Automatische kriterienorientierte Bewertung der Gebrauchstauglichkeit interaktiver Systeme

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Informatiker
Nicolas Alexander Hamacher
aus Rheinbach

Berichter: Universitätsprofessor Dr.-Ing. Karl-Friedrich Kraiss
Universitätsprofessor Dr. rer. nat. Jan Oliver Borchers

Tag der mündlichen Prüfung: 21. Juli 2006

Bibliografische Information der Deutschen Nationalbibliothek
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über **http://dnb.d-nb.de** abrufbar.

ISBN 3-89963-432-2

Bei der Abbildung auf dem Umschlag handelt es sich um ein ca. 2100 Jahre altes Wandgemälde aus einer römischen Villa im Norden Siziliens. Das technische Gerät ist nachträglich in das Bild eingefügt worden, allerdings wäre der Gesichtsausdruck des Herrn angesichts des verheerenden Interaktionsdesigns sicherlich unverändert.

Sternstr. 18, 80538 München
Tel.: 089/66060798
www.dr.hut-verlag.de

Die Informationen in diesem Buch wurden mit großer Sorgfalt erarbeitet. Dennoch können Fehler, insbesondere bei der Beschreibung des Gefahrenpotentials von Versuchen, nicht vollständig ausgeschlossen werden. Verlag, Autoren und ggf. Übersetzer übernehmen keine juristische Verantwortung oder irgendeine Haftung für eventuell verbliebene fehlerhafte Angaben und deren Folgen.

1. Auflage 2006

Druck und Bindung: mvr-druck, Brühl (www.mvr-druck.de)

Für die Familie!

Vorwort

Diese Arbeit ist im Rahmen meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Technische Informatik der Rheinisch-Westfälischen Technischen Hochschule Aachen entstanden. An dieser Stelle möchte ich denjenigen Personen danken, die mich in dieser schönen und ereignisreichen Zeit begleitet und mich bei meiner Arbeit unterstützt haben.

Allen voran gilt mein Dank Herrn Prof. Dr.-Ing. Karl-Friedrich Kraiss, dem Inhaber des Lehrstuhls, für die Betreuung meiner Arbeit, seine Unterstützung und für die Möglichkeit, in großer Freiheit interessante Themen zu verfolgen und erforschen zu können. Weiterhin möchte ich Herrn Prof. Dr. rer. nat. Jan Oliver Borchers für die Übernahme des zweiten Gutachtens, sowie für die vielen Tipps und Anregungen danken.

Ich habe Kommunikation und Diskussion als wesentlichen Teil der Forschung kennen gelernt. In diesem Sinne danke ich vor allem meinen Freunden Michael Hähnel und Jörg Zieren, die meine Arbeit durch viele wissenschaftliche Diskussionen, fortwährende Unterstützung und viele Ratschläge bereichert haben.

Bei der Realisierung des in dieser Arbeit entwickelten Werkzeugs wurde ich ebenfalls umfangreich unterstützt. Großer Dank gebührt Mathias Funk, der mir bei meiner Arbeit mit vielen erfrischenden Ideen und zuverlässigen Implementierungen half, sowie Christian Hönig, der Funktionen, die technisch eigentlich nicht machbar sind, dennoch äußerst elegant realisierte. Beiden möchte ich noch einmal besonders für die große Zuverlässigkeit und die angenehme (und stressresistente) langjährige Zusammenarbeit danken.

Weiterhin danke ich Thomas Duif für seine große Hilfe sprichwörtlich „an allen Ecken und Enden“! Bei der Erstellung und Durchführung der Bewertungen war Jörg Kurscheid maßgeblich beteiligt. Ihm danke ich sehr für sein enormes Engagement und die Mammutarbeit bei der Durchführung der empirischen Bewertung und der Auswertung der Ergebnisse.

Für den Dank an meine Freundin Sandra und meine Familie fehlen mir schlicht die Worte. Gut, dass es dafür auch keiner Worte bedarf!

Aachen, im Oktober 2006 Nico Hamacher

Inhaltsverzeichnis

Symbolverzeichnis

Allgemeine Notation

y	skalarer Wert
E	Erwartungswert
σ	Standardabweichung
$CIE(Y,u,v)$	Farbwert (CIE Yuv-Farbraum)
	Leuchtdichte: Y
	Chrominanz:
	u (Differenz zw. Blauanteil und Leuchtdichte)
	v (Differenz zw. Rotanteil und Leuchtdichte)
$\|M\|$	Kardinalität der Menge M

Verwendete Symbole

Dlg	Menge aller Oberflächenelemente der GUI-Konstellation eines Dialogs
dlg_i	Oberflächenelement einer GUI-Konstellation Dlg, $dlg_i \in Dlg$
D	Menge aller Displayelemente (Elemente, die auf Displays dargestellt sind)
d_i	Displayelement (auf einem Display dargestelltes Oberflächenelement)
	$d_i \in D$

Attribute eines Oberflächenelements e

$e^{[a]}$	Attribut a von Oberflächenelement e
$e^{[x/y/z]}$	x-, y- bzw. z-Position der linken oberen Ecke
	bei Kreisen: Angabe des Mittelpunktes
$e^{[h/w]}$	Höhe (height) bzw. Breite (width) – entfällt bei Kreisen
$e^{[r]}$	Radius (bei Kreisen)
$e^{[x\text{-}/y\text{-}angle]}$	Kippwinkel um die x- bzw. y-Achse
$e^{[lev]}$	grafische Ebene (level)
$e^{[col]}$	Farbe (color)

$t^{[con]}$	textueller Inhalt (content) eines Textelementes t
$t^{[fh]}$	Schrifthöhe (font height)
$t^{[ff]}$	Schriftfamilie (font familie)
$t^{[fs]}$	Schriftsstil (font style)

Bezeichnungen der verwendeten Maße

$M(Dlg)$	Menge von Maßen eines Dialogs Dlg
M	Menge von Maßen über mehrere GUI-Konstellationen
m_i	Wert eines Maßes für $m_i \in M(K)$ bzw. Werte eines Maßes für $m_i \in M$
$G(Dlg)$	*Gridedness* eines Dialogs Dlg
$B_{h/v}(Dlg)$	horizontale/vertikale *Balance* der Oberflächenelemente eines Dialogs Dlg
$F(Dlg)$	Schriftgrößen der Textelemente eines Dialogs Dlg
$WD(Dlg)$	*Widget Density* eines Dialogs Dlg
$Ma_{l/r/t/b}(Dlg)$	Abstandsmaß zum linken, rechten, oberen bzw. unteren Rand eines Dialogs Dlg (*left, right, top, bottom margin*)
ΔE	*Farbabstand* zweier Farbwerte im CIE-Yuv-Farbraum
K	*Kontrastverhältnis* zweier Farbwerte im CIE-Yuv-Farbraum
DAR	*Display Aspect Ratio*
MAR	*Menu Aspect Ratio*
ME	Menge der Menüebenen eines Systemmenüs (für eine Aufgabe)
me_i	Menüebene i mit der Anzahl der Menüoptionen, $me_i \in ME$
VM	Menge der gleichzeitig sichtbaren (visible) Menüs
SMO	Selektierte Menüoption
M_{SMO}	Menge der in einem Menü selektierten Menüoptionen
AC	Liste der Bedienaktionen (action) zur Bearbeitung einer Aufgabe
CE	Menge der benutzten Bedienelemente (control elements) zur Bearbeitung einer Aufgabe
St	Menge der benutzten Stereotypen zur Bedienung einer Aufgabe
$N_{col_{F/B}}(Dlg)$	Anzahl der Vordergrund- (Front) bzw. Hintergrundfarben (Back) eines Dialogs Dlg
$N_{col_{text}}$	Anzahl der Farben aller Textelemente
N_{header}	Anzahl der Überschriften
$N_D(Dlg)$	Anzahl der Displayelemente eines Dialogs Dlg
N_{ME}	Anzahl der Menüebenen eines Systemmenüs (für eine Aufgabe)
N_{VM}	Anzahl der gleichzeitig sichtbaren Menüs
$N_{M_{SMO}}$	Anzahl der selektierten Menüoptionen
N_{AC}	Anzahl der Bedienaktionen
N_{CE}	Anzahl der benutzten Bedienelemente
N_{St}	Anzahl der benutzten Stereotypen

Kapitel 1

Einleitung

Bei der Entwicklung neuer interaktiver Systeme spielt die ergonomische und gebrauchstaugliche Gestaltung der Bedienoberfläche[1] eine immer wichtigere Rolle. So können z.B. gebrauchstaugliche Bedienoberflächen sicherheitskritischer Anwendungen Fehlbedienungen vermeiden. Zudem benutzen immer mehr Firmen die gute Gebrauchstauglichkeit ihrer Produkte als Werbe- und Verkaufsargument[2].

Bei Mobiltelefonen und PDAs ist seit Ende der neunziger Jahre die Handhabbarkeit für die Verbraucher genauso wichtig geworden wie die Funktionalität (BENYON et al. 2005); daher lassen sie die Produkte, mit denen sie nicht zufrieden sind, in immer größerem Maße unberücksichtigt (JONES und SASSER 1995; SHNEIDERMAN und PLAISANT 2005).

Dabei hat sich alleine die Verbreitung von Mobiltelefonen von 1998 bis 2004 mehr als versechsfacht (von 11,2% auf 72,1% der deutschen Haushalte), wobei die Anzahl der Geräte von 4,5 Mio auf 43,5 Mio auf das nahezu 10fache stieg (STATISTISCHES BUNDESAMT 2004). Eine weitere Umfrage ergab, dass eine einfache Bedienung mit 66% aller Nennungen an erster Stelle der Eigenschaften steht, auf die beim Kauf von Mobiltelefonen geachtet wird (ALLENSBACHER BERICHTE 2002). Für Unternehmen kann bereits eine Investitionssteigerung um 5% in die Verbesserung der Gebrauchstauglichkeit den Verkaufserlös um bis zu 35% steigern (BLACK 2002).

Die Faktoren, die einer einfachen Benutzbarkeit interaktiver Geräte entgegen stehen, sind vielschichtig und werden auszugsweise im Folgenden erläutert. Ein wesentlicher Aspekt ist die stetige Zunahme an Funktionalität, ohne dass die Bedienoberfläche mit Anzeigen und Eingabeelementen in adäquatem Maße wächst (DAHM et al. 2005). Das hat u.A. zur Folge, dass der Zugriff auf die Funktionen durch den Einsatz von Menüs realisiert wird, so dass zusätzlich zur reinen Aktivierung von Funktionen die Menübedienung hinzu kommt. Zudem wird

[1] s. Glossar

[2] Aktuell existieren nur wenige genormte Prüfverfahren, die *Usability* oder *Ergonomie* objektiv messen. Dazu gehören z.B. das ITG- oder DATech-Prüfverfahren (VDE/ITG 1995; DATECH 2004). Daneben treten jedoch vermehrt unseriöse Werbesiegel als Marketingmittel auf, wie z.B. das Siegel „100% Usability“ unter http://www.webtranslate.de /HTML /translate_version8.htm

vermehrt domänenfremde Funktionalität miteinander vermischt. So kann heutzutage nahezu jedes Mobiltelefon Termine verwalten oder Emails empfangen, während moderne Fahrerinformationssysteme (FIS) neben der Straßennavigation auch auf Adressbücher eines Mobiltelefons zugreifen. Hinzu kommt, dass jeder Hersteller sich durch eine eigene *Bedienphilosophie* von Geräten der Konkurrenz abzuheben versucht. Dadurch wird der Umstieg auf Geräte von anderen Herstellern für den Bediener zusätzlich erschwert (GEISER 1998).

Die Bedienung von Geräten des täglichen Lebens (z.B. Mobiltelefone oder Digitalkameras) bewerten 57% der Benutzer als zu kompliziert, bei 33% der Benutzer (bzw. 40% der älteren Benutzer) kommt es aufgrund der Komplexität technischer Produkte sogar zu einer *Kaufblockade* (RESEARCH INTERNATIONAL DEUTSCHLAND 2004a; RESEARCH INTERNATIONAL DEUTSCHLAND 2004b). In Amerika wird von den Angestellten jedes dritte Büro-Computersystem als untauglich abgelehnt. Bei Einführung eines neuen Systems steigt der Krankenstand um 300% (TUMA 1997). In Deutschland entsteht aufgrund fehlerhafter Bedienung jährlich ein wirtschaftlicher Schaden von ca. einer Milliarde Euro während dieser Schaden in den USA bereits auf ca. 30 Milliarden Dollar geschätzt wird (NORMAN und NIELSEN 2003).

Fehler im Produktdesign müssen möglichst früh im Entwicklungsprozess entdeckt und behoben werden, da spätere Korrekturen nur mit deutlich höherem Aufwand durchgeführt werden können (NIELSEN 2000; HOLZINGER 2005). So lässt sich durch frühe Tests der Benutzerfreundlichkeit die Entwicklungszeit um bis zu 40% verkürzen (MAYHEW 1999). Laut einer Studie des Fraunhofer-Instituts können die Kosten für Nachbesserungen am fertigen Produkt bis um das Zehntausendfache höher sein als zu Beginn des Entwicklungsprozesses (FREISE 2003).

Diese Tatsachen unterstreichen die Notwendigkeit, die Gebrauchstauglichkeit interaktiver Systeme zu untersuchen und zu steigern. Im Folgenden wird der Prozess der Systementwicklung sowie die Möglichkeiten der Bewertung der Gebrauchstauglichkeit interaktiver Systeme beschrieben.

1.1 Entwicklung und Bewertung von interaktiven Systemen

Die einzelnen Phasen der System- und Softwareentwicklung werden mit Hilfe von Phasenmodellen dargestellt. Die Phasen entsprechen jeweils einem Arbeitspaket mit definierten Arbeitsschritten und Teilergebnissen. Das Ergebnis jeder Phase bietet die Grundlage der jeweils darauf folgenden Phase (KAINDL et al. 1998). Durch die getrennte Darstellung dieser Phasen wird ein größeres Verständnis ihrer selbst und ihrer Interaktion möglich.

Ein phasenorientierter Ansatz stellt eine systematische Darstellung des idealtypischen Ablaufs einer Systementwicklung dar und beschreibt nicht zwangsläufig den realen Prozessablauf. Die Überführung des Modells in ein entsprechend der konkreten Entwicklungspraxis angepasstes *Working Model* ist jeweils erforderlich (STEINWEG 2005).

Abbildung 1.1 zeigt ein einfaches Phasenmodell [3] (KRAISS 1995). Die Definitionsphase dient der Definition und Erarbeitung von Leistungsumfang sowie weiteren Anforderungen des zu entwickelnden Systems auf Grundlage einer Anforderungsanalyse. Auf Basis der festgelegten Definitionen des Systems wird in der Spezifikationsphase die Funktionalität mit allen nötigen (Teil-)Systemen bis ins Detail festgelegt. Eine solche Spezifikation erfolgt heutzutage meist bereits mit Hilfe von Softwarewerkzeugen, die eine elektronische Weiterverarbeitung der Spezifikation erlauben. Das Ziel der Realisierung ist die technische Umsetzung des in der Spezifikation festgelegten Systems. Die realisierten (Teil-)Systeme werden in der Integrationsphase zum geforderten Gesamtsystem zusammen gefügt. Die Nutzungsphase stellt die ordnungsgemäße Anwendung des Systems dar (STEINWEG 2005).

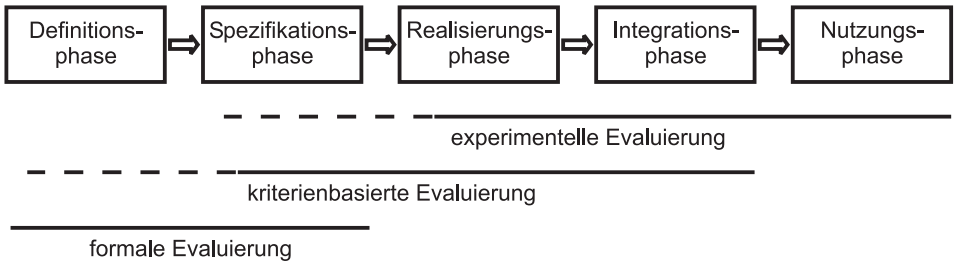

Abbildung 1.1: Phasen der Systementwicklung und Einsatz verschiedener Evaluierungsmethoden (KRAISS 1995).

Ein interaktives System lässt sich nach dem Seeheim-Modell als logisches Architekturmodell[4], wie in Abbildung 1.2 dargestellt, in folgende Komponenten unterteilen (PFAFF 1983):

- Die *Präsentationskomponente*[5] dient der Informationsein- und -ausgabe. Ihre Aufgabe besteht in der Bildschirmverwaltung, der Verwaltung der physikalischen Eingabegeräte, der textuellen und grafischen Ausgabe sowie der Realisierung verschiedener Interaktionstechniken (FOLEY et al. 1984). Sie enthält die dem Benutzer sichtbaren und zugänglichen Komponenten einer Benutzungsschnittstelle.

- Die *Dialogkontrolle* definiert und überwacht die korrekte Abfolge von Dialogschritten, die durch Interaktionstechniken der Präsentationskomponente realisiert werden, und steuert dadurch das dynamische Verhalten einer Benutzungsschnittstelle. Zu ihren Aufgaben gehört u.a. die Überprüfung der syntaktischen Korrektheit ganzer Folgen von Benutzereingaben, Fehlerbehandlung und die Umwandlung von Ausgabeoperationen

[3] Dieses Modell ordnet der Übersichtlichkeit halber die einzelnen Phasen sequentiell ohne Rückkopplungen an. Eine größere Realitätsnähe bieten Modelle (z.B. das V-Modell), die die Rückkopplung von Informationen aus einzelnen Phasen in vorhergehenden Phasen abbilden. Weitere Modelle (z.B. das Spiralmodell) berücksichtigen die Tatsache, dass häufig die einzelnen Entwicklungsphasen zyklisch während der Systementwicklung durchlaufen werden. Eine detaillierte Übersicht unterschiedlicher Phasenmodelle findet sich in (KAINDL et al. 1998).

[4] Der Begriff *logisches* Architekturmodell besagt, dass jede Benutzungsschnittstelle die Funktionalität der drei Komponenten des Seeheim-Modells besitzen sollte, ohne jedoch eine dieser Aufteilung entsprechende monolithische Software-Architektur besitzen zu müssen.

[5] Häufig wird auch der Begriff *User Interface (UI)* oder *Graphical User Interface (GUI)* benutzt, s. Glossar.

der Anwendungsschnittstelle in mehrere Aussagetoken[6], die an die Präsentationskomponente weitergeleitet werden (GÖTZE 1994).

- Die *Anwendungsschnittstelle* beschreibt die Systemfunktionalität aus der Sicht der Benutzungsschnittstelle. Sie definiert die für die Benutzungsschnittstelle zugreifbaren Datenstrukturen und Funktionen der Anwendung sowie die Struktur der Informationen, die zwischen der Benutzungsschnittstelle und der Anwendung ausgetauscht werden.

Diese Struktur gilt heutzutage als Grundlage der meisten dialoggesteuerten Systeme. Eine Bewertung der Gebrauchstauglichkeit umfasst die Präsentationskomponente und die Dialogkontrolle, da sie direkten Einfluss auf das Erscheinungsbild des Gerätes für den Benutzer haben.

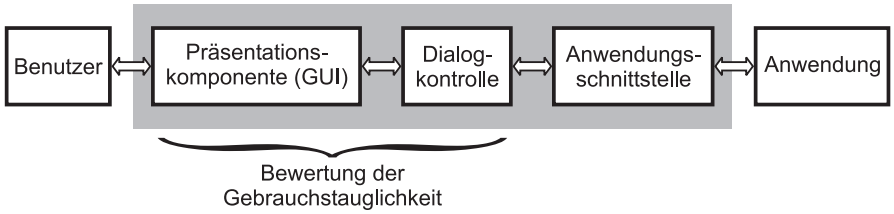

Abbildung 1.2: Komponenten eines Dialogsystems (grau hinterlegt) nach dem Seeheim-Modell (PFAFF 1983). Eine Bewertung der Gebrauchstauglichkeit betrachtet die GUI sowie die Dialogkontrolle.

Teil 11 als Ausmaß, in dem ein Produkt durch bestimmte Benutzer in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und zufriedenstellend zu erreichen (ISO 9241-11 1998). Das Kriterium der *Effektivität* ist dabei die Genauigkeit und Vollständigkeit, mit der Benutzer ein bestimmtes Ziel erreichen. Die *Effizienz* wird als der im Verhältnis zur Genauigkeit und Vollständigkeit eingesetzte Aufwand definiert. Das Kriterium *Zufriedenheit* beschreibt aus der subjektiven Benutzersicht die Freiheit von Beeinträchtigung sowie die positive Einstellung gegenüber der Nutzung des Produkts. Geeignete Maße geben jeweils das Ausmaß des betrachteten Kriteriums zur Erreichung des Ziels an. So sind z.B. die Anzahl von Fehlern oder gelernter Funktionen bei der Aufgabenbearbeitung geeignete Maße zur Bewertung der Effektivität. Ausführungs- und Lernzeiten stellen Maße der Effizienz dar. Die Zufriedenheit lässt sich durch ein Rating der Benutzerakzeptanz oder die Häufigkeit der Nutzung messen (MARRENBACH 2001; ISO 9241-11 1998).

