Lehrstuhl für Mensch-Maschine-Kommunikation

Entwicklung und Bewertung eines Rapid-Prototyping Ansatzes zur multimodalen Mensch-Maschine-Interaktion im Kraftfahrzeug

Franz Bernhard Niedermaier

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Ingolf Ruge
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. rer. nat. Manfred K. Lang, i.R.
	2.	Univ.-Prof. Dr.-Ing., Dr.-Ing. E.h. Günther Schmidt

Die Dissertation wurde am 19.08.2002 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 27.01.2003 angenommen.

Vorwort

Die vorliegende Arbeit entstand während meiner Tätigkeit in der Fahrzeugforschung der BMW Group in München. Sie wurde in enger Kooperation mit dem Lehrstuhl für Mensch-Maschine-Kommunikation der Technischen Universität München angefertigt.

Ich möchte mich an dieser Stelle bei Herrn Prof. Dr. Lang für die Unterstützung der Arbeit, die universitäre Betreuung und die wertvollen Hinweise bei der Durchführung bedanken. Des Weiteren gilt mein besonderer Dank Herrn Dr. Bengler für die fachliche Begleitung bei der BMW Group und die wertvollen Anregungen, die er als häufiger Diskussionspartner in die Arbeit eingebracht hat.

Daneben bedanke ich mich recht herzlich bei den Mitarbeitern der Abteilung Mensch-Maschine-Interaktion und Nutzerforschung für die hervorragende Zusammenarbeit. An dieser Stelle richte ich meinen Dank auch an die anderen Kollegen, Werkstudenten, Praktikanten und alle weiteren Personen, die zum Gelingen dieser Arbeit beigetragen haben.

Schließlich danke ich ganz besonders meiner Freundin Barbara Bumeder und meinen Eltern für ihre beständige Unterstützung. Sie haben dadurch einen wichtigen Anteil zum Erfolg der Arbeit beigesteuert.

München im August 2002

Bernhard Niedermaier

Zusammenfassung

Multimodale Bedienkonzepte bieten im Fahrzeug großes Potenzial, um trotz der zunehmenden Zahl an Fahrerassistenz- und -informationssystemen auch in Zukunft eine an die Fahraufgabe angepasste Bedienung der verfügbaren Funktionen zu ermöglichen. Da diese Nutzungskonzepte den Charakter eines Fahrzeugs sehr stark prägen, werden sie für den Kunden ein wichtiges Unterscheidungsmerkmal darstellen. Deshalb ist es von großer Bedeutung, dass sie bereits in der Konzeptphase des Fahrzeugentwicklungsprozesses erlebbar und damit bewertbar gemacht werden. Allerdings fehlt es hierzu bislang noch an geeigneten Entwicklungswerkzeugen.

Basierend auf einer Analyse der Anforderungen liefert die vorliegende Arbeit daher einen Beitrag zur Entwicklung des theoretischen Fundaments für Werkzeuge zur prototypischen Realisierung multimodaler Bedienkonzepte. Als Grundlagen dienen dabei eine kontextfreie und damit allgemein anwendbare Modellierung der Modalitäten, sowie eine an die Problemstellung angepasste formale Repräsentation des Bediendialogs. Darauf aufbauend werden Maßnahmen beschrieben, mit denen bereits im Spezifikationsprozess die Benutzbarkeit während der Fahrt unterstützt werden kann. Daneben wird eine universell einsetzbare Heuristik zur Ausführung des Dialogs entwickelt.

Zur Verifikation der Realisierbarkeit und der Tauglichkeit des theoretischen Ansatzes erfolgt im nächsten Schritt eine Umsetzung des formalen Gerüsts in einer Referenzimplementierung. Auf Grundlage des dabei entstandenen Werkzeugs wird schließlich ein Nutzertest mit Experten durchgeführt. Dabei zeigt sich, dass die vorgeschlagenen Konzepte und Methoden zur prototypischen Erstellung fahrzeugtauglicher multimodaler Dialoge gut geeignet und für Entwickler sehr verständlich sind. Die vorliegende Arbeit stellt somit eine sinnvolle formale Basis für entsprechend angepasste Rapid-Prototyping Werkzeuge bereit.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1 Einleitung

Multimodalität ist zur Zeit ein großer Hoffnungsträger, um die Bedienung komplexer Mensch-Maschine-Systeme zu erleichtern. Viele Produkte werben mit *„multimodaler Bedienbarkeit“* und die meisten wissenschaftlichen Tagungen im Bereich Mensch-Maschine-Interaktion beschäftigen sich mit diesem Thema. Auf den ersten Blick entsteht dabei leicht der Eindruck, dass die Anzahl der zur Verfügung stehenden Interaktionsmöglichkeiten im direkten Zusammenhang mit der Benutzbarkeit eines technischen Systems steht.

Multimodale Bedienung ist aber mehr als die parallele Disponibilität eines Spracherkenners und eines Bedienelements zur Eingabe von Nutzerintentionen. Die einzelnen Modalitäten müssen in einem schlüssigen Gesamtbedienkonzept integriert sein, um so in der jeweiligen Dialogsituation ihre spezifischen Vorteile optimal dem Nutzer zur Verfügung stellen zu können. Neben der technischen Realisierung sind hierbei eine Vielzahl ergonomischer, psychologischer und ästhetischer Aspekte sowie auch wirtschaftlich-ökonomische Anforderungen zu berücksichtigen, die eine enge interdisziplinäre Zusammenarbeit erforderlich machen.

Dies trifft insbesondere im Automobil zu. Dort muss sichergestellt sein, dass die Beeinflussung der primären Fahrzeugführungs- und -stabilisierungsaufgabe durch nicht fahrtbezogene Bedienvorgänge für *Fahrerinformations- und Fahrerassistenzsysteme* (TICS[1]) in einem verkehrssicheren Rahmen liegt. Damit bereits in einem frühen Stadium des Entwicklungsprozesses möglichst viele der relevanten Aspekte berücksichtigt werden können, ist zur Lösung dieser Aufgabe eine strukturierte Vorgehensweise erforderlich.

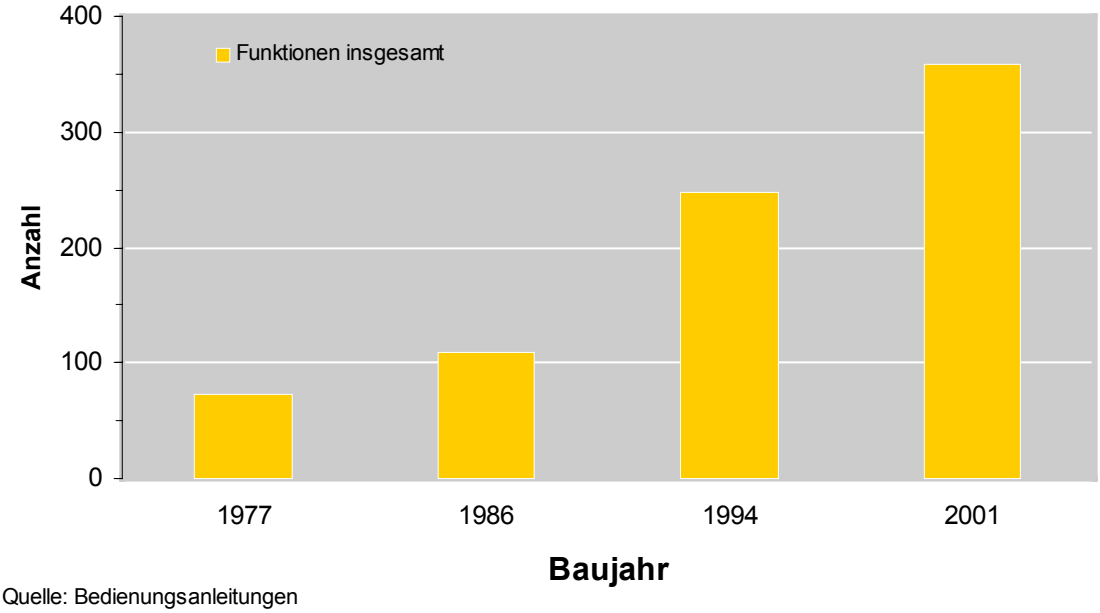
Die vorliegende Arbeit geht auf die für Fahrerinformations- und Fahrerassistenzsysteme zutreffenden speziellen Anforderungen und Randbedingungen ein und schafft basierend auf einem formalen Ansatz ein Rahmenwerk, das den Entwickler bei der Spezifikation und der Erstellung von fahrzeugtauglichen Bedienkonzepten in der frühen Phase des Entwicklungsprozesses optimal unterstützt. Entsprechend dieser Zielsetzung ist sie als eine disziplinübergreifende, praxisorientierte Arbeit mit einem deutlichen ingenieurwissenschaftlichen Schwerpunkt zu verstehen.

---

[1] TICS: Transport Information and Control Systems

1.1 Motivation

Betrachtet man den in Bild 1 dargestellten Anstieg der vom Fahrer während der Fahrt bedienbaren Funktionen in den verschiedenen Modellen einer Luxus-Limousine aus dem Premiumsegment, dann lässt sich eine stetige Zunahme beobachten. Im aktuellen Modell stehen ungefähr fünfmal so viele Funktionen zur Verfügung wie im ersten.



Quelle: Bedienungsanleitungen

Bild 1 Entwicklung der vom Fahrer bedienbaren Funktionen in einer Premium-Luxus-Limousine[1]. Wichtige Meilensteine stellen die Integration von Telefon, Navigation (1994), sowie zusätzlicher umfangreicher Kommunikations- und Unterhaltungskomponenten (2001) dar.

Der Grund hierfür liegt nicht in zusätzlichen Bedienmöglichkeiten zur Fahrzeugstabilisierung, sondern in der beständigen Erweiterung der verfügbaren Sekundärfunktionen. Dies sind zum einen Komfortfunktionen wie die elektrische Sitzverstellung oder Klimaautomatik. Zum anderen wurden Fahrerassistenzsysteme (FAS) wie beispielsweise das Navigationssystem oder ACC[2] zur Unterstützung der Fahrzeugführungs- und der Navigationsaufgabe eingeführt, genauso wie

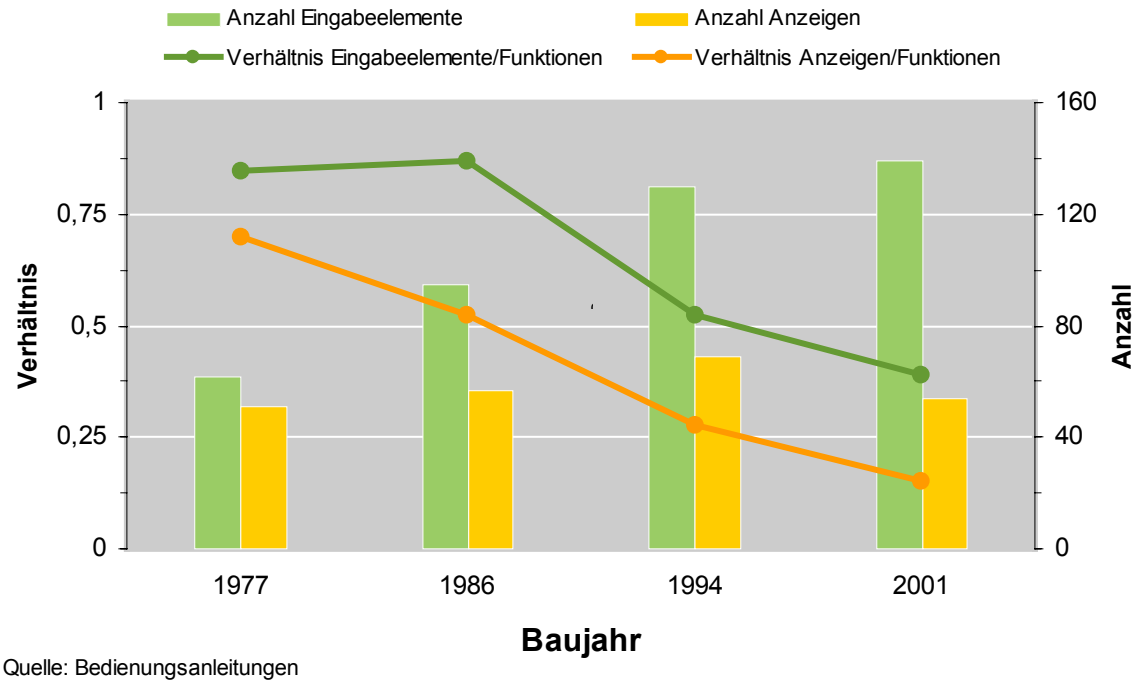
---

[1] Zur besseren Vergleichbarkeit geht die Vollausstattungsvariante des jeweiligen Modells zum Zeitpunkt der ersten Auslieferung in die Untersuchung ein. Neuerungen während eines Modellzyklus sind nicht berücksichtigt.

[2] ACC: Active Cruise Control

Fahrerinformations- (FIS) bzw. Infotainmentsysteme, die dem Fahrer umfangreiche Kommunikations- und Unterhaltungsmöglichkeiten gestatten.

Um trotz dieser starken Funktionszunahme eine verkehrssichere und verständliche Bedienung sicherzustellen, versuchen die Fahrzeughersteller seit jeher die Bedienung so einfach wie möglich zu gestalten. Dies ist vor allem für die Anzahl, Anordnung und Gestaltung der verschiedenen Eingabe- und Anzeigeelemente von Bedeutung, wie Bild 2 am Beispiel der verschiedenen Modelle einer Premium-Luxus-Limousine verdeutlicht. Trotz eines absoluten Anstiegs der Eingabe- und Anzeigeelemente haben sie relativ zu den bedienbaren Funktionen stark abgenommen.



Quelle: Bedienungsanleitungen

Bild 2 Entwicklung der Eingabe- und Anzeigemöglichkeiten absolut und relativ zur Zahl der Funktionen in einer Luxus-Limousine aus dem Premiumsegment

Hierbei handelt es sich allerdings zunächst um keine sprunghafte Entwicklung, sondern um eine sukzessive und partielle Weiterentwicklung der einzelnen Systeme. Aufgrund des mittlerweile erreichten Komplexitätsgrades hat dieses Vorgehen bei exklusiv ausgestatten Modellen seine Grenzen erreicht. Es zeichnet sich daher ein Paradigmenwechsel von den klassischen komponentenorientierten Schalterkonzepten zu gesamtheitlichen Ansätzen in Form integraler Anzeige- und Bedienkonzepte ab. Dabei werden die Informationen auf einem zentral angebrachten, graphikfähigen Display angezeigt. Die Bedienung erfolgt über ein multifunktionales Bedienelement. Durch die örtliche Trennung des Displays vom Bedienelement kann sowohl eine ergonomisch günstige Eingabe, als auch eine gute Ablesbarkeit der Anzeige ermöglicht werden (Bild 3) [Schrievers & Haller 1994] [Bengler, *et al.* 2002]. Die verschiedenen Funktionen werden zu Menüs zusammengefasst und können dadurch hierarchisch gegliedert werden. Dieser Ansatz erlaubt die Integration zusätzlicher Funktionen, ohne neue Bedienkomponenten einbauen zu

müssen. Allerdings sind solche Konzepte nicht per se einfacher zu bedienen, wie die Untersuchungen von [Schattenberg & Debus 2001] zeigen, vielmehr müssen sie durch eine entsprechende Gestaltung des Bediendialogs sehr genau auf die physische und kognitive Leistungsfähigkeit des Menschen in der Fahrsituation eingehen [Jung & Hamberger 2000] [Lang 2002].

Bild 3 Integrale Anzeige und Bedienung mit (links), bzw. ohne (rechts) örtlicher Trennung der Anzeigefläche und des Bedienelements

Ein viel versprechender Ansatz, um eine möglichst optimale Unterstützung des Fahrers zu gewährleisten, stellt die Bereitstellung zusätzlicher Ein- und Ausgabemöglichkeiten dar. Motivierend hierfür sind die erfolgreichen Ergebnisse entsprechender Studien aus anderen Bereichen [Hauptmann 1989] [Vo & Waibel 1993] [Oviatt 1996] [Althoff*, et al.* 2001]. Dabei wurden verschiedene Kombinationen aus Spracherkennung, Gestenerkennung, Kopfbewegungserkennung, Eye-Tracking[1], lippenlesenden Systemen und klassischen Eingabedevices wie Tastatur, Maus und Touchscreen untersucht.

Erste Studien, die hierzu in der Domäne Fahrzeug als WoZ[2]-Untersuchungen von [Bengler 1995] [Salmen*, et al.* 1999] [Detemple 2000] [Liu 2001] durchgeführt wurden, zeigen, dass die kombinierte Verwendung verschiedener Modalitäten wie beispielsweise Sprache und Haptik [Schattenberg & Debus 2001], Sprache und Gestik [Geiger*, et al.* 2001] [Zobl*, et al.* 2001] oder Sprache, Haptik und Gestik [Neuss 2000] auch hier Vorteile in Bezug auf Geschwindigkeit, Fehleranfälligkeit und Ablenkung gegenüber unimodaler Anzeige und Bedienung bringen. Diese

---

[1] Eye-Tracking: Meist videobasierte Verfahren, die aus der aktuellen Augenstellung einer Person das gerade betrachtete Objekt bestimmen. Diese Verfahren besitzen das Potenzial um als Mausersatz für räumliche Lokalisierungs- und Auswahlaufgaben eingesetzt zu werden, weil die Hände des Anwenders für andere Aufgaben frei bleiben.

[2] WoZ: Wizard of Oz

Vorteile können aber nur dann zu Tage treten, wenn die einzelnen Modalitäten in einem sinnvollen Verbund zusammenwirken.

Deshalb ist es bereits in einem sehr frühen Stadium des Entwicklungsprozesses [Jung & Hamberger 2000] [Bengler*, et al.* 2002] erforderlich, dass die zum Teil konkurrierenden Zielvorstellungen aus Design, Ergonomie und Technik zu einem Gesamtkonzept zusammengeführt, bewertet und gegebenenfalls überarbeitet werden. Zur Bewertung dieser Entwürfe werden Methoden des *Usability Engineering* [Nielsen 1993] eingesetzt. Obwohl manche dieser möglichen Verfahren, wie beispielsweise *Paper Mock-Ups*[1], mit relativ geringem Aufwand auskommen, können fundierte Aussagen zur Tauglichkeit eines Ansatzes im Allgemeinen lediglich mit einem zumindest in den wichtigsten Teilen funktionsfähigen System erfolgen. Da sich also die grundsätzliche Interaktionsphilosophie eines Systems bereits sehr früh im Entwicklungsprozess herauskristallisiert, gewinnt die Verfügbarkeit von Prototypen in dieser Phase zusätzlich an Bedeutung. Gerade die Erstellung dieser Prototypen ist aber ein sehr schwieriger und aufwändiger Prozess, da viele der technologischen, gestalterischen und ergonomischen Randbedingungen zu diesem Zeitpunkt noch nicht ausreichend geklärt sind. Deshalb durchlaufen diese Prototypen im Allgemeinen mehrere Überarbeitungsstufen. Aufgrund mangelnder Entwicklungsumgebungen müssen große Teile dieser Prototypen in einer höheren Programmiersprache wie C++ oder Java fest codiert werden. Dies bedeutet, dass die verschiedenen beteiligten Stellen ihre Ideen und Vorstellungen dem jeweiligen Programmierer vermitteln müssen. Diesem sind allerdings die besonderen Anforderungen meist nicht ausreichend bekannt. Insbesondere kurzfristige und kleinere Änderungen sind dadurch mit einem hohen kommunikativen und zeitlichen Aufwand verbunden. Dem Programmierer stehen zwar durch diese offenen Entwicklungsumgebungen sehr viele Freiheitsgrade zur Verfügung, er kann aber auch sehr leicht architektonische, logische oder inhaltliche Fehler begehen, die später sehr schwer aufzufinden und zu korrigieren sind. Gleichzeitigkeit führt diese Offenheit oftmals zu einer unstrukturierten und nicht ausreichend geplanten Programmierung, bei der isolierte Systeme entstehen, deren Teilkomponenten kaum wieder zu verwenden sind.

Aus diesen Gründen besteht ein großer Bedarf an spezifischen Werkzeugen, die zwar offen für neue Bedienkonzepte und Interaktionstechnologien sind, gleichzeitig den Entwickler aber durch den Erstellungsprozess führen und ihn bei der Vermeidung von Fehlern so weit als möglich unterstützen [Maybury & Stock 1999]. Allerdings kann dieser Bedarf bisher noch nicht in einem ausreichenden Umfang befriedigt werden, weil bestehende Werkzeuge wie beispielsweise CASE-Tools[2] ein zu allgemeines Anwendungsfeld haben und andere, wie sprach- oder graphik-

---

[1] Paper Mock-Up: Auf Papierausdrucken beruhende Attrappe des zu untersuchenden Bedienkonzepts.

[2] CASE: Computer Aided Software Engineering

orientierte, zu speziell auf bestimmte Interaktionstechnologien zugeschnitten sind. Zudem geht keines dieser Werkzeuge auf die besonderen Dialoganforderungen im Fahrzeug ein.

1.2 Zielsetzung

Die vorliegende Arbeit hat als Ziel ein theoretisch begründetes Fundament zu schaffen, das als Grundlage für die Entwicklung von Spezifikations- und Rapid-Prototyping-Werkzeugen für multimodale Bedienkonzepte in der Domäne Automobil geeignet ist. Der Ansatz soll dabei die Mächtigkeit besitzen, bereits während der Dialogerstellung Teilaspekte der Dialogqualität im Sinn entsprechender Normen und Richtlinien abzusichern. Die Umsetzbarkeit des Konzepts soll anhand einer beispielhaften Implementierung eines solchen Werkzeugs gezeigt werden. Der Ansatz wird dann anhand dieser Umsetzung evaluiert.

Detaillierter dargestellt umfasst die Zielsetzung folgende Punkte:

- Analyse der in der Domäne Automobil bestehenden Rahmenbedingungen und Ableitung der daraus resultierenden Anforderungen an das Werkzeug bzw. den Ansatz zur Dialogspezifikation. Dabei stehen eher die Belange in der Konzeptphase und Spezifikationsphase als die Anforderungen bei der Erstellung serientauglicher Systeme im Vordergrund.
- Abstraktion der Rahmenbedingungen in eine generelle und formale Beschreibung des multimodalen Bediendialogs. Diese formale Repräsentation soll dabei insbesondere auf folgende Punkte eingehen:
  - Einfache Integration neuer und Wiederverwendung vorhandener Interaktionsmöglichkeiten
  - Zentrale Konfiguration der Interaktionsmöglichkeiten
  - Allgemein anwendbare Beschreibung des Bediendialogs
  - Formalisierung der dialogqualitätssichernden Maßnahmen
  - Spezifikationsgrundlage für spätere Seriensysteme
- Beispielhafte Umsetzung der formalen Beschreibung in einer Referenzimplementierung eines Werkzeugs zum Rapid-Prototyping von multimodalen Bedienkonzepten für TICS:
  - Frühzeitiges Erlebbarmachen des Bedienkonzepts als Basis für Usability Untersuchungen im Simulator oder realen Fahrzeug
  - Optimale Unterstützung der Entwickler im Gestaltungsprozess
  - Unterstützung der interdisziplinären Zusammenarbeit von Technik, Design und Ergonomie

- Evaluation des formalen Ansatzes und der Referenzimplementierung in einem Nutzertest mit Experten auf dem Gebiet der HMI-Entwicklung. Dabei sollen Stärken und Schwächen für zukünftige Weiterentwicklungen identifiziert werden.

1.3 Aufbau der Arbeit

Für ein besseres Verständnis der Vorgehensweise und Zusammenhänge erfolgt zunächst eine kurze Beschreibung des Aufbaus dieser Arbeit.

In Kapitel 2 werden als Erstes die notwendigen Grundlagen und Voraussetzungen zum Verständnis der Arbeit geschaffen. Diese umfassen die Definition wichtiger Begriffe sowie die Thematisierung der verschiedenen Aspekte multimodaler Mensch-Maschine-Systeme. Dabei werden jeweils Forderungen abgeleitet, die von einem Rapid-Prototyping-Werkzeug für fahrzeugtaugliche multimodale Bedienkonzepte adressiert werden müssen.

Basierend darauf wird in Kapitel 3 ein theoretisches Gerüst geschaffen, das diesem Anforderungskatalog bereits auf formaler Ebene begegnen kann. Dies umfasst die kontextfreie Modellierung der Modalitäten und die Repräsentation des Bediendialogs. Darauf aufsetzend werden Maßnahmen zur Unterstützung des Entwicklers bei Erstellung fahrzeugtauglicher Dialoge sowie ein allgemein anwendbares Verfahren zur Interpretation des Dialogs bei der Dialogausführung dargestellt.

Kapitel 4 beschreibt die Umsetzung des theoretischen Konzepts. Dazu werden mit Hilfe einer Referenzimplementierung die Realisierbarkeit und die Tauglichkeit des formalen Ansatzes unter Beweis gestellt. Wesentliche Aspekte sind dabei die Umsetzung der formalen Repräsentation, die Unterstützung des Entwicklers bei der Erstellung des Dialogs und die verschiedenen zur Dialogausführung notwendigen Komponenten.

Als nächster Schritt erfolgt in Kapitel 5 eine Evaluation des theoretischen Ansatzes und der vorgeschlagenen Implementierung. Hierzu wird mit mehreren Experten eine Nutzerstudie durchgeführt.

Schließlich enthält Kapitel 6 eine Zusammenfassung der Arbeit sowie einen Ausblick über sinnvolle und notwendige weitere Schritte.

Kapitel 2 Grundlagen und Einführung in die Thematik

In diesem Kapitel werden zunächst die für das Verständnis der Arbeit notwendigen begrifflichen Grundlagen geschaffen. Basierend darauf folgt eine Einführung in die Thematik. Dabei wird auf technologische Aspekte und Qualitätsmerkmale multimodaler Mensch-Maschine-Systeme eingegangen. Daneben werden Verfahren zur Bewertung und Evaluation von Bedienkonzepten dargestellt. Schließlich werden Anforderungen an Werkzeuge zur Erstellung multimodaler Bedienkonzepte erörtert. Bei der Diskussion der einzelnen Gesichtspunkte wird vom allgemeinen Stand der Technik ausgehend der spezielle Anwendungsfall im Fahrzeug erläutert. Daraus werden die Anforderungen für die vorliegende Arbeit abgeleitet.

2.1 Definition und Abgrenzung grundlegender Begriffe

Die Vermischung von englischen und deutschen Fachbegriffen und die uneinheitliche Verwendung ihrer Abkürzungen sorgen häufig für Verwechslungen. Aus diesem Grund werden zunächst die drei wichtigsten Begriffe *Mensch-Maschine-Interaktion*, *Mensch-Maschine-System* und *Mensch-Maschine-Schnittstelle* erläutert und voneinander abgegrenzt. Gleichzeitig werden die zugehörigen Abkürzungen eingeführt.

2.1.1 Mensch-Maschine-Interaktion

Mensch-Maschine-Interaktion (MMI[1]) oder englisch *Human-Computer-Interaction (HCI)* wird in [Hewett, *et al.* 1992] folgendermaßen definiert:

> *„Human-Computer-Interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human."*

[1] Die Abkürzung *MMI* steht in dieser Arbeit für *Mensch-Maschine-Interaktion.*

MMI bezeichnet die gesamte Fachrichtung, die sich mit den verschiedenen Aspekten der Kommunikation des Menschen mit einem Computer bzw. allgemein mit einer Maschine beschäftigt. Sie ist – wie aus der obigen Definition hervorgeht und auch von [Maybury & Wahlster 1998] [Oviatt 1999] so dargestellt wird – keiner einzelnen Disziplin zuordbar, sondern sie beruht auf einer synergetischen Kombination aus *Technik*, *Design* und *Ergonomie* (Bild 4). Denn nur durch diese interdisziplinäre Kooperation können die technischen Rahmenbedingungen mit den Vorstellungen des Designs und den Anforderungen des Benutzers so zusammengeführt werden, dass ein funktionsfähiges, ästhetisch ansprechendes und gleichzeitig auch bedienbares Produkt entsteht. Wie bereits in Kapitel 1 motiviert wird, ist genau diese Symbiose oberstes Entwicklungsziel in der Domäne Automobil.

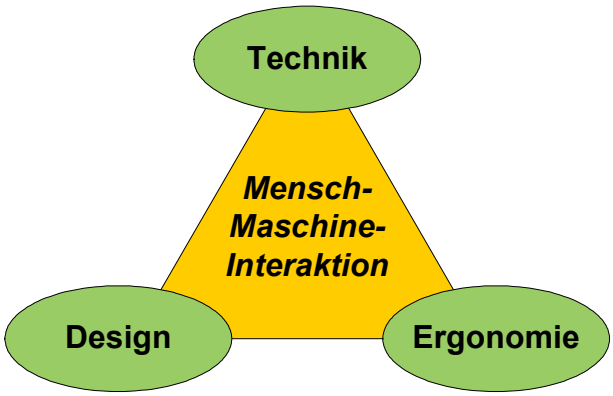


Bild 4 Interdisziplinärer Charakter der MMI

2.1.2 Mensch-Maschine-Systeme

Allgemeine Definitionen des Begriffs *Mensch-Maschine-System (MMS[1])* oder englisch *Human-Machine-System (HMS)* werden in [Geiser 1990] [Johannsen 1993] [Timpe & Kolrep 2000] gegeben.

> *„Bei einem Mensch-Maschine-System wirkt der Mensch mit einer Maschine mit dem Ziel zusammen, eine selbstgewählte oder vorgegebene Aufgabe zu lösen."*

Der Begriff Maschine bezeichnet dabei ein allgemeines technisches System, das beispielsweise eine Produktionsanlage, ein Gerät, ein Software-System oder ein Fortbewegungsmittel sein kann. [Schmidtke 1981] beschreibt eine systemtechnische Darstellung eines MMS. Der Mensch wird durch seine Sensoren zur Aufnahme und seine motorischen Fähigkeiten zur Abgabe von Informationen sowie einer auf seinen Zielen, Regeln und Wissen beruhenden informationsverarbeitenden Komponente modelliert. Die Maschine wird in analoger Weise durch drei Kompo-

[1] Die Abkürzung *MMS* steht in dieser Arbeit für *Mensch-Maschine-System.*

nenten repräsentiert. Über entsprechende Bedienelemente und Anzeigen werden Informationen aufgenommen und abgegeben. Der Zusammenhang zwischen den jeweiligen Ein- und Ausgaben ist dabei durch den zugrunde liegenden technischen Prozess bestimmt. Koppelt sich der Mensch mit seinen sensorischen und aktuatorischen Fähigkeiten an die entsprechenden Schnittstellen der Maschine, entsteht der in Bild 5 dargestellte geschlossene Regelkreis.

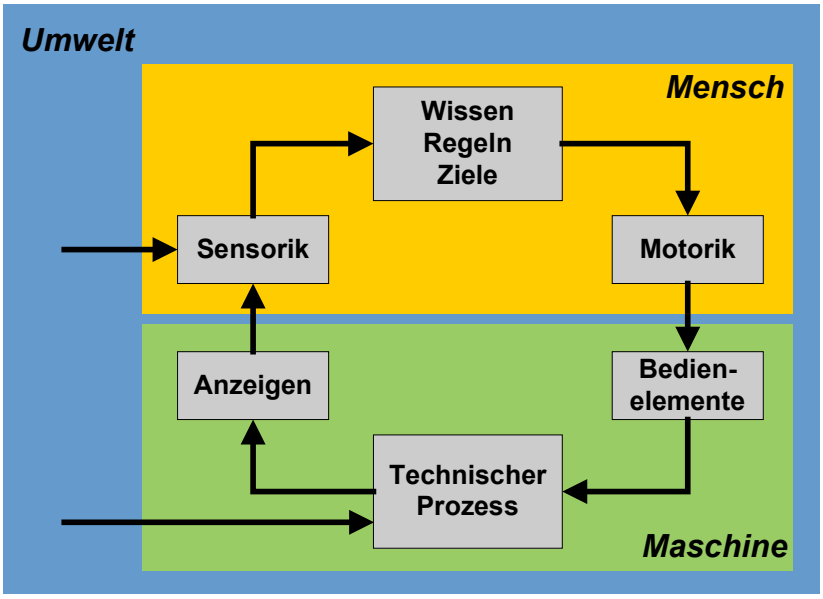


Bild 5 Struktur eines MMS als Regelkreis

Im Gegensatz zu MMI (2.1.1), der übergeordneten Bezeichnung des gesamten Fachgebiets, ist ein MMS also der tatsächliche symbiotische Zusammenschluss eines oder mehrerer Menschen mit einer Maschine um ein bestimmtes Ziel zu erreichen.

Im Fahrzeug steht der Mensch dabei zwei technischen Prozessen gegenüber. Dies ist zum einen der fahrdynamische Prozess zur Regelung und Stabilisierung des Fahrzeugs und zum anderen der Interaktionsprozess zur Bedienung der Fahrerinformationssysteme und -assistenzsysteme. Der letztere steht im Zentrum der vorliegenden Arbeit. Obwohl diese beiden MMS vom technischen Standpunkt aus weitgehend voneinander getrennt betrachtet werden können, dürfen ihre gegenseitigen Auswirkungen auf die kognitive Leistungsfähigkeit des Menschen nicht außer Acht gelassen werden.

2.1.3 Mensch-Maschine-Schnittstelle

Mit den alternativen Bezeichnungen *Mensch-Maschine-Schnittstelle* (*MMS*), *Mensch-Maschine-Interface* (*MMI*), *Man-Machine-Interface* (*MMI*) oder *Human-Machine-Interface* (*HMI*[1]) sind jeweils alle

„Eigenschaften der Komponenten eines MMS, die für die Interaktion mit dem Benutzer relevant sind“,

gemeint [Baggen & Hemmerling 2000]. Ein weit verbreitetes und bei einer lediglich physikalischen Betrachtung durchaus eingängiges Modell reduziert das HMI auf die in Bild 5 dargestellte Grenzfläche zwischen Mensch und Maschine, also auf die verschiedenen Anzeige- und Bedienelemente mit denen der reine Informationsaustausch stattfindet. Tatsächlich ist diese Schnittstelle aber nicht so klar fassbar. Sie muss eher als ein Übergangsbereich verstanden werden, weil weiche Faktoren wie

- Leistungsfähigkeit des Menschen
- Motivation des Menschen
- Kenntnisse und Fähigkeiten des Menschen
- Umwelt (Verkehrssituation)
- Zu bearbeitende Aufgabe
- Technische Randbedingungen

einen großen Einfluss auf die Effektivität des HMI haben. Um diese technischen und benutzerbezogenen Kriterien systematisch erfassen zu können, werden entsprechende *Modelle* eingeführt [Geiser 1990] [Lang 1998]. Ein Modell *Y(X)* ist dabei

„eine idealisierte, eingeschränkte Beschreibung eines Objekts X durch einen Autor Y.“

So kann der Benutzer mit seinem Aufgabenmodell B(A) die Durchführung der Aufgabe planen und mit dem inneren Modell B(S) das Verhalten der Maschine vorhersagen und erklären. Um als kooperatives System den Benutzer optimal unterstützen zu können, ist in der Maschine gleichzeitig ein Modell des Benutzers M(B) und der Aufgabe M(A) notwendig (Bild 6).

[1] *HMI* steht in dieser Arbeit für *Mensch-Maschine-Schnittstelle*. Aus Gründen der Eindeutigkeit wird die englische Abkürzung verwendet.

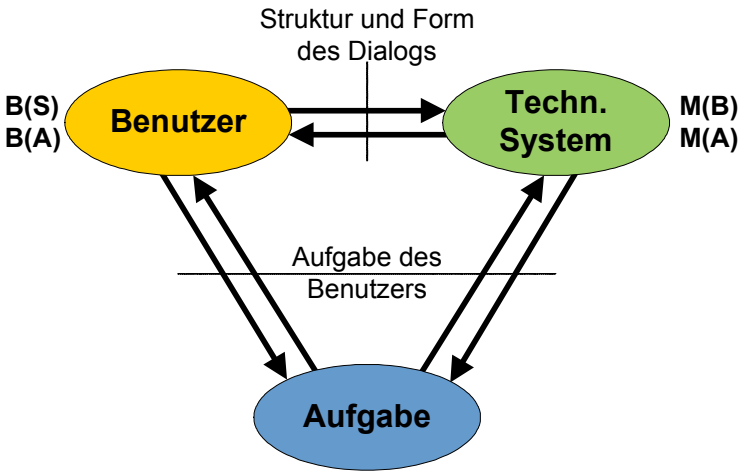


Bild 6 Für HMI relevante Modelle [Lang 1998]

Dieses allgemeine Modell besitzt auch in der Domäne Fahrzeug Gültigkeit. Wichtig ist dabei, dass insbesondere im Modell M(B), aber auch in M(A) die besondere Situation des Benutzers während der Fahrt berücksichtigt wird. Dies wird in 2.3 detaillierter dargestellt.

2.2 Multimodale Mensch-Maschine-Systeme

2.2.1 Multimodalität

2.2.1.1 Definition

„Multimodal systems process combined natural input modes – such as speech, pen, touch, manual gestures, gaze and head and body movements – in a coordinated manner with multimedia system output“ [Oviatt 1999].

Der Begriff *„multimodal“* leitet sich aus *„multi“* (lat.: mehrere, viele) und *„modus“* (lat.: die Art und Weise bezeichnend) ab. Bezogen auf MMI nutzen multimodal bedienbare MMS die verschiedenen spezialisierten Interaktionsformen des Menschen zur Abgabe *und* Aufnahme von Informationen um eine möglichst natürliche und intuitive Bedienung zu ermöglichen. Wichtiges Kennzeichen dabei ist, dass mehrere solcher Ein- und Ausgabemöglichkeiten (siehe 2.2.2) in einem HMI verfügbar sind und in einer konsistenten Weise zusammenwirken.

2.2.1.2 Potenzial Multimodalität

Das Potenzial multimodaler HMIs liegt nach [Maybury & Stock 1999] [Oviatt 1999] in einer effizienten, effektiven, robusten und natürlichen Bedienung (Bild 7). Der experimentelle Nachweis dieser Vorteile wurde in Studien, wie sie beispielsweise von [Hauptmann 1989] [Vo & Waibel 1993] [Oviatt 1996] [Althoff*, et al.* 2001] durchgeführt wurden, mehrfach erbracht. Zusätzlich kann im Fahrzeug durch multimodale Bedienung die Passung zwischen Bedien- und Fahraufgabe optimiert werden [Bengler 2001]. Wie in 2.3 gezeigt wird, ist es allerdings hierzu von

grundlegender Bedeutung, dass bestimmte Gestaltungsaspekte des Mensch-Maschine-Dialogs berücksichtigt werden.

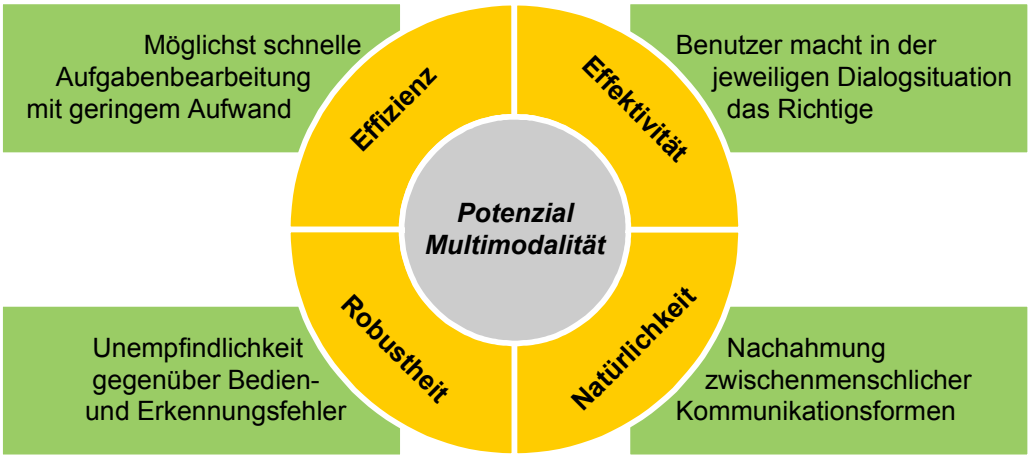


Bild 7 Potenzial Multimodalität

2.2.1.3 Einschränkungen Multimodalität

Die durch gut gestaltete multimodale Systeme erreichbaren Vorteile müssen allerdings mit einem deutlich erhöhten technischen Aufwand bezahlt werden, der sich letztendlich auf die Herstellungskosten und somit den Verkaufspreis niederschlägt. Aufgrund des hohen Preisdrucks stellt dies im Fahrzeugbau einen stark limitierenden Faktor dar. Die größten kostentreibende Faktoren sind hierbei:

- Zusätzliche Hardwarekomponenten,
  wie beispielsweise Mikrofone, Kameras oder spezielle Steuergeräte, die zur Funktionsrealisierung notwendig sind.
- Erhöhte Anforderungen an Infrastruktur,
  um die notwendige CPU-Leistung, Arbeitsspeichergröße, Datentransferrate auf dem Fahrzeugbus etc. zu erreichen.
- Komplexität der Software – Hoher Entwicklungsaufwand.
  Es sind große zeitliche und finanzielle Aufwendungen zur Spezifikation, Realisierung und Verifikation der Softwarekomponenten erforderlich.

2.2.2 Modalitäten

Wie in 2.1.3 bereits angedeutet, stellt die Realisierung des physikalischen Informationsaustauschs zwischen dem Menschen und der Maschine einen wesentlichen Bestandteil des HMIs dar. Der Mensch kann dabei Informationen auf sehr unterschiedliche Arten und Weisen abgeben bzw. aufnehmen. [Hedicke 2000] identifiziert hierzu fünf prinzipiell geeignete Schnittstellen, die in Anlehnung an entsprechende menschliche Sinneskanäle als *Auditives*, *Visuelles*, *Taktiles*, *Olfaktorisches* und *Gustatorisches Interface* bezeichnet werden (Bild 8). Die letzten beiden

Schnittstellen werden aufgrund der geringen Bandbreite und der fehlenden technischen Lösungen in dieser Arbeit nicht betrachtet. Der Informationsaustausch zwischen Mensch und Maschine findet in erster Linie auf akustischem, optischem und haptischem Wege statt.

Da der Mensch seine rezeptorischen und aktuatorischen Fähigkeiten in einer sehr vielfältigen und facettenreichen Weise einsetzen kann, ist eine gesamtheitliche technische Realisierung auf der Ebene dieser Interfaces allerdings unerreichbar. So ist es beispielweise nicht möglich das taktile Interface zu realisieren, weil die mannigfaltigen Möglichkeiten des Menschen, Kräfte aufzunehmen und abzugeben, technisch nicht generell nachgebildet werden können. Isolierte Interaktionsformen des Menschen, wie beispielsweise Sprechen, die Erkennung von Gesten oder die Registrierung von Kräften an speziellen Bedienelementen, können aber technisch nachgebildet werden. Jede so realisierbare Interaktionsmöglichkeit des Menschen mit der Maschine wird als *Modalität* bezeichnet. Je nachdem ob der Mensch eine solche Modalität zur Abgabe oder Aufnahme von Informationen benutzt, wird sie als *Aktionsmodalität* oder *Wahrnehmungsmodalität* bezeichnet [Hedicke 2000].

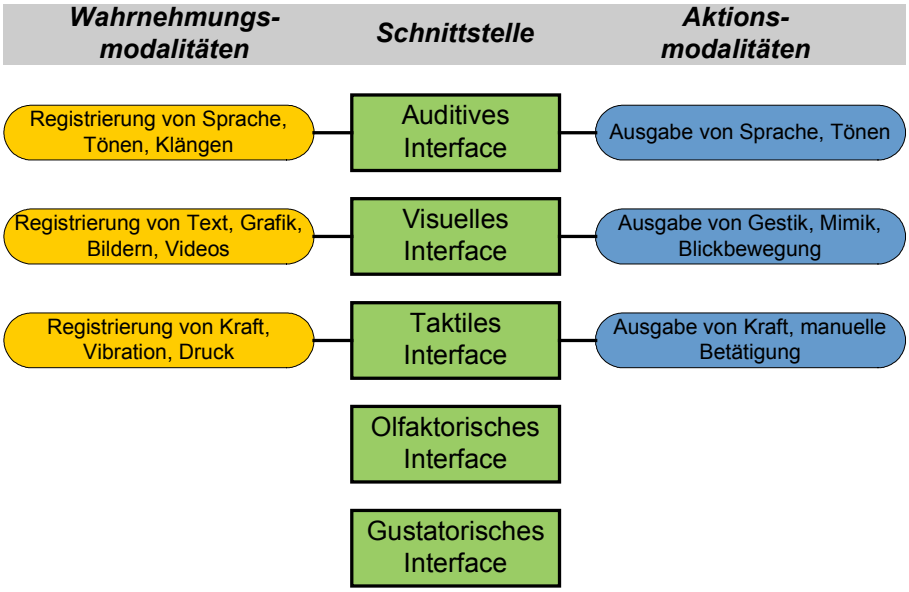


Bild 8 Modalitäten und Zuordnung zu den verschiedenen Schnittstellen bei einem HMI

Die in Bild 8 auf die genutzten physikalischen Effekte abstrahierten Wahrnehmungs- und Aktionsmodalitäten werden nachfolgend anhand einiger Beispiele konkretisiert. Die Darstellung gibt einen Überblick über den aktuellen Stand der Technik und erhebt keinen Anspruch auf Vollständigkeit. Es wird aber jeweils auf die Tauglichkeit der jeweiligen Modalität im Automobil eingegangen.

2.2.2.1 Auditives Interface

Sprachsignale, Töne, Klänge oder Geräusche beinhalten die mittels des auditiven Interface übermittelten Informationen. Dabei dient der Luftschall als Trägermedium. Auf technischer Seite sind hierzu Mikrofone und Lautsprecher als Sensoren und Effektoren erforderlich. Beim Men-

schen erfolgt die Informationsrezeption über das Gehör und die Informationsabgabe über den Vokaltrakt.

Spracherkennung

Die relevante Aktionsmodalität zur Eingabe akustischer Informationen in ein technisches System ist die menschliche Sprache. Mittels Automatischer Spracherkennungssysteme (ASR[1]) wird das Sprachsignal in eine für den Rechner semantisch interpretierbare Form gebracht. Der Erkennungsvorgang besteht aus einem Vergleich des zu erkennenden Signals mit einem Satz relevanter Prototypen, deren Bedeutung bekannt ist. Technisch wird dieser Vergleich im Allgemeinen mit *statistischen Klassifikatoren* durchgeführt, die basierend auf *Hidden-Markov-Modellen* (HMM) und mit speziellen Algorithmen (z.B. Viterbi-Algorithmus) die Wahrscheinlichkeit berechnen, dass das aktuelle Sprachsignal einem speziellen Prototyp entspricht. [Ruske 1994] gibt hierzu eine umfassende Beschreibung.

Sprache als natürlichste menschliche Kommunikationsform stellt eine sehr mächtige Aktionsmodalität dar, insbesondere da die Hände des Benutzers für andere Aufgaben frei bleiben. Allerdings ist diese Technologie sehr empfindlich gegenüber störenden Umwelteinflüssen, weshalb sie stets mit einer gewissen Fehlerkennungsrate behaftet sein wird. Daneben benötigen derzeit kommerziell verfügbare Produkte ein auf bestimmte Kommandowörter und Satzkonstrukte beschränktes Vokabular. Eingaben des Benutzers, die außerhalb dieses Vokabulars liegen (*Out-Of-Vocabulary*), können also nicht erkannt und semantisch interpretiert werden. Deshalb sind spezielle Dialogführungstechniken erforderlich. [Yankelovich 1996] gibt hierzu einen allgemeinen Überblick über grundsätzliche Verfahren. [Niedermaier 1999] beschreibt einen für das Fahrzeug tauglichen Ansatz, der durch die Verwendung von inkrementellen Sprachausgaben[2] die unterschiedlichen Anforderungen von Erstnutzern und Experten adressiert.

Natürlichsprachliche Erkennungssysteme (NL[3]) lösen sich von diesen statischen Wortschätzen und erlauben basierend auf einem sehr großen Vokabular wesentlich freiere Formulierungen durch den Benutzer. Allerdings ist die Vokabulargenerierung, der Erkennungsvorgang selbst und die semantische Interpretation der Eingaben mit einem deutlich höheren Aufwand verbunden. [Pinkal 2000] gibt hierzu einen Überblick über die formale Repräsentation und semantische Verarbeitung natürlicher Sprache.

---

[1] ASR: Automatic Speech Recognition

[2] Mehrstufige Sprachausgaben mit zunehmenden Unterstützungsgrad. Dadurch wird auf die unterschiedlichen Erfahrungsstände der Nutzer im Umgang mit dem System eingegangen.

[3] NL: Natural Language

„Eyes free – Hands Free“ [Bengler*, et al.* 2000], das heißt die weitgehend blickabwendungsfreie und berührungslose Bedienbarkeit, ist der größte Vorteil von Sprache im Fahrzeug. Gleichzeitig müssen aufgrund der nicht konstanten Geräuschbedingungen im Fahrzeug große technologische Anstrengungen unternommen werden, um hohe Erkennungsraten gewährleisten zu können. Daneben sollen diese Systeme unabhängig vom jeweiligen Sprecher und ohne Trainingsphase benutzt werden können.

Sprachausgaben

Sprachausgaben stellen einen Weg dar, um akustische Informationen von der Maschine zum Menschen zu übertragen. Entsprechend der Vorgehensweise bei der Generierung der Sprachausgaben können zwei prinzipielle Verfahren unterschieden werden:

- Ausgabe von Sprachaufzeichnungen (*Pre-Recorded-Speech*)
  Bei diesem einfachen Verfahren sind alle möglichen Ausgaben als Aufzeichnungen eines realen Sprechers statisch in das System integriert. Auf diese Weise können qualitativ hochwertige und gut verständliche Sprachausgaben erzeugt werden. Der starre Wortschatz macht diese Systeme allerdings sehr unflexibel. Anwendungen wie das Vorlesen von freien Texten (z.B. e-mails) sind daher nicht realisierbar. Im Fahrzeug wird dieses Verfahren derzeit in Navigationssystemen und bei Sprachbediensystemen eingesetzt.

- Synthetische Generierung des Sprachsignals
  Ein anderer Ansatz ist die dynamische Erzeugung des Sprachsignals mittels *Sprachsynthesesystemen* (TTS[1]). Diese generieren aus einem geschriebenen Text das entsprechende Sprachsignal. Technologisch können dabei so genannte parametrische (regelbasierte) Verfahren und datenbasierte (konkatenative) Verfahren unterschieden werden [Sproat & Olive 1995] [Holzapfel 2000]. Aufgrund der natürlicher klingenden Sprache werden letztere im Allgemeinen in kommerziell erhältlichen Systemen eingesetzt. Hierbei wird die Sprache aus einzelnen Bausteinen zusammengesetzt, die aus Sprachaufnahmen eines realen Sprechers erzeugt worden sind. Für möglichst natürlich klingende Lautübergänge erweisen sich dabei Diphone als sinnvolle Bausteine.

Mit TTS-Systemen können keine so qualitativ hochwertigen Sprachausgaben wie mit Pre-Recorded-Speech erzielt werden, weshalb ihr Einsatz im Fahrzeug nicht unproblematisch ist. Aufgrund ihrer großen Flexibilität eignen sie sich allerdings insbesondere zum Vorlesen von freien Texten, wie beispielsweise e-mails und Nachrichten, sowie zur prototypischen Implementierung von Bediendialogen.

---

[1] TTS: Text-to-Speech

Nonverbale akustische Anzeigen:

Eine weitere Möglichkeit zur Übertragung von Informationen über das auditive Interface stellen nonverbale akustische Anzeigen dar. Sie werden üblicherweise zur Ausgabe von Warntönen oder zur akustischen Untermalung bestimmter Interaktionsereignisse eingesetzt. [Brewster*, et al.* 1993] zeigt hierzu, dass aus verschiedenen Tonsequenzen, Rhythmen und Klangfarben bestehende *Earcons* ein effektives Kommunikationsmittel zwischen Mensch und Maschine darstellen. Nonverbale akustische Anzeigen finden im Fahrzeug bei Warnhinweisen für plötzlich auftretende Ereignisse Anwendung. Dies ist beispielsweise das Überschreiten einer vom Fahrer eingestellten Geschwindigkeit, die Annäherung der Außentemperatur an den Gefrierpunkt oder der Abstand zum Hindernis bei Einparkhilfen. Da keine Blickzuwendung erforderlich ist, kann mit Hilfe dieser akustischen Anzeigen die Aufmerksamkeit des Benutzers wesentlich schneller gerichtet werden als bei visuellen Displays. Allerdings muss darauf geachtet werden, dass die Bedeutungen der verschiedenen Signale voneinander unterscheidbar bleiben und die Aufmerksamkeit des Fahrers nicht zu stark gebunden wird.

2.2.2.2 Visuelles Interface

Das traditionelle Medium für den Informationsfluss von der Maschine zum Menschen ist das visuelle Interface. Als technische Effektoren werden Displays verwendet. Die dargestellten Informationen werden über die Augen und das visuelle System vom Menschen aufgenommen. Aber auch der umgekehrte Weg ist technisch realisierbar. Hierbei werden Bewegungen des gesamten Körpers oder bestimmter Körperteile des Menschen erfasst und interpretiert. Die Erkennung erfolgt dabei meist über optische oder direkt am jeweiligen Körperteil befindliche mechanische Sensoren.

Gestenerkennung

Erkennungssysteme für Gesten leiten aus den Handbewegungen des Bedieners entsprechende Systemreaktionen ab. Ähnlich wie die Sprachbedienung besitzt diese Modalität den Vorteil, dass für ihre Bedienung kein spezielles Bedienelement erforderlich ist und dass sie aufgrund ihrer Natürlichkeit eine weitgehende intuitive Bedienung erlaubt. Die Interaktion des Benutzers mit dem System ist weitgehend ortsungebunden und kann deshalb bei Bedarf jederzeit beginnen. Insbesondere in VR[1]-Umgebungen stellt die Verwendung eines mit entsprechenden Sensoren bestückten Datenhandschuhs eine sinnvolle Art der Handbewegungserfassung dar. Diese Technik ist in anderen Domänen wie beispielsweise öffentlichen Kiosk-Systemen nicht möglich. Hier müssen berührungslose Erfassungssysteme wie beispielsweise das videobasierte SIVIT[2]-System [Siemens 1998] eingesetzt werden (Bild 9).

---

[1] VR: Virtual Reality

[2] SIVIT: Siemens Virtual Touchscreen

Bild 9 SIVIT – Siemens Virtual Touchscreen

Aber auch im Fahrzeug besitzt Gestenerkennung wegen der intuitiven und einfachen Bedienung großes Potenzial. Aus Komfort- und Sicherheitsgründen sind dabei berührungslose Erkennungssysteme erforderlich. [Akyol*, et al.* 2000] zeigen anhand eines gestengesteuerten Nachrichtenspeichers, wie mit statischen Handgesten einfache Navigationsfunktionen (zurück, vorwärts, abbrechen etc.) effektiv und schnell ausgeführt werden können. Ein nächster Schritt stellt die Erkennung dynamischer Handgesten [Morguet 2000] dar, um kontinuierliche Parameter wie die Lautstärke- oder die Temperatureinstellung sinnvoll verändern zu können.

Eye-Tracking/ Blickbewegungsmessung

Bei der Registrierung von Blickbewegungen (Eye-Tracking) werden die Fixationen im Blickfeld des Benutzers erfasst und daraus das jeweils angeblickte Objekt bestimmt. Zur technischen Realisierung gibt es drei grundsätzliche Methoden:

- videooptische Verfahren
- elektrophyisiologische Verfahren
- invasive Verfahren

Einen Überblick über die verschiedenen Erkennungstechnologien bietet hierzu [Jacob 1995]. Dabei wird gezeigt, wie die Blickbewegungsrohdaten semantisch interpretiert werden können. Daraus werden blickgesteuerte Interaktionstechniken für *WIMP*[1]-basierte HMIs abgeleitet, die eine Maus zur Bedienung obsolet machen.

Hierzu zeigt [Steudel 1999], dass solch eine optomotorische Dialogsteuerung im Fahrzeug bei geeigneten Aufgaben eine geringere kognitive Beanspruchung des Fahrers erzeugt als eine manuelle Bedienung. Eine weitaus größere Bedeutung besitzt Blickbewegungsmessung im

[1] WIMP: Windows Icons Menus Pointers. Bezeichnung für ein traditionelles fensterbasiertes HMI, wie es beispielsweise in Microsoft Windows realisiert ist.

Fahrzeug allerdings für die Bewertung der kognitiven Belastung des Fahrers durch Fahrerassistenz- und -informationssysteme [Partmann*, et al.* 1995] [Kopf & Hermann 1997] [Seifert*, et al.* 2001].

Lippenlesen

Die Analyse der Lippenbewegung ist ein optisches Verfahren zur Interpretation von sprachlichen Äußerungen. Das lippenlesende System extrahiert hierzu aus einer Videosequenz des Mundes eine Hypothese der Äußerung des Benutzers. Als eigenständiges System ist dieser Ansatz bislang nicht leistungsfähig genug. Er stellt aber aufgrund der starken Korrelation der Lippenbewegung mit dem Sprachsignal zusätzliche Informationen bereit, die insbesondere bei einem schlechten akustischen Signal die Fehlerkennungsrate eines Spracherkenners um mehr als die Hälfte reduzieren können [Bregler*, et al.* 1993].

Aufgrund der vorhandenen Störgeräusche wäre die Domäne Automobil prinzipiell ein ideales Anwendungsgebiet für diese Technologie. Allerdings ist nicht sichergestellt, dass der entstehende Nutzen den hohen Aufwand für die videooptische Erfassung rechtfertigt, insbesondere da solch ein System nicht unabhängig vom Bedienplatz funktioniert.

Graphische Anzeigen

Mit einer Datenrate von rund 30 MByte/s bietet das visuelle Interface dem Menschen die größte Bandbreite zur Informationsaufnahme [Lang 1994]. Deshalb benutzen die wichtigsten und am weitesten verbreiteten Wahrnehmungsmodalitäten der MMI diesen Kanal.

Als bedeutendste Vertreter sind hierbei graphische Anzeigen zu nennen. Die bekannteste Variante stellt hierbei die zwei-dimensionale Verwendung in den bereits genannten WIMP-HMIs aktueller Windowssysteme dar. Als Anzeigemedien dienen hierbei Kathodenstrahlmonitore (CRT[1]), Flüssigkristallanzeigen (LCD[2]) oder auch großflächige Projektionssysteme (Video-Beamer). Die noch in Entwicklung befindliche OLED[3]-Technologie verspricht hierbei für die Zukunft große Verbesserungen in Bezug auf Ablesbarkeit, Helligkeit, Leistungsaufnahme, Gewicht, Betriebstemperaturbereich, Designmöglichkeiten und Preis.

Für die Anwendung in VR-Umgebungen werden die Informationen mittels einer stereoskopischen Darstellung drei-dimensional präsentiert. Die getrennte Stimulierung des linken und rechten Auges kann dabei über Shutterbrillen und passend synchronisierten 2D-Anzeigen oder über Head-Mounted-Displays (HMD) realisiert werden. In Verbindung mit Trackingsystemen ist eine

---

[1] CRT: Cathode Ray Tube

[2] LCD: Liquid Crystal Display

[3] OLED: Organic Light Emitting Device

an die Körper- und Kopfbewegung gekoppelte Veränderung des Bildinhaltes möglich. Ein wichtiges Einsatzgebiet für solche Systeme ist die Visualisierung und Bewertung von CAD[1]-Daten.

Zur Darstellung von Karten und Richtungsinformationen hat mit den Navigationssystemen Anfang der 90er Jahre das graphikfähige Farbdisplay im Fahrzeug Einzug gehalten. Diese zunächst 5,5 Zoll großen Displays haben in der Mittelkonsole die Radioanzeige ersetzt und erweitert. Aber auch im Kombiinstrument finden freiprogrammierbare Displays zunehmend sinnvolle Anwendungen [Wagner*, et al.* 2001]. Mit Head-Up-Displays[2] (HUD) könnten in der nahen Zukunft Informationen sehr nah zur Fahrszene und fast ohne Umfokussierung des Auges dargestellt werden [Mutschler 1995]. Langfristig versprechen im Bereich der Mittelkonsole OLEDs, Projektionssysteme oder andere Darstellungstechnologien ästhetisch ansprechende Gestaltungsmöglichkeiten der dann nicht mehr notwendigerweise planaren Anzeigefläche. In Hinblick auf eine an die Fahrsituation angepasste dynamische Veränderung der dargestellten Informationsdichte besitzen diese Systeme großes Potenzial. Sehr gute Ablesbarkeit bei einer gleichzeitig sinnvollen Positionierung ist dabei Voraussetzung für den sicheren Einsatz dieser Systeme im Fahrzeug.

Alphanumerische Anzeigen

Zur einfachen Darstellung alphanumerischer Informationen, wie sie bei einer Vielzahl elektronischer Geräte realisiert ist, finden sich häufig sogenannte 7-Segment- und Punkt-Matrix-Displays. Insbesondere letztere werden im Fahrzeug im Bereich des Kombiinstruments, des Radios und der Klimasteuerung eingesetzt. Mit Einführung der oben genannten graphikfähigen Displays und der Integration der Einzelfunktionen in gesamtheitliche Bedienkonzepte werden diese Einzelanzeigen allerdings zunehmend überflüssig.

Sonstige Anzeigen

Es gibt ein breites Spektrum anderer Anzeigen, die über das visuelle Interface aufgenommen werden. Dazu gehören beispielsweise analoge Anzeigen, wie sie meist als Rundinstrumente oder Linearanzeigen für normalerweise physikalische Größen realisiert sind. Glühlampen oder LEDs[3] werden vorwiegend zur Anzeige des Aktivierungszustands oder als Warnhinweis benutzt. Rundinstrumente finden im Fahrzeug normalerweise für die Geschwindigkeitsanzeige und

---

[1] CAD: Computer Aided Design

[2] HUD: Head-Up-Display. Ca. 2.5m vor dem Fahrer schwebende virtuelle Anzeige, die mittels eines optischen Systems über die Windschutzscheibe in das Sichtfeld des Fahrers eingeblendet wird.

[3] LED: Light-Emitting Diode. Leuchtdiode

den Drehzahlmesser Verwendung. Kammerleuchten im Kombiinstrument und verschiedenste weitere optische Anzeigen werden für Informations- und Warnhinweise eingesetzt.

2.2.2.3 Taktiles Interface

Das taktile Interface wird hauptsächlich für die Informationsübermittlung vom Menschen zur Maschine verwendet. Es ist daher das klassische Gegenstück zum visuellen Interface. Die technischen Sensoren setzen dabei unter Ausnutzung verschiedenster physikalischer Effekte haptische Manipulationen des Benutzers an einem Bedienelement in Steuergrößen für den technischen Prozess des MMS um. Wird mit Hilfe von aktiven Elementen das somatosensorische[1] System des Menschen stimuliert, kann das taktile Interface auch als Wahrnehmungsmodalität eingesetzt werden.

Bedienelemente

Die Anforderungen an ein Bedienelement korrelieren im Allgemeinen stark mit der Art des zu steuernden technischen Prozesses. Es gibt daher eine Vielzahl manueller Eingabeelemente, die sich in der Anzahl, der Art und der Anordnung ihrer Freiheitsgrade stark unterscheiden. Tabelle 1 beinhaltet hierzu eine knappe Gegenüberstellung wichtiger Bedienelemente, die anhand ihrer Bewegungsarten und Bedienmöglichkeiten unterschieden werden. [Boff & Lincoln 1988] ordnen diese bestimmten Aufgabentypen zu und stellen darüber hinaus die verschiedenen Aspekte dar, die bei der Auswahl und Positionierung von Bedienelementen berücksichtigt werden müssen.

Tabelle 1 Klassifikation von Bedienelementen [Boff & Lincoln 1988]

	Diskrete Bedienung	Kontinuierliche Bedienung
Linearbewegung	Drucktaster, Kippschalter, Schiebeschalter, Wippschalter, Druck-Zug-Schalter	Schieberegler, Maus, Analogjoystick, Trackball, Lichtgriffel, Touchpad, Graphiktablett, Touchscreen
Drehbewegung	Drehwahlschalter, Drehsteller, Dekadenschalter, Schlüsselschalter	Drehregler, Einstellrad

Für die MMI mit einem Rechner ist die aus einer Gruppe von Drucktastern bestehende Tastatur nach wie vor das wichtigste Eingabeverfahren. Mit ihr werden Informationen als alphanumerische Zeichen kodiert diskret eingegeben. Mit Hilfe von Maus, Trackball, Graphiktablett, Touchscreen etc. ist dagegen eine orts- oder koordinatenbasierte Kodierung der Informationen mög-

[1] somatosensorisch: Die Körpersinne, das heißt die mechanischen, thermischen und schmerzempfindlichen Rezeptionsfähigkeiten des Menschen betreffend.

lich. Letztere arbeiten entweder mit direkt am Bildschirm dargestellten virtuellen Bedienelementen oder benötigen zur semantischen Interpretation, wie beispielsweise bei der Handschrifterkennung, zusätzliche Erkennungssysteme [Waibel*, et al.* 1995] [Winkler & Lang 1997].

Zur sinnvollen Kontrolle der vorhandenen Funktionen werden im Fahrzeug die unterschiedlichsten diskret und kontinuierlich bedienbaren Eingabeelemente eingebaut. Aufgrund der meist sehr hohen Zahl an Freiheitsgraden wird der Lenkstockhebel hier besonders hervorgehoben (Bild 10).

Quelle: BMW

Bild 10 Komplexes Bedienelement Lenkstockhebel

Aber auch neuere Technologien wie beispielsweise Touchscreens werden in Fahrzeuge integriert. Touchpads besitzen in Verbindung mit Handschrifterkennung großes Potenzial, da sie eine intuitive und schnelle Eingabe von Text erlauben. Eine solche komfortable Texteingabemöglichkeit ist insbesondere in Hinblick auf onlinefähige Bürofunktionalitäten, wie sie beispielsweise in Form von Unified Messaging Systemen[1] (UM) und Personal Information Manager[2] (PIM) in Zukunft im Fahrzeug zur Verfügung stehen könnten, besonders wichtig.

Aktive Bedienelemente – Force-Feedback

Die aktive Übermittlung von Informationen über das taktile Interface von der Maschine zum Menschen wird als *Force-Feedback* bezeichnet. Hierbei werden mittels technischer Aktuatoren die Muskeln und Sehnen des Menschen zur Verdeutlichung der aktuellen Dialogsituation mit mechanischen Kräften beaufschlagt. In VR-Umgebungen und insbesondere bei Computerspielen vermitteln Joysticks, Lenkräder und Mäuse mit Force-Feedback einen realitätsnäheren Eindruck.

Da aktive Bedienelemente im Fahrzeug weder eine visuelle noch eine auditive Belastung des Fahrers erzeugen, werden ihre Einsatzmöglichkeiten bereits seit längerem für Fahrerassistenz-

---

1 Unified Messaging: UM-Services fassen gängige Nachrichtendienste wie Telefax, E-Mail, SMS oder Voice-Mail über eine einzige Schnittstelle zusammen [Bager 2001].

2 Personal Information Manager: Als PDA- oder Desktop-Anwendungen bzw. als Web-Dienste verfügbare Termin-, Aufgaben- und Adressverwaltungsprogramme. Die darin normalerweise enthaltenen Netzwerkfunktionen erlauben eine effiziente Gruppenzusammenarbeit.

systeme untersucht, die Einfluss auf die Longitudinal- oder Lateralführung nehmen [Naab & Reichart 1994]. Dem Fahrer wird dabei mittels Gegendruck am Gaspedal eine Geschwindigkeit empfohlen, bzw. er wird durch entsprechende Lenkradmomente vor dem Verlassen der aktuellen Fahrspur gewarnt [Schraut*, et al.* 2000].

Daneben erleichtern aktive Bedienelemente, wie der Controller im BMW iDrive [Zeller*, et al.* 2001], die sonst weitgehend visuelle Rückkopplung der Dialogsituation zum Fahrer und vereinfachen dadurch die Bedienung komplexer Infotainmentsysteme (Bild 11).

