Workshop SelfOrganization of Adaptive Behavior (SOAVE 2004),
Fortschritt-Berichte VDI, S. 187-197, Reihe 10, Nr. 743

Eine hybride Steuerarchitektur für einen interaktiven mobilen Serviceroboter

H.-J. Böhme, T. Wilhelm, C. Schröter, A. König, C. Martin
Technische Universität Ilmenau, Fachgebiet Neuroinformatik, PF 100565,
98684 Ilmenau
hans-joachim.boehme,torsten.wilhelm,christof.schroeter,alexander.koenig,
christian.martin@tu-ilmenau.de

Zusammenfassung

Dieser Beitrag beschreibt eine hybride Steuerarchitektur für einen interaktiven mobilen Serviceroboter. Dabei werden zunächst die Prämissen diskutiert, die den Entwurf der Gesamtarchitektur unter Berücksichtigung der zu integrierenden Teilleistungen (Navigation und Mensch-Roboter-Interaktion) des Roboters und der bestehenden Hardwarevoraussetzungen determinieren. Nach der Vorstellung der Gesamtarchitektur wird eine knappe Darstellung der einzelnen Teilmodule gegeben. Implementationsaspekte, erste experimentelle Untersuchungen und Erfahrungen mit der vorgeschlagenen Architektur bilden den Abschluss des Beitrags.

1 Motivation und Einordnung

Um Serviceroboter erfolgreich in realen Anwendungsszenarien einzusetzen, müssen diese über eine große Vielfalt von Teilleistungen verfügen. Diese reichen vom (teil)autonomen Aufbau konsistenter Umgebungsmodelle über die robuste Schätzung des aktuellen Zustands bezüglich der Umgebung bis hin zu natürlich und intuitiv bedienbaren Mensch-Maschine-Schnittstellen. Unter Steuerarchitektur und Verhaltenskoordination sollen die Methoden zusammengefasst werden, die für das korrekte Wechselspiel aller auf dem Serviceroboter implementierten Teilleistungen oder Teilverhalten verantwortlich sind. Typischerweise werden die je nach Anwendungsszenario notwendigen Teilleistungen zunächst weitestgehend unabhängig voneinander entwickelt und verifiziert. Mit zunehmender Komplexität des dabei entstehenden Gesamtsystems kommt der Steuerarchitektur und der darin realisierten Verhaltenskoordination eine immer bedeutendere Rolle zu. Diesem Aspekt wird oft wenig Aufmerksamkeit geschenkt, da die Integration zu einem robust funktionierenden Gesamtsystem eher als notwendiges Übel mit wenig wissenschaftlichem Gehalt erscheint. Unsere Erfahrungen, die einem langfristig angelegten Projekt zum Einsatz eines interaktiven mobilen Shopping-Assistenten in einem Baumarkt entstammen, zeigen jedoch, dass der Erfolg einer solchen Applikation maßgeblich von der Frage der Steuerarchitektur determiniert wird. Untermauert werden kann diese These durch eine Reihe von Publikationen (siehe z.B. [16], [4] oder [12]), in denen verschiedene Autoren in ähnlicher Weise argumentieren.

Betrachtet man die historische Entwicklung (siehe dazu insbesondere [2] und [12]), so begann diese mit dem pipeline-orientierten Sense-Plan-Act-Paradigma, das eine universelle Wahrnehmung der Welt annimmt, die Wahrnehmung vollständig von der Verhaltensgenerierung trennt und den Schwerpunkt klar auf den Planungsaspekt legt.

Betont werden sollte die diesem Ansatz innewohnende universelle Repräsentation der Welt, die damit ausschließlich sensorisch geprägt ist, Aspekte einer aktiven, handlungsgetriebenen Wahrnehmung vollständig vernachlässigt und sich stark an das Informationsverarbeitungsparadigma nach MARR [13] mit seiner Trennung von Wahrnehmung und Verhaltensgenerierung anlehnte. Aufgrund der genannten Schwachstellen des Sense-Plan-Act-Paradigmas wurden verschiedene Architekturen vorgeschlagen, die eine Dekomposition in verschiedene Verhaltensmodule vornehmen, diese parallel und teilweise konkurrierend zueinander implementieren, und deren Wechselwirkung durch eine Kontrollstruktur übernommen wird, die den Zugriff der verschiedenen Verhaltensmodule auf die Aktorik steuert. Das wohl bekannteste Beispiel ist die Subsumption-Architektur von BROOKS [7], die die Verwendung von internen Repräsentationen (zumindest solchen auf symbolischem, expliziten Niveau) komplett ablehnt und die Welt als deren beste Repräsentation auffasst. Charakteristisch ist hier weiterhin die angestrebte enge und möglichst unmittelbare Kopplung zwischen Wahrnehmung und Aktion (reaktive Systeme). Nachdem die Subsumption-Architektur den radikalen Bruch mit dem klassischen, pipeline-orientierten Paradigma der Informationsverarbeitung vollzogen hatte, wurde offenbar, dass der völlige Verzicht auf jegliche Arten interner Repräsentationen eine Reihe von Nachteilen mit sich bringt: so bedeutet dies, dass keinerlei explizite interne Planung möglich ist, da diese notwendigerweise ein wie auch immer geartetes internes Modell der externen Welt erfordert. Dies führte letztlich zur Entwicklung einer Architekturfamilie, deren bekanntester Vertreter die so genannte 3T-Architektur [5] ist. Dieses wiederum vertikal organisierte Architekturkonzept (eine sehr gute Charakterisierung findet man in [12], Kapitel 8), trägt der Notwendigkeit von reaktiven und planenden Komponenten innerhalb einer Steuerarchitektur Rechnung. Während schnelle reaktive Komponenten (Skills) mit ihren typischen unmittelbaren Sensor-Aktor-Kopplungen ohne Zuhilfenahme interner Zustandsinformation auf der unteren Hierarchieebene (Skill Management) zu finden sind, stellt die mittlere Ebene (Sequencer), deren Algorithmen von interner Zustandsinformation abhängen, die aber selbst noch keine Planung im Sinne von Suchverfahren realisiert, quasi eine Vermittlungsfunktion zwischen reaktiver unterer Ebene und planender oberer Hierarchieebene dar. Verhaltensbasierte Steuerarchitekturen, wie sie in [2] umfassend beschrieben und diskutiert werden, wurden sehr stark durch die Forschungen auf den Gebieten Ethologie und Psychophysik geprägt. Aus einer eher ingenieurwissenschaftlichen Sicht lassen sich diese Architekturen als eine mögliche Ausprägung der unteren beiden Ebenen der 3T-Architektur auffassen.