Über die Kriterien hinaus definiert die Norm Grundsätze, die bei der Entwicklung interaktiver Systeme Berücksichtigung finden sowie bei einer Bewertung überprüft werden müssen. Als Grundsätze z.B. zur Gestaltung interaktiver Systeme sind im Teil 110 der Norm[7] die Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit sowie die Lernförderlichkeit benannt (ISO 9241-110 2004), während in Teil 12 zur Darstellung von Informationen die Grundsätze Erkennbarkeit,

[6] Ein *Token* bezeichnet eine Informationseinheit; s. Glossar.

[7] Der Teil 110 der ISO 9241 stellt den Entwurf einer Erweiterung für den Teil 10 dar (ISO 9241-10 1996).

Unterscheidbarkeit, Lesbarkeit, Verständlichkeit, Klarheit, Kompaktheit und die Konsistenz definiert werden (ISO 9241-12 1998).

Zur Bewertung der Gebrauchstauglichkeit existieren zahlreiche Methoden, die in Art und Weise der Anwendung sowie in ihren Ergebnissen sehr unterschiedlich sein können. Grundsätzlich lassen sich diese Methoden in drei Kategorien[8] unterteilen, die in unterschiedlichen Phasen der Systementwicklung Anwendung finden und in Abbildung 1.1 dargestellt sind (BURNS 2004):

- *Formale (analytische)* Methoden basieren meist auf normativen Modellen des Benutzers bzw. dessen Verhalten[9]. Eine Analyse dieser Modelle kann z.B. Abschätzungen der Ausführungs- und Lernzeit bei der Bearbeitung einer bestimmten Aufgabe liefern und lässt sich automatisch durchführen. Sofern eine Systemspezifikation elektronisch vorliegt, ist sogar die Erstellung der Benutzermodelle teilweise automatisch möglich (HAMACHER et al. 2002). Wenngleich diese Methoden bereits sehr früh in der Systementwicklung angewandt werden können, ist ihre Aussagekraft jedoch auf die Generierung weniger Bewertungsmaße der Effektivität oder Effizienz begrenzt (HAMACHER und KRAISS 2004).

- *Kriterienorientierte* Methoden[10] fassen Erfahrungswissen von Experten in Form von Guidelines zusammen. Experten bewerten meist intuitiv, was jedoch die Ergebnisse schwer nachvollziehbar macht. Aus diesem Grund ist der Einsatz von Guidelines wichtig, um dieses Wissen auch für andere zugängig und die Kriterien möglichst objektiv fassbar zu machen (POSCH et al. 2004).

  Guidelines differieren im Fokus der Bewertung, im Umfang und in der Anwendung sehr stark. Sie reichen von der allgemeinen Definition der Gebrauchstauglichkeit (z.B. ISO-Norm 9241, s.o.) bis hin zu detailliert festgelegten Gestaltungsregeln einer Bildschirmoberfläche (z.B. die Gestaltungsrichtlinien des OS-X von Apple oder R/3 von SAP (APPLE 2005; SAP 2004)). Solche Guidelines existieren nahezu in allen Phasen der Systementwicklung für Maße der Effektivität und Effizienz und heuristisch auch vereinzelt zur Bewertung hinsichtlich der Benutzerzufriedenheit (HENNINGER 2001).

  Da Richtlinien und Kriterien üblicherweise umgangssprachlich in Form von Dokumenten vorliegen, werden sie nach wie vor meist von Hand ausgewertet, wobei die Interpretation häufig schwierig ist (FARENC und PALANQUE 1999). Automatisierungsansätze entstehen erst in jüngerer Zeit (REITERER 2001).

[8] Eine klassische Aufteilung von Whitefield et al. teilt die Methoden in vier Klassen auf, jeweils zwei für reale Benutzer (Beobachtung der Benutzer, Befragung der Benutzer) und für repräsentative Benutzer (analytische Methoden, Bewertung durch Experten) (WHITEFIELD et al. 1991). Die Klassen, welche reale Benutzer einbeziehen, basieren auf Experimenten, weshalb sie in dieser Arbeit, ebenso wie bei Burns in (BURNS 2004), zu *experimentellen Methoden* zusammengefasst wurden. Weiterhin impliziert die Bewertung durch Experten den Einsatz von Expertenwissen in Form von Kriterienkatalogen.

[9] Eine der bekanntesten formalen Methoden ist die GOMS-Methode, die die Modellierung des normativen Benutzerverhalten erlaubt, s. Glossar

[10] Burns spricht in diesem Zusammenhang von *Guidelines* (BURNS 2004). Im Folgenden werden die Begriffe *Kriterienkataloge* und *Guidelines* synonym verwendet.

- Mit *experimentellen* Methoden wird ein Prototyp des Systems von Probanden bedient und anschließend bewertet. Eine Bewertung stützt sich einerseits auf objektiv erhobene Daten. Das sind z.B. Protokolle der Handlungen der Probanden, Messungen der Augenbewegungen (Eye-Tracking-Verfahren) oder Videoaufzeichnungen. Andererseits werden subjektive Aussagen der Probanden mit Hilfe von Interviews oder Fragebögen erfasst. Diese Bewertungsmethoden sind sehr zeit- und kostenintensiv und benötigen zudem einen lauffähigen Prototypen, der erst in späteren Phasen der Systementwicklung zur Verfügung steht. Allerdings sind experimentelle Methoden die einzigen, durch die neben der Effektivität und Effizienz auch die Zufriedenheit hinlänglich bewertet werden kann.

Während formale Methoden nur selten zum Einsatz kommen, gehören kriterienorientierte und experimentelle Methoden zum Standard einer Bewertung der Gebrauchstauglichkeit. Mit zunehmendem Umfang der Systemfunktionalität sowie der Vereinheitlichung der Bedienung und des User-Interfaces nimmt ebenfalls der Grad der Standardisierung zu, wodurch die Guidelines immer umfangreicher werden. Ebenso existieren Guidelines für die Bewertung unterschiedlicher Aspekte eines Systems. So werden neben reinen Gestaltungsregeln die einheitliche Menüführung sowie die Darstellung und der Inhalt von Texten erfasst. Dadurch steigen die Anforderungen an den Entwicklungsingenieur, da alle diese Guidelines bei der Systementwicklung berücksichtigt werden müssen. Eine groß angelegte Untersuchung zeigte, dass sich 96,1% der deutschsprachigen Webseiten nicht an die essenziellen W3C-Standards[11] halten (BLEICH 2005).

Weiterhin sind Guidelines wie z.B. die ISO-Norm 9241 meist allgemein gehalten und somit nicht direkt für ein konkretes Problem anwendbar. Bei der Anwendung existiert daher ein mehr oder weniger großer Interpretationsspielraum. Aufgrund dieser Tatsache werden Kriterien bzw. Guidelines nicht immer richtig angewendet oder gar nicht erst berücksichtigt.

Souza und Bevan untersuchten, inwieweit Experten in der Lage sind, gegebene Guidelines beim Design eines Menüsystems zu berücksichtigen. Dabei machten die Experten Fehler oder hatten Probleme mit 91% der gegebenen Guidelines. Die durch die Experten entwickelten Menüsysteme verstießen durchschnittlich gegen 11% der gegebenen Regeln (SOUZA und BEVAN 1990). Das liegt vor allem daran, dass Experten unterschiedliche Sichtweisen bei der Auslegung von Kriterien bzw. je nach persönlicher Erfahrung einen unterschiedlichen Fokus bei der Bewertung haben (BLUME et al. 2005)

Ausgehend von den beschriebenen Lebensphasen und Bewertungsmethoden wird nachfolgend erläutert, auf welcher Datengrundlage eine kriterienorientierte Bewertung der Benutzungsfreundlichkeit stattfinden kann. Zur Veranschaulichung erfolgt die Erklärung am Bei-

[11] Die Standards des Word Wide Web Consortium (W3C) behandeln das World Wide Web betreffende Techniken, vor allem die Programmiersprachen zur Realisierung von Webseiten. Ziel der Standards ist eine gleichförmige Darstellung und Bedienung der Seiten auf unterschiedlichen Browsern und Plattformen. Darüber hinaus ermöglichen sie eine größtmögliche Barrierefreiheit, um behinderten Benutzern den Zugang zu den Webseiten zu erleichtern. Allerdings wirken sich nicht alle Vorgaben der Standards auf die Darstellung der Seite aus, sondern beinhalten z.B. auch Vorgaben über den syntaktischen Aufbau von Webseiten.

spiel von Fahrerinformationssystemen in Kraftfahrzeugen.

1.1.1 Eigenschaften von Fahrerinformationssystemen im KFZ

In den KFZ der Oberklasse steigt die Funktionsvielfalt und das Informationsangebot stetig an. Neben der Darstellung fahrrelevanter Informationen wie Navigationsdaten oder Verkehrsfunk wird zusätzlich der Zugriff auf weitere Komfortfunktionen[12] wie Klimaanlage oder Fernsehen angeboten.
Der Zugriff auf diese Funktionen wird in zentralen Fahrerinformationssystemen (FIS) realisiert, in denen heutzutage bereits mehr als 700 Komfortfunktionen integriert sind (KILEY 2004; OERTEL und LACHERMEIER 2005). Diese FIS bestehen aus einem zentral in der Mittelkonsole platziertem Display mit entsprechenden Bedienelementen und gehören mittlerweile zur Standardausstattung in KFZ der Oberklasse und vermehrt der Mittelklasse. Beispielhaft sind in Abbildung 1.3 das FIS des VW-Phaeton (links) sowie das FIS *iDrive* von BMW (rechts) dargestellt. Das Phaeton-System besitzt Hardkeys[13], einen Dreh-Drück-Steller sowie neben dem Display angeordneter Softkeys[13] an zentraler Stelle in der Mittelkonsole. Das BMW-System verfügt dagegen über einen einzigen örtlich vom Display getrennten Dreh-Drück-Steller in Verlängerung der Armlehne.

Viele Funktionen eines FIS, wie die Bedienung der Navigation oder des Radios, müssen auch während der Fahrt zugängig sein. Dabei darf die Bedienung des FIS den Fahrer nur so wenig wie möglich von seiner eigentlichen Fahraufgabe ablenken (KOSTKA et al. 2001). Aus diesem Grund muss gerade bei der Entwicklung von FIS auf eine einfache und konsistente Bedienung geachtet werden (WISSELMANN et al. 2004).

Die Relevanz der Bewertung der Gebrauchstauglichkeit zeigt sich besonders hinsichtlich der steigenden Verbreitung von FIS. Im Zeitraum von Januar bis Dezember 2005 wurden insgesamt 37.827 Neuzulassungen in der Oberklasse und 213.160 in der oberen Mittelklasse[14] registriert (KRAFTFAHRT-BUNDESAMT 2006).

1.1.2 Festlegung und Bewertung des Systemdesigns

In den ersten Phasen des Entwicklungsprozesses (vgl. Abb. 1.1) erfolgt eine Festlegung der Anforderungen und der Funktionen des zu entwickelnden Systems sowie der auszuführenden Aufgaben. Im Rahmen der Spezifikation der Mensch-Maschine-Schnittstelle erfolgt anschließend das Interaktions-, Informations- und Oberflächendesign (NEWMAN und LANDAY 2000). Die Zusammenhänge sind in Abbildung 1.4 dargestellt und im Folgenden beschrieben.

[12] Komfortfunktionen haben im Gegensatz zu Primärfunktionen im KFZ keinen direkten Einfluss auf die Lenk- und Steuerungsaufgaben eines PKW-Fahrers.

[13] s. Glossar

[14] In der Statistik sind die 193.090 Neuzulassungen im Segment *SUV/Geländewagen* sowie 426.955 Neuzulassungen im Segment *Vans* nicht berücksichtigt, wobei viele der entsprechenden Modelle eine Ausstattung ähnlich der von Autos in der oberen Mittelklasse besitzen.

Abbildung 1.3: Fahrerinformationssysteme ohne (links – VW Phaeton) und mit (rechts – BMW iDrive) örtlicher Trennung der Anzeigefläche und Bedienelemente.

Anforderungsanalyse

Durch eine *Anforderungsanalyse* werden die gewünschten Ziele und Eigenschaften des Systems festgelegt. Ausgehend von groben Beschreibungen erfolgt unter Abwägung möglicher Alternativlösungen der Problemstellungen eine schrittweise Verfeinerung. Dabei werden ebenfalls hard- und softwaretechnische Vorgaben sowie das spätere Benutzungsumfeld berücksichtigt (MAYHEW 1999).

Funktionsanalyse

Das Ziel der *Funktionsanalyse* ist die Festlegung der zur Erreichung der definierten Ziele notwendigen Systemfunktionen. Dazu gehören interne Systemfunktionen ebenso wie Funktionen zur Mensch-Maschine-Interaktion. In der Funktionsanalyse werden schrittweise Funktionen identifiziert und ggf. in weitere Unterfunktionen geteilt (EHRLENSPIEL 2002; STEVENS et al. 2004). Das Ergebnis dieses Schrittes ist eine technische Spezifikation des gesamten Systems (FAULK 1997). Dabei kommen zur Erstellung dieser Spezifikation vereinzelt bereits Werkzeuge zur Angabe der Abläufe in Form von Statecharts (s.u.) zum Einsatz.

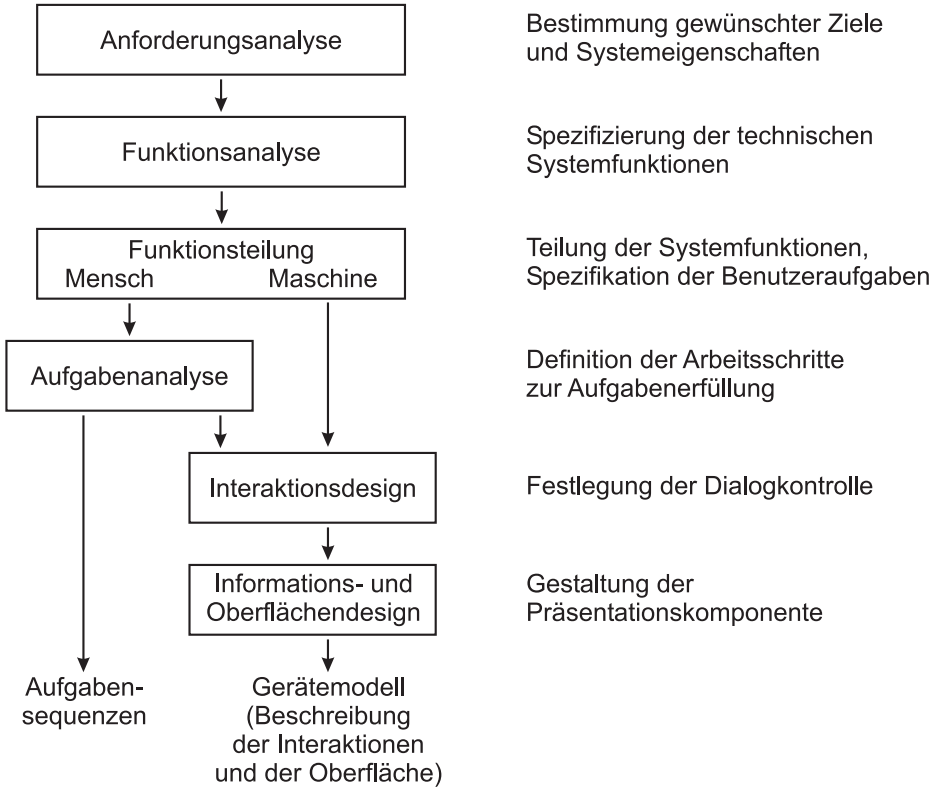

Abbildung 1.4: Schritte der Festlegung des Systemdesigns (NEWMAN und LANDAY 2000; STEVENS et al. 2004).

Funktionsteilung Mensch–Maschine

Nach der Festlegung der Funktionalität sind Entscheidungen darüber zu treffen, welche Teile der Funktionen vom Bediener, vom System oder von beiden auszuführen sind. Dabei ist im besonderen Maße zu beachten, dass sowohl der Bediener als auch das System charakteristische Stärken und Schwächen aufweisen. In der Literatur sind dazu zahlreiche Ausführungen zu finden (FITTS 1951; PRICE 1985), so dass an dieser Stelle auf eine ausführliche Betrachtung verzichtet wird.

Aufgabenanalyse

Aus der Funktionsteilung ergeben sich die Aufgaben, die der Bediener zu erfüllen hat. In der *Aufgabenanalyse* erfolgt eine Aufteilung einzelner Aufgaben in kleinere Teilaufgaben. Dazu werden Arbeitsschritte analysiert, die zur Erfüllung einer Aufgabe notwendig sind. Das Ergebnis dieser Analyse ist eine Liste der manuellen *Aufgabensequenzen*. Zusätzlich sind alle zur Aufgabenerfüllung notwendigen technischen Hilfsmittel, wie Anzeigen und Bedienelemente, zu identifizieren (KRAISS 1995).

Interaktionsdesign

Aufbauend auf der Funktionsteilung und den Aufgabensequenzen der Aufgabenanalyse werden auf der Grundlage der technischen Spezifikation beim *Interaktionsdesign* die Mechanismen für den Zugriff und die Manipulation von Systemdaten im Rahmen der Präsentationskontrolle spezifiziert (PAWAR 2004; GIERSICH et al. 2005). Das wird durch die Auswahl geeigneter Ein- und Ausgabeelemente (z.B. Tasten, Drehregler, Displays) erreicht. Weiterhin müssen die Mechanismen den Endbenutzer dabei unterstützen, die Aufgabensequenzen zum Erreichen eines Ziels zu planen und durchzuführen. Der Fokus liegt dabei auf der Gewährleistung einer konsistenten und aufgabenangemessenen Bedienung sowie der Möglichkeit der fehlertoleranten Eingabe (NORMAN 1988).

Die Abläufe der Bedienung werden häufig bereits in der technischen Spezifikation festgelegt und im Interaktionsdesign übernommen. Zur Spezifikation der Dialogkontrolle kommen häufig Zustandsübergangsdiagramme (Statecharts) zum Einsatz, die auf endlichen Automaten basieren (HAREL 1987; HORROCKS 1999; TIESCHKY et al. 2002). Ein Zustand des Diagramms entspricht einem Systemzustand der Benutzungsschnittstelle. Transitionen verbinden einen Zustand mit einem anderen und repräsentieren die Handlungsoptionen des Benutzers (z.B. Tastendruck).

Abbildung 1.5 zeigt das GUI des FIS Audi MMI (links) mit dem dazu gehörigen Statechart (rechts unten). Im Beispiel ist im Statechart der Zustand *Radio_Menü* ein Unterzustand von *Audi-MMI:power_on*. *Radio_Menü* enthält weitere Unterzustände zur Steuerung der Senderliste oder zum Einstellen des Klangs. Durch Druck der Hardkeys (hk) *hk_navi* oder *hk_car* wechselt das System in die jeweiligen Zustände, die wiederum Unterzustände besitzen.

Informations- und Oberflächendesign

Nach dem Interaktionsdesign erfolgt das *Informations-* und *Oberflächendesign*, das sich mit der Oberflächengestaltung und der Auswahl geeigneter Begriffe für das GUI (Präsentationskomponente) beschäftigt. Dabei steht bei der Oberflächengestaltung u.a. die Farbgebung, die Größe und Anordnung der Elemente – Ein- und Ausgabeelemente sowie die in Displays dargestellten Inhalte – eines UIs im Vordergrund (HIX und HARTSON 1993; MAYHEW 1999). Das Ziel des Informationsdesigns ist die Auswahl geeigneter aussagekräftiger Begriffe für alle textuellen Darstellungen[15]. Das umfasst u.a. die Überschriften und Menüeinträge in Displays sowie Beschriftung von Tasten und Reglern (ROSSON und CARROLL 2002).

Das GUI wird i.d.R. als Liste der abgebildeten Elemente beschrieben. Dazu zählen nicht nur die auf Displays dargestellten Objekte, sondern ebenfalls die Gestaltung der Eingabeelemente. Die Eigenschaften der Elemente sind in Form von Attributen beschrieben. Dazu gehören z.B. die Lage und Größe, die Farbgebung und Beschriftung. Eine GUI-Beschreibung des Audi MMI ist in Abbildung 1.5 oben rechts beispielhaft dargestellt. Bei `display` und `button`

[15] Das Fachgebiet, das sich mit der Bestimmung und dem Einsatz geeigneter Begriffe in Systemen beschäftigt wird *Wording* genannt.