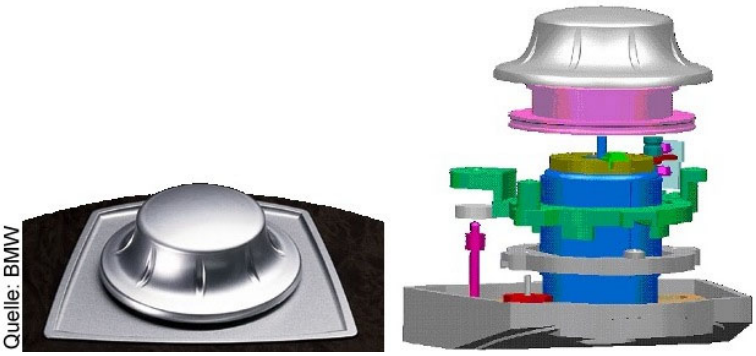


Bild 11 Aktives Bedienelement: iDrive-Controller

2.2.3 Konzeptuelle Architektur eines multimodalen MMS

Die Zusammenführung verschiedener Modalitäten zu einem multimodalen HMI und die Kontrolle der zugehörigen Applikation ist eine komplexe Aufgabe. Erschwerend wirkt sich dabei aus, dass die Methoden möglichst allgemein anwendbar sein sollen, damit zukünftige Anwendungen möglichst effizient und mit einem hohen Wiederverwendbarkeitsgrad realisiert werden können.

Mit der im Folgenden dargestellten konzeptuellen Architektur werden die verschiedenen Aspekte eines multimodalen HMI aufgezeigt und in ihren logischen Zusammenhang gebracht. Obwohl es, wie unten beschrieben, verschiedene Lösungsansätze für die einzelnen Teilprobleme gibt, sind die jeweils zugrunde liegenden Architekturen meist ähnlich aufgebaut. Sie lassen sich im Kern auf das Seeheim-Modell[1] [Green 1985] zurückführen. Dieses postuliert für das HMI eine Aufteilung in drei Schichten sowie eine strenge Trennung von der Applikationsebene (Bild 12). Dadurch kann ein möglichst großer Grad an Wiederverwendbarkeit und Automatisierung erreicht werden.

[1] Seeheim-Modell: Basisarchitektur für graphische Benutzeroberflächen, die 1983 auf einem Workshop in Seeheim, Deutschland, erarbeitet wurde.

- Präsentation
  Statische Aspekte der Bedienoberfläche. Festlegung des Layouts. Interaktion mit den Ein- und Ausgabegeräten. Über Standardbibliotheken weitgehend automatisch realisierbar.

- Dialogmanagement
  Dynamischer Teil der Dialogführung. Festlegung der Dialogstruktur. Kontrolle des Informationsflusses zwischen Präsentations- und Applikationsschicht.

- Applikationsinterface
  Semantisches Modell der Applikationsfunktionalität. Datenaufbereitung und Interaktion mit der Applikation.

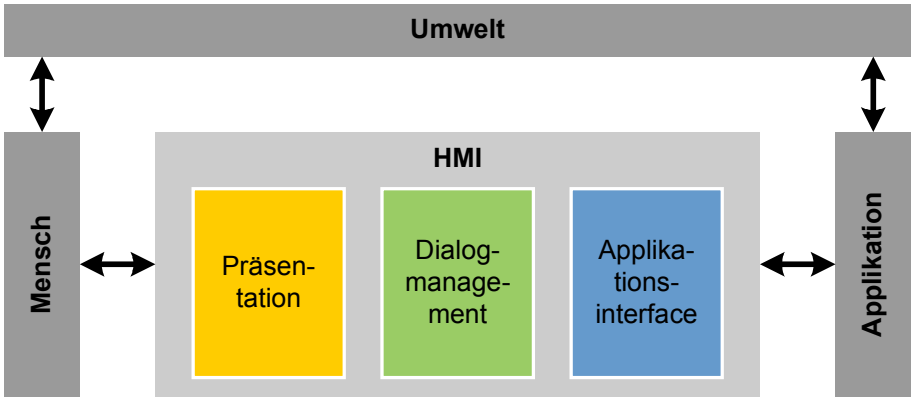


Bild 12 Seeheim-Modell

Die starre Funktionalität graphischer Benutzeroberflächen lässt sich mit dem Seeheim-Modell abbilden. Um allerdings multimodale Bedienkonzepte darstellen zu können, müssen eine Reihe weiterer fundamentaler Aspekte [Maybury & Stock 1999] einbezogen werden, sodass eine Erweiterung des Modells erforderlich ist:

- *Analyse* der häufig ungenauen und mehrdeutigen multimodalen Benutzereingaben, die sowohl synchron, als auch asynchron erfolgen können.

- *Generierung* der multimodalen Systemausgaben in einer koordinierten, kohäsiven und kohärenten Art und Weise.

- *Dialogmanagement* zur Sicherstellung einer effizienten, effektiven und natürlichen Interaktion. Dies sind Trainingsmethoden, Fehlermanagement, Aufgabenbearbeitung und maßgeschneiderte Interaktionsstile. Hierzu müssen unter Umständen notwendige Modelle der Domäne, der Aufgabe, des Nutzers, der Modalitäten oder des Kontextes erstellt werden.

- *Methoden* zur Repräsentation und Schlussfolgerung, als Basis für die oben genannten Forderungen. Insbesondere für ein sinnvolles Dialogmanagement müssen Bewertungsmaße und Bewertungsmethoden bestimmt werden.

Wie Bild 13 zeigt, sind im Bereich der Präsentations- und Dialogmanagementschicht die größten Modifikationen notwendig. Damit die verschiedenen *Aktions*- und *Wahrnehmungsmodalitäten* in einer modularen Weise integriert werden können, ist es nicht mehr sinnvoll, die Interaktion mit dem Benutzer durch eine einzige Komponente abzubilden. Vielmehr wird jede einzelne Modalität als separates Modul betrachtet.

- Um nun aber die von den jeweiligen Modalitäten erkannten Nutzereingaben sinnvoll interpretieren zu können, müssen sie mit einer *Fusionskomponente* zusammengeführt und auf ein gemeinsames semantisches Niveau gebracht werden. Hierzu werden in 2.2.3.1 einige wichtige Informationsfusionsansätze vorgestellt.
- Aber auch im umgekehrten Fall bei der Informationsausgabe durch das HMI ist eine ähnliche Komponente erforderlich. Das *Fissionsmodul* plant die Ausgaben und verteilt die Informationen auf die verschiedenen Wahrnehmungsmodalitäten des Nutzers. Dies ist insbesondere dann wichtig, wenn je nach Situation oder Umfeld unterschiedliche Ausgabemöglichkeiten zur Verfügung stehen.
- Mit Hilfe des *Dialogmanagements* wird der Dialogablauf festgelegt und kontrolliert. Es stellt somit, analog zum Seeheim-Modell, den dynamischen Teil der Interaktion dar. In 2.2.3.2 werden hierzu verschiedene Möglichkeiten zur Beschreibung des Bediendialogs gegenübergestellt.
- Das *Applikationsinterface* stellt die semantische Verbindung zwischen HMI und Applikation her. Es ist für den bidirektionalen Datenaustausch zuständig und triggert Ereignisse bzw. löst Funktionen aus.
- Für eine *sinnvolle* und *natürliche* Interaktion ist es wichtig, dass sich multimodale HMIs an unterschiedliche Randbedingungen adaptieren können. Hierzu ist entsprechendes *Kontextwissen* erforderlich, das den verschiedenen Komponenten bei Bedarf zur Verfügung stehen muss.

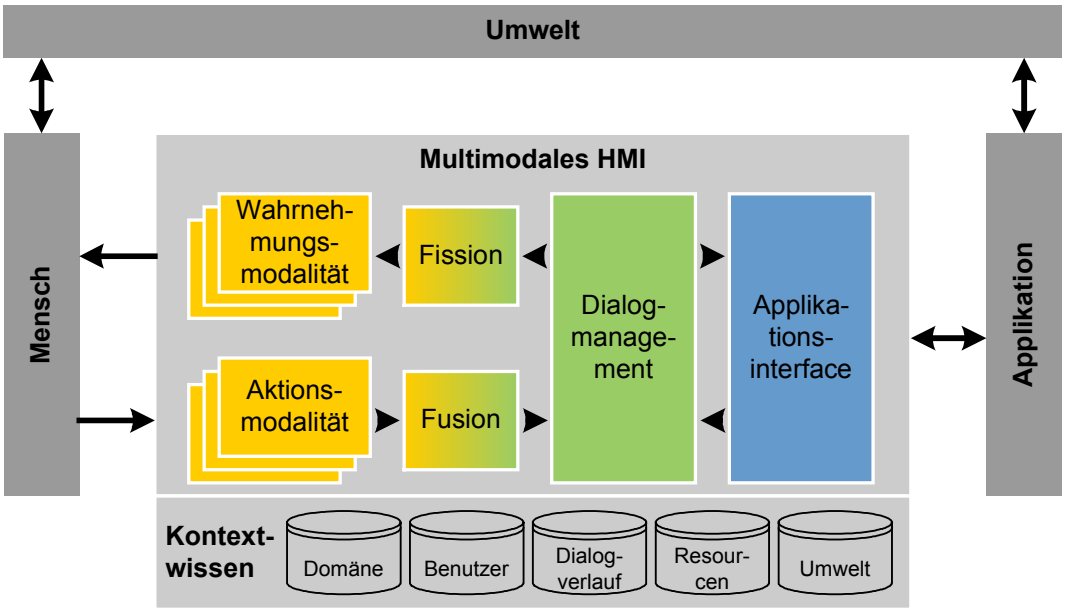


Bild 13 Konzeptuelle Architektur eines multimodalen MMS

Mit entsprechenden Modifikationen und Erweiterungen lässt sich die in Bild 13 dargestellte konzeptuelle Architektur in vielen Arbeiten und Projekten wieder finden. Beispielhaft hierfür sind die Projekte Smartkom [Wahlster, *et al.* 2001] und Embassi [Herfet & Kirste 2001]. Aber auch in agentenbasierten Architekturen wie PAC-Amodeus [Coutaz, *et al.* 1995a] und OAA[1] [Moran, *et al.* 1998] lassen sich die entsprechenden Komponenten identifizieren.

Detaillierte Darstellungen der spezifischen Eigenschaften verschiedener Softwarearchitekturen finden sich in [Coutaz 1993] [Evers 1999] [Kirste & Rapp 2001].

2.2.3.1 Informationsfusion

Einer der kritischsten Aspekte multimodaler Systeme ist die Zusammenführung der Informationen aus den einzelnen Modalitäten auf ein gemeinsames semantisches Niveau und die hierauf basierende gesamtheitliche Interpretation. Nach [Gourdol, *et al.* 1992] [Hall & Llina 1997] kann die Informationsfusion prinzipiell in drei unterschiedlichen Ebenen erfolgen. Je nachdem ob die Vereinigung auf Basis von Sensordaten, Erkennermerkmalen oder Erkennungsergebnissen erfolgt, werden sie von Gourdol als *Lexical Fusion*, *Syntactic Fusion* und *Semantic Fusion* bezeichnet bzw. von Hall als *Data Fusion*, *Feature Fusion* und *Decision Fusion* (Bild 14).

[1] OAA: Open Agent Architecture

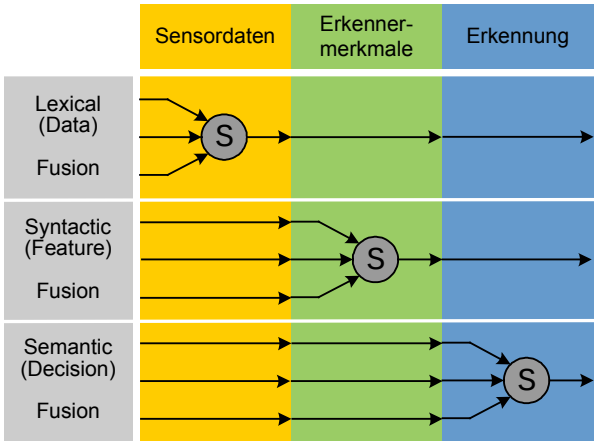


Bild 14 Mögliche Ebenen der Datenfusion [Gourdol, *et al.* 1992] [Hall & Llina 1997]

Da die Sensorsignale verschiedener Modalitäten, wie beispielsweise einer Audioquelle und eines manuellen Bedienelements, im Allgemeinen sehr unterschiedlich sind, macht eine Lexikalische Fusion meist wenig Sinn. Eine Syntaktische Fusion, bei der zunächst Merkmale aus verschiedenen Informationsquellen extrahiert und dann in einem gemeinsamen Erkennungsprozess interpretiert werden, ist nur dann anwendbar, wenn die beiden Merkmale stark miteinander korreliert sind. Als Beispiel sind hier audio-visuelle Erkennungssysteme zu nennen, bei denen Lippenbewegungen und Sprachsignale gemeinsam zur Erkennung einer Äußerung verwendet werden [Bregler, *et al.* 1993] [Movellan & Mineiro 1996]. Um beliebige Modalitäten miteinander kombinieren zu können, werden Eingaben normalerweise erst auf der Bedeutungsebene vereinigt. Dazu erzeugt man zunächst mit separaten Erkennungsprozessen, für jede Modalität getrennt, eine semantische Interpretation der jeweiligen Eingabe. Anschließend werden die verschiedenen Teilinformationen zusammengeführt und daraus eine gesamtheitliche Interpretation der Benutzerintention generiert. Für diese Semantische Fusion sind eine Reihe an Vorgehensweisen bekannt:

Unification [Johnston, *et al.* 1997]

Die von den verschiedenen Modalitäten erkannten Nutzereingaben werden jeweils in *Typed-Feature-Structures* abgelegt. Diese werden dann mit Hilfe eines speziellen Vereinigungsprozesses (Unification) auf ihre Konsistenz geprüft und dann zu einer Interpretation kombiniert. Durch die Verfügbarkeit von Zusatzinformationen wie die Eingabemodalität, die Erkennungssicherheit und des Dialogzustandes stellen *Multidimensional Feature Structures* [Denecke & Yang 2000] eine Erweiterung dieses Ansatzes dar.

Melting Pot [Nigay & Coutaz 1993]

Nutzereingaben aus den einzelnen Modalitäten werden abstrahiert und in *Melting-Pot-Objects* abgelegt. Diese 2-dimensionalen Objekte spiegeln die strukturellen Teile eines Kommandos und ihre zeitlichen Beziehungen wider. Die so codierten Teilinformationen werden dann durch hierarchisch angeordnete Softwareagenten in einem inkrementellen Prozess zur Gesamtinformation zusammengeführt.

MS-MIN: Multi-State Mutual Information Network [Vo 1998]

Die semantischen Teilinterpretationen der einzelnen Modalitäten werden in *Parameter Slots* angeordnet. Diese stellen die Eingabesequenz des MS-MIN dar, welches die Eingaben mit einem Satz an Prototypen vergleicht und mittels eines Bayes´schen Klassifikators diejenige Gesamtinterpretation mit der *maximum a posteriori* Wahrscheinlichkeit berechnet.

Guided Propagation Network [Martin 1998]

Sobald eine Nutzereingabe von einer Modalität erkannt wird, aktiviert sie den zugehörigen Ereignisdetektor im *Guided Propagation Network*. Diese Ereignisdetektoren sind mit multimodalen Einheiten verbunden, an die sie bei Aktivierung Signale versenden. Diese multimodalen Einheiten akkumulieren die jeweils eintreffenden Signale und lösen bei Überschreitung einer Aktivierungsschwelle eine zugehörige Funktion aus.

Diese Verfahren sind geeignet hochparallele multimodale Eingaben, wie sie das Put-That-There-Szenario [Bolt 1980] beschreibt, zu bearbeiten. Wie in 2.2.6 gezeigt wird, besitzt dieser Multimodalitätstyp aber in der Domäne Fahrzeug keine große Relevanz. Daher wird in dieser Arbeit ein nicht ganz so mächtiges Verfahren angewandt (3.4), das aber auf die fahrzeugspezifischen Anforderungen eingeht, leichter anwendbar und vor allem im Bereich der Dialogqualitätssicherung kontrollierbarer ist.

2.2.3.2 Dialogmanagement – Beschreibung und Steuerung des Dialogs

Das Herzstück eines HMI ist die Dialogmanagement-Komponente, da sie den Interaktionsablauf systemseitig kontrolliert. Zur Definition des Mensch-Maschine-Dialogs gibt es verschiedene formale Methoden. [Geiser 1990] [Churcher*, et al.* 1997] [Cohen 1997] bieten hierzu detaillierte Überblicke. Im Folgenden werden die zwei wichtigsten Methoden dargestellt:

Zustandsübergangsnetzwerke

Ein sehr gebräuchliches Verfahren ist die Modellierung des Dialogs als Zustandsübergangsnetzwerk. Dabei wird der Mensch-Maschine-Dialog in Zustände unterteilt, die durch Übergänge miteinander verbunden sind. Auslöser für diese Übergänge können beispielsweise Nutzereingaben oder sonstige Ereignisse in den Applikationen bzw. im Dialogmanagement sein.

Eine klassische Beschreibungsmöglichkeit stellen *endliche Automaten* dar [Hopcroft & Ullman 1990]. Dabei wird das zu beschreibende System in eine endliche Zahl an Zuständen unterteilt. In jedem dieser Zustände werden bestimmte diskrete Eingaben verarbeitet, die wiederum in diskrete Ausgaben des Systems resultieren. Endliche Automaten sind daher besonders gut geeignet zeitliches Verhalten zu modellieren. Gleichzeitig kann die beschriebene Funktionalität aufgrund der visuellen Repräsentation in Form von Kreisen und Pfeilen sehr einfach erfasst werden. Bei großen Systemen mit vielen Zuständen und Übergängen steigt die Komplexität der Beschreibung allerdings sehr schnell an. Um dem entgegenzuwirken wurden eine Reihe von Erweiterungen vorgenommen, die zum einen eine Strukturierung und einen rekursiven Aufruf

der Beschreibung erlauben (RTN[1]/ SDL[2]) und zum anderen Register und Funktionen zur gezielten Steuerung der Übergänge (ATN[3]/ GTN[4]) beinhalten. *Petri-Netze* ermöglichen aufgrund ihrer Struktur die effiziente Beschreibung hochparalleler Systeme. Mit *Statecharts* [Harel 1987] kann wegen der hierarchischen Strukturierungsmöglichkeit des Zustandsautomaten der Komplexitätsanstieg relativ flach gehalten werden. Dies erleichtert die Spezifikation größerer Systeme.

Des Weiteren sind *Sprechakte* (Speech Acts) zu nennen [Hagen & Popowich 2000] [Nuance 2001]. Diese Technik kommt aus dem Bereich reiner Sprachdialogsysteme, bei denen aufgrund der umfangreichen Ausdrucksmöglichkeiten der menschlichen Sprache die Komplexität der Beschreibung sehr hoch ist. Hierbei wird der Dialog durch einzelne Sprechakte repräsentiert, die ganze Interaktionssequenzen, wie beispielsweise die Eingabe einer Telefonnummer, inklusive aller Fehler- und Korrekturaufgaben beschreiben. Der Dialogverlauf wird dann durch eine Verknüpfung dieser Sprechakte modelliert. Im Sinne der Zustandsübergangsnetzwerke kann ein Sprechakt als einzelner „Superzustand“ mit einem sehr komplexen Verhalten aufgefasst werden. Dieser hat auf die übrigen Zustände keine Auswirkungen, wodurch der Komplexitätsanstieg eingedämmt werden kann. Da ein Sprechakt ein recht komplexes Gebilde darstellt, müssen bei der Entwicklung bereits alle Modalitäten bekannt sein. Die Integration zusätzlicher Modalitäten ist deshalb sehr aufwändig. Sprechakte finden daher in hauptsächlich sprachbasierten Systemen Anwendung.

Formale Grammatiken

Eine weitere Möglichkeit zur Dialogbeschreibung kommt aus dem Bereich der Linguistik. Mit Hilfe *formaler Grammatiken* wird die Struktur der „Sprache“ zwischen Mensch und Maschine durch Regeln beschrieben, welche die Konstruktion jeder zulässigen Äußerung erlauben. Eine Regel ordnet einem nichtterminalen Symbol ($\langle S \rangle$) eine Menge an weiteren nichtterminalen ($\langle X \rangle$, $\langle Y \rangle$) bzw. terminalen ('a', 'b', 'y', 'z') Symbolen zu. Durch rekursive Ersetzung der nichtterminalen Symbole auf der rechten Seite, können die jeweils möglichen Sätze der Grammatik erzeugt werden, die durch diese Regel beschrieben werden (Bild 15). Die bekannteste Darstellungsform ist hier die Beschreibung mittels *Backus-Naur-Form* (BNF).

---

[1] RTN: Recursive Transition Network

[2] SDL: Specification and Description Language

[3] ATN: Augmented Transition Network

[4] GTN: Generalized Transition Network

$$\left.\begin{array}{l}\langle S\rangle ::= \langle X\rangle\ \langle Y\rangle \\ \langle X\rangle ::= \text{'}y\text{'} \mid \text{'}z\text{'} \\ \langle Y\rangle ::= \text{'a'} \mid \text{'b'}\end{array}\right\} \quad < S > ::= \text{'ya'} \mid \text{'yb'} \mid \text{'za'} \mid \text{'zb'}$$

Bild 15 Mögliche Sätze einer einfachen Grammatik in BNF

Eine Verallgemeinerung dieser Darstellung stellen *Produktionssysteme* dar. Hier wird der Mensch-Maschine-Dialog in Form von Bedingungs-Aktions-Paaren beschrieben:

WENN (Bedingung), **DANN** (Aktion)

Als Entscheidungsgrundlage für die Bedingungen steht in Produktionssystemen ein Arbeitsspeicher zur Verfügung, der ein Abbild der aktuellen Dialogsituation in Form von Zielen, Aktionen und Nutzereingaben bereitstellt.

Die mangelnde Fähigkeit zeitliches Verhalten zu modellieren erschwert den Einsatz formaler Grammatiken zur Spezifikation umfangreicher Bedienabläufe. Für die Bedienbarkeit eines MMS im Fahrzeug ist aber genau dies ein essentieller Aspekt. Aus diesem Grund wird in dieser Arbeit zur Beschreibung des Mensch-Maschine-Dialogs ein Zustandsübergangsnetzwerk verwendet. Wie in Kapitel 3 dargestellt ist, erfolgt die Dialogbeschreibung in Anlehnung an Statecharts in Form eines hierarchischen Zustandsautomaten, wobei auf spezielle Anforderungen der Mensch-Maschine-Interaktion eingegangen wird.

2.2.4 Systematisierung und Klassifikationsschemata multimodaler Systeme

Es gibt eine große Zahl multimodal bedienbarer technischer Systeme (2.2.5). Allerdings weisen die Realisierungen große Unterschiede bezüglich des Zusammenspiels der einzelnen Modalitäten und des damit verbundenen technologischen Aufwands auf. Um für die Domäne Fahrzeug angepasste Werkzeuge gestalten zu können, muss zunächst geklärt werden, wie die verschiedenen Modalitäten im Automobil zusammenwirken und welche Art von Multimodalität zu erwarten ist. Hierzu ist ein Verfahren notwendig, mit dem die Eigenschaften eines multimodalen Systems erfasst werden können und das es erlaubt, verschiedene Konzepte miteinander zu vergleichen. In der Literatur sind dazu verschiedene Klassifikationsansätze beschrieben:

[Martin 1997] schlägt sechs grundlegende *„types of cooperation“* zwischen den einzelnen Modalitäten vor. Komplementärität und Redundanz werden dabei unter dem Überbegriff *Fusion* als diejenigen Typen zusammengefasst, bei denen Informationen aus unterschiedlichen Modalitäten gemeinsam interpretiert werden:

- *Äquivalenz* (Equivalence)
  Die verschiedenen Modalitäten sind zueinander gleichwertig, d.h. Eingaben werden entweder von der einen oder der anderen bearbeitet.

- *Spezialisierung* (Specialization)
  Eine bestimmt Eingabe kann immer nur mit einer spezifischen Modalität erledigt werden.

- *Redundanz* (Redundancy)
  Der Benutzer gibt unter Umständen identische Informationen über verschiedene Modalitäten ab. Erhält das System übereinstimmende Informationen aus unterschiedlichen Modalitäten, so betrachtet es dies als eine sehr gute Erkennung der Nutzerintention. Es kann dadurch die weitere Dialogführung entsprechend sicher gestalten.

- *Komplementärität* (Complementarity)
  Bruchstücke der Gesamtinformation können über verschiedene Modalitäten abgegeben werden. Diese Teilinformationen müssen zusammengefügt werden und können nur als Ganzes interpretiert werden.

- *Transfer* (Transfer)
  Informationen aus einer Modalität werden von einer anderen Modalität benutzt, um beispielsweise die Erkennungssicherheit zu erhöhen.

- *Simultanität* (Concurrency)
  Mit verschiedenen Modalitäten können unterschiedliche Informationen gleichzeitig eingegeben werden. Die Nutzerintentionen werden unabhängig voneinander interpretiert. Dadurch können mit unterschiedlichen Modalitäten verschiedene Funktionen parallel bedient werden.

[Coutaz*, et al.* 1995b] charakterisieren mit den *CARE-Properties* multimodale Systeme nach ähnlichen Kriterien: *Äquivalenz* (Equivalence), *Zuordnung* (Assignment) = Spezialisierung, *Redundanz* (Redundancy), *Komplementärität* (Complementarity). Ein interessanter Punkt ist hier, dass der Multimodalitätstyp nicht nur vom technischen Standpunkt aus definiert, sondern mit den *U-CARE-Properties* auch die tatsächliche Bedienung durch den Nutzer reflektiert wird. Dadurch sind systematische Aussagen zur präferierten Modalität in der jeweiligen Dialogsituation möglich.

[Neuss 2000] unterteilt Multimodalität in vier Ausprägungen:

- Seriell-redundant
  Für eine Folge von Dialogschritten kann die Aktionsmodalität beliebig oft gewechselt werden. Teilinformationen werden aber jeweils nur mit einer Modalität eingegeben.

- Seriell-exklusiv
  Es gibt zwar mehrere Modalitäten, für eine bestimmte Interaktionssequenz ist die Modalität aber festgelegt.

- Parallel ergänzend
  Simultan getätigte Eingaben mit verschiedenen Modalitäten ergänzen sich.
- Parallel verifizierend
  Simultane Eingaben mit verschiedenen Modalitäten bestätigen sich gegenseitig.

[Nigay & Coutaz 1993] klassifizieren multimodale Systeme anhand des möglichen zeitlichen Gebrauchs der Modalitäten zueinander, der Vorgehensweise, wie Ein- und Ausgaben über die verschiedenen Modalitäten zur Gesamtinformation fusioniert werden und des Abstraktionsniveaus, auf dem Informationen aus den Modalitäten interpretiert werden. Da letzteres normalerweise auf der semantischen Ebene erfolgt, wird es zur Vereinfachung in den folgenden Betrachtungen weggelassen. Die beiden anderen Dimensionen spannen eine Fläche auf, in die das zu untersuchte System eingeordnet wird. Je nach Quadrant wird das System einem Multimodalitätstyp zugeordnet (Bild 16):

- *Synergistisch* (Synergistic)
  Teilinformationen werden parallel mit allen Modalitäten eingegeben und dann zu einer Gesamtinterpretation semantisch miteinander verbunden.
- *Simultan* (Concurrent)
  Informationen können parallel eingegeben werden. Sie werden aber unabhängig voneinander interpretiert und können somit zur parallelen Bedienung verschiedener Funktionen eingesetzt werden.
- *Exklusiv* (Exclusive)
  Informationen werden lediglich sequentiell verarbeitet, d.h. es kann zu einem Zeitpunkt immer nur mit einer Modalität bedient werden. Wenn der Benutzer die Modalität wechselt, werden die Informationen unabhängig voneinander verarbeitet.
- *Alternierend* (Alternate)
  Die verschiedenen Modalitäten können nur sequentiell verwendet werden. Die Eingaben werden verbunden und insgesamt interpretiert.

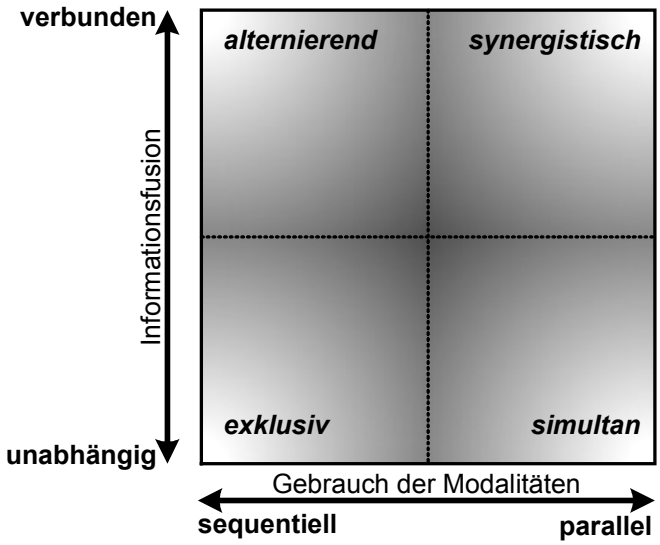


Bild 16 Klassifikationsschema für multimodale Systeme [Nigay & Coutaz 1993]

Die oben vorgestellten Klassifizierungsverfahren überschneiden sich in vielen Punkten und weisen daher teilweise starke Ähnlichkeiten auf. Die zuletzt vorgestellte Methode [Nigay & Coutaz 1993] wird im Folgenden verwendet, da der vorgeschlagene analytische Ansatz auf eine sehr anschauliche Art und Weise den Vergleich verschiedener Systeme erlaubt.

2.2.5 Anwendungen

Die nachfolgend dargestellten Beispiele geben einen Überblick über die Einsatzmöglichkeiten multimodaler Systeme in verschiedenen Domänen. Tabelle 2 enthält Auskunftssysteme, Office-Anwendungen, Assistenzsysteme und Anwendungen zur Manipulation von graphischen Darstellungen. Tabelle 3 umfasst Bediensysteme aus dem Fahrzeugkontext.

Tabelle 2 Multimodal bedienbare MMS

Nr	System	Beschreibung
1	*Put-That-There*	[Bolt 1980]: Manipulation von geometrischen Objekten auf einer Leinwandprojektion. Sprachkommandos werden durch handbasierte Zeigegesten unterstützt.
2	*Cubricon*	[Neal & Shapiro 1991]: Mausbasierte Zeigegesten und natürlichsprachliche Eingaben mittels Spracherkennung oder Tastatur zur Kontrolle eines interaktiven Kartensystems im militärischen Kontext. Systemausgaben erfolgen aufeinander abgestimmt über graphische Displays und per Sprache.

3	*Voice-Paint Notebook*	[Gourdol*, et al.* 1992]: Bedienung eines Malprogramms bzw. elektronischen Notizblocks per Spracherkennung, mausbasierten Zeigegesten und Tastatur. Visuelles Feedback erfolgt über ein graphisches Display.
4	*MATIS*	[Nigay*, et al.* 1993]: Per Sprache, Maus und Tastatur bedienbares Flugplanauskunftssystem mit einer graphischen Anzeige. Aufgrund der multithreading-Funktionalität kann der Nutzer mehrere Anfragen parallel bearbeiten.
5		[Faure & Julia 1994]: Stiftbasierte Gestenerkennung in Verbindung mit Spracherkennung. Vom Nutzer eingegebene Skizzen werden in entsprechende graphische Objekte übersetzt. Ausgaben erfolgen an einem graphischen Display.
6	*QuickSet*	[Cohen*, et al.* 1997]: Kontrolle einer militärischen Strategiesimulation mittels Stifteingabe auf einem HandHeld-PC in Verbindung mit Spracheingabe. Graphische Ausgaben am HandHeld und auf einer Leinwandprojektion sowie Sprachrückmeldungen dienen zur Informationsanzeige.
7	*VoiceLog*	[Bers*, et al.* 1997]: Online-Abfrage von Fahrzeugblaupausen und Bestellsystem für Fahrzeugteile per Sprache und stiftbasierten Gesten. Die visuelle Darstellung erfolgt in einem Web-Browser.
8	*HOME*	[Bekiaris*, et al.* 1997]: Auf Ältere und Behinderte abgestimmtes Bedienkonzept für Haushaltsgeräte. Zur Informationseingabe sind freie Sprachäußerungen, Touchscreen und Handgesten vorgesehen. Systemausgaben werden per Sprache oder als graphische Anzeigen realisiert werden.
9	*Talky*	[Streit 1999]: Ein elektronischer Kalender und Zeitplaner, der per Spracheingabe, Tastatur und mausbasierten Zeigegesten bedienbar ist. *Visual Utterances* – aus einer formalen graphischen Struktur, Text und einem abstrahierten menschlichen Gesicht (Smiley) bestehende visuelle Anzeigen – dienen zur Informationsanzeige.
10	*MSVT Roomdesigner*	[LaViola 1999]: VR-Visualisierungswerkzeuge zur Strömungssimulation bzw. zur Planung der Einrichtung eines Raumes. Der Arbeitsbereich ist jeweils per Sprache und Handgesten manipulierbar. Letztere werden mit einem Datenhandschuh erfasst. Die graphische Darstellung ist dreidimensional und als Rückprojektion realisiert.

12	*SmartKom*	[Wahlster*, et al.* 2001]: Kommunikations-, Informations- und Kontrollsystem, das Sprache, Gestik und Gesichtsausdruck sowohl als Aktions-, als auch Wahrnehmungsmodalitäten erlaubt. Je nach Anwendungsszenario wird ein SIVIT (Smartkom Public), ein PDA (Smartkom Mobile) oder ein Webpad (Smartkom Home/ Office) als Endgerät eingesetzt.
13	*Embassi*	[Herfet & Kirste 2001]: Bedien- und Servicesystem zur Unterstützung des Menschen in der heimischen Umgebung. In den drei Szenarien Living Room, Automotive und Terminal Access werden verschiedenste Modalitäten eingesetzt um die Bedienung von Haushaltsgeräten und die Kontrolle der Umgebungsbedingungen so angenehm und einfach wie möglich zu gestalten. Im Szenario Terminal Access wird insbesondere auf die Anforderungen behinderter Menschen eingegangen.

Tabelle 3 Multimodalität in der Domäne Fahrzeug

Nr	System	Beschreibung
14	*Bordmonitor und Sprachsteuerung* - Seriensystem -	[BMW 1998a] [BMW 1998b]: Radio, Navigation, Telefon und TV werden über einen Dreh-Drücksteller am Bordmonitor bedient. Davon entkoppelt können per Sprache Telefon- und Navigationsfunktionalitäten sowie eine Notizfunktion kontrolliert werden.
15	*Comand und Linguatronic* - Seriensystem -	[Daimler-Benz 1998]: Informations- und Bedienungszentrale für diverse Infotainmentfunktionen wie Radio, CD-Spieler und weitere optionale Ausstattungen wie z.B. CD-Wechsler, Telefon usw. Davon unabhängig können bestimmte Funktionen des Telefons und der Audiosysteme per Sprache bedient werden.
16	*MoTiV-MMI*	[Bengler*, et al.* 2000]: Integration von Sprache und manueller Bedienung über Lenkradtasten in ein gemeinsames Konzept zur Steuerung eines Radios und eines Navigationssystems. Systemausgaben sind am Bordmonitor und per TTS möglich.
17	*CarMMI*	[Neuss 2000]: Sprache, manuelle Bedienung und eine simulierte Gestenerkennung (WoZ) zur Steuerung von Infotainmentfunktionen. Das Systemfeedback erfolgt am Bordmonitor und über Sprachausgaben.

18		[Schattenberg & Debus 2001]: Abbildung des Funktionsumfangs der Mercedes S-Klasse in einem zentralen Anzeige- und Bedienkonzept. Zur Eingabe stehen eine spezielle Bedieneinheit sowie Spracherkennung zur Verfügung. Systemrückmeldungen erfolgen visuell am Bordmonitor oder auditiv.
19	*iDrive* - Seriensystem -	[Zeller*, et al.* 2001] [BMW 2001]: Bedienung sämtlicher Kommunikations-, Informations- und Assistenzsysteme über ein zentrales Bedienelement und eine graphische Anzeige am Bordmonitor. Zusätzlich können Infotainmentfunktionen per Sprache gesteuert werden.
20	*EBA*	[Marrenbach*, et al.* 2001]: Elektronische Betriebsanleitung, die per Sprache und mit einem besonderen manuellen Bedienelement gesteuert werden kann. Systeminformationen werden über das visuelle und auditive Interface an den Benutzer übermittelt.

Zur Hervorhebung von Ähnlichkeiten zwischen diesen Systemen werden sie in das oben vorgestellte Klassifikationsschema eingeordnet. Es zeichnen sich deutlich drei Gruppen ab (Bild 17):

A. Die nicht für die Domäne Fahrzeug entwickelten Systeme erlauben im Allgemeinen die parallele Eingabe von Informationen über mehrere Modalitäten und interpretieren diese im Verbund. Da bei diesen Systemen die volle Aufmerksamkeit des Benutzers für Bedienvorgänge zur Verfügung steht, sind dort synergistisch multimodale Ansätze sinnvoll.

B. Systeme, wie sie derzeit tatsächlich im Fahrzeug erhältlich sind, verarbeiten Informationen unabhängig voneinander. Es findet also keine Informationsfusion statt. Technisch ist zwar eine parallele Bedienung verschiedener Funktionen mit unterschiedlichen Modalitäten, wie z.B. der Eingabe einer Telefonnummer per Sprache und der parallelen manuellen Eingabe eines Navigationsziels, möglich. Aufgrund der hohen kognitiven Beanspruchung des Fahrers sind solche simultan-multimodalen Bedienvorgänge selten, weshalb diese Systeme stärker exklusiv-multimodal einzuordnen sind.

C. Bei experimentellen Systemen aus dem Forschungsbereich, die für den Einsatz im Fahrzeug konzipiert sind, ist eine deutliche Tendenz in Richtung alternierend-multimodal festzustellen. Die verschiedenen Modalitäten können dabei im Verbund eingesetzt werden, wobei immer nur eine gleichzeitig erlaubt ist. Die verschiedenen Modalitäten werden dabei so zur Verfügung gestellt, dass sie sich zu einem Gesamtbedienkonzept zusammenfügen. Es wird also dem Faktum, dass der Fahrer komplexe Funktionen im Allgemeinen nacheinander ausführt, dadurch Rechnung getragen, dass die verschiedenen Modalitäten dazu benutzt werden können, um eine Funktion effizient auszuführen und nicht mehrere simultan.

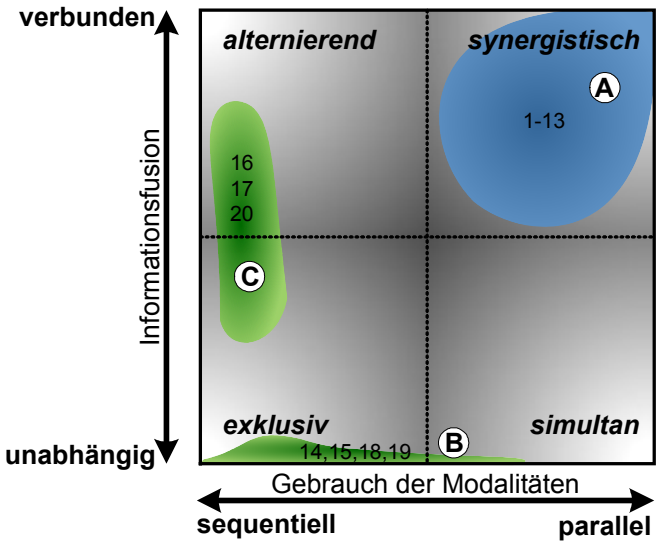


Bild 17 Multimodalitätsart der Systeme in Tabelle 2 (A) und Tabelle 3 (B,C)

2.2.6 Multimodalitätsart im Fahrzeug

Bild 17 zeigt, dass multimodale MMS je nach Aufgabentyp und Domäne deutliche Unterschiede aufweisen. Auffälligstes Merkmal für Fahrzeugsysteme ist dabei der sequentielle Gebrauch der verschiedenen Modalitäten. Der Verzicht auf den parallelen Einsatz der Modalitäten und damit auf die sehr mächtig erscheinenden synergistisch multimodalen Systeme lässt sich folgendermaßen begründen:

- Die Primäraufgabe des Nutzers ist die Führung und Stabilisierung des Fahrzeugs. Die hier betrachteten nicht fahrtbezogenen Systeme werden also nur „nebenbei“ bedient. Die durch diese Bedienung entstehende mentale Beanspruchung des Fahrers darf aber die Verkehrssicherheit nicht negativ beeinflussen. Die für einen parallelen Gebrauch notwendige Koordination der einzelnen Modalitäten durch den Fahrer würde möglicherweise diesem Ziel entgegenwirken. Multimodale Bedienung soll im Fahrzeug aber die kognitive Last des Nutzers niedrig halten, indem in jeder Dialogsituation die optimalen Bedienmöglichkeiten zur Verfügung stehen. Hierfür ist ein sequentieller Gebrauch der Modalitäten ausreichend.

- Räumliche Ortsspezifikationen (*spatial location commands*) sind laut [Oviatt*, et al.* 1997] der wichtigste Aufgabentyp für den synergistischen Einsatz mehrerer Modalitäten. Da aber solche Aufgaben, bei denen die räumliche Position, die Größe, die Orientierung oder die Kontur eines Objektes verändert werden müssen, in den betrachteten Systemen nicht vorkommen, fehlt die Hauptmotivation für den notwendigen hohen technologischen Aufwand.

- Redundante, parallele Eingaben durch den Benutzer stellen eine potentielle Möglichkeit dar, die Erkennungssicherheit der Nutzerintention zu erhöhen. Allerdings zeigen Untersuchungen von [Oviatt 1999] [Neuss 2000], dass die Struktur natürlicher multimodaler Kommunikation des Menschen nicht auf Redundanz beruht, weshalb synergistisch multimodale Ansätze nicht erforderlich sind.

Der in Bild 17 erkennbare Unterschied zwischen kommerziellen (B) und experimentellen (C) multimodalen Systemen im Fahrzeug spiegelt die aktuellen Bemühungen der Automobilhersteller wider, dem Fahrer in der jeweiligen Dialogsituation die am besten geeignete Modalität zur Verfügung zu stellen und somit eine sehr gute Anpassung an die Fahrsituation zu erreichen. Untersuchungen von [Rosnitschek 2000] [Detemple 2000] untermauern die Richtigkeit dieser Anstrengungen. Bei zukünftigen Seriensystemen ist somit eine Entwicklung von exklusiv- zu alternierend-multimodalen Systemen zu erwarten. Der in dieser Arbeit betrachtete Ansatz muss daher mit multimodalen Systemen umgehen können, die in dem in Bild 18 dargestellten Bereich liegen. Synergistische Multimodalität besitzt dabei keine große Relevanz, weshalb auf den dazu notwendigen aufwändigen Fusionsmechanismus verzichtet wird (3.4).

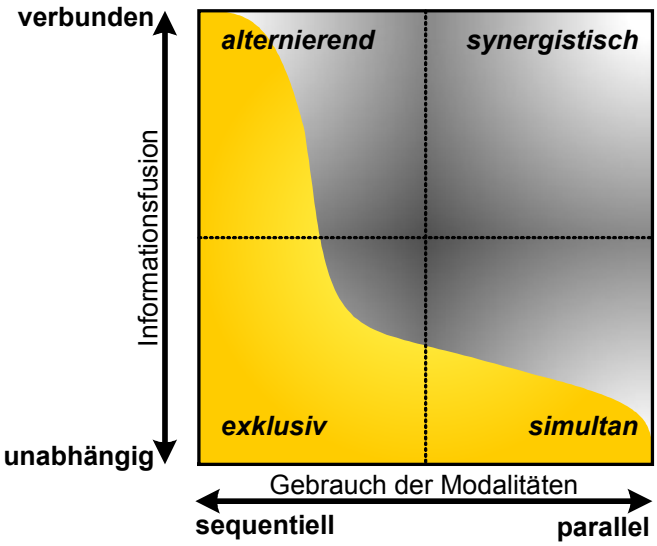


Bild 18 Im Fahrzeug relevante Multimodalitätsart

2.3 Gestaltungsaspekte von Mensch-Maschine-Systemen – Sicherung der Dialogqualität

Neben den bereits dargestellten technischen Aspekten stellt die inhaltliche Gestaltung einen wesentlichen Bestandteil bei der Realisierung qualitativ hochwertiger multimodaler MMS dar. Insbesondere die Güte der in 2.1.3 genannten Modelle der Maschine vom Benutzer M(B) und der Aufgabe M(A), sowie die damit verbundene Gestaltung des Bediendialogs sind entscheidend für die Gebrauchstauglichkeit eines HMI. Für eine strukturierte Betrachtung wird zunächst der Begriff Gebrauchstauglichkeit definiert, an dem sich der Dialogentwurf orientieren soll. Im

Folgenden werden dann Gestaltungshinweise vorgestellt, die allgemein auf MMS und insbesondere auf TICS zutreffen.

2.3.1 Gebrauchstauglichkeit – Usability

Ein wichtiges Ziel beim Entwurf eines HMI ist eine möglichst hohe *Gebrauchstauglichkeit* (*Usability*) des resultierenden Systems. Definitionen von Gebrauchstauglichkeit sind:

- [Usability] *is a concept compromising the effectiveness, efficiency and satisfaction with which specified users can achieve specified goals in a particular environment [ISO9241 1998].*
- *Extent to which an end-user is able to carry out required tasks successfully and without difficulty using a computer application system [Ravden & Johnson 1989].*

Eine auf empirischen Parametern beruhende, inhaltliche Definition findet sich in [Baggen & Hemmerling 2000]. Gebrauchstauglichkeit äußert sich in:

- *der Leichtigkeit und Schnelligkeit, mit der sich die Benutzung technischer Komponenten erlernen lässt,*
- *der Leichtigkeit und Schnelligkeit, mit der sich technische Komponenten für eine Aufgabe benutzen lassen,*
- *der Akzeptanz technischer Komponenten bei den Benutzern,*
- *dem Aufwand, den Benutzer freiwillig zur Kenntniserweiterung über technische Komponenten erbringen,*

2.3.2 Allgemeine Gestaltungshinweise – Styleguides

Zur Unterstützung des Entwicklers bei der Einhaltung der jeweils relevanten Aspekte während des Dialogentwurfs gibt es für verschiedene Domänen und Aufgaben so genannte *Gestaltungshinweise* (*Styleguides*). Die Darstellung der hierbei zu beachtenden Punkte erfolgt meist in Form von Auflistungen mit jeweils entsprechenden Erläuterungen. Für rein sprachbasierte Dialogsysteme gibt es hierzu eine Vielzahl an Styleguide-Sammlungen [Schumacher*, et al.* 1995] [Krahmer*, et al.* 1997]. In der Domäne Bildschirmarbeitsplatz ist als kleinster gemeinsamer Nenner ein weithin akzeptiertes Kriteriensystem als Norm festgelegt [ISO9241 1996]. Die darin enthaltenen sieben Merkmale für die gebrauchstaugliche Gestaltung technischer Komponenten sind so allgemein gehalten, dass sie im Kern auf beliebige Bereiche übertragbar sind (Bild 19).

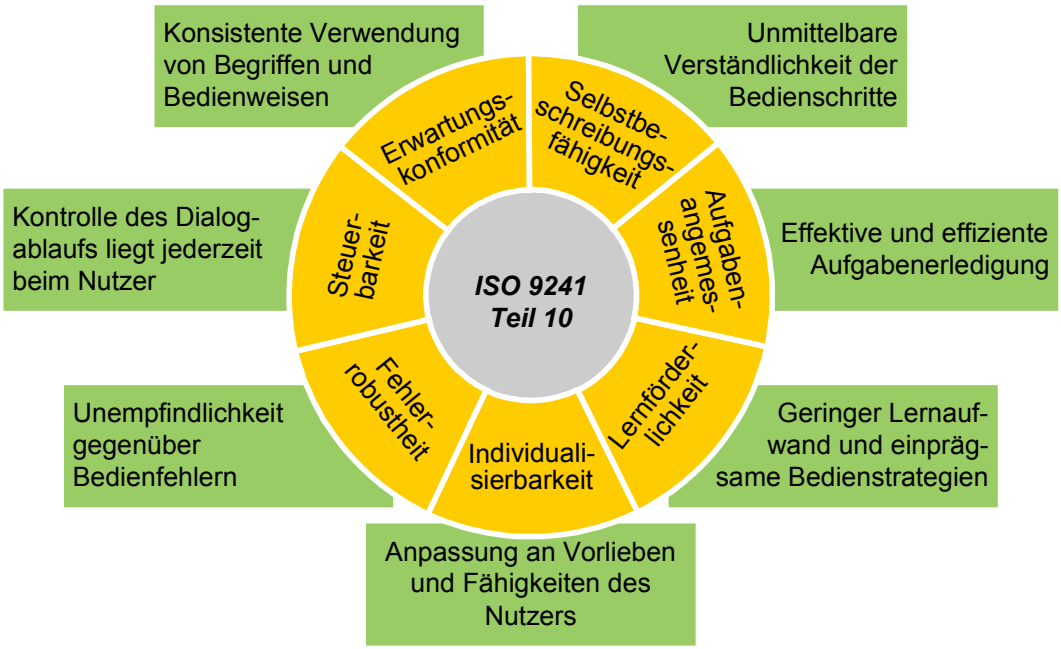


Bild 19 Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Grundsätze der Dialoggestaltung [ISO9241 1996]

2.3.3 Gestaltungshinweise für Fahrerinformationssysteme und Fahrerassistenzsysteme

Neben diesen in 2.3.2 genannten allgemeinen Gestaltungshinweisen liegt ein wichtiger Qualitätsaspekt bei den in dieser Arbeit betrachteten Dialogen in der Benutzbarkeit während der Fahrt. [Haller 1999] macht deutlich, dass eine Standardisierung dieser besonderen fahrzeugspezifischen Anforderungen sinnvoll und notwendig ist. Die Automobilhersteller und die Zulieferindustrie haben hierzu in der ISO TC22/ SC13/ WG8 den Normentwurf *„Straßenfahrzeuge – Ergonomische Aspekte von Fahrerinformations- und -assistenzsystemen – Grundsätze und Prüfverfahren des Dialogmanagements“* [ISO/DIS15005 2000] erarbeitet. Von der Europäischen Gemeinschaft wurde mit der *„Commission Recommendation on safe and efficient in-vehicle information and communication systems: A European Statement of principles on human machine interface.“* [2000/53/EC 2000] eine entsprechende Richtlinie erstellt, um die ergonomische und verkehrssichere Gestaltung von HMIs in Fahrzeugen zu beschreiben. Die Inhalte der Dokumente sind dabei sehr breit gefächert, wobei auf Fragen des Designs und der Positionierung von Anzeigen und Bedienelementen, der Präsentation von Informationen und auf eine adäquate Mensch-Maschine-Interaktion eingegangen wird. Es werden jeweils für die Entwicklung und Bewertung von Bediendialogen wichtige Prinzipien dargestellt.

Tabelle 4 Prinzipien zur Gestaltung von Fahrerinformations- und -assistenzsystemen nach [ISO/DIS15005 2000] und [2000/53/EC 2000]

ISO/ DIS 15005	**Statement of Principles**
• Eignung für den Gebrauch während der Fahrt: – Kompatibilität mit der Fahrzeugführung – Einfachheit – Timing/ Prioritäten • Eignung für TICS Aufgaben: – Konsistenz – Kontrollierbarkeit • Eignung für den Fahrer: – Selbsterklärungsfähigkeit – Konformität mit der Fahrererwartung – Fehlertoleranz	• Allgemeine Designprinzipien • Einbau- und Positionierungsprinzipien für Anzeigen und Bedienelemente • Prinzipien zur Informationsaufbereitung und -darstellung • Prinzipien zur Interaktion mit Anzeigen und Bedienelemente sowie zur Dialogführung • Prinzipien zum grundsätzlichen Systemverhalten • Prinzipielle Gestaltung von Anleitungen und Hilfesystemen zur Bedienung

Die Prinzipien werden dabei in Form von Anforderungen und Empfehlungen dargestellt und teilweise durch Beispiele ergänzt (Tabelle 5).

Tabelle 5 Darstellung der Prinzipien zur Gestaltung von Fahrerinformations- und -assistenzsystemen (Beispiel aus [ISO/DIS15005 2000])

Prinzip:
Konsistenz
Anforderung:
Die Informationsdarstellung und die Dialoge eines TICS-Gerätes müssen hinsichtlich der Merkmale Sinnesmodalität, Ort der Darstellung, Ausrichtung und Dialogmanagement konsistent sein.
Beispiel:
Eine Menüfunktion wird immer auf die gleiche Art oder Arten aufgerufen.

Obwohl insbesondere den Statement of Principles vorgeworfen wird, dass sie sehr vage und unspezifisch formuliert sind, einseitig ausgerichtet sind und daher viele Wiederholungen beinhalten, sowie nicht genügend auf die individuellen Leistungsfähigkeiten des Nutzers eingehen [Janssen 2000], stellen diese beiden Prinzipiensammlung eine gute Richtschnur zur Ausrichtung des Entwicklungsprozesses dar. In dieser Arbeit wird dabei versucht auf einer formalen Ebene während der Dialogerstellung den Dialogentwickler bei der Einhaltung dieser Richtlinien zu unterstützen. Die vorgeschlagenen Maßnahmen zur Sicherstellung der Dialogqualität (3.3) adressieren dabei insbesondere folgende Aspekte:

- Dialogverifikation
- Dialogkonsistenz
- Dialoginhalt

2.4 Bewertung und Evaluation von Mensch-Maschine-Systemen

Da in den jeweiligen Kriterien- und Prinzipiensammlungen (2.3) meist keine operationalisierten Verfahren beschrieben werden, mit denen die Einhaltung der Gestaltungshinweise sichergestellt werden können, entstehen große Interpretationsspielräume. Es wurden daher verschiedene Methoden entwickelt, mit deren Hilfe die Konformität eines MMS hinsichtlich eines solchen Kriteriensystems validiert werden kann. Je nach Entwicklungsphase müssen aufgrund der verschiedenen Entstehungszustände des MMS unterschiedliche Methoden angewandt werden [Marrenbach 2001] (Bild 20).

- Formale Methoden
  Basierend auf *Systemspezifikationen* und *Benutzermodellen*, wie beispielsweise GOMS[1], kann der Entwickler für bestimmte Aufgaben *Bewertungsmaße* abschätzen. Hierzu gehören die Zahl der Dialogschritte sowie Ausführungs- und Lernzeiten. Mit Hilfe dieser Methoden können zwar bereits zu einem sehr frühen Zeitpunkt Aussagen zur Gebrauchstauglichkeit getroffen werden, allerdings ist immer nur die Bewertung einzelner Aufgaben möglich. Aussagen zum Gesamtsystem sind im Allgemeinen mit großem Aufwand verbunden. Beispiele für solche Verfahren finden sich in [Nirschl 1990] [Marrenbach 2001].

[1] GOMS: Goals, Operators, Methods, Selection Rules [Card*, et al.* 1983]. Formale Darstellung des Benutzerverhaltens basierend auf einer Beschreibung der Aktionen, die Menschen mit interaktiven Systemen zur Erreichung eines Ziels durchführen.

- Kriterienorientierte Methoden
  Die Beurteilung des Systems erfolgt von *HMI-Experten* anhand von *Prüflisten* und mit Hilfe *heuristischer Verfahren*. Dabei ist kein funktionstüchtiges System notwendig, weshalb diese Verfahren bereits in der Entwurfsphase angewandt werden können. Als prominentester Vertreter ist hier die *Heuristische Evaluation* [Nielsen 1993] zu nennen, die sich mit besonders geringem zeitlichen und personellen Aufwand durchführen lässt. Aber auch andere Verfahren wie *Cognitive-Walkthrough* [Polson*, et al.* 1992] oder die speziell für die Domäne Fahrzeug zugeschnittene *MMI-Prüfliste* [Nirschl & Blum 2000] zählen zu diesen kriterienorientierten Methoden. Andere Beispiele sind *NICE* [Färber & Müller 2000] und *BASE* [Holte 2000].

- Experimentelle Methoden
  Die Beurteilung des Systems erfolgt in Form von Nutzertests mit potenziellen Anwendern in möglichst realistischen Szenarien. Diese empirischen Verfahren liefern sehr verlässliche und genaue Ergebnisse. Allerdings sind sie sehr zeit- und kostenaufwändig, weil zur Bewertung ein zumindest in Teilen funktionsfähiges System notwendig ist.

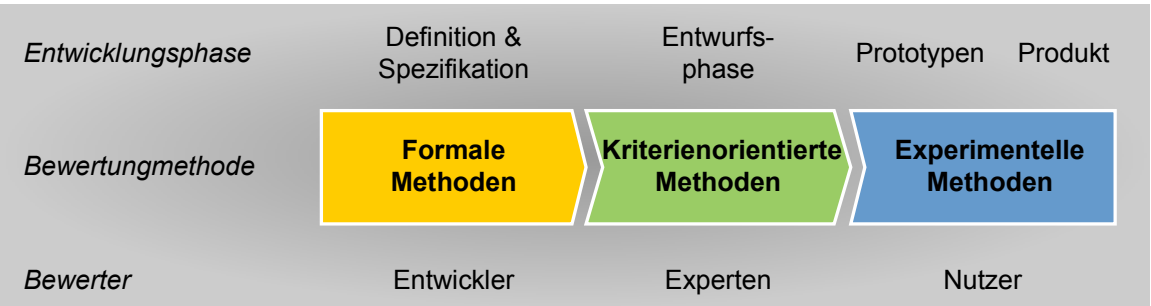


Bild 20 Bewertungsverfahren für MMS in den verschiedenen Entwicklungsphasen

Problematisch bei diesen Verfahren ist allerdings die Integration der Evaluation in den Entwicklungsprozess. Die verschiedenen Methoden setzen zumindest eine Spezifikation oder sogar einen funktionierenden Prototypen voraus. Diese sind zum notwendigen Zeitpunkt aber häufig noch nicht vorhanden. Bei Verfügbarkeit sind dann zusätzlich noch Zeit und Ressourcen erforderlich, um die Evaluationen vorzubereiten, durchzuführen und auszuwerten. Deshalb können diese wichtigen Absicherungen nur ausgewählt eingesetzt werden. [Norman 2001] stellt dieses Dilemma eindrucksvoll dar. Der in dieser Arbeit vorgestellte Ansatz versucht daher den Dialogentwickler schon bei der Erstellung der Spezifikation und des Prototypen so weit zu unterstützen, dass bereits in diesem frühen Stadium – also bevor obige Verfahren angewandt werden können – Maßnahmen zur Sicherung der Dialogqualität getroffen werden. Da in dieser Phase nur generelle Gesichtspunkte berücksichtigt werden können, bleiben die oben vorgestellten Methoden nach wie vor wichtig. Ihr Einsatz soll aber dabei nur noch zur Entdeckung dialogspezifischer Problemstellen notwendig sein.

Für qualitativ hochwertige Systeme ist also von entscheidender Bedeutung, dass die Absicherung der Gebrauchstauglichkeit mit den jeweils sinnvollen Methoden stabil in den Entwicklungs-

prozess integriert ist. [Parnow 2000] zeigt hierfür mit den *Quality Gates* eine mögliche Umsetzung.

2.5 Entwicklungswerkzeuge für multimodale Mensch-Maschine-Systeme

Die Verwendung von speziell angepassten Rapid-Prototyping Werkzeugen verspricht eine deutliche Erleichterung der Entwicklung multimodaler Mensch-Maschine-Systeme. Basierend auf ihren grundsätzlichen Vorteilen und den speziellen Forderungen, die der Fahrzeugentwicklungsprozess an sie stellt, wird zunächst der Fokus dieser Arbeit herausgearbeitet. Daran schließt sich eine Übersicht bestehender und für diese Anwendung relevanter Entwicklungswerkzeuge und Methoden an. Es wird dabei auf die jeweiligen besonderen Eigenschaften und Anwendungsbereiche eingegangen. Abschließend folgt eine Darstellung, wie in der vorliegenden Arbeit Elemente der verschiedenen Ansätze aufgegriffen und in einen gemeinsamen Kontext gestellt werden.

2.5.1 Nutzen von Entwicklungswerkzeugen

Entwicklungswerkzeuge für HMIs besitzen eine Reihe von Vorteilen und Erleichterungen für den Dialogentwickler und rechtfertigen dadurch die für ihre Entwicklung notwendigen Aufwendungen (siehe auch [Geiser 1990]). Wesentliche positive Eigenschaften sind dabei:

- Wiederverwendbarkeit von Komponenten und Funktionalitäten
  Vorgefertigte und im Werkzeug enthaltene bzw. integrierbare Module stellen dem Entwickler Funktionalitäten zur Verfügung, die ad hoc einsetzbar sind und stabil arbeiten. Hierzu gehören beispielsweise die verschiedenen Modalitäten, die Dialogmanagementkomponente oder auch ganze Teildialoge.

- Vermeidung konzeptueller Fehler
  Durch die Vorgabe des konzeptuellen Rahmens zur formalen Beschreibung und Darstellung des Bediendialogs bietet das Werkzeug dem Entwickler ein Gerüst, an dem er sich während des Dialogentwurfs orientieren kann und das ihn dadurch vor grundsätzlichen planerischen Fehlern bewahrt.

- Einfachere Verwendung – breiterer Anwenderkreis
  Die festgelegte und dadurch in allen Projekten wiederzufindende Dialogstruktur erleichtert die Orientierung in bereits bestehenden Arbeiten. Gleichzeitig kann durch eine sinnvolle graphische Darstellung die Dialogbeschreibung von der Quellcode- bzw. Skriptebene gekapselt werden. Dadurch stellen fundierte Programmierkenntnisse nicht mehr die wichtigste Voraussetzung bei der prototypischen Realisierung von Bediendialogen dar. Dies erlaubt einem breiteren Anwenderkreis, insbesondere aus den Bereichen Design und Ergonomie, die Verwendung des Werkzeugs.

- Qualitätssteigerung bei kürzerer Entwicklungszeit
  Jeder der genannten Punkte trägt letztendlich dazu bei, die Gesamtqualität des Bediendialogs zu steigern und gleichzeitig die Entwicklungszeit zu verkürzen. Dabei ist die normalerweise damit verbundene Senkung der Entwicklungskosten ein positiver Nebeneffekt.

2.5.2 Anforderungen aus dem Fahrzeugentwicklungsprozess – Fokus der Arbeit

In der Domäne Fahrzeug müssen allerdings die besonderen Anforderungen des Entwicklungsprozesses berücksichtigt werden. Dieser weist eine deutlich Zweiteilung in Konzeptphase und Serienentwicklung auf (Bild 21). In der Konzeptphase erfolgt eine Sondierung der am Markt verfügbaren Technologien und eine Untersuchung ihrer Eignung für den Einsatz im Fahrzeug. Hierzu werden verschiedene Konzeptstudien aufgebaut, die entweder in einfachen Laboraufbauten, in Sitzkisten für den Fahrsimulator oder in realen Fahrzeugen diese Technologien erlebbar und damit bewertbar machen. Entsprechend der Beurteilungen in diesen Untersuchungen wird daraus die Spezifikation für das Serienprodukt abgeleitet und dieses dann bis zur Serienreife entwickelt.

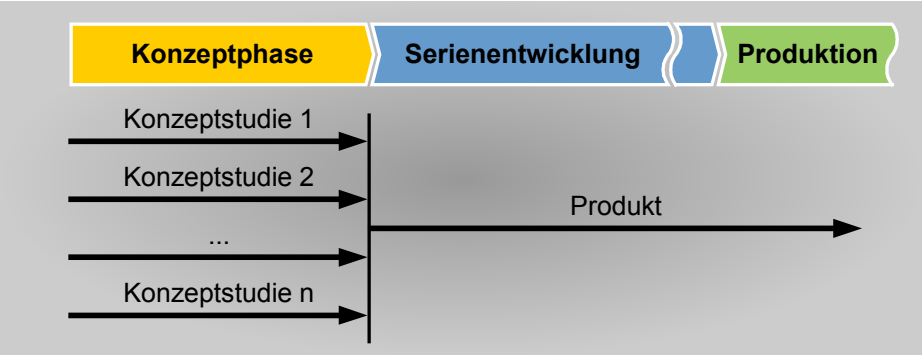


Bild 21 Übersichtsdarstellung Fahrzeugentwicklungsprozess

Je nach Entwicklungsphase muss das Entwicklungswerkzeug unterschiedliche Anforderungen erfüllen:

Konzeptphase

- Schnelles und einfaches Erlebbarmachen des Dialogs
  Zur realitätsnahen Bewertung von Konzepten müssen diese erlebbar gemacht werden. Diese Prototypen sollen möglichst einfach realisierbar sein.
- Einfache Integration neuer Modalitäten
  Um verschiedene technische Realisierungen einer Modalität vergleichen und neue Interaktionstechnologien beurteilen zu können, müssen sie einfach in die Entwicklungsumgebung integrierbar sein.

- Einfache Erweiterung des Funktionsumfangs einer Modalität
  Weiterentwicklungen und zusätzliche Funktionen einer Interaktionstechnologie müssen berücksichtigt werden können.

- Zentrale Konfiguration der Modalitäten und leichte Änderbarkeit des Dialogverhaltens
  Um Gestaltungsschwächen schnell verbessern und mögliche Alternativen einfach ausprobieren zu können, ist es in der Konzeptphase besonders wichtig, dass das Dialogverhalten und die Konfiguration der Modalitäten einfach zu beeinflussen sind. Gleichzeitig können dadurch die Modalitäten besser aufeinander abgestimmt werden.

- Dialogqualitätssichernde Maßnahmen
  Für eine sinnvolle Bewertung der Fahrzeugtauglichkeit einer Technologie bzw. eines Bedienkonzepts müssen die verschiedenen Gestaltungsaspekte (vgl. 2.3) auch in der Konzeptphase berücksichtigt werden. Dies kann teilweise bereits während der Dialogerstellung durch entsprechende in das Werkzeug integrierte Mechanismen erfolgen. Dadurch ist es möglich, eine deutliche Reduzierung des manuellen Aufwands zu erreichen.

- Dokumentationsfunktion
  Um basierend auf einer Konzeptstudie für die spätere Serienentwicklung detaillierte Spezifikationen erstellen zu können, sind automatische oder halbautomatische Verfahren zur Beschreibung und Dokumentation des Verhaltens des Prototyps sinnvoll.

Serienentwicklung

- Internationalisierung
  Um auf sprachliche, aber auch kulturelle Differenzen von Nutzern unterschiedlicher Nationalität eingehen zu können, muss die Sprache und auch das Erscheinungsbild des HMI anpassbar sein.

- Ausstattungsvarianten
  Die aufgrund von Sonderausstattungen zusätzlich im Fahrzeug verfügbaren Funktionen müssen dynamisch in das Bedienkonzept integriert werden. Nicht vorhandene Komponenten dürfen allerdings weder bedienbar sein, noch zu leeren Einträgen in der Bedienoberfläche führen.

- Effizienter Quellcode und Portierbarkeit
  Die normalerweise hohen Stückzahlen in der Fahrzeugproduktion sorgen für einen großen Kostendruck bei den Einzelkomponenten. Um die Anforderungen an Prozessorleistung und Arbeitsspeicher gering zu halten, muss der erzeugte Quellcode möglichst effizient und unter Umständen auf mehrere Zielplattformen portierbar sein.

- Integration in Gesamtfahrzeugkontext
  Für eine problemlose Integration in den Steuergeräteverbund des Gesamtfahrzeugs müssen die entsprechenden Konventionen bezüglich des Kommunikationsverhaltens sowie des Netzwerk- und Fehlermanagements eingehalten werden.

Die Phasen des Entwicklungsprozesses sind allerdings bezüglich der Anforderungen nicht so stark entkoppelt, wie es auf den ersten Blick erscheint. So ist beispielsweise die in der Konzeptphase genannte Einhaltung der verschiedenen Gestaltungsaspekte auch in der Serienentwicklung relevant. Die besonderen Ressourcenanforderungen für das Serienprodukt machen allerdings eine hardwarenahe Entwicklung erforderlich, die meist auf Quellcodeebene durchgeführt werden muss. Die Realisierung der inhaltlichen und funktionalen Umfänge des HMI erfolgt dabei aufgrund detaillierter Spezifikationen. Diese Spezifikationen basieren im Allgemeinen auf den Ergebnissen in der Konzeptphase. Eine wichtige Kernkompetenz bei der HMI-Entwicklung liegt somit in der Erstellung der Spezifikation und den vorausgehenden Untersuchungen während der Konzeptphase. Die vorliegende Arbeit konzentriert sich daher auf die Anforderungen in der Konzeptphase.

2.5.3 Bestehende Entwicklungswerkzeuge

Die im Folgenden beschriebenen Werkzeug- und Methodenklassen sind zur Generierung von HMIs und zur Dialogsteuerung einsetzbar. Zur besseren Vergleichbarkeit werden die jeweils typischen Vor- und Nachteile gegenübergestellt.

