MAIKE: Mobile Assistenzsysteme für Intelligente Kooperierende Räume und Ensembles

Markus Berg und Nils Weber
Hochschule Wismar
Fakultät für Ingenieurwissenschaften
{markus.berg, nils.weber}@hs-wismar.de

Gernot Ruscher und Sebastian Bader
Universität Rostock
Institut für Informatik
{gernot.ruscher, sebastian.bader}@uni-rostock.de

***Zusammenfassung*—Die Bedienung komplexer multimedialer Räume wird zunehmend komplizierter und erfordert für eine optimale Nutzung geschultes Fachpersonal. Im Rahmen des Projektes MAIKE wird eine Architektur entworfen und prototypisch umgesetzt, die eine intelligente multimodale Interaktion mit heterogenen Geräte-Ensembles ermöglicht. Hierdurch kann der Nutzer Zeit und Konzentration auf seine eigentliche Aufgabe verwenden, ohne durch komplizierte Bedienvorgänge belastet zu werden. In diesem Artikel wird das Problem diskutiert und die entwickelte Systemarchitektur vorgestellt.**

I. EINLEITUNG

Intelligente Umgebungen (*Smart Homes* bzw. *Smart Offices*) sind mit Sensorik und Aktorik instrumentierte Räumlichkeiten, deren Aufgabe es ist, die Nutzer bei ihren Aktivitäten zu unterstützen und somit die Bewältigung von wiederkehrenden Aufgaben zu erleichtern. In diesem Zusammenhang rücken sowohl räumlich verteilte Räume (vernetzte Konferenz- und Besprechungsräume) als auch lokal vernetzte Ensembles (intelligente Häuser) in den Fokus der Betrachtung.

Eine zentrale Herausforderung im Bereich solcher Umgebungen ist die intelligente Nutzung der Sensorik für die automatisierte Steuerung: Aktuell verfügbare Systeme benötigen entweder die explizite menschliche Interaktion oder bieten nur einfachste (im Vorfeld festgelegte) Koppelungen von Sensorwerten und auszuführenden Aktionen. Eine intelligente Automatisierung kann damit nicht erreicht werden.

Gleichzeitig stellt die Notwendigkeit der direkten Interaktion erhebliche Anforderungen an den Nutzer: Die Bedienung von Mediensteuerungspulten in reich instrumentierten Funktionsräumen ist meist nicht intuitiv. Tatsächlich werden zum Teil technisch mögliche Funktionen nicht realisiert, da die Bedienung zu kompliziert wäre. Demgegenüber wird vorhandene Funktionalität nicht genutzt, wenn dem Anwender nicht klar ist, wie sie aktiviert werden kann oder weil in der aktuellen Situation die Aktivierung zu aufwändig ist bzw. die Nutzer bei ihrer eigentlichen Aktivität stört. Die Autoren unterscheiden hierbei zwischen expliziter und impliziter Interaktion: Auszuführende Aktionen können dem System einerseits per Sprache oder grafischer Oberfläche mitgeteilt werden, oder andererseits auf Basis von Sensordaten in Verbindung mit Nutzerprofilen und intelligenten Inferenzsystemen automatisch ermittelt werden.

Ein besonderes Merkmal intelligenter Umgebungen ist im Sinne von *Ubiquitous Computing* nach WEISER [16] die Omnipräsenz, verbunden mit gleichzeitiger Unaufdringlichkeit der physikalisch erforderlichen Geräte. Das heißt, die Funktionalität muss immer zur Verfügung stehen, ohne dass viele technische Geräte (die man im schlimmsten Fall am Körper tragen muss) sichtbar sind. Somit rückt die Technik in den Hintergund und die Funktionalität in den Vordergrund. Die Symbiose von Raum und Funktionalität trägt dabei zu einem neuen intuitiven, multimodalen Bedienerlebnis bei, das seine Stärken insbesondere in dynamischen, multifunktionalen Ensembles ausspielt. Hierzu tragen adaptive, dynamische, situationsabhängige Systeme bei, die sich am Kenntnisstand des Nutzers, bekannten Verhaltensweisen und wiederkehrenden Aktionsfolgen orientieren.

So können (je nach Uhrzeit) bestimmte Vorlesungen gestartet bzw. an der letzten Stelle fortgesetzt werden, das Einschalten des Beamers führt zur Abdunkelung der Fenster und dem Herunterfahren der Leinwand und je nach aktueller Raumnutzungssituation (Konferenz, Vorlesung, Besprechung) wird bei Uneindeutigkeiten zwischen „best-match“ Systemreaktion und Nachfrage via GUI bzw. Sprache entschieden. Dem Nutzer soll dabei die kognitive Belastung abgenommen werden, in welcher Reihenfolge welche Geräte auf welche Weise bedient werden müssen. Er äußert sich *in seiner Sprache* und das System ermittelt automatisch die benötigten Zwischenschritte. Die Äußerung kann dabei auch implizit erfolgen: Das System schlägt proaktiv Aktionen vor oder führt diese direkt aus.

Ziel des Projektes MAIKE ist die Untersuchung und Entwicklung IT-basierter Dienstleistungen und Komponenten für *Mobile Assistenzsysteme in intelligenten, kooperierenden Räumen und Ensembles*. Die hierzu benötigten Technologien sind Intentionserkennung und benutzergerechte Interaktion (im Dialog per Sprache und GUI), intelligente Methoden für die Kooperation heterogener Geräte-Ensembles, verteiltes Content Management sowie Basisdienste für die drahtlose Kommunikation. All diese Aspekte werden in einer Architektur zusammengefasst, die die genannten Herausforderungen der Verteiltheit, Intuitivität und Multimodalität adressiert.

Somit werden gemeinsam von den Industrie- und Hochschulpartnern innovative Produkte im Segment der intelligenten Räume – z.B. Arbeitsumgebungen und Konferenzräume – konzipiert, entwickelt und prototypisch realisiert, die zu einer erheblichen Entlastung des Benutzers, Produktivitätssteigerungen und einem spürbaren Komfortgewinn führen.