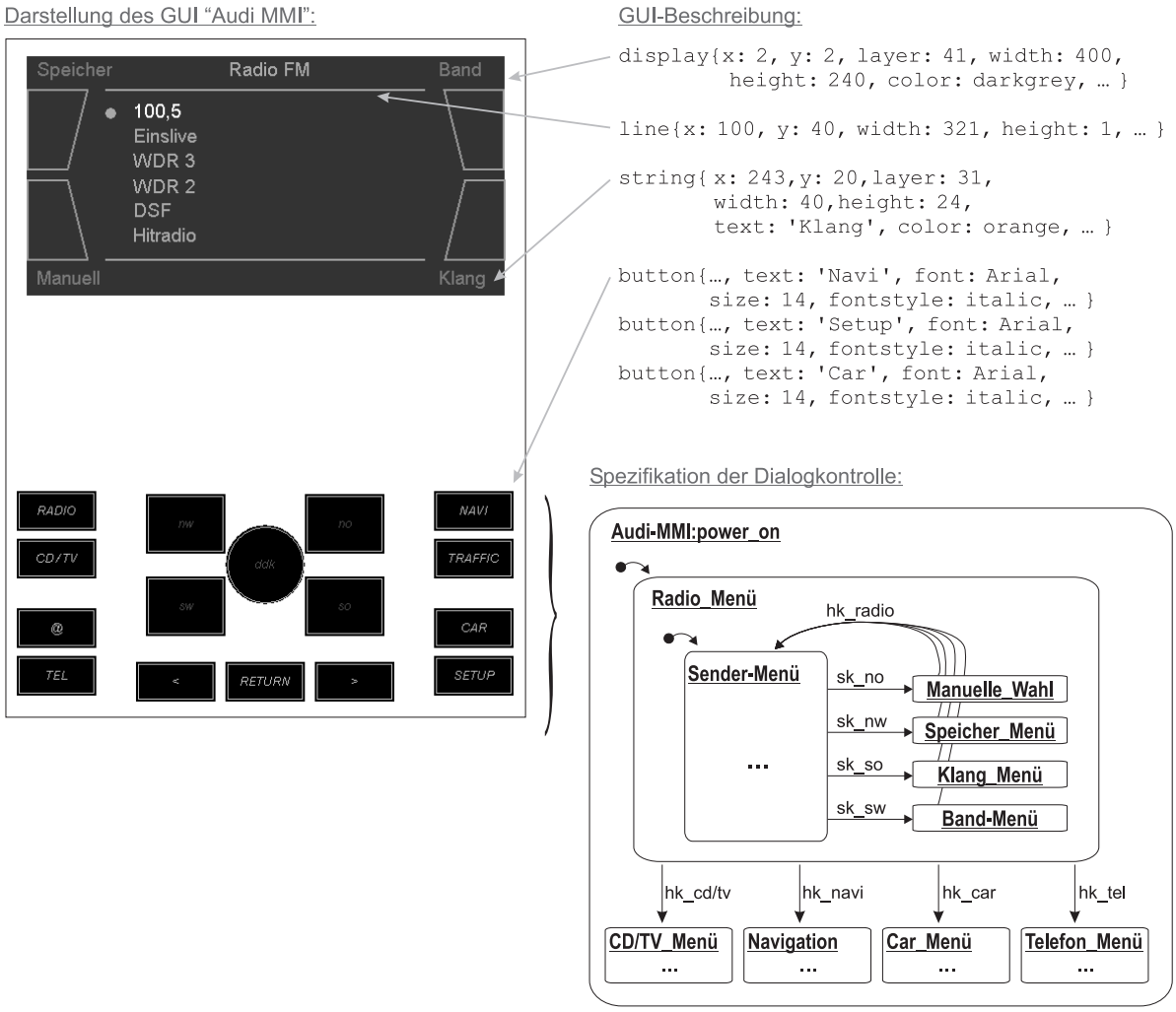

Abbildung 1.5: Beispielhafte Darstellung des Displays und der Eingabeelemente des Audi MMI. Die Beschreibung des GUI sowie die Spezifikation der Dialogkontrolle ist ebenfalls dargestellt.

handelt es sich um eine Beschreibung der Ein- und Ausgabeelemente, `line` und `string` stellen dagegen Informationen dar, die auf dem Display angezeigt werden. Als Attribute besitzen die Elemente z.B. ihre Position und Ausdehnung (`x,y,width,height`) oder ihre Farbe (`color`). Das Attribut `layer` gibt die grafische Ebene an, auf der sich die Elemente befinden; das ermöglicht die Kombination von Elementen übereinander (z.B. ein Rahmen hinter einem Text).

Das Ergebnis des Systemdesigns ist das Gerätemodell, das sich aus der Beschreibung der Interaktionen (als funktionales Modell zur Beschreibung der Präsentationskontrolle) und der Benutzungsoberfläche zusammensetzt.

Designbegleitende Bewertung

Fortgesetztes Bewerten und Anpassen des Designs gewährleistet, dass das Interface die festgelegten Ziele erfüllt (GÖTZE 1994; STEVENS et al. 2004; RÖSSGER 2005). Dazu kommen überwiegend kriterienorientierte Bewertungen zum Einsatz, die das entsprechende Design auf Konformität mit gegebenen Guidelines überprüfen (LOWE et al. 1999; OED et al. 2001; SONG 2002).

Bei der Bewertung von FIS muss besonderes Augenmerk auf konsistenter Gestaltung und Bedienung liegen (s. Kap. 1.1.1), da Konsistenz die Erlernbarkeit des Systems sowie die Bediengeschwindigkeit steigert und dadurch die Ablenkungszeit verringert (STEVENS et al. 2004; RÖSSGER 2005).

Beispielhaft sind im Folgenden einige bewertungsrelevante Eigenschaften eines FIS aufgeführt, die in diesem Zusammenhang bewertet werden müssen (s. Abb. 1.6).

Systembereich	Bewertungsfokus	betroffener Designbereich
Dialogkontrolle	- gleichförmige Bedienung - Verwendung von Bedienstereotypen	Interaktionsdesign
Präsentations-oberfläche	- Anzahl der GUI-Elemente - Anordnung und Platzierung der GUI-Elemente - Farbgebung	Oberflächendesign
	- Verständlichkeit der Begriffe - Vermeidung synonymer Begriffe	Informationsdesign

Abbildung 1.6: Auswahl bewertungsrelevanter Eigenschaften eines Fahrerinformationssystems im Zusammenhang mit den Designbereichen.

Verantwortlich für eine konsistente Bedienung ist die Dialogkontrolle, deren Verhalten im Interaktionsdesign festgelegt wird. Konsistenz wird durch die Verwendung gleicher Bedienelemente zur Bearbeitung ähnlicher Aufgaben, z.B. die Anwahl eines Funktionenbereichs („Navigation“, „Telefon“, etc.) mit Hardkeys; Listenauswahl mit Dreh-Drück-Steller, erreicht (PREECE et al. 2002). Die Einhaltung von erlernten Bedienstereotypen[16] muss dabei ebenso eingehalten werden.

In der Präsentationskomponente beeinflussen die Anzahl und Anordnung sowie die Farbgebung der dargestellten GUI-Elemente die Gestaltungskonsistenz. Im Informationsdesign muss auf den Einsatz verständlicher Begriffe und die Vermeidung von synonymen Begriffen geachtet werden. Da die Semantik von Begriffen nicht automatisch gemessen werden kann, ist eine Bewertung lediglich dahingehend möglich, ob verwendete Begriffe Vorgaben entsprechen.

[16]Der Begriff Stereotyp bezeichnet die erwartungskonforme Bedienung von Stellteilen. Eine ausführliche Erklärung sowie eine Begriffsdiskussion ist im Glossar gegeben.

Als Vorgaben können Ontologien[17] dienen, die empirisch ermittelte Begriffe und ihre semantischen Beziehungen in Form eines Graphen abbilden.

Zur Bewertung des Oberflächen- und Informationsdesigns muss eine formale Beschreibung des GUI (vgl. Abb. 1.5) vorliegen. Die Bewertung des Interaktionsdesigns benötigt zusätzlich zur GUI-Beschreibung Informationen aus der Spezifikation der Dialogkontrolle (z.B. in Form eines Statechart, vgl. Abb. 1.5). Da die Spezifikation der Dialogkontrolle und die Beschreibung des GUI i.d.R. in unterschiedlichen Formaten vorliegen, muss zur Bewertung der Aktionen im Zusammenhang mit dem GUI eine Zuordnung der Zustände einer Interaktionsspezifikation zu der entsprechenden Repräsentation des GUI erfolgen. Ein mögliches Vorgehen ist in Abbildung 1.7 dargestellt und im Folgenden erklärt.

In einer Spezifikation der Dialogkontrolle (in Abb. 1.7 am Beispiel eines Statechart des Audi-MMI) sind die Zustände des Systems hierarchisch definiert. Durch die manuelle Auswahl eines Zielzustands als zu erreichendes Ziel einer Aufgabe ergibt sich ein *Signal/Zustand-Pfad*, wobei die möglichen Aufgabenziele in der Aufgabenanalyse definiert sind. Dieser Pfad enthält alle Zustände und Aktionen, die der Benutzer durchführen muss, um eine Aufgabe erfolgreich bearbeiten zu können. Zur Bewertung der Aktionen im Zusammenhang mit dem GUI im Rahmen des Interaktionsdesigns muss eine Zuordnung der Zustände des Signal/Zustand-Pfades zu den entsprechenden GUI-Repräsentationen in Form einer *Dialogfolge* erfolgen.

Heutzutage stehen zahlreiche Werkzeuge zur Spezifizierung des Interaktionsdesign sowie zur Erstellung des Informations- und Oberflächendesigns zur Verfügung. Dazu gehören z.B. *Statemate Magnum* der Firma ILogix[18], *Rapid* der Firma E-SIM[19], *Virtual Prototypes* von Engenuity Technologies[20], Siemens Simatic HMI[21] sowie als neuestes Produkt das Werkzeug *Inside* von Princess Interactive[22]. Diese Werkzeuge erlauben die elektronische Erstellung und Verarbeitung des Systemdesigns sowie die Simulation eines Systemprototypen. Teilweise werden Sie bereits zur Festlegung der technischen Spezifikation benutzt. Keines der gängigen Spezifikationsformate bzw. Spezifikationswerkzeuge erlaubt jedoch die automatische Generierung einer Dialogfolge (ZIEREN 2000; PFISTERER 2002; VANDERDONCKT 2005). Eine weitere Möglichkeit der Generierung einer Dialogfolge liegt in der Simulation der Interaktion eines Benutzers, dem Aufzeichnen der jeweils durchgeführten Aktion und einer Beschreibung des bei jedem Bedienschritt dargestellten GUI.

Im Laufe der Zeit wurden zahlreiche Werkzeuge entwickelt, um Guidelines elektronisch verarbeiten zu können. Das erleichtert das Bearbeiten, Suchen und Überprüfen der Kriterien deutlich und beschleunigt dadurch die Bewertung. Zudem gewährleistet eine automatische Bewertung im Gegensatz zur einer manuelle Anwendung durch Experten die objektive Anwendung der Kriterien und steigert dadurch die Nachvollziehbarkeit von Bewertungsergebnissen (SOUZA und BEVAN 1990). Allerdings erlauben diese Werkzeuge lediglich die Überprüfung

[17] Ontologien werden ausführlich in Kapitel 3.3 bzw. Anhang C vorgestellt
[18] http://www.ilogix.com
[19] http://www.e-sim.com
[20] http://www.virtualprototypes.ca
[21] www.siemens.de/simatichmi
[22] http://www.princess-interactive.com

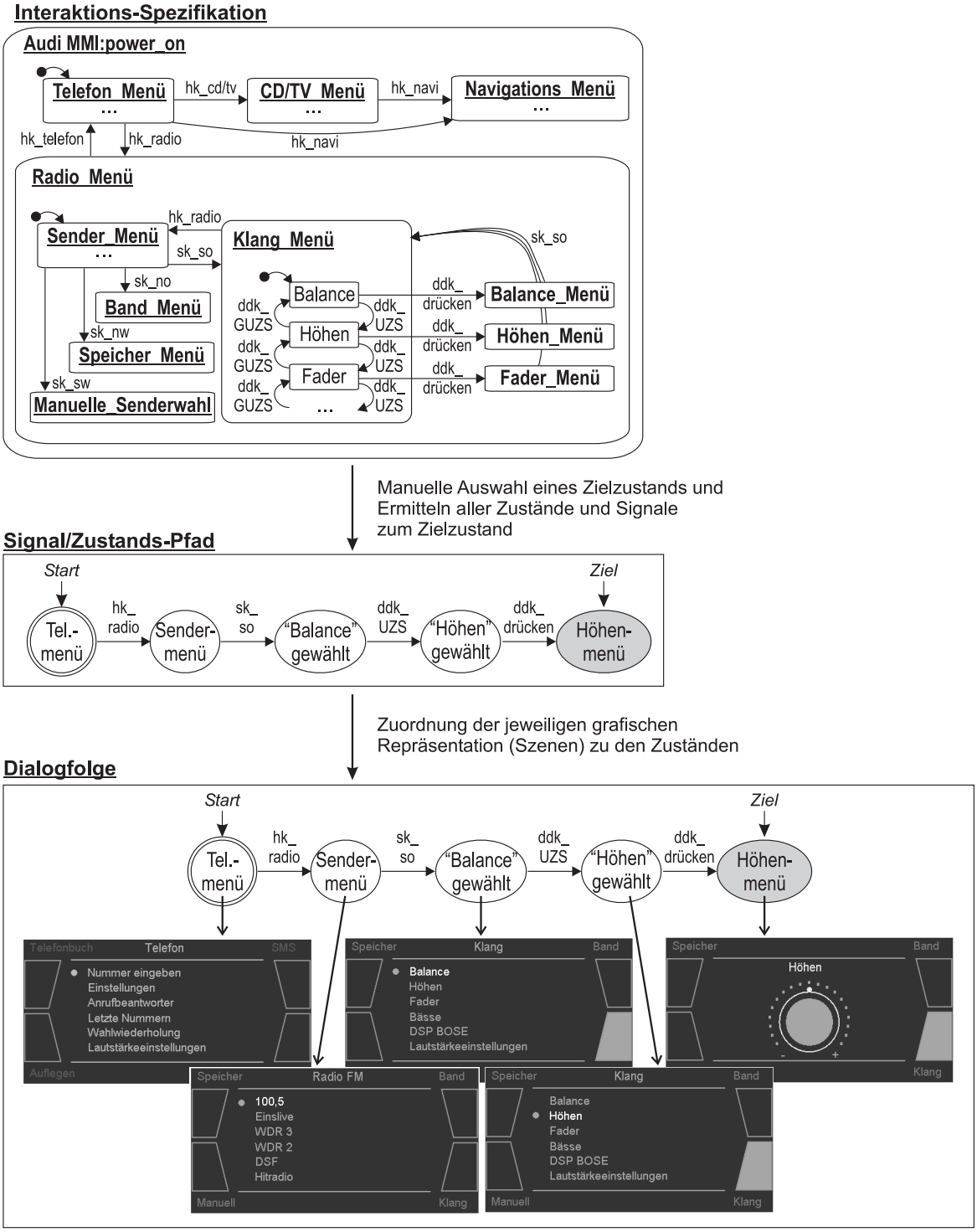

Abbildung 1.7: Mögliche Stufen bei der Erstellung einer Dialogfolge aus einer Interaktionsspezifikation. Nach Anwahl eines Zielzustandes ergibt sich ein Signal/Zustand-Pfad. Durch die Zuordnung der GUI-Repräsentation zu den Zuständen dieses Pfades entsteht eine Dialogfolge.

des Oberflächendesigns. Die Verwendung synonymer Begriffe sowie die Überprüfung auf konsistente Bedienung ist nicht möglich.

Regeln zur Überprüfung von Systembeschreibungen auf Guideline-Konformität haben i.d.R. einen Wenn-Dann-Aufbau. Dieses Konstrukt folgt dem Prinzip funktionaler Sprachen[23]. Häufig müssen jedoch auch komplexere mathematische Berechnungen, wie die Transformationen von Farbwerten in einen anderen Farbraum (z.B. die Berechnung des ΔE-Wertes in Kap. 3.4.2), durchgeführt werden. Dazu eignen sich funktionale Sprachen nicht besonders gut, so dass dafür imperative Sprachen[23] zum Einsatz kommen.

Beispielhaft wird ein zu berücksichtigendes Kriterium der Farbgebung vorgestellt, welches fordert, dass zwei benachbarte GUI-Elemente nicht die Farben Rot und Blau haben dürfen, da rote und blaue Elemente nicht gleichzeitig vom Auge fokussierbar sind (OSACA E.V. 1997). Eine automatisch überprüfbare Implementation dieses Kriteriums ist Abbildung 1.8 gegeben.

```
WENN
     existiert gui-element(id1, farbe1)
     UND existiert gui-element(id2, farbe2)
     UND benachbart(id1, id2)
     UND (
            (farbe1=blau UND farbe2=rot)
            ODER (farbe1=rot UND farbe2=blau)
          )
DANN Fehlermeldung
```

Abbildung 1.8: Beispiel einer Regel zur Überprüfung der Farbgebung.

Die automatisch zu bewertenden Kriterien sind in den existierenden Bewertungswerkzeugen im Programmcode hardcodiert, was umfangreiche Programmierkenntnisse vom Entwicklungsingenieur erfordert und eine Änderung bestehender oder Eingabe neuer Guidelines erschwert. Zudem ist bei den Werkzeugen lediglich ein konkretes Format des Gerätemodells überprüfbar, ein Transfer der implementierten Guideline zur Überprüfung weiterer Systeme ist nicht möglich.

1.2 Zielsetzung der Arbeit

Das Ziel dieser Arbeit ist die Entwicklung und Implementierung eines Werkzeugs zur automatischen kriterienorientierten Bewertung der Gebrauchstauglichkeit interaktiver Systeme. Dazu soll das Werkzeug die Bewertung des Interaktions-, des Oberflächen- und Informationsdesigns ermöglichen.

Die Bewertungsbasis bilden elektronisch vorliegende Systeminformationen bzw. Gerätebeschreibungen, die während der technischen Entwicklung ohnehin anfallen. Den Import dieser Daten muss das Werkzeug ebenso gewährleisten wie die Möglichkeit, sich an das Format der

[23] s. Glossar

zur Verfügung stehenden Systemdaten anpassen zu lassen, um eine möglichst große Flexibilität hinsichtlich Art und Zeitpunkt der Bewertung bieten zu können.

Zur Implementierung der Guidelines muss eine Regelsprache zum Einsatz kommen, die der natürlichen Beschreibung von Guidelines entspricht und möglichst wenig Programmierkenntnisse vom Entwicklungsingenieur verlangt. Zur Implementierung aufwändiger Berechnungen in prozeduraler Form muss eine geeignete prozedurale Sprache an die Regelsprache angebunden werden.

Das Werkzeug soll alle o.g. Funktionalitäten unter einer einheitlichen GUI vereinigen. Dazu muss sowohl die Regeleingabe und -verwaltung, als auch die Steuerung der automatischen Bewertung Unterstützung finden. Die geforderten Eigenschaften eines Werkzeugs zur automatischen kriterienorientierten Bewertung lassen sich in die vier Bereiche *Bewertung des Oberflächendesigns*, *Bewertung des Informationsdesigns*, *Bewertung des Interaktionsdesigns* und *(sonstige) Eigenschaften des Werkzeugs* einteilen. Die wichtigsten Eigenschaften zeigt Tabelle 1.1.

Tabelle 1.1: Übersicht der geforderten Bewertungsmöglichkeiten und Eigenschaften eines Werkzeugs zur kriterienorientierten Bewertung.

Kategorie	**Beschreibung / Fokus der Bewertung**
Bewertung des Oberflächendesigns	Lage/Anordnung
	Farbgebung
	Weitere Attribute von GUI-Elementen
	Oberflächenkonsistenz
Bewertung des Informationsdesigns	Semantik der Begriffe
	Synonyme/Homonyme
Bewertung des Interaktionsdesigns	Angabe über Gedächtnisbelastung
	Stereotypenbenutzung
	Bedienungskonsistenz
Eigenschaften des Werkzeugs	Domänenübergreifende Bewertung
	Unabhängigkeit von GUI-Format
	Freie Programmierung von Regeln
	Einbindung von proz. Sourcecode
	Dialogunterstützung
	Einbindung in Entwicklungsumgebung
	Betriebssystemübergreifend

Im Einzelnen verfolgt diese Arbeit folgende Ziele:

(1) Realisierung der Importmöglichkeit einer formalen GUI-Beschreibung interaktiver Systeme zur Bewertung des Oberflächen- und Informationsdesigns.

(2) Erstellung eines Moduls zur Simulation eines interagierenden Benutzers unter Verwendung von Vorwissen zur Bedienung. Das Ergebnis dieser simulierten Bedienung ist eine Dialogfolge, die als Grundlage der Bewertung des Interaktionsdesigns dient.

(3) Identifizierung geeigneter Maße und Bewertungsregeln zur Bewertung des gesamten Systemdesigns.

(4) Entwicklung eines Editors zur Eingabe und Verwaltung der Bewertungsregeln und -maße aus (3) in funktionaler Form.

(5) Erstellung eines Moduls zur automatischen Überprüfung der formalen GUI-Beschreibung bzw. der Dialogfolge aus (1) und (2) durch die Bewertungsregeln und -maße aus (4).

(6) Integration einer Schnittstelle zum Hinzufügen von Methoden und Berechnungen, die in einer prozeduralem Programmiersprache implementiert sind. Die Schnittstelle muss den Aufruf dieser Methoden im Regeleditor (4) ermöglichen.

(7) Anbindung des Werkzeugs an einen gängigen Ontologieeditor zum Import von Begriffsontologien zur Bewertung des Informationsdesigns.

Bei der Entwicklung des Werkzeugs muss im Hinblick auf Portabilität möglichst eine plattformübergreifende Programmiersprache Verwendung finden.