Entwicklungsumgebungen für Höhere Programmiersprachen

Programmiersprachen wie beispielsweise C, C++ oder Java erlauben aufgrund ihrer Universalität dem Entwickler ein Maximum an Flexibilität und Freiheit bezüglich der Realisierung graphischer Benutzeroberflächen und der darunter liegenden Funktionalität. Deshalb können neue Modalitäten, sofern sie in passenden Programmbibliotheken gekapselt vorliegen, praktisch ohne Restriktionen integriert werden. Aufgrund dieser Offenheit liegt es allerdings auch in der Verantwortung des Entwicklers die Software zu strukturieren und eine sinnvolle informationstechnische Repräsentation des Bediendialogs zu finden. Da aber die Programmierung normalerweise sehr problemorientiert getrieben ist, lehnt man sich bei der Architektur der Software häufig sehr stark an der jeweiligen Dialogstruktur an. Dadurch wird die Wiederverwendbarkeit von Teilfunktionen deutlich erschwert. Daneben steigt aufgrund der fehlenden visuellen Repräsentation des Bediendialogs die Komplexität der Darstellung sehr schnell an und die Gefahr von Funktions- und Implementierungsfehlern nimmt zu. Die Überprüfung aufgaben- und domänenspezifischer Qualitätsaspekte ist nicht möglich. Zur Realisierung sind umfangreiche Programmierkenntnisse erforderlich.

Beschreibungssprachen – Markup Languages

Mit auf *XML*[1] [W3C 2002] [Seeboerger-Weichselbaum 2000] aufsetzenden Beschreibungssprachen können die Funktionsumfänge verschiedener Modalitäten einheitlich beschrieben werden. *VoiceXML*[2] [W3C 2001] ermöglicht beispielsweise die Beschreibung von Sprachdialogen. Die Funktionsumfänge können dabei unabhängig von der jeweiligen Systemplattform und Zielsprache beschrieben werden. Die vorgegebene Syntax zwingt den Entwickler zu einer strukturierten, übersichtlichen und einheitlichen Darstellung des Bediendialogs und erlaubt gleichzeitig eine zentrale Konfiguration der Modalität in einem Dokument. Diese textuelle Spezifikation wird zur Laufzeit interpretiert, wodurch ein hohes Maß an Wiederverwendbarkeit und Funktionssicherheit entsteht. Durch das in der jeweiligen Interpretationseinheit fest definierte Dialogverhalten kann zumindest für die einzelne Modalität ein gewisses Maß an Konsistenz gewährleistet werden. Dialogqualität im Sinne entsprechender Gestaltungsaspekte wird aber von Beschreibungssprachen nicht adressiert. Um den vorgegebenen Funktionsumfang zu erweitern muss der Interpreter selbst modifiziert werden, was mit einem nicht unerheblichen Aufwand verbunden ist. Daneben bereitet die Darstellung komplexer logischer Zusammenhänge in XML große Schwierigkeiten.

CASE Tools

CASE[3] *Tools* dienen zur abstrakten Beschreibung des Verhaltens komplexer technischer Systeme. Aufgrund der umfangreichen graphischen Darstellungsmöglichkeiten sind diese Beschreibungen einfach zu erfassen und leicht verständlich. Darüber hinaus bieten einige CASE Tools (Statemate, Rapid, Rhapsody) umfangreiche Funktionalitäten zur Simulation des Systemverhaltens und zur Codeerzeugung. Dadurch kann aus der graphischen Spezifikation eines Systems ein erlebbarer Prototyp generiert werden. Hierzu wird in [Bengler*, et al.* 2000] anhand des Werkzeugs Statemate gezeigt, wie mit Hilfe von Statecharts ein multimodales Bedienkonzept realisiert werden kann. CASE Tools sind zur Modellierung von Dialogabläufen gut geeig-

---

[1] XML: Extensible Markup Language. Meta-Beschreibungssprache, aus der sich, basierend auf frei definierbaren Tags und Attributen, für spezifische Aufgaben dedizierte Beschreibungssprachen ableiten lassen.

[2] VoiceXML: Voice Extensible Markup Language

[3] CASE: Computer Aided Software Engineering. Rechnergestützte Planung, Organisation und Kontrolle großer Softwareprojekte. Ein wesentliches Kennzeichen stellen dabei die verschiedenen visuellen Beschreibungsmöglichkeiten zur Spezifikation der Architektur sowie des funktionalen und zeitlichen Verhaltens des technischen Systems dar. Als sehr mächtiges Darstellungsverfahren ist hier UML (Unified Modeling Language) zu nennen, da es die Vorteile mehrerer Beschreibungstechniken, wie beispielsweise MSC (Message Sequence Charts) und Statecharts, in sich vereint.

net, da die enthaltenen Spezifikationskomponenten meist auf Zustandsautomaten beruhen und dafür umfangreiche Visualisierungsmöglichkeiten besitzen (vgl. 2.2.3.2). Wegen ihrer allgemeinen Verwendbarkeit bieten diese Werkzeuge dem Entwickler allerdings keine Unterstützung zur aufgabenangepassten Modellierung. Die funktionale Anbindung der Modalitäten kann meist über native Programmierschnittstellen (C-/ Java-Interfaces) oder Systemschnittstellen (Sockets, Multicast, DDE[1], CORBA[2]) realisiert werden. Da CASE Tools stärker für Simulationsaufgaben als für Rapid Prototyping ausgelegt sind, gestaltet sich dies häufig als schwierig und fehleranfällig. Trotzdem gestatten sie eine relativ einfache Wiederverwendung und Erweiterung bestehender Funktionsumfänge. Daneben ermöglichen Model-Checker eine Konsistenzprüfung der Spezifikation. Wegen der fehlenden Domänenanpassung können damit allerdings keine spezifischen Gestaltungsaspekte berücksichtigt werden.

Neben diesen sehr stark die Systemlogik adressierenden CASE Tools gibt es speziell auf die Realisierung graphischer Benutzeroberflächen ausgelegte Werkzeuge (Director, MMI-Tool, VAPS, Altia), die zum Teil eigens für den Automotiv-Bereich entwickelt wurden. Diese erlauben die Animation von Graphikobjekten zur Visualisierung komplexer Menüstrukturen bzw. zur Darstellung analoger und diskreter Informationen. Neben diesen meist ausgezeichneten Graphikfähigkeiten enthalten einige dieser Werkzeuge umfangreiche Skriptsprachen bzw. Programmierschnittstellen, die zur Anbindung der Modalitäten und zur Dialogsteuerung benutzt werden können. Diese führt allerdings auch zu den bereits genannten Problemen bezüglich der unübersichtlichen Dialogbeschreibung, der aufwändigen Integration der Modalitäten, der fehlenden Dialogqualitätssicherung und der mangelnden Nutzerführung.

Sprachdialogwerkzeuge

Zur Entwicklung von Sprachdialogen gibt es insbesondere für CTI[3]-Anwendungen speziell angepasste Werkzeuge (Nuance V-Builder, Speechworks, Vocon-Designer, CSLU-Toolkit). Aufgrund der spezifischen Ausrichtung können diese Werkzeuge die besonderen Anforderungen dieser Domäne optimal berücksichtigen und so den Entwickler bestmöglich durch den Dialogerstellungsprozess führen. Hierzu gehören die fest im Werkzeug integrierten Konfigurationsmöglichkeiten bezüglich Spracherkennung und Sprachausgabe sowie die Methoden zur Definition und Visualisierung des Systemverhaltens. Darüber hinausgehende Anforderungen für multimodale Bedienkonzepte werden dagegen nicht adressiert. Dies ist beispielsweise die flexible Integration weiterer Modalitäten.

---

[1] DDE: Dynamic Data Exchange

[2] CORBA: Common Object Request Broker Architecture

[3] CTI: Computer-Telephone-Integration. Bezeichnung für die zunehmende Automatisierung von Call-Centers durch den Einsatz von Spracherkennung.

Wie aus dieser Gegenüberstellung hervorgeht, haben die verfügbaren Werkzeuge entweder einen so breiten Anwendungsbereich, dass sie auf die oben genannten Anforderungen nicht detailliert genug eingehen können, oder sie sind so exakt für eine bestimmte Anwendung ausgelegt, dass sie zur Erstellung multimodaler Bedienkonzepte nicht ausreichend mächtig sind. Diese Arbeit geht daher einen Mittelweg und zeigt wie verschiedene Ansätze dieser Werkzeuge so zusammengefügt werden können, dass daraus ein sinnvolles Ganzes zur prototypischen Implementierung multimodaler Bedienkonzepte entsteht. Ähnlich den Beschreibungssprachen wird mit Hilfe einer festen Syntax der Funktionsumfang der einzelnen Modalitäten kontextfrei beschrieben (vgl. 3.1). Für eine schnelle und möglichst einfache Integration der Modalitäten wird eine C++-Bibliothek als Kommunikationsschnittstelle bereitgestellt (vgl. 4.3.2). Dadurch kann jede Modalität mit maximaler Flexibilität in einer höheren Programmiersprache oder einem anderen Werkzeug entwickelt werden und steht dann als separates Modul zur Verfügung (vgl. 4.3.3). Auf diese Weise wird ein hoher Grad an Wiederverwendbarkeit erreicht. Der Bediendialog wird als hierarchischer Zustandsautomat beschrieben (vgl. 3.2) und kann in Anlehnung an CASE Tools graphisch spezifiziert werden (vgl. 4.2.2). Die Darstellung ist dabei auf die Randbedingungen in der Domäne Automobil ausgelegt, wodurch eine gute Nutzerunterstützung erreicht wird. Gleichzeitig werden Maßnahmen zur Sicherung der Dialogqualität im Sinne der entsprechenden Gestaltungsaspekte getroffen (vgl. 3.3).

Kapitel 3 Formale Repräsentation

Basierend auf einer Generalisierung und Formalisierung der im vorangegangenen Kapitel dargestellten Anforderungen wird nun ein theoretisches Gerüst für Entwicklungswerkzeuge zur Erstellung fahrzeugtauglicher multimodaler Bedienkonzepte geschaffen. Diese formale Beschreibung umfasst die kontextfreie Modellierung der Modalitäten, die Repräsentation des Bediendialogs, die Maßnahmen zur Sicherung der Dialogqualität sowie eine allgemein verwendbare Heuristik zur Interpretation der Dialogspezifikation während der Ausführung.

Die formale Repräsentation des Dialogs und der Modalitäten ist dabei die Grundlage, auf der alle für das Entwicklungswerkzeug notwendigen zusätzlichen Komponenten aufsetzen (Bild 22). Dies ist zum einen die Dialogerstellung zur Unterstützung des Entwicklers bei der Spezifikation des Dialogs (4.2) und zum anderen die Dialogausführung (4.3). Mit letzterer wird das Bedienkonzept erlebbar gemacht, was eine sinnvolle und aussagekräftige Bewertung des Interaktionsprinzips ermöglicht [Niedermaier & Lang 2001].

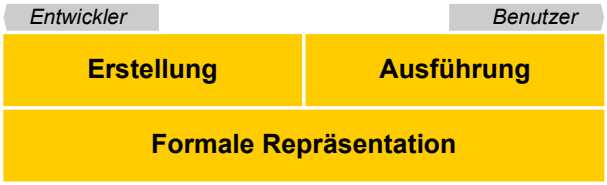


Bild 22 Notwendige strukturelle Komponenten des Entwicklungswerkzeugs

3.1 Kontextfreie Modellierung der Modalitäten

Um eine flexible Integration neuer und eine einfache Funktionserweiterung existierender Modalitäten zu ermöglichen, wird ein Beschreibungsverfahren eingeführt, mit dem die Eigenschaften beliebiger Modalitäten unabhängig vom jeweiligen Funktionskontext dargestellt werden können. Dies erlaubt beispielsweise die Beschreibung der Funktionsumfänge eines Spracherkenners und eines Bedienelements mit denselben semantischen Mitteln. Da die funktionale Realisierung jeweils als eigenständiges Softwaremodul (4.3.3) erfolgt, ist die Einbindung einer neuen Modalität ohne zusätzliche Modifikationen an der Entwicklungsumgebung möglich. Die Interaktion dieser als *Dialogkomponenten* bezeichneten Softwaremodule mit dem Dialogmanager basiert auf den drei semantischen Elementen *Parameter*, *Kommandos* und *Nachrichten* (Bild 23). Bevor also beispielsweise ein neuer Spracherkenner zur Dialogbeschreibung verwendet werden

kann, muss lediglich dessen Schnittstellendefinition in der Entwicklungsumgebung bekannt gemacht werden.

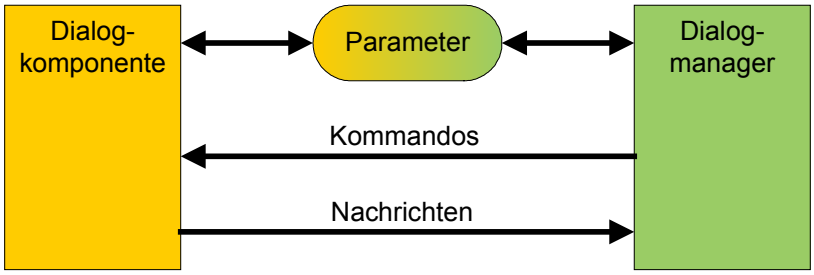


Bild 23 Interaktion Dialogmanager und Dialogkomponente

3.1.1 Dialogkomponenten

Die funktionalen Realisierungen der einzelnen Modalitäten werden als Dialogkomponenten $\mathbf{D} = (D_1 \quad D_2 \quad D_3 \quad \ldots \quad D_n)$ mit $n \in \mathrm{N}_0$ bezeichnet. Der Status einer Dialogkomponente D_i, also der Aktivierungszustand, wird über die Dialogkomponenten-Typfunktion $\vartheta_D : \mathbf{D} \rightarrow \mathbf{Status}$ zugeordnet.

$$[Status]_{D_i} = \vartheta_D(D_i)$$

Das Element $Status_{D_i}$ gibt an, ob eine Dialogkomponente am Dialogmanager angemeldet und somit aktiv in den Dialogablauf integriert ist oder nicht.

$$Status_{D_i} \in \mathbf{Status} \qquad \text{mit } \mathbf{Status} := \left\{ \begin{matrix} \texttt{active} \\ \texttt{inactive} \end{matrix} \right\}$$

Beispiel:

Für einen Spracherkenner `ASR` und ein manuelles Bedienelement `BMWCANDLL` mit $\mathbf{D} = \{\texttt{ASR} \quad \texttt{BMWCANDLL}\}$ bedeutet

- $\vartheta_D(\texttt{ASR}) = [\texttt{inactive}]$ Spracherkenner nicht aktiviert
- $\vartheta_D(\texttt{BMWCANDLL}) = [\texttt{active}]$ Bedienelement bereit

3.1.2 Parameter

Mit Hilfe der verschiedenen Parameter **P** werden einerseits Informationen über den internen Zustand der Dialogkomponenten bereitgestellt und anderseits kann mit ihnen das Verhalten der Dialogkomponenten beeinflusst werden. Die Parameter-Typfunktion $\vartheta_P : \mathbf{P} \rightarrow \mathbf{D} \times \mathbf{P_{Type}} \times \mathbf{P_{Usage}}$ definiert für jeden Parameter P_i über das Tripel

$$\left[D \quad P_{Type} \quad P_{Usage}\right]_{P_i} = \vartheta_P(P_i)$$

- die zugehörige Dialogkomponente $D_{P_i} \in \mathbf{D}$,
- den Typ des Parameters

$$P_{Type_{P_i}} \in \mathbf{P_{Type}} \qquad \text{mit } \mathbf{P_{Type}} := \left\{ \begin{array}{c} \texttt{int} \\ \texttt{float} \\ \texttt{string} \end{array} \right\}$$

- und die Verwendung des Parameters:

$$P_{Usage_{P_i}} \in \mathbf{P_{Usage}} \qquad \text{mit } \mathbf{P_{Usage}} := \left\{ \begin{array}{c} \texttt{DC\_Setting} \\ \texttt{DC\_State} \\ \texttt{DC\_Cmd} \\ \texttt{DC\_Msg} \end{array} \right\}$$

  P_{Usage} gibt an, in welchem Kontext, d.h. zu welchem Zweck der Parameter verwendet wird:

  – *Konfigurationsparameter* (`DC_Setting`):
    Der Inhalt wird durch den Dialogentwickler während der Dialogerstellung festgelegt (4.2.2.2). Auf diese Weise kann das Verhalten der Dialogkomponente konfiguriert werden.

    Beispiel:
    Mit Hilfe des Konfigurationsparameters `ASR_Confidence` legt der Dialogentwickler die mindestens notwendige Erkennungssicherheit des Spracherkenners fest. Ein Erkennungsergebnis wird von der Dialogkomponente nur dann weitergegeben, wenn die Erkennungssicherheit höher als dieser Schwellwert ist.

    $$\vartheta_P(\texttt{ASR\_Confidence}) = [\texttt{ASR float DC\_Setting}]$$

  – *Statusparameter* (`DC_State`):
    Die Wertfestlegung erfolgt während der Dialogausführung durch die Dialogkomponente. Dieser Wert bleibt solange gültig, bis er von der Dialogkomponente aktualisiert wird (3.5.3). Dadurch entsteht ein In-

formationspool, der zu jedem Zeitpunkt der Dialogausführung den jeweils aktuellen Zustand der Dialogkomponenten widerspiegelt. Auf diese Weise werden Informationen über den internen Zustand der Dialogkomponenten zur Dialogsteuerung dauerhaft verfügbar gemacht.

Beispiel:

Der Statusparameter `EC_Button` des manuellen Bedienelements gibt an, ob eine bestimmte Taste gedrückt ist oder nicht. Die Dialogkomponente aktualisiert den Wert bei Veränderung.

$$\vartheta_P(\texttt{EC\_Button}) = [\texttt{BMWCANDLL string DC\_State}]$$

- *Kommandoparameter* $(\texttt{DC\_Cmd})$:
  Der Parameter ist einem Kommando zugeordnet (3.1.3). Beim Aufruf des Kommandos (3.2.10) wird der jeweils aktuelle Wert mit übergeben. Dadurch ist eine detaillierte Beeinflussung des Verhaltens der Dialogkomponente möglich. Der Wert wird durch den Dialogentwickler definiert.

- *Nachrichtenparameter* $(\texttt{DC\_Msg})$:
  Um das Verhalten von aktivierbaren Nachrichten $M_{Act} = \texttt{activatable}$ (3.1.4) zu beeinflussen, werden beim Aktivierungsvorgang die jeweils gültigen Parameterwerte übergeben (3.5.6). Die Festlegung des Wertes erfolgt ebenfalls durch den Dialogentwickler.

3.1.3 Kommandos

Die Steuerung der Dialogkomponenten durch den Dialogmanager basiert auf Kommandos $\mathbf{K}$. Diese können als Funktions- oder Methodenaufrufe zur Beeinflussung des Verhaltens der Dialogkomponenten verstanden werden. Mit Hilfe der Kommando-Typfunktion[1] $\vartheta_K : \mathbf{K} \rightarrow \mathbf{D} \times 2^{|\mathbf{P}|} \times \mathbf{K_{Type}}$ wird hierzu einem Kommando K_i das Tripel

$$\left[D,\ \mathbf{P},\ K_{Type}\right]_{K_i} = \vartheta_K(K_i)$$

zugeordnet, womit

- die zugehörige Dialogkomponente $D_{K_i} \in \mathbf{D}$,
- die zur Parametrisierung notwendigen Kommandoparameter

$$\mathbf{P}_{K_i} \in 2^{\left|\left\{P \in \mathbf{P} \mid \vartheta_P^{PUsage}(P) = \texttt{DC\_Cmd}\right\}\right|}$$

- und der Kommandotyp bestimmt sind.

$$K_{Type_{K_i}} \in \mathbf{K_{Type}} \qquad \text{mit } \mathbf{K_{Type}} := \left\{ \begin{array}{c} \texttt{DC\_Cmd} \\ \texttt{DDM\_Cmd} \\ \texttt{DDM\_Cmd precompile} \end{array} \right\}$$

Mit Hilfe von Kommandos können zum einen während der Dialogausführung bestimmte Funktionen der Dialogkomponente ausgelöst werden. Zum anderen sind sie ein wichtiges Instrument zur zentralen Konfiguration der Dialogkomponenten in der Entwicklungsumgebung. Der Kommandotyp unterscheidet die Kommandos hierzu in

- *Funktionskommandos* (DC_Cmd):
  Diese Kommandos lösen Funktionen aus, die während der Dialogausführung normalerweise als Reaktionen des Systems auf Nutzereingaben wahrgenommen werden können.

---

[1] $2^{|\mathbf{P}|}$: Potenzmenge von $\mathbf{P}$. Menge aller möglichen Teilmengen von $\mathbf{P}$.

$$2^{|\mathbf{P}|} := \left\{ \mathbf{X} \mid \mathbf{X} \subset \mathbf{P} \right\}$$

Beispiel:

Nach einer Aktivierungsaufforderung durch den Benutzer wird der Spracherkenner mit dem Kommando `ASR_RecStart` aktiviert.

$$\vartheta_K(\texttt{ASR\_RecStart}) = [\texttt{ASR} \;\; \varnothing \;\; \texttt{DC\_Cmd}]$$

- *Konfigurationskommandos* (`DDM_Cmd`):
  Mit Hilfe der Konfigurationskommandos kann der Dialogentwickler in den Dialogkomponenten spezielle Nachrichten mit $M_{Type} = \texttt{DdMessage}$ (3.1.4) festlegen. Dies erlaubt eine Anpassung der Dialogkomponente an den jeweiligen Anwendungsfall, ohne dass Änderungen an ihr selbst erforderlich wären. Gleichzeitig können diese Anpassungen unter einer einheitlichen Bedienoberfläche im Werkzeug vorgenommen werden. Die Generierung der Nachrichten erfolgt dynamisch während der Dialogausführung.

  Beispiel:

  Das Konfigurationskommando `ASR_DynList` erlaubt im Spracherkenner die dynamische Generierung von Vokabular aus einer Wortliste. Als Parameter müssen eine Wortliste `Name`, ein Bezeichner `Identifier` und die mindestens notwendige Erkennungssicherheit `Confidence` angegeben werden. Mit Hilfe dieser Funktion können beispielsweise die sprachgesteuerte Senderwahl für ein Radio oder die Sprachwahl von Telefonbucheinträgen eines Mobiltelefons realisiert werden.

$$\vartheta_K(\texttt{ASR\_DynList}) = \begin{bmatrix} & \begin{bmatrix} \texttt{Name} \\ \texttt{Identifier} \\ \texttt{Confidence} \end{bmatrix} & \\ \texttt{ASR} & & \texttt{DDM\_Cmd} \end{bmatrix}$$

- *vorkompilierbare Konfigurationskommandos* (`DDM_Cmd precompile`):
  Analog zu den Konfigurationskommandos kann der Dialogentwickler auch Nachrichten mit $M_{Type} = \texttt{DdMessage precompile}$ (3.1.4) in den Dialogkomponenten definieren. Diese müssen allerdings bereits beim Starten der Dialogausführung in der Dialogkomponente bekannt gemacht werden, da aufgrund zeitlicher oder technischer Restriktionen dies nicht dynamisch erfolgen kann.

  Beispiel:

  Mit dem Kommando `ASR_MenuSelect` können beliebige Sprachkommandos definiert werden, wie sie beispielsweise zum Erreichen bestimmter Menüs notwendig sind. Da diese im Normalfall konstant über den gesamten Dialogverlauf sind, ist es aus Per-

formancegründen sinnvoll, dieses Vokabular gleich zu Beginn der Dialogausführung im Spracherkenner bekannt zu machen.

$$\vartheta_K(\texttt{ASR\_MenuSelect}) = \left[\texttt{ASR} \quad \begin{bmatrix}\texttt{Name}\\ \texttt{Confidence}\end{bmatrix} \quad \texttt{DDM\_Cmd precompile}\right]$$

3.1.4 Nachrichten

Der ereignisbasierte Informationsfluss von der jeweiligen Dialogkomponente D_i zum Dialogmanager basiert auf Nachrichten $\mathbf{M}$. Ausgelöst durch normalerweise asynchrone Ereignisse, wie beispielsweise Nutzereingaben, werden hierzu von den Dialogkomponenten die zugehörigen Nachrichten an den Dialogmanager geschickt. Entsprechend der Dialogdefinition löst dieser dann Aktionen aus. Mit Hilfe der Nachrichten-Typfunktion $\vartheta_M : \mathbf{M} \rightarrow \mathbf{D} \times \mathbf{M_{Act}} \times \mathbf{M_{Type}} \times 2^{|\mathbf{P}|} \times \mathbf{K_{Call}}$ wird einer Nachricht M_i das Tupel

$$\left[D \quad M_{Act} \quad M_{Type} \quad \mathbf{P} \quad K_{Call}\right]_{M_i} = \vartheta_M(M_i)$$

zugeordnet. Die einzelnen Elemente bezeichnen dabei

- die auslösende Dialogkomponente $D_{M_i} \in \mathbf{D}$,
- den Aktivierungstyp

$$M_{Act_{M_i}} \in \mathbf{M_{Act}} \qquad \text{mit } \mathbf{M_{Act}} := \left\{\begin{matrix}\texttt{always}\\ \texttt{activatable}\end{matrix}\right\}$$

  Je nachdem, ob eine Nachricht erst aktiviert werden muss, bevor sie ausgelöst werden kann oder nicht, werden stets aktive (`always`) und aktivierbare (`activatable`) Nachrichten unterschieden (3.5.6).

  Beispiel:

  Um eine möglichst gute Erkennungsrate zu erhalten, darf im Spracherkenner je nach Dialogsituation nur bestimmtes Vokabular aktiviert sein. Deshalb sind alle Nachrichten, die aufgrund einer Spracherkennung ausgelöst werden, vom Typ $M_{Act} = \texttt{activatable}$.

- den Nachrichtentyp

$$M_{Type_{M_i}} \in \mathbf{M_{Type}} \qquad \text{mit } \mathbf{M_{Type}} := \left\{ \begin{array}{c} \texttt{FxMessage} \\ \texttt{DdMessage} \\ \texttt{DdMessage precompile} \end{array} \right\}$$

  Je nach Art und Zeitpunkt der Festlegung unterscheidet man die Nachrichten in

  - *fest definierte Nachrichten* (`FxMessage`)
    Die Nachricht ist fest in der Dialogkomponente implementiert. Sie steht daher in jedem Projekt gleichbedeutend und unveränderlich zur Verfügung.

    Beispiel:
    Die Nachricht `EC_ButtonState` zeigt an, dass sich der Aktivierungszustand eines Tasters des manuellen Bedienelements geändert hat. Die Nachricht ist stets aktiv und fester Bestandteil der Dialogkomponente.

$$\vartheta_M(\texttt{EC\_ButtonState}) = [\texttt{BMWCANDLL always FxMessage} \ \varnothing \ \varnothing]$$

  - *entwicklerdefinierte Nachrichten* (`DdMessage`)
    Der Entwickler kann mit Hilfe von Konfigurationskommandos (3.1.3) Nachrichten speziell für das jeweilige Projekt definieren. Diese flexible Anpassungsmöglichkeit ermöglicht einen breiten Einsatz und ein hohes Maß an Wiederverwendbarkeit der einzelnen Dialogkomponenten.

  - *vorkompilierbare entwicklerdefinierte Nachrichten* (`DdMessage precompile`)
    Diese basieren auf vorkompilierbaren Konfigurationskommandos (3.1.3). Sie werden daher bereits beim Start der Dialogausführung in der jeweiligen Dialogkomponente bekannt gemacht. Ansonsten unterscheiden sie sich nicht von entwicklerdefinierten Nachrichten.

- die Nachrichtenparameter $\mathbf{P}_{M_i} \subseteq \mathbf{P}$. Das Verhalten von aktivierbaren, fest definierten Nachrichten kann beim Aktivierungsvorgang (3.5.6) durch die jeweiligen Werte der Nachrichtenparameter beeinflusst werden. Dabei gilt:

  1. $\forall P \in \mathbf{P}_{M_i} : \ \vartheta_{\mathrm{P}}^{\mathrm{P_{Usage}}}(P) = \texttt{DC\_Msg}$
  2. $M_{Type_{M_i}} = \texttt{FxMessage}$
  3. $M_{Act_{M_i}} = \texttt{activatable}$

- den jeweiligen Kommandoaufruf $K_{Call_{M_i}} \in \mathbf{K_{Call}}$ (3.2.10) bei benutzerdefinierten Nachrichten. Für das zugeordnete Kommando K_{M_i} gilt:

$$\vartheta_{\mathrm{K}}^{\mathrm{K_{Type}}}(K_{M_i}) = \texttt{DDM\_Cmd} \vee$$
$$\vartheta_{\mathrm{K}}^{\mathrm{K_{Type}}}(K_{M_i}) = \texttt{DDM\_Cmd precompile} \qquad \text{mit} \quad K_{M_i} = \kappa^{K}\left(K_{Call_{M_i}}\right)$$

3.2 Modellierung des Bediendialogs

Diese bislang nebeneinander stehenden Dialogkomponenten müssen zu einem sinnvollen Verbund in Form eines Bediendialogs zusammengeführt werden können. Hierzu ist eine informationstechnische Repräsentation der Interaktion zwischen Mensch und Maschine erforderlich. Die Modellierung dieses Bediendialogs erfolgt, in Anlehnung an die in 2.2.3.2 erwähnten Statecharts [Harel 1987], deterministisch in Form eines erweiterten, hierarchischen Zustandsautomaten.

Die Grundelemente des Dialogs

$$A := \left[\mathbf{S} \quad \mathbf{T} \quad \mathbf{R} \quad S_{Act} \quad \mathbf{S_{Hist}}\right]$$

sind *Dialogzustände* $\mathbf{S}$, *Transitionen* $\mathbf{T}$, *Reaktionen* $\mathbf{R}$, der *aktuelle Dialogzustand* S_{Act} und die *Dialoghistorie* $\mathbf{S_{Hist}}$. Diese bedienen sich der *Parameterwerte* $\mathbf{P_{Value}}$, der *Kontexte* $\mathbf{C}$, der *Gültigkeitsbereiche* $\mathbf{G}$ und $\mathbf{G_{Act}}$, der *Kommandoaufrufe* $\mathbf{K_{Call}}$ sowie der *Bedingungen* $\mathbf{B}$.

Der Bediendialog wird in einzelne Zustände (3.2.1) unterteilt, die durch Transitionen (3.2.2) miteinander verbunden sind. Das funktionale Verhalten der Dialogkomponenten in einem Zustand wird durch Reaktionen (3.2.3) gesteuert. Mit Hilfe von Kontexten können hierzu verschiedene Gültigkeitsbereiche (3.2.7) definiert werden. Diese ermöglichen eine speziell an der Problemstellung orientierte hierarchische Strukturierung der Beschreibung. Aufgrund dieser besonderen Anpassung wird die Zahl der manuell zu definierenden Transitionen und Reaktionen reduziert. Dadurch kann der Komplexitätsanstieg bei umfangreichen Dialogspezifikationen flach gehalten werden. Im Folgenden wird das der Dialogbeschreibung zugrunde liegende formale Modell eingeführt und detailliert erläutert.

3.2.1 Zustände

Die Dialogzustände $\mathbf{S}$ sind die Elementarelemente der Dialogbeschreibung. Über die Dialogzustand-Typfunktion $\vartheta_S : \mathbf{S} \rightarrow 2^{|\mathbf{C}|} \times \mathbf{S} \times 2^{|\mathbf{W}|}$ erfolgt eine nähere Beschreibung des jeweiligen Dialogzustandes S_i.

$$[\mathbf{C} \quad S_{Parent} \quad \mathbf{W}]_{S_i} = \vartheta_S(S_i)$$

Hierdurch wird festgelegt:

- Die Zuordnung des jeweiligen Dialogzustandes zu Kontexten $\mathbf{C}_{S_i} \in 2^{|\mathbf{C}|}$. Die Festlegung der Kontexte $\mathbf{C}$ erfolgt je nach Bedarf durch den Entwickler. Wie in 3.2.7 näher erläutert wird, dienen sie zur hierarchischen Gruppierung der Dialogzustände.
- $S_{Parent_{S_i}} \in \{ S \mid S \in \mathbf{S} \wedge S \neq S_i \}$ gibt den Eltern-Dialogzustand in der visuellen Repräsentation an (4.2.2.1). Dabei gibt es $\exists_1 S_i : \vartheta_S^{S_{Parent}}(S_i) = \varnothing$ genau ein S_i, für das kein solcher Zustand definiert ist. Hierbei handelt es sich um den obersten Zustand der visuellen Repräsentation. Die Dialogausführung beginnt bei diesem Zustand.
- In $\mathbf{W}_{S_i} \in 2^{|\mathbf{W}|}$ sind den Zustand charakterisierende Schlüsselwörter abgelegt. Diese können zur konsistenten Konfiguration der Dialogkomponenten bei der Definition von Parameterwerten (4.2.2.2) verwendet werden.

Beispiel:

In Bild 24 sind verschiedene Zustände einer einfachen Menüauswahl dargestellt, wie sie beispielsweise in einem TICS vorkommen könnten.

$$\mathbf{S} := \{Hauptmenü \quad Navigation \quad Entertainment \quad Telefon\}$$

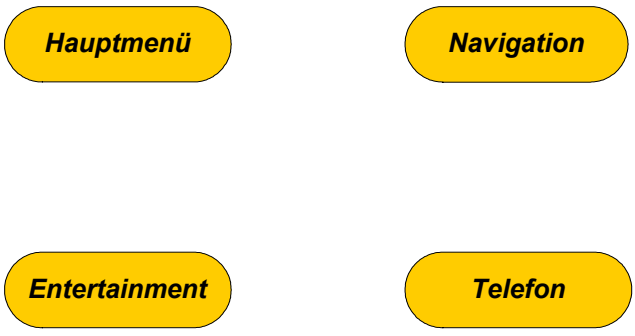


Bild 24 Zustände einer einfachen Menüauswahl

3.2.2 Transitionen

Übergänge zwischen den einzelnen Dialogzuständen werden als Transitionen $\mathbf{T}$ bezeichnet. Sie werden jeweils von einer Nachricht aus einer Dialogkomponente ausgelöst. Mit der Transitionsfunktion $\tau : \mathbf{M} \times \mathbf{G} \rightarrow 2^{|\mathbf{T}|}$ können hierzu die für jede Kombination aus Nachricht M_i und Gültigkeitsbereich G_j möglichen Transitionen $\mathbf{T}_{M_i G_j}$ bestimmt werden. Eine Transition T_k enthält den Zielzustand $S_{next_k} \in \mathbf{S}$ und eine Bedingung $B_k \in \mathbf{B}$. Einer Bedingung liegt ein logischer Ausdruck zugrunde (3.2.11), der je nach Wert die Ausführung des Übergangs in den Dialogzustand S_{next} freigibt oder nicht. Der Gültigkeitsbereich $G_j \in \mathbf{G}$ legt fest, ob die Transition lediglich in einem bestimmten Dialogzustand, in allen Zuständen eines Kontextes oder immer durchgeführt werden soll (3.2.7).

$$\mathbf{T}_{M_i G_j} = \tau(M_i \quad G_j) \qquad \text{mit } \mathbf{T}_{M_i G_j} \subseteq (\mathbf{T} = \mathbf{S} \times \mathbf{B})$$

$$(T_k = [S_{next} \quad B]_K) \in \mathbf{T}_{M_i G_j}$$

Beispiel:

Die in Bild 25 eingezeichneten Nachrichten, die beispielsweise aufgrund von Spracheingaben oder der manuellen Betätigung eines Eingabeelements erzeugt werden, lösen die Übergänge zwischen den einzelnen Dialogzuständen aus. Dies wird durch die Transitionen $\mathbf{T}$ und die zugehörige Transitionsfunktion τ beschrieben:

$$\begin{array}{l}
\tau(\texttt{msgNav} \quad \textit{Hauptmenü} \quad) = \{T_1\} \\
\tau(\texttt{msgTel} \quad \textit{Hauptmenü} \quad) = \{T_2\} \\
\tau(\texttt{msgEnt} \quad \textit{Hauptmenü} \quad) = \{T_3\} \\
\tau(\texttt{msgMenu} \quad \textit{Navigation} \quad) = \{T_4\} \\
\tau(\texttt{msgMenu} \quad \textit{Telefon} \quad) = \{T_5\} \\
\tau(\texttt{msgMenu} \quad \textit{Entertainment} \quad) = \{T_6\}
\end{array}
\quad \text{und} \quad
\mathbf{T} := \begin{Bmatrix} T_1 \\ T_2 \\ T_3 \\ T_4 \\ T_5 \\ T_6 \end{Bmatrix} = \begin{Bmatrix}
[\textit{Navigation} \quad \varnothing] \\
[\textit{Telefon} \quad \varnothing] \\
[\textit{Entertainment} \quad \varnothing] \\
[\textit{Hauptmenü} \quad \varnothing] \\
[\textit{Hauptmenü} \quad \varnothing] \\
[\textit{Hauptmenü} \quad \varnothing]
\end{Bmatrix}$$

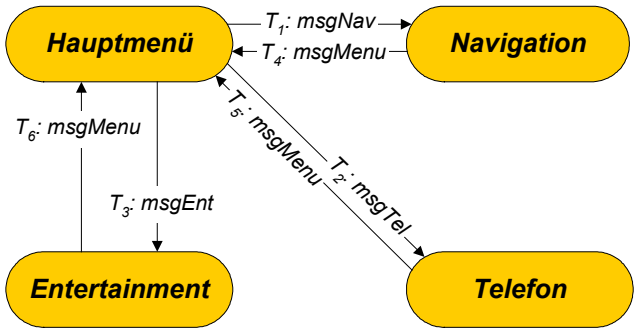


Bild 25 Auslösen von Transitionen zwischen den Zuständen durch Nachrichten

3.2.3 Reaktionen

Das funktionale Verhalten der Dialogkomponenten innerhalb eines Zustandes wird durch Reaktionen gesteuert. Zur Auslösung einer Reaktion ist wie bei den Transitionen eine Nachricht erforderlich, der aber anstatt eines Zielzustandes Kommandos zugeordnet werden. Die Reaktionsfunktion $\rho : \mathbf{M} \times \mathbf{G} \rightarrow 2^{|\mathbf{R}|}$ legt hierzu für jede Kombination aus eintreffender Nachricht M_i und Gültigkeitsbereich G_j eine beliebige Anzahl an Reaktionen $\mathbf{R}_{M_i G_j}$ fest. Eine Reaktion R_k enthält die notwendigen Kommandoaufrufe $\mathbf{K}_{\mathbf{Call}_k} \subseteq \mathbf{K}_{\mathbf{Call}}$ (3.2.10), sowie eine Bedingung $B_k \in \mathbf{B}$ (3.2.11), die zur Auslösung der Reaktion erfüllt sein muss. Je nach zugeordnetem Gültigkeitsbereich $G_j \in \mathbf{G}$ ist die Reaktion nur in bestimmten Zuständen relevant (3.2.7).

$$\mathbf{R}_{M_i G_j} = \rho(M_i \quad G_j) \qquad \text{mit } \mathbf{R}_{M_i G_j} \subseteq \left(\mathbf{R} = 2^{|\mathbf{K}_{\mathbf{Call}}|} \times \mathbf{B}\right)$$

$$\left(R_k = [\mathbf{K}_{\mathbf{Call}} \quad B]_k\right) \in \mathbf{R}_{M_i G_j}$$

Beispiel:

In Bild 26 wird für jeden der Zustände eine Reaktion definiert. Beim Eintreffen der Nachricht `msgPTT` wird jeweils das Kommando `cmdStart` ausgelöst.

$$\begin{aligned}
&\rho(\ \texttt{msgPTT}\ \ \textit{Hauptmenü}\ &&) = \{R_1\}\\
&\rho(\ \texttt{msgPTT}\ \ \textit{Navigation}\ &&) = \{R_2\}\\
&\rho(\ \texttt{msgPTT}\ \ \textit{Telefon}\ &&) = \{R_3\}\\
&\rho(\ \texttt{msgPTT}\ \ \textit{Entertainment}\ &&) = \{R_4\}
\end{aligned}$$

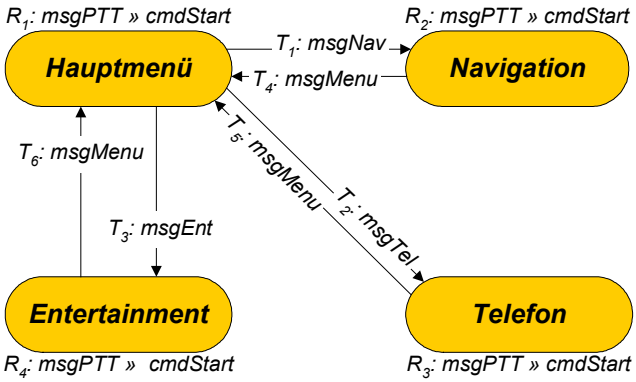


Bild 26 Kontrolle des funktionalen Verhaltens durch Reaktionen

3.2.4 Aktueller Dialogzustand

Die Dialogausführung befindet sich zu jedem Zeitpunkt in einem definierten Zustand. Dieser wird als aktueller Dialogzustand $S_{Act} \in \mathbf{S}$ bezeichnet. Änderungen des aktuellen Dialogzustandes werden durch Zustandswechsel (3.5.6) ausgelöst.

3.2.5 Dialoghistorie

Die Dialoghistorie ist ein LIFO[1]-Speicher, in dem vor Ausführung eines Zustandwechsels der aktuelle Dialogzustand S_{Act} abgelegt wird. Mit Hilfe der Dialoghistorie ist es möglich die Aufrufreihenfolge der Dialogzustände zurückzuverfolgen und somit eine History-Funktion zu realisieren (3.5.6).

$$\mathbf{S_{Hist}} = \begin{bmatrix} S_1 & S_2 & S_3 & ... & S_n \end{bmatrix} \qquad \text{mit } n \in \mathrm{N}_0 \ \wedge \ S_i \in \mathbf{S}$$

Dabei war S_i vor i Transitionen der aktuelle Dialogzustand S_{Act}. Beim Ausführen der Transition wird somit der eben noch aktuelle Zustand an die erste Position gesetzt und alle anderen Elemente werden um eine Position nach rechts verschoben. Die Dialoghistorie verlängert sich dadurch um ein Element.

$$\forall S_i \in \mathbf{S_{Hist}} : \begin{cases} S_i = S_{i-1} & \text{für i} > 1 \ \wedge \ \text{i} \le \text{n}+1 \\ S_1 = S_{Act} & \text{für i} = 1 \end{cases}$$

Transitionen $T = \begin{bmatrix} S_{next} & B \end{bmatrix}$, die einen Schritt in der Dialoghistorie zurückführen, sind am Eintrag $S_{next} = \texttt{xxxLastStatexxx}$ für den Zielzustand erkennbar. Beim Zustandswechsel wird hier der aktuelle Zustand S_{Act} auf den ersten Eintrag $S_1 \in \mathbf{S_{Hist}}$ der Dialoghistorie gesetzt. Um den nun aktuellen Dialogzustand aus der Historie zu löschen, werden die restlichen Elemente jeweils um eine Position nach links verschoben. Diese Operation verkürzt die Dialoghistorie um einen Eintrag.

$$\forall S_i \in \mathbf{S_{Hist}} : \begin{cases} S_i = S_{i+1} & \text{für i} \ge 1 \ \wedge \ \text{i} < \text{n} \\ S_n = \varnothing \end{cases}$$

[1] LIFO: Last In First Out. Speicherstruktur, bei der das zuletzt abgelegte Element als erstes wieder ausgelesen wird.

3.2.6 Parameterwert

Für eine sinnvolle Verwendung der Parameter müssen je nach Dialogsituation unterschiedliche Werte zugewiesen werden können. Hierzu ordnet die Parameterwert-Funktion $\upsilon : \mathbf{P} \times \mathbf{G} \rightarrow 2^{|\mathbf{P}_{\mathbf{Value}}|}$ jedem Parameter P_i für verschiedene Gültigkeitsbereiche G_j Parameterwerte $\mathbf{P}_{\mathbf{Value}_{P_i G_j}} \subseteq \mathbf{P}_{\mathbf{Value}}$ zu. Der im Parameterwert P_{Value_k} abgelegte spezielle Wert V_k wird nur dann verwendet, wenn die zugehörige Bedingung B_k (3.2.11) erfüllt ist.

$$\mathbf{P}_{\mathbf{Value}_{P_i G_j}} = \upsilon(P_i \quad G_j) \qquad \text{mit } \mathbf{P}_{\mathbf{Value}_{P_i G_j}} \subseteq \left(\mathbf{P}_{\mathbf{Value}} = \mathbf{V} \times \mathbf{B}\right)$$

$$\left(P_{Value_k} = [V \quad B]_k\right) \in \mathbf{P}_{\mathbf{Value}_{P_i G_j}}$$

- Der spezielle Wert $V_k \in \mathbf{V}$ ist entweder eine Konstante entsprechend des zugehörigen Parametertyps $\vartheta_P^{P_{Type}}(P_i)$ oder er ist mit einem anderen Parameter $P_j \in \mathbf{P}$ verbunden, dessen in der jeweiligen Dialogsituation aktueller Wert verwendet wird. Für solche Verknüpfungen können lediglich Konfigurations- und Statusparameter verwendet werden, da nur in diesen entsprechende statische Informationen enthalten sind.

$$V = \begin{cases} \text{constant Value} \\ P_j \end{cases} \qquad \text{mit} \quad \vartheta_{\mathrm{P}}^{\mathrm{P_{Usage}}}(P_j) \in \begin{Bmatrix} \texttt{DC\_Setting} \\ \texttt{DC\_State} \end{Bmatrix}$$

- Die Verwendung von Bedingungen ist lediglich bei Konfigurationsparametern $P \in \mathbf{P}: \ \vartheta_{\mathrm{P}}^{\mathrm{P_{Usage}}}(P) = \texttt{DC\_Setting}$ relevant. Bei den drei anderen Parametertypen ist dies nicht sinnvoll, da ihre Werte entweder mit einer Nachricht bzw. einem Kommando fest verbunden sind oder erst während der Dialogausführung mit Werten belegt werden und damit nicht direkt vom Entwickler beeinflusst werden können.

3.2.7 Kontexte – Gültigkeitsbereiche

Um mit planaren Zustandsautomaten in mehreren Zuständen ein identisches Verhalten zu erzeugen, müssen die Definitionen mehrfach per Hand gemacht werden. Die Reaktionen R_1 bis R_4 und die Transitionen T_4 bis T_6 in Bild 26 verdeutlichen hierzu, dass die Komplexität und der manuelle Aufwand bereits in einem einfachen Beispiel sehr schnell ansteigen.

Statecharts adressieren dieses Problem durch eine hierarchische Strukturierungsmöglichkeit [Harel, *et al.* 1987]. Dies bedeutet, dass mehrere Zustände in übergeordneten Zuständen zusammengefasst werden können. Ordnet man einem solchen übergeordneten Zustand ein bestimmtes Verhalten zu, dann ist es gleichzeitig für alle untergeordneten Zustände gültig, ohne dass es für jeden explizit definiert werden muss.

Ein ähnliche Herangehensweise findet auch in dieser Arbeit Verwendung. Die Zustände können zu beliebigen Funktionseinheiten zusammengefasst werden. Diese in Bild 27 dargestellten Kontexte $C_i \in \mathbf{C}$ sind frei definierbar, wobei jeder Zustand mehreren Kontexten zugeordnet werden kann. Dies nimmt die von [Harel & Kahana 1992] umrissene Erweiterungsmöglichkeit der Statecharts mit überlappenden Zuständen auf. Dadurch kann einem Zustand auf sehr einfache und flexible Weise das Verhalten eines weiteren Kontextes hinzugefügt bzw. genommen werden. Die Bedürfnisse bei der Spezifikation von Bediendialogen werden dadurch noch besser berücksichtigt.

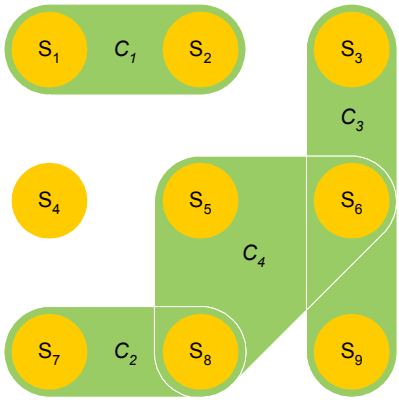


Bild 27 Zuordnung der Zustände S_6 und S_8 zu mehreren Kontexten

Allerdings ist hierzu für jede Transition und Reaktion sowie für alle Parameterwerte von Konfigurationsparametern die Angabe eines Gültigkeitsbereiches $G_i \in \mathbf{G}$ erforderlich. Dieser legt fest, ob die Definition lokal im jeweiligen Dialogzustand, für alle Zustände eines bestimmten Kontexts oder global für alle Dialogzustände gültig ist. Diese eine Definition kann somit das Verhalten in mehreren Zuständen beeinflussen. Dadurch reduziert sich die Zahl der manuell zu erstellenden Definitionen deutlich. Gleichzeitig kann ein konsistentes Verhalten über mehrere Zustände hinweg sichergestellt werden (3.3.2).

$$G_i = \begin{cases} S \in \mathbf{S} & \text{lokal für jeweiligen Dialogzustand} \\ C \in \mathbf{C} & \text{für einen bestimmten Kontext} \\ \varnothing & \text{global} \end{cases}$$

Beispiel:

Bild 28 zeigt, wie die Beschreibung des aus Bild 26 bekannten Beispiels durch die Einführung eines Kontextes und die Verwendung verschiedener Gültigkeitsbereichen vereinfacht werden kann. Die Zahl der Transitionen wird um zwei und die Zahl der Reaktionen um drei reduziert.

$$\begin{aligned} &\tau(\texttt{msgNav}\ \textit{Hauptmenü}) = \{T_1\} \\ &\tau(\texttt{msgTel}\ \textit{Hauptmenü}) = \{T_2\} \\ &\tau(\texttt{msgEnt}\ \textit{Hauptmenü}) = \{T_3\} \\ &\tau(\texttt{msgMenu}\ \textit{Menüauswahl}) = \{T_4\} \end{aligned} \quad \text{und} \quad \rho(\texttt{msgPTT}\ \varnothing) = \{R_1\}$$

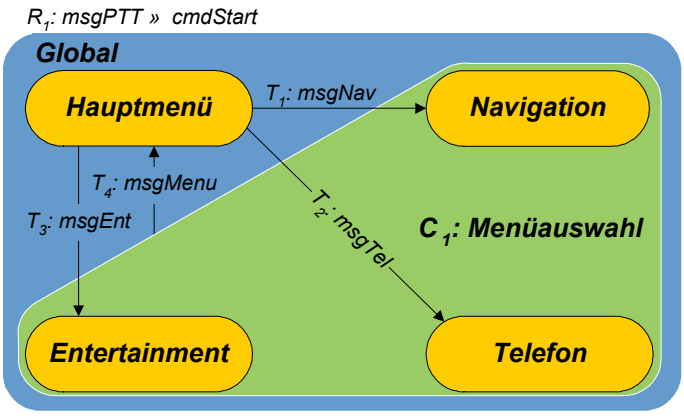


Bild 28 Reduzierung der notwendigen Transitionen und Reaktionen durch hierarchische Strukturierung

3.2.8 Aktuelle Gültigkeitsbereiche

Die aktuellen Gültigkeitsbereiche $\mathbf{G_{Act}}$ umfassen alle für den aktuellen Dialogzustand S_{Act} relevanten Gültigkeitsbereiche. Dazu gehören der aktuelle Dialogzustand und dessen zugeordnete Kontexte sowie der globale Gültigkeitsbereich. Damit können während der Dialogausführung die in der jeweiligen Dialogsituation bedeutsamen Transitionen, Reaktionen und Parameterwerte identifiziert werden (3.5).

$$\mathbf{G_{Act}} := \left\{ G \in \mathbf{G} \;\middle|\; G = S_{Act} \;\vee\; G \in \vartheta_S^{\mathbf{C}}(S_{Act}) \;\vee\; G = \varnothing \right\}$$

Beispiel:

Befindet sich der in Bild 28 dargestellte Dialog im Zustand $S_{Act} = \texttt{Navigation}$, dann gilt für den aktuellen Gültigkeitsbereich:

$$\mathbf{G_{Act}} = \{\texttt{Navigation} \;\; \texttt{Menüauswahl} \;\; \varnothing\}$$

3.2.9 Handhabung von Mehrfachdefinitionen

Für jede Transition und Reaktion sowie für jeden Parameterwert eines Konfigurationsparameters muss ein Gültigkeitsbereich angegeben werden. Dadurch können sich Dialogzustände ergeben, in denen beispielsweise für eine Nachricht sowohl eine Transition mit lokalen als auch mit globalen oder kontextweiten Gültigkeitsbereich definiert ist. Ähnliche Situationen sind bei Reaktionen und den aktuellen Parameterwerten von Konfigurationsparametern möglich. Zur Auflösung dieser Mehrfachdefinitionen werden zwei unterschiedliche Vorgehensweisen angewandt.

Beispiel:

Im Zustand `Entertainment` (Bild 29) kann die Nachricht `msgMenu` sowohl die für den Kontext C_1 definierte Transition T_4, als auch die lokale T_5 auslösen. Im Zustand `Telefon`

sind die Auslösebedingungen für die globale Reaktion R_1 und die lokale R_2 beim Eintreffen der Nachricht `msgPTT` ebenfalls gleichzeitig erfüllt.

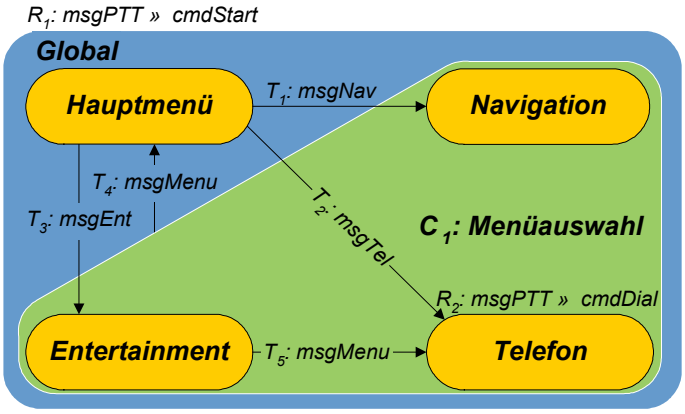


Bild 29 Mehrfachdefinitionen in den Zuständen *Entertainment* und *Telefon*

3.2.9.1 Verdeckungsmechanismus bei Transitionen und Parameterwerten

Da sich der Dialog immer genau in einem Zustand befindet, können nicht mehrere Transitionen gleichzeitig ausgeführt werden. Das Gleiche gilt für Parameter, denen ebenfalls stets nur ein Wert gleichzeitig zugeordnet werden kann. Aus diesem Grund wird für Transitionen und Parameterwerte ein Verdeckungsmechanismus benutzt, der die durch Mehrfachdefinitionen entstehende Uneindeutigkeiten in der Dialogbeschreibung auflöst. Dabei wird nach der Prämisse vorgegangen, dass Definitionen mit kleinerem Gültigkeitsbereich Ausnahmen oder Sonderfälle einer übergeordneten Regel sind und deshalb größere Bedeutung besitzen. Lokale Festlegungen überschreiben daher kontextweite, die wiederum vor globalen Definitionen ausgeführt werden (Bild 30). Auf diese Weise entsteht ein mächtiges Instrument, mit dem auf sehr komfortable Weise sowohl das generelle, als auch das spezielle Verhalten des Dialogs beeinflusst werden kann.

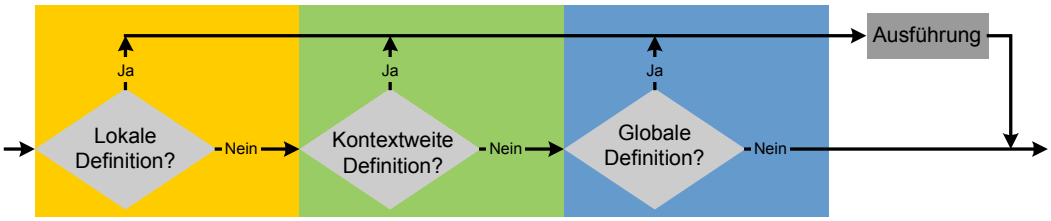


Bild 30 Verdeckung von Transitionen und Parameterwerten mit unterschiedlichen Gültigkeitsbereichen

Beispiel:

Im Zustand *Entertainment* des Beispiels (Bild 29) verdeckt die lokale Transition T_5 die kontextweite definierte T_4. Beim Eintreffen der Nachricht `msgMenu` wird die Transition zum Zustand *Telefon* ausgeführt.

3.2.9.2 Bearbeitungsreihenfolge bei Reaktionen

Da in einem Zustand durchaus mehrere Reaktionen erlaubt sind, ergibt sich hier eine veränderte Situation, die keinen Verdeckungsmechanismus erforderlich macht. Wenn für eine Nachricht in einer Dialogsituation mehrere Reaktionen mit unterschiedlichen Gültigkeitsbereichen definiert sind, werden alle abgearbeitet. Dabei wird vom Speziellen zum Allgemeinen vorgegangen. Das heißt, als Erstes werden lokale, dann kontextweite und schließlich globale Reaktionen ausgelöst (Bild 31). Auf diese Weise kann das generelle Verhalten des Dialogs bei Bedarf um spezielle Aspekte ergänzt werden. Soll dagegen in einer bestimmten Dialogsituation eine sonst gültige Reaktion mit größerem Gültigkeitsbereich nicht ausgeführt werden, dann muss dies über eine entsprechende Bedingung in dieser Reaktion realisiert werden.

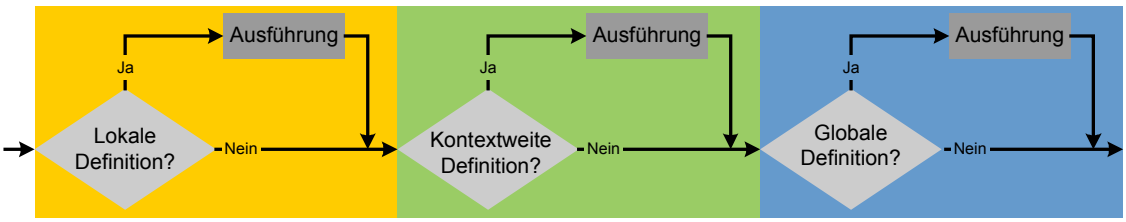


Bild 31 Bearbeitungsreihenfolge für Reaktionen mit unterschiedlichen Gültigkeitsbereichen

Beispiel:

Dies bedeutet im Zustand `Telefon` des Beispiels (Bild 29), dass beim Eintreffen der Nachricht `msgPTT` die lokale Reaktion R_2 vor der globalen R_1 ausgelöst wird.

3.2.10 Kommandoaufruf

Beim Aufruf eines Kommandos einer Dialogkomponente müssen, wie in 3.1.3 bereits angedeutet wurde, die entsprechenden Parameterwerte übergeben werden. Die Kommandoaufruf-Funktion $\kappa : \mathbf{K_{Call}} \rightarrow \mathbf{K} \times 2^{|\mathbf{P_{Value}}|}$ verbindet ein Kommando mit dem dazugehörigen Satz an Parameterwerten zu einem Kommandoaufruf $K_{Call} \in \mathbf{K_{Call}}$.

$$[K \quad \mathbf{P_{Value}}]_{K_{Call}} = \kappa(K_{Call})$$

Wobei nur Werte für Parameter in $\mathbf{P}_{\mathbf{Value}_{KCall}}$ enthalten sein können, die auch zu dem entsprechenden Kommando gehören.

3.2.11 Bedingung

Ob eine bestimmte Definition einer Transition, einer Reaktion oder eines Einstellungsparameterwertes in der jeweiligen Dialogsituation für die Fortführung des Bediendialogs relevant ist, kann neben dem jeweiligen Gültigkeitsbereich auch von einer Bedingung abhängig gemacht werden. Die Bedingungs-Typfunktion $\vartheta_B : \mathbf{B} \rightarrow \mathbf{V_1} \times \mathbf{O_{Comp}} \times \mathbf{V_2}$ ordnet einer Bedingung B_i das Tripel

$$\left[V_1 \quad O_{Comp} \quad V_2\right]_{B_i} = \vartheta_B(B_i)$$

zu. Dieses besteht aus

- einem ersten und einem zweiten Operanden V_1, V_2 mit

$$V_1, V_2 := \begin{cases} \text{constant Value} \\ P_j \in \mathbf{P} \end{cases} \qquad \text{mit} \quad \vartheta_{\mathrm{P}}^{\mathrm{P_{Usage}}}(P_j) \in \begin{Bmatrix} \texttt{DC\_Setting} \\ \texttt{DC\_State} \end{Bmatrix}$$

- und einem Operator.

$$O_{Comp} \in \mathbf{O_{Comp}} \qquad \text{mit } \mathbf{O_{Comp}} := \{== \quad != \quad < \quad > \quad <= \quad >=\}$$

Bei einer Bedingung werden also zwei Operanden logisch miteinander verknüpft, wobei jedem dieser Operanden eine Konstante oder der aktuelle Wert eines Einstellungs- oder Konfigurationsparameters zugewiesen werden kann.

Die Wahrheitsüberprüfung der Bedingung B_i erfolgt mittels der Wahrheitsfunktion $\beta(B_i)$, die anhand der in der jeweiligen Dialogsituation gültigen Operandenwerte folgende Ergebnisse liefert:

$$\beta(B_i) = \begin{cases} \texttt{true} & \text{falls } B_i \text{ erfüllt ist} \\ \texttt{false} & \text{falls } B_i \text{ nicht erfüllt ist} \end{cases}$$

Zur Darstellung komplexerer logischer Ausdrücke können mehrere Bedingungen mittels and/or-Verknüpfungen zusammengefasst werden. Dies erfolgt analog zu der oben beschriebenen Darstellung.

3.3 Maßnahmen zur Sicherung der Dialogqualität

Wie bereits mehrfach bemerkt, liegt ein wichtiger Aspekt bei der Entwicklung multimodaler Dialoge für das Fahrzeug in der für die Fahraufgabe geeigneten Benutzbarkeit. Hierfür dienen als Richtlinien der ISO-Normentwurf [ISO/DIS15005 2000] und das Statement of Principles [2000/53/EC 2000] (2.3.3). Da die formalen Beschreibungen der Modalitäten und des Dialogs keine Informationen über den jeweiligen Kontext – also Aussagen darüber, was und wie bedient wird – enthalten, können qualitätssichernde Maßnahmen ebenfalls nur weitgehend ohne inhaltlichen Bezug realisiert werden. Hierzu gehören in erster Linie Methoden zur formalen Verifikation (3.3.1) und Konsistenzsicherung (3.3.2) des Dialogs. Da das Anwendungsfeld des Werkzeugs und die Verwendung der semantischen Elemente eindeutig festgelegt sind, ist teilweise eine Unterstützung des Entwicklers bei der Einhaltung inhaltlicher Qualitätsmerkmale möglich (3.3.3). Sehr stark inhaltlich geprägte Forderungen der Richtlinien, wie beispielsweise den Fahrer nicht *„visuell zu unterhalten“* [2000/53/EC 2000], können allerdings nicht sichergestellt werden.

Basierend auf der streng formalen Beschreibung des Bediendialogs unterstützen die im Folgenden dargestellten Methoden den Entwickler bei der Einhaltung von Qualitätsaspekten im Sinne der entsprechenden Richtlinien.

3.3.1 Dialogverifikation

Grundlegend für eine sinnvolle Dialogausführung ist die Einhaltung formaler Kriterien in der Dialogdefinition. Der Entwickler muss im Falle eines Verstoßes darauf aufmerksam gemacht werden. Um unnötige und aufwändige Nacharbeiten zu vermeiden, erfolgt dies idealerweise bereits während der Dialogerstellung. Hierzu gehört eine *vollständige* und *eindeutige* Dialogbeschreibung. Dies ist eine fundamentale Voraussetzung, um die in [2000/53/EC 2000] genannten *„Prinzipien zur Interaktion mit den Anzeigen und Bedienelementen“* überhaupt erfüllen zu können.

3.3.1.1 Vollständigkeit

Der Entwickler wird auf Zustände aufmerksam gemacht, die nicht erreicht oder nicht verlassen werden können (Bild 32). Dadurch werden unvollständige Dialogdefinitionen vermieden, die zu einer Blockierung des Dialogablaufs oder zu einer Nichtverfügbarkeit bestimmter Funktionen führen.

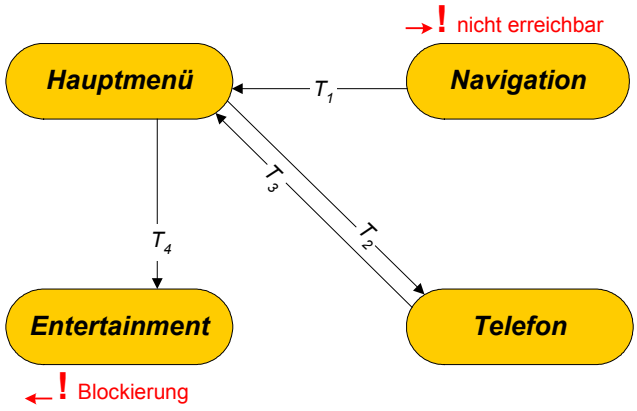


Bild 32 Unvollständige Dialogdefinitionen in den Zuständen *Navigation* und *Entertainment*

Dabei ist ein Zustand S_i genau dann nicht erreichbar, wenn keine Transition T existiert, die zu diesem Zustand führt.

$$\forall \mathrm{T}: \quad \mathrm{S}_{\mathrm{next}} \neq S_i \qquad \text{mit} \quad \left(T = \left[S_{next} \quad B\right]\right) \in \mathbf{T}$$

Ein Zustand S_i führt genau dann zu einer Blockierung des Dialogablaufs, wenn im Gültigkeitsbereich G_i dieses Zustands keine Transition definiert ist.

$$\forall \left[M \quad G_i\right]: \quad \tau\left(M \quad G_i\right) = \varnothing$$
$$\mathbf{T}_{MG_i} = \varnothing \qquad \text{mit} \quad M \in \mathbf{M}$$
$$G_i \in \left\{G \in \mathbf{G} \;\middle|\; G = S_i \;\vee\; G \in \vartheta_S^{\mathbf{c}}\left(S_i\right) \;\vee\; G = \varnothing\right\}$$

3.3.1.2 Eindeutigkeit

Um eine Nachricht $M \in \mathbf{M}$ in einem Gültigkeitsbereich $G \in \mathbf{G}$ als Auslöser für mehrere Transitionen verwenden zu können, müssen jeweils unterschiedliche Bedingungen definiert sein. Dadurch wird weitgehend sichergestellt, dass die Transitionen eines Gültigkeitsbereichs eindeutig sind. Dies ist Voraussetzung für einen determinierten Ablauf des Dialogs.

$$\forall \left[M \quad G\right]: \quad B_1 \neq B_2 \neq \ldots \neq B_n \neq \varnothing$$
$$\text{mit} \quad \tau\left(M \quad G\right) = \begin{Bmatrix} T_1 \\ T_2 \\ \ldots \\ T_n \end{Bmatrix} = \begin{Bmatrix} \left[S_{next} \quad B\right]_1 \\ \left[S_{next} \quad B\right]_2 \\ \ldots \\ \left[S_{next} \quad B\right]_n \end{Bmatrix} \quad \wedge \quad n > 1$$

Beispiel:

Die Transitionen T_1, T_2 und T_3 in Bild 33 werden von der gleichen Nachricht ausgelöst. Da keine einschränkenden Bedingungen definiert sind, ist im Zustand *Hauptmenü* nicht eindeutig festgelegt, welcher Zustandsübergang beim Eintreffen der Nachricht `msgNav` ausge-

löst werden soll. Bei den restlichen Transitionen ist die Kombination aus Nachricht und Gültigkeitsbereich jeweils eindeutig, sodass für diese keine Bedingungen erforderlich sind.