Analysiert man die Vielfalt der heute vorgeschlagenen Steuerarchitekturen, so stellen diese meist eine Kombination aus den bislang benannten prinzipiellen Konzepten unter Berücksichtigung der realen Gegebenheiten der jeweiligen Applikation dar, da mittlerweile Einklang darüber herrscht, dass es *die* ultimative Steuerarchitektur nicht gibt und in absehbarer Zeit wohl auch nicht geben wird. Die zur Verhaltenskoordination entwickelten Verfahren tragen aus heutiger Sicht im wesentlichen zwei grundlegenden Aspekten Rechnung: Zum einen versucht man, der Komplexität des Gesamtsystems durch geeignete Zerlegung (Dekomposition) des Gesamtverhaltens des Systems in sinnvoll abgrenzbare Teilverhalten und deren Integration in eine problemangepasste Steuerarchitektur zu begegnen. Zum anderen werden in verstärktem Maße die auftretenden Unsicherheiten sowohl beim Verhalten des Roboters selbst als auch hinsichtlich der Umgebungseigenschaften mit geeigneten Modellierungsparadigmen berücksichtigt [22].

In den vergangenen Jahren bildete der Aspekt der Verhaltenskoordination innerhalb hybrider Steuerarchitekturen den methodischen Schwerpunkt. So entstanden Ansätze, die ein weiches Umschalten zwischen einzelnen Elementarverhalten mittels dynamischer Systeme modellieren [20][1], die ver-

schiedene Verhalten auf Motor-Schemata abbilden [2] oder die Zerlegung von globalen Missionen in einen Ablauf von Teilprozessen zerlegen und dessen Abarbeitung über definierte Ereignis-Monitore steuern [5].

Obwohl weitgehende Übereinstimmung bezüglich der Anforderungen an eine moderne Steuerarchitektur herrscht, hat in den letzten Jahren quasi jede Arbeitsgruppe *ihre* spezielle Architektur entwickelt und implementiert. Zunehmend wird deutlich, dass dies in mehrfacher Hinsicht nachteilig ist:

- Der Austausch von Modulen und Teilkomponenten ist nur schwer möglich.
- Verschiedene Architekturen lassen sich hinsichtlich ihrer Leistungsfähigkeit nur schwer bewerten.
- Die Abstraktion von der vorhandenen Roboter-Hardware ist meist nicht soweit realisiert, dass eine Architektur auf verschiedenen Plattformen und damit auch arbeitsgruppenübergreifend eingesetzt und entwickelt werden kann.

Um in Richtung Vereinheitlichung und Standardisierung von Steuerarchitekturen voranzukommen, wurden verschiedene Projekte initiiert, z.B. OPEN-R ([9]) oder die an der CMU entwickelte Navigationsarchitektur CARMEN (Carnegie Mellon Navigation Toolkit [18]). Ein weiteres Beispiel in dieser Entwicklung ist das CLARAty-Projekt (Coupled Layer Architecture for Robotic Autonomy [24]), mit dem folgende Ziele verfolgt werden:

- Etablierung einer neu entworfenen Steuerarchitektur als Standard
- Entwurf mittels UML (Unified Modeling Language)
- Objektorientierung in Design und Implementation
- Unterstützung der Open-Source-Bewegung
- Entwicklung einer Standard-Template-Bibliothek
- ausführliche Code-Dokumentation

2 Gesamtarchitektur

Nachfolgend werden zunächst die Prämissen definiert, die einerseits für den Entwurf der vorgeschlagenen Architektur maßgeblich waren und die in aktuellen Arbeiten zum Entwurf von Steuerarchitekturen (siehe z.B. [17]) herausgearbeitet wurden:

- **Transparenz:** Sicherstellung einer größtmöglichen Übersichtlichkeit, die die Einarbeitung in das bestehende Gesamtsystem, die Fehlersuche, die Software-Modellierung und die Integration weiterer Teilmodule wesentlich vereinfachen.
- **Modularität:** Funktional als relativ eigenständig definierte Module sollen weitgehend als solche implementiert werden können.
- **Erweiterbarkeit:** Module tauschen Informationen ausschließlich über eine zentrale Instanz aus, was neben der leichten Erweiterbarkeit auch die Übersichtlichkeit des Gesamtsystems gewährleistet.

- **Portabilität:** Durch eine Abstraktion von der vorhandenen Roboter-Hardware soll die Architektur auf verschiedenen Plattformen eingesetzt werden können.
- **Effizienz:** Die Architektur muss den vorliegenden Erfordernissen hinsichtlich Echtzeitfähigkeit und Performanz gerecht werden.