Das Werkzeug soll sich auf Grund seiner konzeptuellen Struktur generell zur Bewertung jedes interaktiven Systems eignen. Die Verwendung eines allgemein gültigen und domänenunabhängigen formalen GUI-Formats soll die Integration des Werkzeugs potentiell in jede Systementwicklung ermöglichen. Je nach Einsatzart und -gebiet muss dabei lediglich eine Anpassung der überprüfenden Kriterien sowie des zum Einsatz kommenden Vorwissens erfolgen. Im Rahmen dieser Arbeit wird das Werkzeug am Beispiel einer Bewertung von Fahrerinformationssystemen in einem KFZ vorgestellt.

1.3 Kapitelübersicht

Die Arbeit ist im weiteren Verlauf wie folgt gegliedert:

In Kapitel 2 findet eine Betrachtung der Methoden zur Bewertung der Gebrauchstauglichkeit interaktiver Systeme sowie eine Abgrenzung der kriterienorientierten Bewertung zum restlichen Methodeninventar statt. Eine Klassifizierung der Abstraktionsgrade von Guidelines sowie ein allgemeines Vorgehen zur Anwendung ist ebenfalls gegeben. Im Anschluss erfolgt die Vorstellung einer Auswahl existierender Werkzeuge zur kriterienorientierten Bewertung. Ausgehend von den Einsatzmöglichkeiten der Bewertungsmethoden und den Schwächen bestehender Systeme wird das Konzept für das in dieser Arbeit entwickelte Werkzeug vorgestellt.

Das Kapitel 3 befasst sich mit der Darstellung eines Formalismus zur Beschreibung eines GUI. Nachfolgend wird die Möglichkeit der Implementierung von Guidelines als Regeln zur Bewertung des Systemdesigns vorgestellt. Zur Bewertung der eingesetzten Begriffe sowie

zur Abbildung der Dialoghierarchie eignen sich empirisch erhobene Ontologien, die im Anschluss erklärt werden. Abschließend erfolgt die detaillierte Vorstellung ausgewählter Maße zur Bewertung des Oberflächen-, Informations- und Interaktionsdesigns.

Der Einsatz eines interaktiven Bedienagenten zur Generierung einer Dialogfolge ist in Kapitel 4 erläutert. Nach einer Übersicht über den Aufbau des Agenten folgt eine detaillierte Beschreibung der Funktionen und Komponenten. Dabei wird das Vorgehen zur Erkennung des aktuellen Zustands des interaktiven Systems, der Identifikation des Folgezustands zum Erreichen eines Aufgabenziels sowie die Bewertung der Anzeige-Bedienelemente-Konstellation zur Auswahl einer Bedienaktion erklärt.

Ausgehend von dem Konzept des Werkzeugs aus dem zweiten Kapitel gibt Kapitel 5 einen Überblick über den Aufbau, die Funktionalität und die Realisierung des in dieser Arbeit entwickelten Bewertungswerkzeugs REVISER. Nach der Vorstellung einer allgemeinen Architektur von Expertensystemen folgt die Erklärung des Aufbaus und der Komponenten des Werkzeugs. Als zentraler Bestandteil des Werkzeugs wird der Kriterieneditor beschrieben, der die Implementierung von Guidelines in Regelform erlaubt. Anschließend sind die Möglichkeiten der Ablaufsteuerung einer kriterienorientierten Bewertung vorgestellt.

In Kapitel 6 findet als Evaluierung des Werkzeugs die Bewertung zweier Fahrerinformationssysteme statt. Zur Bewertung der FIS werden entsprechende Bewertungsregeln implementiert. Ebenfalls erfolgt die Vorstellung der verwendeten Begriffsontologie sowie des vom Bedienagenten benötigten Bedienwissens. Auf Basis der Regeln erfolgt die automatische kriterienorientierte Bewertung des jeweiligen Systemdesigns mit der Analyse der Bewertungsergebnisse. Anschließend wird das Vorgehen und die Durchführung einer empirischen Evaluierung der gleichen FIS vorgestellt. Den Abschluss bildet eine vergleichende Analyse der Ergebnisse der automatischen mit der empirischen Bewertung, um die Tauglichkeit des Werkzeugs zu verdeutlichen.

Im letzten Kapitel sind die Ergebnisse der vorliegenden Arbeit zusammenfassend dargestellt und es wird ein Ausblick auf weitere Entwicklungsperspektiven gegeben.

Kapitel 2

Methoden zur Bewertung der Gebrauchstauglichkeit

Bei der Entwicklung und Erstellung von Methoden zur Bewertung der Gebrauchstauglichkeit liegt der Fokus grundsätzlich auf dem Erreichen eines konkreten Bewertungsziels. Allgemeine Ziele lassen sich durch drei einfache Fragen charakterisieren (GEDIGA und HAMBORG 2002):

1. *„Which is better?“* – Vergleichende Bewertung
   Die Evaluierung zielt auf einen Vergleich zwischen Systemalternativen, z.B. um für einen bestimmten Anwendungszweck das beste Produkt auszuwählen, um im Rahmen des Protoypings zwischen Designalternativen zu entscheiden oder die Optimierung von Versionen zu kontrollieren (HOLZ AUF DER HEIDE 1993). Ein Beispiel für diesen Ansatz ist die vergleichende Untersuchung von Fahrerinformationssystemen in Mittel- und Oberklassewagen der Firma SirValUse (SIRVALUSE 2004).

2. *„How good?“* – Qualitative Bewertung
   Bei dieser Zielsetzung geht es um die Bestimmung der Ausprägung gewünschter oder geforderter Systemeigenschaften. Hierzu zählt die Bewertung von Systemen zum Ende des Entwicklungsprozesses, z.B. in Bezug auf zuvor bestimmte Usability-Ziele oder die Überprüfung auf Normenkonformität (HEGNER 2003).

3. *„Why bad?“* – Identifizierung von Schwachstellen
   Die Evaluierung soll Schwachstellen als Ausgangspunkt für die Gestaltung oder direkte Gestaltungsvorschläge liefern. Der typische Anwendungsbereich liegt in der am Prototyping orientierten Entwicklung bzw. Weiterentwicklung von Systemen (HIX und HARTSON 1993; KARAT 1994).

Im Folgenden werden allgemeine Ziele und Methoden der Bewertung der Gebrauchstauglichkeit interaktiver Systeme vorgestellt (Kap. 2.1). Darauf aufbauend werden in Kapitel 2.2 die Möglichkeiten der kriterienorientierten Bewertung detailliert betrachtet. Das allgemeine Vorgehen zur Formulierung, Auswahl und Anwendung geeigneter Kriterien steht dabei im

Vordergrund. Anschließend erfolgt die Vorstellung und der Vergleich von Werkzeugen zur automatischen, kriterienorientierten Bewertung in Kapitel 2.3. Aufbauend auf den Erkenntnissen aus der Untersuchung bestehender Werkzeuge wird das Konzept eines neuen Werkzeuges zur automatischen Bewertung vorgestellt (Kap. 2.4).

2.1 Klassifizierung von Bewertungsmethoden

Die Unterteilung von Bewertungsmethoden in die drei Klassen formale, kriterienorientierte und empirische Methoden ist mittlerweile weit verbreitet (KRAISS 1995; TIMPE et al. 2002; GEDIGA und HAMBORG 2002). Im Folgenden werden einige Eigenschaften dieser Methodenklassen betrachtet.

Bewertung in unterschiedlichen Entwicklungsphasen

Die Zuordnung der Methodenklassen zu den Phasen der Systementwicklung ist in Abb. 1.1 (Seite 3) dargestellt. Formale und kriterienorientierte Methoden werden dabei überwiegend entwicklungsbegleitend eingesetzt, da der Aufwand der Bewertung im Gegensatz zu empirischen Methoden relativ gering ist. Dabei können je nach Entwicklungsstadium des Prototyps neu hinzukommende Features evaluiert und die Ergebnisse schrittweise in das weitere Design eingebunden werden (ZIEGLER und ILG 1993). Gerade die Prüfung auf Konformität des entwickelnden Systems zu bestehenden Regelwerken nimmt dabei durch die zunehmende Etablierung firmeninterner, nationaler und internationaler Standards eine immer größere Rolle ein (TIMPE et al. 2002)

Bei bereits fertig gestellten Produkten zielt die Bewertung nicht auf eine weitere Verbesserung des Designs, sondern dient der Produktauswahl und als Qualitätsnachweis (ZIEGLER und ILG 1993). Dazu kommen überwiegend empirische Methoden zum Einsatz. Mit Hilfe dieser Methoden lassen sich nicht nur objektive Maße erheben, sondern ebenso die subjektive Meinung der Probanden ermitteln. Durch die Verwendung eines realen (Software-)Prototypen und dem Einsatz von Probanden sind diese Methoden aus Zeit- und Kostengründen nur sehr eingeschränkt entwicklungsbegleitend einsetzbar.

Grad der Entwicklungsunterstützung

Häufig werden Bewertungsmethoden nach dem Grad der Entwicklungsunterstützung zwischen *formativen*[1] und *summativen* Methoden unterschieden (HOWARD und MURRAY 1987; HOLYER 1993; GEDIGA und HAMBORG 2002; BURNS 2004).

[1] Die Terminologie der *formativen* und *formalen* Bewertung erscheint auf den ersten Blick gleich, unterscheidet sich jedoch wesentlich in der Bedeutung im Bewertungskontext. Eine formative Bewertung weist auf den formenden Charakter der Methode hin, z.B. durch konstruktive Vorschläge zur Behebung von Gestaltungsfehlern. Formale Methoden basieren dagegen auf einem formalisierten Modell der Benutzeraktionen, auf Grund dessen sich z.B. Lern- und Ausführungszeiten bei der Aufgabenbearbeitung vorhersagen lassen (MARRENBACH

Eine *formative* Bewertung ist in der Lage, auf Grund von Gestaltungsstandards Entwurfsfehler zu finden, die dem späteren Benutzer eine Bedienung erschweren können. Formative Methoden sind darüber hinaus wesentlich an der Entwicklung des Systems beteiligt, indem sie konstruktive Vorschläge zur Verbesserung der Systemgestaltung erlauben. Die Bewertungen führen meistens Experten durch (FARENC 1997). Die kriterienorientierte Bewertung ist somit die einzige, die formativ sein kann.

Summative Methoden beziehen häufig die Benutzer in die Bewertung mit ein, dabei kann es sich um Benutzermodelle oder reale Benutzer handeln (BURNS 2004). Aus diesem Grund zählen formale sowie empirische Methoden hauptsächlich zur summativen Bewertung. Summative Methoden eignen sich vor allem zur vergleichenden Bewertung von ähnlichen Systemen bzw. Designvarianten. Diese Methoden liefern abstrakte Aussagen über die Benutzbarkeit eines Produkts im Vergleich mit anderen oder vor dem Hintergrund von Kriteriensystemen. Der Nachteil solcher Methoden liegt darin, dass sie nur beiläufig und informell Hinweise auf das „Wie“ einer Umgestaltung in Richtung der Kriteriensysteme generieren können (TIMPE et al. 2002).

Art der Datengrundlage

Bei der Bewertung der Gebrauchstauglichkeit lassen sich Methoden danach unterteilen, ob eine Bewertung auf Grund eines realen oder simulierten Systems bzw. Benutzers durchgeführt wird. So unterscheiden sich z.B. Methoden, die auf realen Benutzern basieren (empirische Bewertung) von denen, die Benutzeraktionen lediglich simulieren (formale Bewertung). Diese Einteilung eignet sich für nahezu alle Arten von Methoden, ist daher weitgehend etabliert (NIELSEN 1994; AVOURIS 2001; IVORY und HEARST 2001; SNYDER 2003) und wird im Folgenden vorgestellt (s. Abb. 2.1).

Die Grundlage der formalen Bewertung ist die *analytische Modellbildung*, welche Methoden beinhaltet, die eine Bewertung sowohl auf einem Modell des Systems als auch auf einem Modell des Endbenutzers durchführen. Dabei kommen auf der Systemseite i.d.R. technische Spezifikationen und auf der Benutzerseite Modelle zur Abbildung kognitiver Prozesse zum Einsatz. Das Ergebnis solcher Bewertungen sind Vorhersagen von Maßen der Gebrauchstauglichkeit wie z.B. Ausführungs- oder Lernzeiten (HAMACHER und MARRENBACH 2001). Häufig basieren diese Methoden auf der Simulation der Interaktion eines Benutzers mit dem System. So beschreibt die GOMS-Methode[2] als Beispiel einer populären formalen Bewertungsmethode sensorische, motorische und kognitive Aktionen des Benutzers bei der Aufgabenbearbeitung (KIERAS 2004).

Empirische Methoden bewerten unter Einbeziehung realer Benutzer. Beim *Testen* interagieren Benutzer mit einem Interface (Prototyp oder fertiges System). Ziel ist die erfolgreiche Bearbeitung vorgegebener Aufgaben. Bei der Aufgabenbearbeitung werden die Benutzer beobach-

2001). Somit kann theoretisch eine formale Bewertungsmethode durchaus auch formativ sein, in der Praxis existiert jedoch kein nennenswertes Beispiel.

[2] s. Glossar

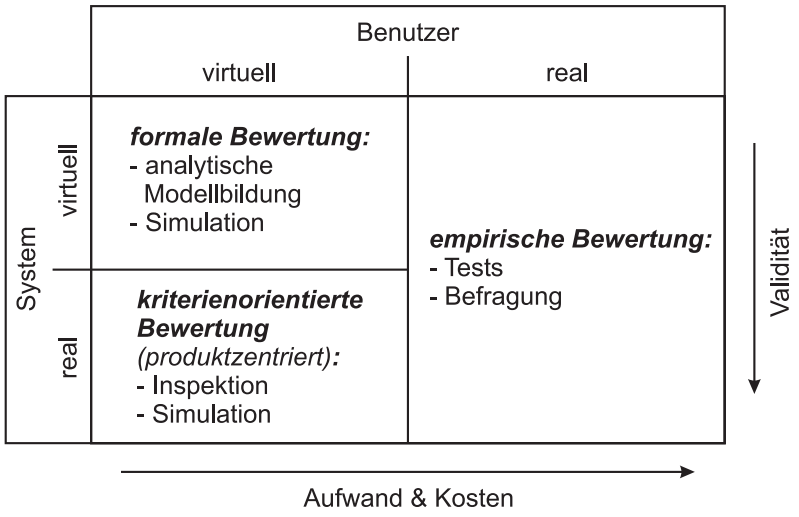

Abbildung 2.1: Einordnung von Bewertungsmethoden nach Art des Vorgehens (angelehnt an (FITZPATRICK 1999)).

tet und relevante Daten (z.B. Aktionen) gespeichert und später ausgewertet (ABDULKHAIR 2004). *Befragungen* ergeben als Grundlage einer Systembewertung die Meinungen von Endbenutzern. Dabei wird die subjektive Meinung von Endbenutzern entweder durch Fragebögen oder Interviews festgestellt. Zu diesen Methoden gehören ebenso (verdeckte oder offene) Beobachtungen der Benutzer beim Bedienen des Systems. Bei virtuellen Systemen ist häufig die Systemfunktionalität nur beschränkt vorhanden, so dass überwiegend Nutzerbefragungen durchgeführt werden. Tests eignen sich dagegen vor allem zur Bewertung mit realen Systemen (RAUTERBERG 1997).

Die kriterienorientierte Bewertung basiert überwiegend auf der Betrachtung realer Systeme ohne Einbeziehung realer Benutzer. Die *Inspektion* bezeichnet dabei die Bewertung eines interaktiven Systems durch einen Experten auf Basis von Heuristiken oder Guidelines (HENNINGER et al. 1995). Solche Bewertungen werden üblicherweise entwicklungsbegleitend durchgeführt. Dabei reicht je nach Fokus der Bewertung bzw. betrachtetem Designbereich bereits eine Beschreibung oder Spezifikation des Systems. In späteren Entwicklungsphasen wird sie vor allem an einem Teilsystem bzw. einem Prototypen durchgeführt (NIELSEN 1994). Zur Überprüfung von Aspekten des Interaktionsdesigns (vgl. Abb. 1.6 auf Seite 12) lassen sich typische Benutzeraktionen simulieren. Der Unterschied zur analytischen Modellbildung besteht in der tatsächlichen Durchführung der Interaktionen. Dadurch können zusätzliche Störfaktoren berücksichtigt und bewertet werden (IVORY und HEARST 2001).

Da Benutzerbefragungen sehr aufwändig sind, lassen sich Bewertungen auf Basis virtueller Benutzer (z.B. Benutzermodelle) häufig schneller und kostengünstiger durchzuführen (RAUTERBERG 1996). Die Validität der Aussagen bezüglich der Gebrauchstauglichkeit hängt dabei direkt mit dem Umfang der zur Verfügung stehenden Systemdaten zusammen (LINDGAARD et al. 2005).

2.2 Möglichkeiten der kriterienorientierten Bewertung

Eine kritische Betrachtung aktueller Kriteriensysteme und Evaluationsansätze offenbart auf den ersten Blick ein verwirrendes Dickicht an Gestaltungskriterien (MARIAGE et al. 2005). Mit Fokus auf die Zielsetzung der Kriterien bei der Bewertung lässt sich eine Ordnung herstellen, wobei das zentrale Ordnungsmerkmal die Nähe zum zu bewertenden System darstellt (TIMPE et al. 2002; BOWEN 2005). Diese Ordnung ist in Abbildung 2.2 dargestellt.

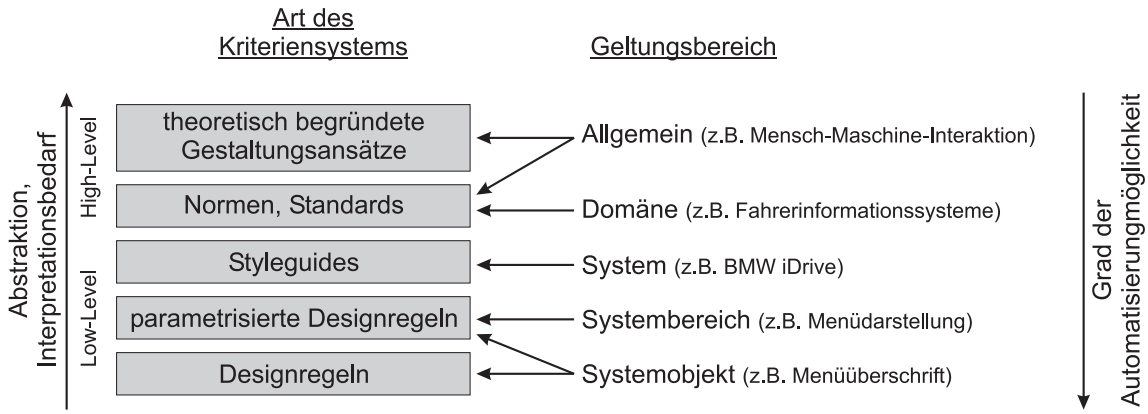

Abbildung 2.2: Arten und Geltungsbereiche von Kriteriensystemen.

Theoretisch begründete Evaluationsansätze sind bewusst ohne Bezug auf eine konkrete technische Gestaltung formuliert und nutzen allgemeine arbeitswissenschaftliche Erkenntnisse, z.B. (ISO 9241-10 1996). Der Nachteil liegt in der relativ großen Distanz der Kriterien zum technischen Gestaltungsgegenstand, den konkreten Benutzern und Aufgaben.

Normen und Standards beschäftigen sich mit Kriterien bezüglich einer Domäne. Dabei sind die Angaben so abstrakt, dass Aussagen unabhängig von herstellerspezifischen Eigenschaften gemacht werden, jedoch konkret genug, um domänenspezifische Eigenschaften von anderen Domänen abzugrenzen. So existieren z.B. ISO-Normen für Bildschirmgeräte (ISO 9241) oder Softwaresysteme (ISO 9126), die inhaltlich ähnlich, in der Ausprägung jedoch je nach Anwendungsgebiet verschieden sind (ISO 9241 1997; ISO9126 2003). Ein weiteres Beispiel sind die Human Factor Design Standards des US Department of Transportation, die sich allgemein mit der Gestaltung von Informationssystemen in KFZ beschäftigen (DEPARTMENT OF TRANSPORTATION 1998).

Die größte Nähe zum Gestaltungsgegenstand haben die Styleguides und Designregeln der Hersteller interaktiver Systeme. Sie stellen konkrete Regelwerke auf. Dabei werden Styleguides für komplette Systeme angegeben, um eine einheitliche Gestaltung verschiedener Ein- und Ausgabeelemente zu sichern und ein *Corporate Identity* zu gewährleisten . So existieren für alle großen Softwaresysteme entsprechende Styleguides, z.B. für SAP R/3, OS-X von Apple, OS2 von IBM oder Windows XP von Microsoft (SAP 2004; APPLE 2005; IBM 1992; MICROSOFT 2001).

Konkrete und in der Praxis direkt überprüfbare Aussagen für einzelne Gestaltungsobjekte bilden auf unterster Ebene die Designregeln (MARIAGE et al. 2005). Zusammengefasste bzw.

parametrisierte Designregeln können für einzelne Systembereiche (z.B. Menüs oder Tastengestaltung) angewendet werden (SMITH 1988; SCAPIN et al. 2001a).