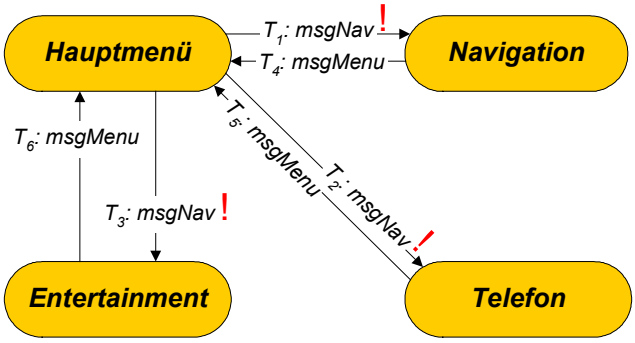
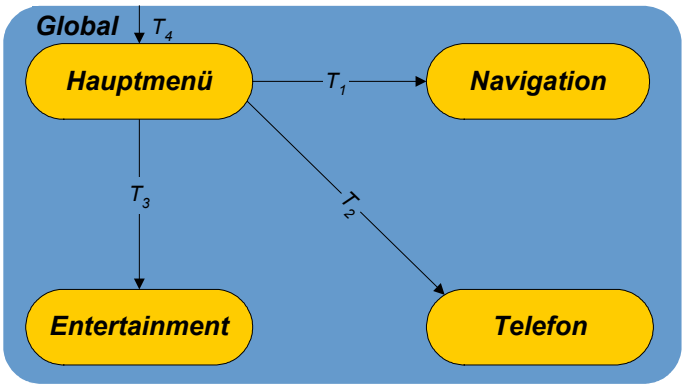


Bild 33 Nicht eindeutige Dialogdefinition im Zustand `Hauptmenü`

3.3.1.3 Erreichbarkeit und Entfernung zweier Zustände

Bei ansteigender Größe des Bediendialogs ist es zunehmend schwieriger zu entscheiden, ob von einem Zustand $S_{Start} \in \mathbf{S}$ aus ein anderer Zustand $S_{Dest} \in \mathbf{S}$ unter Verwendung bestimmter Modalitäten bzw. Dialogkomponenten $\mathbf{D'} \subseteq \mathbf{D}$ erreichbar ist (Bild 34). Für die Gebrauchstauglichkeit des Systems ist dabei neben der bloßen Erreichbarkeit auch die Entfernung dieser beiden Dialogzustände von Interesse, das heißt die minimale Zahl notwendiger Interaktionsschritte um von S_{Start} nach S_{Dest} zu gelangen.



S_{Start}	S_{Dest}	Distance
`Hauptmenü`	`Navi-gation`	1
`Enter-tainment`	`Navi-gation`	2

Bild 34 Erreichbarkeit und Entfernung von Zuständen

Der Dialogentwickler benötigt daher Unterstützung, um solche zustandsübergreifende Analysen und Verifikationen des Dialogs durchzuführen. Die *Distance* zwischen zwei Zuständen kann in Anlehnung an einer Breitensuche [Lang, *et al.* 1997] anhand der durch die Dialogdefinition festgelegten Transitionen berechnet werden.

- Hierzu erhält jeder Dialogzustand eine Markierung, die zu Beginn der Berechnung nicht gesetzt ist.

$$\forall S_i \in \mathbf{S}: \quad S_i^{Marker} := \texttt{false}$$

- Der Entfernungszähler wird initialisiert.

$$Distance := 0$$

- Die Zustandsliste $\mathbf{S'} \subseteq \mathbf{S}$ wird mit dem Startzustand S_{Start} initialisiert. Diese Hilfsgröße enthält je nach Iterationsschritt die jeweils betrachteten Zustände.

$$\mathbf{S'} := \{S_{Start}\}$$

- Folgende Schritte werden solange wiederholt, bis der Zielzustand in der Zustandsliste enthalten $S_{Dest} \in \mathbf{S'}$ oder die Zustandsliste leer $\mathbf{S'} = \varnothing$ ist.

  - Die Markierungen aller in der Zustandsliste enthaltener Zustände werden gesetzt.

$$\forall S_i \in \mathbf{S'}: \quad S_i^{Marker} := \texttt{true}$$

  - Extraktion derjenigen Zustände, die unter Verwendung der Dialogkomponenten **D'** von den in der Zustandsliste enthaltenen Zuständen aus erreicht werden können und deren Markierungen noch nicht gesetzt sind. Mit diesen wird die Zustandsliste überschrieben.

$$\mathbf{S'} := \left\{ S_{next} \in \mathbf{S} \;\middle|\; [S_{next} \quad B] \in \tau(\mathbf{M'} \quad \mathbf{G'}) \wedge S_{next}^{Marker} = \texttt{false} \right\}$$

$$\text{mit} \quad \mathbf{M'} := \left\{ M \in \mathbf{M} \;\middle|\; \vartheta_M^D(M) \in \mathbf{D'} \right\}$$
$$\mathbf{G'} := \left\{ G \in \mathbf{G} \;\middle|\; G \in \mathbf{S'} \vee G \in \vartheta_S^{\mathbf{C}}(\mathbf{S'}) \vee G = \varnothing \right\}$$

  - Der Entfernungszähler wird um eins erhöht.

$$Distance := Distance + 1$$

- Falls $S_{Dest} \in \mathbf{S'}$ ist, dann kann unter Verwendung der Dialogkomponenten **D'** auf mindestens einem Weg der Zielzustand S_{Dest} vom Startzustand S_{Start} aus erreicht werden. Der kürzeste Weg umfasst dabei die im Entfernungszähler *Distance* enthaltene Anzahl an Zustandsübergängen.

3.3.2 Dialogkonsistenz

Konsistenz ist eine generelle Anforderung an benutzungsgerechte HMIs. Ihre Sicherstellung ist insbesondere im Fahrzeug sehr wichtig, da der Nutzer in dieser Domäne zur Auflösung von Uneinheitlichkeiten unnötig Aufmerksamkeit von der Fahraufgabe abwenden muss (2.3.3). Neben der Festlegung des Dialogverhaltens existiert bei multimodalen Dialogen – im Gegensatz zu unimodalen – eine weitere für die Dialogkonsistenz relevante Dimension. Dies ist die Sicherstellung des sinnvollen und konsistenten Zusammenwirkens der einzelnen Modalitäten. Der in

dieser Arbeit betrachtete Ansatz unterstützt den Dialogentwickler bei der Realisierung konsistenter Dialoge im Sinne dieser beiden Aspekte in folgender Weise.

3.3.2.1 Definition und Modifikation des Dialogverhaltens

Die hierarchische Strukturierungsmöglichkeit (3.2.7) ist das wichtigste Element zur Sicherstellung konsistenten Dialogverhaltens in verschiedenen Dialogsituationen. Durch die Verwendung von Gültigkeitsbereichen für Transitionen, Reaktionen und Parameterwerten sowie der Zuordnung der Dialogzustände zu Kontexten kann in unterschiedlichen Dialogzuständen mit einer einzigen Definition identisches Verhalten sichergestellt werden. Insbesondere Modifikationen sind dadurch in einer konsistenten Art und Weise mit sehr geringem Aufwand durchführbar. So kann einem neuen Dialogzustand bereits bestehendes Verhalten hinzugefügt werden, indem er lediglich den entsprechenden Kontexten zugeordnet wird. Gleichzeitig müssen Änderungen am Dialogverhalten nur ein einziges Mal durchgeführt werden, um sich auf alle relevanten Dialogzustände auszuwirken. Die bei nicht hierarchisch strukturierbaren Dialogbeschreibungen erforderlichen fehleranfälligen und unter Umständen inkonsistenten Mehrfachdefinitionen werden dadurch vermieden.

Beispiel:

Durch die Zuordnung des neu definierten Zustands `Telefon` in Bild 35 zum Kontext C_1 ist die Transition T_4 automatisch auch in diesem Zustand wirksam. Dies gilt auch für die globale Reaktion R_1. Eventuell während der Dialogentwicklung notwendig werdende Modifikationen an T_4 oder R_1 müssen jeweils nur einmal gemacht werden um sich auf alle relevanten Dialogzustände auszuwirken.

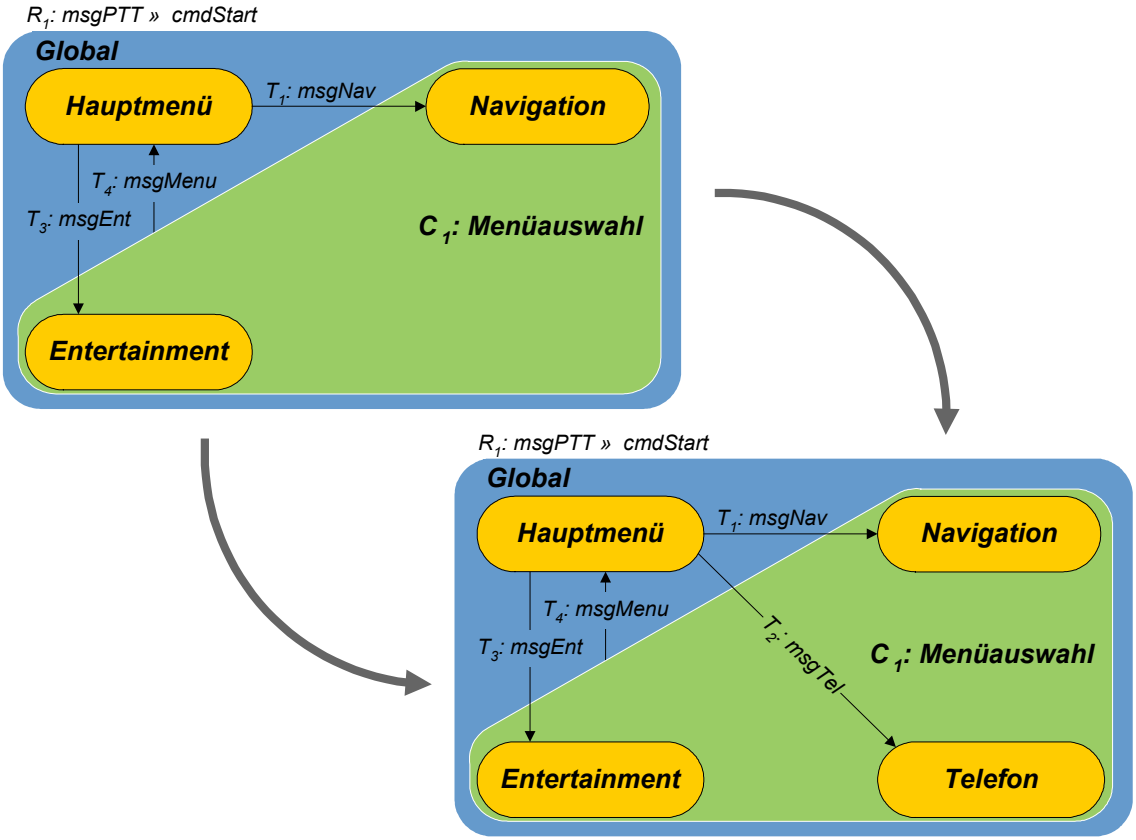


Bild 35 Konsistentes Dialogverhalten durch hierarchische Gliederung der Dialogbeschreibung

3.3.2.2 Konfiguration der Dialogkomponenten

Daneben müssen für eine konsistente Zusammenwirkung die verschiedenen Dialogkomponenten aufeinander abgestimmt werden. Häufig treten aber genau an dieser Stelle Probleme auf, weil beispielsweise für jede Modalität eine eigene Konfigurationsdatei erforderlich ist. Diese verschiedenen Dateien können normalerweise nur mit einem erheblichen manuellen Synchronisationsaufwand konsistent gehalten werden. Die in der kontextfreien Beschreibung der Modalitäten (3.1) enthaltenen Konfigurationsmöglichkeiten ermöglichen daher eine zentrale Einflussnahme auf das Verhalten der einzelnen Dialogkomponenten. So können mit Hilfe der Konfigurationsparameter sowie der parametrisierbaren Kommandos und Nachrichten beispielsweise das Vokabular des Spracherkenners mit den Sprachausgaben und bestimmten Displayanzeigen innerhalb der gemeinsamen Oberfläche des Entwicklungswerkzeugs abgestimmt und bei Bedarf modifiziert werden (4.2.2.2). Fehleranfällige dezentrale Verfahren werden dadurch vermieden.

Dies unterstützt ein zusätzlicher Mechanismus, bei dem jedem Dialogzustand charakteristische Schlüsselwörter zugeordnet werden (3.2.1). Auf diese Schlüsselwörter kann dann bei der Definition von Parameterwerten zugegriffen werden, wodurch die konsistente Verwendung der Begriffe und Bezeichner zwischen den Dialogkomponenten zusätzlich erleichtert wird.

Beispiel:

Bild 36 verdeutlicht hierzu, wie bei einer dezentralen Konfiguration der Dialogkomponenten in verschiedenen Dateien Inkonsistenzen entstehen, die bei einem zentralen Ansatz sehr leicht vermieden werden können.

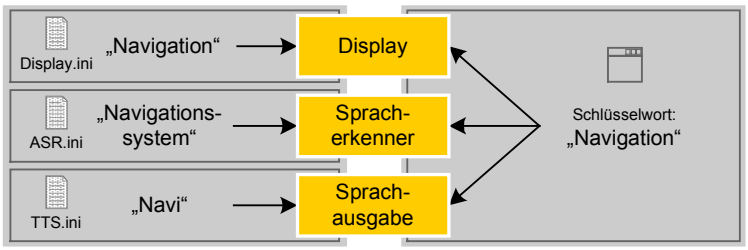


Bild 36 Konfiguration der Dialogkomponenten: dezentral (links) in unterschiedlichen Dateien und zentral (rechts) im Entwicklungswerkzeug

3.3.3 Dialoginhalt

Für ein allgemein verwendbares Werkzeug ist es nicht sinnvoll, Annahmen über den Inhalt eines speziellen Dialogs zu machen. Es darf also keine Rolle spielen, welches TICS wie bedient wird. Im Gegensatz zu den oben genannten Methoden, die den Entwickler bei der Einhaltung von formal prüfbaren Merkmalen unterstützen, ist die Sicherstellung inhaltlicher Qualitätsaspekte somit nicht direkt umsetzbar. Für umfassende inhaltliche Evaluationen müssen die in 2.4 genannten Methoden herangezogen werden. Trotzdem können durch eine sinnvolle Verwendung der zur Verfügung stehenden Mittel bereits bei der Dialogerstellung inhaltliche Qualitätsmerkmale berücksichtigt werden. Dadurch kann von Beginn des Entwicklungsprozesses an ein zusätzlicher positiver Einfluss auf die Gebrauchstauglichkeit des Systems ausgeübt werden.

3.3.3.1 Interaktionslänge

Um dem Benutzer eine gute Kontrolle über den Dialogablauf zu ermöglichen, sind unterbrechbare Interaktionssequenzen von angemessener Länge [2000/53/EC 2000] eine Forderung an Bediendialoge im Fahrzeug. Da normalerweise jeder Interaktionsschritt zwischen Benutzer und System von einer Transition begleitet wird, kann die als *Distance* (3.3.1.3) berechnete Entfernung für den Start- und Endzustand einer Aufgabe als Abschätzung für die zugehörige Interaktionslänge benutzt werden. Somit steht bereits sehr früh im Entwicklungsprozess eine Kennzahl zur Verfügung, anhand der konzeptuelle Entscheidungen bezüglich Interaktionsablauf und Bedienlogik diskutiert und getroffen werden können.

3.3.3.2 Einfachheit und Verständlichkeit der Ausgaben

Eine weitere Anforderung an fahrzeugtaugliche Bedienkonzepte stellen einfache und verständliche Systemausgaben dar. Die in Verbindung mit der konsistenten Konfiguration der Dialogkomponenten (3.3.2.2) bereits erwähnten Schlüsselwörter (3.2.1) bieten hierzu eine sinnvolle Hilfestellung, da auf diese einmal festgelegten Begriffe während der gesamten Dialogdefinition immer wieder zurückgegriffen werden kann. Die so erreichbare konsistente Verwendung des

Vokabulars begünstigt ein gutes Verständnis beim Benutzer. Gleichzeitig kann bei kritischen Parametern die maximale Zeichenlänge auf eine bestimmte Anzahl an Zeichen begrenzt werden. Dies ermöglicht die Berücksichtigung geometrischer Restriktionen, wie sie beispielsweise aufgrund begrenzter Anzeigeflächen im Display auftreten. Dadurch können Verständnisprobleme vermieden werden, die etwa aus abgeschnittenen Zeichen oder einer zu umfangreichen Darstellung resultieren.

3.3.3.3 Geringe Ablenkung von der Fahraufgabe

Ein allgemeines Gestaltungsprinzip für TICS ist die Vermeidung von Inkompatibilitäten zwischen Bedien- und Fahraufgabe. Dabei ist, wie auch in 3.3.2 beschrieben wird, ein widerspruchsfreier Bedienablauf grundlegende Voraussetzung, um eine möglichst geringe Ablenkung von der Fahraufgabe zu erzielen. Deshalb stehen dem Dialogentwickler mit der hierarchischen Strukturierung des Dialogs und der zentralen Konfiguration der Dialogkomponenten verschiedene Werkzeuge zur Sicherung der Dialogkonsistenz zur Verfügung.

3.4 Informationsfusion – Informationsfission

Wie in 2.2.3.1 dargestellt wird, ist die Fusion der Informationen aus den Dialogkomponenten auf semantischer Ebene die sinnvollste Methode bei multimodalen MMS. Die Erkennungsvorgänge können dabei voneinander unabhängig ablaufen, was eine funktionale Entkopplung der Dialogkomponenten erlaubt. Dadurch ist ein modularer Aufbau des Gesamtsystems möglich. Dabei muss die technische Realisierung dieser semantischen Fusion die Leistungsfähigkeit besitzen, um den im Fahrzeug vorzufindenden Bedienweisen gerecht zu werden. Gleichzeitig sollte sie aber auch mit den Anforderungen an das Entwicklungswerkzeug harmonieren und beispielsweise den Durchgriff auf die Dialogkomponenten zur zentralen Konfiguration und Steuerung oder die Durchführung dialogqualitätssichernder Maßnahmen nicht behindern. Im umgekehrten Fall treffen zur Verteilung der Informationen vom Dialogmanagement auf die Dialogkomponenten die selben Anforderungen auch für die Informationsfissionskomponente zu. Daher sind in dieser Arbeit sowohl die Fusions- als auch die Fissionskomponente eng mit dem Dialogmanagement und damit auch mit der Dialogbeschreibung verbunden (Bild 37).

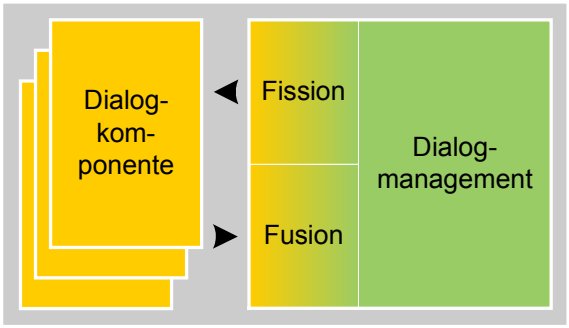


Bild 37 Enge Verbindung von Fusion und Fission mit Dialogmanagement

Die verschiedenen in 2.2.3.1 beschriebenen Fusionsverfahren kapseln die Informationen aus den Modalitäten vom Dialogmanagement ab. Sie bilden eine Vorverarbeitungsstufe, in der die Nutzerintention auf ein semantisches Niveau normalisiert und dann an das Dialogmanagement weitergegeben wird (Bild 13). Dieses Vorgehen ist insbesondere zur Realisierung synergistisch multimodaler Systeme im Stile von *„Put that there“* [Bolt 1980] notwendig, weil es die gemeinsame Interpretation zeitlich nah zusammen liegender Informationen aus unterschiedlichen Modalitäten erlaubt. Dies erfordert aber in der Fusionskomponente eine Zuordnung der Informationen aus den einzelnen Modalitäten zu dem abstrakten Datenmodell, das vom Dialogmanagementmodul verarbeitet wird. Dadurch werden die Modalitäten weitgehend von der Dialogbeschreibung entkoppelt. Dies ermöglich zwar einen einfachen Austausch einzelner Modalitäten, die Parametrisierung und Steuerung der Modalitäten, die Informationsfusion und der Bedienablauf stellen aber dann drei voneinander getrennte Arbeitsbereiche dar, die von einem Entwicklungswerkzeug kaum abgedeckt werden können. Insbesondere die Maßnahmen zur Sicherung der Dialogqualität sowie die zentrale Konfiguration und Steuerung der Dialogkomponenten werden dadurch erschwert. Bei einer Vergegenwärtigung der Erkenntnisse aus 2.2.6 ergeben sich im Fahrzeug allerdings veränderte Voraussetzungen. Im Vergleich zu anderen Domänen spielt nämlich hier synergistische Multimodalität aufgrund der Sekundärbedienung der betrachteten Systeme keine bestimmende Rolle. Die Nutzer geben während der Fahrt Informationen hauptsächlich sequentiell ab, sodass eine alternierend multimodale Bedienbarkeit für TICS im Allgemeinen ausreichend ist. Aufgrund dieser besonderen Randbedingungen wird deshalb in dieser Arbeit ein kombiniertes Fusions- und Fissionsverfahren eingesetzt, das weniger auf synergistisch multimodale Bedienung eingeht und dafür eine engere Kopplung zwischen Dialogmanagement und Dialogkomponenten erlaubt. Dadurch kann ein in sich schlüssiger Bediendialog leichter sichergestellt werden (3.3).

Informationen werden von den Dialogkomponenten mittels Nachrichten übertragen (3.1.4). Aufgrund der sequentiellen Bedienung durch den Nutzer kann die physikalische Zusammenführung dieser Nachrichten mit Hilfe eines Multiplexers einfach realisiert werden (Bild 38).

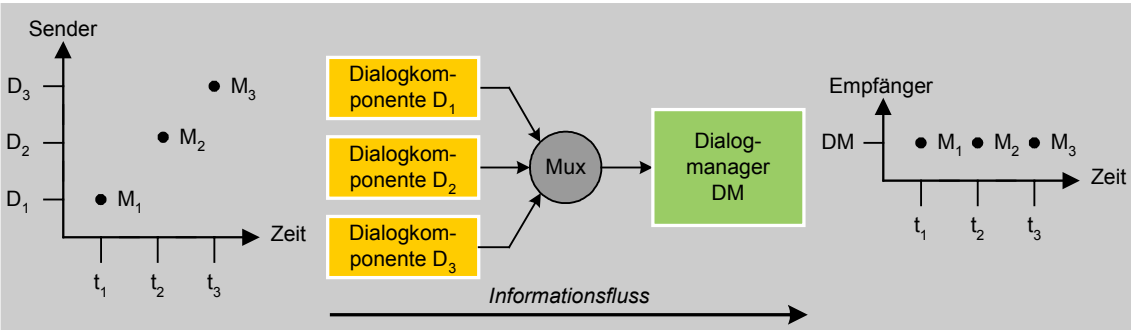


Bild 38 Fusionsmechanismus – Zusammenführung der Nachrichten M_i durch einen Multiplexer zum Dialogmanager

Die nach ihrem Auslösezeitpunkt geordneten Nachrichten werden dem Dialogmanager ohne weitere Verarbeitung direkt zugeführt. Sie lösen dort entsprechend der aktuellen Dialogsituation Reaktionen oder Transitionen aus (3.5.3). Die semantische Fusion findet somit direkt auf der

Nachrichtenebene in der Dialogbeschreibung statt. Dies bedeutet, dass für jede mögliche Nachricht aus den unterschiedlichen Dialogkomponenten die entsprechenden Transitionen und Reaktionen definiert werden müssen. In Verbindung mit den verschiedenen Konfigurationsmöglichkeiten (3.1) kann man somit zentral im Entwicklungswerkzeug die Dialogkomponenten kontrollieren und ihr Verhalten je nach Dialogsituation genau festlegen (4.2.2). Die hierzu unter Umständen zusätzlich notwendigen Transitionen und Reaktionen bleiben aufgrund der verschiedenen Gültigkeitsbereiche (3.2.7) und einer entsprechenden Visualisierung im Werkzeug überschaubar.

Die Informationsfission zur Verteilung der Informationen vom Dialogmanager auf die verschiedenen Dialogkomponenten wird in analoger Weise realisiert. Der Dialogentwickler kann mit den Kommandos das Verhalten jeder Dialogkomponente in der jeweiligen Dialogsituation festlegen. Während der Dialogausführung erzeugt der Dialogmanager die entsprechenden Kommandos, die dann mit Hilfe eines Demultiplexers der jeweiligen Dialogkomponente zugeordnet werden und dort das gewünschte Verhalten auslösen (Bild 39).

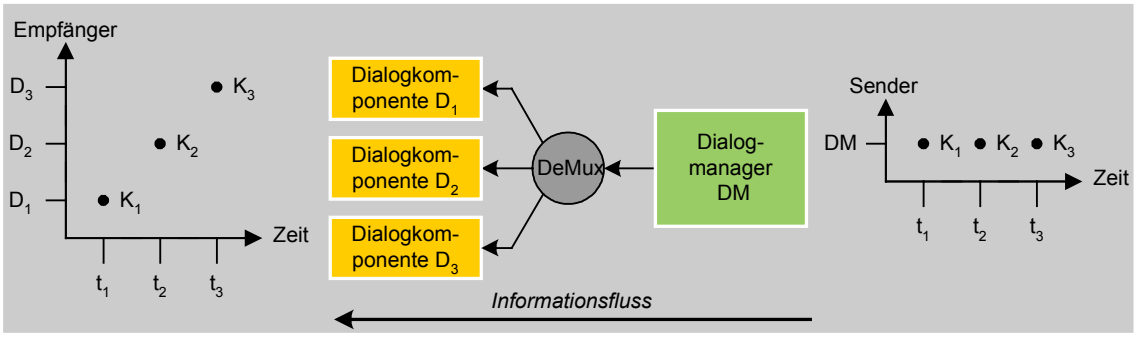


Bild 39 Fissionsmechanismus – Verteilung der Kommandos K_i durch einen Demultiplexer auf die verschiedenen Dialogkomponenten

3.5 Formale Beschreibung der Dialogausführung

Unter zu Hilfenahme der obigen formalen Darstellungen der Dialogkomponenten und des Bediendialogs wird nun die Heuristik vorgestellt, auf deren Basis der Dialogmanager (4.3.1) den Dialogablauf kontrolliert.

Beim Dialogstart wechselt das Dialogmanagement in den Zustand *Bereit* (Bild 40). Es ist dann in der Lage auf bestimmte Ereignisse zu reagieren. Dabei handelt es sich um eintreffende Nachrichten von angemeldeten Dialogkomponenten. Aber auch die Dialoginitialisierung, das Dialogende sowie das An- und Abmelden von Dialogkomponenten fallen unter diese Ereignisse.

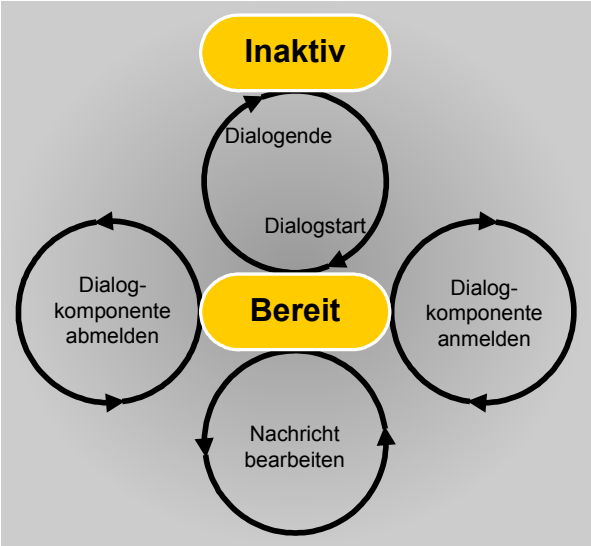


Bild 40 Zustandsmodell des Dialogmanagements

3.5.1 Dialogstart

Beim Starten der Dialogausführung findet ein Zustandswechsel von *Inaktiv* nach *Bereit* statt. Hierbei müssen einige Maßnahmen zur Dialoginitialisierung vorgenommen werden:

1. Zurücksetzen der Dialogkomponenten
   Da zu Beginn des Dialogablaufs noch keine Dialogkomponente angemeldet sein kann, muss sichergestellt sein, dass im Dialogmanagement alle Dialogkomponenten abgemeldet sind.

$$\forall D_i \in \mathbf{D} \quad \text{gilt}: \ \vartheta_{\mathrm{D}}^{\mathrm{Status}}(D_i) = \texttt{inactive}$$

2. Auffinden des Startzustandes
   Per Konvention beginnt der Bediendialog stets im obersten Zustand der visuellen Repräsentation (4.2.2). Dieser Zustand ist dadurch gekennzeichnet, dass er keinen Elternzustand S_{Parent} besitzt (3.2.1). Beim Start des Dialogs muss dieser einmalige Zustand S_{Start} identifiziert werden.

$$S_{Start} := S_i \qquad \text{mit} \quad \exists_1 S_i : \ \vartheta_{\mathrm{S}}^{\mathrm{S_{Parent}}}(S_i) = \varnothing$$

3. Zustandswechsel in den Startzustand
   Um den Bediendialog in eine definierte Ausgangsposition zu bringen erfolgt nun ein Zustandswechsel (3.5.6) in den Startzustand mit:

$$S_{next} := S_{Start}$$

3.5.2 Dialogkomponente anmelden

Zur dynamischen Integration neuer Dialogkomponenten in den Dialogablauf müssen folgende Teilschritte durchgeführt werden.

1. Zurücksetzen der Statusparameter
   Um zu verhindern, dass eine frühere Dialogausführung Auswirkungen auf den aktuellen Dialogablauf hat, werden bei der Neuanmeldung einer Dialogkomponente D_i alle eventuell vorhandenen Werte der zugehörigen Statusparameter gelöscht. Hierzu wird bei allen Statusparametern $\mathbf{P}_{D_i}^{\mathbf{DC\_State}}$ der Dialogkomponente D_i

   $$\mathbf{P}_{D_i}^{\mathbf{DC\_State}} := \left\{ P \in \mathbf{P} \;\middle|\; \vartheta_P^D(P) = D_i \;\wedge\; \vartheta_P^{P_{Usage}}(P) = \texttt{DC\_State} \right\}$$

   sichergestellt, dass die zugehörigen Parameterwerte

   $$\mathbf{P}_{\mathbf{Value}_{D_i}}^{\mathbf{DC\_State}} := \upsilon\left(\mathbf{P}_{D_i}^{\mathbf{DC\_State}} \quad \varnothing \right)$$

   gelöscht werden.

2. Vorkompilierung der entwicklerdefinierten Nachrichten
   Nun werden die entwicklerdefinierten Nachrichten (3.1.4), die vor dem Start der Dialogausführung angelegt werden müssen, in der neuen Dialogkomponente D_i bekannt gemacht. Hierzu muss für jede der entsprechenden Nachrichten $\mathbf{M}_{D_i}^{\mathbf{DdMessage\ precompile}}$ über den Kommandoaufruf K_{Call_M} das zugehörige Kommando K_M und der entsprechende Parametersatz $\mathbf{P}_{\mathbf{Value}_M}$ ermittelt werden. Gemeinsam mit dem Namen der Nachricht werden diese an die Dialogkomponente D_i zur Verarbeitung übertragen.

   $$\forall M \in \mathbf{M}_{D_i}^{\mathbf{DdMessage\ precompile}}:$$
   $$\begin{aligned} \left[K \quad \mathbf{P}_{\mathbf{Value}} \right]_M &= \kappa\left(K_{Call_M} \right) \\ &= \kappa\left(\vartheta_M^{K_{Call}}(M) \right) \end{aligned}$$

   mit

   $$\mathbf{M}_{D_i}^{\mathbf{DdMessage\ precompile}} := \left\{ M \in \mathbf{M} \;\middle|\; \vartheta_M^D(M) = D_i \;\wedge\; \vartheta_M^{M_{Type}}(M) = \texttt{DdMessage precompile} \right\}$$

3. Im Dialogmanagement aktivieren
   Um die neue Dialogkomponente D_i in das Dialogmanagement aufzunehmen, wird sie als aktiv gekennzeichnet:

   $$\vartheta_D^{Status}(D_i) := \texttt{active}$$

4. Zustandswechsel in den aktuellen Zustand
   Die Synchronisierung der neuen Dialogkomponente mit den restlichen Dialogkomponenten und der aktuellen Dialogsituation wird durch einen erneuten Zustandswechsel (3.5.6) in den aktuellen Dialogzustand erreicht.

$$S_{next} := S_{Act}$$

3.5.3 Eintreffende Nachricht bearbeiten

Während der Dialogausführung muss jede von einer Dialogkomponente D_{in} eintreffende Nachricht M_{in} entsprechend der jeweils zugrunde liegenden Dialogdefinition bearbeitet werden. Hierzu sind folgende Teilschritte erforderlich:

1. Statusparameter aktualisieren
   Mit jeder Nachricht kann eine Dialogkomponente aktuelle Werte für eine beliebige Anzahl ihrer Statusparameter mitschicken (4.3.2). Damit diese nicht verloren gehen und als Grundlage für die weitere Dialogführung verwendet werden können, müssen sie im Dialogmanager gespeichert werden. Zur Identifizierung der einzelnen Parameter in der Nachricht wird neben dem jeweiligen Wert auch der zugehörige Name $P_{in} \in \left\{ P \in \mathbf{P} \,\middle|\, \vartheta_P^D(P) = D_{in} \right\}$ übermittelt. Für jeden so empfangenen Wert V_{in} erfolgt eine Aktualisierung des entsprechenden Parameterwertes im Dialogmanager.

$$P_{Value} := \left[V_{in} \quad \varnothing \right] \quad \text{mit}$$
$$\exists_1 P_{Value} : \quad P_{Value} \in \left(\mathbf{P}_{\mathbf{Value}\, P_{in}\varnothing} = \upsilon(P_{in} \quad \varnothing) \right)$$

2. Reaktionen überprüfen
   Im nächsten Schritt werden die für die Nachricht M_{in} im aktuellen Gültigkeitsbereich $\mathbf{G_{Act}}$ definierten Reaktionen herausgefiltert:

$$\mathbf{R}_{M_{in}\mathbf{G_{Act}}} = \rho(M_{in} \quad \mathbf{G_{Act}})$$

   Dann muss für

$$\forall R_i \in \mathbf{R}_{M_{in}\mathbf{G_{Act}}} \quad \text{mit} \quad R_i = \left[\mathbf{K_{Call}} \quad B \right]_i$$

   die Bedingung $\beta(B_i)$ überprüft werden. Falls diese erfüllt ist, werden die den Kommandoaufrufen $\mathbf{K}_{\mathbf{Call}_i} \subseteq \mathbf{K_{Call}}$ zugeordneten Kommandos sowie die entsprechenden Parametersätze an die zugehörigen Dialogkomponenten verschickt.

   Anmerkung:

   Falls in diesem Zustand mehrere Reaktionen mit unterschiedlichen Gültigkeitsbereichen ausgeführt werden müssen, dann wird die in 3.2.9.2 beschriebene Bearbeitungsreihenfolge verwendet.

3. Transitionen überprüfen
   Schließlich muss noch überprüft werden, ob für die Nachricht M_{in} im aktuellen Gültigkeitsbereich $\mathbf{G_{Act}}$ Transitionen definiert sind.

$$\mathbf{T}_{M_{in}\mathbf{G_{Act}}} = \tau(M_{in} \quad \mathbf{G_{Act}})$$

   Analog zu den Reaktionen wird für

$$\forall T_i \in \mathbf{T}_{M_{in}\mathbf{G_{Act}}} \quad \text{mit } T_i = [S_{next} \quad B]_i$$

   die Bedingung $\beta(B_i)$ überprüft und darauf basierend ein Wechsel zum Zustand S_{next} ausgelöst oder nicht.

   Anmerkung:
   Falls in diesem Zustand mehrere Transitionen mit unterschiedlichen Gültigkeitsbereichen gültig sind, also kein eindeutiges T_i existiert, dann wird der in 3.2.9.1 beschriebene Verdeckungsmechanismus angewandt.

3.5.4 Dialogkomponente abmelden

Wenn sich eine Dialogkomponente D_i aus dem Dialogablauf abmeldet, muss sie im Dialogmanagement deaktiviert werden.

$$\vartheta_D^{Status}(D_i) := \texttt{inactive}$$

3.5.5 Dialogende

Beim Beenden des Dialogs werden alle Dialogkomponenten im Dialogmanagement abgemeldet.

$$\forall D_i \in \mathbf{D} \quad \text{gilt}: \ \vartheta_D^{Status}(D_i) = \texttt{inactive}$$

3.5.6 Zustandswechsel

Der Zustandswechsel ist ein für die Dialogsteuerung elementar wichtiger Vorgang. Da sich je nach Dialogzustand das Verhalten der Dialogkomponenten grundsätzlich unterscheiden kann, müssen sie bei jedem Zustandswechsel neu konfiguriert bzw. parametrisiert werden. Für ein definiertes Dialogverhalten ist es dabei wichtig, dass der Wechsel schnell und synchron bei allen Dialogkomponenten abläuft. Im Folgenden werden die notwendigen Teilschritte für einen Wechsel in den Zustand S_{next} dargestellt. Die Beschreibung erfolgt unabhängig von dem spezifischen Auslöser (siehe oben).

1. Auslösen der Nachricht `LeaveState`
   Da die Syntax der Dialogbeschreibung vorsieht, durch das Verlassen eines Zustandes bestimmte Reaktionen auszulösen, schickt der Dialogmanager zunächst eine entsprechende Nachricht `LeaveState` an sich selbst und bearbeitet diese entsprechend 3.5.3. Diese Nachricht kann nur Reaktionen auslösen, da andernfalls die gerade bearbeitete Transition ad absurdum geführt werden würde.

2. Festlegung des aktuellen Dialogzustandes
   Je nach dem ob zu einem beliebigen nächsten oder zu dem in der Dialoghistorie vorhergehenden Zustand gewechselt werden soll, müssen bei der Festlegung des aktuellen Dialogzustandes S_{Act} zwei Fälle unterschieden werden:

   - $S_{next} \neq$ `xxxLastStatexxx`
     Es erfolgt ein Wechsel zu einem beliebigen nächsten Zustand. Hierzu wird zunächst der eben noch aktuelle Dialogzustand S_{Act} in der Dialoghistorie $\mathbf{S_{Hist}}$ gespeichert (3.2.5) und dann mit dem neuen Zielzustand aktualisiert.

     $$S_{Act} := S_{next}$$

   - $S_{next} =$ `xxxLastStatexxx`
     Der Wechsel erfolgt zu dem einen Schritt in der Dialoghistorie zurückliegenden Zustand. Der aktuelle Zustand S_{Act} muss deshalb auf den ersten Eintrag der Dialoghistorie S_1 (3.2.5) gesetzt werden.

     $$S_{Act} := S_1 \quad \text{mit } S_1 \in \left(\mathbf{S_{Hist}} = [S_1 \quad S_2 \quad S_3 \quad ... \quad S_n]\right)$$

     Falls in der Dialoghistorie kein Eintrag enthalten ist, kann diese Zustandswechsel nicht ausgeführt werden.

3. Einleitung des Zustandswechsels in den Dialogkomponenten
   Wie unten gezeigt wird, sind zur Realisierung eines Zustandswechsels mehrere Nachrichten erforderlich. Um ein definiertes Verhalten der Dialogkomponenten während dieses Vorgangs sicher zu stellen, wird der Zustandswechsel durch eine besondere Nachricht `StateChange` angekündigt. Gleichzeitig mit dieser Nachricht wird der neue aktuelle Dialogzustand S_{Act} übermittelt. Falls notwendig, kann die jeweilige Dialogkomponente nun in einen für den Zustandswechsel erforderlichen Modus wechseln.

4. Übermittlung der aktuellen Werte für die Konfigurationsparameter
   Um die durch den Zustandswechsel notwendigen Modifikationen der Einstellungen durchzuführen, werden für jede aktive Dialogkomponente

   $$\forall D_i \in \mathbf{D} \quad \text{mit } \vartheta_D^{Status}(D_i) = \texttt{active}$$

die jeweils zugehörigen Konfigurationsparameter

$$\mathbf{P}_{D_i} := \left\{ P \in \mathbf{P} \;\middle|\; \vartheta_P^D(P) = D_i \;\wedge\; \vartheta_P^{P_{Usage}}(P) = \texttt{DC\_Setting} \right\}$$

mit den im neuen Dialogzustand gültigen Parameterwerten übertragen. Hierzu sind die im aktuellen Gültigkeitsbereich $\mathbf{G_{Act}}$ möglichen Parameterwerte $\mathbf{P}_{\mathbf{Value}_{P_j \mathbf{G_{Act}}}}$ erforderlich.

$$\forall P_j \in \mathbf{P}_{D_i} : \qquad \mathbf{P}_{\mathbf{Value}_{P_j \mathbf{G_{Act}}}} = \upsilon(P_j \quad \mathbf{G_{Act}})$$

Von diesen findet derjenige Wert V_k Verwendung, bei dem die zugehörige Bedingung $\beta(B_k)$ erfüllt ist.

$$\forall P_{Value_k} \in \mathbf{P}_{\mathbf{Value}_{P_j G_{Act}}} \quad \text{mit} \quad P_{Value_k} = \left[V \quad B \right]_k$$

Anmerkung:

Falls in diesem Zustand mehrere Parameterwerte mit unterschiedlichen Gültigkeitsbereichen gültig sind, also kein eindeutiges P_{Value} existiert, dann wird der in 3.2.9.1 beschriebene Verdeckungsmechanismus angewandt.

5. Aktivierung der aktivierbaren Nachrichten
   Als Nächstes wird in jeder verfügbaren Dialogkomponente die Bereitstellung derjenigen aktivierbaren Nachrichten $\mathbf{M_{Act}}$ angestoßen, die im aktuellen Gültigkeitsbereich $\mathbf{G_{Act}}$ Transitionen oder Reaktionen auslösen können.

$$\forall D_i \in \mathbf{D} \quad \text{mit} \quad \vartheta_D^{Status}(D_i) = \texttt{active}$$

$$\mathbf{M_{Act}} := \left\{ M_i \in \mathbf{M} \;\middle|\; \vartheta_M^D(M_i) = D_i \quad \wedge \right.$$
$$\vartheta_M^{M_{Act}}(M_i) = \texttt{activatable} \quad \wedge$$
$$\left. \left(\tau(M_i \quad \mathbf{G_{Act}}) \neq \varnothing \;\vee\; \rho(M_i \quad \mathbf{G_{Act}}) \neq \varnothing \right) \right\}$$

   Da die Randbedingungen zur Auslösung dieser aktivierbaren Nachrichten über entsprechende Parameter verändert werden können, müssen für $\forall M_i \in \mathbf{M_{Act}}$ die jeweils gültigen Parameterwerte der zugehörigen Parameter $\mathbf{P}_{M_i} := \vartheta_M^{\mathbf{P}}(M_i)$ mit übertragen werden.

$$\forall P \in \mathbf{P}_{M_i} \qquad \exists_1 P_{Value} : \quad P_{Value} \in \left(\mathbf{P}_{\mathbf{Value}_{P \mathbf{G_{Act}}}} = \upsilon(P \quad \mathbf{G_{Act}}) \right)$$

6. Beenden des Zustandswechsels in den Dialogkomponenten
   Die Dialogkomponenten werden über das Ende des Zustandwechsels durch die Nachricht `StateChangeEnd` informiert und können daraufhin in einen eventuell notwendigen Bereitschaftsmodus wechseln.

7. Auslösen der Nachricht `EnterState`
   Die Syntax der Dialogbeschreibung sieht vor, dass beim Betreten eines Zustandes Reaktionen ausgelöst werden können. Deshalb schickt der Dialogmanager die Nachricht `EnterState` an sich selbst und bearbeitet diese entsprechend 3.5.3.

Kapitel 4 Referenzimplementierung – Das Werkzeug EmMI

Um die Realisierbarkeit und Tauglichkeit des in Kapitel 3 dargestellten theoretischen Konzepts zu demonstrieren, wird es beispielhaft implementiert. Das bei dieser Umsetzung entstehende Werkzeug *„EmMI“* (**E**nvironment for **m**ultimodal Human-**M**achine-**I**nterfaces) ist als Referenz- und Experimentiersystem zu verstehen.

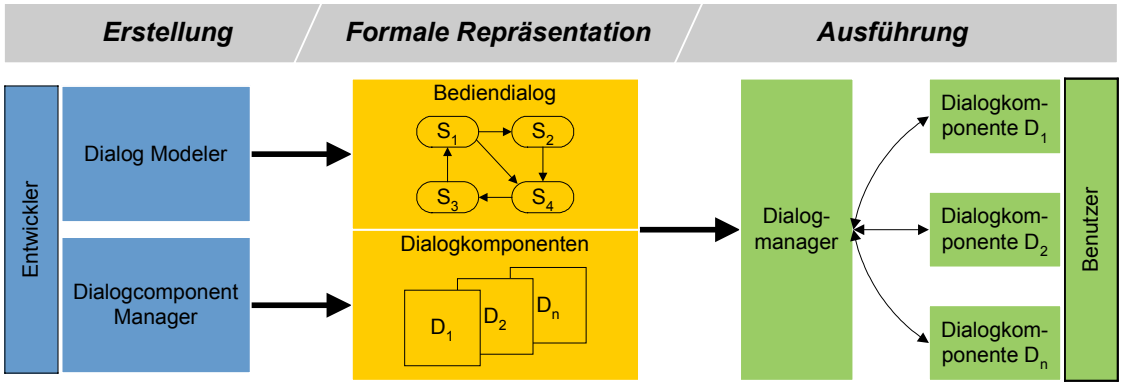


Bild 41 Struktur und Wirkungsgefüge EmMI (Environment for multimodal Human-Machine-Interfaces)

Die Struktur und das Wirkungsgefüge der einzelnen Bestandteile von EmMI sind in Bild 41 dargestellt. Deutlich können die drei Hauptbereiche *Erstellung*, *Formale Repräsentation* und *Ausführung* erkannt werden. Sie spiegeln die Kernmodule wieder, die bei der prototypischen Implementierung von Bediendialogen durch das Werkzeug bereitgestellt werden müssen. Dabei bildet die geeignete formale Beschreibung des Bediendialogs und der Dialogkomponenten das Fundament für die weiteren Module. Darauf basierend erstellt der Entwickler den speziellen Bediendialog. EmMI unterstützt ihn bei dieser Spezifikation und hilft ihm zugleich relevante Aspekte der Dialogqualität zu beachten. Für eine fundierte Bewertung der Gebrauchstauglichkeit muss der Bediendialog aber tatsächlich erlebbar gemacht werden. Hierzu können mit Hilfe eines universell verwendbaren Dialogmanagers beliebige Dialoge ausgeführt werden. Dieser Dialogmanager interpretiert dabei die auf der formalen Beschreibung beruhenden Dialogdefinitionen und kontrolliert entsprechend den Dialogablauf. Gleichzeitig begünstigen die starke Modularisierung und die zentrale Konfigurationsmöglichkeit eine große Flexibilität und einen hohen Wiederverwendbarkeitsgrad der Dialogkomponenten.

Im Folgenden werden diese drei Hauptbereiche und ihr jeweiliger Bezug zur formalen Repräsentation in Kapitel 3 dargestellt.

4.1 Formale Repräsentation

Die Beschreibung der Dialogkomponenten und des Bediendialogs basiert auf den in 3.1 und 3.2 eingeführten formalen Darstellungen. Die informationstechnische Umsetzung des theoretischen Modells erfolgt mittels der relationalen SQL[1]-Datenbank MS-Access 2000 [Spona & Radke 1999]. Die Verwendung einer Datenbank als Grundlage zur Spezifikation des Bediendialogs resultiert aus folgenden Motiven:

- Die mengenalgebraischen Darstellungen der formalen Repräsentation in Kapitel 3 lassen sich ausgezeichnet auf Tabellen und Abfragen abbilden.
- Die formale Repräsentation muss zwingend auf die strukturierte Syntax der Datenbank abgebildet werden. Dadurch wird ein für die Referenzimplementierung notwendiger umfassender theoretischer Ansatz mit einer unverfälschten Umsetzung sichergestellt. Denn Ausnahme- und Spezialfälle können nicht einfach als solche bearbeitet werden, sondern müssen ebenfalls durch den allgemeinen Ansatz abgedeckt werden.
- Mit den in relationalen Datenbanken enthaltenen Mechanismen zur Beschreibung von Abhängigkeiten und Beziehungen zwischen den Daten kann ohne zusätzlichen Programmieraufwand die Konsistenz der Daten sichergestellt werden.
- Mit Hilfe von SQL-Abfragen können aus der Dialogbeschreibung schnell und effizient spezifische Informationen extrahiert werden. Dies ist insbesondere bei den Maßnahmen zur Sicherung der Dialogqualität und bei der Dialogausführung hilfreich.

Diese informationstechnische Umsetzung der formalen Repräsentation stellt einen essentiellen Bestandteil von EmMI dar, weil alle Komponenten und Funktionen darauf basieren. Sie ist allerdings als solche nicht direkt greifbar, da sie nur entsprechend visuell aufbereitet bei der Dialogerstellung oder als erlebbarer Bediendialog zu Tage tritt.

[1] SQL: Structured Query Language. Allgemein gebräuchliche und weitgehend standardisierte Programmiersprache, mit der auf einfache und effiziente Weise komplexe Abfragen in Datenbanken realisiert werden können.

4.2 Erstellung

Mit dem *Dialogcomponent Manager* und dem *Dialog Modeler* stehen dem Entwickler zwei graphische Werkzeuge zur Verfügung, die eine komfortable Spezifikation des Bediendialogs und eine einfache Beschreibung der Dialogkomponenten erlauben (Bild 41). Sie sind in Visual Basic realisiert. Diese Programmiersprache wird aus folgenden Gründen ausgewählt:

- Die Erstellung komplexer GUIs ist damit einfach und schnell möglich.
- Visual Basic bietet eine gute Kopplung zu MS-Access 2000, wodurch ein einfacher Zugriff auf den Inhalt und den Aufbau der Datenbank möglich wird.
- Als objektorientierte Programmiersprache verfügt Visual Basic über ausreichende Mächtigkeit, um komplexe Funktionen zu realisieren, wie sie beispielsweise zur Umsetzung der Maßnahmen zur Sicherung der Dialogqualität notwendig sind.

4.2.1 Dialogcomponent Manager

Der Dialogcomponent Manager (Bild 42) ist das graphische Front-End zu der in 3.1 beschriebenen kontextfreien Modellierung der Modalitäten. Der Entwickler kann damit die im Projekt enthaltenen Dialogkomponenten verwalten und beschreiben. Es müssen dabei lediglich die Schnittstellen festgelegt werden, da die funktionalen Realisierungen der einzelnen Dialogkomponenten als separate Module ausgeführt sind (4.3.3).

Der Dialogcomponent Manager besteht aus folgenden drei Hauptbereichen:

1. Graphische Übersicht über die einzelnen Dialogkomponenten
   Auf der linken Seite sind alle im Projekt verfügbaren Dialogkomponenten in einer Baumstruktur dargestellt. Für jede Dialogkomponente werden die vier Unterebenen *Konfiguration*, *Status*, *Nachrichten* und *Kommandos* automatisch angelegt. Sie repräsentieren jeweils die zugehörigen Konfigurations- und Statusparameter sowie die Nachrichten und Kommandos. Um den Funktionsumfang einer Dialogkomponente in EmMI bekannt zu machen, müssen die spezifischen Schnittstellenelemente diesen vier Ebenen zugeordnet werden.

2. Eingabebereich
   Entsprechend der Auswahl in der graphischen Darstellung kann im rechten Bereich des Dialogcomponent Managers das jeweilige Element genau spezifiziert werden. Je nach gewähltem Element variiert dieser Teil der Oberfläche, wodurch die Definition der jeweils notwendigen Informationen ermöglicht wird.

   Beispiel:
   In Bild 42 ist die Eingabemaske für die Definition eines Konfigurationsparameters dargestellt.

3. Kontrollbereich
   Im rechten unteren Bereich befinden sich die wichtigsten Kontrollfunktionen, um neue Elemente anzulegen oder bestehende zu löschen. Daneben ermöglicht eine Importfunktion die Einbindung bestehender Schnittstellendefinitionen von Dialogkomponenten aus anderen Projekten. Dadurch wird sowohl der Arbeitsaufwand als auch die Fehleranfälligkeit in diesem Arbeitsschritt deutlich reduziert.

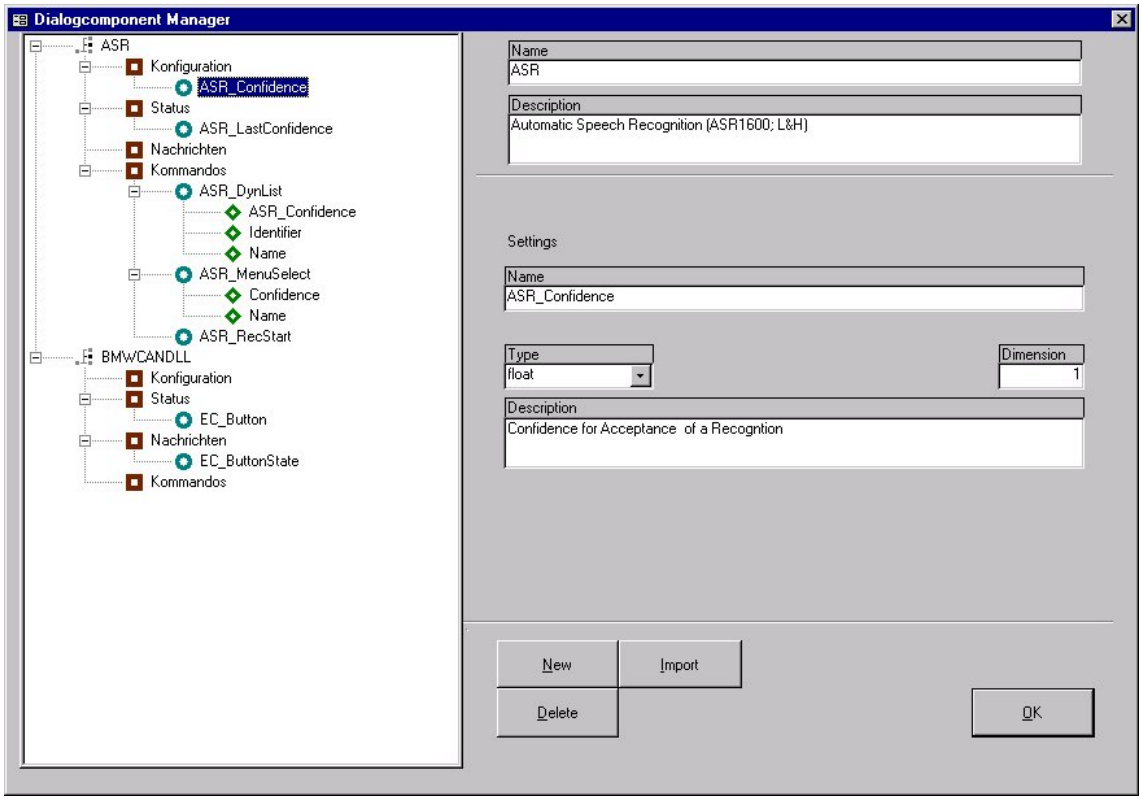


Bild 42 Dialogcomponent Manager

4.2.2 Dialog Modeler

Der Dialog Modeler (Bild 43) ist die zentrale Bedienoberfläche zur Definition des Dialogablaufs und zur gemeinsamen Konfiguration der einzelnen Dialogkomponenten. Gleichzeitig sind darin die in 3.3 beschriebenen Maßnahmen zur Sicherung der Dialogqualität realisiert.

Der Aufbau des Dialog Modelers gliedert sich in drei Bereiche:

1. Basisfunktionen
   Auf der linken Seite befinden sich die Basisfunktionen. Damit können Dialogzustände angelegt oder gelöscht sowie der Detaillierungsgrad der Dialogdarstellung beeinflusst werden.

2. State Tree
   In der Mitte wird im *State Tree* der Bediendialog als Baumstruktur visualisiert. Gleichzeitig dient diese Darstellung zur graphischen Spezifikation des Dialogablaufs.

3. State Specifier
   Mit dem rechts angeordneten *State Specifier* kann das genaue Dialogverhalten angezeigt, definiert und modifiziert werden. Die Auswahl des zu bearbeitenden Zustandes erfolgt dabei über den State Tree. Hierzu werden die gewählten Dialogzustände zusätzlich in den Feldern „Selected State" und „Destination" dargestellt.

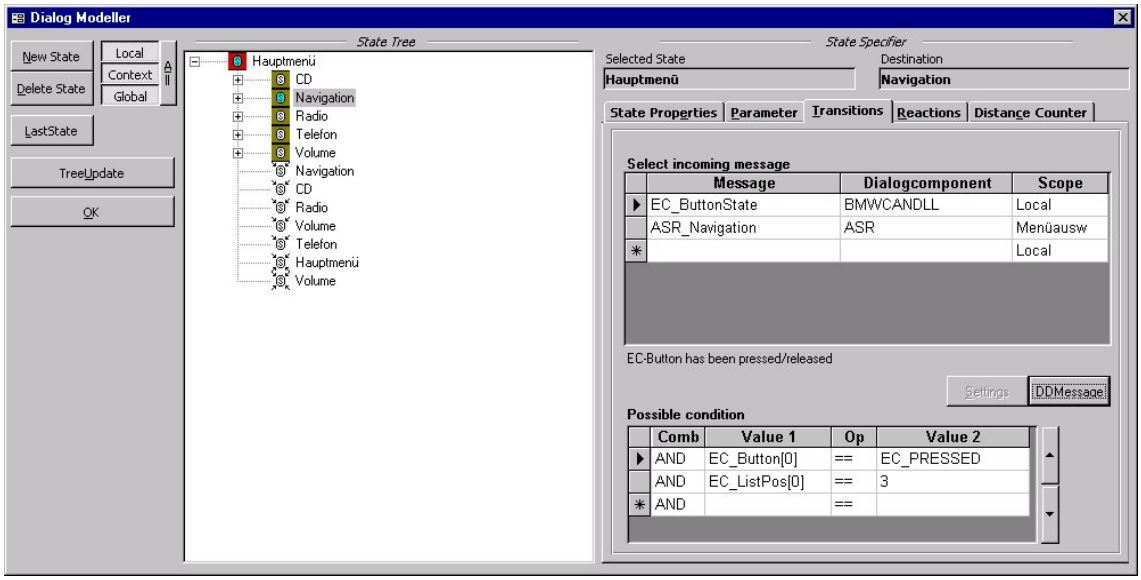


Bild 43 Dialog Modeler

4.2.2.1 State Tree

Im State Tree wird der logische Zusammenhang der einzelnen Dialogzustände visualisiert. Gleichzeitig kann der Entwickler darin den Dialogablauf graphisch spezifizieren und Zustände zur detaillierten Konfiguration im State Specifier auswählen. Der Dialog wird als Baumstruktur dargestellt, wobei jeweils ein Knoten des Baums einen Dialogzustand repräsentiert. Dabei sind alle von einem Zustand aus erreichbaren Zustände als untergeordnete Knoten eingezeichnet. Ähnlich zu Dateinavigationsprogrammen, wie beispielsweise MS-Explorer, kann sich der Entwickler somit durch eine geeignete Darstellung des Baums sowohl einen generellen Überblick über den Gesamtdialog, als auch eine detaillierte Sicht auf den jeweiligen Dialogausschnitt verschaffen (Bild 44). Die in Kapitel 3 benutzte planare Darstellung in einer Netzstruktur scheint zwar auf den ersten Blick eine intuitivere Repräsentation des Dialogs zu ermöglichen, weil sowohl die hierarchische Strukturierung als auch die logischen Zusammenhänge der Zustände mit einem Blick erfasst werden können. Da aber bei Modalitäten, wie beispielsweise Spracherkennung, Übergänge oft unabhängig von logischen Menüebenen realisiert werden müssen, entstehen viele sich überschneidende Kontexte und Transitionen, wodurch die Übersichtlichkeit der

Darstellung sehr schnell nicht mehr gegeben ist. Insbesondere für umfangreichere Dialoge ist die Darstellung als Baumstruktur somit besser geeignet.

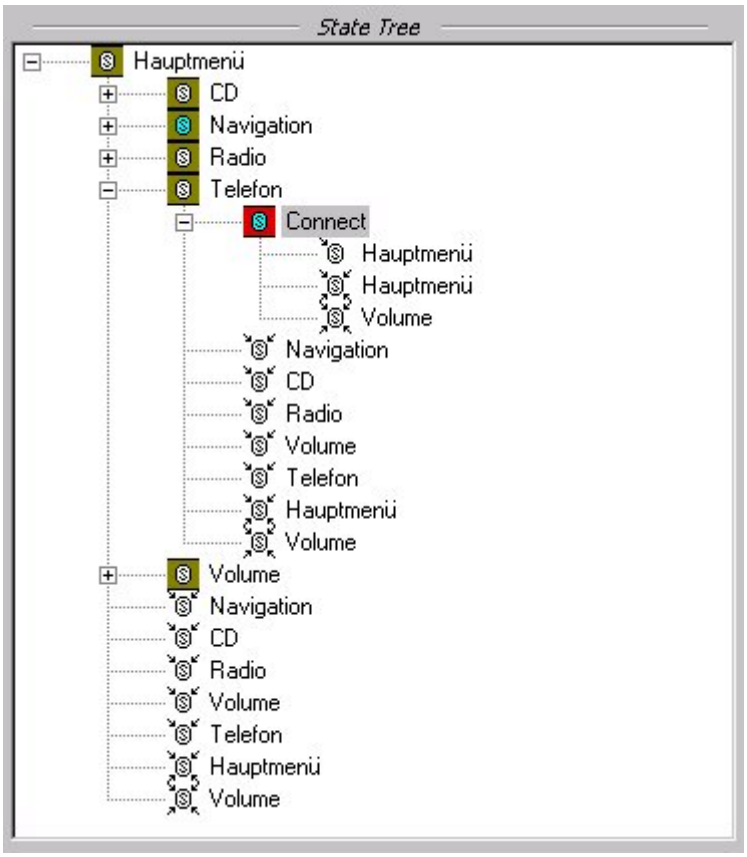


Bild 44 Graphische Darstellung des Dialogs als Baumstruktur im State Tree

Jeder Dialogzustand ist genau an einer Stelle der Dialogbeschreibung definiert. Alle anderen Verweise auf diesen Dialogzustand sind lediglich Verknüpfungen zu dieser Definition (Bild 44). Sie können anhand der Icons voneinander unterschieden werden (Tabelle 6). Dabei ist es für die Dialogbeschreibung irrelevant, an welcher Stelle der jeweilige Zustand definiert wird. Für eine einfache Orientierung und Navigation in der Dialogbeschreibung empfiehlt sich allerdings eine Anordnung entsprechend der Hauptinteraktionssequenzen. Darunter sind die Standardbedienvorgänge zu verstehen, mit denen beispielsweise von einem Hauptmenü aus die jeweiligen Funktionen erreicht und dann ausgelöst werden.

Der Entwickler wird auf Verstöße gegen die Vollständigkeit der Dialogbeschreibung (3.3.1.1) aufmerksam gemacht. Das heißt Zustände, die nicht erreichbar sind oder die zu einer Blockierung des Dialogs führen würden, weil keine Transition mehr von ihnen wegführt, werden durch eine entsprechende Markierung gekennzeichnet (Tabelle 6). Damit kann bereits während der Dialogspezifikation ein für den Dialog unverzichtbares Qualitätsmerkmal sichergestellt werden, das empirisch nur mit sehr hohem Aufwand verifizierbar wäre.

In Tabelle 6 ist die Bedeutung der im State Tree anzutreffenden Icons dargestellt. Durch einen Doppelklick mit der Maus wird ein Zustand zum *Selected State*, was an der rot-blauen Hinterle-

gung des Icons erkennbar ist (Bild 43). Der Selected State legt den im State Specifier bearbeitbaren Zustand fest. Gleichzeitig dient er als Elternzustand bei der Neudefinition von Dialogzuständen. Es ist immer nur genau ein Zustand Selected State. Den Zielzustand einer Transition oder zur Entfernungsberechnung mittels des Distance Counters bezeichnet man als *Destination*. Er wird durch einen einfachen Klick festgelegt und ist anhand der blauen Kennzeichnung zu identifizieren. Es kann auch hierfür stets nur ein Zustand ausgewählt sein.

Tabelle 6 Bedeutung der Icons im State Tree

Normal	Selected State	Destination	Beschreibung
			Definition eines Dialogzustandes
			Verknüpfung mit einem Dialogzustand aufgrund einer Transition mit lokalem Gültigkeitsbereich
			Verknüpfung mit einem Dialogzustand aufgrund einer Transition mit kontextweitem Gültigkeitsbereich
			Verknüpfung mit einem Dialogzustand aufgrund einer Transition mit globalem Gültigkeitsbereich
			Der Dialogzustand kann nicht erreicht werden
			Der Dialogzustand kann nicht verlassen werden: Blockierung des Dialogs

4.2.2.2 State Specifier

Das Verhalten eines Dialogzustandes wird mit Hilfe des State Specifiers detailliert festgelegt. Dieser besteht im Wesentlichen aus einem Registersteuerelement mit fünf Registern. Die darin enthaltenen Eingabemasken stellen das Herzstück der zentralen Konfiguration der Dialogkomponenten und der Spezifikation des Dialogverhaltens dar. Daneben werden in den Feldern „Selected State“ und „Destination“ die entsprechend im State Tree gewählten Zustände angezeigt. Um einen Dialogzustand im State Specifier bearbeiten zu können, muss dieser als Selected State gewählt sein.

State Properties

Im Register „State Properties“ werden spezifische Eigenschaften des Zustandes definiert (Bild 45). Hierzu gehört im Feld „State description“ eine fakultative Beschreibung des Zustandes. Im

Bereich „Assign contexts“ kann der Zustand einer beliebigen Anzahl an Kontexten zugeordnet werden (3.2.1). Die Verwaltung der verfügbaren Kontexte ist über den Schalter „Contexts“ erreichbar. Bei „Enter keywords“ können für den Dialogzustand charakteristische Begriffe als Schlüsselwörter angegeben werden. Diese stehen dann zur Konfiguration des Verhaltens der Dialogkomponenten zur Verfügung.

Beispiel:

Der Zustand *Hauptmenü* in Bild 45 ist dem Kontext *Menüauswahl* zugeordnet und durch die Schlüsselwörter *Hauptmenü* und *Anfang* charakterisiert.

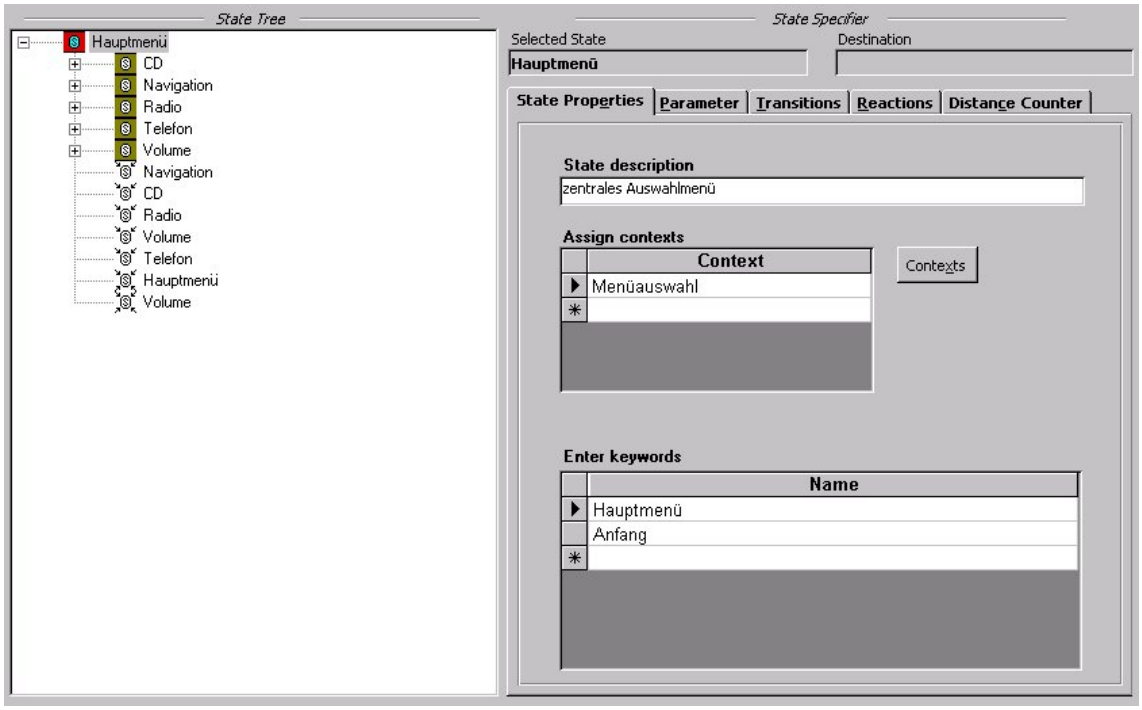


Bild 45 Zuordnung der Kontexte und Definition der Schlüsselwörter im State Specifier

Parameter

Die Definition der Parameterwerte (3.2.6) für Konfigurationsparameter (3.1.2) erfolgt im Register „Parameter“ (Bild 46). Dazu muss zunächst bei „Select Parameter“ der Parameter ausgewählt und im Feld „Scope“ der gewünschte Gültigkeitsbereich festgelegt werden. Die Auswahl erfolgt über ein Kombinationsfeld, in dem alle Konfigurationsparameter der vorhandenen Dialogkomponenten zur Auswahl stehen. Im Bereich „Enter Value“ wird dann der spezielle Parameterwert festgelegt. Bei der Eingabe des Wertes wird sowohl der Typ, als auch die Einhaltung einer eventuell definierten maximalen Zeichenlänge sichergestellt. Zur einfachen Festlegung des Wertes werden die Schlüsselwörter der als Selected State und Destination gewählten Zustände in der Auswahlliste des Kombinationsfeldes angeboten. Dadurch wird der Entwickler neben der zentralen Konfigurationsmöglichkeit der Dialogkomponenten zusätzlich bei deren konsistenten Parametrisierung unterstützt.

