügt. Dies hat den Vorteil, dass das Planungsmodul ebenfalls durch eine andere technische Realisierung ausgetauscht werden kann, ohne die anderen Module anpassen zu müssen. Außerdem kann im Ausführungsmodul für jede Planungsaufgabe die Domäne spezifisch und nur mit den für die aktuelle Aufgabe relevanten Daten zusammengestellt werden. Dadurch kann die Wissensbasis sehr umfangreich und mächtig sein, das Planungsmodul arbeitet aber trotzdem effizient, weil es nur den relevanten Teil der Fakten zur Verfügung bekommt.

Die Robotersteuerung ist die Schnittstelle zu den Sensoren und Aktuatoren des Roboterassistenten. In ihr sind die Aktionen, die das Ausführungsmodul aktiviert bzw. deaktiviert, als hardwarenahe Steuerungs- bzw. Regelungsprozesse mit ggf. zeitkritischer Rückführung realisiert. Diese Aktionen haben definierte Schnittstellen zum Ausführungsmodul. Wenn also eine andere Robotersteuerung die gleichen Schnittstellen der Aktionen aufweist, kann die Kontrollarchitektur sehr einfach auf eine andere Roboterhardware portiert werden.

5.8.2 Regelablauf

Das Zusammenwirken der Teilsysteme im regulären Ablauf bei der Aufgabenausführung zeigt Bild 5-11. Zur Veranschaulichung wird die Darstellung von Sequenzdiagrammen der UML gewählt, obwohl es sich bei den Modulen nicht um Objekte im Sinne der UML handelt.

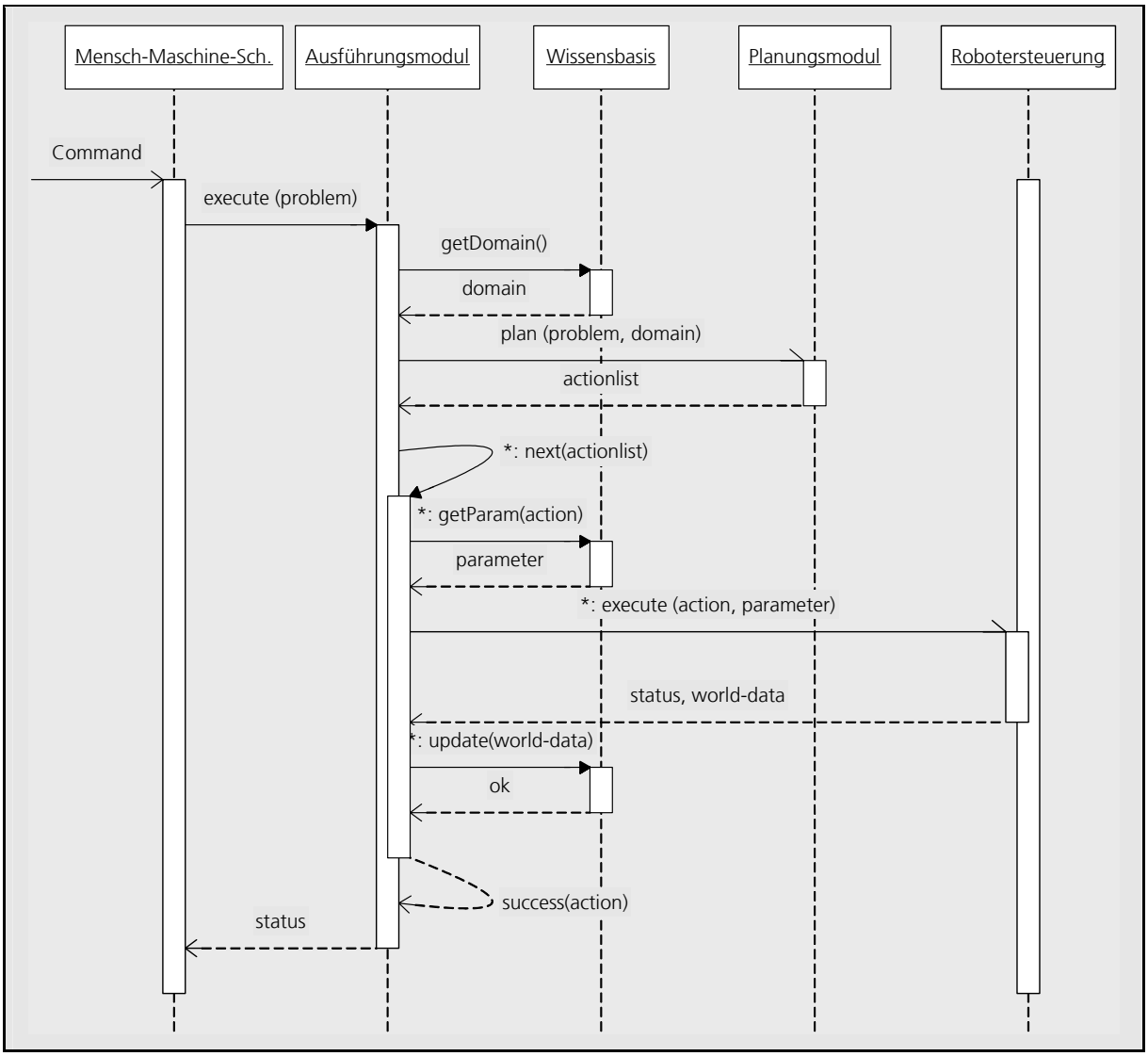


Bild 5-11: Reguläre Abfolge der Modulaufrufe bei der Aufgabenausführung.

Der Steuerungsfokus liegt die meiste Zeit beim Ausführungsmodul. Die Mensch-Maschine-Schnittstelle ist parallel aktiv für den Fall, dass neue Kommandos eingegeben werden. Auch die Robotersteuerung ist immer aktiv, sobald die Hardware des Roboterassistenten initialisiert wurde.

Nach Eingabe eines Kommandos zur Aufgabenausführung werden von der Wissensbasis die notwendigen Fakten zur Domänenbeschreibung abgefragt. Mit der Aufgabenstellung und der Domänenbeschreibung wird ein Plan, d.h. eine partiell geordnete Liste von Aktionen, vom Planungsmodul generiert. Diese wird iterativ vom Ausführungsmodul abgearbeitet. Dabei rufen die Aktionen rekursiv weitere Aktionen auf. Die Aktionen des Ausführungsmoduls rufen ggf. Parameter aus der Wissensbasis ab und aktivieren parametrierte Aktionen in der

Robotersteuerung. Am Ende jeder Aktion aktualisiert diese ihren Effekt auf die Umwelt in der Wissensbasis.

5.8.3 Erfüllung der Anforderungen

In Bild 5-12 wird das Konzept für die Kontrollarchitektur überprüft, ob es die in Kapitel 3 erarbeiteten Anforderungen (vgl. Bild 3-6) erfüllt.

Anfor-derung	Erfül-lung	Begründung
Ã1 Modularität und Erweiterungsfähigkeit		
(A 16)	✓	Die Handlungen sind aus kombinierbaren Aktionen gestaltet.
(A 17)	✓	Das Spektrum der Fähigkeiten ist erweiterbar durch Hinzufügen neuer Aktionen.
(A 20)	✓	Die Roboterkontrollarchitektur ist modular aufgebaut.
Ã2 Schnittstellen zur Robotersteuerung		
(A 1)	✓	Die Kontrollarchitektur unterstützt Navigationssysteme in der Robotersteuerung.
(A 2)	✓	Die Kontrollarchitektur unterstützt die Ansteuerung eines Manipulatorsystems in der Robotersteuerung.
(A 3)	✓	Die Kontrollarchitektur beinhaltet eine Mensch-Maschine-Schnittstelle zur Kommunikation.
(A 6)	✓	Die Kontrollarchitektur unterstützt auch zeitkritische Funktionen in der Robotersteuerung.
(A 12)	✓	Der Roboterassistent kontrolliert seine Handlungen in der Robotersteuerung in Echtzeit.
Ã3 Explizite Umweltrepräsentation		
(A 8)	✓	Dem Roboterassistenten genügt zum Handeln eine partielle Umweltbeobachtungsfähigkeit.
(A 9)	✓	Die Kontrollarchitektur hat eine interne Repräsentation der Umwelt in der Wissensbasis.
(A 10)	✓	Die interne Repräsentation der Umwelt wird anhand von Sensordaten aktualisiert.
(A 5)	✓	Die Wissensbasis enthält sowohl Objektindividuen als auch Klassenzuordnungen der Elemente.
Ã4 Adaptierende Verhaltenskontrolle		
(A 11)	✓	Der Roboterassistent passt seine Handlungen eine sich verändernde Umwelt an.
(A 13)	✓	Die Aktionen des Roboterassistenten werden auf Erfolg überwacht.
(A 14)	✓	Jede Aktion hat ein definiertes Ende, wenn sie erfolgreich ausgeführt worden ist.
(A 15)	✓	Die Kontrollarchitektur unterstützt eine Methode zur Störungsbehandlung.
Ã5 Planungskomponente		
(A 7)	✓	Die Kontrollarchitektur beinhaltet ein Planungsmodul als Komponente zur Entscheidungsfindung.
(A 4)	✓	Der Roboterassistent plant aus einfachen Befehlen selbstständig (komplexe) Handlungen.
(A 19)	✓	Planungssysteme können mit Klassen von Objekten planen.
(A 18)	✓	Planungssysteme sind mächtig, um verschachtelte Aufgabenausführungen zu planen.

Bild 5-12: Überprüfung der Anforderungen für die entwickelte Kontrollarchitektur.

6 Realisierung der Kontrollarchitektur

Zur Erprobung des in den vorangegangenen Kapiteln entwickelten Konzepts wird die Kontrollarchitektur auf einem Demonstrator realisiert. Als Hardware dient der Roboterassistent Care-O-bot® II. Das Ausführungsmodul wird mit der Skriptsprache Python (vgl. Kapitel 5.3.3) implementiert und es werden die Robotersteuerung und die weiteren Module Planungsmodul, Wissensbasis und Mensch-Maschine-Schnittstelle daran angebunden.

6.1 Roboterassistent Care-O-bot® II

Care-O-bot® II (Bild 6-1) ist ein Prototyp eines Roboterassistenten für Haushalt und Pflege, der Hilfsbedürftigen zukünftig einfache Arbeiten im Haushalt abnehmen soll. Care-O-bot® II wurde der Öffentlichkeit erstmals auf der Hannover Messe Industrie 2002 vorgestellt (Hans et al. 2002).

Bild 6-1: Roboterassistent Care-O-bot® II.

Care-O-bot® II entspricht der Definition eines Roboterassistenten von Kapitel 2.1. Er verfügt über eine mobile Plattform, einen Manipulator mit Greifer, über Sensorik zur Umgebungserfassung und über ein mobiles Bedienpanel zur Kommunikation. Der mechanische Aufbau mit seinen Teilsystemen und die Rechnerarchitektur werden in den folgenden Abschnitten kurz vorgestellt.

6.1.1 Elektromechanische Komponenten

Care-O-bot® II ist mechanisch aus einer mobilen Plattform mit Differenzialkinematik und einer darauf aufgesetzten, oberen Ebene aufgebaut (Bild 6-2). Die mobile Plattform dient den Mobilitätsfunktionen, die obere Ebene den Anwendungsfunktionen. Die Trennung in mobile Plattform und obere Ebene ist Teil des modularen Konzeptes: beide Teilsysteme lassen sich unabhängig voneinander betreiben und entwickeln.

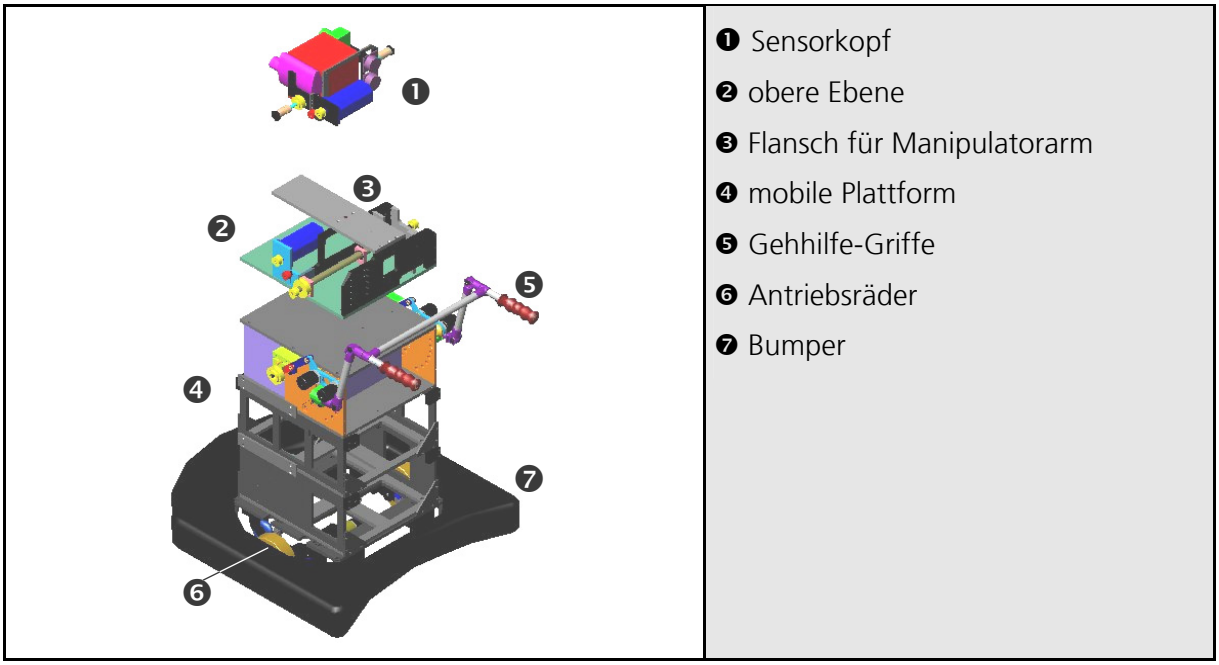


Bild 6-2: Mechanik des Roboterassistenten Care-O-bot® II.

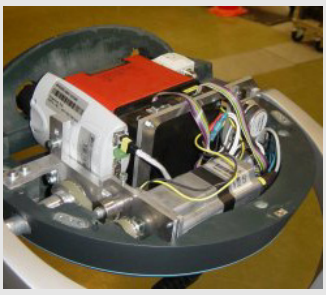
Mobilität: mobile Plattform

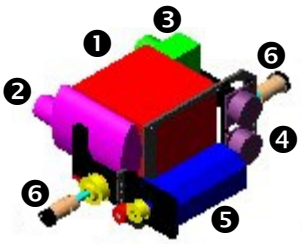
Die mobile Plattform kann über Lasernavigation selbstständig navigieren und Hindernissen ausweichen. Neben der reinen Mobilitätsfunktion dient sie auch als Stütze und intelligente Gehhilfe, d.h. der Mensch kann mit der Plattform über zwei ausfahrbare Gehhilfe-Griffe (siehe Bild 6-2,❺) physisch interagieren. Es werden dem Benutzer dabei verschiedene Stufen der Autonomie angeboten: Im Modus „Fahren nach Führung“ kann der Benutzer die Plattform in eine gewünschte Richtung „schieben“, wobei Hindernisse erkannt und umfahren werden. Im Modus „Fahren nach Plan“ folgt der Benutzer der Plattform auf einem berechneten, optimierten Weg zu einem vorher spezifizierten Ziel. Die Funktion „intelligente Gehhilfe“ (Bild 6-3) stellt gegenüber konventionellen Gehhilfen eine Verbesserung bezüglich Sicherheit und Komfort dar. Sie wird über das Bedienpanel konfiguriert.

Bild 6-3: Care-O-bot® II als intelligente Gehhilfe.

Umgebungserfassung: Sensorkopf

Zur Umgebungserfassung verfügt Care-O-bot® II über einen neigbaren Kopf mit 2-D-Laserscanner und Kameras (Bild 6-4). Durch das Neigen des 2-D-Laserscanners wird ein dreidimensionales Entfernungsbild der Umgebung erstellt. Diese Daten werden zur Berechnung kollisionsfreier Trajektorien für die Manipulation genutzt. Eine der beiden Kameras ist eine WebCam, deren Bilder auf das mobile Bedienpanel übertragen werden. Der Benutzer erhält so einen Eindruck, was der Roboter „sieht". Die andere Kamera dient der Bildverarbeitung, insbesondere zur Identifikation zu manipulierender Objekte. Im Sensorkopf sind außerdem noch Lautsprecher für eine Sprachausgabe montiert.