Theoretische Evaluationsansätze und Normen stellen *High-Level-Rules* mit relativ großem Abstraktionsgrad dar, während Styleguides und Designregeln zu den *Low-Level-Rules* gezählt werden (COSTABILE und MATERA 2001).

Im Folgenden wird das Vorgehen zur Erfassung von Expertenwissen in Form von Guidelines erläutert.

2.2.1 Erfassung von Expertenwissen

Die für die Erstellung der Kriteriensysteme verantwortlichen Teilnehmer sowie der Einfluss unterschiedlicher Wissensgebiete sind je nach Art des Systems unterschiedlich. Sie sind in Abbildung 2.3 dargestellt und im Folgenden erläutert.

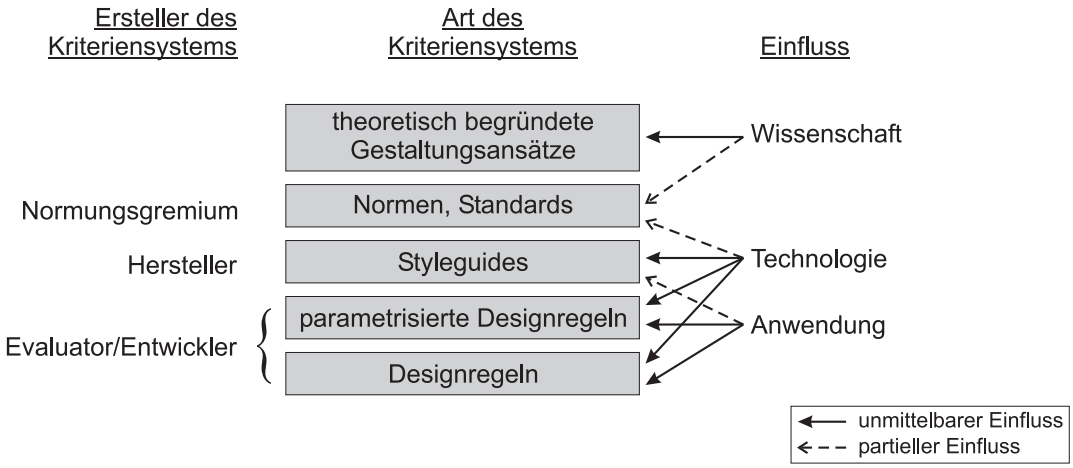

Abbildung 2.3: Teilnehmer und Einfluss bei der Erstellung unterschiedlicher Kriteriensysteme.

Theoretisch begründete Gestaltungsgrundsätze sind das Ergebnis wissenschaftlicher Arbeiten und bilden häufig die Grundlage für **Normenwerke** und **Standards**. Letztere sind meist konkret auf eine Anwendung in einer Domäne ausformuliert. Die der Domäne zu Grunde liegende Technologie steht dabei jedoch nicht im Fokus und wird daher lediglich im Ansatz berücksichtigt (TIMPE et al. 2002). Verantwortlich dafür sind Normungsgremien oder -komissionen wie z.B. das Deutsche Institut für Normung e.V. (DIN)[3] oder die International Organization for Standardization (ISO)[4].

Styleguides werden von Herstellern interaktiver Systeme entwickelt. Sie enthalten neben den gültigen Kriterien von Normen und Standards zusätzliche Faktoren wie *Corporate Identity* sowie technologische Aspekte, die vor allem eine konsistente Gestaltung des GUI vorschreiben (SCAPIN et al. 2001a).

[3] http://www.din.de
[4] http://www.iso.org

(parametrisierte) Designregeln werden von Entwicklern oder Evaluatoren erstellt. Da diese bis ins Detail ausformulierten Regeln für konkrete Systemobjekte oder -bereiche gelten, berücksichtigen sie in vollem Maße sowohl den Anwendungskontext als auch Technologieaspekte (TIMPE et al. 2002).

Der Prozess der Erstellung von Guidelines ist nicht festgelegt, da die zu betrachtenden Anwendungsfelder sowie die Abstraktionsebenen (vgl. Abb. 2.2) zu unterschiedlich sind, als dass konkret eine allgemein gültige Methodik definiert werden könnte (TATZLAFF und SHWARTZ 1991). Das prinzipielle Vorgehen bei der Erstellung von Guidelines lässt sich in drei Phasen einteilen. Die *Vorbereitungsphase* umfasst die Festlegung der Domäne und der Art der Guideline (gemäß Abb. 2.2) sowie das Sammeln und Aufbereiten bestehenden Wissens. In der *Erstellungsphase* erfolgt die Ausformulierung der Guideline, deren Anwendbarkeit in der *Evaluierungsphase* hinsichtlich der Abdeckung technologischer und systemspezifischer Aspekte an bereits fertigen Produkten bewertet wird. Eine ausführliche Beschreibung der Entwicklung von Guidelines bieten (O'HARA 1993; BONNER 1997; TROMP et al. 2004).

Der folgende Abschnitt stellt das allgemeine Vorgehen der Anwendung von Guidelines zur automatischen Bewertung in zwei Schritten vor: die Auswahl für die Bewertung relevanter Guidelines sowie die Implementierung der Guidelines als automatisch überprüfbare Regeln.

2.2.2 Anwenden von Kriterien

Nahezu alle Kriteriensysteme und Guidelines liegen heutzutage als gedruckte oder elektronische Dokumente vor (PARUSH 2001). Ebenso unterschiedlich wie Umfang und Abstraktionsgrad von Guidelines (s. Kap. 2.2.1) sind die Anwendungsmöglichkeiten und Vorgehensweisen.

Aufgrund der großen Menge existierender Guidelines muss für die Bewertung eines interaktiven Systems eine Auswahl getroffen werden. Eine allgemein gültige Vorgehensweise ist von Limbourg und Vanderdonckt in (LIMBOURG und VANDERDONCKT 2001) ausführlich beschrieben.

Zur automatischen Überprüfung auf die Einhaltung von Kriterien wird im Folgenden eine allgemeine datengetriebene[5] Methodik vorgestellt. Die fünf dazu notwendigen Schritte lehnen sich an die Methode von Beirekdar et al. an, sind in Abbildung 2.4 dargestellt und werden im Folgenden erklärt (BEIREKDAR et al. 2002).

[5] Im Bereich der Expertensysteme (s. Glossar) basiert der *datengetriebene* Ansatz auf dem Vorhandensein und der Überprüfung von Eigenschaften von Daten. Im Gegensatz dazu überprüft der *zielgetriebene* Ansatz ob ein vorgegebenes Ziel durch bestehende Daten erreicht werden kann oder nicht. Vgl. dazu die Beschreibung von *Inferenz* im Glossar.

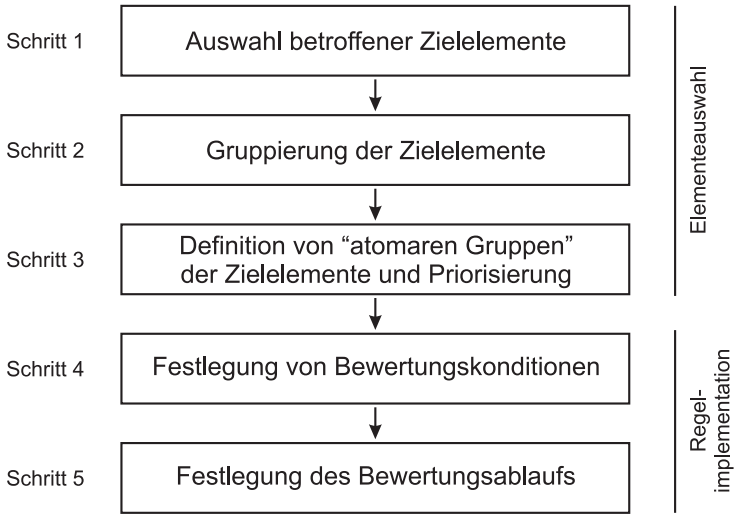

Abbildung 2.4: Vorgehen bei der Anwendung von Guidelines zur Bewertung interaktiver Systeme.

Zum besseren Verständnis soll beispielhaft eine formale Beschreibung eines GUI auf die spektrale Distanz der benutzten Farben[6] überprüft werden. Die zu implementierende Regel lautet:

> Farbgebung - Spektrale Distanz (Rot-Blau-Kombination):
> Farben mit zu großer spektraler Distanz dürfen nicht kombiniert werden. [...]
> Rote und blaue Bilder sind vom Auge nicht gleichzeitig scharf darstellbar.
> (OSACA E.V. 1997)

Die Syntax der GUI-Beschreibung gibt in diesem Beispiel (s. Abb. 2.5 auf Seite 28) die GUI-Elemente durch ihren Typ gefolgt von Attribut-Wert-Paaren an. Es existiert der Typ `geo-element`[7] für Linien, Vierecke, Kreise etc., sowie `text-element` für Texte (Beschriftung, Menüelement etc.). Folgendes Text-Element demonstriert das Format:

```
(text-element (id 4)(xpos 10)(ypos 10)(text \Titel\)(farbe blau)...)
```

Alle Elemente der GUI sind im Beispiel entsprechend beschrieben und liegen zur Bewertung vor.

Im **ersten Schritt** müssen diejenigen Systemelemente identifiziert werden, deren Attribute durch die Regeln überprüft werden sollen. Da die Regel die Farben benachbarter bzw. übereinander liegender Elemente überprüft, kommen dafür alle Elemente in Frage, die das Attribut `farbe` aufweisen, in obigem Beispiel sind dies alle vorhandenen Elemente.

Anschließend werden im **zweiten Schritt** die zu überprüfenden Elemente gruppiert, wobei

[6] Die Farbgebung eines GUI lässt sich hinsichtlich mehrerer Aspekte hin überprüfen. Ein Aspekt berücksichtigt den Effekt der chromatischen Aberration im menschlichen Auge. Dabei werden kurze Wellenlängen stärker gebrochen als lange. Kombinationen mit Farben stark unterschiedlicher Wellenlänge (z.B. rot und blau) werden aus diesem Grund als unangenehm empfunden, führen schneller zur Ermüdung und sollten daher vermieden werden (GOLDSTEIN 2002).

[7] `geo-element` als Abkürzung für *geometrisches Element*.

Elemente zusammengefasst werden, deren Attribute gleiche Eigenschaften haben sollen. Im Beispiel ist eine Aufteilung der GUI-Elemente vom Typ `geo-element` in Linien- und Flächenelemente sinnvoll, da nur eine Kombination von Flächen in den Farben Rot und Blau ein Problem bei der Fokussierung des Auges darstellt. Das Ergebnis ist im Beispiel die Bildung von drei Gruppen: *Linienelemente, Flächenelemente, Textelemente*.

Ziel des **dritten Schritts** ist die Herausbildung von „atomaren Gruppen“ aus den im zweiten Schritt gebildeten Gruppen. Dabei erlaubt eine atomare Gruppe per Definition die präzise und eindeutige Formulierung eines Ziels der Regel (BEIREKDAR et al. 2002). Aufgrund des geringen Umfangs des Beispiels bilden alle drei Gruppen (*Linienelemente, Flächenelemente, Textelemente*) aus dem zweiten Schritt bereits atomare Gruppen.
Darüber hinaus wird in diesem Schritt den jeweiligen Gruppen eine Priorität für den Ablauf der Bewertung zugewiesen. Diese Priorität steuert später die Reihenfolge der Überprüfung (s. Schritt 5), wodurch die Geschwindigkeit der Regelüberprüfung gesteigert werden kann. Schlägt die Überprüfung einer hoch priorisierten Gruppe bereits fehl, brauchen niedriger priorisierte Gruppen nicht mehr überprüft zu werden.

Die Überprüfung der Farbgebung von Text- und Flächenelementen ist im oben stehenden Beispiel hoch priorisiert, da gerade bei Texten der Fehler der chromatischen Aberration besonders schwer wiegt. Linien werden üblicherweise nur zum Design eines GUI eingesetzt, sind daher nicht informationstragend, so dass deren Farbgebung hinsichtlich der chromatischen Aberration nicht so schwer wiegt wie bei Texten. Daher bekommt die Überprüfung von Linienelementen eine niedrige Priorität. Die Regel lässt sich aus o.g. Gründen sinnvollerweise auf folgende Kombinationen der Menge anwenden:

Textelemente ↔ Flächenelemente (Priorität 1)
Flächenelemente ↔ Flächenelemente (Priorität 2)
Linienelemente ↔ Text-/Flächenelemente (Priorität 3)

Im **vierten Schritt** erfolgt die detaillierte Festlegung von Bewertungsregeln. Das beinhaltet die konkrete Überprüfung der Werte einzelner Attribute der atomaren Mengen. Der Aufbau der Kondition erfolgt als Wenn-Dann-Regel, wobei im Wenn-Teil die Abfrage der Attribute und im Dann-Teil die Reaktion auf die Abfrage erfolgt (KÖPPEN et al. 2001). Die Bedingungen im Wenn-Teil können Variablen beinhalten sowie kombiniert werden. Dazu stehen die logische Verknüpfungen `nicht`, `und` und `oder` zur Verfügung.

Aus obigem Beispiel folgt die Implementierung der Regel zur Überprüfung des Farbkontrasts in Abbildung 2.5. Es wird die Existenz (`existiert`) eines Text- und eines Flächenelements geprüft, wobei *`id1`*, *`id2`*, *`farbe1`* und *`farbe2`* Variablen sind und zur Laufzeit mit den Werten der entsprechenden Attribute belegt werden. Die Funktion `nachbar` überprüft, ob die Elemente nebeneinander liegen oder überlappen, da nur bei solchen Elementen ein Problem der Fokussierung auftreten kann. Die letzte Bedingung prüft, ob die Elemente die Farben Rot bzw. Blau haben. Sind alle Bedingungen für zwei Elemente erfüllt, erfolgt eine Fehlermeldung. Die Implementierung der beiden weiteren Regeln zur Überprüfung der Farbgebung von Text- und Flächenelementen erfolgt analog.

```
WENN
     existiert (text-element (id id1) (farbe farbe1))
     UND existiert (flaechen-element (id id2) (farbe farbe2))
     UND nachbar(id1, id2)
     UND (
            (farbe1=blau UND farbe2=rot)
            ODER (farbe1=rot UND farbe2=blau)
          )
DANN Fehlermeldung
```

Abbildung 2.5: Beispiel einer Regel zur Überprüfung des Farbkontrasts zweier GUI-Elemente in Pseudocode.

Der **fünfte Schritt** dient der Steuerung der Regelbearbeitung gemäß der im dritten Schritt gesetzten Prioritäten. Oben stehende Regel überprüft die Farbgebung von Text- und Flächenelementen (Priorität 1 – s. Schritt 3). Erst wenn durch diese Regel keine Fehlermeldung generiert wurde, erfolgt die Überprüfung mit der nächst höheren Priorität, in diesem Beispiel die Kombination von Flächenelementen (Priorität 2). Erst wenn dort ebenfalls keine Fehlermeldung auftritt, werden Linienelemente überprüft (Priorität 3).

Der Einsatz der priorisierten Überprüfung, gesteuert durch die Festlegung im dritten und fünften Schritt, erlaubt eine deutlich beschleunigte Bewertung. Weiterhin werden durch dieses Vorgehen Fehlermeldungen[8] in Abhängigkeit der Schwere der Fehler generiert, so dass die Ergebnisse der Regelüberprüfungen übersichtlich bleiben und eine schnelle Fehlerbehebung ermöglichen (BEIREKDAR et al. 2002).

2.3 Werkzeugunterstützung bei der kriterienorientierten Bewertung

Am häufigsten wird eine Bewertung der Gebrauchstauglichkeit interaktiver Systeme mit Hilfe von Guidelines durchgeführt (LANDAUER 1988; OED et al. 2001; TIMPE et al. 2002). Es sind ausreichend Guidelines für eine umfassende Bewertung der Gebrauchstauglichkeit interaktiver Systeme vorhanden, da das meiste Wissen über Gebrauchstauglichkeit heutzutage bereits derart erfasst ist (REITERER 2001). Daher bringen Werkzeuge, die eine (Teil-)Automatisierung der kriterienorientierten Bewertung bieten können, einen signifikante Verkürzung des Bewertungs- und Entwicklungsaufwands interaktiver Systeme und somit eine Verringerung der dazu benötigten Ressourcen (SMITH 1988; FARENC 1997).

Die Möglichkeiten der Automatisierung reichen von Werkzeugen zur einfachen Hilfestellung bei der Verwaltung über Werkzeuge zur Analyse des Systemdesigns bis hin zu Werkzeugen

[8] Mit diesem Ansatz ist die Generierung eines Alarms möglich. Die Möglichkeit der Modellierung mehrerer Alarmstufen in Abhängig der Schwere des Fehlers ist im Ausblick diskutiert (s. Abschnitt 7).

zum automatischen Erstellen einer Benutzungsoberfläche. Da zahlreiche Bewertungswerkzeuge existieren, ist im Folgenden eine repräsentative Auswahl gegeben; Stary et al., Ivory, Mourouzis et al. und Gomaa et al. bietet darüber hinaus sehr umfangreiche Übersichten (STARY et al. 1995; IVORY und HEARST 2001; MOUROUZIS et al. 2004; GOMAA et al. 2005).

2.3.1 Werkzeuge zur Guidelineverwaltung

Da die Kriterienkataloge im Lauf der letzten Zeit immer umfangreicher werden, liegt eine einfache Möglichkeit der Unterstützung der kriterienorientierten Bewertung in der reinen Verwaltung elektronisch vorliegender Kriterien, wobei die Durchführung der Bewertung den Experten überlassen wird (VOGT 2001; SCAPIN et al. 2001b). Dazu können für jede elektronische Guideline virtuelle Karteikarten angelegt werden, die u.a. folgende Informationen beinhalten (LIMBOURG und VANDERDONCKT 2001):

- den **Titel** der Guideline
- eine kurze **Beschreibung** des Inhalts
- die **Referenz/Quelle** (z.B. Buch, URL)
- die **Organisation** bzw. der **Ersteller** der Guideline
- das **Datum** des Erscheinens
- den **Typ** der Guideline (s. Abb. 2.2)
- **Beispiel(e)** zur Anwendung

Weiterhin ist es möglich, die Kriterien in Gruppen zusammenzufassen. Dabei ist eine Gruppierung sowohl nach Art der Aussage der Kriterien (value based) als auch nach Art des zu bewertenden Systemobjekts (object based), z.B. Menüelement oder Schalter, sinnvoll (STONEBRAKER et al. 1989). Der Zugriff auf die Kriterien erfolgt mittels einer Suchmaske, wie sie in den meisten Datenbankanwendungen üblich ist (LIMBOURG und VANDERDONCKT 2001).

Da die Werkzeuge zur Guidelineverwaltung keine automatische Überprüfung vorliegender Guidelines erlauben, wird auf sie nicht im Detail eingegangen. Der Vollständigkeit halber erfolgt lediglich eine kurze Übersicht der wichtigsten Werkzeuge:

ISO 9241-Evaluator

Auf Basis eines Expertensystems stellt dieses Werkzeug die Konformität mit der ISO 9241 sicher. Die Kriterien der ISO sind in 300 *Test-Items* abgebildet, die wiederum jeweils einem technischen Aspekt der Komponente (z.B. Eingabemodalität, Fehlerbehandlung) und ei-

nem ergonomischen Prinzip aus dem Teil 10 der ISO[9] (z.B. Selbstbeschreibungsfähigkeit, Steuerbarkeit) zugeordnet sind (OPPERMANN und REITERER 1997). Jedes Test-Item enthält ausführliche Erklärungen des entsprechenden Kriteriums und dessen Überprüfung.

Das Werkzeug gibt dem Bewerter die Möglichkeit, in Freitext Beschreibungen von Systemeigenschaften sowie Screenshots zu den Test-Items hinzuzufügen.. Das Ergebnis einer Bewertung ist ein textueller Report mit statistischen Angaben (REITERER 1995).

Inra (Interface Ratgeber)

Inra ist ein an der HU-Berlin entwickeltes Hypertext-basiertes Informationssystem (WANDKE und HÜTTNER 2001). Es enthält Gestaltungsregeln und -beispiele zu Dialogtechniken, zur Präsentation von Information auf Bildschirmen, zu Hilfe- und Unterstützungsfunktionen und zum Fehlermanagement. Die Abfrage der Informationen erfolgt manuell und die Darstellung der Ergebnisse ist statisch (WANDKE et al. 2001).

GUIDEAS (Guidance for Developing Assistance)

Das Werkzeug[10] ist Teil des Embassi-Projektes[11] und stellt ein Expertensystem zur Erstellung des Oberflächen- und Informationsdesigns von Assistenzsystemen dar. Zu Beginn macht der Systementwickler Angaben über die Benutzer des Systems, die Aufgaben, die sie damit ausführen, die situativen Bedingungen und die Ziele, die mit der Assistenzfunktion verfolgt werden (TSCHÖPE und NITSCHKE 2002). Auf dieser Basis generiert das in GUIDEAS implementierte Expertensystem unter Verwendung von gewichteten Wenn-Dann-Regeln einen Vorschlag zur Gestaltung der Assistenzfunktion. Die Vorschläge geben an, welche Funktionen Assistenz erfordern und mit welchen Eingabemodalitäten und welcher Darstellung im Rahmen des Oberflächen- und Interaktionsdesign dieses unterstützt werden soll (NITSCHKE und WANDKE 2002; GRAF et al. 2004). Eine Erweiterung der Regelbasis ist nicht vorgesehen.