Falls das Bedienkonzept dies erforderlich macht, kann durch die Angabe einer Bedingung bei „Possible condition“ die Gültigkeit der Wertzuweisung von den während der Dialogausführung aktuellen Werten bestimmter Statusparameter abhängig gemacht werden. Dies entspricht der formalen Darstellung in 3.2.11.

Sind in einem Dialogzustand für einen Parameter unterschiedliche Parameterwerte mit verschiedenen Gültigkeitsbereichen definiert, dann findet der in 3.2.9.1 erläuterte Verdeckungsmechanismus Anwendung. Dadurch ist für den Entwickler in jedem Zustand nur der jeweils gültige Wert sichtbar.

Beispiel:

Die Wertzuweisung für den Parameter `GUI_TX_MainMenu` besitzt globale Gültigkeit und basiert auf den Schlüsselwörtern, die für den Zustand *`Hauptmenü`* angegebenen sind (Bild 46). Um eine der weiteren Wertdefinitionen in diesem Zustand zu bearbeiten, muss der entsprechende Parameter bei „Select Parameter“ ausgewählt werden.

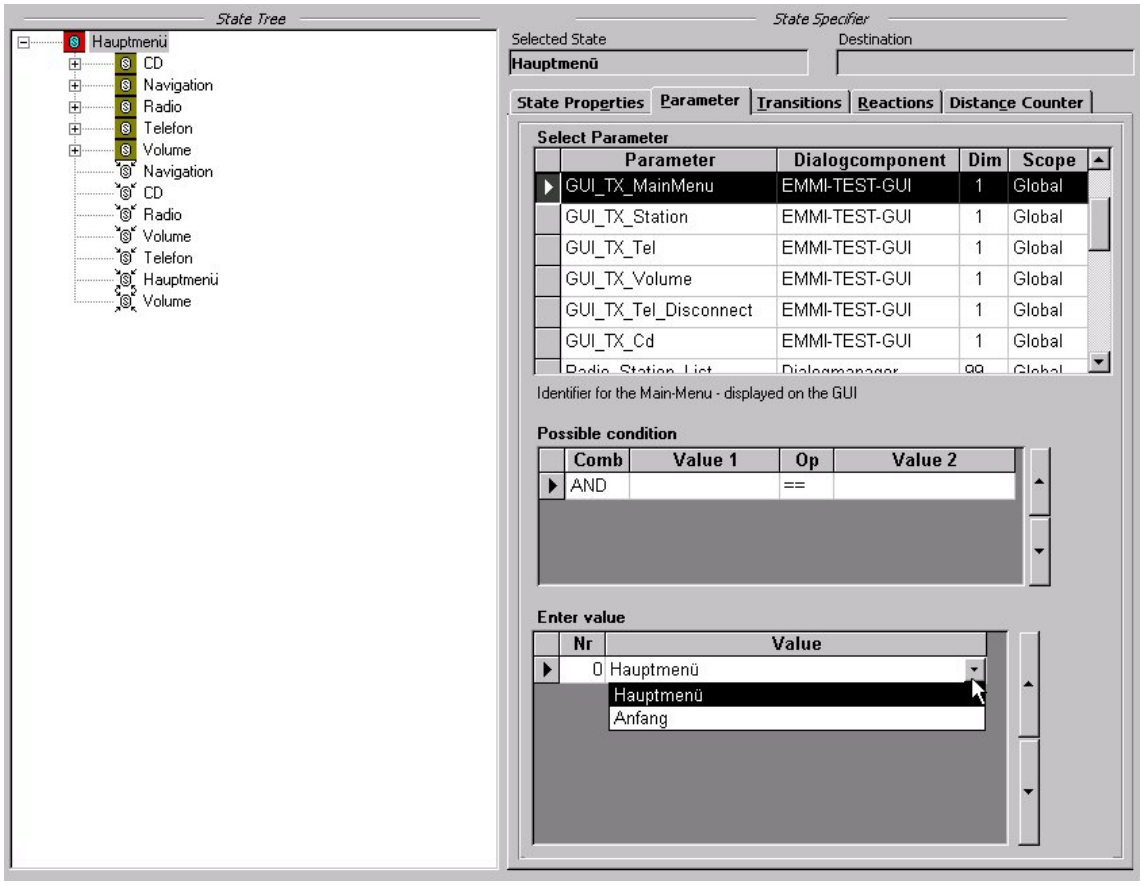


Bild 46 Definition eines Konfigurationsparameterwerts über ein Schlüsselwort im State Specifier

Transitions

Zustandsübergänge von Selected State nach Destination werden im Register „Transitions“ definiert (Bild 47). Hierzu wird bei „Select incoming message“ die zur Auslösung der Transition

(3.2.2) notwendige Nachricht einer Dialogkomponente ausgewählt. Im Feld „Scope“ kann der Gültigkeitsbereich der Transition angepasst werden. Der Schalter „DDMessage“ öffnet die Verwaltung der entwicklerdefinierten Nachrichten. Diese können dann, genau wie aktivierbare Nachrichten, über den Schalter „Settings“ konfiguriert werden. Dies ist ein wichtiger Bestandteil der zentralen Konfiguration der Dialogkomponenten.

Falls notwendig kann durch die Angabe einer Bedingung bei „Possible condition“ die Ausführung der Transition von den aktuellen Werten bestimmter Statusparameter abhängig gemacht werden.

Wenn für die angegebene Nachricht bereits eine weitere Transition mit identischem Gültigkeitsbereich definiert ist, zwingt der State Specifier den Entwickler zur Eingabe von Bedingungen für beide Transitionen. Dadurch wird die Eindeutigkeit der Dialogbeschreibung (3.3.1.2) sichergestellt. Sind für diese Transitionen unterschiedliche Gültigkeitsbereiche angegeben, dann ist keine Bedingung erforderlich, da in diesem Fall der Verdeckungsmechanismus (3.2.9.1) greift.

Beispiel:

Für den Zustandsübergang von *Hauptmenü* nach *Navigation* in Bild 47 sind zwei Transitionen definiert. Die eine ist lokal im Zustand *Hauptmenü* gültig und wird durch die Nachricht `EC_ButtonState` ausgelöst, wobei zusätzlich eine Bedingung erfüllt sein muss. Die andere Transition wird durch die Nachricht `ASR_Navigation` ausgelöst. Sie ist in allen Zuständen des Kontextes *Menüauswahl* gültig.

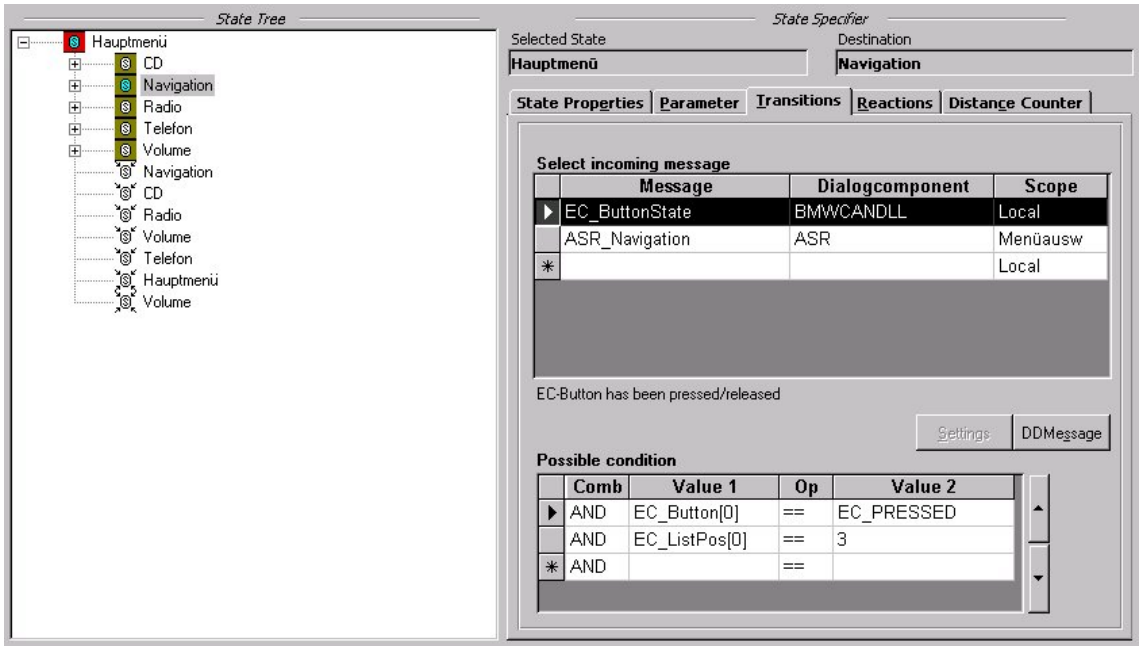


Bild 47 Lokale Transition unter Angabe einer Bedingung im State Specifier

Reactions

Das funktionale Verhalten des als Selected State gewählten Zustandes wird im Register „Reactions” festgelegt (Bild 48). Die wichtigsten Bestandteile der in 3.2.3 beschriebenen Reaktionen

sind die auslösende Nachricht und die damit verbundenen Kommandos. Erstere wird analog zur Vorgehensweise bei den Transitionen im Bereich „Select incoming message“ ausgewählt und konfiguriert. Letztere werden bei „Outgoing commands“ zugeordnet. Die Festlegung der speziellen Kommandoparameterwerte (3.2.6) für den jeweiligen Kommandoaufruf (3.2.10) geschieht in dem zur Auswahl des Kommandos geöffnetem Dialogfeld. Dabei ist für diese Parameterwerte nicht nur die Angabe einer frei definierbaren Konstante möglich, sondern sie können auch mit Konfigurations- bzw. Statusparametern oder den Schlüsselwörtern der als Selected State bzw. Destination gewählten Zustände verknüpft werden. Diese Parametrisierung der Kommandos ist ein elementares Element der zentralen Konfiguration der Dialogkomponenten und liefert somit einen wichtigen Beitrag zur Sicherung der Dialogqualität.

Auch die Ausführung von Reaktionen kann durch die Angabe einer Bedingung bei „Possible condition“ von den aktuellen Werten bestimmter Statusparameter abhängig gemacht werden.

Im Gegensatz zu den Transitionen sind bei den Reaktionen die Maßnahmen zur Sicherung der Eindeutigkeit der Dialogbeschreibung nicht erforderlich. Sie bedürfen daher keiner funktionalen Realisierung in EmMI. Vielmehr kann eine Nachricht zur Auslösung unterschiedlicher Reaktionen in einem Zustand benutzt werden. Die Bearbeitungsreihenfolge ist dabei durch den in 3.2.9.2 beschriebenen Mechanismus festgelegt.

Beispiel:

In Bild 48 ist eine Reaktion mit globalem Gültigkeitsbereich dargestellt, die von der Nachricht `DM_EnterState` ausgelöst wird. Es wird dabei das Kommando `EC_SetEffect` erzeugt. Im Zustand *`Hauptmenü`* sind eine Reihe weiterer Reaktionen mit unterschiedlichen Gültigkeitsbereichen definiert. Diese müssen zuerst bei „Select incoming message“ ausgewählt werden, um dann in den weiteren Feldern bearbeitet werden zu können.

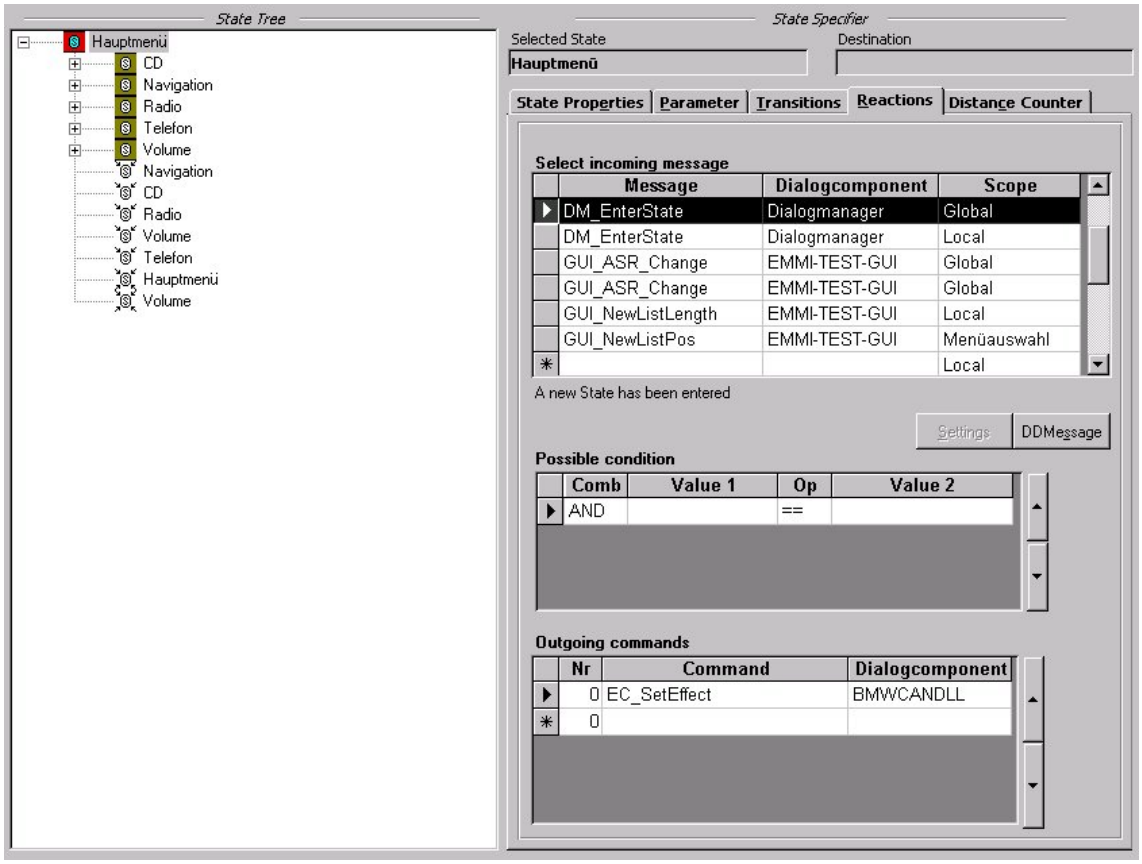


Bild 48 Reaktion mit globalem Gültigkeitsbereich im State Specifier

Distance Counter

Im Reiter „Distance Counter“ ist das in 3.3.1.3 beschriebene Verfahren zur Berechnung der Entfernung zweier Zustände implementiert (Bild 49). Dabei wird die minimal erforderliche Zahl an Transitionen bestimmt, mit denen ein Übergang von dem als Selected State zu dem als Destination gekennzeichneten Zustand möglich ist. Das Ergebnis wird im Feld „Distance (Steps)“ angezeigt. In die Berechnung fließen dabei nur diejenigen Transitionen ein, die von einer bei „Select Dialogcomponents“ markierten Dialogkomponente ausgelöst werden. Dadurch kann der Entwickler sehr effizient verifizieren, ob eine Funktion unter Verwendung einer bestimmten Modalitätskombination bedient werden kann. Die Entfernung stellt dabei eine früh im Entwicklungsprozess verfügbare Maßzahl zur Abschätzung des Bedienaufwands dar. Mit dem Distance Counter werden somit Qualitätsmerkmale des Bediendialogs hinsichtlich Interaktionslänge und Erreichbarkeit adressiert.

Beispiel:

Um vom Zustand *Connect* zum Zustand *Navigation* in Bild 49 zu gelangen sind unter Verwendung der Dialogkomponenten ASR und BMWCANDLL mindestens zwei Transitionen erforderlich.

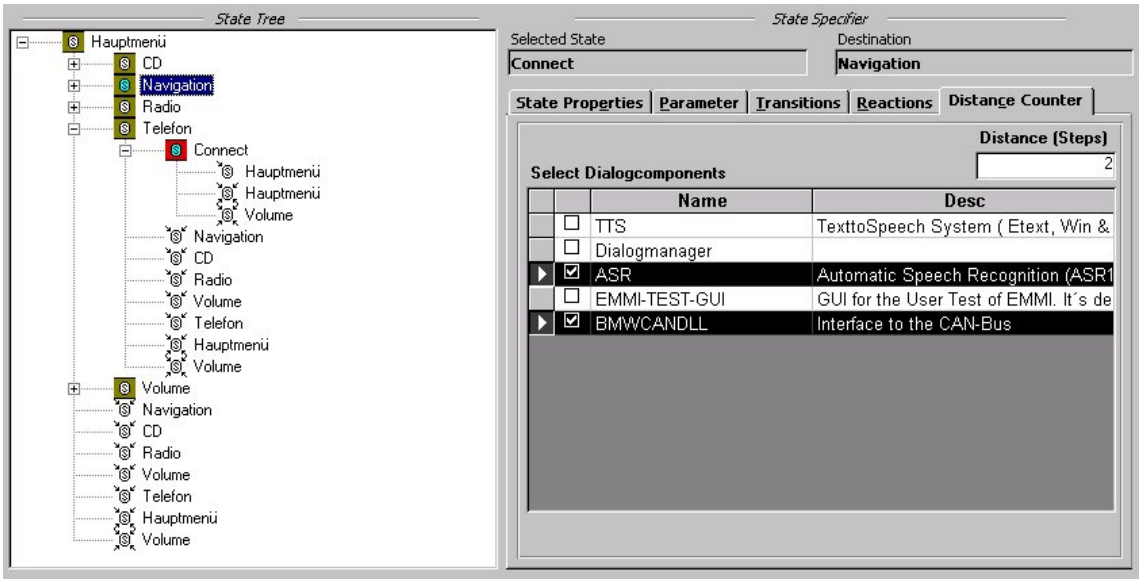


Bild 49 Bestimmung der Entfernung zweier Zustände im State Specifier

4.3 Ausführung

Bei der Ausführung des Dialogs kontrolliert der *Dialogmanager* den Dialogablauf. Er steuert und synchronisiert hierzu die verschiedenen *Dialogkomponenten* (Bild 41). Aufgrund der starken Modularisierung muss diese Steuerung über eine gemeinsame Schnittstelle mit einem einheitlichen *Kommunikationsprotokoll* erfolgen.

Durch die Separierung der verschiedenen Komponenten in selbständige Softwaremodule wird ein sehr hoher Flexibilitätsgrad erreicht. Dies betrifft vor allem die Erweiterung, den Austausch und die Wiederverwendung einzelner Dialogkomponenten. Gleichzeitig können die teilweise sehr rechenintensiven Module auf mehrere Rechner skaliert werden. Diese Verteilung der Rechenlast führt zu einer besseren Performance des Gesamtsystems. Daneben erleichtert dies eine betriebssystemunabhängige Entwicklung und Einbindung der Dialogkomponenten in EMMI.

Der Dialogmanager und die Kommunikationsschnittstelle sind in der Programmiersprache C++ [Stroustrup 1998] [Kruglinski, *et al.* 1998] implementiert. Da das Kommunikationsinterface als eigenständige Bibliothek gekapselt ist, kann es bei der Entwicklung einer neuen Dialogkomponente einfach eingebunden werden. Soweit sinnvoll und möglich basieren dabei die Dialogkomponenten auf bereits bestehenden und bewährten Werkzeugen.

4.3.1 Dialogmanager

Im Dialogmanager ist die in 3.5 dargestellte Heuristik zur Ausführung des Bediendialogs realisiert. Damit können alle Bediendialoge erlebbar gemacht werden, die auf der in 3.2 eingeführten formalen Beschreibung basieren. Aufgrund der kontextfreien Modellierung der Modalitäten

(3.1) ist es dabei möglich, beliebige Dialogkomponenten zu kontrollieren, ohne dass Modifikationen am Dialogmanager selbst notwendig sind.

Bild 50 zeigt den Dialogmanager. Im Feld „Database“ wird der auszuführende Bediendialog festgelegt. Dabei handelt es sich um den Pfad zu der mit dem Dialog Modeler erstellten Dialogspezifikation. Durch Aktivieren der Umschaltfläche „(Dis-)Connect“ wird die Dialogbeschreibung geöffnet und die Dialogausführung gestartet. Die restlichen Anzeigen dienen zur Überwachung und Kontrolle des Dialogablaufs. Im Bereich „Dialog Components“ sind die in der Dialogspezifikation enthaltenen Dialogkomponenten und ihr momentaner Aktivitätszustand erkennbar. Dort besteht auch die Möglichkeit Nachrichtenpakete manuell an die Dialogkomponenten zu verschicken. Der Dialogablauf kann im Bereich „Status“ verfolgt werden, wobei mit den Kontrollkästchen eine Anpassung der angezeigten Informationsdichte möglich ist.

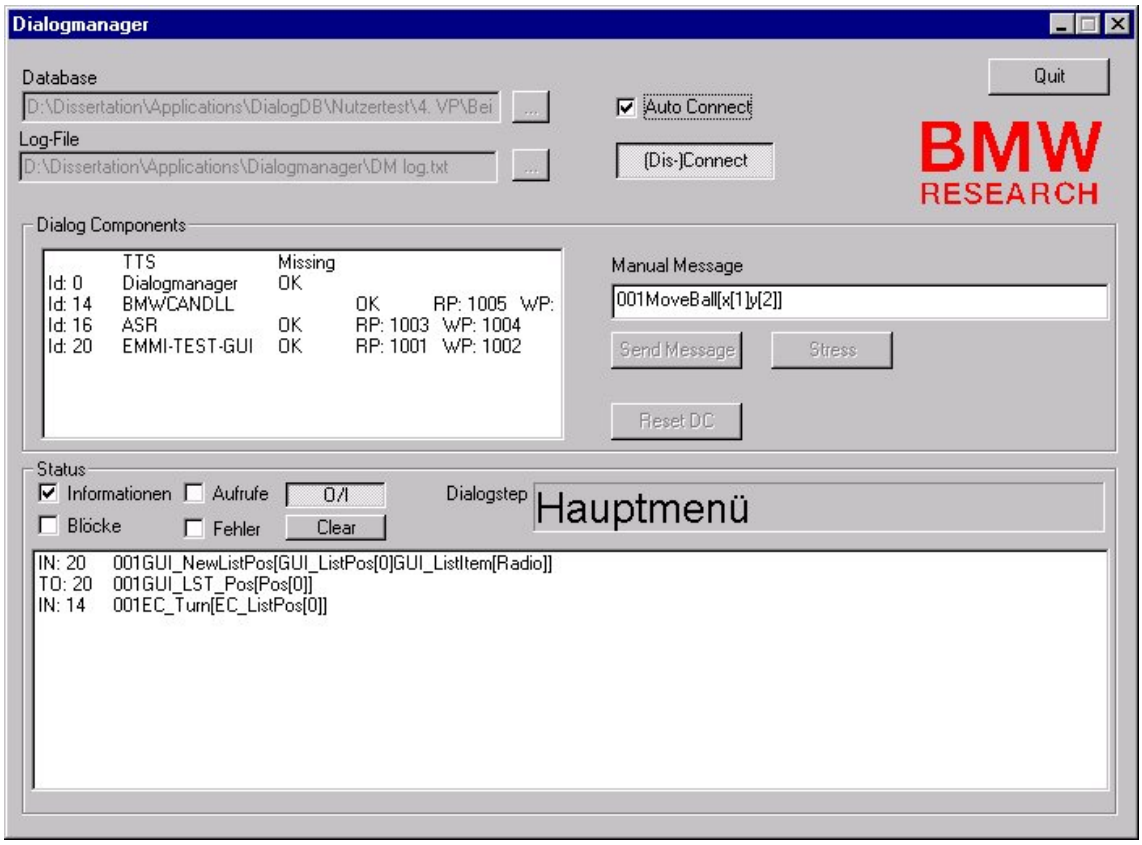


Bild 50 Dialogmanager

4.3.2 Kommunikationsschnittstelle und Kommunikationsprotokoll

Um die Entwicklung neuer Dialogkomponenten so einfach wie möglich zu gestalten, ist die Kommunikationsschnittstelle als eigenständige C++-Bibliothek gekapselt. Damit beschränkt sich der Aufwand zur Einbindung einer Dialogkomponente in EmMI auf wenige Methodenaufrufe. Die Informationsübertragung basiert dabei auf einer Kommunikation über Sockets [Kruglinski, *et*

al. 1998]. Dies erlaubt zwar eine einfache Verteilung der Dialogkomponenten auf mehrere Rechner, allerdings zeigt sich, dass dies auch zu spürbaren Reaktionszeiten des Systems führen kann. Für Bediendialoge, die auf einem Rechner ausführbar sind, würde daher ein alternativer, schnellerer Übertragungskanal – beispielsweise über einen gemeinsamen Speicherbereich – eine sinnvolle Erweiterung dieser Bibliothek darstellen.

Die eigentlichen Informationen werden in Nachrichtenpaketen übertragen, deren Aufbau in Bild 51 beschrieben ist. Die Schnittstellenbibliothek enthält hierzu ebenfalls spezielle Methodenaufrufe, sodass sowohl zur Generierung als auch zur Analyse der Nachrichtenpakete kein zusätzlicher Aufwand erforderlich ist. Ein Nachrichtenpaket besteht aus drei Grundelementen:

1. Priorität
   Mit Hilfe dieser drei Ziffern könnte eine Priorisierung der Bearbeitungsreihenfolge zeitlich nah zusammenliegender Nachrichtenpakete realisiert werden. Sie finden allerdings in dieser Implementierung keine Berücksichtigung.

2. Bezeichner
   Anhand des Bezeichners kann der Inhalt des Nachrichtenpakets identifiziert werden. Hierbei handelt es sich normalerweise um Namen von Nachrichten oder Kommandos der jeweiligen Dialogkomponente bzw. Kontrollkommandos zur Dialogsteuerung. Daneben existieren noch eine Reihe weiterer interner Befehle zum Auf- und Abbau der Kommunikation zwischen Dialogmanager und Dialogkomponente.

3. Daten
   Der Datenblock enthält eine beliebige Anzahl an Einzelinformationen. Diese können jeweils anhand des Parameternamens eindeutig identifiziert werden. Jedem Parameternamen ist dabei ein Wert zugeordnet, der aus einem beliebig großen Informationsarray bestehen kann.

$$
\begin{aligned}
\langle \textit{Nachrichtenpaket} \rangle &::= \langle \textit{Priorität} \rangle\ \langle \textit{Bezeichner} \rangle\ \text{'['}\ \langle \textit{Daten} \rangle\ \text{']'} \\
\\
\langle \textit{Priorität} \rangle &::= \langle \textit{Ziffer} \rangle\ \langle \textit{Ziffer} \rangle\ \langle \textit{Ziffer} \rangle \\
\langle \textit{Bezeichner} \rangle &::= \langle \textit{Zeichen} \rangle\ \{\ \langle \textit{Zeichen} \rangle\ \} \\
\langle \textit{Daten} \rangle &::= \{\ \langle \textit{Parametername} \rangle\ \text{'['}\ \langle \textit{Wert} \rangle\ \text{']'}\ \} \\
\\
\langle \textit{Parametername} \rangle &::= \langle \textit{Zeichen} \rangle\ \{\ \langle \textit{Zeichen} \rangle\ \} \\
\langle \textit{Wert} \rangle &::= \{\ \langle \textit{Zeichen} \rangle\ \{\ \langle \textit{Zeichen} \rangle\ \}\ \text{'|'}\ \} \\
\\
\langle \textit{Zeichen} \rangle &::= \langle \textit{Ziffer} \rangle\ |\ \langle \textit{Buchstabe} \rangle \\
\langle \textit{Ziffer} \rangle &::= \text{'0'}\ |\ \text{'1'}\ |\ \text{'2'}\ |\ \ldots\ |\ \text{'9'} \\
\langle \textit{Buchstabe} \rangle &::= \text{'a'}\ |\ \text{'b'}\ |\ \text{'c'}\ |\ \ldots\ |\ \text{'z'}\ |\ \text{'A'}\ |\ \text{'B'}\ |\ \text{'C'}\ |\ \ldots\ |\ \text{'Z'}
\end{aligned}
$$

Bild 51 Aufbau der Nachrichtenpakete zur Kommunikation zwischen Dialogmanager und Dialogkomponenten

Beispiel:
Die Transition aus Bild 47 könnte durch das in Bild 52 dargestellte Nachrichtenpaket herbeigeführt werden. Der Name der auslösenden Nachricht `EC_ButtonState` legt den Bezeichner fest und die zur Erfüllung der Bedingung notwendigen Parameterwerte werden im Datenblock übertragen.(`EC_Button`=*`EC_Pressed`*; `EC_ListPos`=*3*)

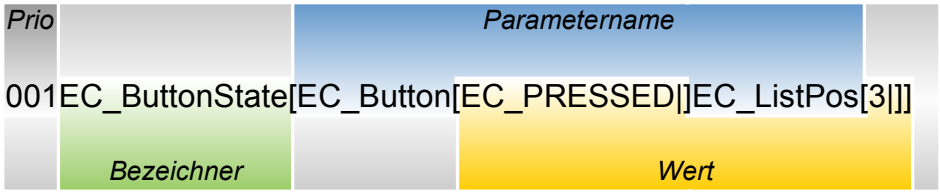


Bild 52 Nachrichtenpaket zur Übermittlung einer Nachricht mit Statusparameterwerten von einer Dialogkomponente zum Dialogmanager

4.3.3 Dialogkomponenten

Dialogkomponenten sind die funktionalen Realisierungen der am Bediendialog beteiligten Modalitäten. Sie bilden die technischen Sensoren und Effektoren, auf denen die Kommunikation mit dem Benutzer während der Dialogausführung basiert (Bild 41). Daher tragen sie wesentlich dazu bei, den Dialog erlebbar zu machen. Für einen möglichst flexiblen Einsatz und eine hohe Wiederverwendbarkeit wird jede Dialogkomponente als selbständiges Softwaremodul realisiert. Das Verhalten wird vom Entwickler mit dem Dialog Modeler festgelegt. Die Steuerung während der Dialogausführung erfolgt entsprechend dieser Spezifikation durch den Dialogmanager. Hierzu muss der Funktionsumfang der jeweiligen Dialogkomponente auf die in 3.1 dargestellten semantischen Mittel abgebildet werden.

Im Folgenden werden die im Rahmen dieser Arbeit entstandenen Dialogkomponenten kurz vorgestellt. Sie stellen beispielhafte Umsetzungen im Fahrzeug einsetzbarer Aktions- und Wahrnehmungsmodalitäten dar. Um den Entwicklungsaufwand möglichst gering zu halten, dienen als Grundlage für die Realisierung ausschließlich bestehende Werkzeuge oder Bibliotheken. Dabei werden möglichst solche Werkzeuge eingesetzt, die bereits im Entwicklungsprozess enthalten sind und daher wenig Einarbeitungsaufwand für die jeweilige Fachstelle erfordern.

4.3.3.1 Spracherkennung

Die benutzerunabhängige Erkennung von Sprache ist in der Dialogkomponente `ASR` realisiert. Diese basiert exemplarisch auf dem Spracherkenner *ASR1600* und dem zugehörigen Entwicklungs-API[1] (Version 3.22) von Lernout & Hauspie [Lernout & Hauspie 1998]. Die Implemen-

[1] API: Application Programming Interface

tierung der Dialogkomponente erfolgt in C++. Es werden dabei Funktionen bereit gestellt, die typischerweise zur Umsetzung von TICS erforderlich sind:

- Menüauswahl
  Für eine einfache und flexible Realisierung von Auswahl- und Menünavigationsfunktionen können beliebige Sprachkommandos in Form von vorkompilierbaren, entwicklerdefinierten Nachrichten festgelegt werden. Daraus wird beim Start der Dialogausführung das für den Spracherkenner notwendige Vokabular generiert.

- Navigationszieleingabe
  Zur Realisierung der Zieleingabe für Navigationssysteme sind die 300 größten Städte Deutschlands mit allen zugehörigen Straßennamen über aktivierbare Nachrichten per Sprache auswählbar.

- Zifferneingabe für Telefonnummern
  Für Telefonnummern steht eine aktivierbare Nachricht bereit, mit der Ziffernfolgen eingegeben und bei Bedarf korrigiert werden können.

- Auswahl aus dynamischen Listen
  Für dynamisch veränderliche Listen, wie beispielsweise der Senderliste des Radios oder den Telefonbucheinträgen im Mobiltelefon, können entwicklerdefinierte Nachrichten festgelegt werden, deren Vokabular bei jeder Aktivierung neu generiert wird.

Der Aufbau des GUI der Dialogkomponente `ASR` ist in Bild 53 zu erkennen. Im oberen Teil wird bei dem Punkt „Communication“ die Verbindung mit dem Dialogmanager kontrolliert. In den weiteren Feldern stehen verschiedene Kontroll- und Überwachungsfunktionen zur Verfügung. Hierzu gehören die manuelle Versendung von Nachrichten an den Dialogmanager, die Auswahl der Log-Datei sowie die Ausgabe von Statusinformationen. Im Bereich „Automatic Speech Recognition“ werden weitere Informationen zum Zustand des Spracherkenners und zum aktuellen Erkennungsergebnis angezeigt. Daneben können verschiedene von der jeweiligen Dialogdefinition unabhängige Parameter des Spracherkenners beeinflusst werden.

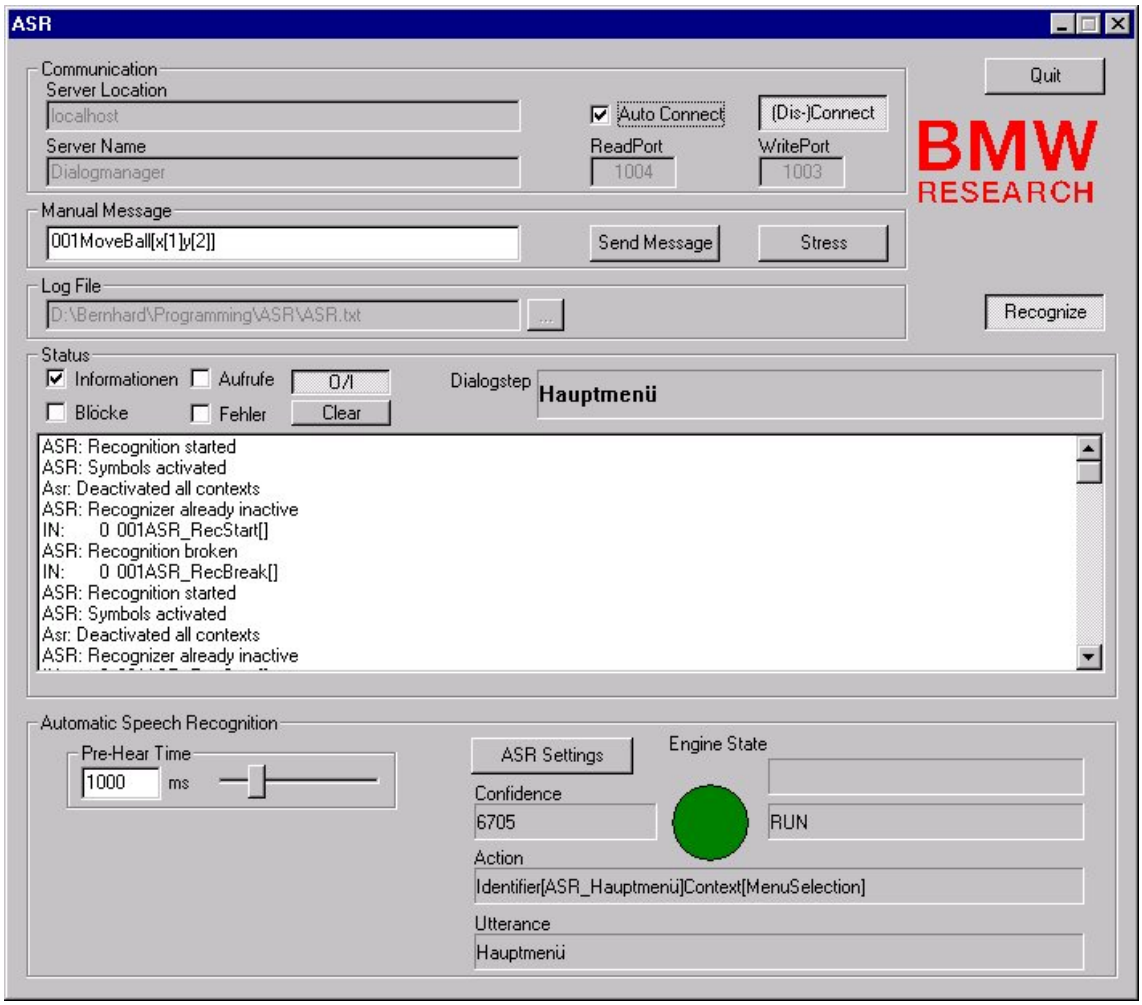


Bild 53 Dialogkomponente `ASR` zur Spracherkennung

4.3.3.2 Graphische Anzeigen

Zur Erstellung graphischer Anzeigen wird *Director Shockwave Studio* (Version 8) genutzt [Macromedia 2000a]. Dies ist ein Autorensystem zur Erstellung interaktiver Multimedia-Produktionen, das aufgrund seiner umfangreichen graphischen Fähigkeiten beim Entwurf rein displayorientierter Bedienkonzepte bereits intensiv eingesetzt wird. Gleichzeitig ist es wegen der integrierten Skriptsprache sehr flexibel verwendbar. Die Einbindung der zur Kommunikation mit dem Dialogmanager notwendigen Schnittstellenbibliothek (4.3.2) erfolgt dabei über die vorhandene Programmierschnittstelle als so genanntes *Xtra* [Macromedia 2000b]. Somit können unter Verwendung der Skriptsprache sowohl eintreffende Kommandos in die zugehörigen Systemreaktionen umgesetzt, als auch Nachrichten für den Dialogmanager generiert werden.

Bild 54 zeigt hierzu beispielhaft eine Anzeige für ein zentrales Display. Aufgrund der sehr starken Kopplung des logischen Bedienablaufs mit der zugehörigen graphischen Anzeige müssen Modifikationen jeweils genau aufeinander abgestimmt werden. Daher ist eine Wiederverwendung bestehender graphischer Anzeigen für neue Bedienkonzepte nicht einfach möglich.

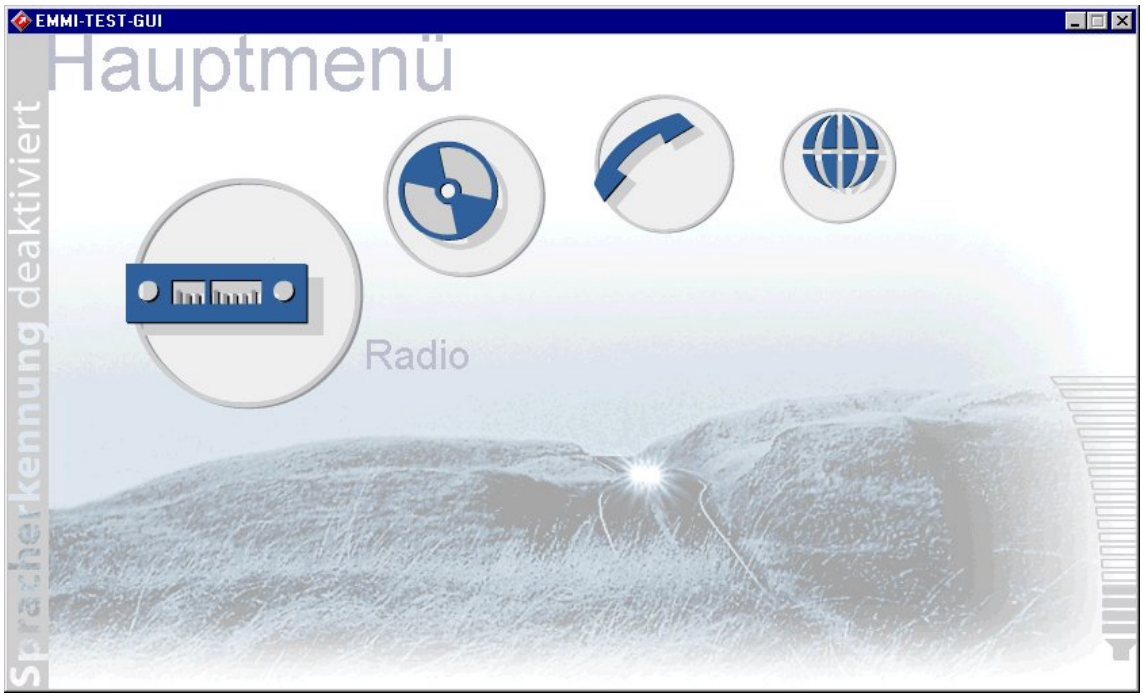


Bild 54 Dialogkomponente zur graphischen Anzeige von Informationen in einem zentralen Display

4.3.3.3 Sprachausgabe

Die Dialogkomponente `TTS` ermöglicht die synthetische Generierung von Sprachausgaben. Sie basiert auf dem Sprachsynthesesystem *Etex* (Version 2.0) mit der *Elan Text To Speech Engine* (Version 2.0). Unter Verwendung des zugehörigen API [Elan 1999] erfolgt die Implementierung der Dialogkomponente in C++. Der Funktionsumfang erlaubt die Festlegung der Äußerung, die Steuerung der Ausgabe und die Parametrisierung des Synthesesystems.

In Bild 55 ist zu erkennen, dass die Dialogkomponente `TTS` einen ähnlichen Aufbau wie `ASR` (4.3.3.1) besitzt und lediglich im unteren Bereich Unterschiede aufweist. Dort werden neben Informationen zum Zustand des Synthesesystems die momentane Parametrisierung und die aktuelle Äußerung angezeigt.

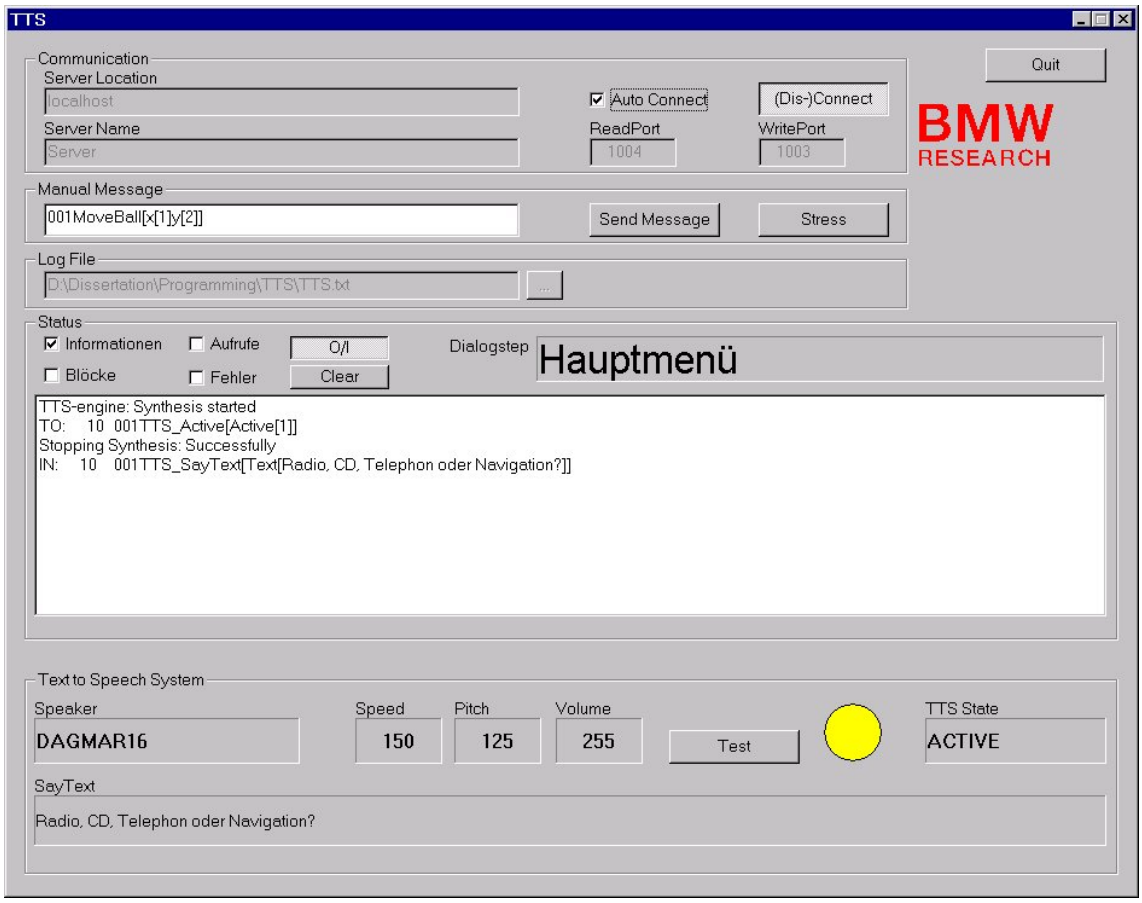


Bild 55 Dialogkomponente `TTS` zur synthetischen Generierung von Sprachausgaben

4.3.3.4 CAN-Interface

Als manuelles Bedienelement wird der aus dem BMW iDrive bekannte Controller verwendet. Da dieser lediglich über eine CAN[1]-Schnittstelle verfügt, ist eine entsprechende Kommunikationsmöglichkeit erforderlich. Hierzu existieren mit CANoe (Version 3.2) [Vector 2002] bzw. CANalyzer (Version 3.2) [Vector 2001] zwei verbreitete Entwicklungs- und Analysewerkzeuge für CAN-Bussysteme. Sie erlauben die Beobachtung, Untersuchung und Ergänzung des Datenverkehrs auf der Busleitung. Ähnlich wie in 4.3.3.2 ist es dabei möglich, das Kommunikationsinterface zum Dialogmanager als DLL[2] über die vorhandene Programmierschnittstelle einzubinden. Auf die damit bereitgestellte Funktionalität kann mit Hilfe der in CANoe und CANalyzer verfügbaren Programmiersprache zugegriffen werden. Diese ermöglicht zur Kommunikation mit

[1] CAN: Controller Area Network. Standardisiertes, serielles Bussystem mit einem großen Anwendungsbereich in Kraftfahrzeugen und industriellen Anlagen.

[2] DLL: Dynamic Link Library

dem Dialogmanager sowohl die Bearbeitung eintreffender Kommandos, als auch die Generierung notwendiger Nachrichten. Durch die Verarbeitung zusätzlicher CAN-Botschaften kann der Funktionsumfang dieser Dialogkomponente sehr leicht erweitert werden. Dies würde sogar die Integration realer Geräte wie Radio, Telefon oder Navigationssystem in die Ausführung des Dialogs ermöglichen. In Bild 56 ist hierzu der CANalyzer während der Steuerung des Controllers dargestellt.

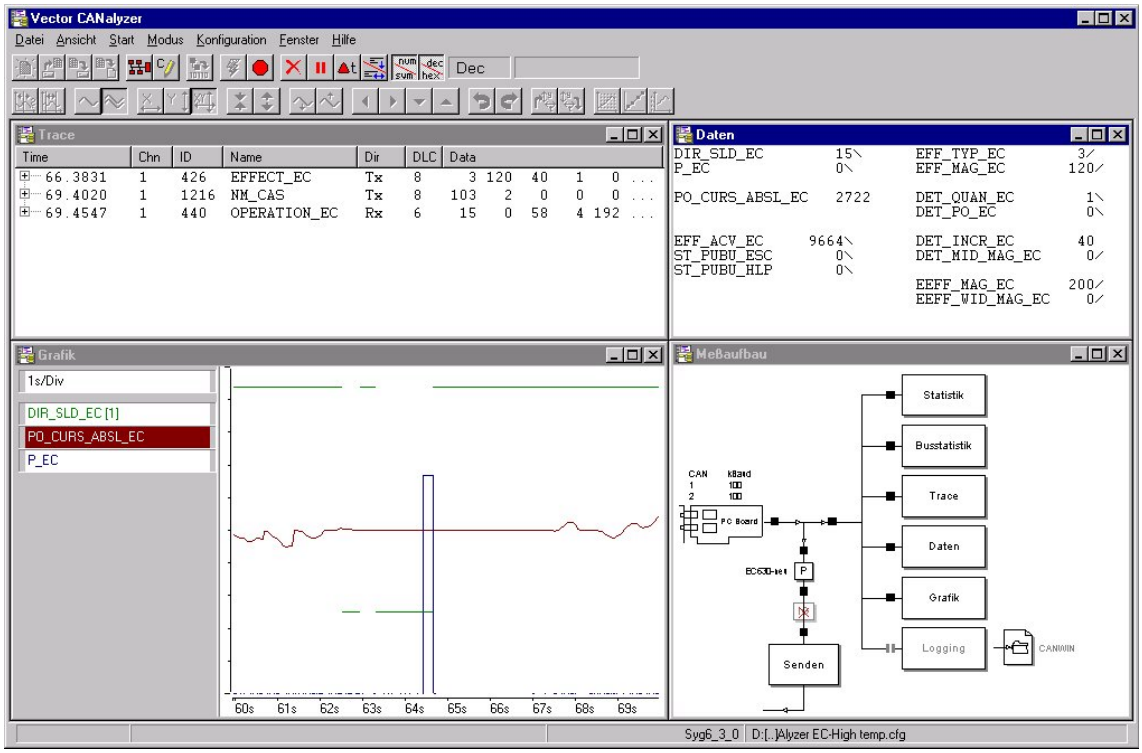


Bild 56 Dialogkomponente zur Anbindung des Controllers über den CAN-Bus

Kapitel 5 Evaluation

Abschließend werden der theoretische Ansatz und die vorgeschlagene Umsetzung evaluiert. Für eine möglichst praxisnahe Bewertung wird mit Experten auf dem Gebiet der HMI-Entwicklung eine Nutzerstudie durchgeführt. Diese müssen dabei mit EmMI ein multimodales Bedienkonzept realisieren.

5.1 Ziele des Nutzertests

Ziel der Studie ist die Bewertung der bereitgestellten Konzepte und Methoden hinsichtlich ihrer Tauglichkeit und Effizienz. Neben den grundsätzlichen Aussagen zur Eignung sollen auch Problemstellen identifiziert werden, die unter Umständen eine wirkungsvolle Bedienung behindern.

Zentrale Bewertungspunkte sind hier:

- Formale Repräsentation
  Sind die semantischen Mittel zur Modellierung der Modalitäten und zur Erstellung multimodaler Dialoge geeignet und ausreichend? An welchen Stellen sind Modifikationen und Erweiterungen im formalen Modell erforderlich?
- Qualitätssicherung der Dialoge
  Inwieweit sind die bereitgestellten Mittel zur Sicherung der Dialogqualität einsetzbar und zur Erstellung „guter Dialoge“ nützlich?
- Aufgabenangemessenheit des Werkzeugs
  Sind die in EmMI umgesetzten Methoden zur visuellen Darstellung und zur Erstellung des Dialogs für den Entwickler verständlich und vor allem effizient? Werden die Möglichkeiten zur zentralen Kontrolle und Konfiguration der Dialogkomponenten begriffen und sinnvoll eingesetzt?

5.2 Versuchsbeschreibung

Um fundierte Aussagen zu diesen Bewertungspunkten treffen zu können, ist zunächst die Auswahl einer sinnvollen Evaluationsmethode erforderlich. Basierend darauf wird ein passender Versuchsplan entwickelt. Im nächsten Schritt erfolgt die Festlegung der Informationsquellen, die als Grundlage der Bewertung dienen sollen. Schließlich werden die Anforderungen an die Versuchspersonen definiert.

5.2.1 Wahl der Evaluationsmethode

Bei der Auswahl der Evaluationsmethode müssen die speziellen Eigenschaften des Bewertungsgegenstands und die Ziele der Untersuchung berücksichtigt werden. In diesem Fall sind zwei grundsätzliche Methoden zur Bewertung denkbar:

- Empirische Evaluation – Benutzertest [Meister 1987]
  Der Evaluationsgegenstand wird einem Benutzertest unterzogen, wobei er mit einem oder mehreren vergleichbaren Produkten oder mit mehreren verschiedenen Konfigurationen verglichen wird. Die Bewertung erfolgt anhand subjektiv vergleichenden Bewertungen der Benutzer und durch objektiv messbare Parameter. Es ist eine relativ große Anzahl an Versuchspersonen erforderlich, wobei diese kein spezielles Domänenwissen bezüglich des Evaluationsinhaltes benötigen.

- Heuristische Evaluation [Nielsen 1993]
  Der Evaluationsgegenstand wird anhand von standardisierten Check- und Kriterienlisten bewertet. Die Beurteilung muss durch Experten auf dem Gebiet des Evaluationsinhaltes erfolgen, da nur diese die dazu notwendige Erfahrung besitzen. Zur Durchführung heuristischer Evaluationen reichen im Allgemeinen drei bis fünf Personen aus.

Bei der Entscheidung für eine dieser beiden Methoden müssen die im Folgenden genannten Randbedingungen bedacht werden. Sie sprechen für eine Bewertung des Werkzeugs angelehnt an den Methoden der Heuristischen Evaluation.

- Als Versuchspersonen sind Experten auf dem Gebiet der HMI-Entwicklung erforderlich, da nur diese das zur Erstellung von Mensch-Maschine-Dialogen notwendige Fachwissen und die entsprechende Erfahrung besitzen.

- Aufgrund dieses erforderlichen Fachwissens sind nur wenige Versuchspersonen und diese mit eingeschränkter Zeit verfügbar.

- Die Versuchspersonen müssen erst mit dem System vertraut gemacht werden. Für eine sinnvolle Bewertung müssen gleichzeitig die zu bearbeitenden Aufgaben eine gewisse Komplexität aufweisen. Die Versuche erfordern daher einen hohen Einarbeitungsaufwand und sind sehr zeitaufwändig in der Durchführung.

- Es existieren bislang keine so spezifisch ausgerichteten Werkzeuge, die einen direkten Vergleich erlauben würden.

5.2.2 Versuchsplan

Für eine praxisnahe Bewertung werden die in 5.1 genannten Evaluationsaspekte anhand einiger bei der Dialogerstellung häufig wiederkehrenden und daher kritischen Szenarien untersucht. Diese drei Szenarien bilden das Grundgerüst des Versuchs:

- *Dialogerstellung*
  Erstellung eines komplett neuen Dialogs. Ausgehend von einem noch leeren Projekt muss die Versuchsperson den Bediendialog vollständig neu aufbauen.

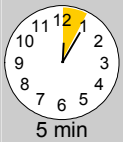
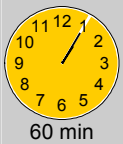
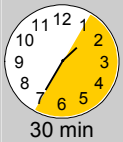
- *Dialogmodifikation*
  Änderung und Erweiterung eines bestehenden Dialogs, der von einer dritten Person erstellt worden ist. Die Versuchsperson muss sich in dem bereits existierenden Projekt zurechtfinden und Modifikationen daran vornehmen.

- *Fehlersuche*
  Auffinden und Beheben von Fehlern in einem bestehenden Dialog. Hierzu muss die Versuchsperson Fehlerbeschreibungen in die formale Darstellung des Bediendialogs umsetzen und dann das jeweilige Problem identifizieren.

Die Versuchspersonen besitzen zwar ein grundsätzliches Verständnis bezüglich der HMI-Entwicklung, allerdings kennen sie das spezifische Werkzeug EmMI nicht. Aus diesem Grund werden sie mit Hilfe eines *Tutorials* an das Werkzeug herangeführt. Dabei werden die einzelnen Funktionen des Werkzeugs schrittweise eingeführt und an einem Beispiel erklärt. Diese neuen Funktionen bauen stets auf bereits vermittelten Informationen auf. Das Tutorial kann allerdings nicht unabhängig von den oben genannten Szenarien bearbeitet werden, weil dadurch zum einen die zeitlichen Dimensionen des Versuchs gesprengt werden würden und zum anderen auftretende Verständnisprobleme beim Erstkontakt nicht ausreichend erfasst werden könnten. Deshalb wird es in den ersten Abschnitt der Versuchsbearbeitung – der Dialogerstellung – integriert. Das Tutorial und die Anleitungen für die beiden anderen Versuchsabschnitte liegen den Versuchspersonen in schriftliche Form vor und garantieren so eine einheitliche Aufgabeneinweisung. Dies birgt allerdings die Gefahr, dass Verständnisschwierigkeiten in diesen Anleitungen die Gesamtbewertung beeinflussen. Deshalb werden in drei Iterationen mit jeweils einer Vorversuchsperson zunächst die inhaltliche Verständlichkeit und Fehlerfreiheit der Versuchsanleitungen sichergestellt.

Der zeitliche Ablauf des Versuchs ist in Tabelle 7 dargestellt. Nach einer kurzen Einweisung durch den Versuchsleiter beginnt die Aufgabenbearbeitung. Sowohl im Tutorial als auch bei der Dialogmodifikation werden jeweils zwölf Aufgaben bearbeitet. Sie sind dabei zu vier bzw. drei Aufgabenblöcken zusammengefasst. Da keine weiteren Erklärungen erforderlich sind, halbiert sich bei der Dialogmodifikation die Bearbeitungszeit auf 30 Minuten. Bei der Fehlersuche erhalten die Versuchspersonen Beschreibungen von sechs Fehlern, wie sie beispielsweise in einem Nutzertest entdeckt werden könnten. Die Aufgabe besteht darin, diese Fehler in einem vorbereiteten Projekt zu lokalisieren und zu beheben. Abschließend werden die persönlichen Eindrücke der Versuchspersonen zum Werkzeug mit Hilfe eines Fragebogens festgehalten. Die Versuchsanleitungen und Fragebögen sind im Anhang C abgedruckt.

Tabelle 7 Versuchsablauf

Dauer	Versuchsabschnitt
 5 min	**Einführung in EmMI**
 60 min	**Tutorial/ Dialogerstellung** 1. Aufgabenblock 2. Aufgabenblock 3. Aufgabenblock 4. Aufgabenblock
 30 min	**Dialogmodifikation** 1. Aufgabenblock 2. Aufgabenblock 3. Aufgabenblock
 30 min	**Fehlersuche**
 15 min	**Fragebogen zu EmMI**

5.2.3 Datenerhebung

Um sowohl objektive Messdaten als auch subjektive Beurteilungen der Versuchspersonen in die Bewertung einfließen zu lassen, werden folgende Informationsquellen herangezogen:

- Das strukturierte Protokoll des Versuchsleiters, welches bei kritischen Stellen anhand einer Videoaufzeichnung nachträglich ergänzt werden kann. Hierzu wird jede Aufgabe in einzelne Aufgabenschritte unterteilt. Während des Versuchs werden die nicht erfolgreich bearbeiteten Aufgabenschritte und die dafür beobachteten Gründe protokolliert. Gleichzeitig erfolgt eine Einteilung der Fehler in Verständnis-, Ausführungs- und sonstige Fehler. Dies erfolgt in Anlehnung an die Fehlerklassifikation nach [Reason 1990]. In Tabelle 8 ist exemplarisch für eine Aufgabe ein solches Fehlerprotokoll dargestellt.

- Verständnisfragen an die Versuchspersonen während des Versuchs. Während der Dialogmodifikation im zweiten Teil der Aufgabenbearbeitung werden gezielt Fragen gestellt, um das bisherige Verständnis der Versuchspersonen bezüglich der formalen Dialogrepräsentation und des Werkzeugs zu überprüfen.
- Anmerkungen der Versuchspersonen während des Versuchs. Diese meist spontanen Äußerungen sind insbesondere bei der Identifikation von besonders positiv oder negativ empfundenen Aspekten des Werkzeugs sehr aufschlussreich.
- Fragebogen nach dem Versuch. Damit wird die subjektive Beurteilung der Versuchspersonen zu den verschiedenen Bewertungspunkten (5.1) explizit abgefragt.

Tabelle 8 Auszug aus dem strukturierten Protokoll

Aufgabe: Reaktion anlegen	**V**	**A**	**S**	**Bemerkung**
Dialogzustand auswählen				
Reiter auswählen „Reaction“				
Nachricht auswählen	*X*			*Bestehende globale Nachricht gewählt => Gültigkeitsbereich nicht verstanden*
Kommando – klicken				
Kommando – auswählen				
Kommando – Parameter auswählen				
Kommando – verknüpfter Parameterwert		*X*		*Zuweisung Array unklar => GUI-Problem*

V: Verständnisproblem A: Ausführungsproblem
S: Sonstiges Problem

5.2.4 Versuchspersonen

Neben den drei Vorversuchspersonen stehen für den eigentlichen Nutzertest insgesamt vier Experten bezüglich Mensch-Maschine-Interaktion im Fahrzeug zur Verfügung. Sie arbeiten in den Bereichen Forschung, Design und Entwicklung und besitzen jeweils eine akademische Ausbildung zum Elektrotechniker oder Informatiker. Sie sind somit mit den Problemen bei der Entwicklung von Bediendialogen vertraut und können gleichzeitig die in EmMI umgesetzten Konzepte und Methoden beurteilen. Diese Zahl an Versuchspersonen korrespondiert mit der in [Nielsen 1993] vorgeschlagenen Menge von drei bis fünf Personen zur Erreichung eines optimalen Kosten-Nutzen-Verhältnisses.

5.3 Ergebnisse

Im Folgenden werden die Ergebnisse des Nutzertests dargestellt. Die Auswertung beginnt mit einer Untersuchung der grundsätzlichen Effizienz der Versuchspersonen während der Aufgabenbearbeitung. Im nächsten Schritt werden die speziellen Problemstellen dargestellt, klassifiziert und diskutiert. Den Abschluss bildet eine Betrachtung des konzeptuellen Verständnisses und der subjektiven Bewertung der Versuchspersonen.

5.3.1 Effizienz der Aufgabenbearbeitung

Als erstes wird in den ersten beiden Versuchsabschnitten die Effizienz untersucht, mit der die Versuchspersonen das Werkzeug einsetzen können. Ein häufig benutztes Maß ist hier die notwendige Bearbeitungszeit zur Durchführung bestimmter Aufgaben durch trainierte Benutzer. Obwohl die Versuchspersonen Experten auf dem Gebiet der HMI-Entwicklung sind, besitzen sie keine Erfahrung mit dem speziellen Werkzeug EmMI. Deshalb ist die Bearbeitungszeit zur Bestimmung der Effizienz nicht geeignet, da sie sehr stark vom jeweiligen Lernverhalten der Versuchsperson beeinflusst wird. Die Effizienz wird deshalb lediglich auf den Erfolg der Bearbeitung bezogen. Sie spiegelt dadurch zwar auch den Lernerfolg wider, ist aber vom individuell notwendigen zeitlichen Aufwand entkoppelt. Hierzu wird zunächst jede Aufgabe in einzelne Arbeitsschritte unterteilt. Die Effizienz der Bearbeitung definiert sich dann folgendermaßen:

$$\text{Effizienz} := \frac{\text{Anzahl korrekt gelöster Arbeitsschritte}}{\text{Anzahl notwendiger Arbeitsschritte}}$$

Die Zahl der notwendigen Arbeitsschritte ergibt sich aus den zur Bearbeitung des jeweiligen Aufgabentyps erforderlichen Teilschritten. Sie korrespondiert daher mit der Zeilenzahl im strukturierten Protokoll des Versuchsleiters für die entsprechende Aufgabe (vgl. Tabelle 8).

Beispiel:

Die in Tabelle 8 dargestellte Aufgabe umfasst sieben Arbeitsschritte. Davon werden fünf korrekt bearbeitet. Die Effizienz der Bearbeitung bei dieser Aufgabe beträgt daher

$$\text{Effizienz} = \frac{5}{7} = 0{,}71$$

5.3.1.1 Effizienz über alle Aufgaben

Für eine globale Betrachtung erfolgt zunächst eine Untersuchung des gesamten Effizienzverlaufs. Bild 57 zeigt hierzu, dass die Effizienz der Aufgabenbearbeitung bei allen Versuchspersonen grundsätzlich sehr hoch ist. Besonders bei der ersten Versuchsperson (VP 1), aber auch bei allen anderen sinkt die Effizienz tendenziell am Anfang etwas ab und steigt dann wieder an. Dies lässt sich dadurch erklären, dass die zunächst sehr starke Führung durch das Tutorial ab Aufgabe R1.5 deutlich reduziert wird.