Die aus den definierten Anforderungen abgeleitete Architektur besteht aus eigenständigen Teilmodulen, die ausgehend von der aktuellen sensorischen Situation entweder Aktionen des Systems vorschlagen oder Sensorinformation verarbeiten. Ein Modul zum Dialogmanagement, das maßgeblich typische Zustände des Gesamtsystems determiniert, kontrolliert, welche der Verhaltensmodule zum aktuellen Zustand aktiv werden dürfen, wobei *aktiv* den Zugriff auf die Aktuatorik des Systems bedeutet. In diesem Kontext weist unser Ansatz Ähnlichkeiten zu der in [3] vorgeschlagenen, situationsbasierten Architektur auf, wobei diese im Gegensatz zu der hier diskutierten Architektur nicht über eine zentrale Kommunikationsstruktur verfügt. Neben den genannten Prämissen muss die Architektur auch die vorhandenen Hardware-Ressourcen berücksichtigen sowie die Anbindung von Sensorik und Aktorik unterstützen. Die in diesem Beitrag beschriebene Architektur beinhaltet direkte Sensor-Aktor-Kopplungen (Hindernisvermeidung), Planungskomponenten auf kurzer (Personen- und Posentracking) und mittlerer Zeitskala (Pfadplanung), Weltwissen in Form zentral verfügbarer (Umgebungsmodell) oder nur lokal genutzter (Hautfarbmodell) Repräsentationen und eine universelle Kommunikationsstruktur, über die alle Teilsysteme Informationen austauschen. Sie wurde auf dem Roboter PERSES (B21 von RWI IS Robotics, ausgestattet mit omnidirektionaler Kamera, Kopf-Kameras, Sonarsensoren, Touchdisplay und Gesicht), der als Plattform für den interaktiven mobilen Shopping-Assistenten fungiert, umgesetzt und ordnet sich in die letztgenannte Familie der hybriden Architekturen ein.

Abbildung 1 zeigt die umgesetzte Steuerarchitektur, die auf dem Roboter implementiert wurde. Als zentrale Ebene und Kommunikationsstruktur für alle Module fungiert ein *Blackboard*, das die Anbindung an die Roboter-Hardware realisiert. Auf dem Blackboard setzen alle Module auf, die Teilverhalten realisieren. Diese können zwei funktional zusammenhängenden Blöcken, Navigation und Mensch-Roboter-Interaktion (MRI), zugeordnet werden. Für jedes Teilmodul besteht die Möglichkeit, je nach Bedarf Module für Visualisierungen in einem grafischen Benutzer-Interface (GUI) anzubinden, die sowohl auf die entsprechenden Teilmodule als auch direkt auf das Blackboard zugreifen können. Weiterhin besteht via WLAN eine kontinuierliche Verbindung zwischen dem Roboter und einem Telepräsenz-Arbeitsplatz, über den ein menschlicher Fachberater bei Bedarf kontaktiert werden kann.

3 Komponenten der Gesamtarchitektur

3.1 Blackboard

PERSES ist mit zwei On-board-Rechnern (linker und rechter Rechner) ausgestattet. Entsprechend existieren intern zwei hinsichtlich ihrer Schnittstelle zu den angekoppelten Teilmodulen identisch aufgebaute Blackboards, die über eine TCP-Verbindung kommunizieren, wobei die beiden Blackboards hinsichtlich ihrer Anbindung an die Roboter-Hardware differieren. Um den Transfer von Bilddaten zwischen den beiden internen Blackboards möglichst zu vermeiden, wird der linke Rechner nahezu ausschließlich für Bildverarbeitungsprozesse verwendet, während sich der rechte Rechner um das Auslesen der nicht-visuellen Sensorik und die Ansteuerung der Aktorik kümmert.
So wird beispielsweise die Koordinate eines via Touch-Interface ausgewählten Produkts auf dem linken Rechner ermittelt und für die Planung eines Pfades zur Zielposition an den rechten Rechner übertragen.

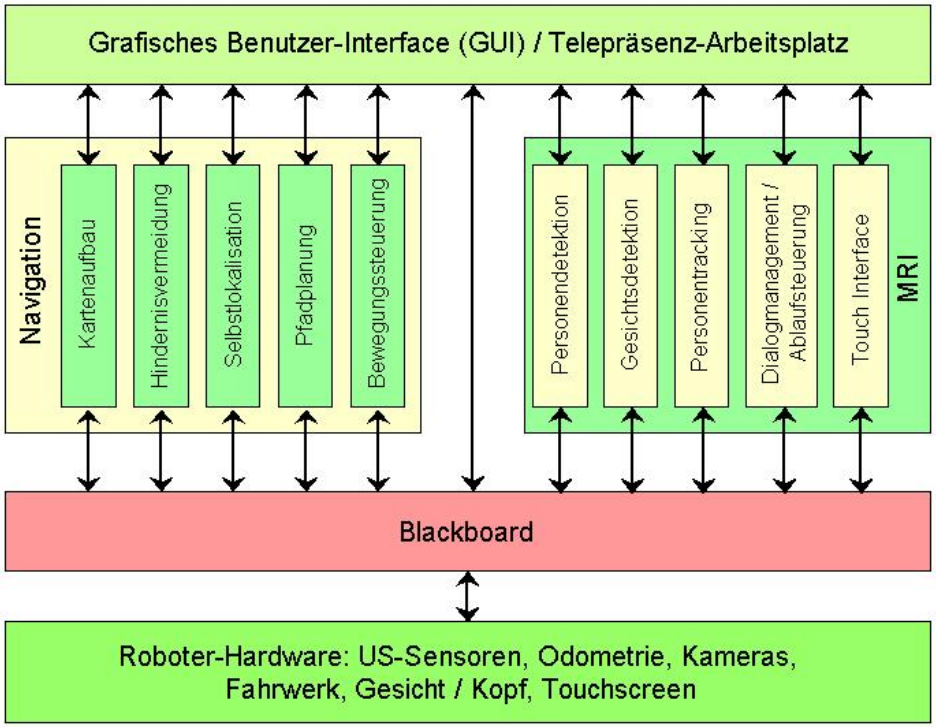

Abbildung 1. Überblick über die Gesamtarchitektur.