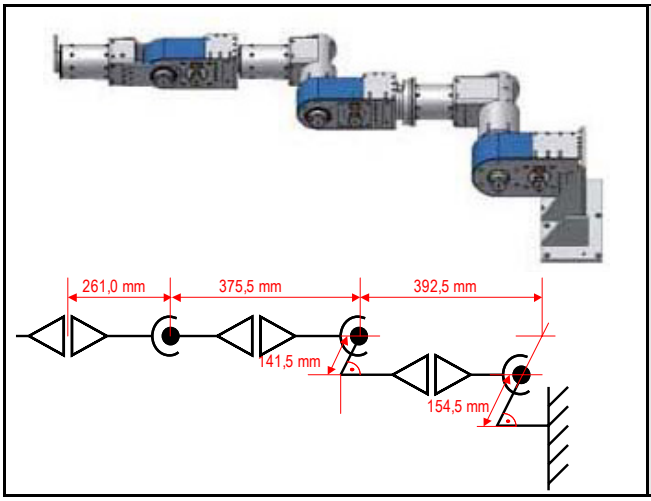


❶ 2-D-Laserscanner (Leuze)
❷ WebCam (Axis)
❸ CCD-Kamera (THE IMAGING SOURCE)
❹ Lautsprecher
❺ Antriebsmotor für Neigbewegung
❻ Drehachse

Bild 6-4: Sensorkopf von Care-O-bot® II (links: Foto; Mitte: Konstruktion).

Manipulation: Manipulator, Greifer und Handsensorik

Für Care-O-bot® II wurde ein 6-achsiger Manipulator entwickelt (Bild 6-5). Sein Arbeitsraum ist so dimensioniert, dass der Roboterassistent sowohl auf den Boden als auch in ein hohes Regal greifen kann. Seine Nutzlast ist für typische Gewichte von Haushaltsgegenständen ausgelegt.



Technische Daten (Neobotix 2003):

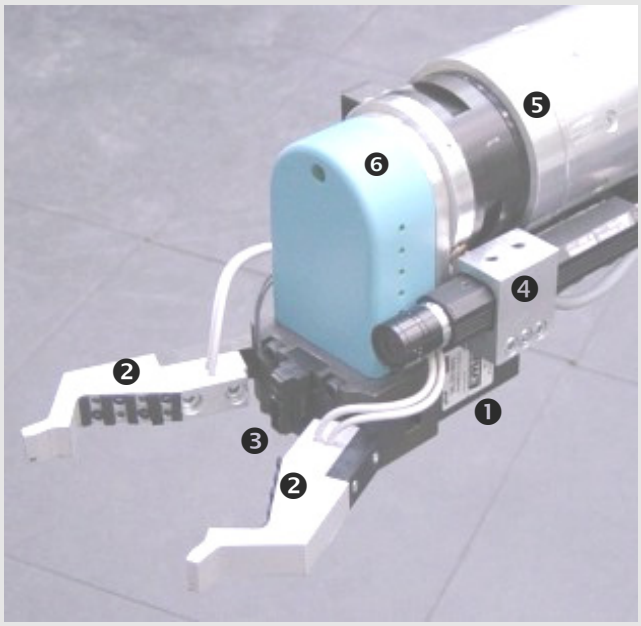
- 6-Achs-Kinematik
- Arbeitsraum: ca. 1 m Radius
- Nutzlast: ca. 2…4 kg
- Masse: ca. 25 kg

Bild 6-5: Manipulator von Care-O-bot® II.

Zum Greifen wurde ein elektrischer Zweifinger-Winkelgreifer mit Positions- und Stromfeedback angefertigt (Bild 6-6). Der Gesamtdrehwinkel beträgt 180°, d.h. 90° pro Finger. Die gummierten, prismatischen Finger mit verengendem Fortsatz erlauben sowohl den Griff großer Gegenstände als auch kleiner Gegenstände in den Fingerspitzen. Die Greifkraft ist für Massen bis 1,5 kg ausgelegt.

Folgende Sensorsysteme unterstützen die Handhabung von Objekten und die Interaktion mit dem Menschen (vgl. Bild 6-6):

- In den Fingern ist je ein Array aus vier Paaren von **Reflexlichttastern** zur Objektdetektion. Damit lässt sich der Griff überwachen, ob ein Objekt innerhalb der Finger ist bzw. ob es verloren gegangen ist, und es lässt sich beim Zugreifen eine seitliche Positionsungenauigkeit ausregeln. Durch Verändern des Öffnungswinkels der Finger wird die Orientierung der Sensoren für verschiedene Detektionsaufgaben ausgerichtet.



Technische Daten des Greifers (❶):

- Zweifinger-Winkelgreifer (je 90°)
- Positions- und Stromfeedback

Technische Daten der Sensoren:

- Array aus vier Paaren Reflexlichttaster pro Finger (❷): 0…5 cm
- Optischer Distanzsensor (❸): 4…30 cm
- CCD-Handkamera (❹)
- Kraft-Momenten-Sensor (❺)

Mikrocontroller

- Zur Sensordatenvorverarbeitung (❻)

Bild 6-6: Zweifinger-Winkelgreifer mit Multisensorik.

- Ein **optischer Distanzsensor** zwischen den Fingern erlaubt beim Zugreifen frühzeitig eine Abstandsmessung zu Objekten.
- Über eine **Handkamera** kann die Objektidentifikation und Positionsregelung unterstützt werden.
- Mit einem **Kraft-Momenten-Sensor** werden externe Kräfte, die auf den Greifer wirken, gemessen.
- Der Greifercontroller liefert ein **Positions- und Stromfeedback**, über das sich die Winkelposition der Finger und die Greifkraft ermitteln lassen.

Kommunikation: Mobiles Bedienpanel

Zur informatorischen Mensch-Maschine-Interaktion dient ein mobiles Bedienpanel (Bild 6-7). Eine Funkverbindung ermöglicht die Kommunikation, auch wenn sich der Roboter in einem anderen Raum aufhält. Über Menüs kann der Mensch Kommandos an den Roboter auswählen. Auf das Panel werden das aktuelle Kamerabild und Statusmeldungen des Roboters übertragen, so dass der Benutzer immer überwachen kann, was der Roboter gerade ausführt.

	Technische Daten des Bedienpanels: - Modell: Fujitsu Stylistic 3500X Pen Tablet - Auflösung: 1024 x 768 Pixel - Graphik: 3-D-Beschleunigung, 4 MB VRAM - Stromversorgung: 3100 mAh LiIon-Batterie - Abmessungen: 280 mm x 215 mm x 27,4 mm - Gewicht: 1,46 kg - Datenübertragung: IEEE 802.11b Wireless-LAN

Bild 6-7: Mobiles Bedienpanel von Care-O-bot® II.

Die Halterung für das Bedienpanel an Care-O-bot® II enthält eine Docking-Station zum Aufladen des Akkus.

6.1.2 Rechnerarchitektur

Bild 6-8 zeigt die Rechnerarchitektur von Care-O-bot® II. Sie ist übereinstimmend zur Mechanik strukturiert: Der Roboter besteht aus zwei autonomen Systemen, der mobilen Plattform mit den Gehhilfen und der oberen Ebene mit Manipulator und Sensorkopf. Beide Systeme arbeiten vollständig autonom mit eigenem PC, Stromversorgung und Not-Aus-Schaltkreis.

Der PC für die mobile Plattform steuert die autonome Navigation des Roboters sowie die Höhenverstellung der Gehhilfen. Der PC der oberen Ebene kontrolliert den Manipulatorarm und den Greifer. Außerdem verarbeitet er die Daten aus dem Sensorkopf und steuert dessen Neigung.

Die beiden Rechner sowie die WebCam im Sensorkopf sind über Ethernet miteinander verbunden. Das Bedienpanel ist über Wireless-LAN ebenfalls an das Ethernet angebunden.

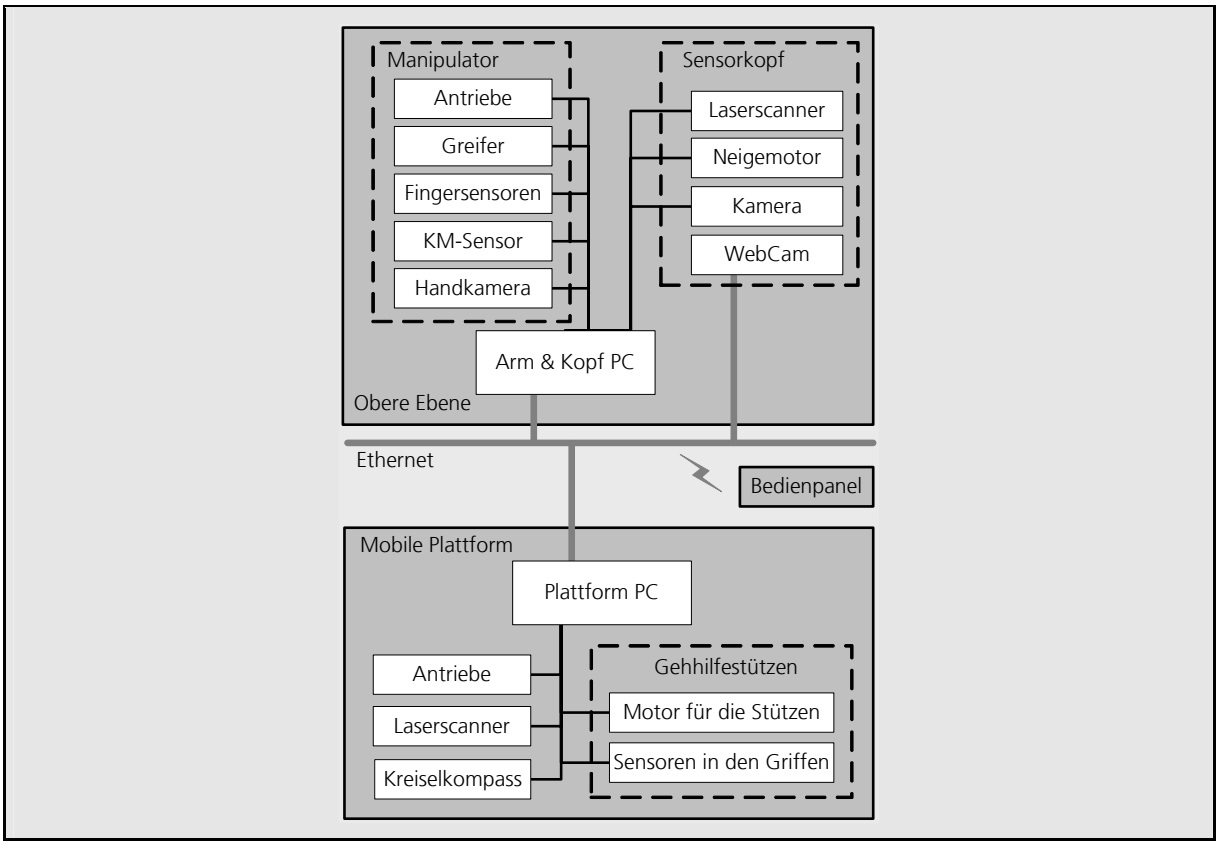


Bild 6-8: Rechnerarchitektur von Care-O-bot® II.

6.2 Realisierung des Ausführungsmoduls

Das Ausführungsmodul der Kontrollarchitektur wird in der Skriptsprache Python (vgl. Kapitel 5.3.3) programmiert. Ein großer Vorteil dieser gewählten Skriptsprache ist ihre gute Unterstützung von Netzwerkkommunikation. Damit ist es einfach möglich, über Ethernet von einem zentralen Ausführungsmodul Aktionen auf verschiedenen Computern zu starten und zu überwachen. Dies ist für die verteilte Rechnerarchitektur des Demonstrators (vgl. Abschnitt 6.1.2) notwendig.

Das Ausführungsmodul führt sowohl sequenzielle und nebenläufige als auch kontinuierliche und endliche Aktionen (Bild 5-3) aus. Die einzelnen Aktionen können hierarchisch weitere Aktionen aufrufen. Dabei können beliebige Argumente und Rückgabewerte übergeben werden. Sequenzielle Aktionen blockieren die aufrufende Aktion für die Dauer ihrer Abarbeitung. Sie sind als gewöhnliche Funktionen realisiert. Nebenläufige Aktionen blockieren die aufrufende Aktion nicht. Mit Synchronisationsmechanismen ist es möglich, auf das Ende einer oder mehrerer endlicher oder zyklischer Aktionen zu warten (Bild 6-9) und daraufhin eine andere Aktion zu starten.

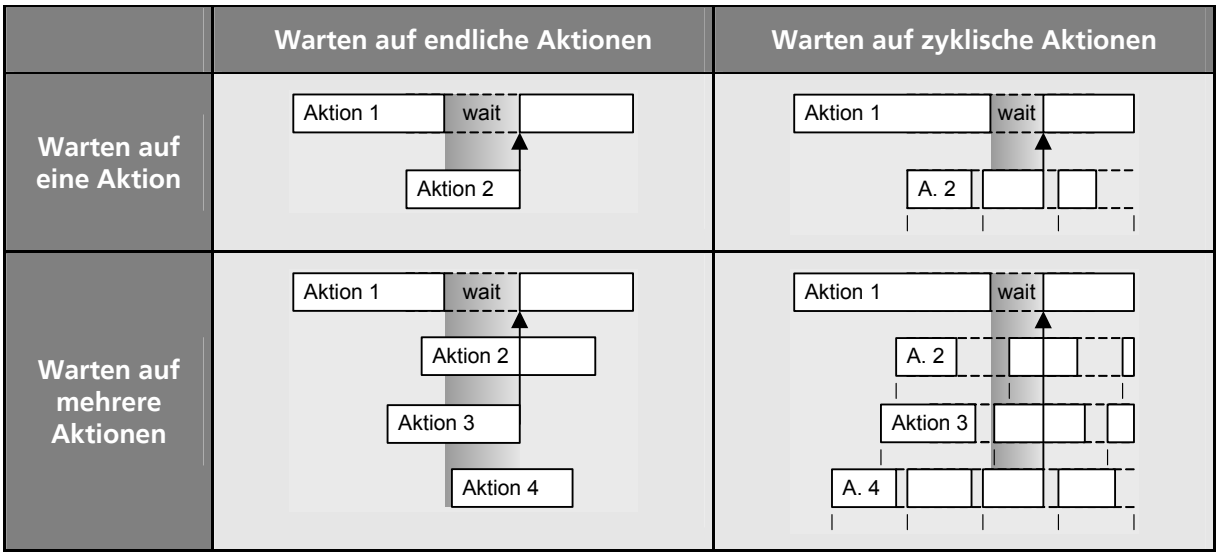


Bild 6-9: Synchronisationsmechanismen für nebenläufige Aktionen (Baum 2003), (Hans, Schraft 2003).

Die nebenläufigen Aktionen sind mit Hilfe des Thread-Moduls von Python realisiert. Jede nebenläufige Aktion ist ein nebenläufiger Thread. Bei jedem Aktionszyklusstart wird ein Semaphor[9] belegt, das zu jedem Zyklusende wieder freigegeben wird. Die Wartefunktionen überwachen die Freigabe der Semaphore und erkennen darüber das Ende eines Thread-Zyklus bzw. einer nebenläufigen Aktion. Beim Warten auf mehrere Aktionen wird auf das Ende der ersten aus einer Menge von Aktionen gewartet (Bild 6-9 unten links). Entsprechend wird bei zyklischen Aktionen auf das erste Ende eines Zyklus gewartet (Bild 6-9 unten rechts). Bei zyklischen Aktionen kann außerdem auf die Erfüllung einer Bedingung gewartet werden.

Darüber hinaus sind für nebenläufige Aktionen die folgenden weiteren Methoden implementiert:

- `stop()` bewirkt das Beenden der Aktion und ermöglicht dieser zuvor noch das Abarbeiten von Aufräumroutinen.
- `csleep()` versetzt eine Aktion in Ruhe; sie bleibt jedoch unterbrechbar für einen Stopp-Befehl.