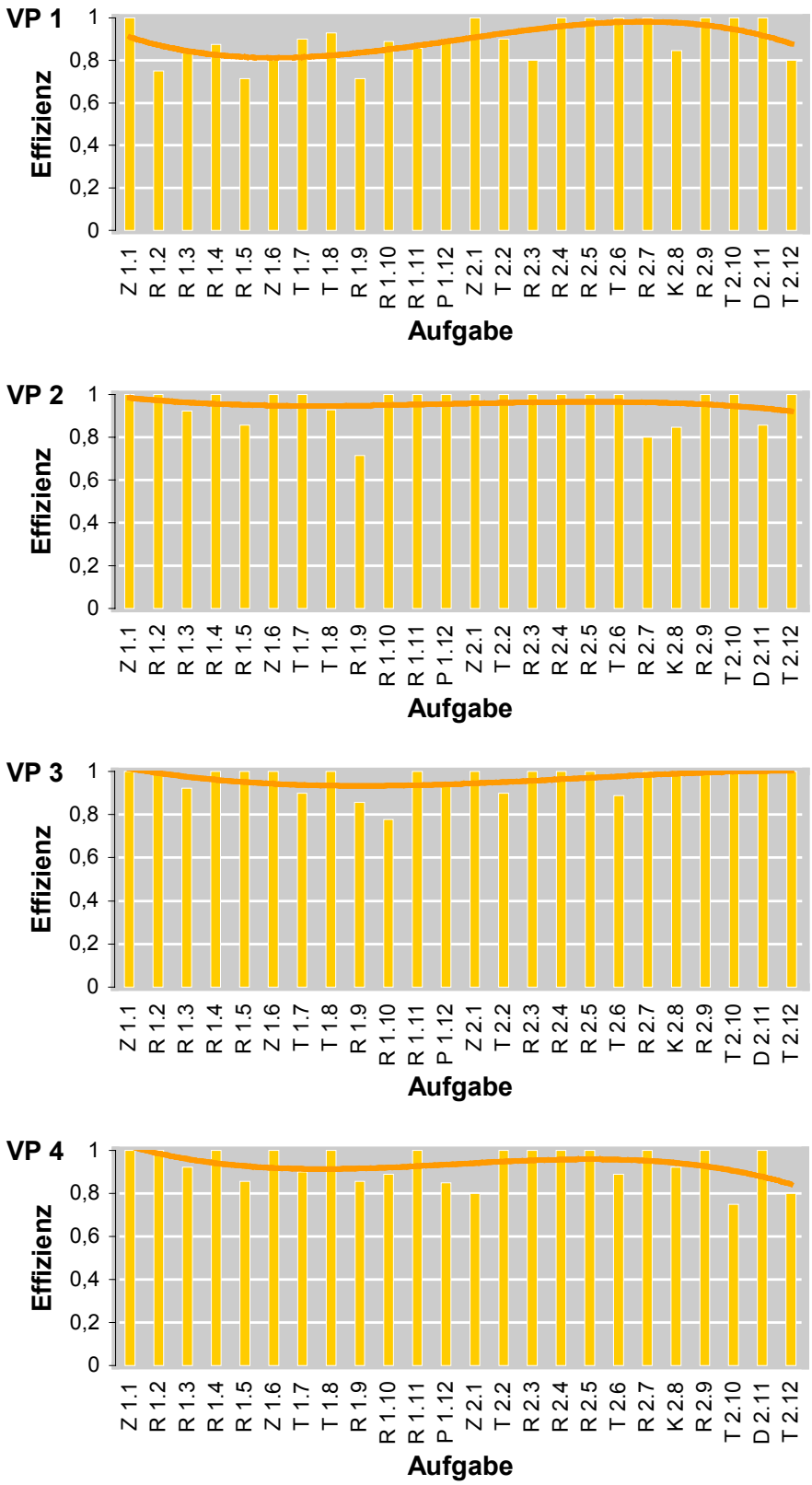


Bild 57 Effizienz aller Aufgaben (Tutorial und Modifikation)

Betrachtet man die Effizienz lediglich ab dieser Aufgabe R1.5 (Bild 58), so sind insbesondere bei VP 1 und VP 3 positive Trends festzustellen. Allerdings sind diese Trends nicht besonders ausgeprägt. Hierzu lassen sich mehrere Gründe anführen:

- Die Effizienz ist allgemein sehr hoch, so dass deutlichere Steigerungen nicht möglich sind.
- Es kommen im Verlaufe des Experiments immer wieder neue Aufgabentypen hinzu, die beim erstmaligen Auftreten eine geringere Effizienz aufweisen (z.B. R1.9 und K2.8).
- Die Zahl der Bearbeitungsschritte für eine Aufgabe ist nicht besonders groß. Deshalb kann ein einziger Fehler bereits für einen relativ großen Ausreißer in der Effizienz sorgen. (Beispiel: 1 Fehler bei n=5 Arbeitsschritten: $\Delta Eff=0.2$)

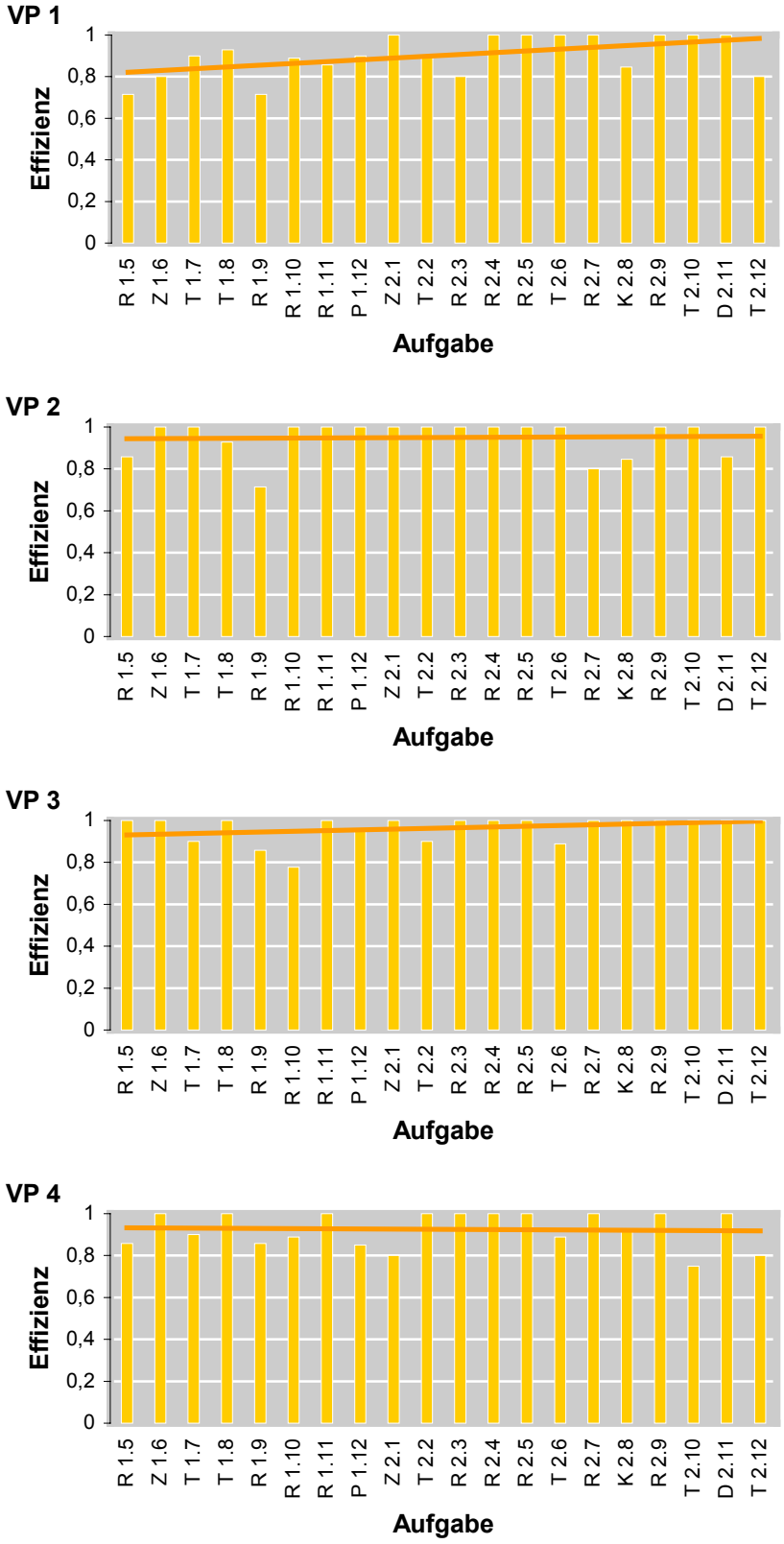


Bild 58 Effizienz aller Aufgaben ohne Betrachtung der Einführung (Tutorial und Modifikation)

5.3.1.2 Effizienz über spezifische Aufgabentypen

Eine detailliertere Sicht auf die Effizienz der Aufgabenbearbeitung erlaubt die getrennte Betrachtung der beiden für die Dialogserstellung wichtigsten Aufgabentypen, der Reaktionen und Transitionen:

- Reaktionen
  Mit Hilfe von Reaktionen kann der Entwickler das funktionale Verhalten der Dialogkomponenten festlegen (3.2.3). Hierzu muss im Wesentlichen eine eintreffende Nachricht mit den auszulösenden Kommandos verknüpft werden (4.2.2.2). In Bild 59 ist bei den Reaktionen eine Stabilisierung der Effizienz bei 100% erkennbar. Dies lässt den Schluss zu, dass die Versuchspersonen nach der Lernphase mit den Reaktionen keinerlei Probleme mehr haben.

- Transitionen
  Die Übergänge zwischen den einzelnen Dialogzuständen werden als Transitionen bezeichnet (3.2.2). Die Hauptaufgabe besteht dabei darin, den Gültigkeitsbereich sowie den Zielzustand festzulegen und mit der auslösenden Nachricht zu verbinden (4.2.2.2). Wie in Bild 60 dargestellt ist, bewegt sich die Effizienz der Bearbeitung von Transitionen ebenfalls auf einem sehr hohen Niveau. Allerdings ist außer bei VP 2 keine Stabilisierung zu beobachten. Der Grund hierfür könnte in der im Vergleich zu den Reaktionen geringeren Zahl an Aufgaben sein. Der Lernprozess für die Transitionen scheint also bei den meisten Versuchspersonen noch nicht abgeschlossen zu sein.

Wie in 5.3.5.2 gezeigt wird, korrespondieren diese objektiven Ergebnisse in sehr guter Weise mit den subjektiven Bewertungen der Versuchspersonen.

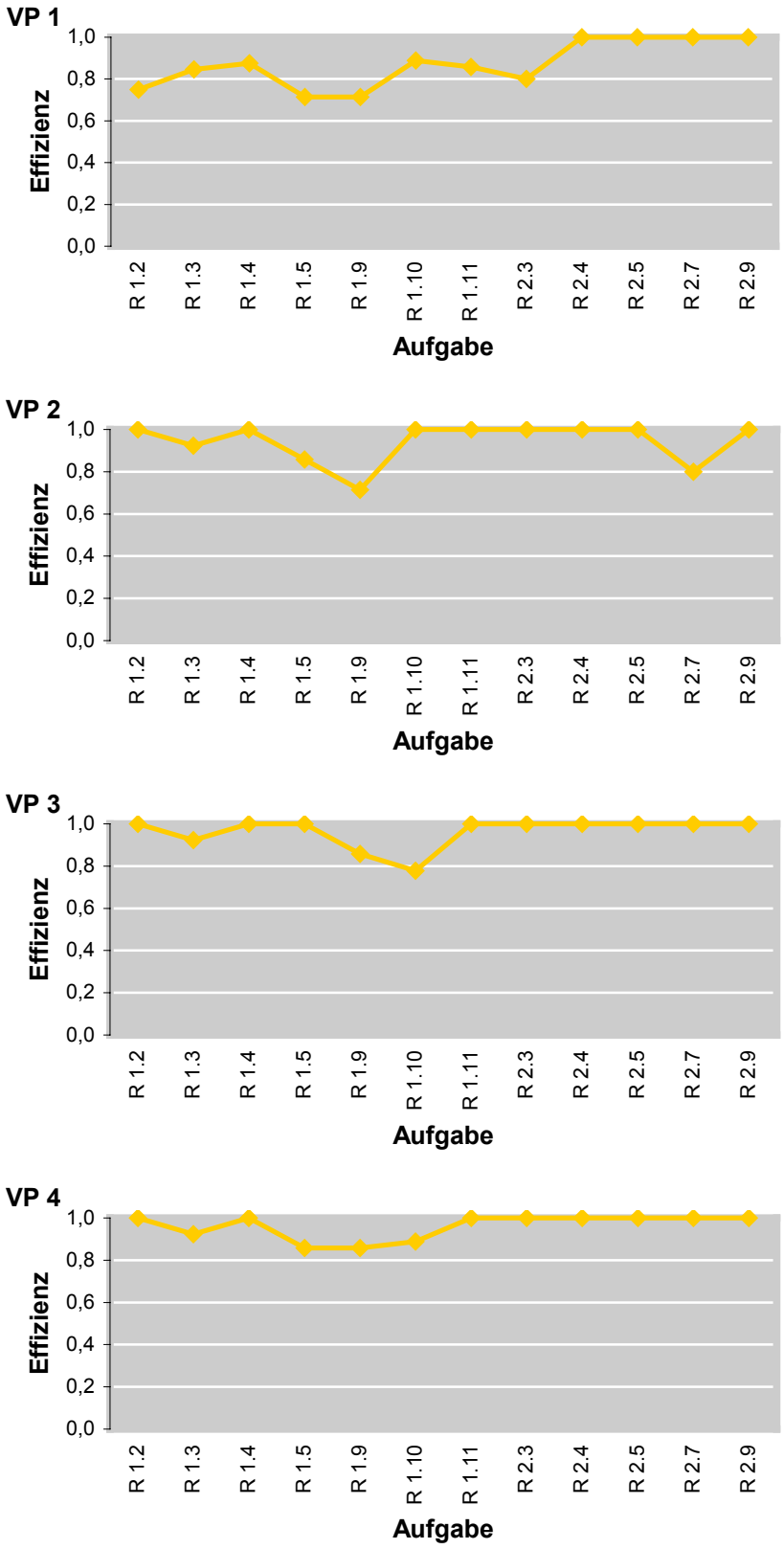


Bild 59 Effizienz der Reaktionen (Tutorial und Modifikationen)

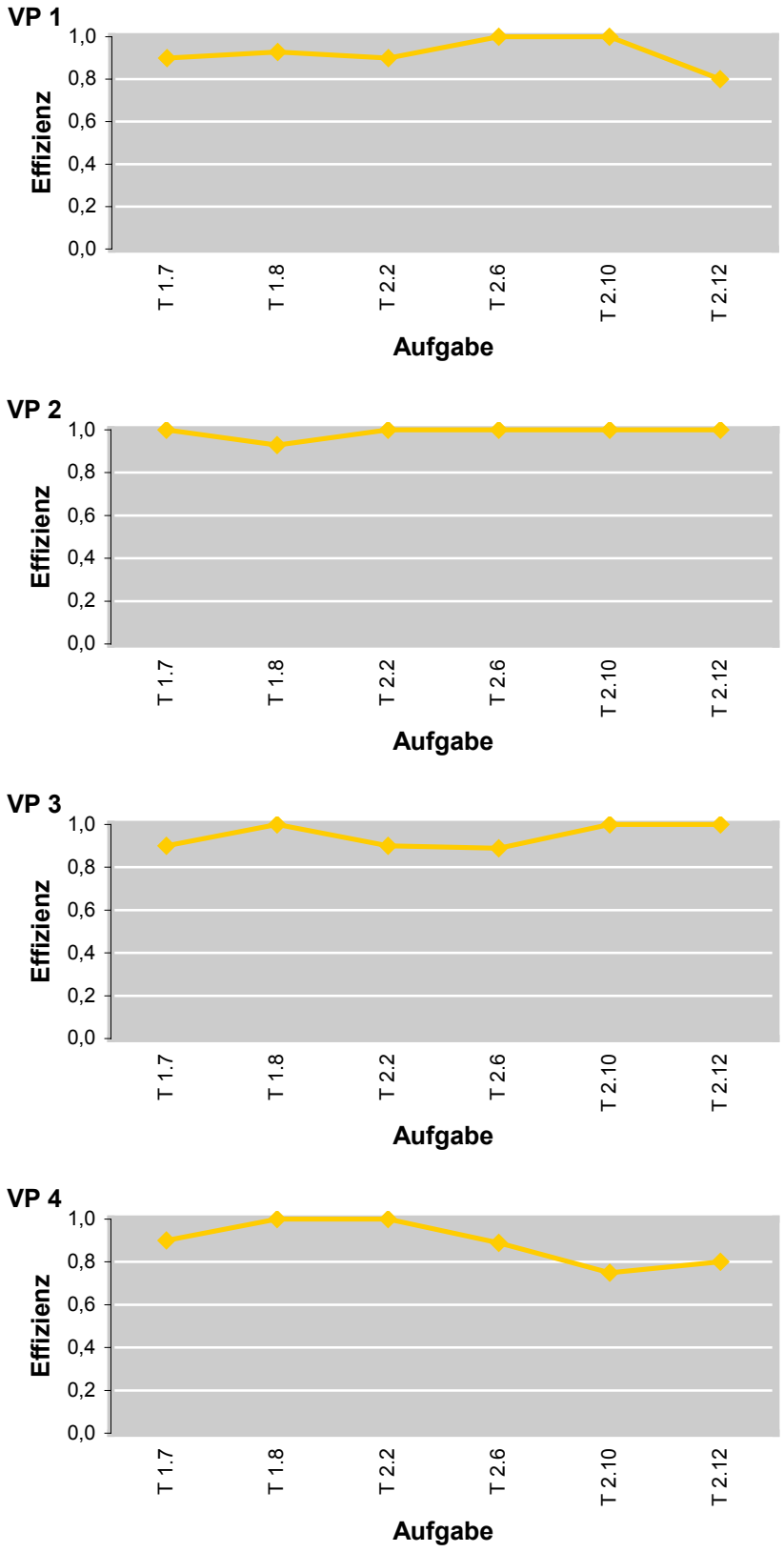


Bild 60 Effizienz der Transitionen (Tutorial und Modifikationen)

5.3.2 Problemlokalisierung

Neben dieser globalen Betrachtung der bearbeiteten Aufgaben werden nun die spezifischen Problemstellen untersucht. Die Problemlokalisierung und -bewertung basiert dabei auf den Anmerkungen des Versuchsleiters zu den einzelnen Aufgabenschritten im strukturierten Protokoll (Tabelle 8).

Insgesamt bearbeiten die vier Versuchspersonen im Tutorial und während der Dialogmodifikation 96 Aufgaben mit insgesamt 780 Arbeitsschritten. Bei 50 dieser 780 Arbeitsschritte können Schwierigkeiten beobachten werden (Bild 61). Das entspricht einem Anteil von 6,4%. Die Problemstellen sind allerdings nicht über alle Versuchspersonen gleich verteilt. Bei VP 1 und VP 4 können mehr Schwierigkeiten als bei VP 2 und VP 3 festgestellt werden. Der insgesamt sehr geringe Fehleranteil weist darauf hin, dass die Versuchspersonen mit dem Werkzeug gut zurecht kommen. Dies ist auch auf das – zumindest am Anfang – sehr detaillierte Tutorial zurückzuführen.

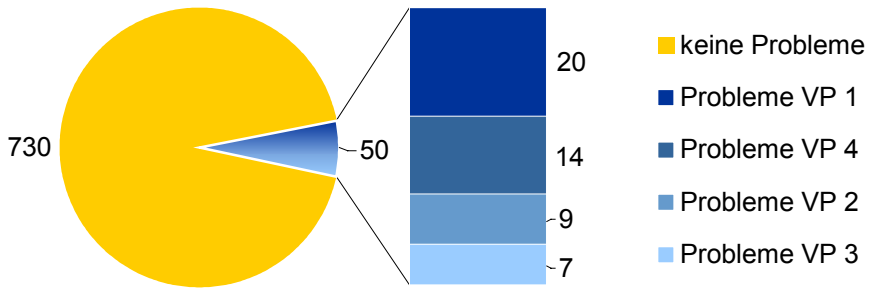


Bild 61 Verteilung problembehafteter Arbeitsschritte auf die Versuchspersonen

5.3.2.1 Probleme bei den einzelnen Aufgabenschritten der Aufgaben

Nun werden die kritischen Arbeitsschritte identifiziert, die offenbar leicht zu Bedienungsschwierigkeiten bei den Versuchspersonen führen. Hierzu sind in Bild 62 diejenigen Arbeitsschritte dargestellt, bei denen im Versuch Probleme beobachtet werden können. Sie sind nach der Zahl der Versuchspersonen geordnet, die bei der Bearbeitung mindestens einmal Schwierigkeiten haben. Das heißt je mehr Versuchspersonen bei einem Aufgabenschritt Fehler machen, desto höher wird er in der Liste eingestuft.

Die Aufgabenschritte, bei denen am häufigsten Probleme auftreten, sind:

- Die Eingabe von Parameterwerten bei der Parametrisierung von Kommandos
- Die Zuordnung des Gültigkeitsbereichs bei Werten für Konfigurationsparameter
- Die Auswahl von Dialogzuständen bei Reaktionen
- Die Auswahl von Nachrichten bei Reaktionen
- Die Auswahl von Dialogzuständen bei Transitionen

Insgesamt treten bei 21 der maximal 43 Aufgabenschritte Probleme auf. Dabei ist wiederum eine deutliche Zweiteilung der Versuchspersonengruppe zu erkennen. VP 1 und VP 4 haben bei 12 bzw. 11 Aufgabenschritten Probleme, was um den Faktor 1.6 bis 2.0 höher liegt als bei VP 2 und VP 3. Eine ähnliche Aussage ergibt sich auch aus den in Bild 61 dargestellten absoluten Fehlerzahlen.

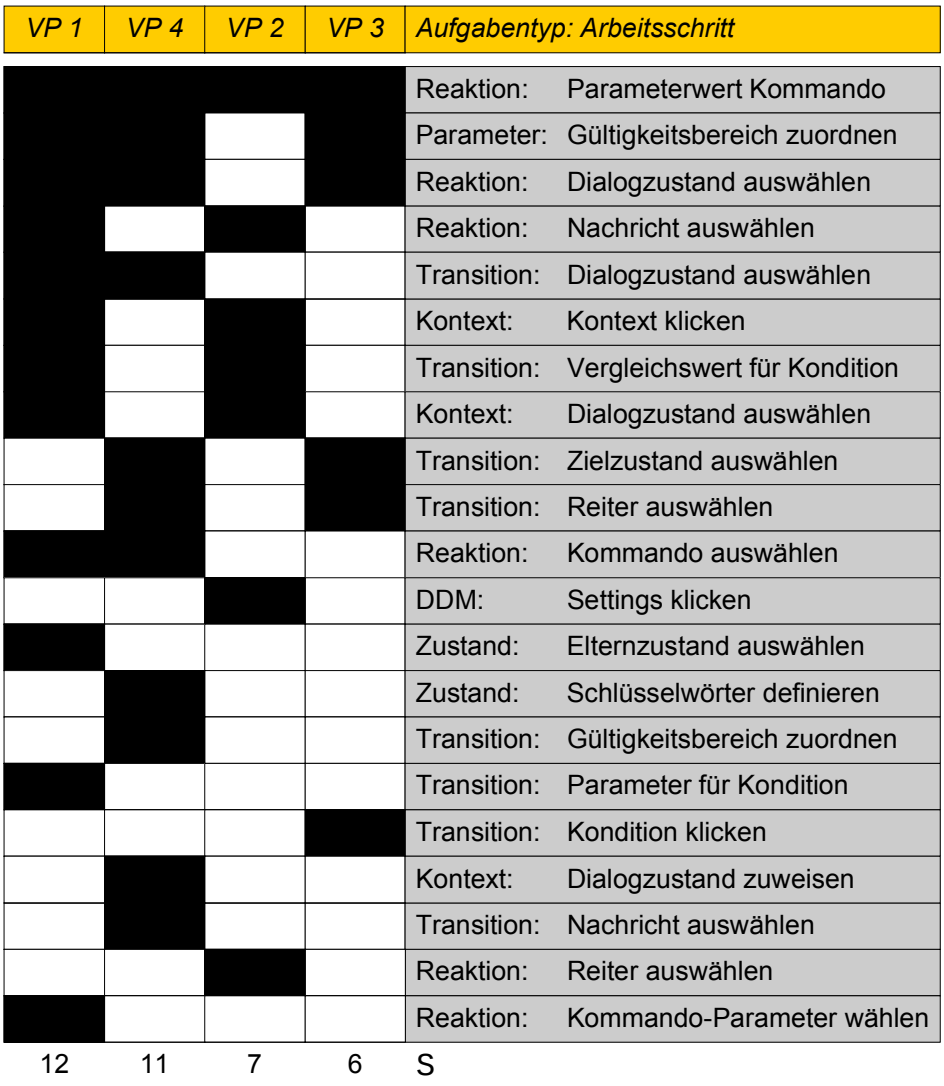


Bild 62 Problemlokalisierung bei den Aufgabenschritten

5.3.2.2 Identifikation von Problemauslösern – Fehlertaxonomie

Die Darstellung in 5.3.2.1 zeigt zwar die Arbeitsschritte, bei denen Fehler vorkommen, allerdings liefert sie keine Hinweise, warum diese Probleme auftreten. Aus diesem Grund wird mittels einer Klassifikation der Problemauslöser eine Fehlertaxonomie erstellt. Diese Gruppierung der Fehler zu Problemklassen basiert auf den Beobachtungen des Versuchsleiters und den Anmerkungen der Versuchspersonen. In Anlehnung an das Klassifikationsschema für mensch-

liche Fehler nach [Reason 1990] in *mistakes*, *slips* und *lapses* können drei Hauptkategorien für die Bearbeitungsprobleme identifiziert werden:

1. Verständnisfehler (*mistakes*)
   Fehler bei der Bildung einer Handlungsabsicht, da die Versuchsperson die Aufgabe nicht auf das zur Verfügung stehende Methoden- und Werkzeuginventar abbilden kann. Bei niedrigem Lern- bzw. Erfahrungsniveau im Umgang mit einem Werkzeug lassen sich diese Fehler meist auf ein fehlendes Verständnis bei der Verwendung des Werkzeugs zurückführen.

2. Ausführungsfehler (*slips* und *lapses*)
   Die Versuchsperson weiß, was sie tun möchte, es treten aber Schwierigkeiten bei der Durchführung auf. Diese Probleme können häufig durch Modifikationen im HMI des Werkzeugs vermieden werden. In dieser Nutzerstudie können bei den Ausführungsfehlern fünf Problemklassen beobachtet werden:

   - Arbeitsschritt vergessen
     Bei der Ausführung der Aufgabe wird ein Arbeitsschritt vergessen.
   - Graphische Darstellung
     Der Fehler ist auf eine uneindeutige graphische Darstellung zurückzuführen.
   - Unachtsamkeit
     Das Problem tritt durch fehlende Aufmerksamkeit bei der Durchführung der Aufgabe auf.
   - Vorgehensweise
     Das Vorgehen bei der Durchführung der Handlungsabsicht ist nicht klar, obwohl den Versuchspersonen ihr Ziel klar ist.
   - Auffinden
     Die Versuchsperson geht zunächst richtig vor, um das Ziel zu erreichen. Sie gerät aber in ein Problem, weil sie beispielsweise einen Listeneintrag übersieht und deshalb glaubt einen Fehler gemacht zu haben.

3. Sonstige Fehler
   Hierunter fallen untypische Fehler, die nichts mit der Bildung oder der Durchführung einer Handlungsabsicht zu tun haben:

   - Eigenschaften Dialogkomponente
     Der Versuchsperson ist das funktionale Verhalten einer Dialogkomponente nicht genügend klar, um die Aufgabe bearbeiten zu können.
   - Bug
     Im Werkzeug tritt ein Fehler auf, der einen Eingriff des Versuchsleiters notwendig macht.

- Herumspielen
  Die Versuchsperson weicht von der Anleitung ab, wodurch die Aufgabenbearbeitung gestört wird.

Wie aus der Fehlertaxonomie in Bild 63 zu entnehmen ist, können deutlich mehr Ausführungsfehler (35) als Verständnisfehler (9) beobachtet werden. Diese geringe Zahl an Verständnisfehlern zeigt, dass die untersuchten Konzepte und Methoden von den Versuchspersonen gut verstanden werden. Dies liefert ein deutliches Indiz für die prinzipielle konzeptuelle Tauglichkeit des Werkzeugs.

Die höhere Zahl an Ausführungsfehlern ist ebenfalls nicht überraschend. Diese Art von Fehlern sind notwendiger Bestandteil der Lernphase. Trotzdem dürfen sich nicht unbeachtet bleiben, weil sie im Allgemeinen Hinweise auf sinnvolle Verbesserungen vor allem im Darstellungsbereich liefern. Dies trifft insbesondere dann zu, wenn bei mehreren Versuchspersonen die gleichen Probleme festgestellt werden können.

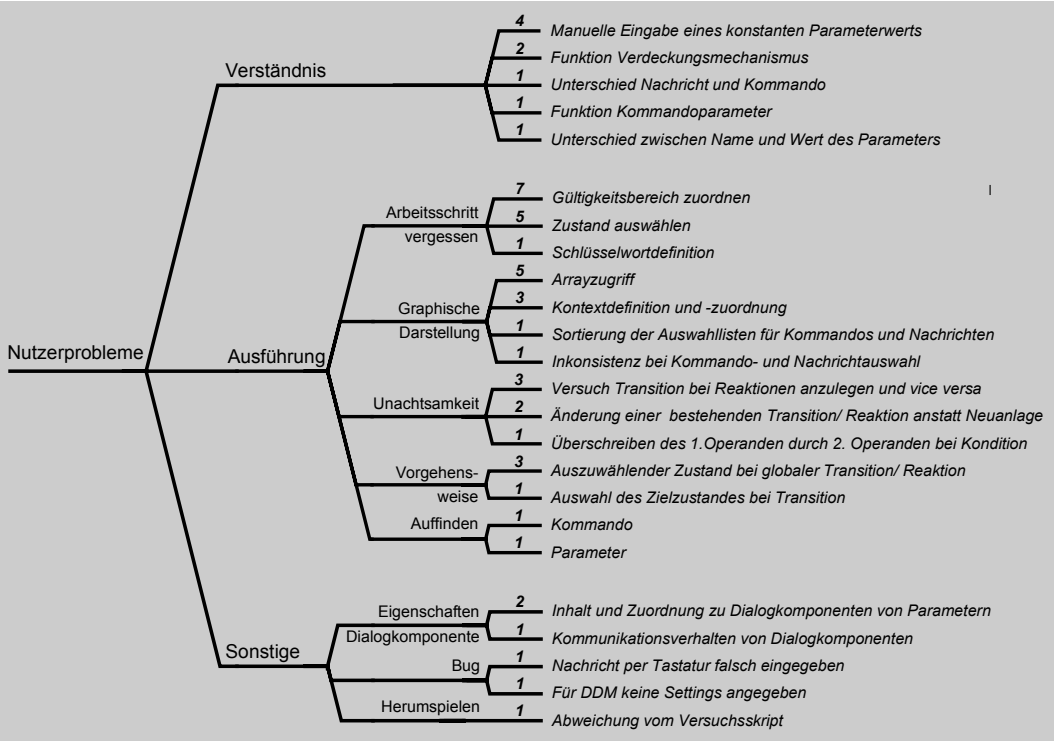


Bild 63 Fehlertaxonomie der Nutzerprobleme und Auftrittshäufigkeit

5.3.2.3 Diskussion signifikanter Problemstellen

Im Folgenden werden die auffälligsten Nutzerprobleme der Fehlertaxonomie im Kontext der jeweiligen Aufgaben diskutiert und gleichzeitig mögliche Abhilfen beziehungsweise Änderungsvorschläge erörtert. Im Anhang C.V findet sich hierzu eine vollständige Zuordnung der Nutzerprobleme zu den verschiedenen Arbeitsschritten mit Bearbeitungsproblemen.

- Manuelle Eingabe eines konstanten Parameterwerts
  Dieses Verständnisproblem ist meist dann zu beobachten, wenn bei der Parametrisierung eines Kommandos offensichtlich nur bestimmte Werte für einen Parameter zugelassen sind. Den Versuchspersonen ist dann nicht klar, dass ihnen dieser Wert nicht zur Auswahl steht, sondern sie ihn tatsächlich eintippen müssen. Dieser Fehler ist normalerweise nur einmal pro Versuchsperson zu beobachten. Er kann durch die Bereitstellung der möglichen Werte als Auswahlliste vermieden werden, wodurch gleichzeitig Eingabefehler vermieden werden.

- Arbeitsschritt vergessen – Gültigkeitsbereich zuordnen
  Die Zuordnung des Gültigkeitsbereichs ist in sechs der sieben Fälle bei der Aufgabe P1.12 vergessen worden. Von diesen sechs fallen allein drei auf VP 4. In dieser Aufgabe wird eine neue Funktion der Entwicklungsumgebung eingeführt. Die Versuchspersonen sind anscheinend mit dieser neuen Funktion so sehr beschäftigt, dass sie vergessen den Gültigkeitsbereich anzupassen. Unter Umständen weist das Tutorial an dieser Stelle nicht genügend genau darauf hin.

- Arbeitsschritt vergessen – Zustand auswählen
  Die Versuchsperson vergisst den für die jeweilige Aufgabe notwendigen Dialogzustand auszuwählen. Dieses Problem tritt an mehreren Stellen auf und kann daher keiner bestimmten Aufgabe zugeordnet werden. Entsprechend den Aussagen einiger Versuchspersonen liegt es wahrscheinlich daran, dass der aktuell ausgewählte Dialogzustand visuell nicht genügend deutlich hervorgehoben ist und es dadurch leicht zu Verwechslungen kommt. An dieser Stelle sollte eine Überarbeitung der graphischen Gestaltung der Icons im State Tree (4.2.2.1) und der entsprechenden Anzeige im State Specifier (4.2.2.2) in Betracht gezogen werden.

- Graphische Darstellung – Arrayzugriff
  Da Parameterwerte grundsätzlich als Arrays, also als Vektoren, behandelt werden, kann bei der Kommandoparametrisierung ebenfalls ein ganzes Parameterarray übergeben werden. Hierzu muss lediglich im Feld „ArrayNr“ des Dialogfelds „Enter Value“ kein Wert angegeben werden (Bild 64). Hierbei haben allerdings alle Versuchspersonen Probleme, weil dieses Vorgehen nicht offensichtlich ist. Durch die Einführung einer Check-Box zur Auswahl des gesamten Arrays könnte die graphische Darstellung dieser Funktion verbessert und das Problem vermieden werden.

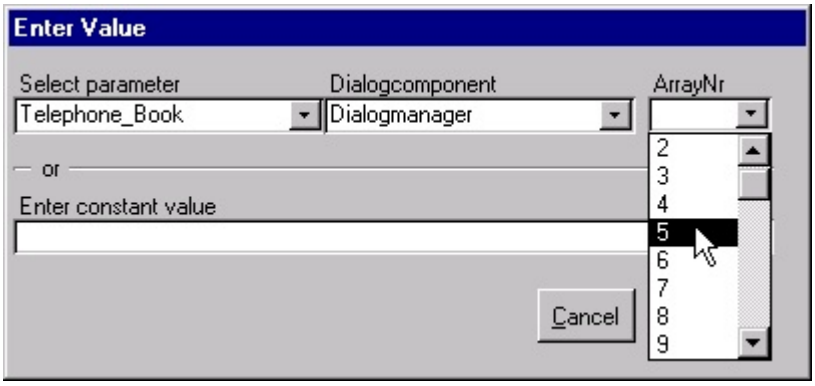


Bild 64 Nutzerproblem bei der Verwendung eines Parameterarrays zur Festlegung eines Parameterwertes

- Graphische Darstellung – Kontextdefinition und -zuordnung
  Im Reiter „State Description“ gibt es eine Schaltfläche „Contexts“ und eine Listenüberschrift „Context“ (Bild 45). Die Versuchspersonen sind durch die doppelte Verwendung dieses Begriffs verwirrt. Dieses Problem kann durch eine eindeutige Verwendung der Bezeichner sehr leicht aufgelöst werden.

- Unachtsamkeit – Versuch Transition bei Reaktion anzulegen und vice versa
  Bei drei Versuchspersonen kann jeweils einmal der Versuch beobachtet werden, eine Transition im Reiter für die Reaktionen anzulegen und umgekehrt. Der Grund hierfür liegt wahrscheinlich im ähnlichen Aufbau der beiden Reiter im oberen Bereich (Bild 47 und Bild 48). Dieser ähnliche Aufbau ist allerdings auch gewünscht, weil dadurch die bereits gelernte Vorgehensweise leichter auf die neue Funktion transferiert werden kann. Die Versuchspersonen bemerken ihren Fehler meist von alleine, weil ihnen im zweiten Teil des jeweiligen Dialogfelds entweder Informationen für die vorhandenen Eingabefelder oder eben Eingabefelder für die noch nicht spezifizierten Informationen fehlen. Daneben überprüft das Werkzeug, ob beispielsweise bei der Angabe einer Transition der Zielzustand gewählt ist, sodass bereits die Eingabe der ersten Information nicht möglich ist. Eine deutlichere graphische Unterscheidung der Reiter „Reaction“ und „Transition“ wäre sinnvoll, wenn sichergestellt werden kann, dass der Wissenstransfer bei der Aufgabendurchführung nicht darunter leidet.

- Vorgehensweise – Auszuwählender Zustand bei globaler Transition/ Reaktion
  Um eine globale Transition bzw. Reaktion anzulegen ist es egal, in welchem Zustand diese Definition erfolgt. Aus diesem Grund wird hierzu im Tutorial ab einem gewissen Zeitpunkt kein spezieller Zustand mehr angegeben. Dies bereitet VP 1 und VP 2 zunächst Schwierigkeiten bei der Umsetzung, obwohl ihnen das Ziel der Aufgabe klar ist. Dieses Problem wird in kurzer Zeit durch Lernen überwunden.

Diese Analyse zeigt, dass fast alle der wiederholt aufgetretenen Problemstellen als Ausführungsfehler klassifiziert werden können, deren Ursachen in der geringen Lernzeit, Unaufmerksamkeiten der Versuchspersonen oder in Gestaltungsmängel des Dialog Modeler zu suchen sind. Viele der Auslöser für diese Problemstellen können mit relativ geringem Aufwand beseitigt

werden, wobei bestimmte Schwierigkeiten aber auch ein wichtiges Element der Lernphase darstellen. Wichtig anzumerken ist der sehr geringe Anteil an Verständnisfehlern. Dies deutet auf ein hervorragendes Verständnis der Experten bezüglich des zugrunde liegenden formalen Modells hin. Die insgesamt niedrige Zahl an Problemstellen weist EmMI als eine sehr sinnvolle Umsetzung des theoretischen Konzepts aus, das selbst beim Erstkontakt bereits einfach zu verwenden ist.

5.3.3 Verständnisfragen

Während der Durchführung der Dialogmodifikation werden Verständnisfragen an die Versuchspersonen gestellt, um ihr bisheriges Verständnis bezüglich der formalen Dialogrepräsentation und des Werkzeugs zu überprüfen. Die Fragen sind in der Versuchsanleitung an den betreffenden Stellen eingefügt und müssen schriftlich bearbeitet werden (Anhang C.II).

Ein Aufgabenbereich in diesem Versuchsabschnitt ist die Realisierung einer Lautstärkeeinstellung. Die Versuchspersonen müssen als Erstes mit eigenen Worten beschreiben, was zur Umsetzung dieser Volume-Funktion zu tun ist. Die Ergebnisse zeigen, dass die Versuchspersonen bereits zu diesem Zeitpunkt ein grundsätzliches Verständnis bezüglich der Funktionsweise des Werkzeugs entwickelt haben. Daher können sie die aus der Aufgabenstellung hervorgehenden Teilschritte gut umsetzen. Nicht genannte Detailaspekte berücksichtigen sie kaum, weil ihnen die Routine im Umgang mit dem Werkzeug und das notwendig Wissen über die Funktionsweise der einzelnen Dialogkomponenten fehlt.

Daneben zeigt die fast vollständig richtige Bearbeitung der restlichen Verständnisfragen, dass die Versuchspersonen die Aufgaben nicht nur mechanisch umsetzen, sondern ein gutes Verständnis für das zugrunde liegende theoretische Konzept in EmMI entwickelt haben. Dies bestätigt die Ergebnisse in 5.3.1 und 5.3.2.

5.3.4 Fehlersuche

Im dritten Abschnitt des Versuchs müssen die Versuchspersonen anhand einer schriftlichen Problembeschreibung Fehlerstellen in einem bestehenden Projekt identifizieren und korrigieren. Dabei wird der größte Teil der Fehler von den Versuchspersonen sehr zielstrebig behoben (Bild 65). Dies liefert ein weiteres Indiz dafür, dass die Versuchspersonen bereits ein sehr gutes konzeptuelles Verständnis bezüglich der formalen Repräsentation des Dialogs und der Modalitäten erlangt haben. Gleichzeitig bestätigt es auch die Eignung des Werkzeugs zum nachträglichen Auffinden und Beheben von Fehlern. Lediglich bei Aufgabe 5 sind bei zwei Versuchspersonen in Zusammenhang mit der Definition von Parameterwerten Schwierigkeiten feststellbar. Dies untermauert die Beobachtungen aus den ersten beiden Versuchsabschnitten, wo an dieser Stelle ebenfalls Probleme aufgetreten sind.

Daneben müssen sich die Versuchspersonen während der Fehlersuche, genau wie bei der Dialogmodifikation, in einem von einer dritten Person erstellten Dialog zurechtfinden. Die dabei erreichten guten Bearbeitungsergebnisse zeigen die Eignung von EmMI für eine sinnvolle Unterstützung der Zusammenarbeit im Team.

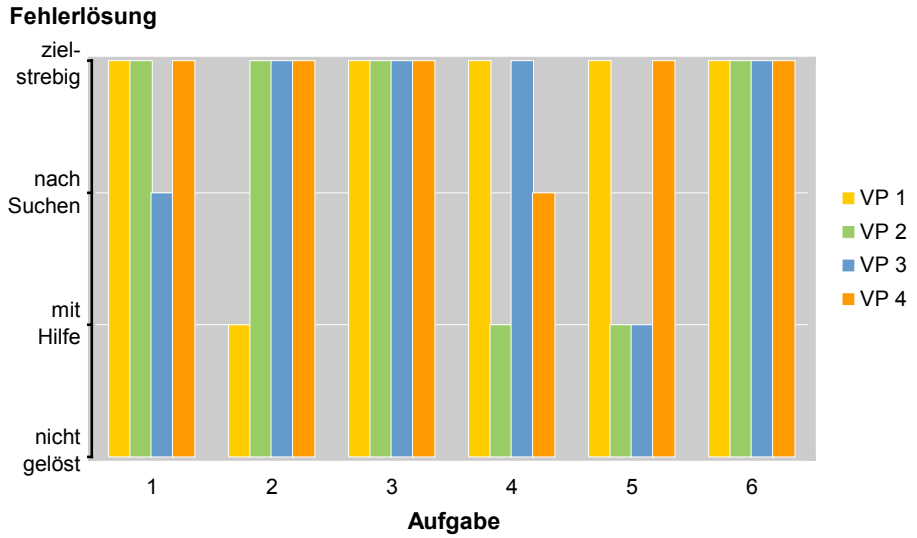


Bild 65 Bearbeitungserfolg bei der Fehlersuche[1]

5.3.5 Fragebogen

Im letzten Teil des Nutzertests werden die subjektiven Eindrücke der Versuchspersonen mit Hilfe eines Fragebogens abgefragt. Es wird dabei auf die Gebrauchstauglichkeit und Verständlichkeit von EmMI eingegangen, wobei insbesondere die implementierten Konzepte und Methoden im Zentrum der Betrachtung stehen. Diese umfassen die Modellierung des Bediendialogs und der Modalitäten sowie die Maßnahmen zur Sicherung der Dialogqualität.

5.3.5.1 Gebrauchstauglichkeit

Im ersten Fragenteil beurteilen die Versuchspersonen die Gebrauchstauglichkeit des Werkzeugs nach dem Schulnotenprinzip. Bild 66 zeigt, dass alle Fragen mit einem Median[2] von zwei bewertet werden. Selbst unter Berücksichtigung der Tatsache, dass viele Versuchspersonen tendenziell etwas positiver bewerten, wird die Gebrauchstauglichkeit eindeutig als gut eingestuft.

---

[1] Die Bewertung erfolgt analog zu den Aufgaben in den beiden vorhergehenden Versuchsabschnitten anhand eines strukturierten Protokolls durch den Versuchsleiter.

[2] Der Median gibt den in der Mitte einer Zahlenreihe liegenden Wert an. Das heißt, die eine Hälfte der Zahlen hat Werte, die kleiner sind als der Median, und die andere Hälfte hat Werte, die größer sind als der Median. Bei kleinen Stichproben wirken sich dabei einzelne Extrembewertungen nicht so stark auf das Gesamtergebnis aus, wie bei der Verwendung des Mittelwerts. (Beispiel: $\text{Median}(1 \quad 2 \quad 2 \quad 2 \quad 6) = 2$)

Eine sehr gute Bewertung des Werkzeugs wird dabei durch die im Laufe der Untersuchung zu Tage getretenen Problemstellen (5.3.2) verhindert. Wie bereits verdeutlicht wurde, handelt es sich dabei nicht um konzeptuelle Schwächen, sondern vielmehr um Probleme im Bereich der Gestaltung des Dialog Modeler. Hierzu gehören beispielsweise nicht eindeutige Begriffsverwendungen oder verbesserungswürdige Dialogfelder. Diese Mängel können mit geringem Aufwand beseitigt werden.

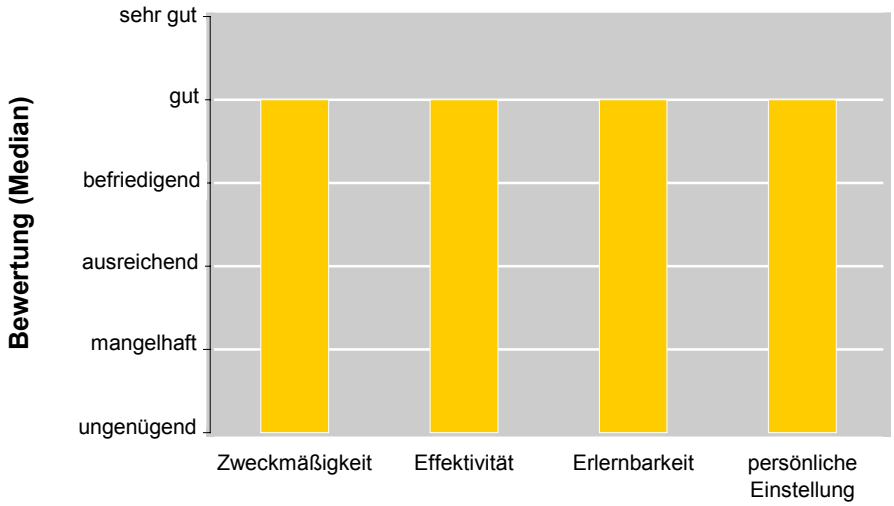


Bild 66 Subjektive Bewertung der Gebrauchstauglichkeit durch alle Versuchspersonen

5.3.5.2 Modellierung des Bediendialogs

Einen wichtigen Teil des Fragebogens nimmt die Bewertung der Modellierung des Bediendialogs ein. Dabei wird sowohl auf die formale Repräsentation als auch auf die Realisierung in EmMI eingegangen. Die Ergebnisse lassen sich folgendermaßen zusammenfassen:

- Formale Repräsentation
  Die Verständlichkeit der formalen Dialogrepräsentation, also der verschiedenen semantischen Elemente wie beispielsweise Zustände, Reaktionen und Transitionen, wird zwischen gut und sehr gut bewertet. Lediglich bei der Erlernbarkeit gibt es leichte Unterschiede. Die Erlernbarkeit der Zustände und der Reaktionen wird als sehr gut eingestuft. Die Parameter und Transitionen erhalten hier jeweils nur ein Gut.

- Realisierung in EmMI
  Der Dialog Modeler (4.2.2), also das Modul zur Dialogspezifikation, wird im Median mit einer Zwei bewertet, wobei teilweise eine Tendenz in Richtung drei vorhanden ist. Dabei scheint insbesondere der State Specifier hinsichtlich Layout und Bedienungsfreundlichkeit noch verbesserungswürdig zu sein. Die graphische Darstellung des Dialogs im State Tree wird als gut verständlich bewertet.

Diese subjektiven Beurteilungen durch die Versuchspersonen stimmen sehr gut mit den beobachteten Ergebnissen überein. Insbesondere der Optimierungsbedarf am GUI, wie beispielsweise bei der Festlegung von Parameterwerten (5.3.2.2), kann auch bei der Auswertung des Versuchprotokolls festgestellt werden.

5.3.5.3 Kontextfreie Modellierung der Modalitäten

Im Bezug auf die Modalitäten bzw. Dialogkomponenten wird deren kontextfreie Repräsentation als sehr positiv bewertet. Als wichtige darauf basierende Funktion schätzen die Experten dabei die Möglichkeit zur zentralen Konfiguration der Dialogkomponenten als sehr sinnvoll ein. Gleichzeitig empfinden sie auch die entsprechende Realisierung in EmMI als sehr verständlich. Dieses Ergebnis deckt sich mit den Beobachtungsprotokollen, da im Versuch kaum Verständnisschwierigkeiten im Umgang mit Dialogkomponenten aufgetreten sind (Bild 63).

5.3.5.4 Dialogqualitätssicherung

Im Nutzertest werden die Versuchspersonen sehr stark von der Versuchsanleitung geführt. Da deswegen die Form und der Inhalt des Bediendialogs bereits festgelegt sind, treten die Vorteile der Maßnahmen zur Sicherung der Dialogqualität nicht im vollen Umfang zu Tage. Sie können dadurch keiner fundierten objektiven Beurteilung unterzogen werden. Aus diesem Grund basiert die Bewertung der Funktionen zur Dialogqualitätssicherung hauptsächlich auf den Einschätzungen der Versuchspersonen. Diese Vorgehensweise ist legitim, da die Versuchspersonen aufgrund ihres Expertenstatus zu einem fundierten Urteil fähig sind.

Die verschiedenen Funktionen werden folgendermaßen bewertet:

- Hierarchische Dialogstrukturierung
  Der erste Bewertungspunkt ist hierbei die Sicherung der Dialogkonsistenz durch die hierarchische Strukturierung des Dialogs. Dies umfasst die Definition von Kontexten, die Zuordnung verschiedener Gültigkeitsbereiche (3.2.7) für Transitionen, Reaktionen und Parameterwerte und die Handhabung von Mehrfachdefinitionen (3.2.9). Diese Maßnahmen werden mit einem Median von eins durchwegs als sehr positiv bewertet. Dies trifft uneingeschränkt für die Zweckmäßigkeit zu. Die Verständlichkeit der Funktionsweise und die Kompliziertheit der Nutzung wird von jeweils einer Versuchsperson mit einer Drei bewertet. Hier scheinen manche Versuchspersonen noch Verbesserungsbedarf zu sehen, was sich auch an der Fehlertaxonomie (5.3.2.2) und den Problemen bei den Aufgabenschritten (5.3.2.1) ablesen lässt.

- Schlüsselwörter
  Der Einsatz der Schlüsselwörter (3.2.1) zur konsistenten Konfiguration (4.2.2.2) der Dialogkomponenten ist mit einem Median von 1,5 allen Versuchpersonen sehr verständlich. Allerdings erscheint ihnen der Nutzen nicht ganz offensichtlich. Ein möglicher Grund könnte sein, dass die Schlüsselwörter nicht genügend oft vorkommen und daher nicht vollständig gelernt werden. Eventuell wird auch im Dialog Modeler der Bezug zwischen Schlüsselwörtern, Dialogzustand und der Defini-

tion von Parameterwerten nicht genügend deutlich. Es ist aber auch denkbar, dass das Beispielprojekt nicht groß genug ist, um den mit Schlüsselwörtern verbundenen Nutzen sichtbar zu machen.

- Dialogverifikation
  Die Maßnahmen zur Sicherstellung von vollständigen und eindeutigen Dialogbeschreibungen (3.3.1) werden von allen Versuchspersonen als sehr sinnvoll und als sehr verständlich eingestuft. Lediglich die graphischen Kennzeichnungen von Zuständen, die nicht erreicht oder verlassen werden können, sollten etwas markanter gestaltet werden.

- Distance Counter
  Der Distance Counter (3.3.1.3) ist nicht Bestandteil des Versuchs, da die bearbeiteten Projekte zu klein sind, um die Einsatzmöglichkeiten sinnvoll darzustellen. Deshalb wird im Fragebogen die Funktionsweise lediglich erklärt und basierend darauf von den Versuchpersonen beurteilt. Die relativ breite Streuung bei der Bewertung ist aus diesem Grund nicht überraschend. Der Distance Counter wird tendenziell als sinnvoll eingestuft. Einsatzfelder sehen die Versuchspersonen als Instrument zur Fehlersuche und bei der Bewertung verschiedener HMI-Ansätze. Die *Distance* könnte ihrer Meinung nach durchaus als Diskussionsgrundlage für Konzeptentscheidungen benutzt werden (vgl. 3.3.3.1).

5.4 Mögliche Erweiterungen

Der Versuch hat sich auch als Ideenquelle für mögliche sinnvolle Erweiterungen des Werkzeugs erwiesen. Es handelt sich dabei durchwegs um funktionale Erweiterungen, die auf Basis der bestehenden Konzepte realisiert werden können:

- Dokumentorfunktion, um aus der Dialogdefinition eine Beschreibung des Dialogs, beispielsweise in Form eines HTML-Dokuments, zu generieren. Dieses könnte als Grundlage für die Spezifikation des Serienprodukts verwendet werden. Dieser Prozess könnte basierend auf der formalen Dialogrepräsentation sehr einfach automatisiert werden.

- Zusätzliche Kontroll- und Bearbeitungsfunktionen im Dialog Modeler für den Entwickler. Diese könnten beispielsweise einen Gesamtüberblick über alle Parameterwerte bei der Kommandofestlegung ermöglichen. Es ist aber auch eine Suchfunktion denkbar, die den Entwickler beim Auffinden von Reaktionen bzw. Transitionen unterstützt, die eine bestimmte Nachricht oder ein bestimmtes Kommando beinhalten.

- Auf Basis des Distance Counters sind weitere Funktionen denkbar, die beispielsweise auf zu lange Interaktionssequenzen automatisch hinweisen oder einen Index für die mittlere Interaktionsdauer bestimmen. Dadurch könnten insbesondere den Dialoginhalt (3.3.3) betreffende Maßnahmen zur Sicherung der Dialogqualität verfeinert werden.
- Vergrößerung der Funktionalität des Dialogmanagers. In bestimmten Situationen während der Dialogspezifikation würde sich eine Möglichkeit zur Aktualisierung der Konfigurationsparameterwerte und der Parametrisierung von Nachrichten unabhängig von einem Zustandswechsel als sehr nützlich erweisen. Dies könnte als zusätzliches Kommando des Dialogmanagers realisiert werden. Daneben würde ein Statusparameter mit dem Namen des jeweils aktuellen Dialogzustandes die Flexibilität der Dialogbeschreibung deutlich erhöhen.

5.5 Fazit

Grundsätzlich ergeben die Auswertungen der Fragebögen und der Beobachtungsprotokolle sehr gut übereinstimmende Ergebnisse. Die Versuchspersonen sind also in der Lage, ihre Leistung während des Versuchs adäquat zu bewerten. Für das Versuchsdesign selbst lässt sich daraus auf eine sinnvolle Wahl der Evaluationsmethode und der Messdaten schließen. Diese Vorgehensweise ist daher auch für zukünftige Bewertungen ähnlicher Entwicklungswerkzeuge empfehlenswert.

Die Ergebnisse des Versuchs machen deutlich, dass die bereitgestellten Konzepte und Methoden zur Erstellung multimodaler Dialoge tauglich und verständlich sind. Nach nur geringer Einarbeitungszeit mit dem Tutorial können die Versuchspersonen Dialoge erstellen, bestehende Projekte modifizieren, Fehler auffinden und korrigieren. Dies ist möglich, da der Entwickler aufgrund des eng definierten Anwendungsbereichs und der strukturierten Herangehensweise des Werkzeugs optimal durch den Spezifikationsprozess geführt und dadurch vor einer Vielzahl konzeptueller Fehler bewahrt wird. Die Repräsentation der Dialoge mit Hilfe von Zuständen, Reaktionen und Transitionen ist in Verbindung mit den verschiedenen Gültigkeitsbereichen als sehr sinnvoll einzustufen. Die hierarchische Dialogstrukturierung stellt hierbei ein sehr wichtiges und mächtiges Instrument dar, dessen adäquate Nutzung für den Entwickler jedoch einen gewissen Lernaufwand erfordert.

Die kontextfreie Modellierung und die zentrale Konfiguration der Modalitäten sind für die Versuchspersonen leicht verständlich und sinnvoll. Aufgrund der dadurch erreichbaren Flexibilität bei der Verwendung und der erhöhten Konsistenz im Verhalten verschiedener Dialogkomponenten stellen sie vor allem in der Konzeptphase wichtige Bestandteile entsprechender Dialogentwicklungswerkzeuge dar.

Die meisten der aufgetretenen Fehler sind Ausführungsfehler. Dies ist in der Lernphase zu erwarten und daher nicht weiter überraschend. Ein großer Teil dieser Problemstellen ist dabei auf suboptimale Realisierungen des GUI zurückzuführen. Das Auffinden dieser in erster Linie graphischen Schwächen ist zwar nicht Kernziel des Versuchs, da sie aber sehr eng mit der Be-

nutzbarkeit – vor allem beim Erstkontakt – zusammenhängen, treten sie praktisch automatisch zu Tage. Sie können im Allgemeinen mit geringem Aufwand behoben werden.

Die Methoden zur Sicherung der Dialogqualität sind im Großen und Ganzen als positiv einzustufen. Die Maßnahmen zur Dialogverifikation und die Möglichkeit zur hierarchischen Strukturierung des Dialogs werden besser bewertet als die Schlüsselwörter und der Distance Counter. Dies ist mitunter auf den kleinen Projektumfang und den damit verbundenen geringen Einsatzmöglichkeiten dieser Funktionen zurückzuführen.

Die angeregten Erweiterungen des Werkzeugs betreffen keine konzeptuellen Aspekte, sondern sind ergänzende Funktionen, die basierend auf den bestehenden Konzepten realisiert werden können. Das vorgeschlagene theoretische Fundament scheint daher die zur Erstellung multimodaler Dialoge notwendigen Anforderungen zu berücksichtigen. Es stellt somit eine sehr gute Basis für die Entwicklung zukünftiger Werkzeuge dar.

Kapitel 6 Diskussion und Ausblick

Abschließend erfolgt eine zusammenfassende Darstellung der vorliegenden Arbeit. Hierzu werden ausgehend von der Motivation und der Zielsetzung die Vorgehensweise und die Ergebnisse im Überblick beschrieben. Dabei sind sowohl die disziplinübergreifenden als auch die ingenieurwissenschaftlichen Aspekte dieser Arbeit deutlich erkennbar. Den Schluss bildet ein Ausblick über die nächsten Schritte, sinnvolle Erweiterungen und mögliche Anwendungen.

6.1 Diskussion

Ausgangssituation ist die beständige Zunahme im Fahrzeug verfügbarer Funktionen. Dieser Anstieg resultiert aus den vielfältigen TICS und Komfortfunktionen, die dem Fahrer die Fahrzeugführung leichter und angenehmer gestalten sowie zusätzliche Kommunikations- und Unterhaltungsmöglichkeiten bieten. In diesem Zusammenhang besitzen multimodale Bedienkonzepte großes Potenzial, um diese Funktionen auch in Zukunft weiterhin verkehrssicher und verständlich bedienbar zu machen. Allerdings fehlen hierzu bislang geeignete Entwicklungswerkzeuge, mit denen solche Bedienkonzepte bereits in der Konzeptphase erlebbar und beurteilbar gemacht werden können.

In dieser Arbeit werden daher die theoretischen Grundlagen zur Entwicklung von Rapid-Prototyping-Werkzeugen für multimodale Bedienkonzepte im Fahrzeug untersucht und formal beschrieben. Dabei stellt die Einbeziehung der ergonomischen Randbedingungen im Fahrzeug einen wesentlichen Aspekt der Untersuchung dar.

Die Arbeit gliedert sich hierzu in vier Bereiche. Basierend auf einer Analyse der Rahmenbedingungen in der Domäne Fahrzeug erfolgt zunächst eine Ableitung der Werkzeuganforderungen in der Konzeptphase. Diese Rahmenbedingungen werden dann abstrahiert und in eine generelle und formale Beschreibung des Bediendialogs überführt. Im nächsten Schritt folgt eine Referenzimplementierung des theoretischen Ansatzes. Schließlich werden der formale Ansatz und die Implementierung in einem Nutzertest mit Experten evaluiert.

Die Untersuchung der Rahmenbedingungen zeigt, dass während der Konzeptphase große Flexibilität bezüglich der verwendeten Modalitäten und der Definition des Dialogverhaltens notwendig ist. Zur Spezifikation des Bediendialogs sollten daher alle Modalitäten und der Dialogablauf selbst über eine gemeinsame und einheitliche Schnittstelle konfigurierbar bzw. definierbar sein. Dabei ist es zur Sicherstellung der verkehrstauglichen Bedienbarkeit notwendig, geeignete Maßnahmen möglichst früh im Entwicklungsprozess zu treffen. Hierzu müssen entsprechende

Richtlinien beachtet werden, die Prinzipien zur Dialogqualität im Fahrzeug beschreiben [ISO/DIS15005 2000] [2000/53/EC 2000]. Für eine fundierte Bewertung des Konzepts ist es dabei notwendig, möglichst früh ein erlebbares System zu erhalten. Daneben ist zu berücksichtigen, dass sich der Gebrauch der verschiedenen Modalitäten im Fahrzeug deutlich von multimodalen Systemen in anderen Domänen unterscheidet. Aufgrund der kognitiven Belastung durch die Fahraufgabe und des besonderen Aufgabentyps ist im Fahrzeug lediglich eine alternierend multimodale Bedienung vorzufinden. Der hohe technische Aufwand zur Realisierung synergistisch multimodaler Systeme ist daher nicht erforderlich.

Im nächsten Schritt werden diesen Rahmenbedingungen generalisiert und formalisiert. Dies umfasst folgende Gesichtspunkte:

- Kontextfreie Modellierung der Modalitäten als Dialogkomponenten
  Zur Beschreibung des Funktionsumfangs der Modalitäten steht ein einheitliches Set an semantischen Mitteln bereit. Damit können beliebige Modalitäten unabhängig von ihrem speziellen Funktionsumfang als Dialogkomponenten dargestellt werden. Dies ermöglicht eine flexible Integration neuer und Wiederverwendung bestehender Modalitäten, ohne dass Modifikationen am Entwicklungswerkzeug selbst vorgenommen werden müssen. Gleichzeitig können damit beliebige Modalitäten unter einer einheitlichen Bedienoberfläche verwaltet und konfiguriert werden.

- Modellierung des Bediendialogs als hierarchischer Zustandsautomat
  Der Bediendialog, also die informationstechnische Repräsentation der Interaktion zwischen Mensch und Maschine, wird als erweiterter hierarchischer Zustandsautomat beschrieben. Hierzu muss der Dialog in einzelne Zustände unterteilt werden. Diese sind durch Transitionen miteinander verbunden. Reaktionen legen das funktionale Verhalten innerhalb eines Zustandes fest. Die Möglichkeit der flexiblen Zuordnung der einzelnen Zustände zu verschiedenen Kontexten stellt dabei eine explizit an die Anforderungen der Dialogspezifikation angepasste Erweiterung klassischer hierarchischer Zustandsautomaten dar. Dadurch wird die Zahl der zu definierenden Transitionen und Reaktionen gering gehalten, womit ein geringerer Komplexitätsanstieg der Dialogbeschreibung bei zunehmender Dialoggröße verbunden ist.

- Maßnahmen zur Sicherung der Dialogqualität
  Basierend auf der formalen Modellierung der Modalitäten und des Bediendialogs werden Maßnahmen zur Sicherung der Dialogqualität beschrieben, die bereits während des Spezifikationsprozesses eingesetzt werden können. Ziel ist es die Einhaltung der erforderlichen Dialogmerkmale so früh wie möglich systematisch zu unterstützen. Auf Grundlage des mathematischen Modells können dabei bestimmte Aspekte sogar analytisch kontrolliert werden. An diesen Stellen sind deshalb keine zeitaufwändigen empirischen Überprüfungen notwendig. Folgende Gesichtspunkte werden adressiert:

  - Dialogverifikation
    Durch eine entsprechende Kennzeichnung der Zustände im Entwicklungswerkzeug wird der Entwickler auf unvollständige und uneindeutige Stellen in der Dialogbeschreibung aufmerksam gemacht. Mit dem Distance Counter steht ein Werkzeug zur Überprüfung der Erreichbarkeit und Berechnung der Entfernung zweier Zustände zur Verfügung.

  - Dialogkonsistenz
    Bei multimodalen Systemen hat Dialogkonsistenz zwei verschiedene Ausprägungen. Hierbei ermöglicht zum einen die hierarchische Modellierung des Dialogs die Sicherstellung identischer Bedienmöglichkeiten in verschiedenen Zuständen. Zum anderen erleichtert die zentrale Konfigurierbarkeit der Dialogkomponenten im Dialog Modeler die konsistente Parametrisierung der Modalitäten.

  - Dialoginhalt
    Um die allgemeine Verwendbarkeit des Werkzeugs nicht zu gefährden, sind Annahmen über den Inhalt eines speziellen Dialogs unzulässig. Daher können inhaltliche Qualitätsaspekte nicht direkt adressiert werden. Durch eine sinnvolle Verwendung der verfügbaren formalen Mittel und Methoden ist es aber trotzdem möglich inhaltliche Gesichtspunkte wie die Interaktionslänge, die Einfachheit und Verständlichkeit der Ausgaben sowie die Kompatibilität mit der Fahraufgabe zu berücksichtigen.

- Formale Beschreibung der Dialogausführung
  Um die Dialogspezifikationen erlebbar machen zu können, wird unter zu Hilfenahme der formalen Modellierung der Modalitäten und des Bediendialogs eine Heuristik zur Dialogausführung entwickelt. Diese ist universell für beliebige Bediendialoge und Dialogkomponenten einsetzbar.

Zur Demonstration der Realisierbarkeit und Tauglichkeit des theoretischen Gerüsts wird darauf basierend EmMI (Environment for multimodal Human-Machine-Interfaces) als Referenz- und Experimentiersystem implementiert. EmMI ist im wesentlichen durch die drei Hauptbereiche gekennzeichnet:

- Formale Repräsentation
  Die formale Repräsentation der Modalitäten und des Dialogs wird in einer relationalen SQL-Datenbank abgebildet.

- Erstellung
  Der Dialogcomponent Manager und der Dialog Modeler ermöglichen eine bequeme Verwaltung der Dialogkomponenten sowie die komfortable Erstellung des Bediendialogs. Sie kapseln damit die formale Repräsentation in der Datenbank und bereiten diese für den Entwickler graphisch auf. Gleichzeitig sind im Dialog Modeler die verschiedenen Maßnahmen zur Sicherung der Dialogqualität realisiert.

- Ausführung
  Um eine fundierte Bewertung eines Bedienkonzepts zu ermöglichen, muss es schließlich erlebbar gemacht werden. Hierzu interpretiert der Dialogmanager basierend auf einer Heuristik zur Dialogausführung die Definition des Dialogs. Er kontrolliert und synchronisiert dabei die Dialogkomponenten und steuert den Dialogablauf entsprechend dieser Definition. Der Dialogmanager ist unabhängig von einem speziellen Bediendialog und kann mit beliebigen Dialogkomponenten umgehen. Die funktionale Realisierung der Modalitäten und damit der eigentlichen Informationsaustausch zwischen Mensch und Maschine erfolgt über die Dialogkomponenten.

Schließlich werden die bereitgestellten Konzepte und Methoden hinsichtlich ihrer Tauglichkeit und Effizienz untersucht. Dazu erfolgt eine Bewertung des formalen Ansatzes und der Referenzimplementierung EmMI in einem Nutzertest. Angelehnt an die Methoden der Heuristischen Evaluation spezifizieren hierzu vier Experten auf dem Gebiet der HMI-Entwicklung mit EmMI einen Bediendialog. Angeleitet durch ein Tutorial müssen sie verschiedene Aufgaben bearbeiten, die an den Problemstellungen des realen Entwicklungsprozesses orientiert sind. Die Bewertung stützt sich dabei sowohl auf systematische Beobachtungen des Versuchsverlaufs, als auch auf subjektive Beurteilungen der Experten. Das Experiment zeigt dabei grundsätzlich ein sehr positives Ergebnis.

- Die Konzepte und Methoden sind für die Experten sehr klar. Fehler treten hauptsächlich wegen Ausführungsproblemen auf. Die Versuchspersonen wissen also, was zu tun ist, sie haben lediglich Schwierigkeiten bei der Umsetzung.

- Die formale Dialogrepräsentation wird begrüßt. Zur effizienten Verwendung der hierarchischen Strukturierungsmöglichkeiten ist allerdings ein gewisser Lernaufwand erforderlich.

- Die kontextfreie Repräsentation der Modalitäten und die darauf basierende Möglichkeit zur zentralen Konfiguration der Dialogkomponenten sind für die Experten leicht verständlich.
- Die Maßnahmen zur Sicherung der Dialogqualität werden positiv beurteilt und als sehr sinnvoll erachtet.

Ausgehend von einer Analyse der Anforderungen liefert die vorliegende Arbeit somit einen Beitrag zur Entwicklung des theoretischen Fundaments für Werkzeuge, mit denen fahrzeugtaugliche multimodale Bedienkonzepte prototypisch realisiert werden können. Wesentliche Kennzeichen sind dabei die kontextfreie und damit allgemein anwendbare Modellierung der Modalitäten sowie die an die Problemstellung angepasste formale Repräsentation des Bediendialogs. Basierend darauf werden Maßnahmen beschrieben, um bereits während des Spezifikationsprozesses die Benutzbarkeit des Bedienkonzepts während der Fahrt zu sichern. Des Weiteren erfolgt die Entwicklung einer universell einsetzbaren Heuristik zur Ausführung des Dialogs. Im Sinne einer ingenieurswissenschaftlichen Arbeit wird dann die Umsetzbarkeit und Tauglichkeit des theoretischen Ansatzes gezeigt. Hierzu wird EmMI, eine auf diesem formalen Gerüst beruhende Referenzimplementierung, realisiert. Die Bestätigung der Eignung des Konzepts erfolgt schließlich in Form eines Nutzertests.

6.2 Ausblick

Im Sinne einer ingenieurswissenschaftlichen Arbeit stellt die Einbringung dieser Ergebnisse in den Entwicklungsprozess einen wichtigen nächsten Schritt dar. Gleichzeitig dürfen aber auch notwendige weitergehende wissenschaftliche Untersuchungen sowie die Beobachtung von eventuellen Einflüssen aus anderen Bereichen nicht vernachlässigt werden.

Für eine sinnvolle Einflussnahme auf den Fahrzeugentwicklungsprozess sind dabei folgende Strategien denkbar:

- Anpassung von EmMI für den Einsatz im Entwicklungsprozess
  Um EmMI im Entwicklungsprozess einsetzen zu können, müssten eine Reihe von Erweiterungen und Verbesserungen vorgenommen werden. Diese betreffen vor allem die Optimierung der Performance, die Erleichterung der Installation und zusätzliche Funktionen zur Projektverwaltung. Allerdings ist diese Variante aufgrund des entstehenden Support- und Wartungsaufwands nicht zu bevorzugen.
- EmMI als Schulungswerkzeug
  Da eine Verwendung von EmMI in der bestehenden Form im Entwicklungsprozess nicht anzuraten ist, könnte es in einem ersten Schritt als Schulungswerkzeug für Entwickler und Einsteiger in die Thematik genutzt werden. Wegen der Spezialisierung auf den Anwendungsbereich stellt es ein ausgezeichnetes Experimentierfeld dar und verspricht somit gute Lerneffekte. Daneben bietet es angesichts der sehr formalen Herangehensweise eine gute Anleitung zur strukturierten Entwicklung von Bediendialogen. Dies ist insbesondere aufgrund des interdis-

ziplinären Charakters der HMI-Entwicklung und der daraus resultierenden Schnittstellen zwischen den verschiedenen Aufgabenbereichen ein sehr wichtiger Aspekt. Durch die saubere und systematische Darstellung der Zusammenhänge wäre es möglich, die Problematik bei allen beteiligten Fachstellen auf einem gemeinsamen Niveau zu thematisieren.