II. HERAUSFORDERUNGEN

Die Integration interaktiver Steuerungen in intelligente Umgebungen stellt besonders hinsichtlich der Akzeptanz und Benutzerfreundlichkeit im Alltag eine besondere Herausforderung dar. Der Vielfalt der Aktionsmöglichkeiten solcher Ensembles steht das Ziel einer intuitiven Bedienung gegenüber. Häufig wird das Funktionspotential solcher Räume nicht vollständig genutzt, weil komplexe grafische Menüs oder komplizierte, schwer zu erfassende Abläufe die Bedienung erschweren. Zusätzliches technisch versiertes Personal ist in vielen Fällen aus Kostengründen nicht verfügbar und meist auch im Rahmen von Alltagssituationen, die die Nutzer eigenständig lösen möchten, nicht erwünscht. Zur Lösung dieser Probleme müssen verschiedene Faktoren betrachtet werden:

- *Natürlichsprachlichkeit:* Abkehr von festen Kommandos zugunsten freier Sprache.
- *Dialogfähigkeit:* Dezente sprachliche oder multimodale Interaktion mit dem Benutzer als Hilfestellung.
- *Ubiquität:* Ständige Ansprechbarkeit/Präsenz des Systems.
- *Sensorik:* Präzision und Aussagekraft der Sensordaten.

Auch wenn es mittlerweile sehr solide Sprachsteuerungen gibt, werden diese häufig nicht eingesetzt, weil die Sprachbefehle einserseits zu restriktiv oder im anderen Extrem zu umfangreich sind. Meist wird daher ausschließlich eine sprachunterstützte Navigation durch grafische Menüs unterstützt. Die Sprachausgabe beschränkt sich in der Regel auf Fehlermeldungen oder es wird gänzlich auf sie verzichtet und die Systembestätigung zeigt sich ausschließlich in der Ausführung des gewünschten Befehls. Muss dann noch ein besonderer Modus gestartet werden, um mit dem System per Sprache oder Gestik interagieren zu können, werden viele Nutzer auf die Vorteile einer solchen automatischen Steuerungsanlage verzichten und die klassischen Wege der Bedienung wählen.

Um dem entgegenzuwirken, wird die Integration einer allzeit präsenten natürlichsprachlichen ASR (Automatic Speech Recognition) im Zusammenspiel mit einer komplexen multimodalen Dialogkomponente angestrebt. *Natürlichsprachlichkeit* bedeutet in diesem Zusammenhang die Abkehr von Systeminteraktion im Kommandostil hin zu einer freien („natürlicheren“) Sprache, welche ohne feste Befehlsstrukturen auskommt. Das Stichwort *Multimodalität* bezieht sich auf die dynamische Integration der einzelnen Interaktionsschnittstellen. Die Informationen der Raumsensorik, Spracheingaben und die situationsoptimierte Wahl grafischer oder akustischer Kanäle bilden einen hochgradig verflochtenen Kontext, der die Reaktionen des kooperativen Ensembles sowohl implizit als auch explizit beeinflusst. Im Folgenden werden einige Beispiele für die Steuerungmöglichkeiten und das Zusammenspiel mit den anderen Modulen detailliert erläutert.

Die Interaktion in Form eines solchen multimodalen Dialoges und die Entwicklung einer hierfür geeigneten Architektur sind ambitionierte Projektziele. Der Dialog soll folglich nicht nur auf Sprache basieren. Je nach Situation wird automatisch der geeignetste Kanal zur Interaktion mit dem Nutzer gewählt. Diese Entscheidungen beruhen ebenso auf Informationen über den aktuellen Status des Raumes als auch auf dem Status der Akteure. Um ein übermäßiges Nachfrageaufkommen zu vermeiden, ist es hierbei von entscheidender Bedeutung, fehlende Angaben durch intelligente Einbeziehung von Kontextinformationen zu ergänzen. Ein komplexes Zusammenwirken der einzelnen Automatisierungsmodule ist aus diesem Grunde unverzichtbar. Die zur Auflösung ambivalenter Aktionen benötigten Informationen werden in einer zentralen Wissensbasis gespeichert. Gleichzeitig sind für jedes Modul Zuständigkeiten definiert, die bei der optimalen Umsetzung von Nutzeranfragen zum Tragen kommen.

Die Genauigkeit der *Sensorik* und deren ständige Präsenz sind weitere Faktoren für eine im Alltag einsetzbare multimodale Steuerungsanlage. Um sinnvolle Rückschlüsse aus Sensordaten ziehen zu können, müssen deren Parameter ständig in die automatisch getroffenen Entscheidungen einbezogen werden. Die Sensorik wird eingesetzt, um beispielsweise Informationen über den Aufenthaltsort oder die Identität von Personen zu erhalten oder um allgemeine Parameter wie Temperatur oder Lichtverhältnisse auszulesen.

Eine weitere Herausforderung besteht in der zentralen Verwaltung des aktuellen Systemzustandes auch in räumlich getrennten, verteilten Architekturen. Sämtliche Parameter mit Einfluss auf das Verhalten müssen ohne Verzögerung in der Datenbasis vorgehalten und laufend aktualisiert werden, ohne dabei Inkonsistenzen in offenen Fragestellungen zu provozieren. Hierzu wird in Abschnitt V eine Architektur vorgestellt.

Zur Veranschaulichung der Zielstellungen des Projektes werden im folgenden Abschnitt beispielhaft Abläufe in automatisierten Konferenzräumen beschrieben. Dabei wird von einem häufig wechselnden Personal ausgegangen, bei dem kein fundiertes Vorwissen zur Bedienung derartiger Räume angenommen werden kann. Der Raum beinhaltet mehrere Projektoren, die auf verschiedene Wände gerichtet sind und somit nur von bestimmten Plätzen eine optimale Sicht gewährleistet werden kann. Die Strategien zur dynamischen Verteilung der Bildschirminhalte variieren in den verschiedenen Szenarien. Weiterhin verfügt der Raum über Positionssensordaten, Helligkeits- und Temperaturinformationen, Mikrofonierung zur Spracherkennung und mobile Geräte zur Fernsteuerung.

Besprechung/Präsentation: Ausgehend von der Sitzplatzbelegung im Raum werden verschiedene System-Modi unterschieden. Deutet die Situation auf eine Präsentation hin, werden die Leinwände heruntergefahren, die Projektoren gestartet und die verfügbaren Medien (z.B. eingesteckter USB-Stick) nach Präsentationen durchsucht, welche dann zur Anzeige angeboten werden. Diese Schritte sollten bei Bedarf in einem dezenten Dialog verifiziert werden, damit mögliche Fehlinterpretationen nicht zu unbeabsichtigtem Verhalten des Raumes führen. Für den Präsentationsstart könnte sich der Nutzer per Sprache an das System wenden oder aber eine eingeblendete Oberfläche zur Steuerung nutzen. Hier sind auch multimodale Menüs zur Auswahl der verfügbaren Dateien

vorgesehen. Bei dynamischen Inhalten muss der Wortschatz zur Laufzeit entsprechend erweitert und die der Erkennung zu Grunde liegenden Grammatiken angepasst werden. Auch die im Raum steuerbaren Geräte sollen flexibel verwaltet werden. So sollen beispielsweise Notebooks und deren Dienste sowie andere kompatible Geräte automatisch in die Funktionalität des Raumes integriert werden.