Das Blackboard ist selbst für die Aktualisierung der Sensordaten (Bilder, Odometrie, Sonar, usw.) verantwortlich. Auf das Blackboard können beliebige Threads für Berechnungen zugreifen, welche Sensordaten oder Ergebnisse von anderen Modulen lesen und ihre eigenen Berechnungsergebnisse auf das Blackboard schreiben. Diese werden wie die Sensordaten zwischen den beiden Blackboards synchronisiert. Weiterhin wird auf dem Blackboard vermerkt, in welchem Verhaltensmodus sich das Gesamtsystem momentan befindet. Dieser Aspekt wird in den Abschnitten 3.2 und 3.3 noch detailliert diskutiert.
Das Blackboard als zentrale Kommunikationsstruktur bietet ebenso eine Reihe eher pragmatisch motivierter Vorteile: so wird über einen Zeitstempel-Mechanismus dafür gesorgt, dass alle Sensordaten über eine einheitliche und konsistente Zeitbasis verfügen. Dies ist insbesondere für die Zuordnung der Odometriedaten zu den entsprechenden externen Sensorinformationen oder für Algorithmen zur Sensorfusion bei multimodal operierenden Verfahren, wie dem Personentracking, wesentlich. Weiterhin ermöglicht ein speziell eingerichteter Mechanismus, dass über das Blackboard aufgenommene Experimentaldaten quasi im Offline-Modus abgespielt werden können, was den Test verschiedener Verfahren ohne Vor-Ort-Präsenz des Roboters gestattet.
Nachfolgend werden zunächst die Teilmodule, die in die beiden funktionalen Blöcke *Navigation* und *Mensch-Roboter-Interaktion* eingeteilt sind, kurz beschrieben.

3.2 Mensch-Roboter-Interaktion (MRI)

Der Gesamtkomplex Mensch-Roboter-Interaktion beinhaltet die *Personendetektion*, eine *Gesichtsdetektion*, das *Personentracking* das *Dialogmanagement* sowie das *Touch-Interface* zur unmittelbaren Bedienung. Die *Situationserkennung* als integraler Bestandteil des Moduls *Dialogmanagement* nimmt im Rahmen der MRI einen Sonderstatus ein (siehe Abschnitt 3.2.2).

3.2.1 Personendetektion, Gesichtsdetektion und Personentracking

Um es dem Roboter zu ermöglichen, sein gesamtes Umfeld nach potentiellen Nutzern abzusuchen, werden eine omnidirektionale Kamera und ein Ring von Sonarsensoren verwendet. Die Personendetektion nutzt neben einer Hautfarbdetektion im omnidirektionalen Bild auch die Sonarsensoren, die ein Entfernungsprofil aus der Umgebung des Roboters liefern. Die Grundlage des Detektions- und Tracking-Systems bildet der Condensation-Algorithmus [11]. Die Berechnung der Sample-Gewichte erfolgt über eine Kombination von Hautfarbdetektion und Sonarmessungen. Dabei erhalten die Samples ein hohes Gewicht, die auf einer hautfarbenen Region liegen und sich zudem in einer Richtung befinden, für die die Sonarsensoren eine kurze Entfernung messen. Diese Art der Sensorfusion bewirkt, das die Aufmerksamkeit des Roboters auf nahe hautfarbene Objekte gerichtet wird [25]. Um die Hautfarbdetektion weitgehend unabhängig von der aktuellen Beleuchtung zu machen, wird ein automatischer Weißabgleich auf einer im omnidirektionalen Bild dauerhaft eingeblendeten Weiß-Referenz durchgeführt [25].

Probleme bereitet dem Detektions- und Tracking-Mechanismus die Tatsache, dass es außer Gesichtern auch andere Hautfarbregionen in der näheren Umgebung des Roboters geben kann. Um z.B. keine Farbdosen oder Holztüren als potentielle Nutzer zu detektieren, wird die zunächst als Hypothese getrackte Region mit einem *Gesichtsdetektor* nochmals auf das Vorhandensein eines Gesichtes verifiziert. Um ein hochaufgelöstes Bild der getrackten Region zu erhalten, wird der Kopf von PERSES in die entsprechende Richtung gedreht und der Zoom einer Frontalkamera entsprechend der ermittelten Entfernung des Objektes so eingestellt, das ein Gesicht das komplette Bild ausfüllen würde. Als eigentliches Detektionsmodul kommt das Verfahren von Viola und Jones [23] zum Einsatz, welches sowohl in Bezug auf die Detektionsgüte als auch hinsichtlich der benötigten Rechenzeit bei einem Vergleich verschiedener Verfahren am besten abschnitt (siehe dazu auch [25]).

3.2.2 Dialogmanagement / Ablaufsteuerung

Der Ablauf einer Interaktion mit einem Nutzer wird mit einem deterministischen Zustandsgraphen gesteuert, dessen Zustände die typischen beim Dialog auftretenden Situationen widerspiegeln. Dabei sind für jeden Zustand Übergänge in bestimmte Folgezustände definiert, die an feste Bedingungen geknüpft sind, siehe Abbildung 2. Nach dem Programmstart ist der Zustand *Idle* aktiv und es wird unmittelbar in den Zustand *Ignore* gewechselt. In diesem Zustand werden keine Beobachtungen ausgewertet. Nach einer Zeitspanne von 10 Sekunden wird in den Zustand *Explore* gewechselt, in dem aktiv nach einem Nutzer gesucht wird. Bei Auffinden eines Gesichts wird in den Zustand *Greet* gewechselt, in dem der Nutzer begrüßt wird. Unmittelbar danach wird im Zustand *Explain* die Aufgabe und die Funktionsweise des Roboters erläutert. Nachdem sich der Nutzer durch Betätigen einer Taste auf dem Display angemeldet hat, folgt der Zustand *Dialog*. Entsprechend den hier spezifizierten Wünschen wird in den Zustand *Guide* für eine Lotsenfahrt zu einem bestimmten Produkt oder in den Zustand *Follow* gewechselt, in dem sich der Roboter als Begleiter des Kunden in dessen Nähe aufhält. Nachdem sich der Nutzer abgemeldet hat, erfolgt eine Verabschiedung im Zustand *Dispand* und es wird schließlich wieder in den Zustand *Ignore* gewechselt.