[9] Semaphore sind ein Mechanismus zur Kontrolle des Zugriffs auf Ressourcen, die mehreren Prozessen zur Verfügung stehen, aber zu jedem Zeitpunkt von höchstens einem dieser Prozesse tatsächlich verwendet werden können (Schneider 1997).

- `getState()` gibt den aktuellen Status der Aktion wieder. Möglich sind: `running`, `ended`, `stopping`, `stopped`, `timeout` oder `failed`. Diese Unterscheidungen sind vor allem zur Fehlerdiagnose während der Entwicklung vorteilhaft.
- `returnValue()` gibt den letzten Rückgabewert einer erfolgreich beendeten Aktion bzw. den des letzten erfolgreich beendeten Zyklus wieder.

6.2.1 Behandlung von Störsituationen

Die Behandlung von Störungen wird mittels so genannter *exceptions* in Python realisiert. Tritt eine Störung in einer Aktion auf, so wird diese beendet und eine Störungsmeldung an die aufrufende Aktion übergeben. Dort kann sie abgefangen werden und eine Gegenmaßnahme eingeleitet werden. Wird die Störung nicht abgefangen, so wird auch diese Aktion beendet und die Meldung an die ihr übergeordnete Aktion übergeben. Auf diese Weise wird die Störung hierarchisch weiter nach oben gegeben, bis eine Aktion sie behebt; zuletzt erhält der Benutzer des Roboterassistenten die Störungsmeldung und kann mit neuen Anweisungen Hilfestellung geben.

6.2.2 Anbindung an die Robotersteuerung

Die Sensoren und Aktuatoren des Roboterassistenten werden über PCs angesteuert (vgl. Bild 6-8). Auf beiden PCs läuft dazu je eine Robotersteuerung, die auch schnelle und zeitkritische Regelungen unterstützt. Jeder Sensor und jeder Aktuator hat über die Robotersteuerung eine spezifische Schnittstelle zum Ausführungsmodul, über die parametrierte Aktionen aktiviert werden.

Die Robotersteuerung der mobilen Plattform enthält alle Funktionen zur autonomen Navigation. Als Aktionen werden nur Start- und Stoppbefehl sowie die Befehle zum Fahren zu einer festen bzw. relativen Position zur Verfügung gestellt. Die Bahnplanung und das Ausweichen von Hindernissen übernimmt die Robotersteuerung der Plattform selbstständig.

Entsprechend enthält die Robotersteuerung der oberen Ebene alle Funktionen für den Arm, den Greifer und den Sensorkopf, die als Aktionen aktiviert werden können. Zum Beispiel kann über eine Aktion abgefragt werden, ob sich ein bestimmter Objekttyp im Blickfeld der Kamera befindet. Als Ergebnis werden die Pixel-Koordinaten aller Objektinstanzen dieses Typs zurückgegeben. Dies entspricht der Aktion „Objekt erkennen“ der Analyse von Bild 3-5. Mit der Aktion `planTrajToObject()` werden diese Bild-Koordinaten mit dem Laserscan verbunden, die

Weltkoordinaten des Objektes zum Greifen abgeleitet und es wird eine kollisionsfreie Trajektorie für den Arm zum Zugreifen berechnet.

Eine Übersicht über die Aktionen der Robotersteuerung gibt Bild 6-10.

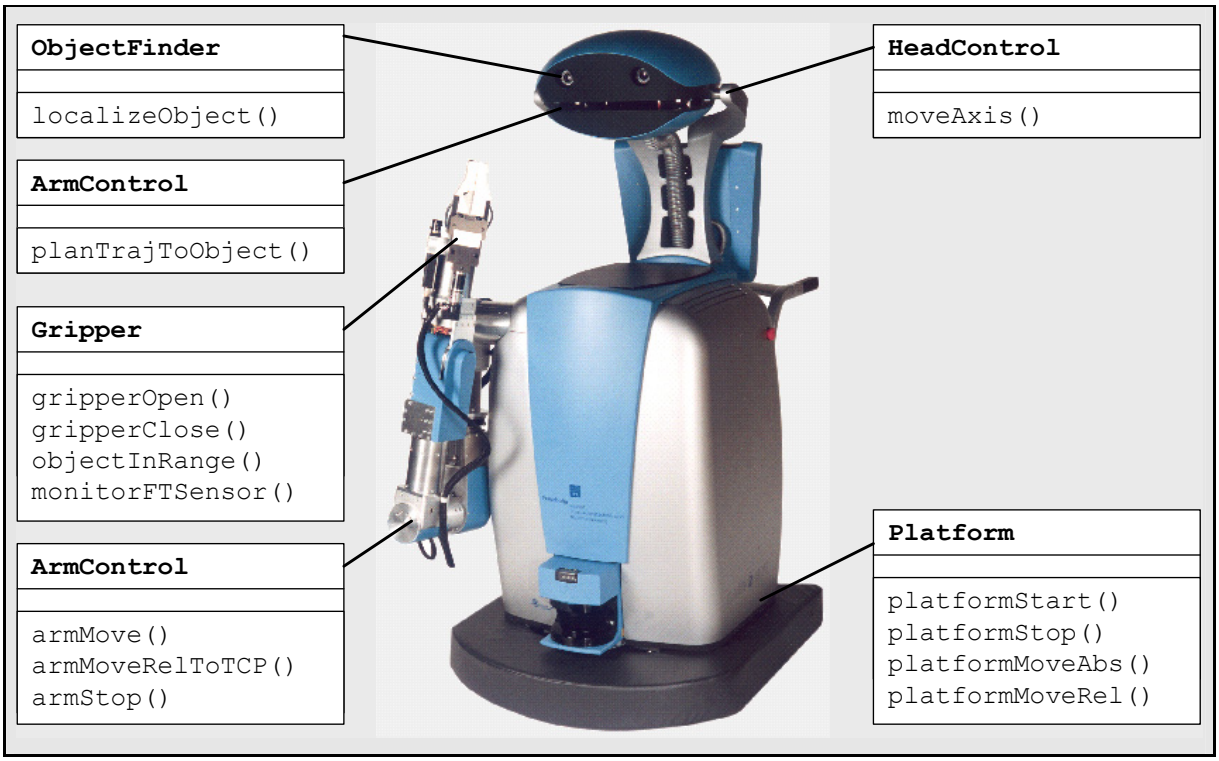


Bild 6-10: Aktionen der Robotersteuerung.

6.2.3 Aktionen für den Hol- und Bringdienst

In der Konzeption des Planungsmoduls (Kapitel 5.4.1) wurden die zu planenden Aktionen „Navigation", „Objektaufnahme", „Transport" und „Objektabgabe" für das Planungsproblem identifiziert und prädikatenlogisch modelliert (Bild 5-8). Im Folgenden wird die Umsetzung in Aktionen des Ausführungsmoduls entwickelt.

Die „Navigation" und der „Transport" unterscheiden sich nur darin, ob der Roboterassistent ein Objekt trägt oder nicht. Sie sind in „Fahren" zusammengefasst. Bild 6-11 zeigt die Aktionsaufrufe der Aktion. Im Gegensatz zur Analyse von Bild 3-4 wird das „Hindernisse vermeiden" hier nicht explizit aufgerufen, weil es bereits in der Robotersteuerung des Demonstrators in den Navigationsalgorithmen enthalten ist.

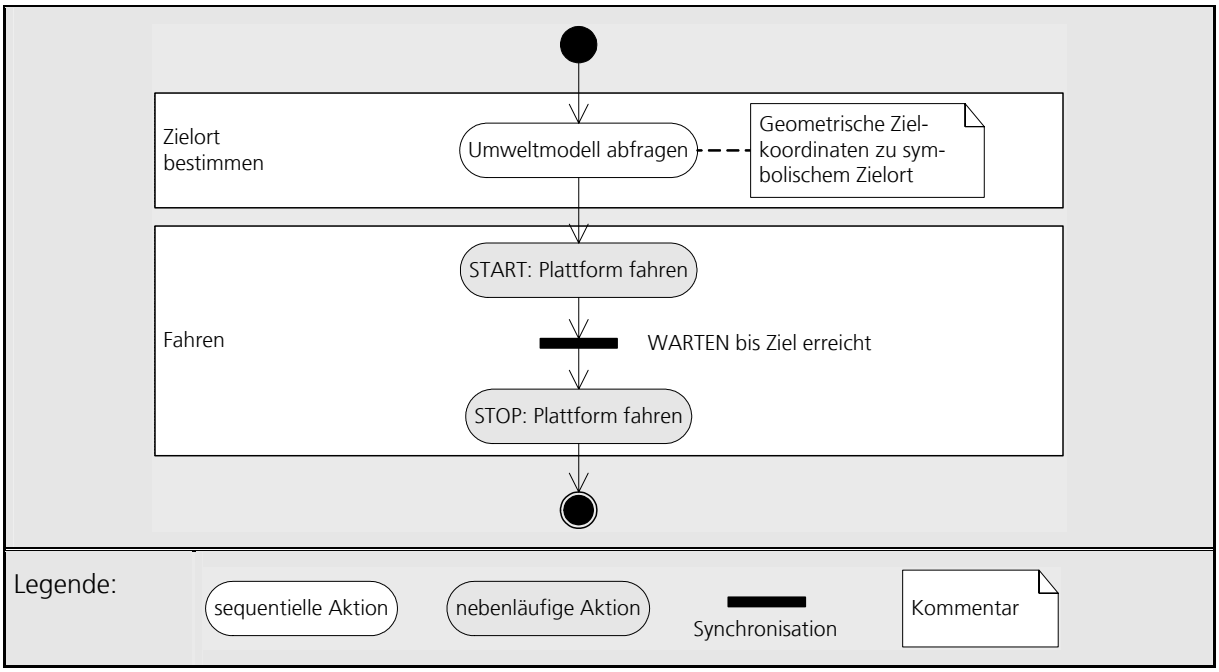


Bild 6-11: Aktion „Fahren“.

Bild 6-11 zeigt nur den positiven Verlauf, wenn jede Aktion erfolgreich beendet wird. Die Fälle, in denen das Abfragen des Umweltmodells scheitert oder das Ziel nicht erreicht wird, werden extra abgefangen und sind in Bild 6-11 nicht dargestellt. Wie in Abschnitt 6.2.1 beschrieben, werden Störsituationen erkannt und Störmeldungen an die übergeordnete Aktion gesendet. Das gleiche gilt für die Darstellungen der anderen Aktionen in Bild 6-12 bis Bild 6-15.

Bei der Aktion „Objektaufnahme“ wird unterschieden, ob das Objekt von einer Person entgegengenommen wird (physische Interaktion, Bild 6-12), oder ob es von einer Abstellfläche aufgenommen wird (Bild 6-13). Bei der Entgegennahme von einer Person streckt der Roboterassistent den Arm zur Person, öffnet den Greifer und wartet, dass die Person das Objekt in den Greifer hält. Dann schließt er den Greifer und fährt den Arm wieder ein. Weil die Plattform beim Fahren die Person erkennt und den Roboterassistenten vor ihr positioniert, genügt zum Ausstrecken des Armes eine feste Trajektorie, die nicht geplant wird. Das gleiche gilt beim Einziehen des Armes.

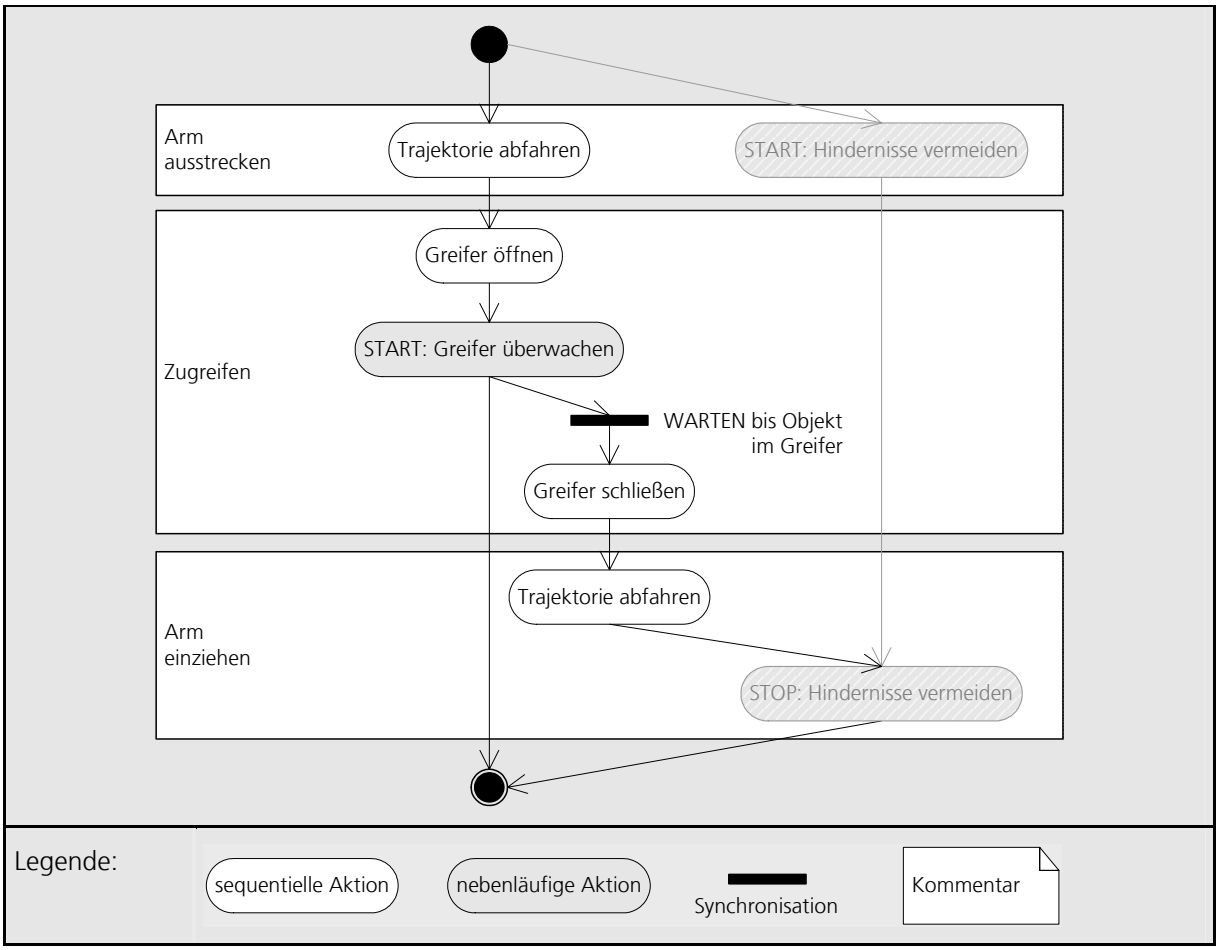


Bild 6-12: Objektaufnahme von einer Person.

Wie in Kapitel 5.2.2 konzipiert, ist das Überwachen des Greifers eine überlappende Aktion, die bis zur Abgabe des Objektes aktiv bleibt. In Bild 6-12 und in Bild 6-13 wird deshalb das Überwachen als nebenläufige Aktion nur gestartet.

Das Starten und Stoppen der Hindernisvermeidung ist in Bild 6-12 und in den folgenden Bildern jeweils halbtransparent eingezeichnet, wie es konzipiert wurde. Weil der spezielle Demonstrator nicht über Sensorik zur Hinderniserkennung beim Abfahren von Trajektorien verfügt, konnten diese Aktionen hier nicht realisiert werden.

Bild 6-13 zeigt die Aktionen zur Aufnahme eines Objektes von einer Ablagefläche. Um ein Objekt aufzunehmen, muss zunächst dessen Position genau bekannt sein. „Objekt erkennen“ und „Umwelt (in 3-D) erfassen“ sind Aktionen, die aktiv die Umwelt erfassen und mit aktuellen Sensorwerten die exakte Position bestimmen. Bei mehreren gleichen Objekten wird über eine Abfrage der Wissensbasis („Objekt identifizieren“) das geplante Objekt ausgewählt.

Anschließend berechnet ein geometrischer Planer die kollisionsfreie Trajektorie für den Manipulator zum Zugreifen.