Weiterhin ist es sinnvoll, in Abhängigkeit äußerer Einflüsse, einzelne Komponenten vorübergehend zu deaktivieren. So wäre bei einem sehr hohen Lärmpegel im Raum die Sprachsteuerung sehr fehleranfällig, während eine grafische Abfrage nur hilfreich ist, wenn der Vortragende sich auch in Sichtweite des Displays befindet. Das System sollte anhand der verfügbaren Parameter entscheiden, welche Kanäle sich für die Nutzerinteraktion in der aktuellen Situation eignen.

Ist das Auditorium kreisförmig um den Besprechungstisch verteilt, ist die Sicht auf die Hauptleinwand meist nur eingeschränkt vorhanden. Um eine optimale Sicht von jedem Sitzplatz aus zu gewährleisten, muss je nach Sitzordnung eine geeignete Strategie zur Anzeige der Präsentationen gefunden werden. Im einfachsten Fall kann der Bildschirminhalt dupliziert werden. Wenn jedoch auch Nebeninformationen angezeigt werden sollen, müssen komplexere Strategien gefunden werden, um die Sicht auf die verschiedenen Inhalte zu ermöglichen.

Videokonferenz: Die Durchführung von Video- oder Telefonkonferenzen erfordert eigene Strategien zur Verteilung der Bildschirminhalte. Hier ist es von entscheidender Bedeutung, dass die Hauptakteure der Konversation jeweils relevante Inhalte und aussagekräftige Perspektiven des Gesprächspartners auf der Gegenstelle eingeblendet bekommen.

Um auch Inhalte und Funktionen des entfernten Raumes über den zentralen Informations- und Funktionspool zugänglich machen zu können, muss das Wissen über die Systemkomponenten, deren Status, der gesamte Systemwortschatz, Nutzerprofile mit Berechtigungen und sämtliche Resourcen gemeinsam verwaltet werden.

III. Anforderungen

Die einzelnen Interaktionskomponenten sind nicht in jeder Situation in der Lage, autark dedizierte Entscheidungen zu treffen. Für den Fall, dass Befehle unscharf formuliert wurden, sich Parameter im laufenden Betrieb ändern oder aus anderen Gründen Anpassungen des Raumzustandes notwendig sind, muss eine Kommunikationsebene für den Austausch zwischen den Komponenten eingeführt werden, um fehlende Informationen in die Entscheidung einzubeziehen. Werden die Interaktionskomponenten Sprache, GUI und Intentionserkennung hinsichtlich ihrer Kompetenzen analysiert, ergibt sich die in Tabelle I dargestellte Übersicht über die Verteilung der Zuständigkeiten im System.

Die Zuständigkeiten der Module sind als persistentes Weltwissen im zentralen Datenspeicher hinterlegt. Dadurch werden offene Fragen zum aktuellen Systemzustand und mögliche Lösungswege transparent und adressierbar. Situationsabhängig

		GUI	Sprache	Intention
Kommandosteuerung	implizit	○	○	○
	explizit	○	○	
Dialogfähigkeit		○	○	
Sensorik				○

Tabelle I
Aufgaben der Module

können Informationen zur Ergänzung unklarer Interaktionsschritte gezielt abgefragt werden. Die GUI und die Sprachkomponente stellen hierbei einen interaktiven Zugang zum System dar über den eine explizite Ausführung von Kommandos möglich ist. Diese beiden Module stellen den grafischen und akustischen Kanal und die Logik des Systemdialoges zur Verfügung. Ein weiterer Zugang zum System wird über die proaktive Assistenz der Intentionserkennung ermöglicht. Hierbei handelt es sich jedoch eher um eine implizite Bindung da aus dem Nutzerverhalten im Hintergrund bestimmte Reaktionen des Raumes abgeleitet werden.

Im Rahmen des MAIKE-Projektes konnten zusammenfassend die folgenden wesentlichen Anforderungen definiert werden, die an eine Intelligente Umgebung gestellt werden und sich direkt aus den zuvor dargestellen Herausforderungen in diesem Arbeits- und Forschungsthema ergeben:

Um Geräte, die neu in einen Intelligenten Besprechungsraum mitgebracht werden, „on-the-fly“ in die Rauminfrastruktur integrieren zu können und somit eine dynamische Erweiterbarkeit sicherzustellen, muss eine möglichst lose Kopplung der einzelnen Komponenten angestrebt werden. Die Infrastruktur wird als dynamisches Ensemble verstanden, das sich jederzeit ändern kann, indem neue Geräte hinzukommen oder verschwinden. Damit neue Geräte den aktuellen Weltzustand schnellstmöglich erfassen können, ist es notwendig, diesen persistent an einer zentralen Stelle für alle Komponenten erreichbar zu machen, da jede Infrastrukturänderung anderenfalls eine enorme Netzwerklast provozieren würde.

Eine natürliche intuitiv zu verwendende Mensch-System-Schnittstelle wird als essentiell für die Akzeptanz einer Intelligenten Umgebung angesehen. Da menschliche Kommunikation stets multimodal ist, wird an entsprechende Systeme die Anforderung gestellt, ebenso multimodale Interaktionsstrukturen verarbeiten zu können. Die Interaktionskomponenten, die den einzelnen Modalitäten entsprechen, müssen demzufolge in die Lage versetzt werden, aus unklaren Situationen oder Nutzeräußerungen resultierende Ambiguitäten aufzulösen. Dazu wird ein möglichst generischer Mechanismus benötigt, um der Anforderung der dynamischen Erweiterbarkeit des Systems nicht entgegenzuwirken.

Aus der Multimodalität zwischenmenschlicher Interaktion ergibt sich ebenso direkt der dringende Bedarf nach impliziten Steuerungsmöglichkeiten. Darunter verstehen die Autoren eine proaktive Assistenz durch das System, die mögliche Ziele der Nutzer aus seinem Verhalten ableitet und daraus resultierende Aktionen auslöst, ohne dass ein explizites Kommando formu-

liert werden muss. Dies wird in MAIKE durch die Integration einer speziellen Komponente zur Intentionserkennung realisiert, die durch permanente Auswertung des aktuellen Welt- bzw. Nutzerzustandes unter Zuhilfenahme unterschiedlicher Sensordaten die jeweiligen Wahrscheinlichkeiten potentieller Nutzerziele inferiert.