Für jeden Zustand können Ausnahmezustände definiert werden, die dann aktiviert werden, wenn bestimmte Bedingungen, wie z.B. das erfolgreiche Tracken des Nutzers, nicht mehr erfüllt sind. In diesen Ausnahmezuständen kann dann geeignet reagiert werden. Z.B. wird im Zustand *Closer* der Nutzer aufgefordert, etwas näher an den Roboter heranzutreten. Ist die entsprechende Bedingung erfüllt, kann wieder in den Zustand gewechselt werden, in dem die Ausnahme auftrat, ansonsten wird der Kommunikationszyklus ganz abgebrochen (Zustand *Ignore*). Ebenso sind für jeden Zustand bzw. jeden Übergang bestimmte Aktionen definiert, z.B. ein bestimmter Gesichtsausdruck, eine Bewegung des Kopfes, Sprachausgaben oder die Darstellung bestimmter Seiten auf dem Display. Bedin-

gungen für Zustandsübergänge können sein: Timeouts, das Vorhandensein von Hautfarbe bzw. eines Gesichtes oder eine bestimmte Eingabe auf dem Display. Sind die nötigen Bedingungen nicht erfüllt, wird im aktuellen Zustand verweilt. Dabei wird versucht, den Nutzer über zusätzliche Ausgaben zur Erfüllung der Bedingungen zu bewegen.

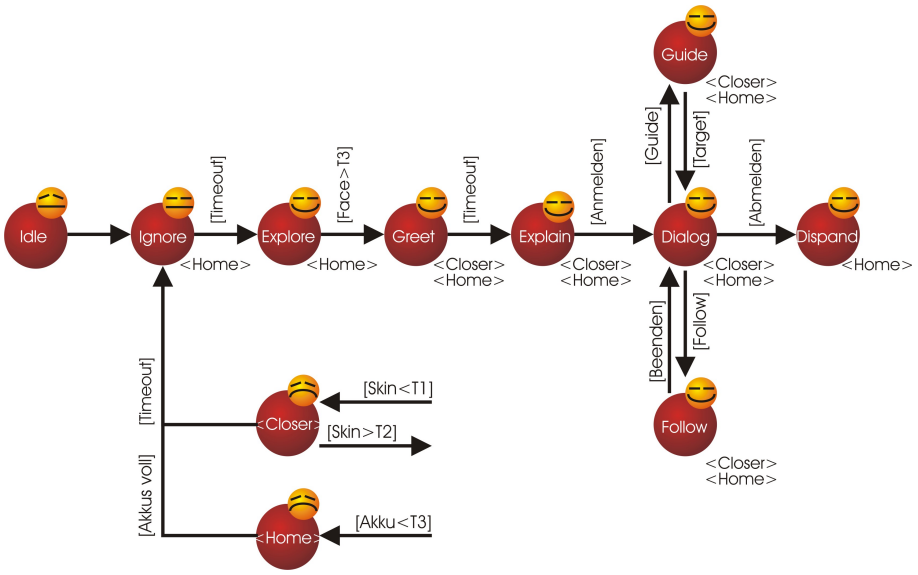

Abbildung 2. Zustandsgraph des Moduls Dialogmanagement / Ablaufsteuerung. Ausnahmezustände sind durch spitze Klammern gekennzeichnet. So wird z.B. die Fahrt zur Akkuladestation über den Zustand Home *repräsentiert, in dem keinerlei Interaktion mit einem Kunden mehr stattfindet.*

3.3 Elementarverhalten

Die einzelnen Zustände des Moduls Dialogmanagement / Ablaufsteuerung werden auf verschiedene vordefinierte „Bewegungsmodi“ abgebildet, da die Interaktion natürlich auch die Navigation des Roboters mit beinhaltet. Diese Bewegungsmodi, die nachfolgend kurz aufgelistet sind, werden über das Blackboard wieder allen Teilmodulen bekannt gemacht. Wie bereits in Abbildung 2 werden Ausnahmezustände durch spitze Klammern gekennzeichnet.

- *Ignore*: zufälliges Umherfahren ohne Auswertung der Nutzerdetektion
- *Explore*: zufälliges Umherfahren und Auswertung der Nutzerdetektion
- *Greet, Explain, Dialog, Disband*: Stehenbleiben und Ausrichtung des Körpers nach dem Nutzer
- *Follow*: Verfolgen der aktuellen Nutzerposition; Hinterherfahren
- *Guide*: Anfahren eines Zielpunktes (Produkt) mit Auswertung der Nutzerdetektion
- *<Closer>*: Stehenbleiben; Aufforderung, wieder näher zu kommen
- *<Home>*: Anfahren eines Zielpunktes (Ladestation) ohne Auswertung der Nutzerdetektion

Aus den Aktivitäten in jedem Elementarverhalten ergibt sich, dass nur die Ergebnisse bestimmter Verhaltensmodule relevant sind. Allerdings müssen trotzdem immer alle Module ihre Ergebnisse kontinuierlich berechnen, da bei einem Umschalten des Zustandes sofort aktuelle Informationen benötigt werden.

3.4 Navigation

Der Gesamtkomplex Navigation umfasst Teilmodule für *Kartenaufbau*, *Selbstlokalisation*, *Hindernisvermeidung*, *Pfadplanung* sowie *Bewegungssteuerung*. Die genannten Teilmodule operieren entweder kontinuierlich oder abhängig von einem der Elementarverhalten (siehe Abschnitt 3.3), die auf dem Blackboard aktiviert sind.