Die Trajektorie des Manipulator-Ausstreckens soll bis kurz vor das Objekt gehen. Beim Zugreifen wird der Arm etwas weiter relativ zum TCP bewegt, so dass der Greifer um das Objekt schließt. Gleichzeitig kann über ein Tracking (vgl. S. 54) die Greifbewegung nachgeregelt werden, um eventuelle Ungenauigkeiten der zuvor stattgefundenen Umgebungserfassung und Bahnplanung auszugleichen. Nach dem Zugreifen wird das Objekt zuerst ein wenig angehoben, damit bei einer nachfolgenden Bewegung das Objekt nicht gegen die Unterlage stößt, auf der es stand.

Bei der Objektabgabe wird wie bei der Objektaufnahme unterschieden, ob das Objekt an eine Person übergeben wird (Bild 6-14) oder ob es auf eine Ablagefläche abgesetzt wird (Bild 6-15). Bei der Übergabe an eine Person wird wie bei der Annahme von einer Person eine feste Trajektorie für den Manipulator gewählt. Die Orientierung zur Person stellt die Plattform beim Fahren sicher. Wenn das Objekt angeboten wird, wird der Kraft-Momenten-Sensor überwacht und darüber erkannt, wenn die Person am Objekt zieht. Dann wird der Greifer geöffnet und das Objekt losgelassen. Erst wenn von der Greiferüberwachung bestätigt wird, dass sich kein Objekt mehr im Greifer befindet, wird dieser geschlossen und der Manipulator wieder eingezogen. Die Greiferüberwachung, die beim Aufnehmen des Objektes aktiviert wurde, wird deaktiviert.

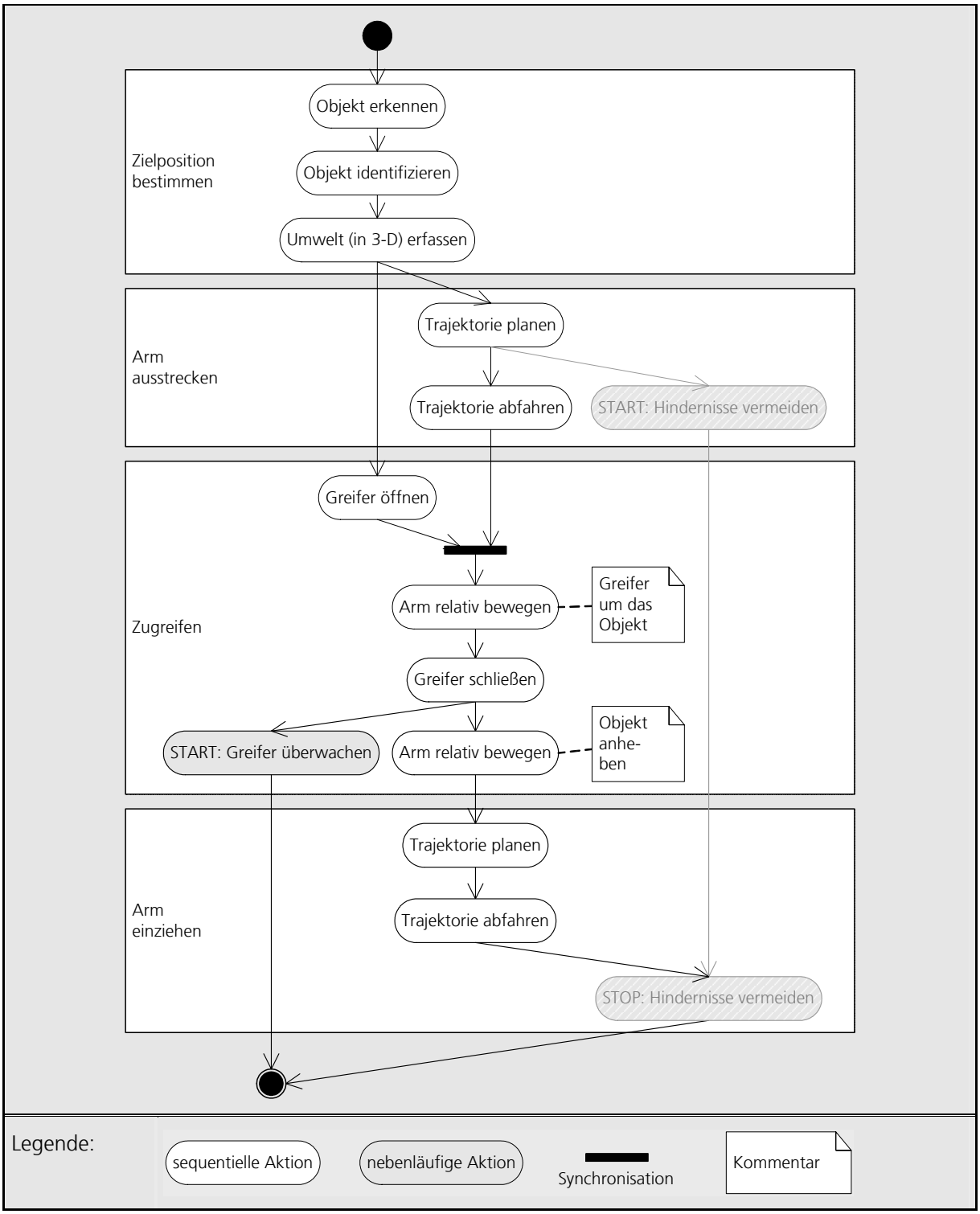


Bild 6-13: Objektaufnahme von einer Abstellfläche.

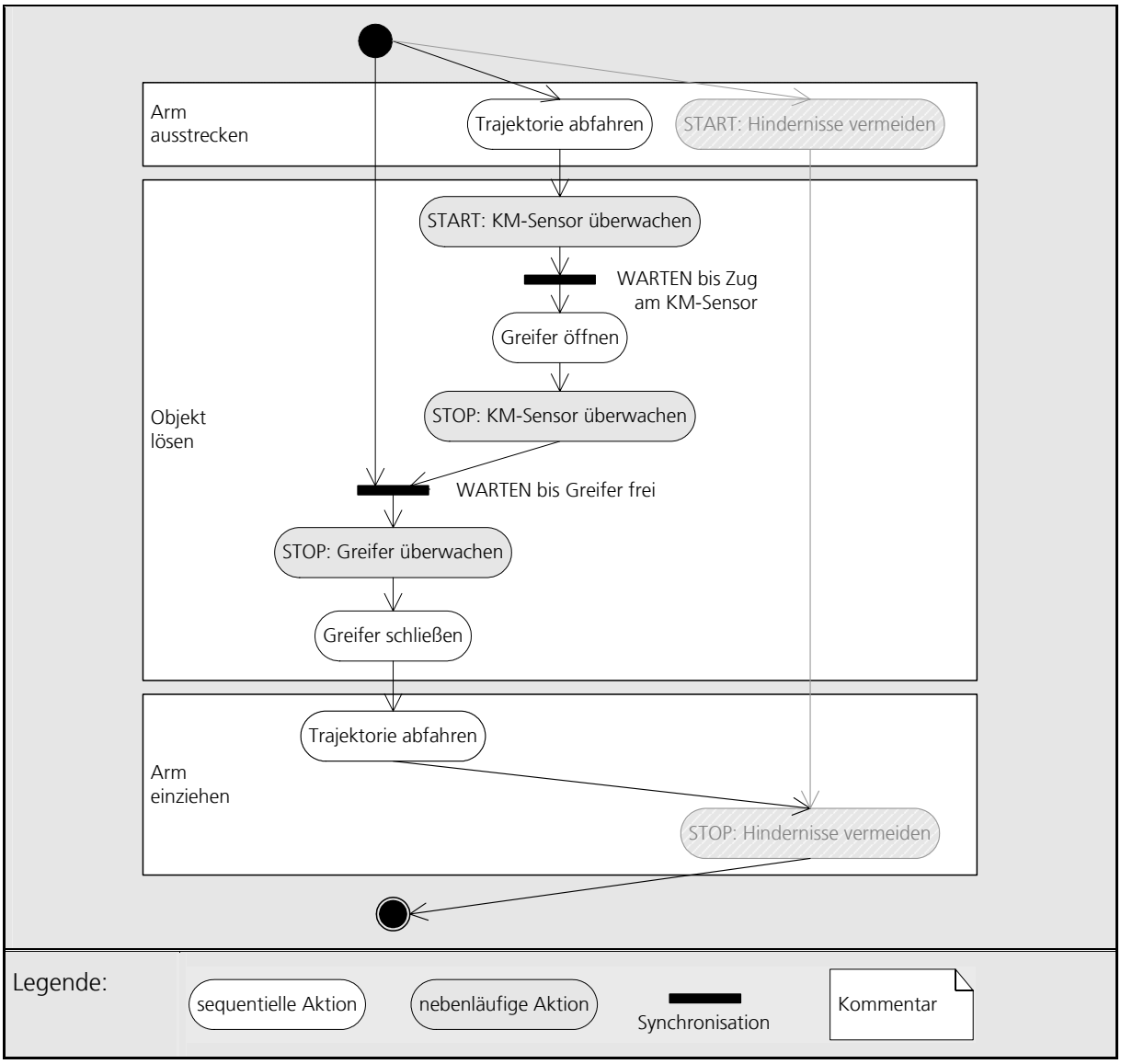


Bild 6-14: Objektabgabe an eine Person.

Bei der Objektabgabe auf eine Ablagefläche (Bild 6-15) wird zuerst eine freie Ablageposition bestimmt. Bei der anschließenden Planung der Trajektorie dorthin wird berücksichtigt, dass sich ein Objekt im Greifer befindet, d.h. die Trajektorie endet ca. 15 cm über der eigentlichen Stellfläche. Diese Höhe ist abhängig von den zu handhabenden Objekten und kann je nach Objekt parametriert werden. Von dort an bewegt sich der Manipulator langsam abwärts, wobei gleichzeitig der Kraft-Momenten-Sensor überwacht wird. Darüber wird der Kontakt des Objektes mit der Oberfläche der Ablage erkannt, der Greifer geöffnet und die Armbewegung gestoppt. Auf diese Weise wird die Aktion zuverlässig ausgeführt trotz der vorhandenen Ungenauigkeit bei der 3-D-Erfassung der Umwelt und beim Abfahren der Trajektorie. Damit der Greifer wieder geschlossen werden kann, wird der Manipulator ein Stück vom Objekt zurück bewegt.

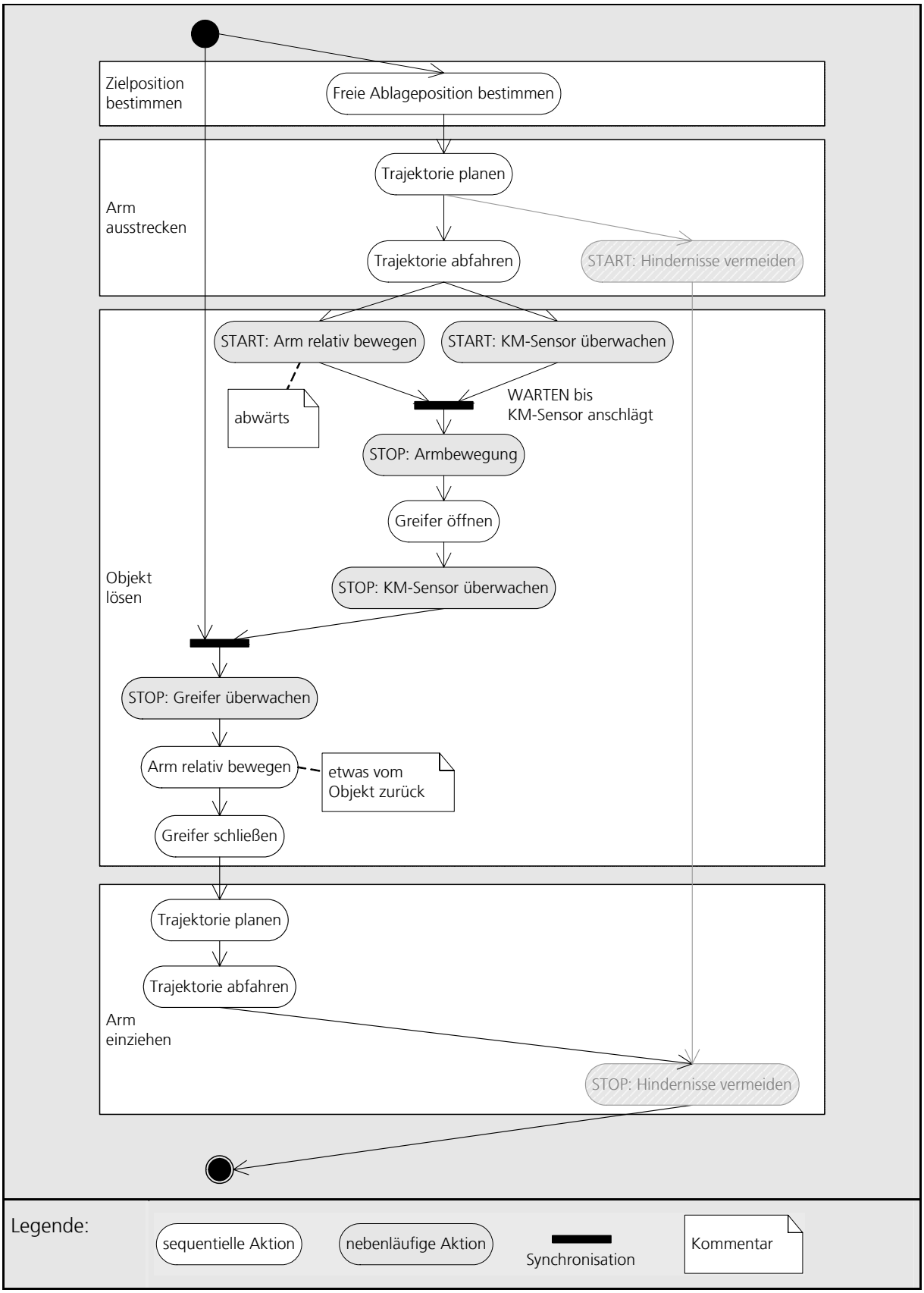


Bild 6-15: Objektabgabe auf eine Ablagefläche.

Der Vergleich von Bild 6-12 bis Bild 6-15 bestätigt, dass mit der entwickelten Kontrollarchitektur eine sehr gute Modularisierung erreicht und die Aktionen aus wiederkehrenden Teilschritten einfach zusammengefügt werden. Beispielsweise wird immer die gleiche Aktion „Trajektorie abfahren“ gebraucht; in einem Falle wird eine feste, im anderen Falle eine dynamisch berechnete Trajektorie als Parameter übergeben.

Die Mechanismen zum Starten und Stoppen von nebenläufigen Aktionen und die bedingungsgesteuerten Warte-Mechanismen unterstützen die einfache Programmierung eines reaktiven Verhaltens.

6.3 Realisierung der weiteren Softwaremodule

Die weiteren Softwaremodule der Kontrollarchitektur sind das Planungsmodul, um die groben Handlungsschritte zu planen, eine Wissensbasis, in der das Wissen über die Umwelt gespeichert und gewartet wird, und eine Mensch-Maschine-Schnittstelle. Bei dem Demonstrator laufen diese Softwaremodule und das Ausführungsmodul auf dem Bedienpanel und steuern von dort das Verhalten. Über Netzwerkverbindungen werden die Aktionen auf den zwei Steuerrechnern für die mobile Plattform und den Manipulator aktiviert bzw. deaktiviert.

6.3.1 Planungsmodul

Es ist die Aufgabe des Planungsmoduls, automatisch Aktionslisten zur Handlungsführung zu generieren. Wie in Kapitel 5.4 konzipiert, sind symbolische Planungssysteme dafür besonders geeignet. Da das Planungsproblem des Hol- und Bringdienstes einem Benchmark-Problem des Planungs-Wettbewerbes sehr ähnelt (vgl. Kapitel 4.4.2), bietet es sich an, ein Planungssystem dieses Wettbewerbes zu verwenden. Als Varianten stehen alle Planer aus Bild 4-12 zur Verfügung. Durch die Verwendung der standardisierten Repräsentationssprache PDDL ist das Planungssystem als Modul austauschbar.