2.3.2 Werkzeuge zur automatischen Überprüfung von Benutzungsoberflächen anhand von Guidelines

Grundlage einer automatischen Bewertung der Gebrauchstauglichkeit interaktiver Systeme auf Basis von Guidelines ist eine formale Beschreibung des zu bewertenden Systems. Weiterhin müssen die Kriterien in einer Form vorliegen, die eine elektronische Verarbeitung und Überprüfung erlaubt. Eine Methode dazu ist bereits in Abschnitt 2.2.2 vorgestellt worden.

[9] Der Teil 10 der ISO 9241 befindet sich momentan (2006) in der Überarbeitung. Die aktuellste Version ist daher mit „Teil 110“ betitelt (ISO 9241-110 2004).

[10] Eine Demo-Version von GUIDEAS ist unter http://141.20.70.200/Guideas/ verfügbar.

[11] http://www.embassi.de

KRI/AG

Das Werkzeug KRI/AG (Knowledge-based Review of user Interface) wurde 1992 von Löwgren und Nordqvist vorgestellt (LÖWGREN und NORDQVIST 1992). Die Oberfläche einer Software für X-Windows-Systeme kann bewertet werden, wenn ihr GUI mit dem Werkzeug TeleUSE erstellt wurden, was die Anwendungsmöglichkeiten stark einschränkt. In KRI/AG sind die Guidelines von Smith und Mosier (SMITH und MOSIER 1986) sowie die Motif Styleguides[12] implementiert.

Eine formale Beschreibung des Softwaredialogs, die TeleUSE generiert, wird in KRI/AG auf Konformität mit den Guidelines überprüft. Ein Bewertungsbericht gibt Auskunft über mögliche Fehler im Oberflächen- und Informationsdesign der Software.

Basis der Bewertung bildet ein Expertensystem, in dem die Regeln fest in Form einer proprietären Programmiersprache implementiert sind. Eine Eingabe neuer oder Änderung bestehender Regeln kann nur im Sourcecode selbst stattfinden (LÖWGREN et al. 1996). Durch die Implementierung der Guidelines als funktionale Regeln[13] kann das Werkzeug auch sehr umfangreiche Dialoge bewerten.

Das Werkzeug wurde erfolgreich an einem Dialog zur Darstellung einer taktischen Landkarte im Rahmen einer militärischen Software eingesetzt. KRI/AG konnte Fehler im Oberflächen- und Informationsdesign, z.B. in Form von fehlenden Beschriftungen, entdecken.

ErgoVal

Mit Fokus auf dem strukturellen Aufbau einer Benutzungsoberfläche bietet das Werkzeug *ErgoVal*, das 1997 von Farenc an der Universität Toulouse vorgestellt wurde, die Überprüfung auf Guideline-Konformität (FARENC 1997). Ziel des Werkzeuges ist die Bewertung des Oberflächendesigns von Software, die mit Microsoft-Visual-Tools erstellt wurde.

Nach dem Einlesen des zu bewertenden Softwaredialogs in ErgoVal erfolgt eine Dekomposition der Bildschirmelemente. Dabei werden die Elemente und deren Eigenschaften nach einer fest vorgegebenen objektorientierten Struktur klassifiziert. Beispielhaft ist ein Auszug der gegebenen Struktur (GUI-Elemente und ihre Zusammenhänge) in Abbildung 2.6 gegeben. Dabei sind mehrere Relationen der angegebenen Elemente zueiander möglich. So kann ein Element ein anderes enthalten (örtliche Beziehung - *Localisation relation*). Die *Aggregation relation* gibt an, wenn ein Element eine (oder mehrere) Instanzen eines anderen Elementes enthalten kann, während die *Semantic relation* alle Elemente beinhaltet, durch die das Aussehen und das Verhalten eines Elementes definiert wird.

Den einzelnen Elementen der objektorientierten Struktur lassen sich gezielt Kriterien zuordnen, die überprüft werden können. Farenc hat dazu beispielhaft 205 Kriterien ausgewählt und in ErgoVal integriert (FARENC und PALANQUE 1999).

Ein Editor erlaubt das Erstellen und Bearbeiten der Kriterien. Allerdings ist eine Zuordnung

[12] Mittlerweile werden die Motif Styleguides vom Gnome Projekt weiter entwickelt: http://www.gnome.org
[13] s. Glossar

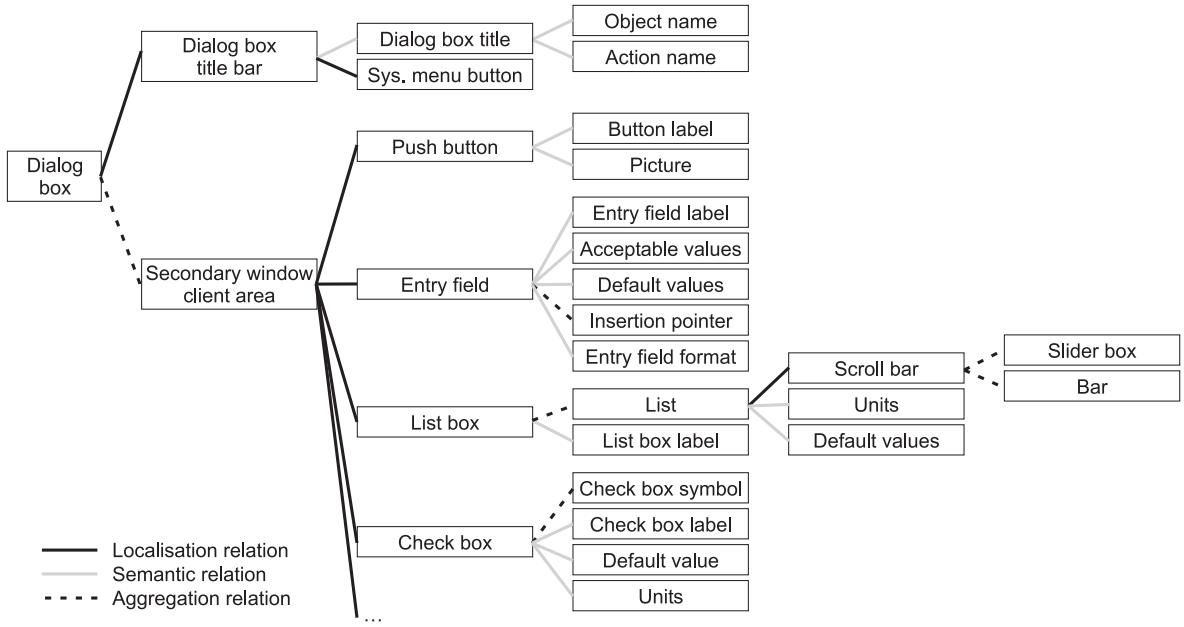

Abbildung 2.6: Dekomposition der UI-Elemente im Werkzeug ErgoVal (FARENC und PALANQUE 1999).

zu einem konkreten Bildschirmelement bzw. Attribut zwingend vorgegeben. Das schränkt die Möglichkeiten der Regeln deutlich ein. So lässt sich beispielsweise die Schriftgröße eines *button label* (s. Abb. 2.6) überprüfen, wobei unabhängig davon die Überprüfung der Schriftgröße eines *entry field label* ebenfalls explizit erfolgen muss. Es ist nicht möglich, allgemeine Aussagen über die Schriftgrößen unabhängig von einem speziellen Element zu machen.

Das Format der Regeln sowie die Erklärung, wie der zu bewertende Softwaredialog in ErgoVal importiert wird, ist nicht veröffentlicht. ErgoVal wurde mehrfach erfolgreich zur Bewertung und Verbesserung von Softwaredialogen bei der Französischen Post eingesetzt (FARENC und PALANQUE 1999).

Sherlock

Sherlock ist ein Werkzeug zur Verwaltung und automatischen Überprüfung von Guidelines und wurde am FORTH Institute of Computer Science in Griechenland 2000 entwickelt (GRAMMENOS et al. 2001). Ziel von Sherlock ist die Bewertung des Oberflächendesigns von Software, deren Benutzungsoberfläche in Visual Basic implementiert ist. Der Aufbau des Werkzeugs ist in Abbildung 2.7 dargestellt.

Zentrale Komponente der Architektur ist der *Sherlock-Server*. Er verwaltet die Guidelines sowie deren Anwendung. Regeln müssen in Microsoft Visual C++ oder Visual Basic implementiert sein und werden dynamisch als ActiveX-DLL[14] beim Server eingebunden. Das bietet die größtmögliche Flexibilität bei der Implementierung, da auch umfangreiche und komplizierte Berechnungen von Maßen möglich sind. Allerdings erfordert jede Änderung eine Neucompilierung des Sourcecodes. Dementsprechend muss der Bewerter umfangreiche Kenntnisse in

[14] s. Glossar

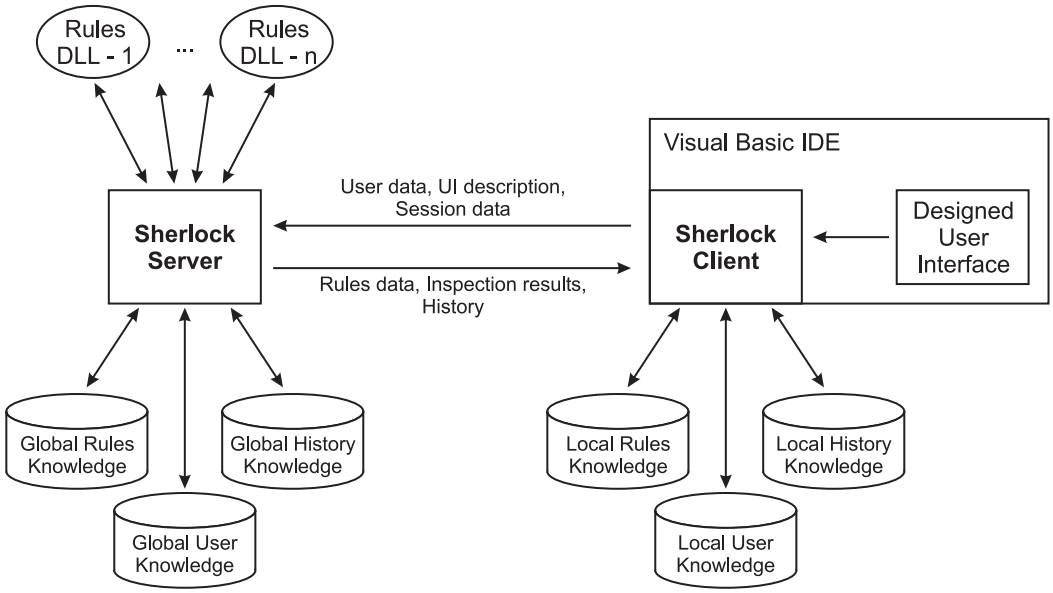

Abbildung 2.7: Architektur des Werkzeugs Sherlock (aus (GRAMMENOS et al. 2000)).

einer der Programmiersprachen haben.

Die IDE der Microsoft Entwicklungsumgebung Visual Basic wird um den Sherlock-Client erweitert. Dort hat der Entwickler die Möglichkeit, die beim Server verfügbaren Regeln für eine Bewertung auszuwählen. Anschließend wird der Visual Basic-Sourcecode der entwickelten Software (inkl. der GUI-Beschreibung) an den Server geschickt. Der Server wendet die Regeln auf die Daten an und sendet dem Client die Bewertungsergebnisse. Dabei handelt es sich um textuelle Hinweise mit Direktverweisen auf das betroffene GUI-Element.

So kann z.B. überprüft werden, ob in einem Dialog der „OK“-Button links vom „Cancel“-Button angeordnet ist. Ist dies nicht der Fall, macht der Client den Vorschlag, die Buttons zu tauschen. Diese Diagnose ist möglich, da einem Button u.a. der Status „OK“ oder „Cancel“ direkt zugewiesen werden kann. Durch die Integration in die Visual Basic IDE bietet der Client dem Entwickler die Möglichkeit, die entsprechende Komponente direkt zu bearbeiten. Eine detaillierte Beschreibung der Komponenten und deren Verwendung findet sich in (GRAMMENOS et al. 2000).

Die verteilte Struktur von Sherlock ermöglicht eine klare Trennung von Regelimplementation und -anwendung. Die Dialogbewertung ist komfortabel in die Entwicklungsumgebung integriert. Dadurch ist jedoch nur eine Bewertung von Dialogen möglich, die in Microsoft Visual-Programmiersprachen erstellt sind.

Kwaresmi

Zur automatischen Bewertung der Gebrauchstauglichkeit von Webseiten wird das Werkzeug *Kwaresmi*[15] an der Universität Namur entwickelt (VANDERDONCKT und BEIREKDAR 2005).

[15]Der Name weist auf den arabischen Mathematiker Al-Kwaresmi hin, der im zehnten Jahrhundert lebte und dessen Namen den Ursprung für das Wort *Algorithmus* bildet.

Ziel der Entwicklung ist die Trennung zwischen der Spezifizierung von Kriterien von deren Implementierung im Sourcecode durch den Einsatz einer abstrakten Kriterienbeschreibungssprache, der *Guideline Definition Language (GDL)* (BEIREKDAR et al. 2005). Das zu Grunde liegende Konzept ist in Abbildung 2.8 dargestellt.

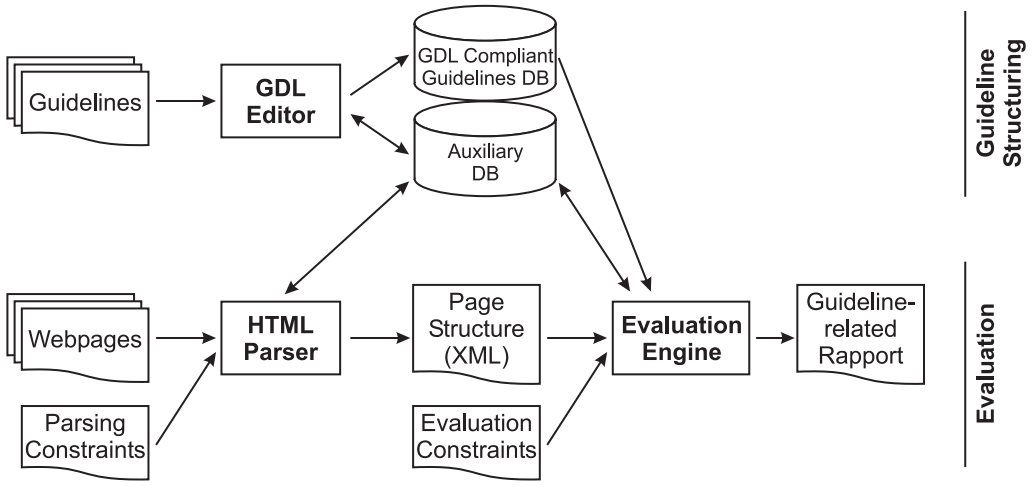

Abbildung 2.8: Konzept des Werkzeugs Kwaresmi (aus (BEIREKDAR et al. 2003)).

Ein Editor ermöglicht das Eingeben und Verwalten der Regeln, die in einer Datenbank gespeichert sind. Zur Bewertung wird der Inhalt einer Webseite mit allen Darstellungsattributen in eine XML-basierte Form übersetzt. Anschließend wendet die Bewertungsmaschine (*evaluation engine*) die in der GDL vorliegenden Regeln auf die Informationen der Webseite an. Ein Bericht über die Übereinstimmung der überprüften Inhalte auf Regelkonformität bildet das Ergebnis der Bewertung.

Die Überprüfung erfolgt überwiegend auf syntaktischer Ebene. Der HTML-Code[16] einer Webseite enthält zwar Darstellungsattribute von Informationen (Texte, Bilder), die endgültige Darstellung (z.B. Anordnung von Elementen, Größe von Schrift) bleibt jedoch dem Webbrowser überlassen und kann daher nur bedingt überprüft werden (IVORY et al. 2001). Durch die starke Einschränkung auf die Überprüfung von HTML-Attributen lassen sich ebenfalls keine Regeln für die Überprüfung mehrerer Elemente implementieren, wie z.B. die die Überprüfung auf Unterschiede im Farbkontrast von Elementen. Dies muss für alle möglichen Kombinationen einzeln implementiert werden.

Eine zusätzliche Datenbank (*Auxiliary DB*, vgl. Abb. 2.8) speichert u.a. Typenbezeichnungen von HTML-Elementen, um einen eindeutigen Zugriff auf diese Elemente sowohl für den Editor, also auch für den Parser und die Bewertungsmaschine zu gewährleisten. Zudem können Vorgabewerte in der Datenbank gespeichert werden, die sich ebenfalls durch Regeln mit den entsprechenden Werten der Webseite vergleichen lassen (z.B. eine Tabelle mit Vorder- und Hintergrundfarben mit gutem Kontrastverhältnis).

Genauso wie im Werkzeug ErgoVal ist es lediglich möglich, solche Regeln in der GDL zu

[16]HTML ist ein Akronym für *Hypertext Markup Language*, dem Format, in dem Webseiten implementiert sind.

beschreiben, die sich auf Attribute von genau einem konkreten HTML-Element beziehen. Regeln, die zur Überprüfung der Farbgestaltung unabhängig vom jeweiligen Element lediglich das Attribut *color* betrachten, sind nicht formulierbar (BEIREKDAR et al. 2003).

Kwaresmi (sowie das Nachfolgewerkzeug *Destine*, in dem Kwaresmi als Bewertungsmodul integriert ist) wurde mehrfach erfolgreich zur Bewertung von Internetseiten eingesetzt (BEIREKDAR et al. 2005).

2.3.3 Werkzeuge zur automatischen Erstellung von Benutzungsoberflächen

Die umfassendste Unterstützung bieten Werkzeuge, die auf Basis von Guidelines in der Lage sind, selber GUIs zu erstellen oder zu optimieren. Da Werkzeuge zur automatischen Erstellung von Benutzungsoberflächen deterministisch sind, werden sie mit gleicher Regelbasis auch stets ein gleiches Ergebnis erzeugen. Allerdings kann es durchaus mehrere „gute“ Designs geben (LÖWGREN und NORDQVIST 1992). Daher können automatisch erstellte Benutzungsoberflächen lediglich als Grundlage für ein Design verstanden werden (VICENTE et al. 1997). Der Fokus einer Werkzeugunterstützung sollte daher vor allem in der automatischen Überprüfung interaktiver Systeme auf Konformität mit Guidelines bzw. auf deren Verletzung liegen (VOGT 2001). Aus diesem Grund werden im folgenden lediglich zwei repräsentative Werkzeuge vorgestellt.

AIDE (semi-Automated Interface Designer and Evaluator)

Zur Bewertung und Erstellung von Designvorlagen für Softwareprogramme wurde 1995 von Sears das Werkzeug AIDE entwickelt (SEARS 1995). Das Werkzeug basiert u.a. auf Kriterien von Tullis, Kim und Foley zur Bewertung von Ausführungszeit und Suchzeiten zum Finden von Informationen und Eingabefeldern in Software-GUIs (TULLIS 1988; KIM und FOLEY 1993). Die Maße sind vollständig operationalisiert und werden auszugshaft im Folgenden aufgelistet:

- Fläche (in Prozent), die für die Informationsdarstellung benutzt wird
- vertikale und horizontale Symmetrie (*balance*) der Elemente
- vertikale und horizontale Verteilung (*alignment*) der Elemente
- Anzahl der Gruppen der Elemente

Die jeweiligen Maße werden gewichtet und zu einem Gesamtmaß addiert. Wichtigster Faktor der Formel ist dabei die Berechnung eines Maßes, das den Aufwand bei der Bedienung berücksichtigt. Sears nennt dieses Maß *Layout Appropriateness* und berechnet es als die Summe der Wegstrecke eines Zeigerelementes (z.B. einer Maus) von einem Bildschirmelement *i*

zu einem Bildschirmelement j (berechnet mit der Formel von Fitts[17]) multipliziert mit der Benutzungswahrscheinlicheit p_{ij} diesen Weges. Diese Wahrscheinlichkeiten müssen experimentell ermittelt und in AIDE eingegeben werden (SEARS 1993b).

Für die Beschreibung der Dialogelemente verwendet AIDE das Textformat der Interface-Entwicklungsumgebung *Simple User Interface Toolkit (SUIT)* (PAUSCH et al. 1991). Durch die Festlegung auf dieses wenig verbreitete, proprietäre Format sind die Anwendungsmöglichkeit von AIDE deutlich eingeschränkt. Eine Liste der Dialogelemente, die der Dialog enthalten muss, bildet die Grundlage für AIDE, mit der zufällig ein Initialdesign errechnet wird (s. linkes Beispiel in Abb. 2.9).