- Einbindung der Konzepte in bestehende Entwicklungsmethoden
  Ein sehr sinnvoller Ansatz stellt die Betrachtung von EmMI als „Proof-of-Concept" des in dieser Arbeit entwickelten Konzepts dar. Dieses muss im nächsten Schritt in bestehende Entwicklungsmethoden integriert werden. Hierzu sind zwei Strategien vorstellbar:

  - Aufgrund des ähnlichen Beschreibungskonzepts bieten sich auf Zustandsautomaten basierende CASE-Tools an. Die Einbindung ist in Form einer speziell angepassten Bedienoberfläche oder eines Wizards[1] denkbar. Ähnlich wie im Dialog Modeler würde der Entwickler damit bei der Dialogspezifikation unterstützt werden. Auf diese Weise könnte die aufgabenangepasste Nutzerführung optimal mit der semantischen Mächtigkeit dieser Werkzeuge kombiniert werden. Gleichzeitig wäre Support, Wartung und Weiterentwicklung des Werkzeugs durch den Hersteller sichergestellt. Allerdings entsteht bei diesem Ansatz eine große Abhängigkeit von einem speziellen Werkzeug.
  - Umsetzung in eine hersteller- und werkzeugunabhängige Beschreibungssprache. Mit der Darstellung des formalen Modells in einer Datenbankstruktur sind bereits die wichtigsten Vorarbeiten zur Entwicklung eines eigenständigen XML-Dialekts geleistet. Ähnlich wie bei VoiceXML wäre die Etablierung einer speziellen Beschreibungssprache denkbar, auf die beliebige Entwicklungswerkzeuge aufsetzen könnten. Dadurch wäre gleichzeitig eine größere Durchgängigkeit im Entwicklungsprozess erreichbar, da Brüche effizient vermieden werden könnten, die aufgrund der unterschiedlichen Beschreibungsmethoden der in den verschiedenen Phasen benutzten Werkzeuge entstehen.

[1] Wizard: Assistenzfunktion, die den Benutzer Schritt für Schritt durch den Bearbeitungsprozess führt.

Neben diesen sehr pragmatischen Vorstellungen dürfen allerdings mögliche Weiterentwicklungen auf formaler Ebene nicht aus den Augen verloren werden. Das Potenzial des theoretischen Gerüsts ist hier bei weitem noch nicht ausgeschöpft und bietet Raum für eine Vielzahl von Erweiterungen. Diese bedürfen jedoch noch einer fundierten wissenschaftlichen Untersuchung.

- Dokumentationsfunktion als Spezifikationsgrundlage für Serienentwicklung
  Mit Hilfe einer Dokumentationsfunktion könnten Bedienkonzepte, die in der frühen Phase des Fahrzeugentwicklungsprozesses entstandenen sind, sehr einfach als Grundlage zur Spezifikation des Serienprodukts benutzt werden. Ein möglicher Weg hierzu wäre die automatische Generierung eines entsprechenden HTML-Dokuments oder einer UML-Beschreibung aus der formalen Darstellung des Dialogs.

- Optimierung der Dialogdarstellung für den Entwickler
  Zur graphischen Visualisierung des Bediendialogs zeigt EmMI mit dem Dialog Modeler eine auf die Bedürfnisse des Entwicklers zugeschnittene Lösungsmöglichkeit. Dieser Bereich bietet aber auch Optimierungspotenzial, wie die Ergebnisse der Evaluation zeigen. Hierbei müssen vor allem effiziente Darstellungsformen gefunden werden, mit denen sowohl hierarchische als auch netzförmige Dialogstrukturen einfach und übersichtlich dargestellt werden können. Diese Darstellungsmethoden dürfen dabei selbst von umfangreicheren Dialogspezifikationen nicht an ihre Leistungsgrenzen gebracht werden.

- Kombination mit weiterführenden Bewertungsmethoden
  Für eine stärkere Einbindung in den Bewertungsprozess ist eine Verknüpfung der Beschreibung des Bediendialogs mit ausführlicheren Bewertungsmethoden notwendig. Diese beispielsweise auf Benutzermodellen oder Prüflisten beruhenden Verfahren benötigen allerdings eine vollständige Spezifikation des Dialogs, sodass sie erst in einem zweiten Schritt nach dem Erstellungsprozess durchgeführt werden können (2.4). Dabei stellt die streng formale Modellierung des Dialogs eine hervorragende Grundlage dar, um daraus in einem voll- oder semiautomatischen Prozess die Eingangsmodelle für diese Bewertungsverfahren abzuleiten.

- Verfeinerung der Maßnahmen zur Sicherung der Dialogqualität
  Aufsetzend auf der Funktionalität des Distance Counters ist eine Reihe weiterer Funktionen zur Analyse des Dialogs denkbar. Hierzu gehört beispielsweise die Berechnung des mittleren Abstands aller Zustände, um daraus eine Abschätzung für die „mittlere Interaktionslänge“ des Dialogs zu erhalten. Hierbei könnten gleichzeitig die am weitesten entfernten Zustände des Dialogs ermittelt werden. Durch die Festlegung einer maximalen Entfernung für zwei Zustände wäre es möglich, mit einer ähnlichen Funktion den Entwickler auf Zustandspaare aufmerksam zu machen, die diese Grenze überschreiten. Die automatische Ermittlung des vom Selected State maximal entfernten Zustandes könnte sich bei der Dialogspezifikation ebenfalls als sehr brauchbar erweisen.

Daneben müsste eingehend untersucht werden, inwieweit aus der in den einzelnen Zuständen möglichen Zahl an Transitionen und Reaktionen eine Aussage zur Bedienkomplexität des Gesamtdialogs abgeleitet werden kann.

Möglicherweise können in diesem Zusammenhang auch formale Kriterien zur Erfassung der Erlernbarkeit von Bediendialogen abgeleitet werden. Mit Hilfe solcher objektiver Kennzahlen könnten Optimierungsarbeiten am Dialog deutlich vereinfacht werden. Hierzu ist allerdings noch eine Vielzahl grundsätzlicher Untersuchungen erforderlich.

Gleichzeitig müssen aber auch mögliche Veränderungen im Bedienverhalten der Nutzer in nicht fahrzeugspezifischen Bereichen wie beispielsweise PC, Home Entertainment oder Mobile Devices beobachtet werden. Eine synergistische Verwendung der Modalitäten (2.2.4) könnte dort, wie es derzeit in verschiedenen Forschungsprojekten [Wahlster*, et al.* 2001] [Herfet & Kirste 2001] untersucht wird, zur Ausprägung neuer Bedienmetaphern und Bedienstereotypen führen. Aufgrund der ansteigenden Vernetzung mit diesen Bereichen ist es möglich, dass eine solche Entwicklung im Fahrzeug nicht unberücksichtigt bleiben kann. Hierzu werden im Rahmen des Kooperationsprojekts FERMUS[1] zwischen der BMW Group, der DaimlerChrysler AG, der Siemens VDO Automotive AG und dem Lehrstuhl für Mensch-Maschine-Kommunikation der Technischen Universität München bereits konkrete Untersuchungen durchgeführt [Althoff*, et al.* 2002] [McGlaun*, et al.* 2002]. Die Ergebnisse dieser Forschungsaktivitäten könnten eine Erweiterung des in der vorliegenden Arbeit entwickelten theoretischen Gerüsts hinsichtlich synergistischer Multimodalität notwendig machen. Dabei müsste aber nach wie vor ein sehr großes Augenmerk auf die Methoden zur Sicherstellung einer geringen kognitiven Belastung des Fahrers gelegt werden.

[1] Fehlerrobuste Multimodale Sprachdialoge

Anhang A Abkürzungsverzeichnis

ACC	Active Cruise Control
AMODEUS	Assimilating Models of Designers, Users and Systems
API	Application Programming Interface
ASR	Automatic Speech Recognition
ATN	Augmented Transition Network
BNF	Backus-Naur-Form
CAD	Computer Aided Design
CAN	Controller Area Network
CASE	Computer Aided Software Engineering
CORBA	Common Object Request Broker Architecture
CRT	Cathode Ray Tube
CTI	Computer Telephone Integration
DDE	Dynamic Data Exchange
DLL	Dynamic Link Library
EmMI	Environment for multimodal Human-Machine-Interfaces
FAS	Fahrerassistenzsystem
FERMUS	Fehlerrobuste Multimodale Sprachdialoge
FIS	Fahrerinformationssystem
GOMS	Goals, Operators, Methods, Selection Rules
GTN	Generalized Transition Network
GUI	Graphical User Interface
HMI	Human-Machine-Interface
HMM	Hidden-Markov-Modell

HTML	Hypertext Markup Language
HUD	Head-Up-Display
ISO	International Organization for Standardization.
LCD	Liquid Crystal Display
LED	Light-Emitting Diode
MMI	Mensch-Maschine-Interaktion
MMS	Mensch-Maschine-System
MSC	Message Sequence Charts
MoTiV	Mobilität und Transport im intermodalen Verkehr
NL	Natural Language
OAA	Open Agent Architecture
OLED	Organic Light Emitting Device
PAC	Presentation, Abstraction, Control
PDA	Personal Digital Assistant
PIM	Personal Information Manager
RTN	Recursive Transition Network
SDL	Specification and Description Language
SIVIT	Siemens Virtual Touchscreen
SMS	Short Message Service
SQL	Structured Query Language
TICS	Transport Information and Control Systems
TTS	Text-to-Speech
UM	Unified Messaging
UML	Unified Modeling Language
VR	Virtual Reality
XML	Extensible Markup Language
WIMP	Windows Icons Menus Pointers
WoZ	Wizard of Oz

Anhang B Glossar

Aktionsmodalität	→Modalität zur Informationsübertragung vom Menschen zur Maschine
Bediendialog	→Dialog
Dialog	Austausch von Informationen zwischen Fahrer und System zur Erreichung eines bestimmten Zieles. Ein Dialog kann entweder vom Fahrer oder vom System initiiert werden. Ein Dialog ist eine bestimmte Abfolge von miteinander verknüpften Bedienaktionen und Informationsdarstellungen. Die Interaktion kann mehr als eine Sinnesmodalität umfassen [ISO/DIS15005 2000].
Dialogkomponente	Bezeichnung in EmMI für die funktionale Realisierung einer einzelnen →Modalität
Dialogmanagement	Systemseitige Instanz der →Mensch-Maschine-Schnittstelle zur Steuerung und Kontrolle des →Dialogs zwischen Mensch und Maschine
Endlicher Automat	[...] mathematisches Modell eines Systems mit diskreten Ein- und Ausgaben. Das System befindet sich in einem aus einer endlichen Anzahl von [...] Zuständen. Der Zustand eines Systems umfasst die Informationen, die sich aus den bisherigen Eingaben ergeben haben und die benötigt werden, um die Reaktion des Systems auf noch folgende Eingaben zu bestimmen [Hopcroft & Ullman 1990].
Erweiterter Endlicher Automat	Erweiterung →Endlicher Automaten um Speicherelemente
Fahrerassistenzsystem	Technisches System zur Unterstützung des Fahrers bei der Planungs-, Regelungs- oder Stabilisierungsaufgabe. Durch die Entschärfung fahrerspezifischer Defizite leistet es einen Beitrag zur Erhöhung des Fahrkomforts und der Fahrsicherheit.

Fahrerinformationssystem	Technisches System, das im Fahrzeug umfangreiche →Infotainmentfunktionen bereitstellt. Die Bedienung erfolgt normalerweise über ein →Integriertes Anzeige- und Bedienkonzept.
Fission	Umsetzung und Verteilung der Systemreaktion auf die zur Verfügung stehenden →Wahrnehmungsmodalitäten
Force Feedback	Bezeichnung für die aktive Übermittlung von Informationen über das taktile Interface von der Maschine zum Menschen. Hierbei werden mittels technischer Aktuatoren die Muskeln und Sehnen des Menschen zur Verdeutlichung der aktuellen Dialogsituation mit mechanischen Kräften beansprucht.
Fusion	Zusammenführung der Informationen aus den einzelnen →Aktionsmodalitäten auf ein gemeinsames semantisches Niveau
Gebrauchstauglichkeit	Grad der Effektivität, Effizienz und Zufriedenheit, mit der das zu bewertende technische System bestimmte Benutzergruppen bei der Aufgabenbewältigung unterstützt
Infotainment	Neologismus aus Information und Entertainment, Bezeichnung für kombinierte Informations-, Kommunikations- und Unterhaltungsfunktionen
Integrierte Anzeige- und Bedienung	Integration verschiedener Funktionen von →Fahrerinformations- bzw. →Fahrerassistenzsystemen in eine gemeinsame Menüstruktur, wobei Systemausgaben auf einem günstig positionierten graphikfähigen Display angezeigt werden. Die Bedienung erfolgt mit einem zentralen Bedienelement, das häufig von der Anzeigefläche örtlich getrennt ist, um eine gute Erreichbarkeit zu gewährleisten.
Mensch-Maschine-Interaktion	Wissenschaftliche Fachrichtung, die sich mit dem Entwurf, der Gestaltung, der Realisierung und der Bewertung von →Mensch-Maschine-Schnittstellen beschäftigt
Mensch-Maschine-Schnittstelle	Komponenten eines →Mensch-Maschine-Systems, die für den Informationsaustausch zwischen Mensch und Maschine relevant sind
Mensch-Maschine-System	Zusammenwirken von Mensch und Maschine mit dem Ziel eine selbstgewählte oder vorgegebene Aufgabe zu lösen
Modalität	In der →Mensch-Maschine-Schnittstelle bereitgestellte Möglichkeit zur Interaktion des Menschen mit der Maschine

Multimodalität	Mehrere →Modalitäten umfassende →Mensch-Maschine-Schnittstelle. Dabei werden die verschiedenen spezialisierten Interaktionsformen des Menschen zur Abgabe und Aufnahme von Informationen genutzt, um eine möglichst natürliche und intuitive Bedienung zu ermöglichen.
Styleguides	Gestaltungshinweise zur Unterstützung des Entwicklers bei der Einhaltung für die jeweilige Domäne und Aufgabe relevanter Aspekte während der Erstellung des →Dialogs
Transport Information and Control Systems	Sammelbegriff für →Fahrerinformationssysteme und →Fahrerassistenzsysteme: eine einzelne Funktion (z.B. Routenführung) oder eine Reihe von Funktionen, die so entwickelt wurden, dass sie in einem System zusammen arbeiten [ISO/DIS15005 2000]
Usability	→Gebrauchstauglichkeit
Usability Engineering	Prozess, in dessen Verlauf die →Gebrauchstauglichkeit eines Produktes definiert, gemessen und verbessert wird
Wahrnehmungsmodalität	→Modalität zur Informationsübertragung von der Maschine zum Menschen
Wizard of Oz-Studie	Nutzerstudie eines technisch nicht verfügbaren Systems. Das funktionale Verhalten wird durch einen Menschen (Wizard) simuliert, ohne dass dies der Versuchsperson bekannt ist.

Anhang C Nutzertest

Nachfolgend sind die Anleitungen zu den einzelnen Versuchsabschnitten der Nutzerstudie abgedruckt. Daran anschließend befindet sich der Fragebogen zur subjektiven Beurteilung von EmMI durch die Experten. Den Abschluss bildet eine detaillierte Auswertung der während des Versuchs aufgetretenen Fehlerstellen. Dazu erfolgt eine genaue Zuordnung der Bedienprobleme der einzelnen Versuchspersonen zu den entsprechenden Arbeitsschritten.

C.I Versuchsabschnitt Tutorial – Dialogerstellung

1 Dialog Modeler

Um den Dialog Modeler zu starten, klicken Sie: .\Aufgabe 1\EmMI.mdb

Dies ist die Bedienoberfläche, mit der Sie den Dialogablauf festlegen und die einzelnen Dialogkomponenten konfigurieren können.

Der Dialog Modeler besteht aus drei Bereichen (Abb. 1):

- Auf der linken Seite befinden sich die Basisfunktionen, mit denen Sie unter anderem Dialogzustände anlegen und löschen können.
- In der Mitte werden im „State Tree“ die Dialogzustände des gesamten Bediendialogs in einer Baumstruktur angezeigt.
- Auf der rechten Seite können Sie in den einzelnen Reitern des „State Specifier“ das Dialogverhalten des im State Tree ausgewählten Dialogzustandes anzeigen, eingeben und modifizieren. In den mit „Selected State“ bzw. mit „Destination“ bezeichneten Feldern werden die gerade ausgewählten Dialogzustände angezeigt.

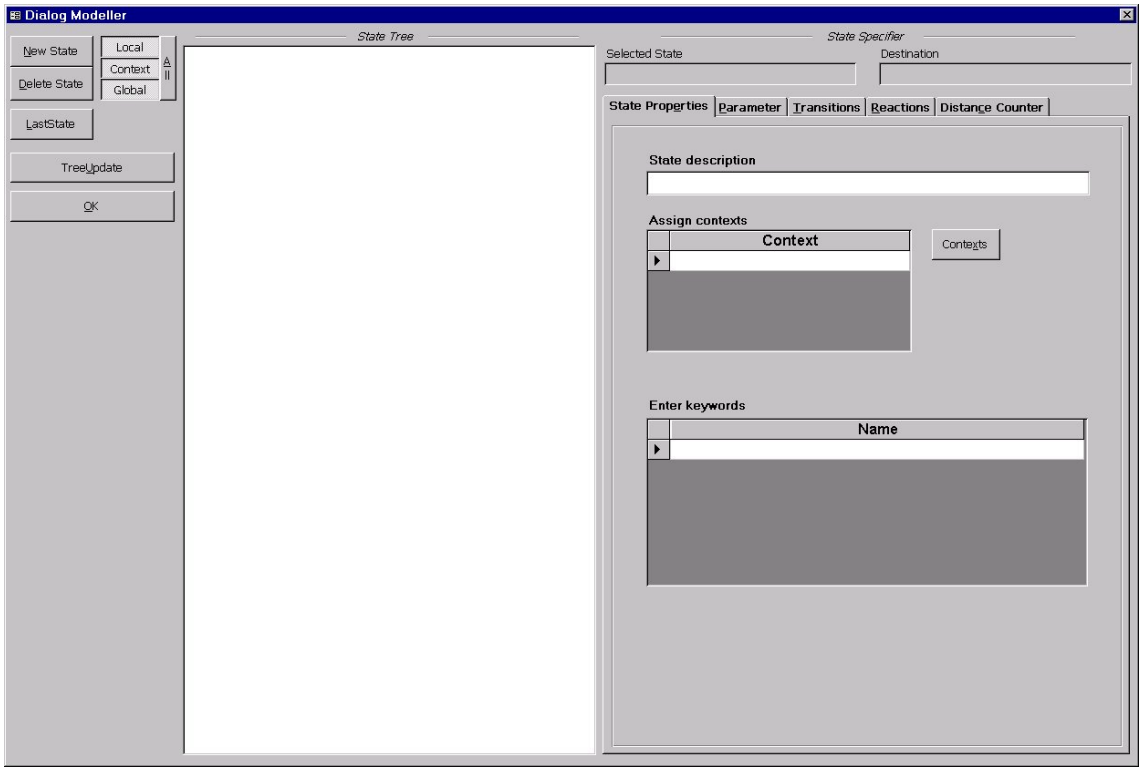


Abb. 1 Dialog Modeler

2 Dialogzustand *Hauptmenü*

2.1 Zielsetzung

Wie Sie an dem Einführungsbeispiel gesehen haben, gibt es ein zentrales Auswahlmenü – das Hauptmenü. Im ersten Schritt soll zunächst dieses Hauptmenü mit dem Controller bedienbar gemacht werden.

Dazu sind folgende Arbeitsschritte notwendig:

1. Um für einen Dialogzustand ein Verhalten definieren zu können, muss dieser zunächst vorhanden sein.
2. Beim Betreten des Zustandes *Hauptmenü* muss im GUI die richtige Bildschirmanzeige dargestellt werden.
3. Das haptische Feedback des Controllers muss parametrisiert werden.
4. Die Länge der im GUI angezeigten Liste muss mit der Anzahl an Rastenstellungen des Controllers übereinstimmen.
5. Drehbewegungen am Controller müssen entsprechend im GUI dargestellt werden.

2.2 Umsetzung

2.2.1 Zustand *Hauptmenü* anlegen

Um einen Dialogzustand anzulegen klicken Sie zunächst auf „New State“. Daraufhin erscheint im mittleren Bereich des Dialog Modeler ein Zustand *new*, dem Sie den Namen *Hauptmenü* geben (Abb. 2).

Abb. 2 Dialogzustand *Hauptmenü* anlegen

Schlüsselwörter definieren

Dialogzustände bearbeiten Sie immer auf der rechten Seite des Dialog Modeler im State Specifier. Im Reiter „State Properties“ können Sie dem eben angelegten Zustand so genannte Schlüsselwörter zuweisen. Als Schlüsselwörter werden Begriffe eingesetzt, die den jeweiligen Dialogzustand charakterisieren. Auf diese Schlüsselwörter kann später bei der Parametrisierung der einzelnen Dialogkomponenten zugegriffen werden. Geben Sie für den Zustand Hauptmenü bei „Enter keywords“ die Schlüsselwörter *Hauptmenü* und *Anfang* an (Abb. 3).

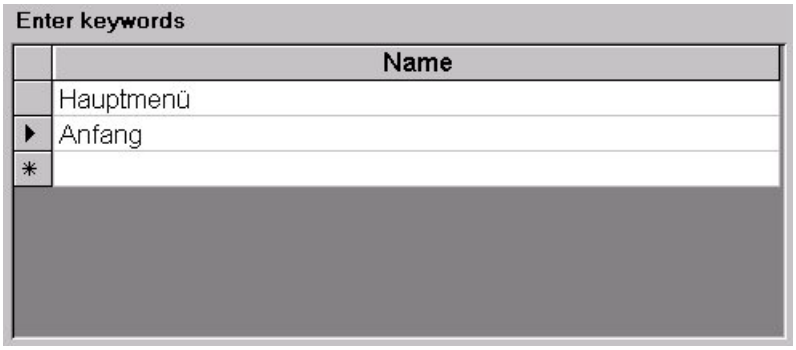


Abb. 3 Schlüsselwörter definieren

Zusammenfassung:

Abschließend wird jeder Teilschritt nochmals tabellarisch zusammengefasst.

Neuer Zustand	Schlüsselwort
Hauptmenü	*Hauptmenü*
	Anfang

Schritt 1 Zustand *Hauptmenü* anlegen

2.2.2 Reaktion für Anzeige im GUI anlegen

Nun muss dafür gesorgt werden, dass beim Betreten des Dialogzustandes *Hauptmenü* im GUI die entsprechende Anzeige dargestellt wird. Hierzu muss eine Reaktion definiert werden. Zur Auslösung der Reaktion dient die Nachricht `DM_EnterState`, die immer beim Betreten eines neuen Dialogzustandes vom Dialogmanager DM ausgelöst wird. Daraufhin versendet die Reaktion das Kommando `GUI_MM_Enter`, wodurch das GUI zur entsprechenden Anzeige umschaltet (Abb. 4).

Abb. 4 Reaktion

Vorgehensweise

Um den 1. Teil der Reaktion – die auslösende Nachricht – einzugeben, wechseln Sie im State Specifier zum Reiter „Reactions“ und wählen entsprechend Abb. 5 bei „Select incoming message“ die Nachricht `DM_EnterState` der Dialogkomponente `Dialogmanager` aus.

Hinweis:

Alle für den Demonstrator notwendigen Dialogkomponenten sind mit ihrem spezifischen Funktionsumfang bereits in EmMI integriert.

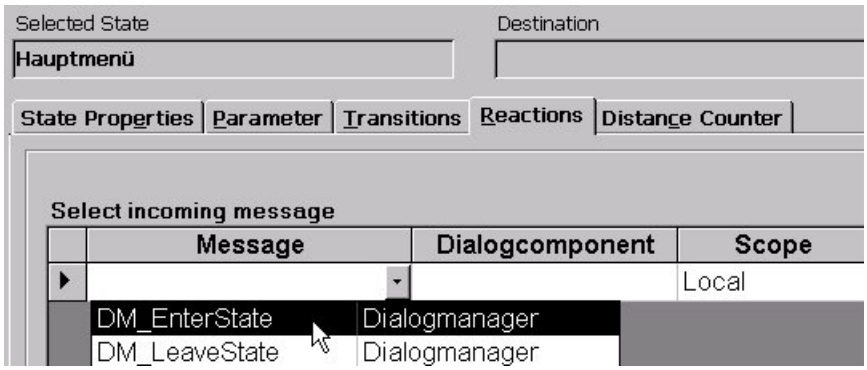


Abb. 5 Auslösende Nachricht einer Reaktion festlegen

Nun wird der 2. Teil der Reaktion (Abb. 4) – das zu versendende Kommando – definiert:

Klicken Sie hierzu im unteren Bereich des Reiters, bei „Outgoing commands“ auf das Feld „Command” und wählen dann in der daraufhin erscheinenden Dialogbox bei „Select Command“ das Kommando `GUI_MM_Enter` aus (Abb. 6). Dieses Kommando schaltet im GUI zum Hauptmenü (MM: Main Menu) um. Schließen Sie dann die Dialogbox „Reaction – Command Setting“.

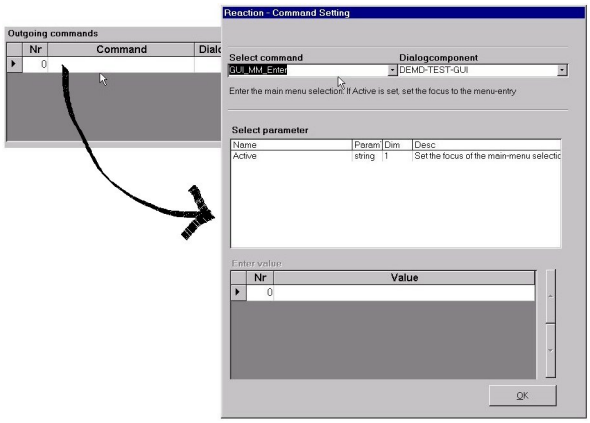


Abb. 6 Auszulösendes Kommando einer Reaktion

Zusammenfassung

Beim Betreten des Dialogzustandes *`Hauptmenü`* wird nun das Kommando `GUI_MM_Enter` an die Dialogkomponente `EMMI-TEST-GUI` versandt, wodurch diese zur Hauptmenüanzeige umschaltet.

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Hauptmenü	DM_EnterState	Local	GUI_MM_Enter		

Schritt 2 Reaktion für GUI-Anzeige anlegen

2.2.3 Reaktion zur Konfiguration des Controllers anlegen

Um eine vernünftige Funktion des Controllers EC zu gewährleisten, muss dessen haptisches Feedback für die Drehbewegungen parametrisiert werden. Analog zur GUI-Umschaltung müssen Sie jetzt eine weitere Reaktion definieren.

Für den ersten Teil der Reaktion legen Sie hierzu bei „Select incoming message" eine zweite Nachricht `DM_EnterState` an. Da die Einstellung zunächst für alle weiteren Dialogzustände gelten soll, muss der Gültigkeitsbereich „`Scope`“ auf *'Global'* gesetzt werden (Abb. 7).

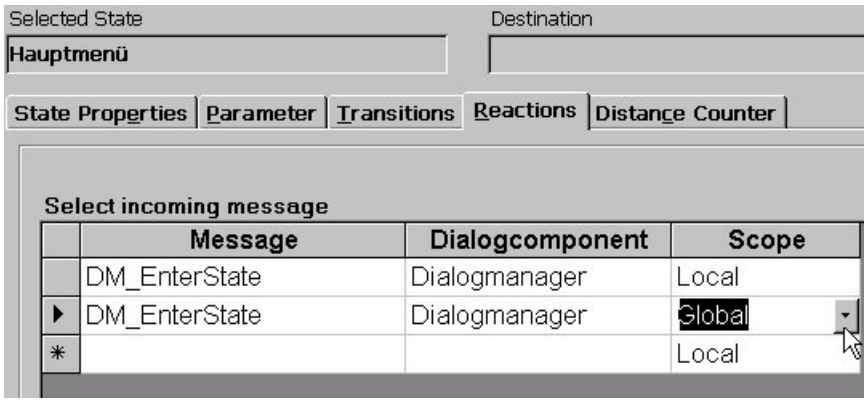


Abb. 7 Globale Reaktion

Um nun das Force-Feedback tatsächlich festzulegen, müssen Sie den zweiten Teil der Reaktion (Abb. 4) – das zu versendende Kommando – definieren. Klicken Sie dazu bei „Outgoing Commands“ auf das Feld „Command“ und wählen dann in der daraufhin erscheinenden Dialogbox bei „Select Command“ das Kommando `EC_SetEffect` der Dialogkomponente `BMWCANDLL` aus (Abb. 8). Diesem Kommando müssen nun zusätzlich mehrere Parameter für das Force-Feedback-Verhalten mitgegeben werden.

Um den Typ des Force-Feedbacks einzustellen, wählen Sie zunächst bei „Select parameter“ den Parameter `EC_Eff_Type` aus. Diesem Parameter muss nun ein Wert zugeordnet werden, der das Verhalten des Controllers definiert.

Klicken Sie dazu auf das Feld „Value“. Es öffnet sich das Fenster „Enter Value“. Geben Sie dann bei „Enter constant value“ den konstanten Wert *'EC_HARDBOUNDEDDETENT'* ein (Abb. 8).

Hinweis:

Bei der Eingabe von Strings ist auf die Groß-/Kleinschreibung zu achten.

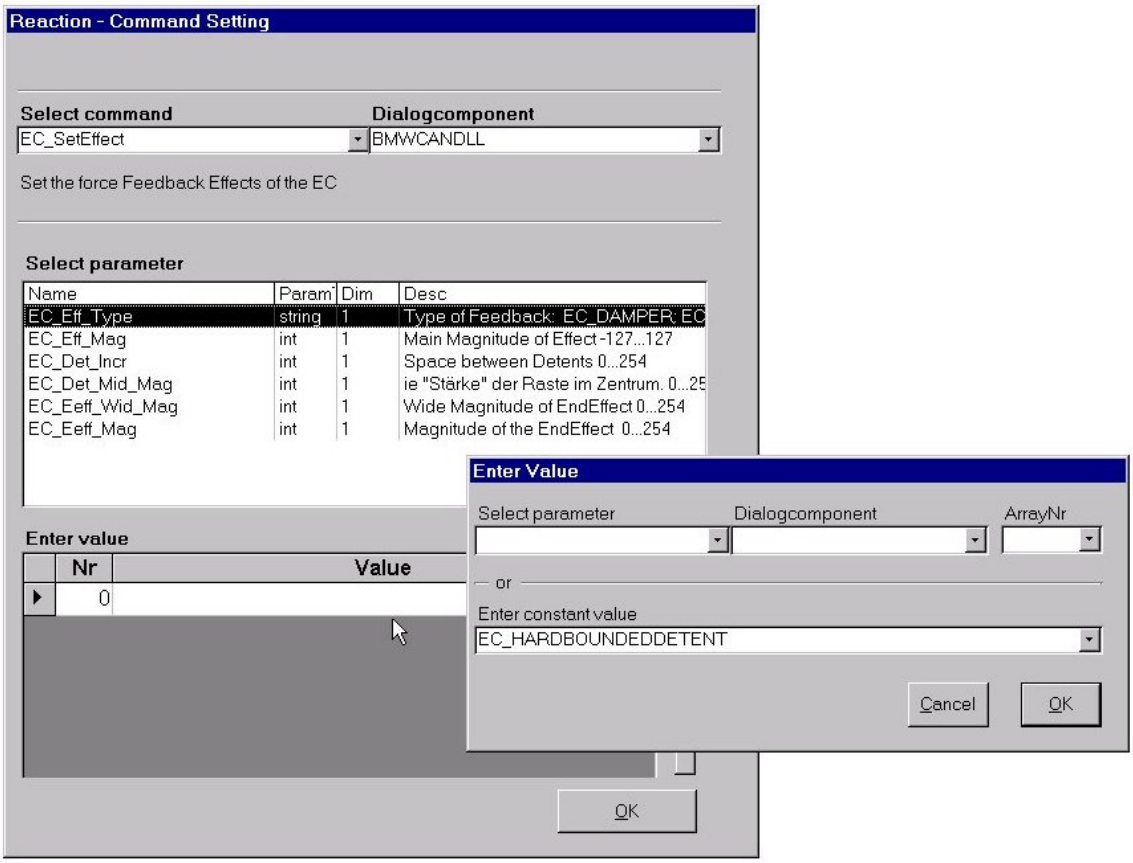


Abb. 8 Kommandoparametrisierung mit konstanten Werten

- Als Nächstes legen Sie nun die Gegenkraft der Rasterstellungen fest, indem Sie für den Parameter `EC_Eff_Mag` den konstanten Wert *120* angeben.
- Bestimmen Sie dann die Weite einer Raste mit `EC_Det_Incr` = *40*.
- Definieren Sie schließlich mittels `EC_Eeff_Mag` = *200* die Gegenkraft der rechten und linken Listenanschläge.

Zusammenfassung

Parametrisierung des Force-Feedback des Controllers.

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Hauptmenü	DM_Enter State	*Global*	EC_Set Effect	EC_Eff_Type	*'EC_HARDBOUNDEDDETENT'*
				EC_Eff_Mag	*120*
				EC_Det_Incr	*40*
				EC_Eeff_Mag	*200*

Schritt 3 Reaktion zur Parametrisierung des Force-Feedback-Verhaltens

2.2.4 Reaktion zur Synchronisation der Listenlänge im Dialogzustand *Hauptmenü*

Sobald sich die angezeigte Listenlänge – z.B. durch einen Menüwechsel – ändert, muss die Zahl der Rastenstellungen des Controllers mit dieser neuen Listenlänge angepasst werden. Zur Auslösung der Reaktion dient die Nachricht `GUI_NewListLength`, die immer bei Änderung der Listenlänge vom GUI ausgelöst wird. Daraufhin versendet die Reaktion das Kommando `EC_SetList`, womit die Zahl der Rastenstellungen festgelegt werden kann.

Für den ersten Teil der Reaktion legen Sie hierzu bei „Select incoming message" die Nachricht `GUI_NewListLength` mit lokalem Gültigkeitsbereich an.

Um die Zahl der Rastenstellungen zwischen dem rechten und linken Anschlag des Controllers anzupassen, geben Sie das zugehörige Kommando an, indem Sie auf „Command“ klicken und dann in der daraufhin erscheinenden Dialogbox bei „Select command" das Kommando `EC_SetList` auswählen.

Für dieses Kommando müssen nun zusätzlich zwei Parameter festgelegt werden.

Legen Sie die Zahl der Rastenstellungen fest, indem Sie bei „Select parameter“ `EC_Det_Quan` auswählen und dann auf Value klicken. Im Dialogfeld „Enter Value“ ordnen Sie diesem Parameter nun keinen konstanten Wert zu, sondern verknüpfen ihn bei „Select parameter“ mit dem aktuellen Wert des Parameters `GUI_ListLength` (Abb. 9). In diesem Parameter ist die Länge der aktuellen Liste im GUI gespeichert.

Hinweis:

In den bei „Select parameter“ zur Verfügung stehenden Parametern ist zu jedem Zeitpunkt der Dialogausführung der jeweils aktuelle Zustand der Dialogkomponenten verfügbar. Ihr Inhalt wird durch die Dialogkomponenten automatisch festgelegt.

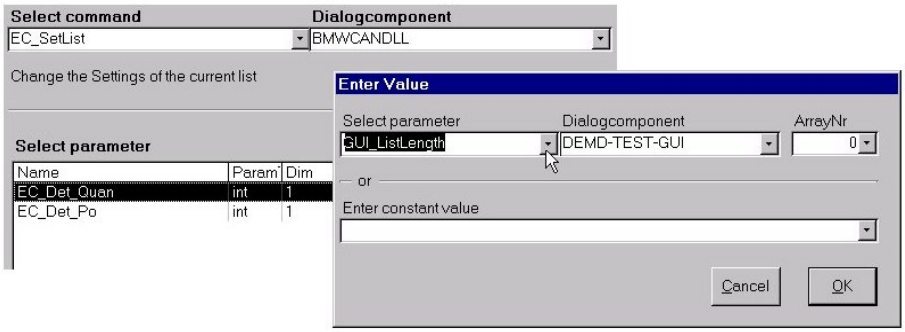


Abb. 9 Kommandoparametrisierung mit anderen Parametern

Als Nächstes müssen Sie den Parameter `EC_Det_Po`, der die Listenposition angibt, auf der sich der Controller beim Initialisieren der neuen Liste befinden soll, mit dem Parameter `GUI_MM` verknüpfen. In diesem Parameter ist der zuletzt gewählte Listeneintrag aus dem Hauptmenü abgelegt.

Zusammenfassung

Synchronisierung von Listenlänge und Startposition zwischen GUI und Controller im Hauptmenü.

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Hauptmenü	GUI_New ListLength	*Local*	EC_SetList	EC_Det_Quan	GUI_List Length[0]
				EC_Det_Po	GUI_MM[0]

Schritt 4 Reaktion zur Synchronisierung der Listenlängen

2.2.5 Reaktion nach Drehung am Controller

Wenn am Controller gedreht wird, verschickt er die Nachricht `EC_Turn` und legt seine aktuelle Listenposition im Parameter `EC_ListPos` ab. Die im GUI angezeigte Liste muss daraufhin mit dem Kommando `GUI_LST_Pos` aktualisiert werden, indem im Parameter `Pos` diese neue Listenposition übergeben wird. Da dies für alle Dialogzustände zutrifft, kann der Gültigkeitsbereich der Reaktion als „Global“ definiert werden.

Legen Sie diese Reaktion nun an.

Zusammenfassung

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Hauptmenü	EC_Turn	*Global*	GUI_LST_Pos	Pos	EC_ListPos[0]

Schritt 5 Reaktion zur Synchronisation der Listenposition

2.3 Testlauf

Starten

Beenden Sie nun den Dialog Modeler und führen einen 1. Testlauf durch. Starten Sie hierzu das Programm:

.\Aufgabe1\startmeup.bat

Es öffnen sich einige Fenster. Im Fenster “EMMI-TEST-GUI“ ist die Benutzeroberfläche zu sehen. Sie befinden sich im Hauptmenü und können nach einigen Sekunden Initialisierungszeit mit dem Controller in der Liste hin und her drehen, wobei die rechten und linken Anschläge mit der Anzeige übereinstimmen.

Beenden

Nun ist es zwar möglich in der Liste zwischen den einzelnen Elementen hin und her zu springen, aber es ist noch nicht möglich ein Element auszuwählen. Dies wird nun im nächsten Arbeitsschritt realisiert.

Beenden Sie zunächst den Testlauf durch einen Doppelklick auf:

.\Aufgabe1\shutdown.exe

3 Dialogzustand *Radio*

3.1 Zielsetzung

Aus dem Testlauf resultiert sofort das nächste Ziel, nämlich die verschiedenen Menüpunkte erreichbar zu machen. Zunächst soll der Menüpunkt *Radio* realisiert werden.

Dazu sind folgende Arbeitsschritte notwendig:

1. Um das Verhalten des Dialogs für diesen Menüpunkt definieren zu können, muss ein entsprechender Zustand in der Dialogbeschreibung vorhanden sein.
2. Damit der Dialogzustand *Radio* erreicht und auch wieder verlassen werden kann, müssen Transitionen vom *Hauptmenü* zum *Radio* und umgekehrt definiert werden.
3. Beim Betreten des Zustandes muss im GUI die richtige Bildschirmanzeige dargestellt werden.

3.2 Umsetzung

3.2.1 Dialogzustand *Radio* anlegen

Zunächst soll der Menüpunkt *Radio* realisiert werden. Da dieser Menüpunkt dem *Hauptmenü* untergeordnet ist, sollte er auch entsprechend im State Tree dargestellt sein.

Klicken Sie auf „New State“ und legen dann einen neuen Dialogzustand mit dem Namen *Radio* an (Abb. 10). Selektieren Sie den Dialogzustand *Radio* durch einen Doppelklick, worauf er rot hinterlegt wird und geben als Schlüsselwörter *Radio* und *Sender* an.

Abb. 10 Dialogzustand *Radio* erstellen

Zusammenfassung

Elternzustand	Neuer Zustand	Schlüsselwort
Hauptmenü	*Radio*	*Radio*
		Sender

Schritt 6 Dialogzustand *Radio* anlegen

3.2.2 Transitionen erstellen

Die gelben Markierungen (Abb. 11) in den Symbolen der beiden Zustände im State Tree sind Hinweiszeichen auf unzureichend definierte Stellen im Dialog. Der wegweisende Pfeil (Spitze nach links) bedeutet, dass der jeweilige Zustand nicht verlassen werden kann.

Abb. 11 Zustand kann nicht verlassen werden.

Globale Transition zum Zustand *Hauptmenü*

Entsprechend der visuellen Darstellung im GUI soll der Zustand *Hauptmenü* von jedem beliebigen Zustand aus durch Schieben des Controllers nach Norden erreichbar sein. Dies kann durch eine einzige globale Transition erreicht werden.

Um eine Transition zu definieren wählen Sie zunächst den Ausgangszustand – in diesem Fall *Radio* – durch einen Doppelklick aus. Das zugehörige Symbol wird daraufhin rot markiert und der Name des Zustands rechts oben im Dialog Modeler als „Selected State“ angezeigt.

Als nächstes wird der Zielzustand der Transition – *Hauptmenü* – durch einen einfachen Klick ausgewählt. Der Zustand erhält daraufhin eine blaue Markierung und wird im Feld „Destination“ angezeigt.

Wechseln Sie im State Specifier zum Reiter „Transitions“ und wählen bei „Select incoming message“ die Nachricht `EC_Tlt` aus. Diese Nachricht verschickt der Controller, wenn er in irgendeine Richtung geschoben wird. Die Schiebrichtung wird im Parameter `EC_Tlt_Dir` übermittelt (Abb. 12).

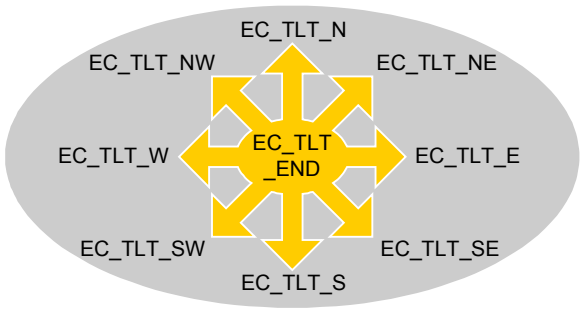


Abb. 12 `EC_Tlt_Dir` – Identifikation der Schieberichtung des Controllers

Da diese Transition in jedem Dialogzustand möglich sein soll, muss der Gültigkeitsbereich **global** definiert werden (Abb.13).

Damit die Transition allerdings nur dann ausgeführt wird, wenn der Parameter `EC_Tlt_Dir` den Wert *'EC_TLT_N'* hat, muss bei „Possible condition“ eine entsprechende Bedingung angegeben werden.

Beim Klick auf das freie Feld unter „Value 1“ öffnet sich die Dialogbox „Enter Value“. Wählen Sie als ersten Operand der Kondition den Parameter `EC_Tlt_Dir[0]`. Schließen Sie dann die Dialogbox. Stellen Sie nun sicher, dass im Feld „Op“ der Operator `'=='` ausgewählt ist. Um schließlich den zweiten Operanden einzugeben klicken Sie auf das freie Feld unter „Value 2“ und geben dann bei „Enter constant value" den konstanten Wert *`'EC_TLT_N'`* ein (Abb.13).

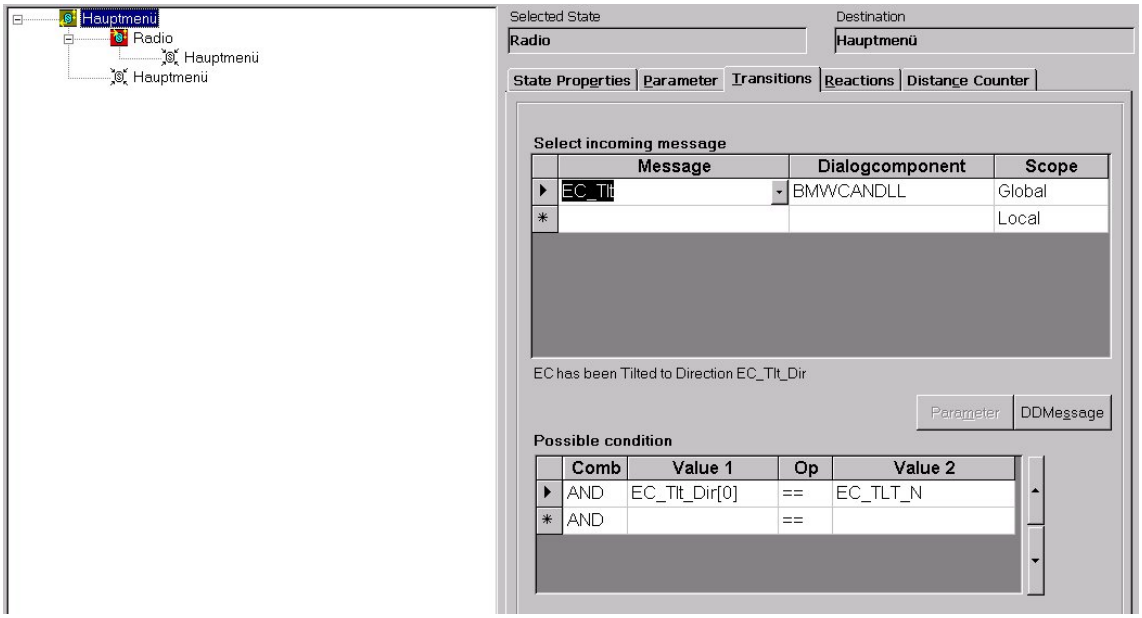


Abb.13 Transition und Kondition erstellen

Zusammenfassung

Zustand	Zielzustand	Scope	Nachricht	Bedingung
Radio	*Hauptmenü*	*Global*	EC_TLT	EC_Tlt_Dir[0] == *'EC_TLT_N'*

Schritt 7 Globale Transition zum Zustand *Hauptmenü*

Lokale Transition zum Dialogzustand *Radio*

Abb. 14 Zustand kann nicht erreicht werden.

Wie Sie bemerken, hat der Hinweispfeil des Dialogzustandes *Radio* seine Richtung geändert. Der auf den Zustand hinweisende Pfeil (Spitze nach rechts) bedeutet, dass der jeweilige Zustand nicht erreicht werden kann (Abb. 14). Der Dialogzustand könnte zwar nun verlassen werden, er kann allerdings noch nicht erreicht werden. Aus diesem Grund müssen Sie eine Transition vom Dialogzustand *Hauptmenü* in den Dialogzustand *Radio* definieren, die nur dann gültig ist, wenn der Knopf des Controllers gedrückt ist und wenn der Controller auf Listenpositi-

on 0 steht. Der Wert 0 kommt daher, weil das Radiomenü im GUI das erste Element der Liste ist.

Hinweis:

Beachten Sie, dass die Definition der Transition im State Specifier vom Zustand „Selected State“ *Hauptmenü* zum Zielzustand „Destination“ *Radio* erfolgt.

Zusammenfassung

Zustand	Zielzustand	Scope	Nachricht	Bedingung
Hauptmenü	*Radio*	*Local*	EC_Button State	EC_Button[0]==*'EC_PRESSED'* AND EC_ListPos[0]==*0*

Schritt 8 – Lokale Transition zum Zustand *Radio*

3.2.3 Reaktion zur Aktualisierung der GUI-Anzeige beim Betreten des Dialogzustandes *Radio*

Beim Betreten des Dialogzustandes *Radio* soll durch die Nachricht DM_Enter_State im GUI zum Radio-Menü umgeschaltet werden. Dies kann mit dem Kommando GUI_Station_Enter erreicht werden. Dabei müssen im Parameter Station_List die aktuell verfügbaren Sender übergeben werden. Diese sind in dem Array Radio_Station_List[] abgelegt.

Legen Sie diese Reaktion nun an.

Hinweise:

- Die Definition der Reaktion erfolgt im Zustand „Selected State“ Radio.
- Das gesamte Array wird übergeben, wenn im Dialogfeld „Enter Value“ kein Wert für „ArrayNr“ angegeben wird.

Zusammenfassung

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Radio	DM_Enter State	*Local*	GUI_Station Enter	Station_List	Radio_Station List[ ]

Schritt 9 Reaktion zur Aktualisierung der GUI-Anzeige

3.3 Testlauf

Führen Sie einen Testlauf durch und überprüfen Sie, ob Sie in das Untermenü Radio und von dort zurück ins Hauptmenü wechseln können.

4 Funktionalität Radio-Menü

Als nächster Schritt soll nun die Funktionalität des Radio-Menüs realisiert werden. Es soll also möglich sein den aktuellen Sender per Controller zu wechseln.

4.1 Zielsetzung

Zur Erstellung der Funktionalität des Radio-Menüs sind folgende Arbeitsschritte notwendig:

1. Die Listenlänge und die aktuelle Listenposition müssen zwischen Controller und GUI synchronisiert werden.
2. Drehbewegungen des Controllers müssen im GUI abgebildet werden.
3. Drehbewegungen des Controllers sollen den Radiosender wechseln.

4.2 Umsetzung

4.2.1 Reaktion zur Synchronisation der Listenlänge im Dialogzustand *Radio*

Durch die Nachricht `GUI_NewListLength` werden mit dem Kommando `EC_SetList` die Listenlängen von GUI und Controller synchronisiert. Zuerst müssen Sie den Parameter `EC_Det_Quan` mit dem Parameter `GUI_ListLength` verknüpfen. Danach wird als aktuelle Rastenstellung *`EC_Det_Po`* die Listenposition des aktuellen Radiosenders *`GUI_Station[0]`* gesetzt (analog 2.2.4).

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Radio	GUI_NewList Length	Local	EC_SetList	EC_Det_Quan	GUI_List Length[0]
				EC_Det_Po	GUI_Station [0]

Schritt 10 Reaktion zur Synchronisation der Listenlänge

4.2.2 Reaktion nach Drehung des Controllers

Zur Synchronisation der Drehbewegung des Controllers mit der Anzeige im GUI ist im Dialogzustand *Radio* bereits eine globale Reaktion für die Nachricht `EC_Turn` definiert. Diese Funktionalität wurde bereits in 2.2.5 angelegt und muss somit **nicht** erneut eingegeben werden.

4.2.3 Reaktion zum Wechsel des Radiosenders

Um zum jeweils in der Liste ausgewählten Sender zu wechseln, wird eine lokale Reaktion für die Nachricht `EC_Turn` erstellt. Der Senderwechsel wird mit dem Kommando `GUI_SetStation` realisiert. Im Parameter `Station` wird die Position des gewünschten Senders in der Senderliste angegeben. Diese Position entspricht der aktuellen Rastenstellung `EC_ListPos[0]` des Controllers.

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Radio	EC_Turn	Local	GUI_SetStation	Station	EC_ListPos[0]

Schritt 11 Reaktion zum Wechsel des Radiosenders

4.3 Testlauf

Führen Sie einen Testlauf durch und überprüfen Sie, ob Sie die Sender wechseln können.

5 Menübeschriftungen

5.1 Zielsetzung

Im GUI sind Textfelder vorgesehen, um die einzelnen Menüpunkte zu bezeichnen. Diese sollen im Folgenden definiert werden.

5.2 Umsetzung

5.2.1 Parameter für Menübeschriftungen festlegen

Klicken Sie im State Specifier auf den Reiter „Parameter“ und legen dann bei „Select Parameter“ einen neuen Eintrag an. Wählen Sie aus der angebotenen Liste den Parameter `GUI_TX_MainMenu` aus. Setzen Sie den Gültigkeitsbereich auf *'Global'*, damit der festzulegende Wert in jedem Dialogzustand gültig ist.

Hinweis:

Mit Hilfe der hier zur Verfügung stehenden Parameter können grundsätzliche Einstellungen an den einzelnen Dialogkomponenten vorgenommen werden. Ihr Inhalt wird durch den Dialogentwickler während der Dialogerstellung festgelegt.

Bei „Enter Value“ kann der festzulegende Wert direkt eingegeben oder aus der angebotenen Liste ausgewählt werden. Diese Listeneinträge entsprechen den Schlüsselwörtern des ausgewählten Zustandes „Selected State“ bzw. des Zielzustandes „Destination“ (Abb. 15). Stellen Sie deshalb sicher, dass der Zustand *Hauptmenü* ausgewählt und *Radio* als Zielzustand gewählt ist.

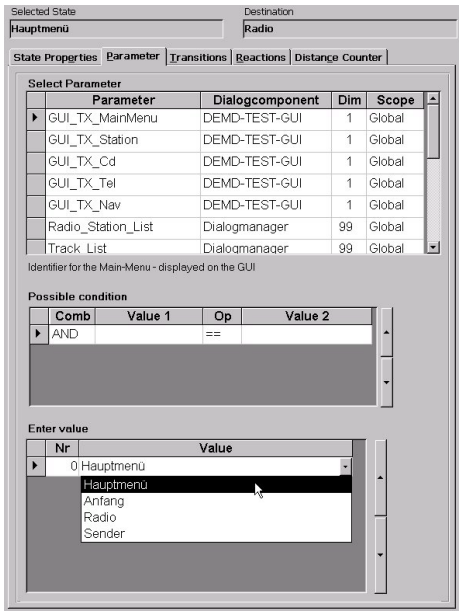


Abb. 15 Parameterdefinition

Legen Sie nun analog hierzu die Werte für die Parameter `GUI_TX_Station`, `GUI_TX_CD`, `GUI_TX_Tel` und `GUI_TX_Nav` entsprechend Schritt 12 fest.

Hinweis:

Die Menübeschriftungen für CD, Telefon und Navigation müssen Sie bei „Enter Value“ per Hand eingeben, da diese Zustände noch nicht existieren und deshalb auch keine Schlüsselwörter definiert wurden.

Zusammenfassung

Parameter	**Scope**	**Wert**
`GUI_TX_MainMenu`	`Global`	*`Hauptmenü`*
`GUI_TX_Station`	`Global`	*`Radio`*
`GUI_TX_Cd`	`Global`	*`CD`*
`GUI_TX_Tel`	`Global`	*`Telefon`*
`GUI_TX_Nav`	`Global`	*`Navigation`*

Schritt 12 Parameter für Menübeschriftungen festlegen

5.3 Testlauf

Führen Sie einen weiteren Testlauf durch und überprüfen Sie, ob die Beschriftungen angezeigt werden.

C.II Versuchsabschnitt Dialogmodifikation

1 Funktionalität Lautstärkeeinstellung

Im folgenden Versuchsabschnitt realisieren Sie die Lautstärkefunktion. Starten Sie bitte hierzu das Projekt EmMI.mdb im Verzeichnis „.\Aufgabe2“.

In diesem Projekt sind die Funktionen Radio, CD, Telefon und Navigation des Ihnen bereits bekannten Demonstrators soweit realisiert, dass sie mit dem Controller bedient werden können.

1.1 Vorüberlegung – Notwendige Arbeitsschritte

Die Lautstärkeeinstellung soll von jedem beliebigen Zustand aus durch Schieben des Controllers nach Osten erreicht werden. Danach kann die Lautstärke durch Drehen des Controllers eingestellt werden. Aus der Lautstärkeeinstellung soll in den jeweiligen Ausgangszustand durch einfaches Drücken zurückgekehrt werden können. Beschreiben Sie nachfolgend knapp mit eigenen Worten, was zur Realisierung der beschriebenen Funktionalität zu tun ist. Verwenden Sie hierzu die Begriffe „Zustand“, „Reaktion“, „Transition“ und „Gültigkeitsbereich“.

Frage 1 Arbeitsschritte zur Realisierung der Lautstärkeeinstellung

1.2 Zielsetzung

Folgende Arbeitsschritte sind notwendig:

1. Dialogzustand Volume erstellen. Für die Lautstärkeinstellung muss zunächst ein neuer Dialogzustand `Volume` erstellt werden.
2. Globale Transition definieren.
3. Reaktion zur Aktualisierung des GUI beim Betreten des Dialogzustandes *`Volume`*, damit die Volume-Anzeige aktiviert wird.
4. Reaktion zur Synchronisation der Listenlänge im Menü *`Volume`* anlegen.
5. Reaktion zur Synchronisierung des Controllers mit Lautstärke definieren.
6. Transition in den vorhergehenden Dialogzustand anlegen, damit durch Drücken des Controllers vom Zustand *`Volume`* aus der vorhergehende Zustand erreicht werden kann.
7. Reaktion zur Aktualisierung des GUI beim Verlassen des Dialogzustandes erstellen.

1.3 Umsetzung

1.3.1 Dialogzustand *`Volume`* erstellen

Elternzustand	Neuer Zustand	Schlüsselwort
`Hauptmenü`	*`Volume`*	*`Lautstärke`*
		`Volume`

Schritt 1 Dialogzustand *`Volume`* anlegen

1.3.2 Globale Transition definieren

Zustand	Zielzustand	Scope	Nachricht	Bedingung
	`Volume`	`Global`	`EC_Tlt`	`EC_Tlt_Dir[0]==’EC_TLT_E’`

Schritt 2 Globale Transition

Hinweis:

Nachdem Sie bei „Select incoming message“ die Nachricht ausgewählt und den Gültigkeitsbereich eingestellt haben, erscheint ein Warnhinweis mit dem Inhalt, dass für diese Nachricht eine Kondition eingegeben werden muss. Dies kommt daher, weil für die Nachricht `EC_Tlt` bereits eine Nachricht mit identischem Gültigkeitsbereich vorliegt. Da nicht beide Transitionen gleichzeitig ausgeführt werden können, müssen sie durch die Konditionen voneinander abgegrenzt werden.

Wodurch wird diese Transition ausgelöst und was bewirkt sie?

Frage 2 Globale Transition

1.3.3 Reaktion zur Aktualisierung der GUI-Anzeige beim Betreten des Dialogzustandes *Volume*

Beim Betreten des Zustandes *Volume* wird mit Hilfe des Kommandos GUI_Volume_Enable die Lautstärkeanzeige aktiviert.

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Volume	DM_EnterState	Local	GUI_Vol_Enable		

Schritt 3 Reaktion zur Aktualisierung der GUI-Anzeige

1.3.4 Reaktion zur Synchronisation der Listenlänge im Menü *Volume*

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Volume	GUI_NewList Length	Local	EC_SetList	EC_Det_Quan	GUI_List Length[0]
				EC_Det_Po	GUI_Volume [0]

Schritt 4 Reaktion zur Synchronisation der Listenlänge

Was passiert hier?

Frage 3 Reaktion zur Synchronisation der Listenlänge

1.3.5 Reaktion zur Synchronisierung des Controllers mit Lautstärke

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Volume	EC_Turn	Local	GUI_SetVolume	Volume	EC_ListPos [0]

Schritt 5 Reaktion zur Lautstärkeeinstellung

Was passiert hier?

Frage 4 Reaktion zur Lautstärkeeinstellung

1.3.6 Transition in den vorhergehenden Dialogzustand

Durch Drücken des Controllers soll die Lautstärkeeinstellung in den Zustand verlassen werden, aus dem sie aufgerufen worden ist. Es muss also eine Transition zum in der Dialoghistorie vorhergehenden Dialogzustand definiert werden.

Wählen Sie hierzu für den Zielzustand „Destination“ keinen Zustand im State Tree aus, sondern klicken auf den Schalter „Last State“ auf der linken Seite des Dialog Modeler. Definieren Sie dann wie gewohnt die Transition.

Zustand	Zielzustand	Scope	Nachricht	Bedingung
Volume	xxxLastStatexxx	Local	EC_Button_State	EC_Button[0]== *'EC_PRESSED'*

Schritt 6 Lokale Transition in den vorhergehenden Dialogzustand

Hinweis:

Diese Transition wird nun im State Tree des Dialog Modelers ebenfalls angezeigt.

1.3.7 Reaktion zur Aktualisierung der GUI-Anzeige beim Verlassen des Dialogzustandes

Zustand	Nachricht	Scope	Kommando	Parameter	Wert
Volume	DM_LeaveState	Local	GUI_Vol_Disable		

Schritt 7 Reaktion zur Aktualisierung der GUI-Anzeige

Was passiert hier?

Frage 5 Reaktion zur Aktualisierung der GUI-Anzeige

1.4 Testlauf

Starten Sie einen Testlauf und überprüfen Sie, ob Sie die Lautstärke verändern können.

Der Testlauf zeigt, dass sich beim Drehen des Controllers zwar die Lautstärke verändert, gleichzeitig aber auch noch das aktive Element des im Hintergrund befindlichen Menüs.

1.5 Problembehandlung

1.5.1 Problemanalyse

Da das im Testlauf festgestellte Problem im Dialogzustand *Volume* auftritt, beginnen Sie die Problemanalyse auch in diesem Zustand. Selektieren Sie per Doppelklick diesen Zustand im State Tree.

Das unerwünschte Verhalten tritt bei Drehungen am Controller auf. Diese werden durch die Nachricht `EC_Turn` an den Dialogmanager gemeldet. Betrachtet man nun die Reaktionen im Zustand *Volume*, so erkennt man, dass es zwei Reaktionen auf die Nachricht `EC_Turn` gibt:

- Eine lokal definierte, die mit dem Kommando `GUI_SetVolume` die gewünschte Lautstärke setzt.
- Daneben gibt es allerdings noch eine global definierte Reaktion, die mit dem Kommando `GUI_LST_Pos` das selektierte Element der Liste im Hintergrund verändert und somit für das unerwünschte Verhalten verantwortlich ist.

Es muss also sichergestellt werden, dass diese globale Reaktion im Zustand *Volume* nicht mehr ausgeführt wird.

1.5.2 Problembehebung

Die Reaktion darf also nicht mehr global gültig sein. Damit man sie aber nicht in allen notwendigen Dialogzuständen per Hand mit lokalem Gültigkeitsbereich eingeben muss, gibt es die Möglichkeit, beliebige Dialogzustände zu sogenannten Kontexten zusammenzufassen. Im Feld „Scope“ kann dann als Gültigkeitsbereich ein dem aktuellen Dialogzustand zugeordneter Kontext ausgewählt werden.

Hierzu sind folgende Schritte notwendig:

1. Kontext anlegen.
2. Kontext den Dialogzuständen zuordnen.
3. Gültigkeitsbereich der Reaktion ändern.

Kontext anlegen

Klicken Sie im Reiter „State Properties“ auf den Schalter „Contexts“. In der daraufhin erscheinenden Dialogbox können Sie die verfügbaren Contexte verwalten. Legen Sie einen Kontext an, in dem Sie auf „New“ klicken und dann als Namen für den neuen Kontext *Menüauswahl* angeben (Abb. 1). Schließen Sie dann die Dialogbox.

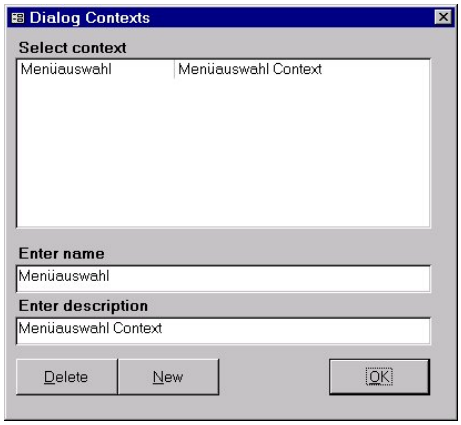


Abb. 1 Kontext *Menüauswahl* anlegen

Name	Beschreibung
Menüauswahl	*'Menüauswahl Kontext'*

Schritt 8 Kontext *Menüauswahl* anlegen

Kontext den Dialogzuständen zuordnen

Nun müssen alle Zustände, in denen obige Reaktion erfolgen soll, dem Kontext *'Menüauswahl'* zugeordnet werden. Hierzu gehören *Hauptmenü*, *Radio*, *CD*, *Telefon* und *Navigation*. Selektieren Sie den jeweiligen Zustand durch einen Doppelklick und wählen bei „Assign contexts" den Kontext *Menüauswahl* aus. Führen Sie dies für alle genannten Zustände durch (Abb. 2).

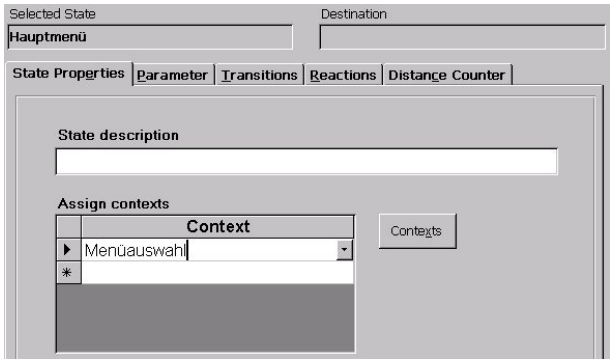


Abb. 2 Kontextzuordnung

Zustand	Kontext
Hauptmenü	*Menüauswahl*
Radio	*Menüauswahl*
CD	*Menüauswahl*
Telefon	*Menüauswahl*
Navigation	*Menüauswahl*

Schritt 9 Kontextzuordnung

Gültigkeitsbereich der Reaktion ändern

Ändern Sie schließlich den Gültigkeitsbereich der globalen Reaktion auf die Nachricht `EC_Turn`. Beantworten Sie zuerst unten stehende Fragen.

Wie muss diese Änderung aussehen?
Der Gültigkeitsbereich der Reaktion wird von *Global* nach _____ geändert.

In welchen Zuständen kann die Änderung erfolgen? (Bitte ankreuzen)	
O *Hauptmenü*	**O** *Navigation*
O *Radio*	**O** *Volume*
O *Telefon*	

Frage 6 – Gültigkeitsbereich der Reaktion ändern

Führen Sie die Änderungen nun entsprechend durch.

1.5.3 Testlauf

Überprüfen Sie die korrekte Funktion des Volume-Menüs in einem weiteren Testlauf.

2 Integration der Spracherkennung (ASR)

Im folgenden Versuchsabschnitt werden Sie einige Funktionen sprachbedienbar machen.

2.1 Vorüberlegung zur Integration der Spracherkennung

Bevor Sie mit der Realisierung beginnen, beantworten Sie bitte nachfolgende Fragen. Bei jeder Frage ist jeweils nur eine Antwort völlig richtig. Wählen Sie daher immer nur eine Antwort aus.

1.	Was muss sinnvollerweise getan werden, um die ASR-Funktionalität (Automatic Speech Recognition) zu realisieren und in EmMI zu integrieren?
O	Die ASR-Funktionen werden in einer eigenen Dialogkomponente realisiert. Die Schnittstellen zu dieser Dialogkomponente werden in Form von Nachrichten, Kommandos und Parameter in EmMI integriert und stehen dann zur Dialogbeschreibung im Dialog Modeler zur Verfügung.
O	Der Funktionsumfang des Dialogmanagers wird um die ASR-Funktionen erweitert. Da dieser den Dialog steuert, kann somit zur Laufzeit sehr leicht per Sprache auf den Dialogablauf Einfluss genommen werden.
O	Zur Realisierung der ASR-Funktionen muss ein eigener Dialogzustand angelegt werden. In diesem werden dann mit Hilfe von Nachrichten und Kommandos die Fähigkeiten des Spracherkenners angelegt.

2.	Wie kann der Dialogzustand `Radio` mit Hilfe eines Sprachkommandos erreicht werden?
O	Der Dialogzustand `Radio` kann nicht direkt per Sprache erreicht werden. Es muss ein weiterer Dialogzustand angelegt werden, der mit einer auf das Sprachkommando abgestimmten Transition erreicht wird und dieselben Definitionen enthält wie `Radio`.
O	Es muss eine zusätzliche Transition zum Dialogzustand `Radio` eingeführt werden. Diese Transition muss von einer Nachricht getriggert werden, die aufgrund des entsprechenden Sprachkommandos ausgelöst wird.
O	Im Dialogzustand `Radio` muss eine Reaktion auf eine Nachricht angelegt werden, die aufgrund des entsprechenden Sprachkommandos ausgelöst wird.

3.	Wie kann man sinnvoll und effizient erreichen, dass in jedem beliebigen Zustand durch Schieben des Controllers nach Westen das Kommando zum Starten der Spracherkennung ausgelöst wird?
O	Es muss im Zustand Hauptmenü eine lokale Reaktion erstellt werden, die von der Nachricht des Controllers getriggert wird und daraufhin das Kommando zum Starten der Spracherkennung auslöst.
O	Es muss eine globale Reaktion erstellt werden, die von der Nachricht des Controllers getriggert wird und daraufhin das Kommando zum Starten der Spracherkennung auslöst.
O	Es muss für jeden bestehenden Dialogzustand eine lokale Reaktion erstellt werden, die von der Nachricht des Controllers getriggert wird und daraufhin das Kommando zum Starten der Spracherkennung auslöst.

2.2 Zielsetzung

Verwenden Sie bitte das Projekt „EmMI.mdb“ im Verzeichnis „.\Aufgabe3“. In dem Ihnen nun bereits bekannten Projekt ist jetzt eine weitere Dialogkomponente mit dem Namen `ASR` verfügbar. In dieser Dialogkomponente ist die Funktion der Spracherkennung implementiert.