Um die Ontologie aus Kapitel 5.5.1 in die Planungsdomäne zu überführen, genügt zunächst Level 1 von PDDL (vgl. Bild 4-10). Die höheren Levels können die Domäne erweitern, so dass beispielsweise das Fahren in den Nachbarraum „teurer“ als das Fahren innerhalb des Raumes bewertet wird. Auf diese Weise wird der Plan so optimiert, dass möglichst wenige Wege zurückgelegt werden.

Für den Demonstrator wird das Planungssystem „metric-FF“ verwendet. Es wird als externer Prozess vom Ausführungsmodul gestartet, seine Ausgabe wird gelesen und daraus die

Aktionsliste extrahiert. Diese wird in einer Schleife Eintrag für Eintrag abgearbeitet, indem die Aktionsstrings in Parameter und Befehl zerlegt werden, das Befehlswort in einen syntaktisch gültigen Funktionsnamen konvertiert wird und die entsprechende Aktion mit Parametern aufgerufen wird.

6.3.2 Datenbank für die Wissensbasis

In der Wissensbasis ist in erster Linie symbolisches Faktenwissen über die Umwelt gespeichert. Daraus wird die Domäne für das Planungsmodul erstellt. Eine Ontologie für dessen Konzept wurde in Kapitel 5.5.1 entwickelt. Außerdem verfügt die Wissensbasis über Daten, die zur Aktionsausführung erforderlich sind, z.B. Modelle zur Erkennung des zu greifenden Objektes oder Weltkoordinaten der Orte.

Für eine Realisierung sind verschiedene Datenstrukturen möglich. Damit die Realisierung der Wissensbasis als Modul gekapselt ist, sollen die Daten über eine standardisierte Schnittstelle zur Verfügung gestellt werden. Hierfür bietet sich SQL an, da SQL wie auch die symbolische Umweltbeschreibung auf Prädikatenlogik basiert. *MySQL* (MySQL 2003) ist eine frei verfügbare Datenbank, die für den Demonstrator verwendet worden ist. Die Datenbank hat eine Schnittstelle zum Ausführungsmodul, so dass über einfache Aktionen die nötigen Umweltinformationen für die Handlungen abgefragt werden können.

Die Datenbank enthält entsprechend der Konzeption (Bild 6-16)

- Informationen, welche **Objekte** sich in der Umwelt befinden,
- **Objekteigenschaften** wie z.B. Referenzmodelle für die Objekterkennung,
- Informationen, welche **Möbel** sich in der Umwelt befinden und ihre Koordinaten,
- Informationen über die möglichen **Positionen** auf den Möbeln,
- Informationen über die möglichen **Orte**, auf denen sich der Roboterassistent bewegen kann und
- Informationen über die **aktuelle Situation** mit allen Relationen.

In der Tabelle „Situation“ sind die Prädikate von Gleichungen (5-10) bis (5-17) mit ihren aktuellen Argumenten gespeichert. Die Koordinaten der Positionen sind in lokalen Koordinaten gespeichert, d.h. wenn das Möbel verschoben wird, verschieben sich die Positionen mit.

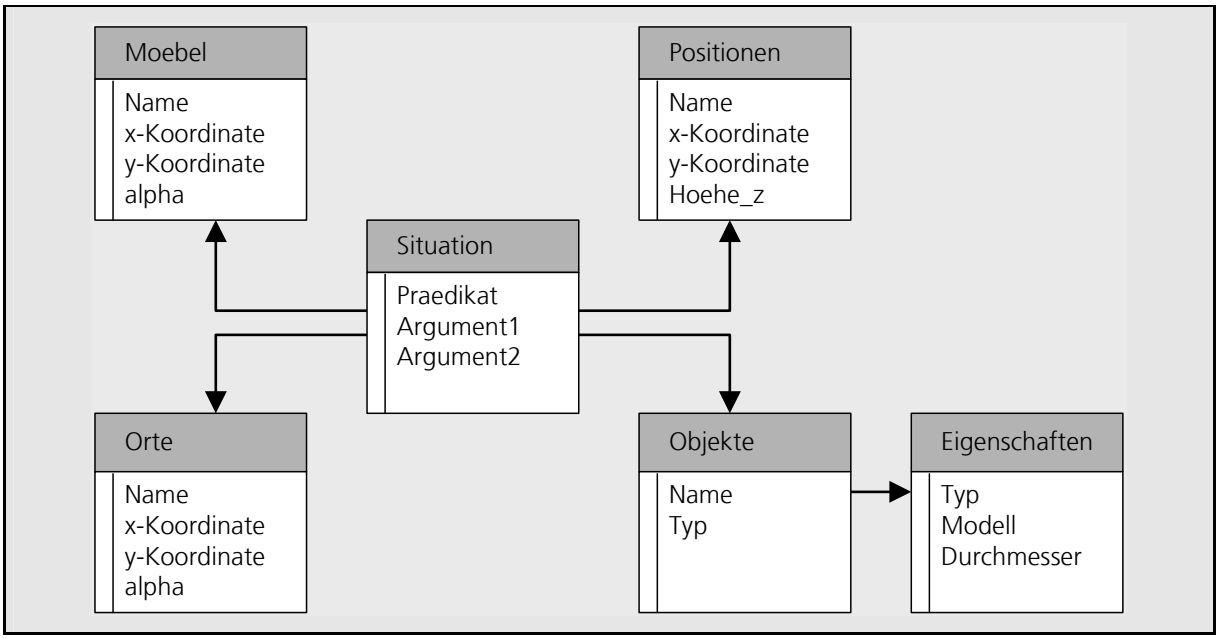


Bild 6-16: Fakten der Wissensbasis.

Generieren der Planungsdomäne

Die Domänen-Beschreibung, die das Planungsmodul benötigt, wird bei jeder Aufgabe dynamisch aus den Daten der Wissensbasis generiert. Dafür stehen die Listen der möglichen Objekte, Positionen und Orte sowie die Situationsbeschreibung zur Verfügung. Über einen Filter werden nur die relevanten Daten in die Domänen-Beschreibung aufgenommen.

Wartung des Umweltwissens

Wie in Kapitel 5.6 konzipiert, trägt das Ausführungsmodul den Effekt seines Handelns in die Situationsbeschreibung ein. Dies wird mittels eigener Aktionen realisiert, die als Schnittstelle zur Wissensbasis dienen. Hierfür existieren spezifische Aktionen, wie z.B. zum Aktualisieren der Position des Roboterassistenten. Sie werden nach jeder Aktion bzw. Störungsbehandlung aufgerufen und aktualisieren damit zeitnah das Umweltwissen. Ergänzend gibt es Aktionen zum Ausführen beliebiger SQL-Statements.

Die Schnittstelle zur Datenbank unterstützt transitive Abfragen: ist beispielsweise eine Tasse auf einer Untertasse auf einem Tisch, kann ermittelt werden, dass die Tasse auf dem Tisch ist, obwohl in der Datenbank lediglich der Eintrag besteht, dass die Tasse auf der Untertasse steht. Dies ist für die manuelle Eingabe und intuitive Wartung der Datenbank notwendig und erleichtert ihre Pflege.

6.3.3 Mensch-Maschine-Schnittstelle

Die Kommandierung des Roboterassistenten erfolgt über das Bedienpanel mit Touchscreen (Bild 6-7). Durch Anschluss eines Headsets können auch Schlüsselworte über Sprache eingegeben werden. Die Rückmeldung über den aktuellen Systemzustand findet durch Sprachausgaben und grafische Darstellung auf dem Bildschirm statt. Es werden die aktuelle Aktion des Roboter-assistenten und eventuelle Störungsmeldungen an den Benutzer zurückgegeben. Bei Störungen kann der Mensch helfend eingreifen.

Die Benutzeroberfläche hat nur zum Ausführungsmodul eine Schnittstelle. Dieses kapselt die Daten und macht sie unabhängig von der Realisierung der anderen Module. Zur Kommandierung werden über dynamische Listen nur die Aktionen zur Verfügung gestellt, die der Roboterassistent tatsächlich ausführen kann.

Entsprechend werden zum Editieren der Wissensbasis nur relevante, gültige Eingaben über dynamische Listen eingeblendet. Dies erleichtert die Bedienung und verhindert falsche Eingaben. Dabei werden menschliche Begriffe verwendet, d.h. der Benutzer wählt den Objekttyp (z.B. „Cola“) und das Ausführungsmodul konvertiert die Eingabe in ein eindeutiges Element (z.B. „Cola3“). Entsprechend verwaltet das Ausführungsmodul die Positionen, d.h. der Benutzer wählt intuitiv und umgangssprachlich nur das Möbel (z.B. „Kommode“) als Abstellposition und muss nicht genau wissen, wo sich die Kommode befindet und wo genau auf der Kommode Platz ist. Die Verwaltung der einzelnen Positionen und Objekte erfolgt dynamisch, so dass die Komplexität der Kontrollarchitektur vor dem Benutzer verborgen bleibt.

Seinen menschlichen Interaktionspartner erkennt der Roboterassistent anhand der Geometrie von Beinen im Laserscan oder über Gesichtserkennung mit einer Kamera.

7 Validierung und Bewertung

In einem Testszenario wurde untersucht, ob der Roboterassistent Care-O-bot® II die Erwartungen von Kapitel 3.1 erfüllt und ob die Kontrollarchitektur die an sie gestellten Anforderungen erfüllt. Das Szenario wird in Abschnitt 7.1 beschrieben. Die Experimente und ihre Ergebnisse sind in Abschnitt 7.2 dokumentiert und bewertet.

7.1 Haushaltsszenario

Als Testszenario diente eine Wohnungsumgebung (Fläche: 5 m x 5 m) mit Wohnzimmermöbeln bestehend aus einem Sofa, einem Couchtisch, einem Bücherregal, einer Kommode und einem Esstisch. Sie sind in der Menge von Möbeln $\mathcal{M}$ zusammengefasst. Außerdem befanden sich in der Wohnung verschiedene handhabbare Objekte $\mathcal{O}$ gemäß Gleichungen (7-1):

$$\begin{aligned}
\mathcal{M} &= \{\texttt{Sofa},\texttt{Couchtisch},\texttt{Buecherregal},\texttt{Kommode},\texttt{Esstisch}\} \\
\mathcal{S} &= \{\texttt{Saft1},\texttt{Saft2}\} \\
\mathcal{C} &= \{\texttt{Cola1},\texttt{Cola2},\texttt{Cola3}\} \\
\mathcal{B} &= \{\texttt{Bier1},\texttt{Bier2},\texttt{Bier3}\} \\
\mathcal{P} &= \{\texttt{Pringles1},\texttt{Pringles2}\} \\
\mathcal{O} &= \mathcal{S} \cup \mathcal{C} \cup \mathcal{B} \cup \mathcal{P}
\end{aligned} \qquad \textbf{(7-1)}$$

Für das Sofa wurde eine, für die anderen Möbel wurden je drei Ablagepositionen definiert. Mit Gleichung (5-13) ergibt sich daraus die Summe der Ablagepositionen zu $|\mathcal{A}| = 13$. Darüber hinaus kann der Roboter selbst ein Objekt im Greifer aufnehmen. Zusammen gibt es also 14 Ablagepositionen. Die Anzahl der greifbaren Objekte beträgt $|\mathcal{O}| = 10$. Nach (3-1) ergeben sich daraus $\frac{14!}{(14-10)!} \approx 3{,}6$ Mrd. verschiedene Kombinationen, welche Objekte sich an welchen Ablagepositionen befinden können.

Innerhalb dieser Umgebung sollte der Roboterassistent nun auf Kommando beliebige, spezifizierte Objekte selbstständig holen und zu anderen Abstellpositionen bringen. In Fällen, in denen er nicht mehr „weiter weiß“, sollte er den Benutzer um Hilfe bitten (vgl. Kapitel 2.3).

Dieses Szenario wurde sowohl in einer Simulation als auch in der Realität erprobt (Bild 7-1). In der Simulation konnte im Zeitraffer getestet werden, ob der Roboterassistent immer an die richtigen Orte fährt. Im Wohnungsumfeld wurde das Gesamtsystem erprobt und die Robustheit der Lösung überprüft. Das Testszenario wurde nicht nur im Versuchsfeld, sondern auch auf verschiedenen Messen einem großen Publikum demonstriert.

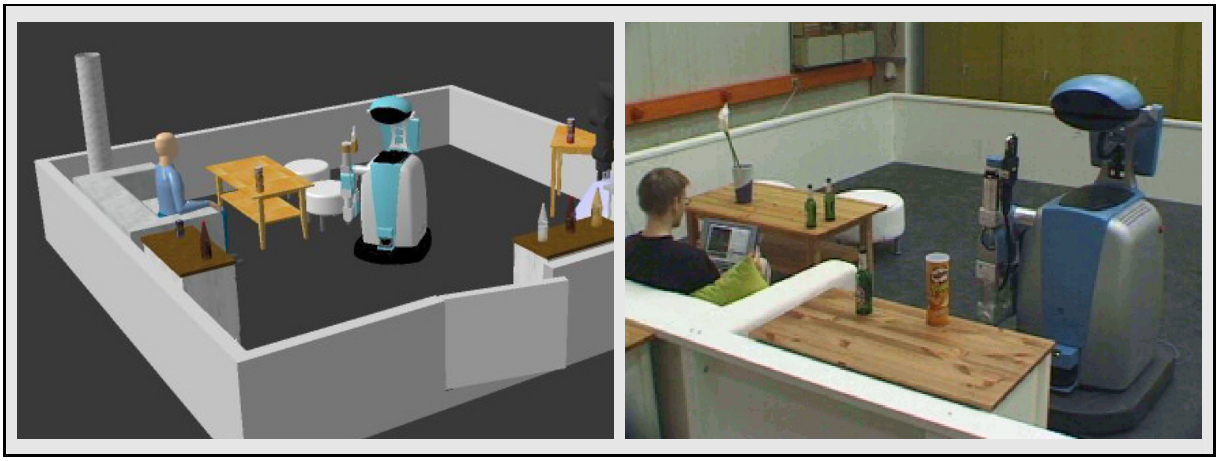


Bild 7-1: Simuliertes (links) und realisiertes (rechts) Szenario für den Hol- und Bringdienst in einer Wohnungsumgebung.

7.2 Experimentelle Untersuchungen

Für die Experimente können die einzelnen Objekte aus (7-1) prinzipiell an beliebigen Positionen sein. Zum einfacheren Verständnis galt für die folgenden Beispiele die Ausgangssituation, dass kein Objekt am Sofa war und die anderen Objekte sortiert waren. Es waren alle Bierflaschen auf dem Couchtisch, alle Pringles-Dosen auf dem Bücherregal, alle Säfte auf dem Esstisch und alle Cola-Flaschen auf der Kommode.

7.2.1 Zielgerichtetes Verhalten

Durch den Einsatz des Planungsmoduls in der Kontrollarchitektur soll der Roboterassistent ein zielgerichtetes Verhalten zeigen. Dies wurde mit den Zielvorgaben von Bild 7-2 überprüft. Das Planungsmodul lief auf dem Bedienpanel mit einem Pentium III 800 MHz Prozessor.

$\texttt{auf(x,m)} \mid \texttt{x} \in \mathcal{O}, \texttt{m} \in \mathcal{M}$	**(7-2)**
bewirkt, dass sich das Objekt x auf Möbelstück m befinden soll.	
$\exists \texttt{x} \in \mathcal{C} : \texttt{x} \rightarrow \texttt{auf(x,m)} \mid \texttt{m} \in \mathcal{M}$	**(7-3)**
bewirkt, dass sich ein beliebiges Objekt vom Typ "Cola" auf Möbelstück m befinden soll. Diese Zielvorgabe entspricht dem Befehl „Bringe eine Cola zu m“.	
$\forall \texttt{x} \in \mathcal{C} : \texttt{x} \rightarrow \texttt{auf(x,m)} \mid \texttt{m} \in \mathcal{M}$	**(7-4)**
bewirkt, dass sich alle Objekte vom Typ "Cola" auf Möbelstück m befinden sollen unter der Bedingung $\lvert\mathcal{C}\rvert \leq \lvert\mathcal{A}_m\rvert$. Für $\lvert\mathcal{C}\rvert > \lvert\mathcal{A}_m\rvert$ ist das Problem unlösbar.	
$\neg\exists \texttt{x} \in \mathcal{O} : \texttt{x} \rightarrow \texttt{auf(x,m)} \mid \texttt{m} \in \mathcal{M}$	**(7-5)**
sorgt dafür, dass sich kein Objekt mehr auf dem Möbelstück m befindet. Diese Zielvorgabe entspricht dem Befehl „Räume m auf“. Das letzte Objekt behält der Roboter allerdings im Greifer.	
$\neg\exists \texttt{x} \in \mathcal{O} : \texttt{x} \rightarrow \left(\texttt{auf(x,m)} \vee \texttt{imGreifer(x)}\right) \mid \texttt{m} \in \mathcal{M}$	**(7-6)**
erweitert (7-5) so, dass sich nach der Ausführung auch kein Objekt mehr im Greifer befindet.	