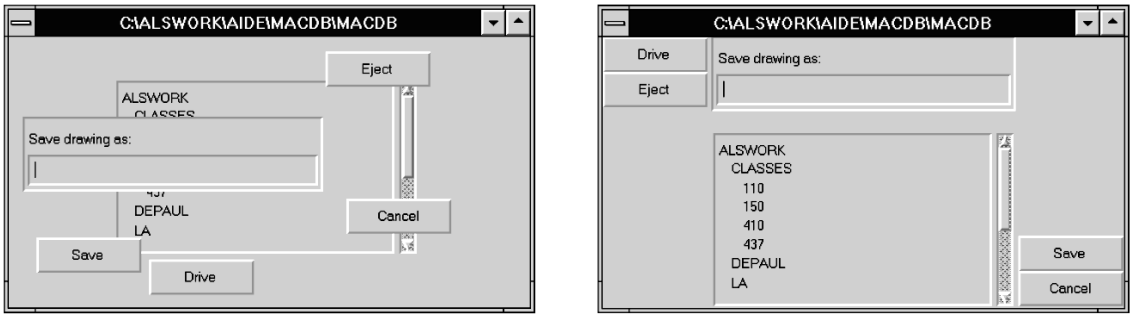

Abbildung 2.9: Beispieldialog im Initialzustand vor (links) und im Endzustand nach der Umordnung der Dialogelemente durch das Werkzeug AIDE (rechts).

Grundlage der Berechnung der *Layout Appropriateness* bildet die Angabe der Benutzungswahrscheinlichkeiten von Eingabeelementen, beispielhaft in Abb. 2.10 dargestellt. Auf Basis dieser Wahrscheinlichkeiten wird die Lage der Dialogelemente mit Hilfe eines Branch-and-Bound-Algorithmus iterativ angepasst, bis das Enddesign – das Design mit dem besten Gesamtmaß – erreicht ist (s. rechtes Beispiel in Abb. 2.9). Die Konsistenz der Lage von Dialogelementen über mehrere Dialoge hinweg wird dabei jedoch nicht berücksichtigt.

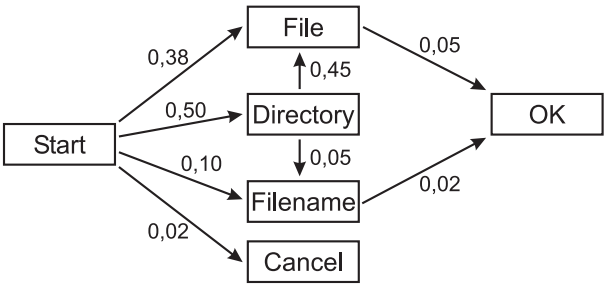

Abbildung 2.10: Baum der Wahrscheinlichkeiten, dass Benutzer von einem Dialogelement zu einem anderen wechseln.

AIDE wurde mehrfach zur Erstellung und Bewertung kleinerer Eingabedialoge eingesetzt (SEARS 1995). Allerdings eignet sich das Werkzeug nicht für die Erstellung umfangreicher Dialoge wie kompletter Programmoberflächen. Bedienungswahrscheinlichkeiten (zur Berechnung des *Layout Appropriateness*-Maßes) lassen sich nur für die Bearbeitung einer Aufgabe

[17] s. Glossar

angeben. Da umfangreiche Dialoge häufig den Zugriff auf mehrere Funktionen erlauben, ist eine Berechnung der entsprechenden Wahrscheinlichkeiten nicht möglich.

Ergo-Conceptor

Das Werkzeug *Ergo-Conceptor* zur automatischen Erstellung von Softwareoberflächen wurde von Moussa und Riahi an der Universität von Tunis entwickelt (MOUSSA et al. 2002). Das Ziel des Werkzeugs ist die Integration von Guidelines zur Dialoggestaltung. Der Prozess der Dialogerstellung ist in Abb. 2.11 dargestellt.

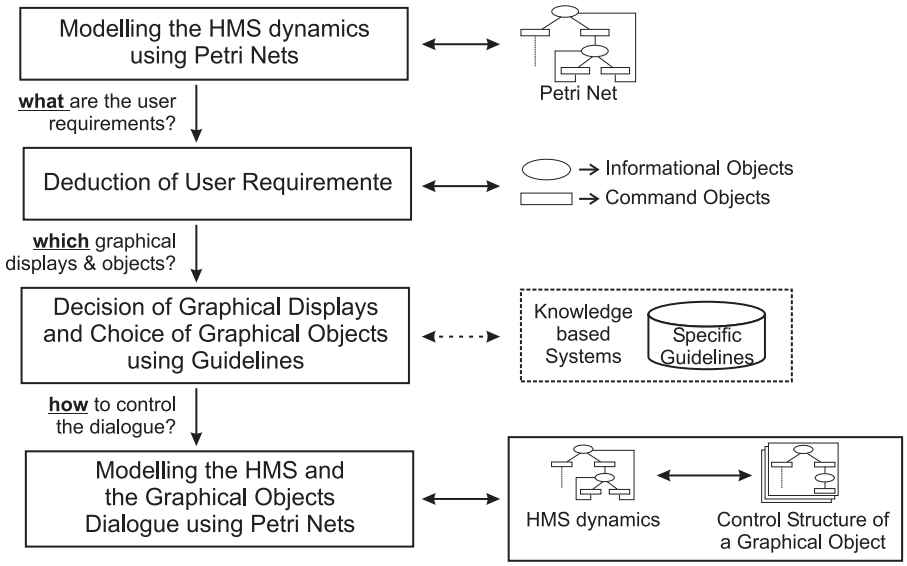

Abbildung 2.11: Prinzip der Guideline-basierten Dialogerstellung in Ergo-Conceptor (nach (MOUSSA und RIAHI 2001)).

Auf Basis einer umfangreichen Requirements-, Funktionen- und Aufgabenanalyse werden im ersten Schritt die Prozesse und möglichen Aufgabenbearbeitungen einer Software in Form von Petri-Netzen[18] manuell modelliert. Voraussetzung der Modellierung ist, dass sich jede interne Zustandsänderung des Systems auch in einer Änderung der Benutzungsoberfläche auswirkt.

In Petri-Netzen kann ein Zustandswechsel der *Stellen*[18] ein Schalten der *Transitionen*[18] auslösen. Aus diesem Grund werden im nächsten Schritt automatisch *Stellen* zu Informationsobjekten und *Transitionen* zu Kommandoobjekten zugeordnet.

Mit Hilfe eines wissensbasierten Systems, welches auf einer Guideline-Datenbasis aufbaut, erfolgt anschließend eine Zuweisung von Informationsobjekten zu Ausgabeelementen sowie von Kommandoobjekten zu Eingabeelementen. Ausgabe- und Eingabeelemente stellen konkrete grafische Komponenten eines Softwaredialogs dar.

Im letzten Schritt erfolgt die Generierung des grafischen Dialogs, wobei die Verbindungen zwischen Ausgabeelementen und *Stellen* sowie Eingabeelementen und *Transitionen* zwischen

[18] Petrinetze und deren Elemente sind im Glossar erläutert.

grafischen Dialog und Petri-Netz bestehen bleiben.

Ergo-Conceptor bietet die Möglichkeit der Eingabe neuer und Bearbeitung bestehender Guidelines, die grundsätzlich einen *Wenn-Dann*-Aufbau haben und logisch miteinander verknüpft werden können (MOUSSA et al. 2000).

Das Format der Regeln sowie Anwendungsbeispiele von Ergo-Conceptor sind nicht veröffentlicht. Durch die Verbindung von Petri-Netzen und Oberflächenelementen kann das Werkzeug mehr als Entwicklungs- denn als Bewertungswerkzeug verstanden werden, wenngleich die integrierten Regeln auf Aspekten der Gebrauchstauglichkeit aufbauen. Durch die enge Verknüpfung von Wenn-Dann-Regeln mit konkreten Oberflächenelementen ist es ebenso wie bei ErgoVal und Kwaresmi nicht möglich, Regeln unabhängig von konkreten Elementen allgemein z.B. zur Farbgestaltung anzugeben.

2.3.4 Vergleich der Werkzeuge zur kriterienorientierten Bewertung

Alle vorgestellten Werkzeuge eignen sich zur kriterienorientierten Bewertung interaktiver Systeme oder Software. Eine Übersicht ist in Tabelle 2.1 gegeben. Da die Werkzeuge *ISO 9241 Evaluator*, *Inra* und *Guideas* lediglich der Verwaltung von Kriterien dienen, sind sie in der Tabelle nicht berücksichtigt. Zum besseren Vergleich ist das in dieser Arbeit entwickelte Werkzeug REVISER ebenfalls aufgeführt.

Ein wesentlicher Unterschied der Werkzeuge zur automatischen Bewertung bzw. GUI-Erstellung liegt in der Art und Weise der Regeleingabe und -verarbeitung. In *KRI/AG*, *Sherlock* und *AIDE* sind die Regeln in Sourcecode (in der Programmiersprache C) programmiert. Ein Ändern bestehender oder Einfügen neuer Regeln erfordert einen Eingriff in den Programmtext des Werkzeuges selber (bis auf *Sherlock*, in dem die Regeln als Sourcecode-Bibliotheken in Form von DLL's eingebunden werden) sowie Kenntnisse in einer Programmiersprache. Deutlich mehr Flexibilität bieten dagegen *Ergo-Conceptor*, *ErgoVal* und vor allem *Kwaresmi*, die eine komfortable Regelbearbeitung mittels eines Editors erlauben sowie eine Syntax zur Formulierung der Regeln vorgeben. *ErgoVal*, *Kwaresmi* und *Ergo-Conceptor* schränken jedoch die Anwendungsmöglichkeiten ein, da sie ausschließlich vorgegebene Elemente und deren Attribute bewerten können und somit auf ein spezielles, domänenabhängiges Format festgelegt sind.

Die Kombination eines Editors zur schnellen Regeleingabe mit einer Möglichkeit, Sourcecode für die Programmierung umfangreicher Berechnungen einzubinden, bietet lediglich *Ergo-Conceptor*. Allerdings lassen sich nur Regeln integrieren, die der GUI-Erstellung dienen; Regeln zur Überprüfung von bestehenden Layouts oder zum Vergleich von Attributen der eingesetzten Oberflächenelemente (z.B. zur Bewertung der Farbgestaltung) lassen sich nicht einfügen, wodurch *Ergo-Conceptor* nicht zur Bewertung eines bestehenden Systemdesigns eingesetzt werden kann.

Ein weiterer Unterschied zwischen den Werkzeugen liegt in der Flexibiliät hinsichtlich der zu bewertenden Anwendung und Daten. Alle Werkzeuge sind auf eine konkrete Anwendungs-

domäne festgelegt. Das hat den Nachteil, dass die implementierten Bewertungsregeln nicht für weitere Domänen wiederverwendbar sind. Darüber hinaus schreiben die meisten Werkzeuge (*Sherlock*, *ErgoVal*, *Aide* und *KRI/AG*) das Format der einzulesenden Systemdaten vor. So sind z.B. in ErgoVal die Regeln zur Überprüfung des Designs von Software, die in Visual Basic implementiert wurde, nicht zur Überprüfung des Designs von Java-Software anwendbar, obwohl die zu Grunde liegenden ergonomischen Kriterien gleich sind. Dadurch ist die Anwendung dieser Werkzeuge deutlich eingeschränkt.

Eine Auswahl der Regeln, die zur Bewertung eingesetzt werden sollen, erlaubt nur *Sherlock*. Bei allen anderen Werkzeugen läuft die Bewertung statisch in der Form ab, dass alle Regeln abgearbeitet werden. *Kwaresmi* ist das einzige Werkzeug, das die Ausführungsreihenfolge durch die Angabe von Ausführungsprioritäten beeinflussen kann. Eine Auswahl der zu verwendenden Regeln in Verbindung mit der Festlegung der Ausführungsreihenfolge könnte die flexible Handhabung selbst großer Mengen von Regeln gewährleisten, wird aber von keinem der Werkzeuge geboten.

Alle Werkzeuge ermöglichen die Bewertung des Oberflächendesigns. Die Überprüfung des Informationsdesigns wird partiell von *KRI/AG*, *ErgoVal*, *Sherlock* und *Kwaresmi* angeboten. Dabei erfolgt meist lediglich die Überprüfung, ob eine Beschriftung eines Bedienelementes oder eines Dialoges vorhanden ist. Eine Überprüfung semantischer Beziehungen zwischen Begriffen (zur Bewertung eines Menüdesigns) bzw. eine Überprüfung auf Synonyme bietet keines der vorgestellten Werkzeuge. Das Interaktionsdesign wird lediglich in Ansätzen von zwei Werkzeugen betrachtet. *AIDE* berücksichtigt beim Oberflächendesign die Länge der Wege, die bei der Bedienung mit einer Maus im Dialog zurückgelegt werden müssen, und versucht, diese Wege zu minimieren. *Ergo-Conceptor* beinhaltet eine Datenbank von Eingabeelementen, die den *Transitionen* von Petri-Netzen regelbasiert zugeordnet werden. Allerdings wird die Anordnung dieser Eingabeelemente im Dialog z.B. zur Unterstützung konsistenter Bedienfolgen nicht weiter überprüft. Eine Überprüfung des Einsatzes von Bedienstereotypen sowie eine Abschätzung der Gedächtnisbelastung ist in keinem der Werkzeuge möglich.

Die Anwendung der meisten Werkzeuge liegt in der Überprüfung von Softwaredialogen (bzw. Webseiten bei *Kwaresmi*). Eine automatische Überprüfung der Benutzungsoberflächen von interaktiven Geräten wie z.B. Mobiltelefonen oder Fahrerinformationssystemen ist mit keinem der vorgestellten Werkzeuge vorgesehen, da durch die Festlegung auf ein bestimmtes Datenformat der Dialogbeschreibung eine domänenfremde Anwendung der Werkzeuge ausgeschlossen ist. Eine Abstraktion vom Datenformat der Dialogbeschreibung könnte diesen Umstand beheben, ist aber in keinem Werkzeug realisiert. Somit ist die Bewertung des Systemdesigns interaktiver Systeme bislang nicht automatisch, sondern lediglich manuell, z.B. mit dem *ITG-Prüfverfahren* oder der *MMI-Prüfliste*, möglich (VDE/ITG 1995; BLUM und NIRSCHL 2000).

In Tabelle 2.2 ist dargestellt, inwieweit die bestehenden Werkzeuge die geforderten Eigenschaften eines Werkzeugs zur automatischen kriterienorientierten Bewertung erfüllen. Das in dieser Arbeit entwickelte Werkzeug *Reviser* ist zum Vergleich ebenfalls aufgeführt.

Tabelle 2.1: Übersicht der Werkzeuge zur kriterienorientierten Bewertung interaktiver Systeme.

	KRI/AG (1994)	ErgoVal (1997)	Sherlock (2000)
Art des Werkzeugs	Bewertung	Bewertung	Bewertung
Format der Kriterien	Sourcecode (proprietär)	n. v.[a]	Sourcecode (ActiveX)[b]
Editor zur Kriterien-eingabe und -bearbeitung	nein	ja	nein
Einbindung von Programmcode	nein	ja	ja
Bewertungsablauf steuerbar	nein	nein	nein
Rechner-plattform	Unix	MS Windows	MS Windows
Anwendungsdomäne	Software (X Windows)	Software (MS Visual)	Software (Visual Basic)
Bereich der Bewertung	UI	UI	UI
Anzahl der integrierten Guidelines/Kriterien	70 Guidelines 30 Funktionen[c]	441	beliebig[e] (n. v.[d])
Präsentation der Bewertungsergebnisse	textuell (+ Beispiele)	textuell	textuell
Fokus der eingesetzten Kriterien	Oberflächendesign	Oberflächendesign Informationsdesign	Oberflächendesign

[a] keine Informationen verfügbar.

[b] Die Kriterien werden zwar in Sourcecode programmiert (ActiveX), die Schnittstelle zu Sherlock ist jedoch fest vorgegeben, so dass bestimmte Angaben wie z.B. Beschreibung der Guideline, Bewertungstyp (automatisch, semi-automatisch, manuell) und Literaturreferenzen angegeben werden müssen (GRAMMENOS et al. 2000).

[c] Als *Funkion* wird hier „prozedurales Domänenwissen“ verstanden, welches Funktionswerte berechnet (LÖWGREN und NORDQVIST 1992).

[d] Die Anzahl der tatsächlich zum Einsatz kommenden Kriterien war nicht verfügbar.

Kwaresmi (2005)	AIDE (1995)	Ergo-Conceptor (2002)	**REVISER** (2006)
Bewertung	automatische GUI-Erstellung	automatische GUI-Erstellung	Bewertung
proprietäre Regelsprache (GDL)	Sourcecode	Wenn-Dann-Aufbau (proprietär)	Wenn-Dann-Aufbau (LISP)
ja	nein	ja	ja
nein	nein	ja	ja
partiell (Angabe v. Prioritäten)	nein	nein	ja
plattform-übergreifend[f]	Unix	MS Windows	plattform-übergreifend[f]
Websites	Software (SUIT-Format)	Software (Format n. v.[a])	allg. einsetzbar (interakt. Systeme)
Syntax der UI-Beschreibung[g]	UI	UI Interaktion	UI Interaktion
beliebig[e] (n. v.[d])	5+x[h]	n. v.[d]	beliebig[e]
textuell	neues Design	textuell neues Design	textuell
Oberflächendesign	Oberflächendesign (Informationsdesign)	Oberflächendesign (Informationsdesign) (Interaktionsdesign)	Oberflächendesign Informationsdesign Interaktionsdesign

[e] Durch den Einsatz eines Editors sind beliebig viele Guidelines integrierbar.

[f] Durch die Programmierung des Werkzeugs in Java ist es plattformübergeifend einsetzbar.

[g] In HTML werden nicht alle Faktoren festgelegt, die sich auf die endgültige Darstellung im Browser auswirken. Daher kann nur eine Überprüfung der in HTML programmierten Werte erfolgen.

[h] AIDE basiert auf fünf fest integrierten Maßen. Allerdings ist die Angabe von weiteren *Conditions* möglich (SEARS 1993a). Details über die Möglichkeiten und das Format sind nicht veröffentlicht.

Tabelle 2.2: Übersicht der geforderten Eigenschaften eines Bewertungswerkzeugs und der Erfüllung dieser Eigenschaften durch die bestehenden Werkzeuge.

Aspekte der Bewertung	KRI/AG	ErgoVal	Sherlock	Kwaresmi	AIDE	Ergo-Conceptor	REVISER
Bewertung des Oberflächendesigns	●	●	●	●	●	●	●
Lage/Anordnung	●	○	●	∼	●	●	●
Farbgebung	–	●	●	●	–	∼	●
weitere Attribute von GUI-Elementen	●	●	●	●	○	●	●
Oberflächenkonsistenz	–	–	●	–	○	–	●
Bewertung des Informationsdesigns	○	○	○	○	–	–	●
Semantik der Begriffe	–	–	–	–	–	–	○
Synonyme/Homonyme	–	–	–	–	–	–	●
Bewertung des Interaktionsdesigns	–	–	–	–	○	○	●
Angabe über Gedächtnisbelastung	–	–	–	–	–	–	●
Stereotypenbenutzung	–	–	–	–	–	–	●
Bedienungskonsistenz	–	–	–	–	–	–	●
Eigenschaften des Werkzeugs							
Domänenübergreifend	–	–	–	–	–	–	●
Unabhängigkeit von GUI-Format	–	○	–	–	–	–	●
Freie Programmierung von Regeln	○	○	●	○	–	∼	●
Einbindung von proz. Sourcecode	○	–	●	–	–	–	●
Dialogunterstützung	–	–	○	●	○	∼	○
Einbindung in Entwicklungsumgebung	–	–	●	–	–	○	–
Betriebssystemübergreifend	–	–	–	●	–	–	●

Legende: ● = ja ○ = partiell (s. Text) – = nein ∼ = keine Angaben

2.4 Konzept eines Systems zur kriterienorientierten Bewertung interaktiver Systeme

In den vorausgegangenen Kapiteln wurden verschiedene Methoden und Werkzeuge zur kriterienorientierten Evaluierung interaktiver Systeme aufgeführt. Ausgehend von diesen Erkenntnissen wird nachfolgend das Konzept für ein Werkzeug vorgestellt, dass die flexible kriterienorientierte Bewertung des Oberflächen-, Informations- und Interaktionsdesigns interaktiver Systeme unabhängig vom Spezifikationsformat des Systems erlaubt. Dadurch ist eine entwicklungsbegleitende Bewertung in verschiedenen Phasen der Systementwicklung gewährleistet.

Der Name des Werkzeugs ist REVISER, was als Akronym für „Rapid Evaluation of interactive Systems using Expert Knowledge“ steht. Abbildung 2.12 gibt einen Überblick über die Komponenten von REVISER. Das Werkzeug setzt sich aus den fünf wesentlich Komponenten *GUI-Konvertierung* und *Kriterien-Editor* zur Datenverwaltung, des *Bedienagenten* und der *Simulationskontrolle und Ausgabekomponenten* zur Bewertung sowie dem eingesetzten *Expertensystem*[19] zusammen und integriert diese unter einer einheitlichen Benutzungsoberfläche.