3.4.1 Kartenaufbau

Die sonar-basierte Modellierung der Umgebung des Roboters durch probabilistische Belegtheitsgitterkarten [15] [21] hat sich als sinnvoll und weitgehend praktikabel erwiesen. Vorteilhaft ist vor allem die Bewahrung der metrischen Informationen der Umgebung, was die Attributierung der globalen Umgebungskarte mit Artikelstandorten und Marktbereichen deutlich vereinfacht. Die initiale Karte des Marktes wird vor dem eigentlichen Einsatz des Roboters während einer Teachfahrt mittels Joystick gelernt und als Basisrepräsentation auf dem Blackboard eingetragen. Der Einsatz einer visuell-basierten Odometriekorrektur, die auf der kamerabasierten Auswertung der Fußbodenstruktur der Einsatzumgebung beruht [19], sichert, dass der Odometriefehler klein genug gehalten werden kann, um eine konsistente Karte des gesamten Baumarktes aufzubauen. Während des Kartenaufbaus wird diese fortlaufend mit Merkmalsvektoren, die aus Ansichten der omnidirektionalen Kamera gewonnen werden, attributiert. Diese visuelle Umgebungsrepräsentation bildet die Basis für die spätere Selbstlokalisation des Roboters. Im eigentlichen Einsatz des Roboters kann das Umgebungsmodell dann kontinuierlich adaptiert werden.

3.4.2 Selbstlokalisation

Das Verfahren zur visuell-basierten Selbstlokalisation operiert kontinuierlich und damit unabhängig von den auf dem Blackboard angezeigten aktuellen Elementarverhalten. Es basiert auf dem MCL-Algorithmus, der als robustes und effizientes Werkzeug zur Zustandsschätzung zunehmend Verbreitung findet [8]. Dieses Verfahren wurde für den Einsatz omnidirektionaler Ansichten, die die sensorische Beobachtung des Roboters liefern, angepasst [10]. Mit dem vorliegenden Ansatz ist es möglich, sowohl die Pose des Roboters bei bekanntem Initialzustand zu verfolgen als auch für den Fall, dass die Zustandsinformation aufgrund fehlerhafter Sensorinformation sehr große Unsicherheiten aufweist, eine globale Lokalisation über dem gesamten Zustandsraum innerhalb weniger Bewegungs- und Beobachtungsschritte vorzunehmen.

3.4.3 Pfadplanung und Bewegungssteuerung

Die eigentliche Pfadplanung nutzt die während der Teachfahrt erlernte globale Umgebungsrepräsentation und beruht auf dem bekannten A^*-Algorithmus. Zur Gewährleistung eines ruhigen und glatten Fahrverhaltens bei sicherer Kollisionsvermeidung wurde die Hindernisvermeidung implizit in die Pfadplanung mit integriert, indem eine Abbildung des Hindernisabstandes auf die Weglänge des geplanten Pfades erfolgt. Die Pfadplanung unterscheidet anhand des aktuell gesetzten Elementarverhaltens auf dem Blackboard, ob ein Pfad zu einem benutzerdefinierten Ziel in globalen Weltkoordinaten geplant wird oder ob die aktuelle Position des Nutzers die momentan gültigen (egozentrischen) Zielkoordinaten vorgibt. Die eigentliche Bewegungssteuerung basiert auf dem Einsatz von Vektorfeld-Histogrammen [6].

4 Implementation

Die beschriebene Architektur wurde unter Linux in der Programmiersprache C++ implementiert. Alle in Abbildung 1 dargestellten Teilmodule bilden unabhängige Threads. Die Kommunikation der Teilmodule erfolgt ausschließlich über das Blackboard. Damit wird vermieden, dass durch eine Interprozesskommunikation (wie dies z.B. über explizite Events in der 3T-Architektur erfolgt) die Übersichtlichkeit des Gesamtsystems leidet. Das Blackboard stellt eine einheitliche Schnittstelle für alle darauf zugreifenden Module bereit und sorgt selbst über einen Sensor-Thread dafür, dass alle durch die Roboter-Hardware erfassten Daten aktuell zur Verfügung stehen. Die damit realisierte Abstraktion von der Roboter-Hardware gestattet eine Portierung des Gesamtsystems auf unterschiedliche Plattformen.

Der Dialogsteuerung kommt im Rahmen der Architektur eine Sonderrolle zu: sie definiert anhand der aktuellen Informationen auf dem Blackboard applikationsspezifische Systemzustände und bildet damit eine Kontrollinstanz für alle anderen heterarchisch organisierten Teilmodule.

5 Gesamtverhalten

In diesem Abschnitt soll exemplarisch ein vereinfachter Ablauf der Interaktion zwischen PERSES und einem Nutzer im Baumarkt beschrieben werden. Abb. 3 zeigt dazu einen Interaktionszyklus, der in dieser oder leicht abgewandelter Form im Baumarkt zwischen Roboter und seinem menschlichen Interaktionspartner ablaufen kann.