Da die einzelnen Komponenten, wie intelligente Spracherkennung und Intentionserkennung, aktuelle Forschungsfragen betreffen, ist davon auszugehen, dass ihr Entwicklungsstatus nicht etwa als abgeschlossen, sondern allenfalls als prototypisch betrachtet werden muss. Dieser Umstand führt zu der Anforderung an eine Infrastruktur, die es den Forschern und Entwicklern ermöglicht, den Datenverkehr zwischen den Komponenten so nah wie möglich (und nötig!) am Netzwerkprotokoll zu analysieren. Daher wird ein einfaches, jedoch nicht weniger mächtiges Serialisierungs- und Deserialisierungskonzept benötigt, das nicht nur von Maschinen, sondern auch von Menschen verstanden wird, ohne komplexe Toolkits verwenden oder Experte in XML oder UDP sein zu müssen.

IV. VERWANDTE ARBEITEN

Natürliche Mensch-Computer-Schnittstellen sind, wie bereits beschrieben, gerade im Bereich der Intelligenten Umgebungen zwingend erforderlich, um Bedienbarkeit sicherzustellen oder überhaupt erst zu ermöglichen. Sie sind daher in diversen nationalen und internationalen Forschungsprojekten untersucht worden. Im Folgenden soll eine Auswahl an Projekten aus diesem Umfeld vorgestellt werden. Dabei steht die Frage im Mittelpunkt, inwieweit jeweils die zuvor beschriebenen Anforderungen bereits berücksichtigt wurden.

Den Anwendungsbereich von EMBASSI [14] bildeten Infrastrukturen zur technischen Unterstützung des nicht-beruflichen Alltagslebens. Entwickelt wurden neue Konzepte für intelligente Bedienassistenzsysteme. Dabei wurde die Vision einer nutzerzentrierten Steuerungsphilosophie verfolgt: Es sollte nicht mehr jedes einzelne Gerät eine eigene Nutzerschnittstelle haben, die die komplexe Fülle aller technisch möglichen Funktionen auf seine eigene Weise realisiert, sondern ein komplexes Geräte*ensemble* sollte ein *gemeinsames* Bedienkonzept anbieten. Die folgenden drei Anwendungsszenarien standen dabei im Fokus der Forschungsarbeiten:

- *Infotainmentsysteme im Privathaushalt*: Musikanlage, Telefon, Videorekorder, ...
- *Assistenz im Kraftfahrzeug*: Autoradio, Abstandsmessung, Fahreraufmerksamkeitskontrolle, Reaktion auf Störungsmeldungen, ...
- *Öffentliche Terminalsysteme*: Bank-, Fahrkartenautomaten, ...

Neben zielorientierter Interaktion des Nutzers mit dem System stand der Übergang von unimodalen, menübasierten Dialogen zu polymodalen Konversations- und Dialogstrukturen im Mittelpunkt, mit dem Ziel einer natürlicheren Kommunikation mit dem technischen System. Die Modalitäten der Interaktionskanäle Sprache, Gestik und Grafische Benutzerschnittstelle sowie Fernbedienung und Zeigestab sollten nahtlos miteinander verschmelzen, um gemeinsam eine natürlichere Interaktion mit dem Nutzer zu ermöglichen. Für die genannten Anwendungsbereiche wurden entsprechende Prototypen umgesetzt.

Die Anzahl der Benutzer des Systems, die simultan berücksichtigt werden konnten bzw. sollten, war durchgängig auf eine Person beschränkt. Das EMBASSI-Framework kann als eine Referenztechnologie im Bereich der mobilen Assistenz angesehen werden. Jedoch sieht sie eine kanalbasierte Kommunikation der beteiligten Komponenten vor, was Anforderung nach einem persistenten Weltzustand entgegensteht.

Das SMARTKOM-Projekt [15] versuchte ebenso, die Vorteile sprachlich dialogischer Kommunikation mit den Vorteilen grafischer Bedienoberflächen und gestischen Ausdrucks zu einer intuitiveren Mensch-Technik-Interaktion verschmelzen. Im Gegensatz zu EMBASSI kam zusätzlich eine Komponente zur Erkennung mimischer Elemente zum Einsatz, mit dem Ziel, die Gefühlswelt des Nutzers in den Dialog einzubeziehen. Dadurch sollte beispielsweise die Erkennung einer Umkehrung des Sinngehalts im Falle ironischer Nutzeräußerungen ermöglicht werden. Wie in EMBASSI wurden als Anwendungsszenarien die drei Bereiche Automation der Wohn- bzw. Arbeitsumgebung (*SmartKom Home/Office*), Kfz-Fahrerassistenz (*SmartKom Mobil*) und das Gebiet der öffentlichen Terminalsysteme (*SmartKom Public*) unterschieden. Das Ergebnis sollte auch hier eine höherwertige Benutzerschnittstelle sein, die die menschlichen Sinne in einem weitaus größeren Umfang als bisher berücksichtigt. Die Benutzeranzahl war in jedem der untersuchten Szenarien auf jeweils genau eine Person je Zeiteinheit beschränkt. Implizite Interaktionsmöglichkeiten wurden hier noch nicht berücksichtigt.

Das INHAUS-ZENTRUM [2] in Duisburg soll die Notwendigkeit der Erforschung neuer Technologien im Bereich der Gebäudeautomatisierung zeigen. Diverse am Markt erhältliche Bussysteme werden hier flächendeckend verbaut und eine nahtlose Kommunikation zwischen diesen unterschiedlichen Systemen angestrebt. Im Mittelpunkt der Untersuchungen stehen hier vor allem die Erhöhung der Sicherheit, der sparsame Umgang mit Energieresourcen, die Unterstützung von Senioren bei der Bewältigung ihres Alltags und die Möglichkeit der einfachen Bedienung vorhandener Technik im Wohnbereich.