Die Dialogdefinition ist soweit realisiert, dass eine vollständige Bedienung mit dem Controller möglich ist. Des Weiteren ist die De-/Aktivierung der Spracherkennung und die entsprechende Visualisierung im GUI bereits realisiert.

Ihre Aufgabe ist es nun folgende Funktionen per Sprache bedienbar zu machen:

1. Es soll von jedem Zustand aus möglich sein, mit dem Sprachkommando „Hauptmenü“ bzw. „Anfang“ zum *`Hauptmenü`* zu wechseln.
2. Von allen Zuständen des Kontextes Menüauswahl soll in den Zustand *`Radio`* per Sprache gewechselt werden können.

2.2.1 Globale Transition in den Zustand ***`Hauptmenü`***

Der Zustand *`Hauptmenü`* soll von jedem beliebigen anderen Dialogzustand mit den Sprachkommandos *`'Hauptmenü'`* oder *`'Anfang'`* erreichbar sein. In dem Ihnen vorliegendem Projekt gibt es eine Nachricht `ASR_Hauptmenü`, für die Sie eine entsprechende globale Transition erstellen. Aufgrund des globalen Gültigkeitsbereichs der Transition kann der „Selected State“ frei gewählt werden.

Zustand	Zielzustand	Scope	Nachricht	Bedingung
	`Hauptmenü`	`Global`	`ASR_Hauptmenü`	

Schritt 10 Globale Transition zum Zustand *`Hauptmenü`*

Parametrisierung der Nachricht `ASR_Hauptmenü` zur Vokabularfestlegung

In der Dialogkomponente ASR ist die Nachricht `ASR_Hauptmenü` nicht fest mit dem Sprachkommando verknüpft. Die Zuordnung erfolgt innerhalb von EmMI, indem die Nachricht `ASR_Hauptmenü` entsprechend parametrisiert wird.

Stellen Sie hierzu zunächst sicher, dass die oben definierte Transition für die Nachricht `ASR_Hauptmenü` bei „Select incoming message“ ausgewählt ist und klicken Sie dann auf den Schalter „Settings“. In dem darauf erscheinenden Dialogfeld (Abb. 3) können Sie die Parameter der Nachricht `ASR_Hauptmenü` festlegen. Die Vorgehensweise ist dabei analog zur Parametrisierung der Kommandos. Als Erstes wählen Sie den gewünschten Parameter bei „Select Parameter“ aus und geben dann den zugehörigen Wert bei „Enter Value“ ein.

Geben Sie in dem Array `ASR_SayWord` die Schlüsselwörter *`'Hauptmenü'`* und *`'Anfang'`* an, die beim Spracherkenner zum Auslösen der Nachricht `ASR_Hauptmenü` führen (Abb. 3).

Hinweis:

Zur fehlerfreien Eingabe der Werte im Dialogfeld „Enter Value“ stehen Ihnen wiederum die Schlüsselwörter der gerade selektierten Dialogzustände in der Auswahlliste des Feldes „Enter constant value“ zur Verfügung.

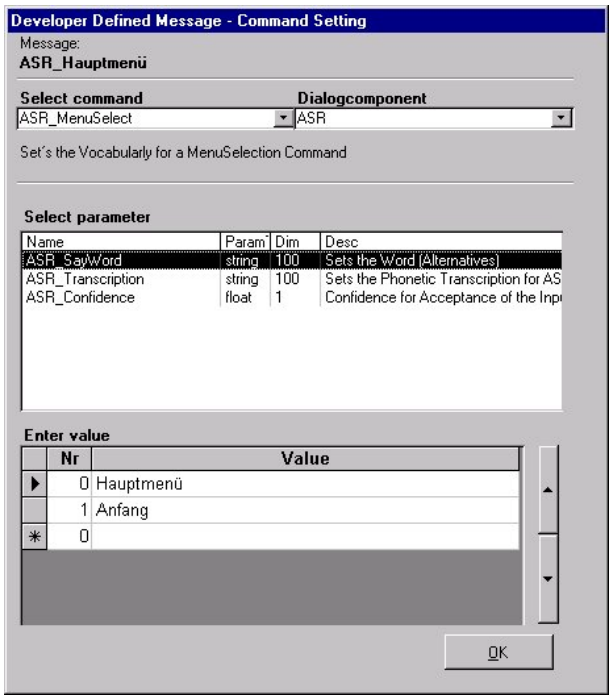


Abb. 3 Parametrisierung der Nachricht `ASR_Hauptmenü`

Nachricht	Kommando	Parameter	Wert
ASR_Hauptmenü	ASR_MenuSelect	ASR_SayWord	*'Hauptmenü'* *'Anfang'*

Schritt 11 Parametrisierung der Nachricht ASR_Hauptmenü

Nun ist es möglich von jedem Zustand des Dialogs per Sprache zum Zustand *Hauptmenü* zu springen.

Hinweis:

Die Nachricht ASR_Hauptmenü ist kein fester Bestandteil der Dialogkomponente ASR. Vielmehr handelt es sich um eine Nachricht, die während der Dialogentwurfszeit mit Hilfe eines besonderen Kommandotyps erstellt worden ist. Diese so genannten „**Developer Defined Messages**“ erlauben es den Nachrichtenumfang der einzelnen Dialogkomponente zentral innerhalb von EmMI für das entsprechende Projekt festzulegen bzw. zu erweitern.

2.2.2 Kontextweite Transition in den Zustand *Radio*

Der Wechsel zum Zustand *Radio* soll nun für alle Zustände des Kontextes Menüauswahl mit dem Sprachkommando „Radio“ bzw. „Sender“ ermöglicht werden. Wählen Sie also die Zustände „Selected State“ und „Destination“ entsprechend aus. Allerdings ist in dem bestehenden Projekt noch keine passende Nachricht der Dialogkomponente ASR vom Typ 'Developer Defined Message' definiert, die für die Transition in den Zustand *Radio* benutzt werden könnte.

Deshalb legen Sie diese nun im ersten Schritt an.

Nachricht ASR_Radio anlegen

Durch einen Klick auf den Button „DDMessage“ öffnet sich ein Fenster, in dem Sie alle Nachrichten vom Typ „Developer Defined Message“ verwalten können.

Legen Sie eine neue Nachricht an, indem Sie im Feld „Name“ den Namen *'ASR_Radio'* eingeben (Abb. 4).

Nun muss festgelegt werden, welche Dialogkomponente bzw. welche Funktion hinter dieser Nachricht steht. Klicken Sie hierzu auf „Settings“. In der daraufhin erscheinenden Dialogbox wählen Sie im Feld „Select Command“ das Kommando `ASR_MenuSelect` aus und geben analog zu 2.2.1 für den Parameter `ASR_SayWord` die Schlüsselwörter *'Radio'* und *'Sender'* an.

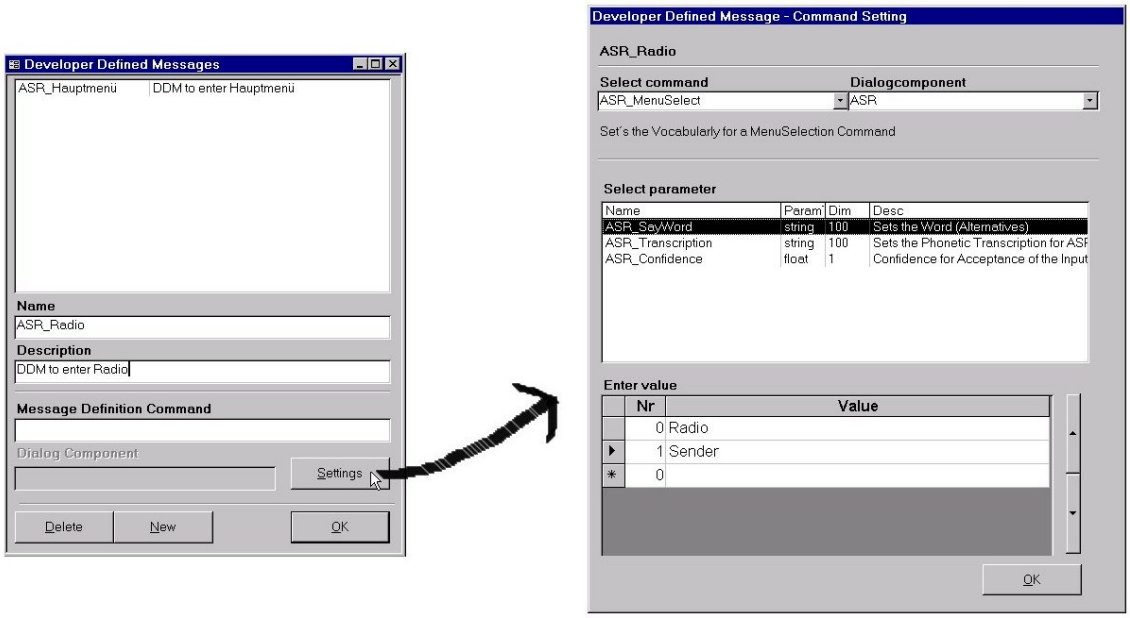


Abb. 4 Developer Defined Message – Nachricht `ASR_Radio` anlegen

Im Spracherkenner ist nun eine Nachricht bereitgestellt, die immer dann, wenn das Sprachkommando „Radio“ oder „Sender“ erkannt wird, die Nachricht ASR_Radio auslöst.

Nachricht	Kommando	Parameter	Wert
`ASR_Radio`	`ASR_MenuSelect`	`ASR_SayWord`	*'Radio'* *'Sender'*

Schritt 12 Developer Defined Message - Nachricht `ASR_Radio` anlegen

Transition anlegen

Legen Sie nun als zweiten Schritt der Aufgabe 2.2.2 eine Transition in den Zustand `Radio` an, die für alle Zustände des Kontextes `Menüauswahl` definiert ist und von der eben angelegten Nachricht `ASR_Radio` getriggert wird.

Zustand	Zielzustand	Scope	Nachricht	Bedingung
Hauptmenü	*Radio*	Menüauswahl	ASR_Radio	

Schritt 13 Transition zum Zustand *Radio* anlegen

2.3 Testlauf

Starten Sie einen Testlauf. Der Spracherkenner kann durch Schieben des Controllers nach Westen aktiviert bzw. deaktiviert werden. Die Anzeige am linken Rand des GUI informiert über den aktuellen Zustand der Spracherkennung.

Überprüfen Sie, ob die eben gemachten Zustandswechsel per Sprache realisiert werden können.

C.III Versuchsabschnitt Fehlersuche

Das mittlerweile fertiggestellte Projekt wurde mit Hilfe einiger Testpersonen auf Fehler und Vollständigkeit hin untersucht. Dabei wurden einige Problemstellen identifiziert.

Versuchen Sie nun anhand des Fehlerprotokolls aus dem Test die Fehlerstellen zu lokalisieren und zu beheben. Verwenden Sie dazu das Projekt im Verzeichnis „.\Aufgabe4“.

Gehen Sie dabei immer nach folgendem Muster vor:

1. Überprüfen Sie den Fehler im laufenden Prototypen.
2. Beschreiben Sie die wahrscheinliche Ursache des Problems.
3. Versuchen Sie den Fehler zu beheben.
4. Führen Sie einen Testlauf durch.

Nr	Fehlerbeschreibung	Information
1	Wenn man mit dem Controller vom Hauptmenü in das CD-Menü wechseln möchte, passiert nichts.	• Nachricht `EC_ButtonState`: Controller wurde gedrückt oder losgelassen. • Das CD-Menü hat die Listenposition `GUI_ListPos[0]`=1.
2	Das Sprachkommando „Radio“ wechselt aus allen Zuständen des Kontextes `Menü-auswahl` in das CD-Menü und nicht – wie gewünscht – kontextweit in das Radio-Menü.	• Nachricht `ASR_Radio`: ASR hat Sprachkommando „Radio“ erkannt.
3	Das Telefon-Menü kann nur vom Hauptmenü aus per Sprache erreicht werden. Die zusätzlich geforderten Übergänge vom Navigation-/Radio-/CD-Menü sind nicht möglich.	• Nachricht `ASR_Telefon`: ASR hat Sprachkommando „Telefon“ erkannt. • Die Zustände Hauptmenü, Navigation, Radio, CD und Telefon gehören dem Kontext `Menüauswahl` an.
4	Beim Drehen des Controllers im Volume-Menü verändert sich die Lautstärke nicht.	• Nachricht `EC_Turn`: Controller wurde zur Position `EC_ListPos` gedreht. • Kommando `GUI_SetVolume`: Stellt Lautstärke auf den Wert `Volume`.

5	Die Anzeige „Auflegen“ während des Gesprächaufbaus ist schwer verständlich und soll deshalb durch „Beenden“ ersetzt werden.	• Gesprächsaufbau wird im Zustand `Connect` repräsentiert. • Die Anzeige während des Gesprächsaufbaus wird im Parameter `GUI_TX_Tel_Disconnect` festgelegt.
6	Das Navigation-Menü soll neben dem Kommandowort „Navigation“ auch mit dem Kommando „Karte“ erreicht werden.	• Nachricht `ASR_Navigation`: Löst die Transition zum Navigation-Menü per Sprache aus.

C.IV Fragebogen Expertenbewertung

Im zurückliegenden Experiment hatten Sie die Gelegenheit EmMI mit einem kurzen Tutorial kennen zu lernen und damit beispielhaft einen multimodalen Dialog zu erstellen bzw. zu modifizieren.

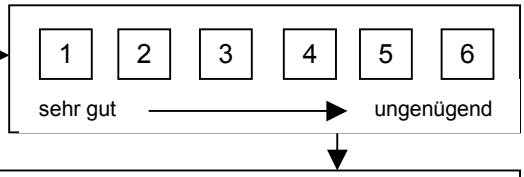
1 Gebrauchstauglichkeit

Uns interessiert nun, wie Sie die Gebrauchstauglichkeit von EmMI zur Erstellung und Gestaltung multimodaler Dialoge einschätzen. Versuchen Sie bitte die Gebrauchstauglichkeit von EmMI zu bewerten, indem Sie wie folgt vorgehen:

1. Geben Sie zu spezifischen Aspekten der Gebrauchstauglichkeit von EmMI zuerst ein Rating ab. Als Bewertungsgrundlage nutzen Sie bitte ein Rating von sehr gut (1) bis ungenügend (6).
2. Nach dieser Bewertung haben Sie die Gelegenheit Ihre Beurteilung zu EmMI in einem Kommentar näher zu erläutern.

1.1 Zur Zweckmäßigkeit von EmMI

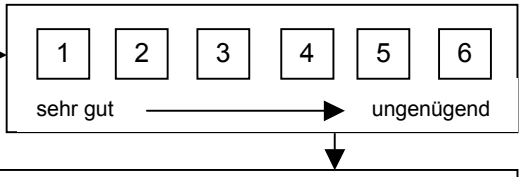
Wie gut sind die bereitgestellten Methoden und semantischen Mittel geeignet, um einen multimodalen Dialog zu spezifizieren und prototypisch zu realisieren?



(mögliche Kommentare)

1.2 Zur Effektivität von EmMI

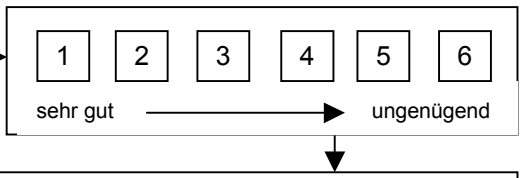
Wie effektiv sind die bereitgestellten Methoden und semantischen Mittel, um multimodale Dialoge einfach, schnell und fehlerfrei zu erstellen?



(mögliche Kommentare)

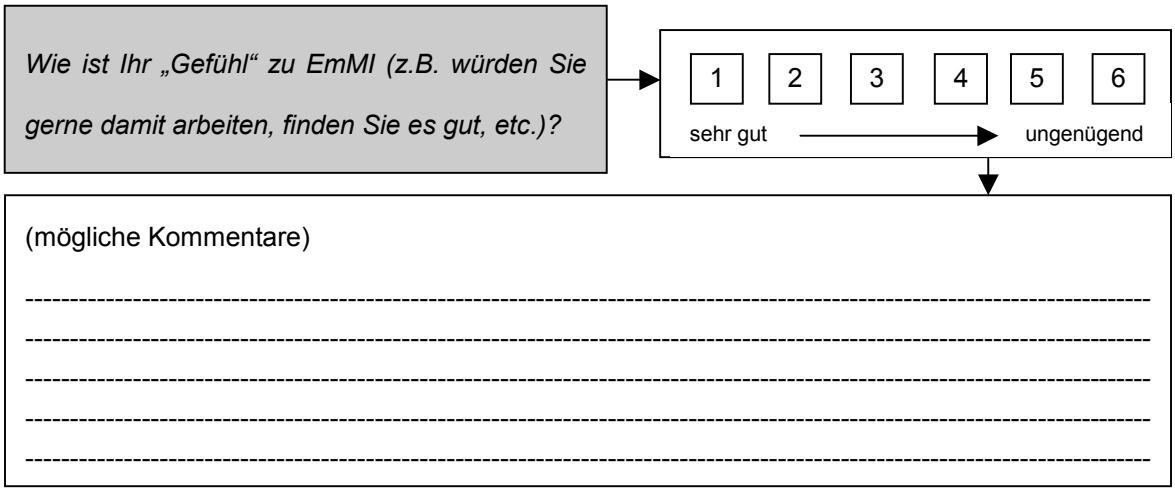
1.3 Zur Erlernbarkeit von EmMI

Wie verständlich ist EmMI aufgebaut, um sich in die adäquate Nutzung möglichst rasch einzuarbeiten?



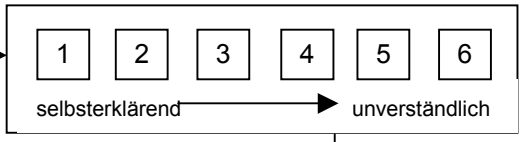
(mögliche Kommentare)

1.4 Zur persönlichen Einstellung zu EmMI



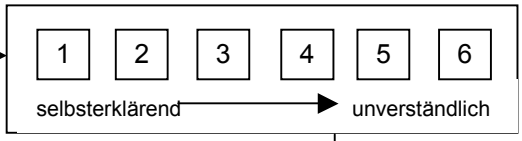
2 Zum Verständnis des Dialog Modelers

Wie verständlich ist die Darstellung des Dialogmodells als Baumstruktur im Bereich „State Tree“ (Mitte)?



(mögliche Kommentare)

Wie verständlich ist der Aufbau des Dialog Modelers im Bereich „State Specifier“ (rechte Seite)?



(mögliche Kommentare)

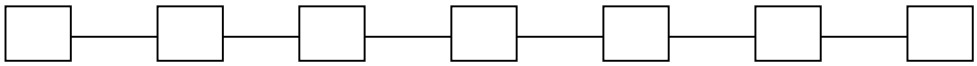
---

3 Zur Semantik der Dialogdefinition

Als wesentliche semantische Elemente von EmMI haben Sie Dialogzustände, Transitionen, Reaktionen und Parameter kennengelernt. Beurteilen Sie bitte im Folgenden diese semantischen Elemente an Hand von Eigenschafts-Paaren.

Beispiel:

EmMI ist ...

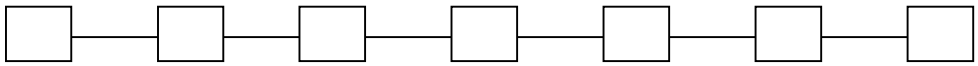


gut schlecht

- Halten Sie EmMI für sehr gut, sollten Sie das linke, äußere Kästen über „gut“ ankreuzen.
- Halten Sie EmMI für sehr schlecht, sollten Sie das rechte, äußere Kästchen über „schlecht“ ankreuzen.
- Halten Sie EmMI weder für „gut" noch „schlecht", sollten Sie ein passendes Kästchen in der Mitte ankreuzen.

3.1 Dialogzustand

Das semantische Element „Dialogzustand“ in EmMI ist ...



verständlich unverständlich

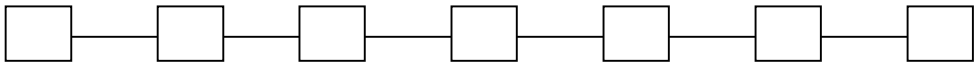
Die Erlernbarkeit im Umgang ist ...

spontan/einfach langwierig

(Mögliche Anmerkungen)

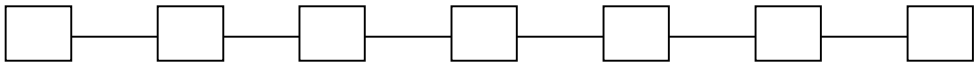
3.2 Transitionen

Das semantische Element „Transition“ in EmMI ist ...



verständlich unverständlich

Die Erlernbarkeit im Umgang ist ...



spontan/einfach langwierig

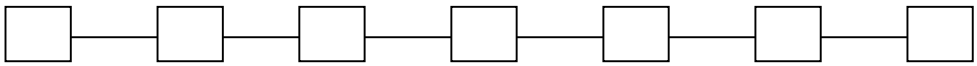
(Mögliche Anmerkungen)

3.3 Reaktionen

Das semantische Element „Reaktion“ in EmMI ist ...

verständlich unverständlich

Die Erlernbarkeit im Umgang ist ...

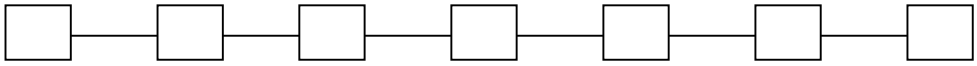


spontan/einfach langwierig

(Mögliche Anmerkungen)

3.4 Parameter

Das semantische Element „Parameter“ in EmMI ist ...



verständlich unverständlich

Die Erlernbarkeit im Umgang ist ...

spontan/einfach langwierig

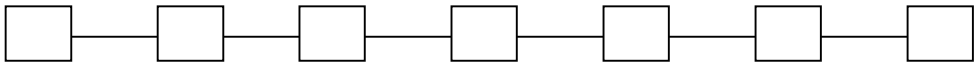
(Mögliche Anmerkungen)

4 Zu den Maßnahmen der Dialogqualitätssicherung

4.1 Hierarchische Gliederung der Dialogbeschreibung

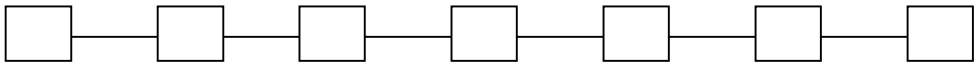
Um in mehreren Zuständen identisches Verhalten zu garantieren, können mit der hierarchischen Gliederungsmöglichkeit die Gültigkeitsbereiche der einzelnen semantischen Elemente entsprechend festgelegt werden. Wie beurteilen Sie Möglichkeit zur hierarchischen Gliederung der Dialogbeschreibung für Transitionen, Reaktionen und Parameter?

Die Definition/ Funktionsweise ist...



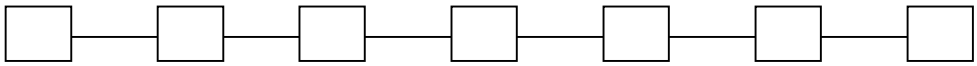
verständlich unverständlich

Die Nutzung ist...



einfach kompliziert

Die Verwendung bei der Spezifikation multimodaler Dialoge ist...



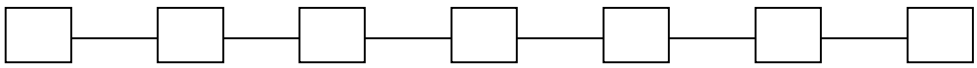
sinnvoll sinnlos

(Mögliche Anmerkungen)

4.2 Schlüsselwörter

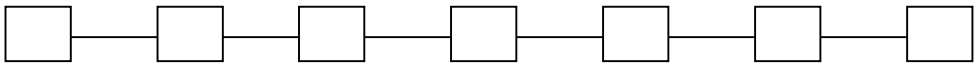
Mit Schlüsselwörter können Sie einen bestimmten Dialogzustand charakterisieren. Wie beurteilen Sie den Gebrauch von Schlüsselwörter, insbesondere zur konsistenten Parametrisierung der einzelnen Dialogkomponenten?

Der Einsatz ist...



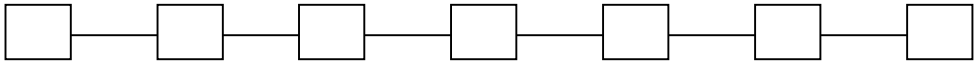
verständlich unverständlich

Die Nutzung ist...



offensichtlich versteckt

Die Verwendung bei der Spezifikation multimodaler Dialoge ist...



sinnvoll sinnlos

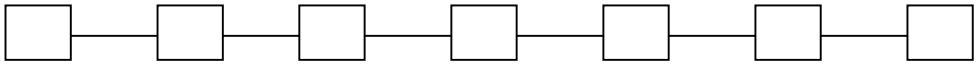
(Mögliche Anmerkungen)

4.3 Distance Counter

Als weiteres Werkzeug zur Erhöhung der Dialogqualität multimodaler Dialoge ist ein „Distance Counter“ denkbar, mit dessen Hilfe die „Entfernung“ zweier Dialogzustände überprüft werden kann. Solch ein Distance Counter wäre also eine Anzeige, die angibt, ob und über wie viele Transitionen man von Zustand A nach Zustand B kommt. (Da die Transitionen normalerweise von Nutzereingaben ausgelöst werden, lässt sich somit eine Aussage über die Zahl der notwendigen Interaktionsschritte machen.)

Für wie sinnvoll würden Sie so ein Werkzeug halten?

Die Verwendung eines Distance Counter in der Designphase von multimodalen Dialogen als Nachweis über die Anzahl der nötigen Bedienschritte eines Dialogs ist...



sinnvoll sinnlos

Wozu wäre so ein Nachweis nützlich?

_____

_____

_____

Leistet er einen Beitrag zur Dialogqualität?

_____

_____

_____

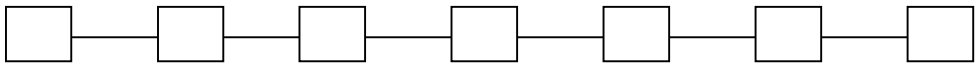
Wie genau würden Sie ihn nutzen?

_____
_____
_____

4.4 Erreichbarkeitsanzeige

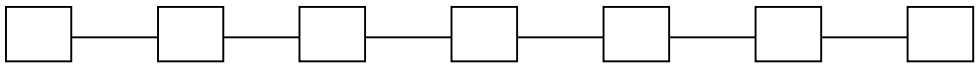
Im Bereich „State Tree“ werden diejenigen Dialogzustände, die nicht erreichbar sind oder nicht verlassen werden können, weil keine entsprechenden Transitionen definiert worden sind, besonders gekennzeichnet.

Die Anzeige ist...



verständlich unverständlich

Die Verwendung bei der Gestaltung multimodale Dialoge ist...



sinnvoll sinnlos

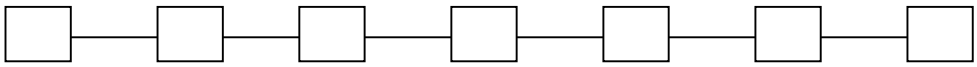
(Mögliche Anmerkungen)

5 Zu den Dialogkomponenten

Zur Realisierung eines multimodalen Dialogs müssen verschiedene Dialogkomponenten mit unterschiedlichsten technologischen Voraussetzungen zusammengeführt werden. In EmMI erfolgt die Kommunikation zu diesen Dialogkomponenten ausschließlich über Nachrichten und Kommandos. Diese kontextfreie Modellierung der Schnittstellen erlaubt die Integration zusätzlicher Dialogkomponenten, ohne Veränderungen an EmMI selbst vornehmen zu müssen. Das Verhalten der Dialogkomponenten kann dann mittels Kommandos und Parameter zentral im Dialog Modeler konfiguriert werden.

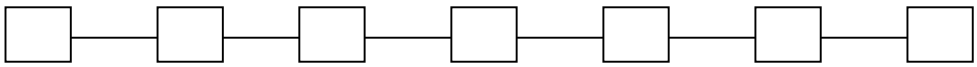
Abschließend einige Fragen zum Verständnis dieses Konzepts:

Eine kontextfreie Behandlung von Dialogkomponenten ist grundsätzlich ...



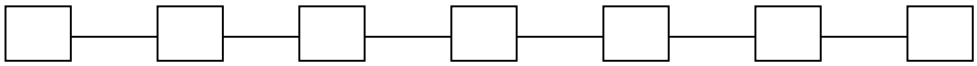
sinnvoll sinnlos

Die kontextfreie Behandlung der Dialogkomponenten in EmMI ist ...



verständlich unverständlich

Eine zentrale Konfigurationsmöglichkeit der Dialogkomponenten ist grundsätzlich ...



sinnvoll sinnlos

Die zentrale Konfigurationsmöglichkeit der Dialogkomponenten in EmMI ist ...

verständlich unverständlich

Eine kontextfreie Kommunikationsschnittstelle zu den Dialogkomponenten ist grundsätzlich...

sinnvoll sinnlos

Die kontextfreie Kommunikation der Dialogkomponenten über Nachrichten und Kommandos in EmMI ist ...

verständlich unverständlich

6 Zu Ihrer Person

Abschließend nun noch einige Fragen zu Ihrer Person:

Welche Studienrichtung haben Sie besucht?

_____

Mit welchen Anwendungen haben Sie am PC am häufigsten zu tun?

_____
_____
_____

Wie lange beschäftigen Sie sich schon intensiv mit Mensch-Maschine-Interaktion und Bediendialogen?

_____ *Jahre*

Welche Tools haben Sie bisher im Bereich Mensch-Maschine-Interaktion und zur Realisierung von Bediendialogen verwendet?

_____
_____
_____

Vielen Dank für Ihre Mitarbeit!!!

C.V Detaillierte Auswertung der Fehlerstellen

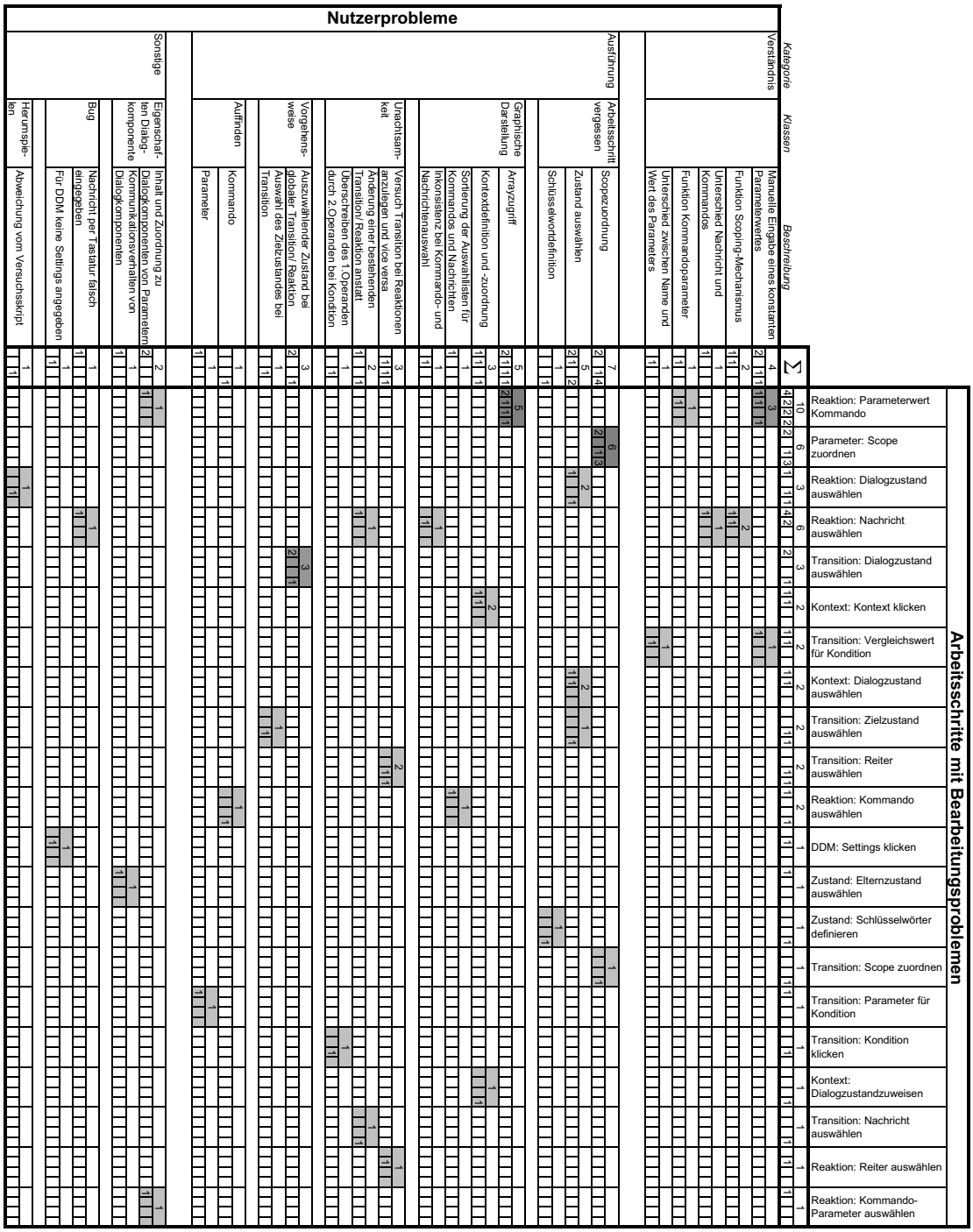


Bild 67 Zusammenhang Nutzerprobleme und Arbeitsschritte mit Bedienproblemen

Literaturverzeichnis

[2000/53/EC 2000] 2000/53/EC: *Commission Recommendation of 21 December 1999 on safe and efficient in-vehicle information and communication systems: A European statement of principles on human machine interface.* In: Official Journal of the European Communities L19 43 (25.01.2000), S. 64-68

[Akyol*, et al.* 2000] Akyol, S. ; Canzler, U. ; Bengler, K. & Hahn, W.: *Gestengesteuerter Nachrichtenspeicher im Kraftfahrzeug*. In: Gärtner, K.-P. (Hrsg.): *Proceedings of Multimodale Interaktion im Bereich der Fahrzeug- und Prozessführung. DGLR-Bericht 2000-02*, (München, 2000), Deutsche Gesellschaft für Luft- und Raumfahrttechnik e.V. (DGLR), S. 319-328

[Althoff*, et al.* 2001] Althoff, F. ; McGlaun, G. & Lang, M.: *Combining Multiple Input Modalities for Virtual Reality Navigation - A user study*. In: *Proceedings of HCII 2001: 9th International Conference on Human Computer Interaction*, (New Orleans, 2001)

[Althoff*, et al.* 2002] Althoff, F. ; Geiss, K. ; McGlaun, G. ; Schuller, B. & Lang, M.: *Experimental Evaluation of User Errors on the Skill Based Level in an Automotive Environment*. In: *Proceedings of CHI 02: International Conference on Human Factors in Computing Systems*, (Minneapolis, 2002), S. 782-783

[Bager 2001] Bager, J.: *Überall-Büros: Termine, Adressen und Aufgaben per Web-Browser und WAP-Handy verwalten*. In: c´t (2001), Nr. 24, S. 198

[Baggen & Hemmerling 2000] Baggen, R. & Hemmerling, S.: *Evaluation von Benutzbarkeit in Mensch-Maschine-Systemen*. In: Timpe, K.-P. ; Jürgensohn, T. & Kolrep, H. (Hrsg.): *Mensch-Maschine-Systemtechnik*. Düsseldorf: Symposion Publishing, 2000, S. 232-284

[Bekiaris*, et al.* 1997] Bekiaris, E. ; Machate, J. & Burmester, M.: *Towards an intelligent multimodal and multimedia user interface providing a new dimension of natural HMI in the teleoperation of all home appliances by E&D users*. In: *Proceedings of Interfaces 97*, (Montpellier, 1997), S. 226-229

[Bengler 1995] Bengler, K.: *Gestaltung und experimentelle Untersuchung unterschiedlicher Präsentationsformen von Wegleitungsinformation in Kraftfahrzeugen*. Regensburg, Universität Regensburg, Lehrstuhl für Allgemeine und Angewandte Psychologie, Dissertation, 1995

[Bengler, *et al.* 2000] Bengler, K. ; Geutner, P. ; Niedermaier, B. & Steffens, F.: *"Eyes free - Hands free" oder "Zeit der Stille". Ein Demonstrator zur multimodalen Bedienung im Fahrzeug.* In: Gärtner, K.-P. (Hrsg.): *Proceedings of Multimodale Interaktion im Bereich der Fahrzeug- und Prozessführung. DGLR-Bericht 2000-02*, (München, 2000), Deutsche Gesellschaft für Luft- und Raumfahrttechnik e.V. (DGLR), S. 299-307

[Bengler 2001] Bengler, K.: *Aspekte der multimodalen Bedienung und Anzeige im Automobil*. In: Jürgensohn, T. & Timpe, K.-P. (Hrsg.): *Kraftfahrzeugführung*. Berlin: Springer, 2001, S. 195-205

[Bengler, *et al.* 2002] Bengler, K. ; Herrler, M. & Künzner, H.: *Usability Engineering bei der Entwicklung von iDrive*. In: it+ti - Informationstechnik und Technische Informatik 44 (2002), Nr. 3, S. 145-152

[Bers, *et al.* 1997] Bers, J. ; Miller, S. & John, M.: *Designing Conversational Interfaces for Mobile Networked Computing.* In: *Proceedings of Workshop on Perceptual User Interfaces*, (Banff, 1997), S. 19-21

[BMW 1998a] BMW: *Zusatzbetriebsanleitung zur Sprachsteuerung. Kommando geben statt bedienen.* München: BMW AG, 1998a

[BMW 1998b] BMW: *Betriebsanleitung zum Bordmonitor mit Navigation und TV. Freude an der Technik – Freude am Fahren.* München: BMW AG, 1998b

[BMW 2001] BMW: *Betriebsanleitung zum Fahrzeug*. München: BMW AG, 2001

[Boff & Lincoln 1988] Boff, K.R. & Lincoln, J.E. (Hrsg.): *Engineering Data Compendium: Human Perception and Performance.* Bd. 3. Ohio: Harry G. Armstrong Aerospace Medical Research Laboratory Wright-Patterson Air Force Base, 1988

[Bolt 1980] Bolt, R.A.: *Put that there: Voice and gesture at the graphics interface.* In: ACM Computer Graphics 14 (1980), Nr. 3, S. 262-270

[Bregler, *et al.* 1993] Bregler, C. ; Hild, H. ; Manke, S. & Waibel, A.: *Improving connected letter recognition by lipreading*. In: *Proceedings of ICASSP: IEEE International Internatinoal Conference on Acoustics, Speech, and Signal Processing*, Bd. 1, (Minneapolis, 1993), S. 557-560

[Brewster, *et al.* 1993] Brewster, S.A. ; Wright, P.C. & Edwards, A.D.N.: *An evaluation of earcons for use in auditory human-computer interfaces*. In: Ashlund, S. ; Mullet, K. ; Henderson, A. ; Hollnagel, E. & White, T. (Hrsg.): *Proceedings of InterCHI'93*, (Amsterdam, 1993), ACM Press, S. 222-227

[Card, *et al.* 1983] Card, S.K. ; Moran, T.P. & Newell, A.: *The psychology of human-computer interaction*. Hillsdale: Lawrence Erlbaum Associates, 1983

[Churcher, *et al.* 1997] Churcher, G.E. ; Atwell, E.S. & Souter, C.: *Dialogue Management Systems: a Survey and Overview*. Leeds, University of Leeds, School of Computer Studies, Research Report Nr.97.06., 1997

[Cohen 1997] Cohen, P.R.: *Dialogue Modeling*. In: Cole, R. ; Mariani, J. ; Uszkoreit, H. ; Varile, G.B. ; Zaenen, A. ; Zampolli, A. & Zue, V. (Hrsg.): *Survey of the State of the Art in Human Language Technology*. Cambridge: Cambridge University Press, 1997, S. 204-210

[Cohen, *et al.* 1997] Cohen, P.R. ; Johnston, M. ; McGee, D. ; Oviatt, S. ; Pittman, J. ; Smith, I. ; Chen, L. & Clow, J.: *QuickSet: Multimodal Interaction for Distributed Applications*. In: *Proceedings of Multimedia' 97: Fifth International Multimedia Conference*, (1997), ACM Press, S. 31-40

[Coutaz 1993] Coutaz, J.: *Software architecture modeling for user interfaces*. In: Marciniak, J.J. (Hrsg.): *The Encyclopedia of Software Engineering*. Chichester: Wiley, 1993, S. 38-49

[Coutaz, *et al.* 1995a] Coutaz, J. ; Nigay, L. & Salber, D.: *Agent-Based Architecture Modeling for Interactive Systems*. In: Benyon, D. & Palanque, P. (Hrsg.): *Critical Issues in User Interface Engineering*. London: Springer, 1995a, S. 191-209

[Coutaz, *et al.* 1995b] Coutaz, J. ; Nigay, L. ; Salber, D. ; Blandford, A. ; May, J. & Young, R.M.: *Four easy pieces for assessing the usability of multimodal interaction: the CARE properties*. In: *Proceedings of INTERACT'95: Fifth IFIP Conference on Human-Computer Interaction*, (1995b), S. 115-120

[Daimler-Benz 1998] Daimler-Benz: *Sprachbediensystem "Linguatronic" - Betriebsanleitung*. Stuttgart: Daimler-Benz, 1998

[Denecke & Yang 2000] Denecke, M. & Yang, J.: *Partial Information in Multimodal Dialog*. In: *Proceedings of ICMI 2000: International Conference on Multimodal Interfaces*, (Beijing, 2000)

[Detemple 2000] Detemple, P.: *Natürlichsprachliche Bedienung im Kraftfahrzeug - Erweiterung des Dialogkonzeptes um eine Komponente zur Steuerung von Mensch-Maschine-Dialogen*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diplomarbeit, 2000

[Elan 1999] Elan: *Elan Text To Speech Engine V2.00 - Software Development Kit for Windows*. Toulouse: Elan informatique, 1999

[Evers 1999] Evers, M.: *A Case Study on Adaptability Problems of the Separation of User Interface and Application Semantics*. Enschede, University of Twente, Department of Computer Science, Report TR-CTIT 99-14, 1999

[Färber & Müller 2000] Färber, B. & Müller, M.: *Evaluation von Bedienkonzepten mit dem System NICE*. In: Bundesanstalt für Straßenwesen (Hrsg.): *Informations- und Assistenzsysteme im Auto benutzergerecht gestalten*. Bergisch Gladbach: Wirtschaftsverlag NW, 2000, S. 50-55

[Faure & Julia 1994] Faure, C. & Julia, L.: *An Agent-Based Architecture for a Multimodal Interface*. In: *Proceedings of AAAI'94 : Spring Symposium on Intelligent MultiMedia Multi-Modal Systems*, (Palo Alto, 1994), S. 82-86

[Geiger, *et al.* 2001] Geiger, M. ; Zobl, M. ; Bengler, K. & Lang, M.: *Intermodal Differences in Distraction Effects while Controlling Automotive User Interfaces*. In: *Proceedings of HCII 2001: 9th International Conference on Human Computer Interaction*, (New Orleans, 2001), S. 263-267

[Geiser 1990] Geiser, G.: *Mensch-Maschine-Kommunikation*. München: Oldenbourg, 1990

[Gourdol, *et al.* 1992] Gourdol, A. ; Nigay, L. ; Salber, D. & Coutaz, J.: *Two Case Studies of Software Architecture for Multimodal Interactive Systems: VoicePaint and a voice-enabled Graphical Notebook*. In: Larson, J. & Unger, C. (Hrsg.): *Proceedings of IFIP TC2/WG2.7: Working Conference on Engineering for Human Computer Interaction*, (1992), North Holland Publications, S. 271-284

[Green 1985] Green, M.: *Report on Dialogue Specification Tools*. In: Pfaff, G.E. (Hrsg.): *User Interface Management Systems*. Berlin: Springer Verlag, 1985, S. 9-20

[Hagen & Popowich 2000] Hagen, E. & Popowich, F.: *Flexible Speech Act Based Dialogue Management*. In: *Proceedings of 1st SIGdial Workshop on Discourse and Dialogue*, (Hong Kong, 2000)

[Hall & Llina 1997] Hall, D.L. & Llina, J.: *An introduction to multisensor data fusion*. In: Proceedings of the IEEE 85 (1997), Nr. 1, S. 6-23

[Haller 1999] Haller, R.: *HMI - New Technologies and Safety Aspects*. In: *Proceedings of the Conference Traffic Safety on Two Continents.*, Bd. 1, (Malmö, 1999), VTI konferens 13A, S. 7-15

[Harel 1987] Harel, D.: *Statecharts: A visual formalism for complex systems.* In: Science of Computer Programming 8 (1987), Nr. 3, S. 231-274

[Harel, *et al.* 1987] Harel, D. ; Pnueli, A. ; Schmidt, J.P. & Sherman, R.: *On the formal semantics of statecharts*. In: *Proceedings of the Symposium on Logic in Computer Science*, (Ithaca, 1987), S. 54-64

[Harel & Kahana 1992] Harel, D. & Kahana, C.-A.: *On statecharts with overlapping*. In: ACM Transactions on Software Engineering and Methodology 1 (1992), Nr. 4, S. 399-421

[Hauptmann 1989] Hauptmann, A.G.: *Speech and gestures for graphic image manipulation.* In: Helander, M. (Hrsg.): *Proceedings of ACM CHI'89 Conference on Human Factors in Computing Systems*, (1989), S. 241-245

[Hedicke 2000] Hedicke, V.: *Multimodalität in Mensch-Maschine-Schnittstellen*. In: Timpe, K.-P. ; Jürgensohn, T. & Kolrep, H. (Hrsg.): *Mensch-Maschine-Systemtechnik*. Düsseldorf: Symposion Publishing, 2000, S. 203-232

[Herfet & Kirste 2001] Herfet, T. & Kirste, T.: *EMBASSI - Multimodal Assistance für Infotainment & Service Infrastructures*. In: Wolf, G. & Klein, G. (Hrsg.): *Proceedings of International Status Conference: Lead Projects Human-Computer-Interaction*, (Saarbrücken, 2001), Projektträger des BMBF für Informationstechnik: Deutsches Zentrum für Luft- und Raumfahrttechnik (DLR) e.V., S. 35-43

[Hewett, *et al.* 1992] Hewett, T.T. ; Baecker, R. ; Card, S. ; Carey, T. ; Gasen, J. ; Mantei, M. ; Perlman, G. ; Strong, G. & Verplank, W.: *ACM SIGCHI Curricula for Human-Computer Interaction*. New York: ACM, 1992

[Holte 2000] Holte, H.: *Systematik zur Bewertung der Auswirkungen von Sicherheitseinrichtungen im Kraftfahrzeug (BASE)*. In: Bundesanstalt für Straßenwesen (Hrsg.): *Informations- und Assistenzsysteme im Auto benutzergerecht gestalten*. Bergisch Gladbach: Wirtschaftsverlag NW, 2000, S. 56-62

[Holzapfel 2000] Holzapfel, M.: *Konkatenative Sprachsynthese mit großen Datenbanken*. Dresden, Technische Universität Dresden, Institut für Akustik und Sprachkommunikation, Dissertation, 2000

[Hopcroft & Ullman 1990] Hopcroft, J.E. & Ullman, J.D.: *Einführung in die Automatentheorie, formale Sprachen und Komplexitätstheorie*. 2. Aufl. Bonn: Addison-Wesley, 1990

[ISO9241 1996] Norm ISO9241 Teil 10: 1996. Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Grundsätze der Dialoggestaltung

[ISO9241 1998] Norm ISO9241 Teil 11: 1998. Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Anforderungen an die Gebrauchstauglichkeit

[ISO/DIS15005 2000] Norm ISO/DIS15005: 2000. Straßenfahrzeuge - Ergonomische Aspekte von Fahrerinformations- und -assistenzsystemen - Grundsätze und Prüfverfahren des Dialogmanagements

[Jacob 1995] Jacob, R.J.K.: *Eye tracking in advanced interface design*. In: Barfield, W. & Furness, T. (Hrsg.): *Advanced Interface Design and Virtual Environments*. New York: Oxford University Press, 1995, S. 258-288

[Janssen 2000] Janssen, W.: *Driver distraction in the European statement of principles on in-vehicle HMI: a comment*. 2000. URL: http://www-nrd.nhtsa.dot.gov. Zugriff: 07.01.2002

[Johannsen 1993] Johannsen, G.: *Mensch-Maschine-Systeme*. Berlin: Springer, 1993

[Johnston*, et al.* 1997] Johnston, M. ; Cohen, P.R. ; McGee, D. ; Oviatt, S. ; Pittman, J.A. & Smith, I.: *Unification-based multimodal integration*. In: *Proceedings of 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, (1997), S. 281-288

[Jung & Hamberger 2000] Jung, R. & Hamberger, W.: *Individuell - Modular - Ganzheitlich: Anforderungen an ein modernes Bedienkonzept im Kraftfahrzeug*. In: *Elektronik im Kraftfahrzeug.* VDI-Berichte Nr. 1547. Düsseldorf: VDI-Verlag, 2000, S. 797-810

[Kirste & Rapp 2001] Kirste, T. & Rapp, S.: *Architecture for Multimodal Interactive Assistant Systems*. In: Wolf, G. & Klein, G. (Hrsg.): *Proceedings of International Status Conference: Lead Projects Human-Computer-Interaction*, (Saarbrücken, 2001), Projektträger des BMBF für Informationstechnik: Deutsches Zentrum für Luft- und Raumfahrttechnik (DLR) e.V., S. 111-115

[Kopf & Hermann 1997] Kopf, M. & Hermann, K.: *Visual Demands of an Aided vs. an Unaided Navigation Task in Real Traffic*. In: *Proceedings of 7th International Conference on Vision in Vehicles*, (Marseilles, 1997)

[Krahmer*, et al.* 1997] Krahmer, E. ; Landsbergen, J. & Pouteau, X.: *How to obey the 7 commandments for spoken dialogue?* In: Hirschberg, J. ; Kamm, C. & Walker, M. (Hrsg.): *Proceedings of the Workshop on Interactive Spoken Dialog Systems: Bringing Speech and NLP Together in Real Applications*, (Madrid, 1997), S. 82-89

[Kruglinski*, et al.* 1998] Kruglinski, D. ; Sheperd, G. & Wingo, S.: *Inside Visual C++*. 5. Aufl. Unterschleißheim: Microsoft Press, 1998

[Lang 1994] Lang, M.: *Mensch-Maschine-Kommunikation 1*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Kurzmanuskript zur Vorlesung, 1994

[Lang*, et al.* 1997] Lang, M. ; Zwickenpflug, R. & Mass, D.: *Mensch-Maschine-Kommunikation 2*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Ergänzungen und Übungen zur Vorlesung, 1997

[Lang 1998] Lang, M.: *Mensch-Maschine-Kommunikation 2*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Kurzmanuskript zur Vorlesung, 1998

[Lang 2002] Lang, M.: *Usability Engineering*. In: it+ti - Informationstechnik und Technische Informatik 44 (2002), Nr. 1, S. 3-4

[LaViola 1999] LaViola, J.J.J.: *Whole-Hand and Speech Input in Virtual Environments.* Providence, Brown University, Computer Science Department, Master's Thesis, 1999

[Lernout & Hauspie 1998] Lernout & Hauspie: *L&H ASR1500/ ASR1600 Software Developers' Kit*. Ieper: Lernout & Hauspie Speech Products, 1998

[Liu 2001] Liu, Y.-C.: *Comparative study of the effects of auditory, visual and multimodality displays on drivers' performance in advanced traveller information systems*. In: Ergonomics 44 (2001), Nr. 4, S. 425-442

[Macromedia 2000a] Macromedia: *Director 8 Shockwave Studio Benutzerhandbuch*. San Francisco: Macromedia, 2000a

[Macromedia 2000b] Macromedia: *Direxton Xtra Development Kit*. San Francisco: Macromedia, 2000b

[Marrenbach 2001] Marrenbach, J.R.: *Werkzeug-basierte Evaluierung der Benutzerfreundlichkeit interaktiver Endgeräte mit normativen Benutzermodellen*. Aachen, Rheinisch-Westfälische Technische Hochschule Aachen, Lehrstuhl für Technische Informatik, Dissertation, 2001

[Marrenbach*, et al.* 2001] Marrenbach, J.R. ; Kraiss, K.-F. ; Libuda, L. & Bengler, K.: *Development and Multimodal Operation of a Multimedia Car Instruction Manual*. In: *Proceedings of 8th IFAC/ IFIP/ IFORS/ IEA Symposium on Analysis, Design, and Evaluation of Human-Machine Systems*, (Kassel, 2001)

[Martin 1997] Martin, J.-C.: *Towards " intelligent " cooperation between modalities. The example of a system enabling multimodal interaction with a map*. In: *Proceedings of IJCAI-97 Workshop on Intelligent Multimodal Systems*, (Nagoya, 1997)

[Martin 1998] Martin, J.-C.: *TYCOON: Theoretical Framework and Software Tools for Multimodal Interfaces*. In: Lee, J. (Hrsg.): *Intelligence and Multimodality in Multimedia interfaces*. Menlo Park: AAAI Press, 1998

[Maybury & Wahlster 1998] Maybury, M.T. & Wahlster, W.: *Intelligent User Interfaces: An Introduction*. In: Maybury, M.T. & Wahlster, W. (Hrsg.): *Reading in Intelligent User Interfaces*. San Francisco: Morgan Kaufmann Publishers, 1998, S. 1-13

[Maybury & Stock 1999] Maybury, M.T. & Stock, O.: *Multimedia Communication, including Text*. In: Hovy, E. ; Ide, N. ; Frederking, R. ; Mariani, J. & Zampolli, A. (Hrsg.): *Multilingual Information Management: Current Levels and Future Abilities. A study commissioned by the US National Science Foundation and also delivered to European Commission Language Engineering Office and the US Defense Advanced Research Projects Agency*, 1999

[McGlaun*, et al.* 2002] McGlaun, G. ; Althoff, F. & Lang, M.: *Ein neuer Systemansatz für die Integration multimodalen Inputs durch Late Semantic Fusion*. In: *USEWARE 2002 Mensch-Maschine-Kommunikation/ Design.* VDI-Berichte Nr.1678. Düsseldorf: VDI-Verlag, 2002, S. 181-185

[Meister 1987] Meister, D.: *System Effectiveness Testing*. In: Salvendy, G. (Hrsg.): *Handbook of Human Factors*. New York: Johan Wiley & Sons, 1987

[Moran*, et al.* 1998] Moran, D.B. ; Cheyer, A.J. ; Julia, L.E. ; Martin, D.L. & Park, S.: *Multimodal User Interfaces in the Open Agent Architecture*. In: Knowledge-Based Systems 10 (1998), Nr. 5, S. 295-303

[Morguet 2000] Morguet, P.: *Stochastische Modellierung von Bildsequenzen zur Segmentierung und Erkennung dynamischer Gesten*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Dissertation, 2000

[Movellan & Mineiro 1996] Movellan, J.R. & Mineiro, P.: *Modularity and Catastrophic Fusion: A Bayesian Approach with Applications to Audiovisual Speech Recognition*. San Diego, University of California San Diego, Department of Cognitive Science, CogSci. UCSD Technical Report 97.01, 1996

[Mutschler 1995] Mutschler, H.: *Informationsdarstellung im Fahrzeug mit Hilfe eines Head-Up-Displays*, Bundesanstalt für Straßenwesen, Berichte der Bundesanstalt für Straßenwesen Heft F11, 1995

[Naab & Reichart 1994] Naab, K. & Reichart, G.: *Driver Assistance Systems for Lateral and Longitudinal Vehicle Guidance - Heading Control and Active Cruise Support -*. In: *Proceedings of AVEC´94: International Symposium on Advanced Vehicle Control*, (Tsukuba, 1994), S. 449-454

[Neal & Shapiro 1991] Neal, J.G. & Shapiro, S.C.: *Intelligent multi-media interface technology*. In: Sullivan, J.W. & Tyler, S.W. (Hrsg.): *Intelligent User Interfaces*. New York: ACM Press, 1991, S. 11-43

[Neuss 2000] Neuss, R.: *Usability Engineering als Ansatz zum Multimodalen Mensch-Maschine-Dialog*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Dissertation, 2000

[Niedermaier 1999] Niedermaier, B.: *Entwicklung und Bewertung eines nutzerorientierten Dialogkonzepts zur Sprachbedienung eines Autotelefons*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diplomarbeit, 1999

[Niedermaier & Lang 2001] Niedermaier, B. & Lang, M.: *Ein Ansatz zur kontextfreien Repräsentation multimodaler Dialoge unter Berücksichtigung der Dialogqualität*. In: Marzi, R. ; Karavezyris, V. ; Erbe, H.-H. & Timpe, K.-P. (Hrsg.): *Tagungsband der 4. Berliner Werkstatt Mensch-Maschine-Systeme, Berlin, 10.-12.10.2001. "Bedienen und Verstehen".* Fortschr.-Ber. VDI Reihe 22, Nr. 8. Düsseldorf: VDI-Verlag, 2001, S. 261-274

[Nielsen 1993] Nielsen, J.: *Usability Engineering*. Boston: Academic Press, 1993

[Nigay & Coutaz 1993] Nigay, L. & Coutaz, J.: *A design space for multimodal systems: Concurrent processing and Data fusion*. In: Ashlund, S. ; Mullet, K. ; Henderson, A. ; Hollnagel, E. & White, T. (Hrsg.): *Proceedings of InterCHI'93*, (Amsterdam, 1993), ACM Press, S. 172-178

[Nigay*, et al.* 1993] Nigay, L. ; Coutaz, J. & Salber, D.: *MATIS: A Multimodal Airline Travel Information System*. In. SM/WP10, ESPRIT BRA 7040 A-modeus, 1993

[Nirschl 1990] Nirschl, G.: *Verfahren zur integrierten Gestaltung und Bewertung von Mensch-Maschine-Dialogen im Kraftfahrzeug, basierend auf einem Entwicklermodell des Fahrerwissens*. Karlsruhe, Fraunhofer-Institut für Informations- und Datenverarbeitung, Dissertation, 1990

[Nirschl & Blum 2000] Nirschl, G. & Blum, E.J.: *MMI-Prüfliste - Verfahren und Werkzeug zur Bewertung von Mensch-Maschine-Systemen im Kraftfahrzeug*. In: Bundesanstalt für Straßenwesen (Hrsg.): *Informations- und Assistenzsysteme im Auto benutzergerecht gestalten*. Bergisch Gladbach: Wirtschaftsverlag NW, 2000, S. 42-49

[Norman 2001] Norman, D.A.: *Applying the behavioral, cognitive and social sciences to products*. 2001. Essay. URL: http://www.jnd.org. Zugriff: 07.01.2001

[Nuance 2001] Nuance: *SpeechObjects: An Architectural Overview*. 2001. White Paper. URL: http://www.nuance.com. Zugriff: 17.12.2001

[Oviatt 1996] Oviatt, S.: *Multimodal Interfaces for Dynamic Interactive Maps*. In: *Proceedings of CHI '96: Conference on Human Factors in Computing Systems*, (New York, 1996), ACM Press, S. 95-102

[Oviatt*, et al.* 1997] Oviatt, S. ; DeAngeli, A. & Kuhn, K.: *Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction*. In: *Proceedings of CHI'97: Conference on Human Factors in Computing Systems*, (Atlanta, 1997), ACM Press, S. 415-422

[Oviatt 1999] Oviatt, S.: *Ten Myths of Multimodal Interaction*. In: Communications of the ACM 42 (1999), Nr. 11, S. 74-81

[Parnow 2000] Parnow, A.: *Quality Gates der Ergonomie in der Fahrzeugentwicklung*. In: Bundesanstalt für Straßenwesen (Hrsg.): *Informations- und Assistenzsysteme im Auto benutzergerecht gestalten*. Bergisch Gladbach: Wirtschaftsverlag NW, 2000, S. 10-14

[Partmann*, et al.* 1995] Partmann, T. ; Reinig, H.-J. & Struck, G.: *Blickbewegungsmessung als Werkzeug für die Gestaltung und Bewertung von bord- und straßenseitigen Informationssystemen für den Kraftfahrer*. Karlsruhe, Fraunhofer-Institut für Informations- und Datenverarbeitung, Abschlußbericht zum Forschungsvorhaben der Bundesanstalt für Straßenwesen und der Forschungsvereinigung Automobiltechnik e.V., 1995

[Pinkal 2000] Pinkal, M.: *Semantikformalismen für die Sprachverarbeitung*. In: Görz, G. (Hrsg.): *Handbuch der künstlichen Intelligenz*. 3. Aufl. München: Oldenbourg, 2000, S. 739-782

[Polson*, et al.* 1992] Polson, P.G. ; Lewis, C. ; Rieman, J. & Wharton, C.: *Cognitive walkthroughs: A method for theory-based evaluation of user interfaces*. In: International Journal of Man-Machine Studies 36, Nr. 5, S. 741-773

[Ravden & Johnson 1989] Ravden, S.J. & Johnson, G.I.: *Evaluating Usability of Human-Computer Interfaces: A Practical Method*. Chichester: Ellis Horwood Ltd., 1989

[Reason 1990] Reason, J.T.: *Human Error*. Cambridge: Cambridge University Press, 1990

[Rosnitschek 2000] Rosnitschek, M.: *Multimodale Mensch-Maschine-Kommunikation im Fahrzeug*. München, Technische Universität München, Lehrstuhl für Mensch-Maschine-Kommunikation, Diplomarbeit, 2000

[Ruske 1994] Ruske, G.: *Automatische Spracherkennung: Methoden der Klassifikation und Merkmalsextraktion*. 2. Aufl. München: Oldenbourg Verlag, 1994

[Salmen*, et al.* 1999] Salmen, A. ; Großmann, P. ; Hitzenberger, L. & Creutzburg, U.: *Dialog Systems in Traffic Environment*. In: *Proceedings of ESCA: Tutorial and Research Workshop on Interactive Dialogue in Multi-Modal Systems*, (Kloster Irsee, 1999)

[Schattenberg & Debus 2001] Schattenberg, K. & Debus, G.: *Multimodale Anzeige- und Bedienkonzepte zur Steuerung technischer Systeme während der Fahrt im Kraftfahrzeug: Evaluationsbefunde zur Systemweiterentwicklung mit paralleler Sprachbedienung*. In: Jürgensohn, T. & Timpe, K.-P. (Hrsg.): *Kraftfahrzeugführung*. Berlin: Springer, 2001, S. 177-193

[Schmidtke 1981] Schmidtke, H.: *Lehrbuch der Ergonomie*. 2. Aufl. Wien: Hanser, 1981

[Schraut*, et al.* 2000] Schraut, M. ; Naab, K. & Bachmann, T.: *BMW's Driver Assistance Concept for Integrated Longitudinal Support. Paper N°. 2121*. In: *Proceedings of ITS'00: 7th Intelligent Transport Systems World Congress*, (Turin, 2000)

[Schrievers & Haller 1994] Schrievers, G. & Haller, R.: *Evaluation of Man Machine Interfaces by Simulation Techniques*, Commision of the European Communities, DRIVE II Project V2006 WP C2, 1994

[Schumacher*, et al.* 1995] Schumacher, R.M. ; Hardzinski, M.L. & Schwartz, A.L.: *Increasing the usability of interactive voice response systems: Research and guidelines for phone-based interfaces*. In: Human Factors 38 (Juni 1995), Nr. 2, S. 251-264

[Seeboerger-Weichselbaum 2000] Seeboerger-Weichselbaum, M.: *Das Einsteigerseminar XML*. 2. Aufl. Kaarst: bhv-Verlag, 2000

[Seifert*, et al.* 2001] Seifert, K. ; Rötting, M. & Jung, R.: *Registrierung von Blickbewegungen im Fahrzeug*. In: Jürgensohn, T. & Timpe, K.-P. (Hrsg.): *Kraftfahrzeugführung*. Berlin: Springer, 2001, S. 207-228

[Siemens 1998] Siemens: *SIVIT - Siemens Virtual Touchscreen*. Erlangen: Siemens AG, 1998

[Spona & Radke 1999] Spona, H. & Radke, H.-D.: *Das große Buch Access 2000*. Düsseldorf: Data Becker, 1999

[Sproat & Olive 1995] Sproat, R. & Olive, J.P.: *An Approach to Text-to-Speech Synthesis*. In: Kleijn, W.B. & Paliwal, K.K. (Hrsg.): *Speech Coding and Synthesis*. Amsterdam: Elsevier, 1995, S. 611-633

[Steudel 1999] Steudel, S.: *Optomotorische Dialogsteuerung in Fahrumgebungen*. Dresden, Technische Universität Dresden, Institut für Informationssysteme, Diplomarbeit, 1999

[Streit 1999] Streit, M.: *The Interaction of Speech, Deixis and Graphics in the Multimodal Office Agent Talky*. In: *Proceedings of ESCA: Tutorial and Research Workshop on Interactive Dialogue in Multi-Modal Systems*, (Kloster Irsee, 1999)

[Stroustrup 1998] Stroustrup, B.: *Die C++ Programmiersprache*. 3. Aufl. Bonn: Addison-Wesley, 1998

[Timpe & Kolrep 2000] Timpe, K.-P. & Kolrep, H.: *Das Mensch-Maschine-System als interdisziplinärer Gegenstand*. In: Timpe, K.-P. ; Jürgensohn, T. & Kolrep, H. (Hrsg.): *Mensch-Maschine-Systemtechnik*. Düsseldorf: Symposion Publishing, 2000, S. 9-40

[Vector 2001] Vector: *CANalyzer Arbeitshandbuch*. Stuttgart: Vector Informatik, 2001

[Vector 2002] Vector: *CANoe Handbuch*. Stuttgart: Vector Informatik, 2002

[Vo & Waibel 1993] Vo, M.T. & Waibel, A.: *Multimodal Human-Computer Interaction*. In: *Proceedings of ISSD'93: International Symposium on Spoken Dialogue: New Directions in Human and ManMachine Communication*, (Tokyo, 1993), S. 95-101

[Vo 1998] Vo, M.T.: *A Framework and Toolkit for the Construction of Multimodal Learning Interfaces*. Pittsburgh, Carnegie Mellon University, Computer Science Department, Dissertation, 1998

[W3C 2001] W3C: *Voice Extensible Markup Language (VoiceXML) Version 2.0 - Working Draft*. 2001. URL: http://www.w3.org/TR/voicexml20/. Zugriff: 07.01.2002

[W3C 2002] W3C: *The World Wide Web Consortium*. 2002. URL: www.w3.org. Zugriff: 07.01.2002

[Wagner*, et al.* 2001] Wagner, U. ; Angermüller, H. ; Amann, R. ; Grimm, P. ; Hennig, T. & Scholze, M.: *Die Instrumentenkombination im neuen 7er BMW: Anzeigephilosophie und Technik*. In: *Elektronik im Kraftfahrzeug.* VDI-Berichte Nr.1646. Düsseldorf: VDI-Verlag, 2001, S. 941-967

[Wahlster*, et al.* 2001] Wahlster, W. ; Reithinger, N. & Blocher, A.: *SmartKom: Towards Multimodal Dialogues with Anthropormophic Interface Agents*. In: Wolf, G. & Klein, G. (Hrsg.): *Proceedings of International Status Conference: Lead Projects Human-Computer-Interaction*, (Saarbrücken, 2001), Projektträger des BMBF für Informationstechnik: Deutsches Zentrum für Luft- und Raumfahrttechnik (DLR) e.V., S. 23-32

[Waibel*, et al.* 1995] Waibel, A. ; Vo, M.T. ; Duchnowski, P. & Manke, S.: *Multimodal Interfaces*. In: Artificial Intelligence Review 10 (1995), Nr. 3/4

[Winkler & Lang 1997] Winkler, H.-J. & Lang, M.: *On-line Symbol Segmentation and Recognition in Handwritten Mathematical Expressions*. In: *Proceedings of ICASSP'97: IEEE International Conference on Acoustics, Speech, and Signal Processing*, (München, 1997), S. 3377-3380

[Yankelovich 1996] Yankelovich, N.: *How do Users Know What to Say?* In: ACM Interactions 3 (11./12. 1996), Nr. 6

[Zeller*, et al.* 2001] Zeller, A. ; Wagner, A. & Spreng, M.: *iDrive - Zentrale Bedienung im neuen 7er von BMW*. In: *Elektronik im Kraftfahrzeug.* VDI-Berichte Nr.1646. Düsseldorf: VDI-Verlag, 2001, S. 997-1009

[Zobl*, et al.* 2001] Zobl, M. ; Geiger, M. ; Bengler, K. & Lang, M.: *A Usability Study on Hand Gesture Controlled Operation of In-Car Devices*. In: *Proceedings of HCII 2001: 9th International Conference on Human Computer Interaction*, (New Orleans, 2001), S. 166-168