Bild 7-2: Zielvorgaben im Testszenario.

Die Anfragen von Gleichungen (7-2) und (7-3) liefern jeweils eine Lösung in vier Schritten (Navigation, Objektaufnahme, Transport, Objektabgabe) wie in Bild 3-3, wenn die Zielposition frei ist. Der Prozessor benötigt für die Planung 0,07 bzw. 0,08 Sekunden. Ist die Zielposition nicht frei, so muss der Roboterassistent erst zur Zielposition fahren, ein Objekt aufnehmen, um die Position frei zu räumen, dieses woanders abstellen und dann wie im ersten Falle fortfahren. Das Planungsmodul benötigt dann 8 Schritte und 0,09 Sekunden für die Planung.

Soll der Roboterassistent alle Cola-Flaschen auf den Couchtisch räumen (Gleichung (7-4)), so muss er erst die Bierflaschen vom Couchtisch beiseite stellen. Der Plan benötigt 20 Handlungsschritte und ist in 0,33 Sekunden berechnet. Zum Umräumen der Cola-Flaschen auf den Esstisch muss er nur zwei Pringles-Dosen wegräumen. Der Plan benötigt dann 16 Schritte und ist in 0,18 Sekunden berechnet. Wird Gleichung (7-4) so erweitert, dass nicht nur alle Cola-Flaschen auf den Couchtisch, sondern gleichzeitig auch alle Bierflaschen auf die Kommode sollen (Vertauschen der Objekte), so erhöht sich der Plan auf 24 Schritte, seine Berechnung benötigt nur 0,16 Sekunden.

Das Aufräumen des Couchtisches nach Gleichung (7-4) benötigt 10 Handlungsschritte und 0,23 Sekunden zur Planung, nach Gleichung (7-5) benötigt es 12 Schritte und 0,25 Sekunden.

Auf dem Esstisch sind nur zwei Objekte, so dass dort das Aufräumen mit 6 bzw. 8 Schritten möglich ist und die Planung 0,17 bzw. 0,18 Sekunden dauert.

Die Befehle von Bild 7-2 werden einfach über die Benutzeroberfläche der Mensch-Maschine-Schnittstelle angewählt. Die Experimente zeigen, dass der Roboterassistent sehr gut einfache Befehle in komplexe Handlungen umsetzt und dabei auch mit Klassen von Objekten plant (Erfüllung der Anforderungen Ã5).

7.2.2 Robustes Agieren in menschlicher Umgebung

Im regulären Betrieb arbeitet der Roboterassistent die Pläne ab, indem das Ausführungsmodul nacheinander die Aktionen wie in Kapitel 6.2.3 beschrieben aktiviert. Die Objektaufnahme ist dabei sehr robust, weil jedes Mal eine individuelle Trajektorie berechnet wird (vgl. Bild 6-13). Das Objekt kann innerhalb des Arbeitsraumes des Manipulators beliebig auf der Abstellfläche verschoben werden. Bei der Objektabgabe darf die Höhe des Möbels variieren. Das Objekt wird erst bei Kontakt losgelassen. Der Roboterassistent passt seine Handlungen an Details der aktuellen Umgebungssituation an und kontrolliert sie in Echtzeit (Erfüllung der Anforderungen A 11 und A 12).

Während der Experimente wurden folgende Störungsfälle beobachtet und wie folgt behandelt:

- **Aufgabe vom Planungsmodul unlösbar**
  Wird vom Benutzer eine unlösbare Planungsaufgabe vorgegeben (z.B. der Fall $|\mathcal{C}| > |\mathcal{A}_m|$ in Gleichung (7-4)), so wird vom Ausführungsmodul eine Fehlermeldung in der Aktionsliste des Planungsmoduls erkannt und eine Meldung an den Benutzer gegeben.

- **Objekterkennung scheitert**
  Die häufigste Störung im Ablauf des Hol- und Bringdienstes ist ein Scheitern der Objekterkennung während der Objektaufnahme von einer Ablagefläche. Dies hängt mit der Abhängigkeit der Bildverarbeitungsalgorithmen von den Beleuchtungsverhältnissen zusammen. In einer geschlossenen Halle mit konstanter, künstlicher Ausleuchtung konnte eine Zuverlässigkeit von 98 Prozent erreicht werden. Unter allen anderen Umgebungsbedingungen war die Zuverlässigkeit geringer. Die Störung wird vom Ausführungsmodul erkannt und als Maßnahme wird das aktuelle Kamerabild auf das Bedienpanel an den Benutzer gesendet. Der leistet interaktiv dem Roboterassistenten Hilfe, indem er das Objekt manuell markiert. Die Objektaufnahme wird anschließend planmäßig fortgesetzt.

- **Tatsächliche Objektanzahl ist ungleich der erwarteten Anzahl von Objekten**
  Dass die Objektidentifikation scheitert, weil die Anzahl der erkannten Objekte nicht mit der erwarteten Anzahl von Objekten übereinstimmt, hängt in der Regel mit einem Scheitern der Objekterkennung (siehe vorige Störungsbeschreibung) zusammen, wenn diese nicht alle Objekte erkennt. In diesen Fällen hilft der Benutzer mit einer manuellen Markierung auf dem Bedienpanel. Stimmt die Wissensbasis nicht mit der Realität überein, weil die Umwelt verändert wurde, so muss der Mensch manuell die Datenbank editieren. Eine automatische Aktualisierung ist prinzipiell möglich. Weil aber die Objekterkennung so unzuverlässig ist, würde dann zu häufig ein falscher Eintrag in der Wissensbasis vorgenommen, so dass hier die Lösung des manuellen Editierens realisiert wurde.

- **Umwelterfassung in 3-D scheitert**
  Wenn der Laserscanner im Kopf des Roboterassistenten ausfällt, diagnostiziert das Ausführungsmodul diese Störung und gibt eine Meldung an den Benutzer aus.

- **Objekt nicht kollisionsfrei erreichbar**
  Die Planung von Trajektorien für den Manipulator geschieht in dessen Robotersteuerung. Sie kann scheitern, wenn das Objekt mit der Kinematik des Manipulators nicht kollisionsfrei erreichbar ist, z.B. weil es außerhalb des Arbeitsraumes steht. Das Ausführungsmodul diagnostiziert diese Störung und gibt eine Meldung an den Benutzer aus.

- **Objekt nicht aufgenommen**
  In sehr seltenen Fällen kann der Roboterassistent das Objekt nicht aufnehmen, z.B. weil die Trajektorie nicht genau genug war. Dies hängt mit einem mechanischen Spiel der Achsen zusammen oder mit Reflexionen im Scan. In der Kontrollarchitektur wird die Störung von der Aktion „Greifer überwachen“ erkannt und eine Meldung an den Benutzer ausgegeben.

- **Beim Abstellen kein Kontakt zur Ablagefläche**
  Wenn der Kraft-Momenten-Sensor beim Absetzen nicht den Kontakt zur Ablagefläche meldet (vgl. Bild 6-15), wird die Armbewegung nach 15 Zentimetern abgebrochen und der Greifer nicht geöffnet. Der Roboterassistent behält das Objekt im Greifer und zieht den Manipulator wieder ein. Ursache ist in der Regel, dass der Greifer am Objekt entlang gleitet (z.B. an der Verjüngung von Cola-Flaschen) und die Reibungskraft nicht ausreicht. Ein zweiter Versuch gelingt, wenn zuvor der Greifer fester geschlossen wird.

Die Experimente demonstrierten die Anpassungsfähigkeit des Roboterassistenten an eine sich verändernde Umwelt und an Störungen im Ablauf. Die adaptierende Verhaltenskontrolle in der Kontrollarchitektur (Anforderung Ã4) verbindet die automatisch generierten Pläne mit reaktiven Verhaltensweisen und ermöglicht auf diese Weise ein robustes Agieren in menschlicher Umgebung.

7.2.3 Kommunikation und Interaktion

Der Roboterassistent kommuniziert während der Aufgabenausführung per Sprachausgabe welchen Handlungsschritt er als nächstes ausführen wird (vgl. Kapitel 6.3.3). Dies war während der Experimente sehr hilfreich, um sein Verhalten zu verstehen. Auch Messebesucher, die den Roboterassistenten nicht kannten, verstanden dadurch leichter sein Handeln.

Die physische Interaktion bei der gegenseitigen Objektübergabe zum bzw. vom Roboterassistenten (vgl. Bild 6-12 und Bild 6-14) wurde als sehr intuitiv empfunden, da der Roboterassistent sofort auf seine Sensoreingänge reagiert. Bei Experimenten mit gestörter Wireless-LAN Übertragung verlängerte sich diese Reaktion jedoch auf über 10 Sekunden, so dass dann keine intuitive Objektübergabe mehr möglich war. Dies liegt an der speziellen Rechnerarchitektur (vgl. Bild 6-8), bei der die Sensordaten vom PC der oberen Ebene zum Ausführungsmodul auf das Bedienpanel per Funk übertragen und dort ausgewertet werden. Abhilfe würde eine zusätzliche Aktion in der Robotersteuerung der oberen Ebene schaffen, die diese Entscheidung fällt. Die Kontrollarchitektur unterstützt auch diese Alternative.

7.3 Erweiterungsfähigkeit

Die Kontrollarchitektur ermöglicht ein einfaches Erweitern des Verhaltens des Roboterassistenten: Wird das Spektrum der Fähigkeiten des Roboterassistenten in der Robotersteuerung erweitert, so werden diese dem Ausführungsmodul als Aktionen zugänglich gemacht. Handelt es sich um Fähigkeiten, die auch geplant werden sollen, so werden diese noch für das Planungsmodul symbolisch modelliert (ähnlich der Aktionen von Bild 5-8).

Erweiterungen des Objektspektrums werden nur in die Datenbank eingetragen. Sie benötigen keine zusätzliche symbolische Modellierung, da diese dynamisch von den Datenbankeinträgen übernommen wird. Entsprechendes gilt für die Möbelkoordinaten, wenn z.B. neue Möbel in die Wohnung gestellt werden oder wenn der Roboterassistent in einer neuen Umgebung agieren soll.

7.4 Übertragung der Kontrollarchitektur auf andere Serviceroboter

Bei der Konzeption und Entwicklung der Kontrollarchitektur wurde ein sehr allgemeingültiger Ansatz für die Verhaltenskontrolle von Roboterassistenten verfolgt. Sie wurde für den Roboterassistenten im Haushalt Care-O-bot® II realisiert und erprobt. Sie kann aber auch auf andere Roboterassistenten portiert werden. Ihr Einsatz ist immer dann von Nutzen, wenn ein robustes, zielgerichtetes Verhalten in Umgebungen ohne spezielle Anpassungen erreicht werden soll. Der Nutzen ist umso größer, je mehr Fähigkeiten der Roboterassistent hat, die in geplanten Handlungsabläufen kombiniert werden sollen.

Für eine Portierung der Kontrollarchitektur muss der Roboterassistent eine Robotersteuerung haben, in der die einzelnen Fähigkeiten als Aktionen aktiviert werden können (Multithreading). Dies gilt z.B. für den Roboterassistenten für Produktion *rob@work* (Bild 4-1 unten Mitte). Er demonstrierte auf der Hannover Messe Industrie 2001 einen Hol- und Bringdienst.

Servicerobotern ohne Manipulator genügt oft eine Kontrollarchitektur ohne Planungsmodul, da sie ein kleineres Spektrum an Fähigkeiten haben. Die zwei Ausstellungsroboter im Marken- und Kommunikationszentrum der Opel Adam AG in Berlin (Bild 7-3) haben beide eine geeignete Robotersteuerung, das hier entwickelte Ausführungsmodul, welches das Verhalten der Roboter kontrolliert und eine Mensch-Maschine-Schnittstelle. Die Roboter haben die Aufgabe, Besucher zu begrüßen, sie zu Exponaten zu begleiten und diese vorzustellen.

Bild 7-3: Ausstellungsroboter „OSKAR“ und „MONA“ von Opel in Berlin.

8 Zusammenfassung und Ausblick

Roboterassistenten sind Helfer des Menschen und agieren in dessen natürlicher Umgebung, wie zum Beispiel im Haushalt oder in Fertigungshallen. Sie sind charakterisiert durch ihre dem Menschen ähnliche Wahrnehmungs-, Interaktions- und Kommunikationsfähigkeit. Ihr adaptierendes Verhalten wird von ihrer Kontrollarchitektur bestimmt, welche durch ihre internen funktionalen Bestandteile und deren funktionelles Zusammenwirken definiert ist.

Aufgrund der vielfältigen, potenziellen Anwendungsszenarios für Roboterassistenten wurden bei bisherigen Realisierungen stets spezialisierte Einzelanfertigungen entwickelt. Diese Demonstratoren können zwar eine definierte Aufgabe lösen, scheitern jedoch häufig an ihrer mangelnden Vielseitigkeit. Die bisher einzigen kommerziellen Systeme aus Japan können nicht einmal manipulieren. Für Hersteller und Anwender wird ein Roboterassistent jedoch genau dann interessant, wenn dieser nicht nur eine spezielle Aufgabe, sondern das gesamte Spektrum des jeweiligen Tätigkeitsfeldes abdecken kann. Um den Anforderungen verschiedener Aufgaben zu genügen, muss ein Roboterassistent deshalb über eine Kontrollarchitektur verfügen, welche eine Adaption an veränderte Tätigkeitsfelder ermöglicht.

Auf der Basis dieser Feststellung war das Ziel der vorliegenden Arbeit, grundlegende Erkenntnisse zur Modularisierung der Verhaltenskontrolle von Roboterassistenten zu erarbeiten und durch eine systematische Vorgehensweise die Randbedingungen und Potenziale einer geeigneten, modularen Kontrollarchitektur für Roboterassistenten aufzuzeigen.

Den Tätigkeitsfeldern von Roboterassistenten ist gemeinsam, dass das Gesamtsystem mobil ist, an verschiedenen Orten manipulierende Aufgaben wahrnimmt, dabei selbstständig die Umwelt wahrnimmt und interpretiert und mit dem Menschen kommuniziert und interagiert. Es soll sich dabei zielgerichtet verhalten und robust in der menschlichen Umgebung agieren. In dieser Arbeit werden der Hol- und Bringdienst als repräsentative Tätigkeit analysiert und Anforderungen an eine modulare Kontrollarchitektur und ihre Teilsysteme abgeleitet: Die Kontrollarchitektur muss die Ansteuerung geeigneter Aktuatoren und Sensoren über eine Robotersteuerung unterstützen, über eine explizite Umweltrepräsentation zur Modellierung der Bereiche verfügen, die nicht aktuell sensoriell erfasst werden können, eine Planungskomponente einbinden, die das zielgerichtete Verhalten sicherstellt, und eine adaptierende Verhaltenskontrolle muss das Zusammenwirken der Teilsysteme kontrollieren.

Basierend auf diesen Anforderungen werden für die modulare Kontrollarchitektur und ihre Teilsysteme verschiedene Lösungskonzepte entwickelt und im Hinblick auf die gestellten

Anforderungen systematisch bewertet und ausgewählt. Daran anschließend werden die favorisierten Teilsystemlösungskonzepte in ein Gesamtsystem integriert.

Als grundsätzlicher Typ wird für die Kontrollarchitektur eine Mehrebenen-Architektur ausgewählt. Die Entscheidung basiert auf einer Bewertung des Stands der Technik veröffentlichter Architekturkonzepte. Die Mehrebenen-Architektur verknüpft schnelle Regelungsalgorithmen in einer hardwarenahen Robotersteuerung mit langsamen Kognitionsprozessen eines Planungsmoduls über ein Ausführungsmodul.