Dieses Projekt zielt explizit darauf ab, mehrere Nutzer – alle Bewohner des Hauses – simultan bei der Bewältigung ihrer aktuellen Aufgaben zu unterstützen. Im Gegenzug wurde für die Interaktion der Nutzer mit dem System auf eine Reihe von Aspekten verzichtet. Im Vordergrund stand somit nicht die Realisierung einer natürlichen intuitiven Kommunikationsschnittstelle. Eine Verschmelzung mehrerer Kommunikationskanäle wurde nicht angestrebt. Somit musste sich das Projekt auf explizit unimodale Interaktion bzw. auf einfachste Sensor-Aktor-Kopplungen beschränken.

Als wichtige internationale Projekte im betreffenden Umfeld seien an dieser Stelle SEAMLESS HOME SERVICES [11], MAVHOME [5] und INTELLIGENT CLASSROOM [9] lediglich genannt. Eine allgemeine Einführung in die Thematik der Intelligenten Umgebungen, eingehend auf die besonderen Möglichkeiten und Herausforderungen in diesem Bereich, bieten beispielsweise COOK und DAS in [4]. Eine hilfreiche Über-

sicht über realisierte Projekte kann in [6] (COOPERSTOCK et al.) gefunden werden. Auf technische Infrastrukturen und Softwareframeworks für Intelligente Umgebungen im Besonderen wird durch ENDRES et al. in [7] eingegangen.

Im Überblick über die existierenden Ansätze wird deutlich, dass jeweils entweder vollständig multimodale Systeme realisiert wurden, die zwar eine Reihe von Kommunikationskanälen zur Verfügung stellen und durch eine Verschmelzung dieser Kanäle bereits ansatzweise eine natürliche Interaktion zwischen Nutzer und System realisieren konnten, aber auf der anderen Seite lediglich einzelne menschliche Akteure berücksichtigen. Andere Systeme mussten sich auf schlichte unimodale befehlsähnliche Aufrufe bzw. einfachste Sensor-Aktor-Kopplungen beschränken, um eine simultane Unterstützung mehrerer Nutzer zu ermöglichen.

Den Autoren ist kein System bekannt, das als lauffähige Implementation zur Verfügung steht *und* die im vorigen Abschnitt herausgearbeiteten Anforderungen erfüllt oder wenigstens die Erfüllung der Anforderungen ermöglicht.

V. ARCHITEKTUR

Um multimodale Interaktion in verteilten Intelligenten Umgebungen zu ermöglichen, sieht die MAIKE-Architektur (vgl. Abb. 1) zunächst eine Kapselung von Geräten und Interaktionskomponenten vor. Diese werden anschliessend durch eine geeignete Middleware und Koordinationsschicht miteinander verbunden. In dem konsequent dezentralen Ansatz wird auf eine zentrale Steuerungskomponente verzichtet. Im Gegensatz zu anderen Projekten, wie z.B. EOHR [12], werden keine Webservices benutzt. Zur gemeinsamen Datenhaltung wird ein Tuplespace [10] und zur Koordination der Komponenten das Protokoll *Contract Net* [1] verwendet. Nach einer kurzen Erläuterung von Interaktionskomponenten und Gerätekapselungen werden beide Konzepte eingehender beschrieben. Anschließend wird auf eine Reihe von Werkzeugen, die der Unterstützung der Entwicklung dienen, eingegangen.

A. Interaktionskomponenten und Geräte

Interaktionskomponenten dienen der Interaktion mit den Nutzern. Dabei werden unterschiedliche Modalitäten in unterschiedlichen Komponenten gekapselt. Auf der einen Seite gibt es graphische Nutzerinterfaces sowie eine Sprachsteuerung. Aber auch die Intentionsanalyse ist eine solche Komponente zur Interaktion (vgl. Tabelle I). Grundsätzlich unterscheiden wir nach [8] in unserer Architektur momentan drei unterschiedliche primitive Interaktionstypen (Performative) genannt: VERIFY, TELL und ASK mit den nahe liegenden Bedeutungen: Eine Aussage verifizieren, den Nutzern etwas mitteilen, bzw. nach einer Antwort fragen. Die eigentliche Funktionalität der Interaktionskomponenten ist durch eine Reihe zu definierender Funktionen festgelegt:

- `start(Performativ, Parameter)` ⇝ `Id`
- `abort(Id)`
- `waitForResult(Id)` ⇝ `Result`

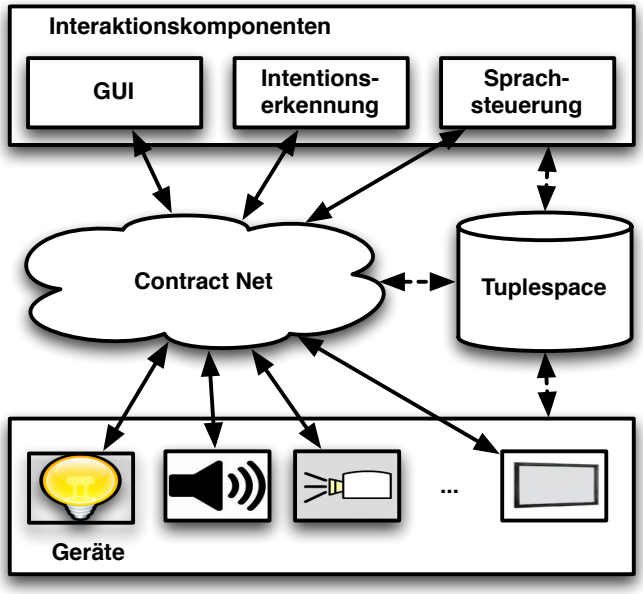

Abbildung 1. Schematische Darstellung der Projektarchitektur

Die übergebenen Parameter sowie das Resultat liegen als Abbildungen von Schlüsselwörtern auf Werte vor, die je nach Performativ auch anders aussehen können.

Eine Nutzerinteraktion wird durch den Aufruf von `start` in einer Komponente gestartet. Anschließend kann diese Interaktion entweder abgebrochen werden oder auf das Resultat gewartet werden. Die Umsetzung der Interaktion mit dem Nutzer obliegt dabei der implementierenden Komponente. So könnte das graphische Nutzerinterface eine VERIFY-Interaktion durch einen kleinen Dialog umsetzen, die Sprachkomponente durch eine Äusserung und die Intentionsanalyse dadurch, dass der aktuelle Raum- und Nutzerzustand unter Berücksichtigung von Hintergrundwissen ausgewertet wird.

Auch wenn momentan nur drei Interaktionskomponenten umgesetzt wurden, ist die Architektur nicht darauf eingeschränkt. So wäre zum Beispiel eine Gestenerkennung eine denkbare Erweiterung. Auch könnten bei Bedarf neue Performative leicht hinzugefügt werden.