Im Zustand *Explore* sucht PERSES zunächst nach einem Nutzer. Die Nutzerdetektion läuft kontinuierlich und sucht nach hautfarbenen Regionen in der näheren Umgebung von PERSES. Sobald ein Kunde in die Nähe kommt, stoppt PERSES seine Fahrt und richtet sich mit Körper und Kopf auf den Kunden aus. Falls der Gesichtsdetektor ein Gesicht detektiert, wechselt PERSES in den Zustand *Greet*: „Herzlich willkommen im toom Baumarkt!“ Da die Person weiterhin in der Nähe bleibt und getrackt werden kann, wechselt PERSES in den Zustand *Explain*, stellt sich selbst vor und erklärt seine Funktionsweise. Sobald sich der Nutzer per Touch-Interface anmeldet, werden die Funktionen von PERSES näher erklärt: „Wenn Sie ein bestimmtes Produkt suchen, wählen Sie bitte *Produktsuche*. Wenn Sie spezielle Fragen haben, wählen Sie bitte die *Verbindung zum Fachberater*. Wenn Sie lediglich meine Begleitung wünschen, wählen Sie bitte *Folgefahrt*.“ Der Kunde sucht nun beispielsweise ein bestimmtes Produkt in einem baumartig strukturierten Menü. Nachdem er dieses gefunden hat, zeigt PERSES eine genauere Beschreibung des Produktes und fragt den Kunden, ob er ihn zum entsprechenden Standort im Markt bringen soll. Falls dies der Fall ist, wechselt PERSES in den Zustand *Guide*, in dem während der Lotsenfahrt der Kunde kontinuierlich getrackt wird. Wenn der Nutzer stehen bleibt oder aus dem Sichtbereich verschwindet, hält PERSES an und fordert ihn auf, doch wieder näher zu kommen. Falls ein Hindernis den geplanten Pfad versperrt, wird es in der aktuellen Karte eingetragen und ein neuer Pfad geplant. Beim Produkt angekommen, stoppt PERSES, wendet seinen Kopf in Richtung des Produktes und vermittelt verbal den entsprechenden Artikelstandort: „Sie finden das gewünschte Produkt hier oben im Regal.“

6 Fazit

Es wurde eine Steuerarchitektur vorgestellt, die auf einem interaktiven mobilen Serviceroboter implementiert wurde. Besondere Berücksichtigung fanden dabei die Aspekte Transparenz, Modularität und leichte Erweiterbarkeit, da die Architektur im Fortgang der experimentellen Untersuchungen kontinuierlichen Veränderungen unterliegt. Unsere bisherigen Erfahrungen zeigen, dass die vorgestellte Architektur diesen Anforderungen gerecht wird. Weiterhin soll hervorgehoben werden, dass mit dem

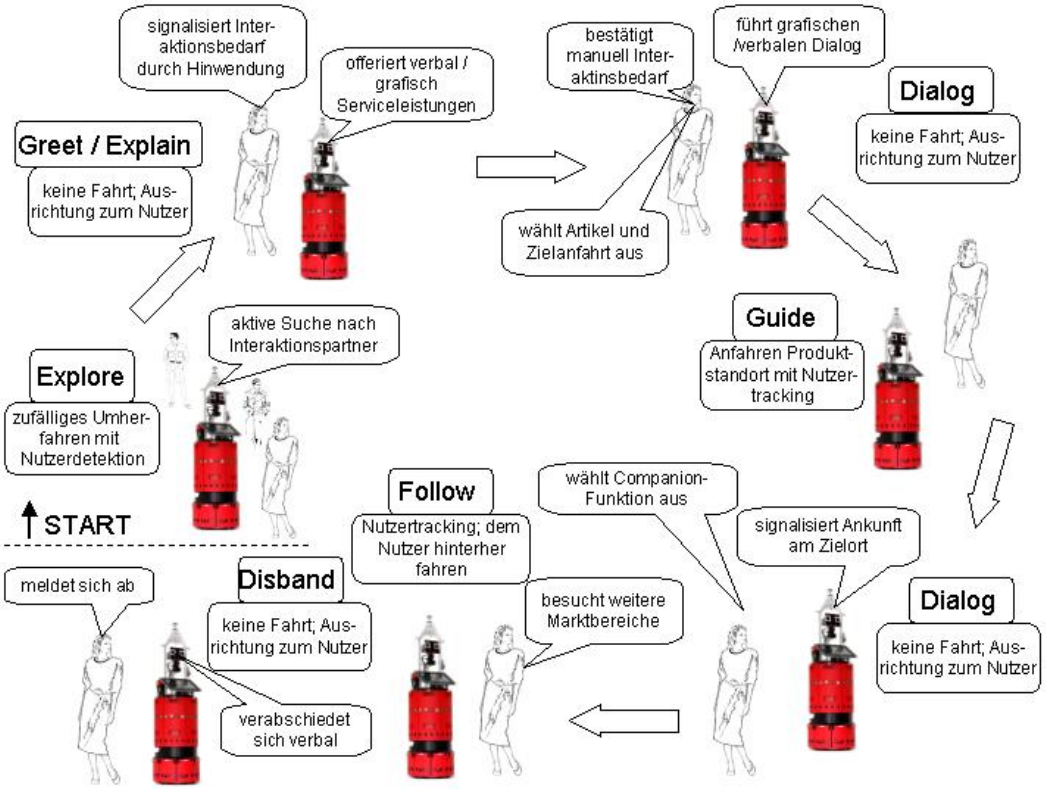

Abbildung 3. Darstellung eines Interaktionszyklus, wie er typischerweise zwischen dem Roboter und seinem Nutzer im Baumarkt stattfindet. Zustände (Situationen), die sich auf das Dialogmanagement beziehen, wurden in ovalen Boxen dargestellt, unmittelbar darunter sind die dazu korrespondierenden Bewegungsmodi angegeben. Die „Sprechblasen“ dienen der Erläuterung der verschiedenen Abschnitte des Interaktionszyklusses.

vorgeschlagenen Architekturkonzept eine kompakte Modellierung mit modernen Methoden der Softwaretechnik (UML-basierte Analyse und Entwurf) umgesetzt werden konnte. Dies kommt vor allem den Mitarbeitern und Studenten zugute, die sich neu in das bestehende System einarbeiten müssen.

Die beschriebene Architektur hat sich im praktischen Einsatz im Rahmen von Projektdemonstrationen bereits als sehr praktikabel erwiesen. So existiert mittlerweile eine weitere Implementierung der Architektur auf dem Robotersystem HOROS [14].