Einen besonderen Schwerpunkt der Arbeit bildet die Konzeption und Entwicklung eines neuen Ausführungsmoduls. Es ist die Zentrale der Verhaltenskontrolle. Die Fähigkeiten des Roboterassistenten werden dem Ausführungsmodul als so genannte Aktionen von der Robotersteuerung zur Verfügung gestellt. Die Aktionen sind parametrierbare, offene oder geschlossene Regelkreise, welche die Hardware ansteuern. Sie sind modular strukturiert, so dass sie mit hierarchischen Aufrufen und sequentiellen Folgen zu komplexeren Aktionen zusammengesetzt werden können. Das Ausführungsmodul koordiniert die Aktivierung der Aktionen, überwacht ihre Ausführung und behandelt auftretende Störsituationen.

Für das Ausführungsmodul wurde im Rahmen dieser Arbeit ein neues Softwarepaket mittels einer Skriptsprache realisiert. Im Gegensatz zu anderen Lösungen des Stands der Technik, die mit Lisp, C oder JAVA realisiert sind, wird die Skriptsprache zur Laufzeit interpretiert und bietet neben der einfachen Integration in die Kontrollarchitektur eine interaktive Kommandozeile, die das Engineering vereinfacht und damit die Entwicklungszeit reduziert. Das entwickelte Ausführungsmodul unterstützt die einfache, intuitive Definition von Aktionen, die Ausführung von sequenziellen und nebenläufigen Aktionen sowie von endlichen und kontinuierlichen Aktionen, die Aktionssynchronisierung und die Erkennung und Behandlung von Störsituationen.

Zur Zielorientierung ruft das Ausführungsmodul ein Planungsmodul auf, welches einen Plan zurück liefert. Dieser besteht aus einer partiell geordneten Liste von Aktionen, deren Abwicklung von der aktuellen Ausgangssituation zur gewünschten Zielsituation führt. Planungssysteme, die sich anhand ähnlicher Planungsprobleme wie dem Hol- und Bringdienst in einem internationalen Wettbewerb messen, existieren bereits im Forschungsbereich der KI. Nach eingehender Prüfung haben sie sich für die Integration in die Kontrollarchitektur als geeignet erwiesen. Die Aufgaben- und Domänenbeschreibung zur Planung werden vom Ausführungsmodul vor jeder Planung dynamisch mit Fakten aus einer Wissensbasis erstellt. Die symbolische Beschreibung beruht auf einer für den Hol- und Bringdienst neu entwickelten Ontologie.

Die im Rahmen dieser Arbeit entwickelte modulare Kontrollarchitektur für Roboterassistenten wurde auf dem Gesamtsystem Care-O-bot® II implementiert und getestet. Care-O-bot® II ist ein Roboterassistent für Haushalt und Pflege bestehend aus einer mobilen Plattform und einem Korpus mit Manipulatorarm und neigbarem Sensorkopf zur Umgebungswahrnehmung. Ein mobiles Bedienpanel dient als Mensch-Maschine-Schnittstelle. Zum Nachweis der Anwendbarkeit der Kontrollarchitektur wurde Care-O-bot über mehrere Tage dauerhaft in einer Modellwohnung betrieben. Es zeigte sich, dass die neu entwickelte modulare Kontrollarchitektur die an sie gestellten Anforderungen erfüllt und der Roboterassistent das geforderte Verhalten hat: Die durchgeführten Experimente bestätigen die Zielorientierung bei der Aktionsausführung sowie eine hohe Robustheit gegenüber Ungenauigkeiten in der Wahrnehmung und gegenüber wechselnden Umgebungsbedingungen. Störsituationen werden sicher erkannt und Gegenmaßnahmen eingeleitet. Kann der Roboterassistent die Störung nicht alleine lösen, so wird von ihm interaktiv der Mensch mit einbezogen.

Insgesamt konnte mit der Realisierung des Hol- und Bringdienstes die Modularisierung der Verhaltenskontrolle in Aktionen nachgewiesen werden. Die Übertragbarkeit einzelner Module wurde durch deren Einsatz in zwei Ausstellungsrobotern nachgewiesen. Aufgabe zukünftiger Arbeiten wird es jetzt sein, die Geschicklichkeit der Roboterassistenten zu erhöhen und das Tätigkeitsspektrum zu erweitern. Neue Fähigkeiten können zu der Kontrollarchitektur einfach als neue Aktionen hinzugefügt werden. Um die Bedienung der Roboterassistenten noch zu vereinfachen, sollen neue Fähigkeiten und Objekte außerdem durch Zeigen und Beobachten selbstständig vom Roboterassistenten erlernt werden. Gelingt dies, so werden in Zukunft die Roboterassistenten im Privathaushalt und in Fertigungshallen als Helfer des Menschen eingesetzt werden können.

9 Summary

A modular control architecture for robot assistants to do fetch-and-carry duties

Robot assistants are expected to be clever helpers for humans characterized by their advanced level of interaction and their ability to cope with both natural environments at homes and at shop floors. They should communicate and interact in a "human-like" way and therefore should take into account shape and mobility of the human body as well as the performance and versatility of the human senses. The ability to adapt their behavior to changing situations is given by their internal control architecture, which is characterized by modularization, e.g. the identification of functional roles, and by the structured interaction between these modules.

Presently, robot assistants exist only in form of early prototypes in a few research laboratories. Each one is a specialist, able to deliver only one kind of service in only one kind of environment. They have not yet proved their cost effectiveness, but this is the main blocking point for companies for investment. The robot assistants need to deliver a wide range of services to meet customer needs. Therefore, robot assistants need a control architecture that supports an adaptation to a wide variety of abilities.

Common ground of all service fields of robot assistants is their need to be mobile, to do something physical at changing locations, to gather information from its environment and to communicate and interact with humans. The robot assistant must act in a goal-oriented and reliable way in human environments. As a representative service, the fetch-and-carry duty has been analyzed, and requirements for the design of a modular control architecture and its subsystems have been identified within the scope of this doctoral thesis: the control architecture has to support a robot control as an interface to the sensors and actuators, it needs an explicit representation of the world, it needs a decision making device for the goal-oriented behavior and it needs an execution module to control the functional interaction between these subsystems that adapt the behavior of the robot assistant to changing situations.

Based on these identified requirements for a modular control architecture, different concepts were generated, weighted and selected for all subsystems. Afterwards, the favorite concepts were integrated into one total concept:

A multi-layer architecture forms the main structure of the control architecture. This decision is based on a review of the state of the art. Multi-layer architectures are able to connect fast closed-loop algorithms of robot controls with slow cognitive processes of planning systems by

an execution module. Additional modules are a symbolic planning system, a knowledge base, a man-machine-interface and a robot control.

The abilities of the robot assistant are provided by the robot control and handed to the execution module as so called actions. These are closed-loop controllers for specific functions with interfaces to the hardware of the robot assistant. Examples include velocity controllers and position controllers for the platform and the manipulator as well as signal processing and interpretation of sensory data, e.g. a map-builder or object identification and localization. Furthermore, the actions contain open-loop controllers e.g. an interpreter of trajectories for the manipulator.

The execution module activates the actions, monitors their execution, detects exceptional situations and takes care of error recovery. The robot control returns state information and error messages for monitoring. Additionally, it returns preprocessed sensory data for decision making of the execution module.

Within the scope of this doctoral thesis, a new execution module was built using a scripting language. In comparison to other execution layers reported to be built in Lisp, C or Java, the scripting language is easier to integrate into existing systems. It is interpreted at runtime and supports a command line prompt. This simplifies engineering and reduces development time. The new execution module supports easy, intuitive definition of actions, execution of discrete or continuous actions as well as execution of sequential or concurrent actions and their synchronization. Actions can create and call other actions hierarchically and any number of arguments can be transferred. There are mechanisms for exception handling to detect errors and initiate recovery actions.

In order to act goal-directed, the execution module of the robot assistant sends a request to a symbolic planning system to generate a list of actions for reaching the goal of a given task. The list of actions is executed step by step by the execution module. The request contains the symbolic description of the task to solve and of the domain. The domain description is dynamically computed on every request with facts of a knowledge base by the execution module. The symbolic description is based on a newly developed ontology for fetch-and-carry duties.

Available planning systems of the international planning competition were investigated for planning fetch-and-carry duties. They are suitable to be integrated into the control architecture by using their standardized interfacing language PDDL. Its latest version supports the consideration of numeric and time constraints.

The developed control architecture has been implemented and evaluated with Care-O-bot® II, a robot assistant for home care. Care-O-bot® II is equipped with a sensor head, manipulator, active walking aid, and a hand-held control panel. The manipulator with six degrees of freedom and a range of about one meter is designed specifically for household tasks. The manipulation of different household objects has been implemented successfully. This includes grasping or placing objects from/to fixed locations as well as the direct passing of objects to/from the human.

In order to demonstrate the fetch-and-carry capabilities of Care-O-bot® II, a sample home environment, containing a sofa, table, and two bookshelves was set up. Several objects of different types (bottle, juice pack or crisps box) were placed in random positions at different locations. A user can sit on the sofa and command Care-O-bot® II by its control panel.

By entering a command, the task is given to the execution module. It configures a domain description with the current position of objects based on the facts of the knowledge base. The planning module returns the list of actions, containing move-, grasp- and drop-actions. These are then executed step by step.

The move-actions trigger the navigation of Care-O-bot's mobile platform. The world-coordinates are connected to the planned, symbolic locations within tables of the knowledge base. For navigation, Care-O-bot® II uses a flexible path planning method for nonholonomic mobile robots within the robot control. A geometric planner generates a path based on a static map of the robot assistant's environment. The generated path is smoothened and eventually modified in reaction to dynamic obstacles.

For the grasp-actions, a new method for grasping different objects autonomously, based on camera and 3-D laser scanner information was implemented: With the help of a camera, objects are initially taught to the robot. The reference images are saved in the knowledge base, related to their symbolic name and type. In order to grasp these objects, the corresponding reference image is loaded and compared to the current image of the camera. Afterwards, Care-O-bot® II tilts its sensor head for laser-scanning the environment. By fusing the data from the camera-based object detection and the 3-D distance information provided by the scanner, the exact location of the object is computed.

The drop-actions use laser-scanning as well: after moving in front of the dropping location, the robot assistant scans the environment and detects free space. A collision-free trajectory is calculated with the same action as for grasping. Care-O-bot® II moves its manipulator along the trajectory a few centimeters above the surface. Then the execution module starts two

concurrent actions letting the arm down and monitoring a force-torque-sensor simultaneously. As soon as the object touches the surface, the contact is detected, the actions stop and the gripper opens.

Every action observes its effect. E.g. with in-finger sensors, the grasp-action checks whether the object is really grasped or not. If an exception occurs, another action is started to recover. If the running action cannot recover the exception, a message is hierarchically sent to the upper action, until one can recover the exception. At last, the message is given to the control panel of the human to help the robot assistant. E.g. if the object detection fails (for example because of insufficient light conditions), an interactive action is started, sending the current image of the camera to the hand-held control panel and the human is asked to mark the desired object. Afterwards, the execution of the grasp-action continues regularly. If there are several same objects in the camera image, the execution module can decide which one is the desired object by comparing the current view with the expected facts from the knowledge base.

Further, every action enters its effect to the knowledge base to keep it up to date. E.g. if the robot assistants lifts an object, the symbolic information of being on the furniture is deleted and of being in the gripper is added.

The experiments with Care-O-bot® II have proven that the modular control architecture fulfills the identified requirements. The robot assistant's behavior is goal-directed and reliable.

Some of the modules of the control architecture, especially the execution module, were transferred to other service-robots for an exhibition. This attests their universal character.

The current control architecture requires the manual programming of the knowledge base. Ongoing research concentrates on algorithms for improved scene analysis to automatically generate and maintain the facts by autonomous exploration by the robot assistant. The abilities of the robot assistant will be improved and new capabilities added to the repertoire of actions.

10 Literatur und Links

ActivMedia 2002 **ActivMedia Robotics, LLC :** *Intelligent Guided Robots*. (2002)
URL: http://www.activrobots.com/ (15.09.2002)

Alami et al. 1998 **Alami, Rachid u.a. :** *An Architecture for Autonomy*.
In: International Journal on Robotics Research 17 (1998) Nr. 4,
S. 315-337

Arkin 1998 **Arkin, Ronald C. :** *Behavior-Based Robotics.*
Cambridge, Mass., USA: MIT Press, 1998

Barto et al. 1995 **Barto, A. G. ; Bradtke, S. J. ; Singh, S. P. :** *Learning to act using real-time dynamic programming.*
In: Artificial Intelligence 72 (1995) Nr. 1, S. 81-138

Bateman et al. 1994 **Bateman, J. A. ; Magini, B. ; Rinaldi, F. :** *The Generalized Upper Model.*
In: Working Papers 1994, European Conference on Artificial Intelligence (ECAI '94), Workshop on Implemented Ontologies, 1994, S. 34-45.
http://www.darmstadt.gmd.de/publish/komet/papers/ecai94.ps (28.5.04)

Baum 2003 **Baum, Winfried :** *Go User Manual.*
Unveröffentlichte Software-Dokumentation, Stuttgart: Fraunhofer-Institut für Produktionstechnik und Automatisierung, 2003

bdw 2004 **N. N. :** *Fingerzeig statt Fernbedienung*.
In: bild der wissenschaft 41 (2004) Nr. 1, Sonderdruck „*Die Fraunhofer-Offensive – 12 Leit-Innovationen, die Deutschland voranbringen*", S.109f

Biundo et al. 1995 **Biundo, Susanne u.a. :** *Planen und Konfigurieren*.
In: Einführung in die Künstliche Intelligenz / Görz, Günther (Hrsg.).
2. Auflage. Bonn, Paris u.a.: Addison-Wesley, 1995, S. 754-811

Blumberg 1997 **Blumberg, Bruce M. :** *Old Tricks, New Dogs: Ethology and Interactive Creatures.*
MIT, Media Laboratory, Learning and Common Sense Section, PhD thesis, 1997

Bonasso et al. 1997 **Bonasso, R. Peter u.a. :** *Experiences with an Architecture for Intelligent, Reactive Agents*.
In: Journal of Experimental and Theoretical Artificial Intelligence 9 (1997) Nr. 1, S. 237-256

Borelly et al. 1998 **Borrelly, Jean-Jacques u.a. :** *The ORCCAD Architecture*.
In: International Journal of Robotics Research 17 (1998) Nr. 4, S. 338-359

Brooks 1986 **Brooks, Rodney A. :** *A robust layered Control System for a Mobile Robot*.
In: IEEE Journal on Robotics and Automation 2 (1986) Nr. 1, S. 14-23

Brooks 1990 **Brooks, Rodney A. :** *Elephants Don't Play Chess*.
In: Robotics and Autonomous Systems 6 (1990) Nr. 1/2, S. 3-15

Brooks 2002 **Brooks, Rodney A. :** *Flesh and Machines : How Robots will change Us.*
Pantheon Books, 2002

Cassandra 1999 **Cassandra, Tony :** *Tony's POMDP Page*.(1999)
URL: http://www.cs.brown.edu/research/ai/pomdp/index.html (31.1.1999)

Chandrasekaran et al. 1999
Chandrasekaran, B. ; Josephson, John R. ; Benjamins, V. Richard : *What Are Ontologies and Why Do We Need Them?*
In: IEEE Intelligent Systems 14 (1999) Nr. 1, S. 20-26

Christaller 2001 **Christaller, Thomas :** *Künstliche Intelligenz*.
In: Fraunhofer-Magazin (2001) Nr. 1, Beilage NetzWelt, S.5

Cycorp 2003 **Cycorp, Inc. :** *Makers of the Cyc Knowledge Server for artificial intelligence-based Common Sense*. (2003)
URL: http://www.cyc.com (26.8.2003)

Dario et al. 1997 **Dario, P. u.a. :** *MOVAID: a mobile robotic system residential care to disabled and elderly people.*
In: Mobile robotics technology for health care services - 1st MobiNet Symposium, Proceedings, Athens, Greece, 15-16 May 1997 / Tzafestas, Spyros G. (Ed.). Athens, 1997, S. 9-14

Duden 1997 **Bd. 5. Duden, Fremdwörterbuch.**
6., auf der Grundlage der amtlichen Neuregelung der deutschen Rechtschreibung überarb. und erw. Aufl. Mannheim; u.a. : Dudenverl., 1997

Fikes, Nilsson 1971 **Fikes, Richard E. ; Nilsson, Nils J. :** *STRIPS: a new approach to the application of theorem proving to problem solving*.
In: Artificial Intelligence 2 (1971) Nr. 3/4, S.189-208

Firby 1989 **Firby, R. James :** *Adaptive Execution in Complex Dynamic Worlds.*
Ann Arbor : University Microfilms International, 1989
Zugl.: New Haven, Yale Univ., PhD thesis, 1989

Fox, Long 2001 **Fox, Maria ; Long, Derek P. :** *PDDL+ : Planning with Time and metric Resources.*
University of Durham, UK, Department of Computer Science, Technical Report, 2001.
http://www.dur.ac.uk/d.p.long/competition.html (27.2.2002)

Fox, Long 2002 **Fox, Maria ; Long, Derek P. :** *PDDL2.1 : An Extension to PDDL for Expressing Temporal Planning Domains.*
University of Durham, UK, Department of Computer Science, Technical Report, 2002.
http://www.dur.ac.uk/d.p.long/competition.html (5.2.2003)

Fridman, Hafner 1997 **Fridman, Noy N. ; Hafner, C. D. :** *The State of the Art in Ontology Design.*
In: AI Magazine 18 (1997) Nr. 3, S. 53-74.