Alle erwähnten *Geräte* sind momentan durch einfache Javaklassen gekapselt. Gleichzeitig ist die MAIKE-Architektur aber sowohl programmiersprachen- als auch betriebssystemunabhängig. Die verfolgte Strategie ist, die Funktionalität der einzelnen Geräte vollständig durch eine Klasse – oder eine Reihe von Methoden – zur Verfügung zu stellen. Dabei werden weder an die Funktionstypen noch an die Namensgebung besondere Anforderungen gestellt. So stellt unsere Implementation einer (dimmbaren) Lampe bspw. folgende Methoden zur Verfügung:

- `turnOn()`
- `turnOff()`
- `dimLight(float)`
- `isOn()` ⇝ `Boolean`
- `getDimValue` ⇝ `float`

B. Ein Tuplespace zur verteilten Datenhaltung

Als persistenter Speicher zur gemeinsamen Datenhaltung aller Komponenten dient ein Tuplespace. Dieser ist prinzipiell eine Multimenge von Tupeln. Diese wiederum sind Datenstrukturen mit unterschiedlicher Anzahl getypter Felder. Das Konzept wurde von GELERNTER [10] zur Kommunikation in verteilten Anwendungen entwickelt: Agenten lesen und schreiben Tupel unter Verwendung folgender Operationen aus bzw. in den Tuplespace:

- `read`, `readIfExists`: Lesen eines Tupels, das zu einem gegebenen Template passt. Während `read` blockiert, bis ein entsprechendes Tupel verfügbar wird, kehrt `readIfExists` sofort zurück, entweder mit einem Tupel oder einem Fehlercode.
- `take`, `takeIfExists`: analog zu `read`, wobei das entsprechende Tupel aus dem Tuplespace entfernt wird.
- `write`: schreibt ein neues Tupel in den Tuplespace

Unter Verwendung der blockierenden Aufrufe können Prozesse auf einfache Weise synchronisiert werden. Eine Datenstruktur, deren Konsistenz sichergestellt werden soll, etwa eine Kommando-Queue, kann durch ein spezielles Semaphor-Tupel geschützt werden. Dieses muss stets entnommen und zurückgeschrieben werden, um die Queue zu blockieren oder wieder freizugeben.

Oft bieten Tuplespaces zusätzlich zu den genannten Funktionen eine erweiterte Funktionalität an: (i) `notify` meldet ein Interesse an spezifischen Tupeln an. Sobald ein entsprechendes Tupel in den Tuplespace geschrieben wird, werden alle angemeldeten Clients benachrichtigt. (ii) Alle Tupel werden mit einer bestimmten `lease time` versehen, einer Zeitspanne, nach deren Ablauf das Tupel automatisch aus dem Tuplespace entfernt wird.

Wie bereits erwähnt, liegt die gesamte Funktionalität eines Gerätes oder einer Interaktionskomponente in einer entsprechenden Klasse gekapselt vor. Für jedes Gerät wird ein Objekt erzeugt, auf welches über den Tuplespace zugegriffen werden kann. Objekte werden durch ihre Attribute und Methoden repräsentiert. Um sie über einen Tuplespace verfügbar zu machen, wird zum einen eine Repräsentation der *Kommando-Queue* benötigt. Diese wird vom Gerät abgearbeitet und ermöglicht dadurch anderen Komponenten Methoden aufzurufen. Wie oben angeführt, benötigt eine Kommando-Queue ein Semaphor-Tupel. Zusätzlich wird jeder Methodenaufruf durch ein einzelnes Queue-Element beschrieben. Dieses enthält einen Index und das Kommando selbst. Die Indizes des ersten und letzten Elements werden im Semaphor-Tupel verwaltet. Daneben wird eine *Property Map* benötigt, die die Attribute enthält. Analog zu den Kommando-Queues werden Property Maps durch ein spezielles Semaphor-Tupel geschützt und enthalten für jedes Attribut ein Paar aus Name und Wert.

C. Koordination in heterogenen Geräte-Ensembles

Durch die oben beschriebene Repräsentation von Objekten im Tuplespace können Interaktionskomponenten und Geräte dargestellt werden. Im Ergebnis hat jede Interaktionskomponente neben der spezifischen modalitätsbedingten Sicht eine Sicht auf den vollständigen Weltzustand im Tuplespace. Eine zentrale Steuerungskomponente ist in diesem Konzept nicht notwendig. Zur Koordination der Komponenten wird *ContractNet* [1] verwendet: Ein Ansatz, um in einem Multiagentensystem Aufgaben in Form eines üblicherweise bei Ausschreibungen verwendeten Verfahrens zu verteilen.

Beispielsweise kann auf diese Weise folgender Ablauf realisiert werden: Die Komponente zur Intentionserkennung habe durch Bayes'sche Inferenz auf Basis der Sensordaten geschlussfolgert, dass eine Vortragssituation vorliegt. Da die resultierenden Aktionen in der aktuellen Situation jedoch mit hohen Kosten bewertet wurden, ist zur Klärung der Situation eine Rücksprache mit den Nutzern notwendig. Diese Aufgabe wird ausgeschrieben, und Komponenten wie Spracherkennung und GUI können sich (unter Angabe von Kosten und Nutzen der Rückfrage unter Einbeziehung der aktuellen Situation) um eine Zuweisung bewerben. Nach erfolgter Ausschreibung wird die beste Komponente ausgewählt und die entsprechende Aktion ausgeführt.

Dazu wird zunächst vom Initiator eine Ausschreibung gestartet, auf die die einzelnen Teilnehmer mit einem Angebot antworten. Ein solches Angebot enthält Qualitäts- und Zeitangaben, unter denen die Aufgabe bewältigt werden kann. Anschliessend wird das beste Angebot ausgewählt und der Auftragnehmer antwortet mit dem gewünschten Resultat. Die Auswahl des besten Angebotes erfolgt auf Basis der im Angebot übergebenen Angaben. Damit ein einzelner Initiator (Komponente im MAIKE-Modell) nicht alle Mitglieder kennen und einzeln nach den Angeboten befragen muss, wurde als zusätzliche Komponente ein *Contract Net Manager* eingeführt, welcher die komplette Abwicklung des Verfahrens übernimmt. Diesem wird lediglich eine Aufgabe und eine Policy zur Auswahl des besten Angebots übergeben und der Initiator erhält vom Manager die vollständige Antwort.