Beim Entwurf einer Steuerarchitektur muss letztlich stets eine Balance zwischen Effizienz, Transparenz und Modularität gefunden werden, da letztere zwangsläufig einen Overhead in der Implementierung mit sich bringen. Mit zunehmender Leistungsfähigkeit der eingesetzten Rechentechnik treten Probleme bezüglich dieses Overhead jedoch mehr und mehr in den Hintergrund, so dass die bestehenden zeitlichen Anforderungen erfüllt werden können.

Als konzeptionell wesentlich erachten wir, dass (i) die Verwendung eines Blackboards eine gute Möglichkeit der Hardware-Abstraktion bietet, (ii) die strenge Entkopplung der Teilmodule durch die ausschließliche Kommunikation über das Blackboard eine einfach zu beherrschende Struktur entsteht und (iii) durch die flache, nicht-hierarchische Organisation der Teilmodule die Architektur sehr einfach erweitert und Teilmodule modifiziert werden können.

Ausdrücklich soll betont werden, dass mit diesem Beitrag weder der Anspruch einer universellen noch vollständig originären Architektur erhoben wird. Vielmehr geht es darum, unsere Erfahrungen beim Architekturentwurf zu verdeutlichen und die Frage der Steuerarchitektur im allgemeinen mehr in den Mittelpunkt des Interesses zu rücken.

Literatur

[1] Althaus, P. and Christensen, H.I. Smooth task switching through behavior competition. *Robotics and Autonomous Systems*, 44:241–249, 2003.

[2] Arkin, R.C. *Behavior-Based Robotics*. MIT Press, 1998.

[3] Bischoff, R. and Graefe, V. Integrating Vision, Touch and Natural Language in the Control of a Situation-Oriented Behavior-Based Humanoid Robot. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume II, pages 999–1004, 1999.

[4] Bonasso, R.P., Firby, R.J., Gat, E., Kortenkamp, D., Miller, D., and Slack, M. Experiences with an Architecture for Intelligent, Reactive Agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 18(1), 1997.

[5] Bonasso, R.P., Kortenkamp, D., and Whitney, T. Using a Robot Control Architecture to Automate Space Shuttle Operations. In *9th Conference on Innovative Applications IAAI'97*, 1997.

[6] Borenstein, J. and Koren, Y. The Vector Field Histogram – Fast Obstacle-Avoidance for Mobile Robots. *IEEE Journal of Robotics and Automation*, 7(3):278–288, 1991.

[7] Brooks, R.A. A robust layered control system for a mobile robot. *Journal of Robotics and Automation*, 2, 1986.

[8] Fox, D., Delleart, F., Burgard, W., and Thrun, S. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. In *Proceedings 16th National Conference on Artificial Intelligence (AAAI-99)*, 1999.

[9] Fujita, M. and Kageyama, K. An open architecture for robot entertainment. In *Proccedings of the First International Conference on Autonomous Agents*, pages 435–442, 1997.

[10] Gross, H.-M., Koenig, A., Schroeter, C., and Boehme, H.-J. Omnivision-based Probabilistic Self-localization for a Mobile Shopping Assistant Continued. In *IEEE/RSJ Intern. Conference on Intelligent Robots and Systems (IROS 2003), Las Vegas*, 2003.

[11] Isard, M. and Blake, A. CONDENSATION – conditional density propagation for visual tracking. *International Journal on Computer Vision*, 29(1):5–28, 1998.

[12] Kortenkamp, D., Bonasso, R.P., and Murphy, R., editors. *Artificial Intelligence and Mobile Robots*. AAAI Press / MIT Press, 1998.

[13] Marr, D. *Vision*. Freeman, San Francisco, 1982.

[14] Martin, C., Böhme, H.-J., and Gross, H.-M. Conception and Realization of a Multi-Sensory Interactive Mobile Office Guide. In *IEEE International Conference on Systems, Man, and Cybernetics*, 2004. to appear.

[15] Moravec, H.P. and Elfes, A. High resolution maps from wide angle sonar. In *International Conference on Robotics and Automation (ICRA)*, pages 116–121, 1985.

[16] Murphy, R.R. *Introduction to AI Robotics*. MIT Press, 2000.

[17] Orebäck, A. and Christensen, H.I. Evaluation of Architectures for Mobile Robotics. *Autonomous Robots*, 14:33–49, 2003.

[18] Roy, N., Montemerlo, M., and Thrun, S. Perspectives on standardization in mobile robot programming. In *IROS 2003*, 2003.

[19] Schroeter, C., Boehme, H.-J., and Gross, H.-M. Extraction of orientation from floor structure for odometry correction in mobile robotics. In *25th Pattern Recognition Symposium DAGM 2003*. Springer Verlag, 2003.

[20] Steinhage, A. and Bergener, T. Dynamical Systems for the Behavioral Organization of an Anthropomorphic Mobile Robot. In *From Animals to Animats 5: Proceedings of the Fifth International Conference on Simulation of Adaptive Behavior (SAB 98)*. M.I.T. Press, 1998.

[21] Thrun, S. Learning Maps for Indoor Mobile Robot Navigation. *Artificial Intelligence*, 99(1):21–71, 1999.

[22] Thrun, S. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–109, 2000.

[23] Viola, P. and Jones, M. Robust Real-time Object Recognition. In *Second International Workshop on Statistical and Computational Theories of Vision*, 2001.

[24] Volpe, R., Nesnas, I., Estlin, T., Mutz, D., Petras, R., and Das, H. The CLARAty Architecture for Robotic Autonomy. In *Proceedings of the 2001 IEEE Aerospace Conference*, 2001.

[25] Wilhelm, T., Böhme, H.-J., and Gross, H.-M. Sensorfusion for Visual and Sonar based People Tracking on a mobile Servicerobot. In *International Workshop on Dynamic Perception*, pages 315–320. IOS press, infix, 2002.