Gat 1991 **Gat, Erann :** *Reliable Goal-Directed Reactive Control of Autonomous Mobile Robots.*
Blacksburg, VA, USA, Virginia Polytechnic Institute and State Univ., PhD thesis, 1991

Gat 1992 **Gat, Erann :** *Integrating Planning and Reacting in a Heterogeneous Asynchronous Architecture for Controlling Real-World Mobile Robots.* In: Proceedings of the 10th National Conference on Artificial Intelligence AAAI-92, 12.-16.7.1992, San Jose/ Calif. / American Association for Artificial Intelligence. Menlo Park, CA, USA : AAAI Press, 1992, S. 809-815

Geffner 2000 **Geffner, Héctor :** *Modeling and Problem Solving.* Folienvortrag zur 9. Herbstschule Kognitionswissenschaft, Freiburg, 10.-15. September 2000

Georgeff, Lansky 1987 **Georgeff, Michael P. ; Lansky, Amy L. :** *Reactive reasoning and planning.* In: Proceedings of the 6th National Conference on Artificial Intelligence AAAI-87, 13.-17.7.1987, Seattle, WA, USA / American Association for Artificial Intelligence. Los Altos, CA, USA. : Kaufmann, 1987, Vol. 2, S. 677-682

Ghallab et al. 1998 **Ghallab, Malik u.a. :** *PDDL-The Planning Domain Definition Language.* In: Proceedings of the 4th International Conference on AI Planning Systems / Simmons, Reid ; Veloso, Manuela ; Smith, Stephen (Hrsg.). Menlo Park, CA, USA : AAAI Press, 1998, 25 S.

Gómez-Pérez 1999 **Gómez-Pérez, Asunción :** *Ontological Engineering: A State of the Art.* In: Expert Update 2 (1999) Nr. 3, S. 33-43

Graefe, Bischoff 2003 **Graefe, Volker ; Bischoff, Rainer :** *Past, Present and Future of Intelligent Robots*. In: IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 2003, 16.-20.7.2003, Kobe. Piscataway, NJ : IEEE Operations Center, 2003, Vol. 2, S. 801-810

Gruber 1993 **Gruber, T. A. :** *A Translation Approach to portable Ontology Specification.* In: Knowledge Acquisition (1993) Nr. 5, S. 199-220

Hägele et al. 2001 **Hägele Martin ; Neugebauer, Jens ; Schraft, Rolf Dieter :** *From Robots to Robot Assistants.*
In: ISR 2001: Proceedings of the 32nd International Symposium on Robotics in conjunction with IMS 2001, April 19-21, 2001, Seoul, Korea, Vol. 1, S. 404-409

Hans et al. 2002 **Hans, Matthias ; Graf, Birgit ; Schraft, Rolf Dieter :** *Robotic Home Assistant Care-O-bot : Past - Present - Future.*
In: IEEE Industrial Electronics Society u.a.: IEEE ROMAN 2002 : Proceedings, 11th IEEE International Workshop in Robot and Human Interactive Communication, 25.-27. September 2002, Berlin. Piscataway, NJ : IEEE, 2002, S. 380-385

Hans, Schraft 2003 **Hans, Matthias ; Schraft, Rolf Dieter :** *Aktionsplanung und -ausführung für Assistenzsysteme – Die Kontrollarchitektur von Care-O-bot® II.*
In: atp - Automatisierungstechnische Praxis 45 (2003) Nr. 11, S. 40-45

Helms et al. 2002 **Helms, Evert ; Dünne, Markus ; Hans, Matthias u.a.:** *rob@work: Assistentensysteme als Helfer in der Produktion*.
In: Robotik 2002 : Leistungsstand, Anwendungen, Visionen, Trends. Düsseldorf : VDI Verlag, 2002, S. 661-667 (VDI-Berichte 1679)

Honda 2004 **Honda Motor Co., Ltd. :** *Asimo*. (2004)
URL: http://world.honda.com/ASIMO/ (23.04.2004)

Huber 1999 **Huber, Marcus J. :** *JAM Agents in a Nutshell, Version 0.61+0.79i.*
URL: http://members-http-4.rwc1.sfba.home.net/marcush/IRS/, (1999)

IPC 2002 **N. N. :** *International Planning Competition.* (2002)
URL: http://www.dur.ac.uk/d.p.long/competition.html. (23.8.2003)

Jung, Fischer 1998 **Jung, Christoph G. ; Fischer, Klaus :** *Methodological Comparison of Agents.*
Saarbrücken, 1998. (DFKI Research Report RR-98-01)

Köhler 2000 **Köhler, Jana :** *Handlungsplanung – ein erster Einstieg*. (2000)
URL: http://www.informatik.uni-freiburg.de/~koehler/was-ist-planen.html (23.8.2003)

Konolige 1997 **Konolige, Kurt :** *COLBERT: A Language for Reactive Control in Sapphira.* In: Advances in artificial intelligence : proceedings / KI-97, 21st Annual German Conference on Artificial Intelligence, Freiburg, Germany, September 9 - 12, 1997 / Gerhard Brewka (Hrsg.). Berlin u.a.: Springer, 1997, 33 S.

Konolige, Myers 1998 **Konolige, Kurt ; Myers, Karen :** *The Saphira Architecture for Autonomous Mobile Robots.* In: Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems / Kortenkamp, David (Hrsg.). Cambridge, MA, USA: MIT Press, 1998, S. 211-242

Kortenkamp et al. 1998 **Kortenkamp, David ; Bonasso, R. Peter ; Murphy, Robin (Hrsg.) :** *Artificial Intelligence and Mobile Robots: Case Studies of Successful Robot Systems.* Cambridge, MA: MIT Press, 1998

KUKA 2003 **KUKA Roboter GmbH :** *Robocoaster. Fahrspaß in der sechsten Dimension – der erste Passagierroboter der Welt.* Gersthofen, 2003 – Firmenschrift.

Levi et al. 2003 **Levi, Paul ; Regele, Ralf ; Bott, Wolfgang :** *ProRobot – Predictions for the Future Development of Humanoid Robots.* In: Autonome Mobile Systeme 2003 / Dillmann, Rüdiger ; Wörn, Heinz ; Gockel, Thilo (Hrsg.). Berlin u.a.: Springer, 2003, S. 292-302

Littman et al. 1997 **Littman, Michael ; Cassandra, Tony ; Kaelbling, Leslie :** *POMDP information page*. (1997) URL: http://www.cs.duke.edu/~mlittman/topics/pomdp-page.html, (1.10.2003)

Maes 1990 **Maes, Pattie :** *Situated agents can have goals.* In: Designing Autonomous Agents / Maes, P. (Hrsg.). Cambridge, MA, USA: The MIT Press, 1990, S. 49-70.

Maes 1991 **Maes, Pattie :** *The agent network architecture (ana).*
In: SIGART Bulletin 2 (1991) Nr. 4, S. 115-120

Mataric 1997 **Mataric, Maja J. :** *Behavior-Based Control: Examples from Navigation, Learning, and Group Behavior.*
In: Journal of Experimental and Theoretical Artificial Intelligence 9 (1997) Nr. 2-3, S. 323-336

Mitsubishi 2004 **Mitsubishi Heavy Industries, Ltd. :** *wakamaru*. (2004)
URL: http.//www.wakamaru.net, (23.4.2004)

Müllerschön 1996 **Müllerschön, Joachim :** *Ein Verfahren zur automatischen Generierung von Steuerprogrammen für Roboterfahrzeuge.*
Berlin u.a.: Springer, 1996.
Stuttgart, Univ., Diss., 1996

MySQL 2003 **MySQL AB :** *MySQL: The World's Most Popular Open Source Database.* (2003)
URL: http://www.mysql.com/, (16.9.2003).

NEC 2004 **NEC Corporation :** *NEC Personal Robot Center*. (2004)
URL: http://www.incx.nec.co.jp/robot/robotcenter_e.html, (23.4.2004)

Neobotix 2003 **Neobotix :** *Product RA-6A*. (2003)
URL: http://www.neobotix.de/products.RA-6A.html, (1.10.2003)

Nicolescu, Mataric 2002
Nicolescu, Monica N. ; Mataric, Maja J. : *A Hierarchical Architecture for Behavior-Based Robots.*
In: Bringing people and agents together - autonomous agents and multiagent systems; Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems, 15.-19. Juli 2002, Bologna, Italien. / Castelfranchi, Cristiano (Hrsg.). New York: ACM Press, 2002, Vol.1, S.227-233

Nilsson 1984 **Nilsson, Nils J. :** *Shakey the robot.*
Menlo Park, California, SRI International, Technical note 323, 1984.

Oesterreich 1997 **Oesterreich, Bernd :** *Objektorientierte Softwareentwicklung mit der Unified Modelling Language.*
3., aktual. Aufl. München; Wien: Oldenbourg, 1997

Pednault 1989 **Pednault, Edwin P. D. :** *ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus.*
In: KR'89 - Proc. of the 1st International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Ontario, Canada, 15.-18. 05.1989 / Hrsg.: Ronald J. Brachman et al.
San Mateo, Calif., USA: Morgan Kaufman, 1989, S. 324-332

Praßler et al. 1999 **Praßler, Erwin ; Hägele, Martin ; Lawitzky, Gisbert ; Stopp, Andreas:** *Executive Summary*. In: Forschungsantrag *MORPHA: Anthropomorphe Assistenzsysteme*, gefördert vom BMBF, Stand Mai 1999.

Puterman 1994 **Puterman, Martin L. :** *Markov decision processes: discrete stochastic dynamic programming.*
New York u.a.: Wiley, 1994

Python 2003 **Python Software Foundation :** *Python Programming Language*. (2003)
URL: http://www.python.org, (22.8.2003)

Rintanen, Hoffmann 2001
Rintanen, Jussi ; Hoffmann, Jörg : *An Overview of Recent Algorithms for AI Planning*.
In: Künstliche Intelligenz 15 (2001) Nr. 2, S. 5-11

Ritter 2003 **Ritter, Arno :** *Ein Multi-Agenten-System für mobile Einrichtungen in Produktionssystemen*.
Heimsheim: Jost Jetter Verlag, 2003. Stuttgart, Univ., Diss., 2003

Roy et al. 2000 **Roy, Nicholas u.a. :** *Towards Personal Service Robots for the Elderly.*
Workshop on Interactive Robots and Entertainment (WIRE 2000),
April 30 - May 1, 2000, Pittsburgh, PA, USA.
http://www.ri.cmu.edu/pubs/pub_3390.html (27.05.2004)

Russel, Norvig 1995 **Russel, Stuart J.; Norvig, Peter :** *Artificial Intelligence: A Modern Approach*.
Engelwood Cliffs, NJ, USA: Prentice-Hall, 1995

Schaeffer, May 1999 **Schaeffer, Christoph ; May, Till :** *Care-O-bot™: A System for Assisting Elderly or Disabled Persons in Home Environments*.
In: Assistive Technology on the Threshold of the New Millennium, Proc. AAATE 99, 1.-4.11.1999, Düsseldorf / Bühler, Christian (Hrsg.).
Amsterdam u.a. : IOS Press, 1999, S. 340-345

Schneider 1997 **Schneider, Hans-Jochen (Hrsg.) :** *Lexikon der Informatik und Datenverarbeitung*.
4., actual. und erw. Aufl. München u.a. : Oldenbourg, 1997

Schraft, Schmierer 1998
Schraft, Rolf Dieter ; Schmierer, Gernot: *Serviceroboter – Produkte, Szenarien, Visionen*.
Berlin u. a.: Springer, 1998

Schraft, Volz 1996 **Schraft, Rolf Dieter ; Volz, Hansjörg :** *Serviceroboter: Innovative Technik in Dienstleistung und Versorgung.*
Berlin u.a.: Springer, 1996

Sesseler 2002 **Sesseler, Ralf :** *Eine modulare Architektur für dienstbasierte Interaktionen zwischen Agenten*.
Berlin, Universität, Diss., 2002

Simmons 1994 **Simmons, Reid G. :** *Structured control for autonomous robots*.
In: IEEE Transactions on Robotics and Automation 10 (1994) Nr. 1, S. 34-43.

Simmons, Apfelbaum 1998
Simmons, Reid G. ; Apfelbaum, David : *A Task Description Language for Robot Control*.
In: Innovations in theory, practice and applications : Proceedings 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS '98; October 13 - 17, 1998, Victoria, B. C., Canada. Piscataway, NJ : IEEE Service Center, 1998, Vol. 3, S. 1931-1937

Simon 1969
Simon, Herbert A. : *The Sciences of the Artificial*.
Cambridge, MA, USA: MIT Press, 1969

Sowa 2000
Sowa, John F. : *Knowledge Representation: Logical, Philosophical, and Computational Foundations*.
Pacific Grove, CA, USA: Brooks/Cole, 2000

Strube 1995
Strube, Gerhard (Hrsg.) : *Kognition*.
In: Einführung in die Künstliche Intelligenz / Görz, Günther (Hrsg.), 2. Aufl.. Bonn, u.a.: Addison-Wesley, 1995, S. 299-359

Strube 2000
Strube, Gerhard : *Handlungskontrolle in dynamischen Umgebungen.*
Folienvortrag zur 9. Herbstschule Kognitionswissenschaft, Freiburg, 10.-15. September 2000.

Traub 2002
Traub, Andreas : *Eine komponentenbasierte Softwarearchitektur für mobile Serviceroboter.*
Heimsheim: Jost Jetter Verlag, 2002. Stuttgart, Univ., Diss., 2002

Tyrrell 1993
Tyrrell, Toby : *Computational Mechanisms for Action Selection.*
Edinburgh, Univ., Centre for Cognitive Science, PhD thesis, 1993

UNECE 2002
United Nations Economic Commission for Europe :
UNECE issues its 2002 World Robotics survey.
Genf, 1. Oktober 2002 (Press Release ECE/STAT/02/01)

UNECE 2003
United Nations Economic Commission for Europe :
UNECE issues its 2003 World Robotics survey.
Genf, 17. Oktober 2003 (Press Release ECE/STAT/03/P01)

Westkämper 2001 **Westkämper, Engelbert :** *Mehr Intelligenz in der Maschine. Technische Intelligenz und Miniaturisierung im Maschinen- und Anlagenbau.* In: F&M 109 (2001) Nr. 10, S. 75-77

Wichert et al. 2002 **Wichert, Georg von u.a.:** *The Robotic Bar - An Integrated Demonstration of Man-Robot Interaction in a Service Scenario.* In: IEEE ROMAN 2002 : Proceedings, 11th IEEE International Workshop in Robot and Human Interactive Communication, September 25-27, 2002, Berlin. Piscataway, NJ : IEEE, 2002, S. 374-379

Wooldridge 1999 **Wooldridge, Michael J. :** *Intelligent Agents.* In: Multiagent Systems - a modern approach to distributed artificial intelligence / Weiss, Gerhard (Hrsg.). Cambridge, MA, USA: MIT Press, 1999, S: 21-35