Die Benutzung des Managers erlaubt eine vollständig unabhängige Programmierung aller Komponenten, da keine Komponente wissen muss, welche anderen es gibt, sondern sich mit allen Anfragen an den Manager wendet. Dabei ist der Manager keinesfalls ein Flaschenhals im System, da es beliebig viele geben kann. Im Resultat wird aber jede Anfrage durch die im aktuellen Zustand am besten geeignete Komponente bearbeitet.

D. Werkzeuge zur Entwicklung

Um die Entwicklung des Gesamtsystems zu erleichtern haben wir eine Reihe von Hilfswerkzeugen implementiert, die im folgenden kurz erläutert werden sollen. Dazu gehören *NetworkPublisher*, *NetworkProxy*, *TuplespacePublisher* und *TuplespaceProxy*.

Der *NetworkPublisher* erweitert ein übergebenes Java-Objekt um unterschiedliche Netzwerksockets und ermöglicht damit eine einfache Kommunikation mit dem Objekt. Die Implementation basiert auf der Java Reflection API, mit deren Hilfe die implementierten Methoden analysiert werden können. Der Publisher stellt einen reinen TCP-Socket mit

`$> nc ourserver 8080`	*establish socket connection*
turnOn	*request to switch lamp on*
true	*object answers with method result*
getDimValue	*request current dim value*
1.0	*object answers with method result*
:quit	*end session*
:ok	*publisher answers*
$>	*back to shell*

Abbildung 2. Beispielsitzung einer Interaktion mit einer Lampe. Alle Nutzereingaben sind fett gedruckt dargestellt.

einem auch von Menschen lesbaren Protokoll, einen Telnet- und HTTP-Socket zur Verfügung. Eine Beispielsitzung ist in Abbildung 2 dargestellt. Das implementierte Protokoll basiert auf S-Expressions [13], welche eine plattform- und programmiersprachenunabhängige Kommunikation erlauben.

Der *NetworkProxy* bildet das Gegenstück zum NetworkPublisher. Damit ist es möglich, ein über den Publisher veröffentlichtes Objekt in einer anderen Java-Anwendung wie ein lokales Objekt zu benutzten. Dazu wird für das veröffentlichte Objekt automatisch ein Proxy-Objekt erzeugt, welches alle Methodenaufrufe entsprechend weiterleitet. Dadurch können alle Anwendungen vollkommen unabhängig von der zugrundeliegenden Middleware implementiert werden. So kann bspw. auch ein Java-Proxyobjekt für ein veröffentlichtes C++-Objekt erzeugt werden.

Analog zum NetworkPublisher verknüpft der *TuplespacePublisher* ein übergebenes Objekt mit dem Tuplespace. Dazu werden die oben erwähnten Kommando-Queues und PropertyMaps automatisch erstellt. Die Implementation basiert wiederum auf der Java Reflection API und veröffentlicht alle öffentlichen Methoden und Attribute des übergebenen Objektes. Weiterhin ist es möglich, einzelne Methoden mit weiteren Parametern zu annotieren, die die Darstellung im Tuplespace beeinflussen. Das Gegenstück bildet der *TuplespaceProxy*, welcher ein lokales Proxy-Objekt für ein im Tuplespace abgelegtes Objekt erzeugt und so wieder eine vollkommen transparente Kommunikation erlaubt.

VI. Evaluation

Wird die vorgestellte Systemarchitektur des Projekts MAIKE im Hinblick auf die eingangs beschriebenen Anforderungen analysiert, können nachfolgende Feststellungen getroffen werden.

Durch die Verwendung eines vergleichsweise einfach gehaltenen und somit auch von Menschen ohne weiteres nachzuvollziehenden Protokolls kann die technische Umsetzung von Prototypen sowie das Debugging von Komponenten in der Entwicklungsphase spürbar beschleunigt werden. In Kombination mit dem beschriebenen *NetworkPublisher* als Mechanismus zur Veröffentlichung von Komponenten via Netzwerk-Socket kann die Implementation der Funktionalität dieser Komponenten unabhängig von jeder Kopplung an die intendierte Middleware erfolgen.

Möglich ist damit eine strikte Trennung von Funktionalität und Netzwerkschicht, jedoch ohne den Vorteil eines lesbaren Protokolls und der damit verbundenen Fehlersuche (Debugging) einzubüßen. Diese Eigenschaft konnte, wie in [3] näher beschrieben, sowohl in unterschiedlichen Projekttreffen als auch in der Lehre bestätigt werden. Die beteiligten Forscher, Entwickler bzw. Studenten waren durchweg – auch ohne tiefgreifendes Expertenwissen in XML, UDP o.ä. – in der Lage, sich zügig in die jeweiligen Problemstellungen einzuarbeiten, eigene Komponenten zu entwickeln und diese in die vorhandene Infrastruktur zu integrieren. Die Anforderung der losen Kopplung der einzelnen Komponenten kann damit als erreicht betrachtet werden.

Wird statt des *NetworkPublishers* (in Kombination mit dem *NetworkProxy*) der *TuplespacePublisher* (zusammen mit dem *TuplespaceProxy*) verwendet, kann jede beliebige Komponente sofort über den Tuplespace verfügbar gemacht werden. Im Ergebnis sind die Properties von allen veröffentlichten Objekten im Tuplespace abgelegt. Dieser stellt auf diese Weise – bei Bedarf persistent – jeder (Interaktions-)Komponente eine Sicht auf den vollständigen Weltzustand bereit.

Aus dem Umfeld der Multiagentensysteme wurde das Interaktionsprotokoll *ContractNet* vorgestellt, das in MAIKE als zentraler Mechanismus zur Auflösung von Ambiguitäten verwendet wird. Interaktionskomponenten können sich per Ausschreibung mit anderen Interaktionskomponenten in Verbindung setzen, eingegangene Angebote prüfen, bewerten und sich für das jeweils plausibelste entscheiden. Im Sinne der Anforderung nach einer losen Kopplung der Komponenten, sieht die Umsetzung in MAIKE einen zentralen *ContractNetManager* vor. Interaktionskomponenten müssen somit nicht unbedingt voneinander wissen, und können jederzeit aus der Infrastruktur aussteigen bzw. in sie zurückkehren.

Der Aspekt der impliziten Interaktion wird in MAIKE durch Hinzunahme einer Interaktionskomponente zur Intentionserkennung bedient. Sie kann häufig wiederkehrende Abläufe im Nutzerverhalten erkennen, miteinander in Beziehung setzen und auf diese Weise Nutzerziele schon erkennen und entsprechende Aktionen auslösen, *bevor* ein explizites Kommando zur Konfiguration des Raumes auffordert. Bei Unklarheiten können – passend zur aktuellen Nutzungssituation – Rückfragen durch andere Komponenten angestoßen werden.

VII. Zusammenfassung

Die technologische Ausstattung von Räumen – sowohl im privaten Bereich als auch im Geschäftsumfeld – wird stetig erweitert. Die große Anzahl von unterschiedlichen Geräten überfordert viele Nutzer jedoch, wodurch das Potential solcher Einrichtungen nicht ausgeschöpft werden kann. Oftmals führt dies zu einer Frustration bzw. Abneigung des Nutzers gegenüber moderner Technik.

Intelligente Räume beinhalten Nutzerschnittstellen, die auf eine einfache und sinnvolle Bedienung ausgerichtet sind. Hierbei wird die Funktionalität des Raumes als Ganzes betrachtet, anstatt die Geräte isoliert zu sehen. Die Technik verschmilzt mit dem Raum, und es rückt somit die Gesamtfunktionalität in den Vordergrund: Die Technik ist allgegenwärtig, agiert jedoch im Hintergrund (*Ubiquitous Computing* nach [16]). Dies versetzt den Nutzer in die Lage, seine Konzentration

auf die eigentliche Aufgabe fokussieren zu können, ohne durch komplizierte Bedienungsvorgänge unnötig belastet und abgelenkt zu werden.

Im Rahmen des Forschungsprojektes MAIKE wird daher eine Architektur entwickelt, die eine intuitive Bedienung in dynamisch erweiterbaren und verteilten Räumen durch eine multimodale Interaktion zwischen Nutzern und einem heterogenen Geräteensemble ermöglicht. Dabei wird ein strikt dezentraler Ansatz verfolgt, bei dem die Komponenten bei Bedarf auf einer sehr abstrakten Ebene miteinander kommunizieren können. Durch die Nutzung eines Tuplespaces haben alle Komponenten Zugriff auf den Weltzustand. Weiterhin ermöglicht der Tuplespace eine verteilte Architektur, wie sie in solchen Ensembles benötigt wird. Der vorgeschlagene Ansatz zur losen Kopplung von Komponenten erlaubt einen plattform- und programmiersprachenunabhängigen Zugriff.

Anfallende Probleme bei der Interpretation des Befehls (Mehrdeutigkeiten, Unvollständigkeit) werden mittels *ContractNet* ausgeschrieben und können somit von der am besten geeigneten Komponente gelöst werden. Auf diese Weise entfällt eine zentrale Komponente, die die Delegation der Aufgaben übernimmt. Bei Unklarheiten wird ein Klärungsdialog initiiert, auf den die Nutzer via Sprache bzw. GUI reagieren können. Je nach Situation kann auch Kontextwissen (z.B. aus Sensorinformationen) genutzt werden, um dem Nutzer unnötige Dialoge zu ersparen, indem bei ausreichender Konfidenz auf einen Klärungsdialog verzichtet wird.

Im weiteren Verlauf des MAIKE-Projektes soll die Referenzimplementation vervollständigt und in Nutzerstudien evaluiert werden.

DANKSAGUNG

Diese Arbeit ist im Landesforschungsverbund *Mobile Assistenzsysteme* entstanden und wird im Rahmen der „Förderung von Forschung, Entwicklung und Innovation des Landes Mecklenburg-Vorpommern“ aus Mitteln des „Europäischen Sozialfond (ESF)“ und Mitteln des „Europäischen Fonds für regionale Entwicklung (EFRE)“ gefördert.

LITERATUR

[1] Fipa Contract Net Interaction Protocol Specification, 2002.

[2] http://www.inhaus-zentrum.de, 2010.

[3] Sebastian Bader, Gernot Ruscher, and Thomas Kirste. A middleware for rapid prototyping smart environments. In *Proceedings of the UbiComp'10 Demo Session*, 2010. accepted.

[4] Diane Cook and Sajal Das. *Smart Environments: Technology, Protocols, and Applications*. Wiley, 2005.

[5] Diane J. Cook, Michael Youngblood, III Edwin O. Heierman, Karthik Gopalratnam, Sira Rao, Andrey Litvin, and Farhan Khawaja. Mavhome: An agent-based smart home. *Pervasive Computing and Communications, IEEE International Conference on*, 0:521, 2003.

[6] Jerremy R Cooperstock, Sidney S Fels, William Buxton, and Kenneth C Smith. Reactive environments – throwing away your keyboard and mouse. *Communications of the ACM*, 40(9):65–73, September 1997.

[7] Christoph Endres, Andreas Butz, and Asa McWilliams. Survey of software infrastructures and frameworks for ubiquitous computing. *Mobile Information Systems Journal*, 1(1), Jan-Mar 2005.

[8] Tim Finin, Yanis Labrou, and James Mayfield. *Software agents*, chapter KQML as an agent communication language, pages 291–316. MIT Press, Cambridge, MA, USA, 1997.

[9] David Franklin. Cooperating with people: The Intelligent Classroom. In *Proc. 15th AAAI*. AAAI Press, 1998.

[10] D. Gelernter. Generative communication in Linda. *ACM Transactions on Programming Languages and Systems*, 7:80–112, 1985.

[11] Antonio Maña, Volkmar Lotz, Sebastian Feuerstack, Marco Blumendorf, Grzegorz Lehmann, and Sahin Albayrak. Seamless home services. In *Developing Ambient Intelligence*, pages 1–10. Springer Paris, 2006.

[12] Markus Berg, Nils Weber, Christoph Eigenstetter, and Antje Düsterhöft. Natürlichsprachliche Dialoge in Raumsteuerungssystemen. In *4. Kongress Multimediatechnik*, 2009.

[13] R. Rivest. S-expressions. Internet-Draft, 1997.

[14] Th. Kirste, T. Herfet, and M. Schnaider. EMBASSI: multimodal assistance for universal access to infotainment and service infrastructures. In *WUAUC'01*, pages 41–50. ACM Press, 2001.

[15] Wolfgang Wahlster. Dialogue systems go multimodal: The SmartKom experience. In *SmartKom: Foundations of Multimodal Dialogue Systems*, pages 3–27. 2006.

[16] Marc Weiser. The computer for the 21st century. *Scientific American*, 265(3):94–104, September 1991.