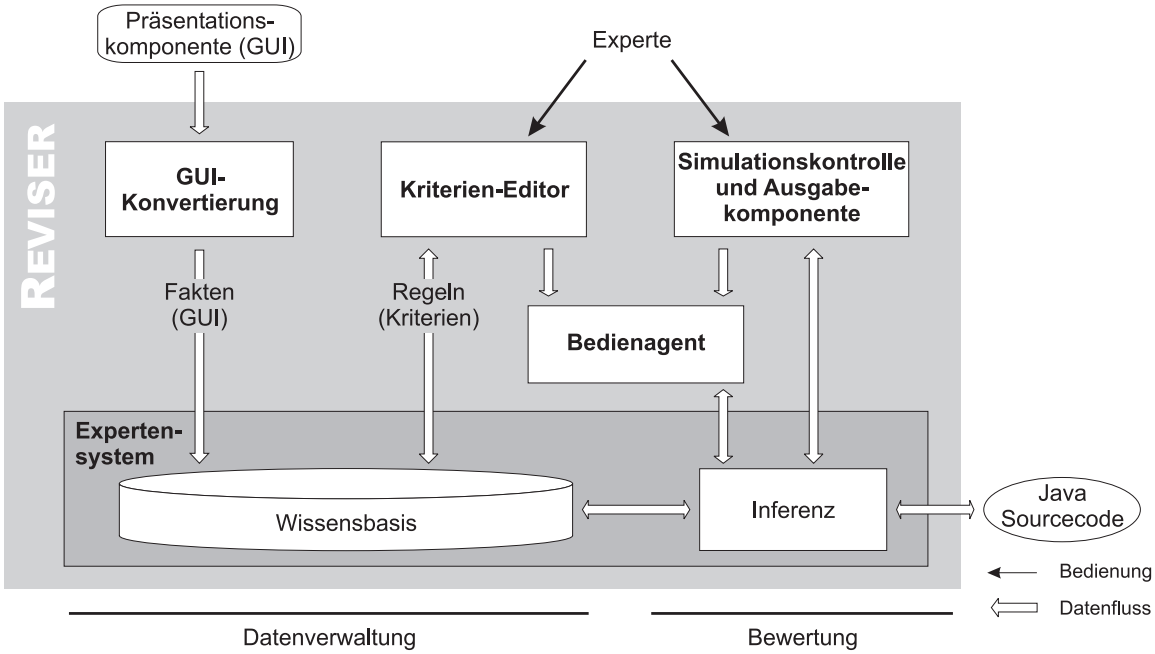

Abbildung 2.12: Komponenten des Werkzeuges REVISERzur kriterienorientierten Bewertung der Benutzungsfreundlichkeit interaktiver Systeme.

REVISER ist als Bestandteil einer werkzeuggestützten Systementwicklung konzipiert. Das zur Systementwicklung eingesetzte Entwicklungswerkzeug (vgl. Abschnitt 1.1.2) stellt die Daten der GUI der Präsentationskomponente bereit, auf denen die Bewertung von REVISER basiert.

GUI-Konvertierung

Die Basis einer Evaluierung der Benutzungsfreundlichkeit eines interaktiven Systems bildet die formale Beschreibung des GUIs (Präsentationskomponente) des Systems. Die Spezifikation der Dialogkontrolle, die üblicherweise als Zustand-Übergangs-Diagramm modelliert ist, lässt sich nahezu ausschließlich auf graphentheoretische Aspekte wie z.B. Erreichbarkeit und Eindeutigkeit der Übergänge hin überprüfen (RUSHBY 2002; NIEDERMAIER 2002; LOER und HARRISON 2004). Darüber hinaus ist ein Import einer vollständigen Spezifikation nur mit so starken Einschränkungen möglich, dass sie als Basis für eine Bewertung der Gebrauchstauglichkeit nicht dienen kann (ZIEREN 2000; PFISTERER 2002; VANDERDONCKT 2005). Aus diesem Grund dient die GUI-Beschreibung des Systems als Basis für die Bewertung in

[19] s. Glossar

REVISER. Je nach Domäne ist diese Beschreibung unterschiedlich. Webseiten beschreiben das GUI allgemein in Form von HTML, während sich für Anwendungssoftware mehrere unterschiedliche Formate durchgesetzt haben. Zur Beschreibung eines GUI eines interaktiven Gerätes hingegen existiert bislang kein einheitlicher Standard. Daher muss REVISER eine größtmögliche Unabhängigkeit vom GUI-Beschreibungsformat besitzen, um möglichst flexibel anwendbar zu sein.

Aus diesem Grund arbeitet REVISER mit einem allgemein gültigen Beschreibungsformat. Ziel ist es, das jeweils benutzte Format durch einen Importfilter in das REVISER-Format zu übersetzen. Dadurch ist die Regelbasis unabhängig vom jeweiligen GUI-Format, wodurch eine Formatsicherheit bei der Implementierung der Guidelines gewährleistet ist. Der Benutzer muss bei einer Änderung des Beschreibungsformates nicht die gesamte Regelbasis ändern, sondern kann die Regeln wiederverwenden. Eine Überprüfung der Syntax des importierten GUI-Formats ist dadurch nicht mehr möglich, bei der Anwendung der Bewertung interaktiver Geräte jedoch auch nicht nötig.

Durch diese Abstraktion bestehender Formatdefinitionen ist es darüber hinaus möglich, Regelwissen, das für eine bestimmte Anwendung implementiert wurde, für eine andere Anwendung wieder verwenden zu können, was den Entwicklungsaufwand deutlich verringern kann.

Kriterien-Editor

Die Implementierung von Kriterien einer Guideline als Regeln erfolgt mit dem dafür in REVISER integrierten Editor. Der Editor muss den Experten beim Erstellen neuer und Bearbeiten bestehender Regeln bestmöglich unterstützen. Dabei soll die Syntax der Regeln in Form von Wenn-Dann-Konstrukten der logischen Formulierung von Kriterien folgen. Darüber hinaus muss es möglich sein, die Attribute konkreter GUI-Elementes zu überprüfen. Darüber hinaus muss REVISER die Abfrage von Attributen über unterschiedliche GUI-Elemente hinweg erlauben, z.B. um die Farbgestaltung unterschiedlicher Elemente miteinander vergleichen zu können.

Der Editor muss den Entwickler durch Eingabedialoge sowie Überprüfung der Syntax und Korrektheit der implementierten Regeln unterstützen. Somit sollen die in den vorangegangenen Kapiteln vorgestellten Möglichkeiten der kriterienorientierten Bewertung gewährleistet werden.

Zur Bewertung des Systemdesigns werden häufig mathematische Berechnungen durchgeführt, z.B. für statistische Kennzahlen oder zur Transformation von Farbwerten in unterschiedliche Farbräume. Dazu muss der Editor eine Schnittstelle zu einer gängigen prozeduralen Programmiersprache bieten, in der diese Berechnungen implementiert und in den Regeln aufgerufen werden können. Die Programmiersprache Java eignet sich dazu besonders, da sie zusätzlich plattformübergreifend einsetzbar ist. Durch diese Kombination von logischen Wenn-Dann-Regeln und Berechnungen in prozeduraler Form erlaubt REVISER dem Experten eine größere Flexibilität der Implementierung als jedes der in Kapitel 2.3 vorgestellten Werkzeuge.

Weiterhin ermöglicht der Kriterien-Editor die Verwaltung einer Begriffsontologie sowie den Zugriff auf die Ontologie zur Bewertung des Informationsdesigns. Eine Möglichkeit der Erstellung ist nicht vorgesehen, da bereits zahlreiche komfortablen Ontologieeditoren[20] existieren und für den Einsatz in einer Domäne eine solche Ontologie nur einmal erstellt werden muss.

Expertensystem zur Bewertung

Zur Überprüfung der GUI-Daten (repräsentiert als Fakten) auf Konformität mit Kriterien (formuliert in Regeln) ist ein Expertensystem[21] in REVISER integriert. Dieses System übernimmt die konsistente Datenhaltung der Fakten und Regeln in einer zentralen Wissensbasis.

Die Bewertung wird in Form einer Inferenz[21] durchgeführt, dazu soll ein gängiger Inferenzalgorithmus zum Einsatz kommen. Dieser muss dem Benutzer jederzeit die Möglichkeit bieten, die Daten, die zur Inferenz benutzt werden, anzeigen zu lassen, um die Ausführung flexibel und nachvollziehbar zu machen.

Simulationskontrolle und Ausgabekomponente

Die Ausführung der Bewertung (Inferenz) soll in einem eigenen Modul erfolgen. Der Benutzer von REVISER muss dabei zu jeder Zeit in der Lage sein, die Inferenz zu steuern, anzuhalten und wieder zu starten. Dabei muss es möglich sein, die Details der Ausführung sowie die benutzten Fakten und Regeln für jeden Schritt anzeigen zu lassen. Dadurch können bei der Entwicklung der Regeln zusätzliche Informationen zum Debuggen angezeigt und die korrekte Ausführung der Regeln sichergestellt werden.

Die Simulationskontrolle soll weiterhin die Auswahl einzelner Regeln zur Bewertung erlauben. Dadurch hat der Experte die Wahl, lediglich einen Teil einer Regelbasis zur Bewertung zu verwenden, um so den Bewertungsfokus zu verändern. Darüber hinaus dient das Modul als Ausgabekomponente für Bewertungsergebnisse, Ausgaben der Simulationskontrolle sowie weiterer Meldungen, die in den Regeln implementiert sind.

Autonomer Bedienagent

Die Bewertung des Interaktionsdesigns basiert auf einer Dialogfolge als aufgabenabhängige Kombination der System- und GUI-Zustände. Die Erstellung einer solchen Dialogfolge auf Basis der formalen Beschreibung des Systems ist aus in Abschnitt 1.1.2 genannten Gründen nicht möglich. Eine alternative Möglichkeit zur Generierung einer Dialogfolge stellt eine simulierte Bedienung des interaktiven Systems dar. Dazu ist in REVISER ein autonomer Bedienagent integriert. Dem Bedienagenten steht für die Bedienung das Vorwissen realer Bediener

[20] Zur Erstellung einer Ontologie hat sich das freie verfügbare Werkzeug *Protégé* der Universität Stanford als De-Facto-Standard durchgesetzt: http://protege.stanford.edu

[21] s. Glossar

zur Verfügung. Dieses Vorwissen beinhaltet z.B. Bedienstereotypen für die erwartungskonforme Bedienung des Systems sowie Begriffsontologien zur Navigation und Bedienung von Menüs.

Während der Bedienung sammelt der Agent die Beschreibung des GUI sowie das Wissen, das er zur Durchführung eines Bedienschritts benötigt. Nach erfolgreicher Bearbeitung einer Aufgabe liegt dadurch eine vollständige Dialogfolge vor, die als Basis der Bewertung des Interaktionsdesigns dient.

Eine Änderung der Struktur und des Verhaltens des Agenten lässt sich durch den Kriterien-Editor vornehmen. Den Zugriff und die Steuerung während der simulierten Bedienung gewährleistet die Simulationskontrolle, die ebenfalls die Ausgaben und Bewertungsergebnisse des Agenten darstellt.

Kapitel 3

Formalisierung ergonomischer Erkenntnisse

Um eine automatische Bewertung interaktiver Systeme auf Konformität mit Guidelines realisieren zu können, muss eine maschinenlesbare Beschreibung sowohl des Systemdesigns als auch des zu überprüfenden Regelwissens vorliegen. Abbildung 3.1 verdeutlicht die Zusammenhänge der Designbewertungen mit den jeweils benötigten Daten zur besseren Übersicht der Inhalte und Aufteilung dieses Kapitels. Die formale Beschreibung des System-GUI wird im folgenden Abschnitt vorgestellt, wobei eine Abgrenzung zu bestehenden Formaten gegeben ist. Eine Möglichkeit, ergonomische Guidelines maschinenlesbar zu formalisieren, ist im darauf folgenden Abschnitt erläutert. Abschließend erfolgt die Vorstellung von Ontologien zur Abbildung der Begriffe und Dialoghierarchie interaktiver Systeme als Basis für die Bewertung des Informationsdesigns. Abschließend sind beispielhaft Guidelines und Maße zur Bewertung des Oberflächen-, Informations- und Interaktionsdesigns von Fahrerinformationssystemen vorgestellt.

3.1 Formale Beschreibung der Benutzungsoberfläche

Die Benutzungsoberfläche beinhaltet unterschiedliche Elemente, die in *Hardwareelemente* und *Displayelemente* aufgeteilt werden können. Zu den Hardwareelementen zählen alle Objekte, die später im Endgerät hardwaretechnisch realisiert sind (z.B. Tasten oder Displays). Displayelemente beschreiben alle Informationen, die auf Displays angezeigt werden, dazu gehören z.B. Begriffe oder Bilder. Beispielhaft ist in Abbildung 3.2 das Fahrerinformationssystem des VW Phaeton mit Hardware- und Displayelementen dargestellt.

Es sind bereits zahlreiche Beschreibungsformalismen entwickelt worden. Diese eignen sich allerdings nicht allgemein zur Beschreibung eines GUI, sondern beschäftigen sich überwiegend mit der Beschreibung von Webseiten bzw. Softwaredialogen, vgl. (IBM 1992; LIMBOURG und VANDERDONCKT 2001; PFISTERER 2002; VANDERDONCKT 2005; BOWEN 2005). Zur Dialogbeschreibung hat sich im Lauf der Zeit ein objektorientierter Ansatz

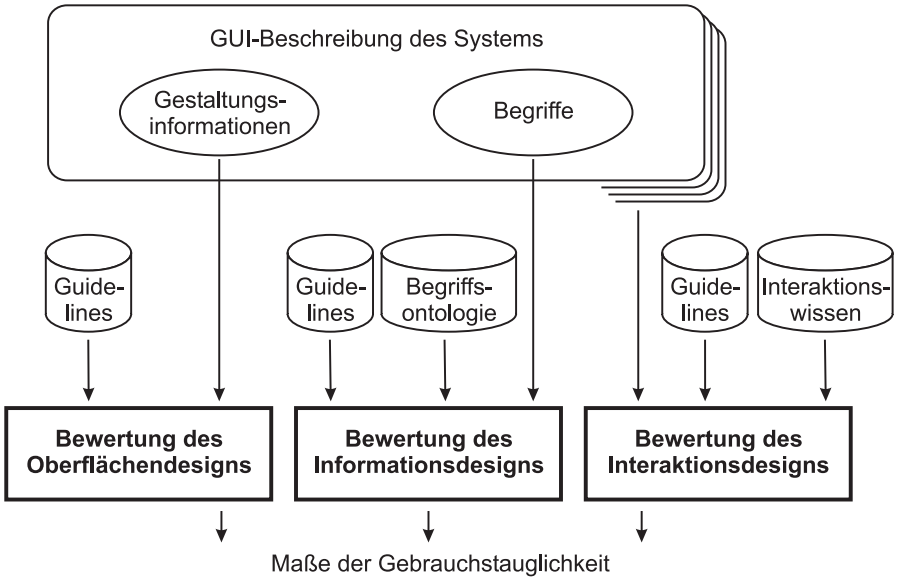

Abbildung 3.1: Datengrundlage der Oberflächen-, Informations- und Interaktionsbewertung. Alle Bewertungen benötigen neben formalisierten Guidelines die GUI-Beschreibung des zu bewertenden Systems.

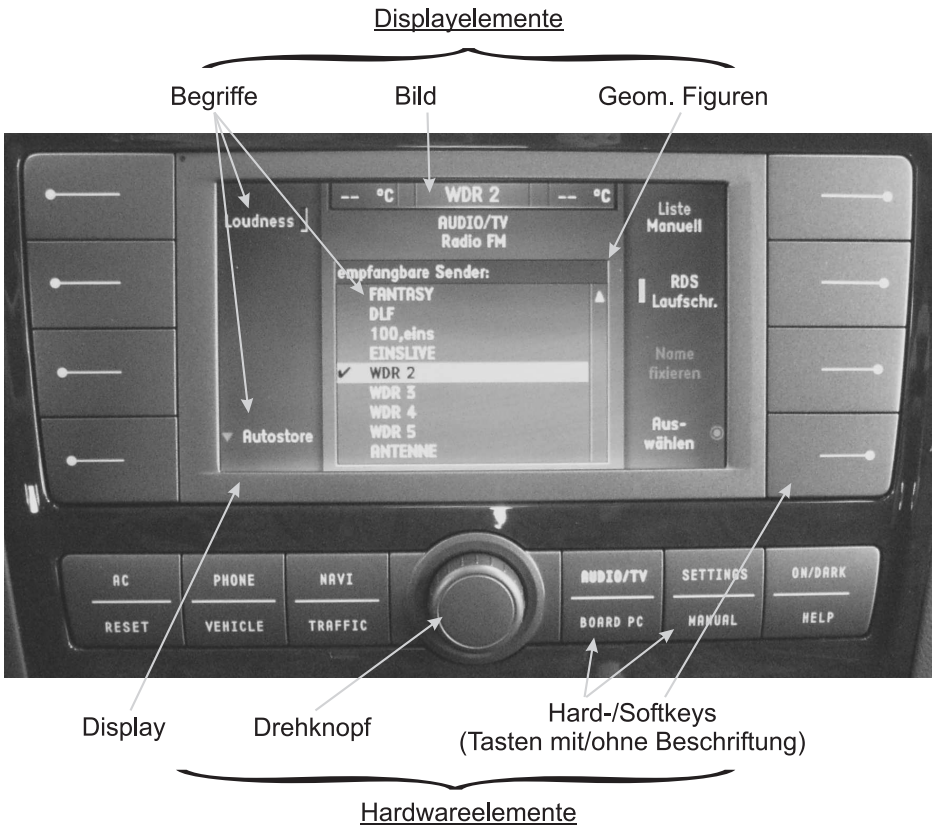

Abbildung 3.2: Fahrerinformationssystem des VW Phaeton mit *Hardwareelementen* (Bedienelemente und Displays) und *Displayelementen* (GUI-Elemente, die auf Displays dargestellt werden).

durchgesetzt. Ausgehend von Grundelementen mit allgemeinen Eigenschaften wie x- und y-Position sowie Höhe und Breite werden weitere Elemente spezifiziert, die Eigenschaften erben und durch die Angaben zusätzlicher Attribute erweitert werden. Ein Beispiel ist im Format des Bewertungswerkzeugs *ErgoVal* in Abbildung 2.6 in Kapitel 2.3.3 gegeben. Ponsard et al. zeigen, dass sich aufbauend auf einem allgemein gültigen objektorientierten Format jedes spezifische Format, wie z.B. zur Beschreibung von Widget[1]-basierter Software, ableiten lässt (PONSARD et al. 2005).

Um alle GUI-Elemente eines interaktiven Systems beschreiben zu können, ist im Rahmen dieser Arbeit ein Beschreibungsformat definiert worden. Das Format sollte dabei in der Lage sein, unabhängig von der Domäne und von Art und Größe des Systems alle sichtbaren GUI-Elemente mit allen Eigenschaften vollständig und eindeutig beschreiben zu können.

Dabei sind die GUI-Elemente (z.B. Displays und Tasten) als 2D-Elemente beschrieben, die frei in einem 3D-Raum angeordnet werden können. Dieses Vorgehen entspricht der Praxis der gängigen Prototyp-Werkzeuge wie *Statemate Magnum* der Firma ILogix oder *Rapid* der Firma E-SIM.

Das Format besteht aus mehreren Objektklassen, die Attribute voneinander erben. Das GUI-Beschreibungsformat mit allen Vererbungsbeziehungen ist als UML-Klassendiagramm in Abbildung 3.3 dargestellt und im Folgenden beschrieben.

Die Klasse *Environment* dient zur Definition der Basis eines kompletten Mensch-Maschine-Interfaces mit x- und y-Ausdehnung sowie einer Hintergrundfarbe. Alle weiteren Objekte sind im Environment platziert. Grundsätzlich darf nur ein Environment definiert sein.

Die Basisklasse aller im Environment abgebildeten Elemente ist *GuiElement*. Dieses Element besitzt Angaben über Raumkoordinaten (x-, y- und z-Position – in Relation zum Environment), die Ausdehnung in x- und y-Richtung sowie die horizontale und vertikale Neigung des Elements. Die GUI ist aus mehreren grafischen Layern[2] zusammengesetzt (0 entspricht dem untersten Layer). Jedes Element befindet sich auf einem der Layer, der durch das Attribut `level` beschrieben ist. Dadurch kann z.B. angegeben werden, welche Beschriftungen zu welchen Elementen gehören (Überdeckung eines Elementes durch Schrift) bzw. ob ein Element durch ein anderes überlagert und dadurch nicht sichtbar ist. Weiterhin ist die Angabe der Farbe, der Transparenz[3] oder der Strichstärke möglich.

Zur Beschreibung geometrischer Figuren sind die Klassen *Line*, *Circle* und *Box* abgeleitet, die die Basisklasse *GuiElement* um entsprechende Attribute erweitern. Bei Line erfolgt

[1] *Widgets* sind gängige Dialogkomponenten, aus denen Softwaredialoge aufgebaut sind. Dazu zählen z.B. Check- und Radioboxen sowie Pull-Down-Menüs.

[2] Die Angabe der Layer ist unabhängig von der z-Position eines Elements. Mehrere 2D-Elemente können genau auf der gleichen Raumebene liegen. Welches Element als oberstes, und welche Elemente dahinter dargestellt werden, entscheidet sich auf Grund des Layers.

[3] Die Angabe der Transparenz erfolgt in Prozent, wobei 100%ige Transparenz bedeutet, dass das Element vollständig durchsichtig ist. Da die in dieser Arbeit modellierten interaktiven Systeme keine transparenten Elemente beinhalten und aufgrund der aufwändigen Berechnung der Transparenz, vor allem von mehreren hintereinander liegenden transparenten Elementen, wird auf eine Betrachtung der Transparenz in dieser Arbeit verzichtet.

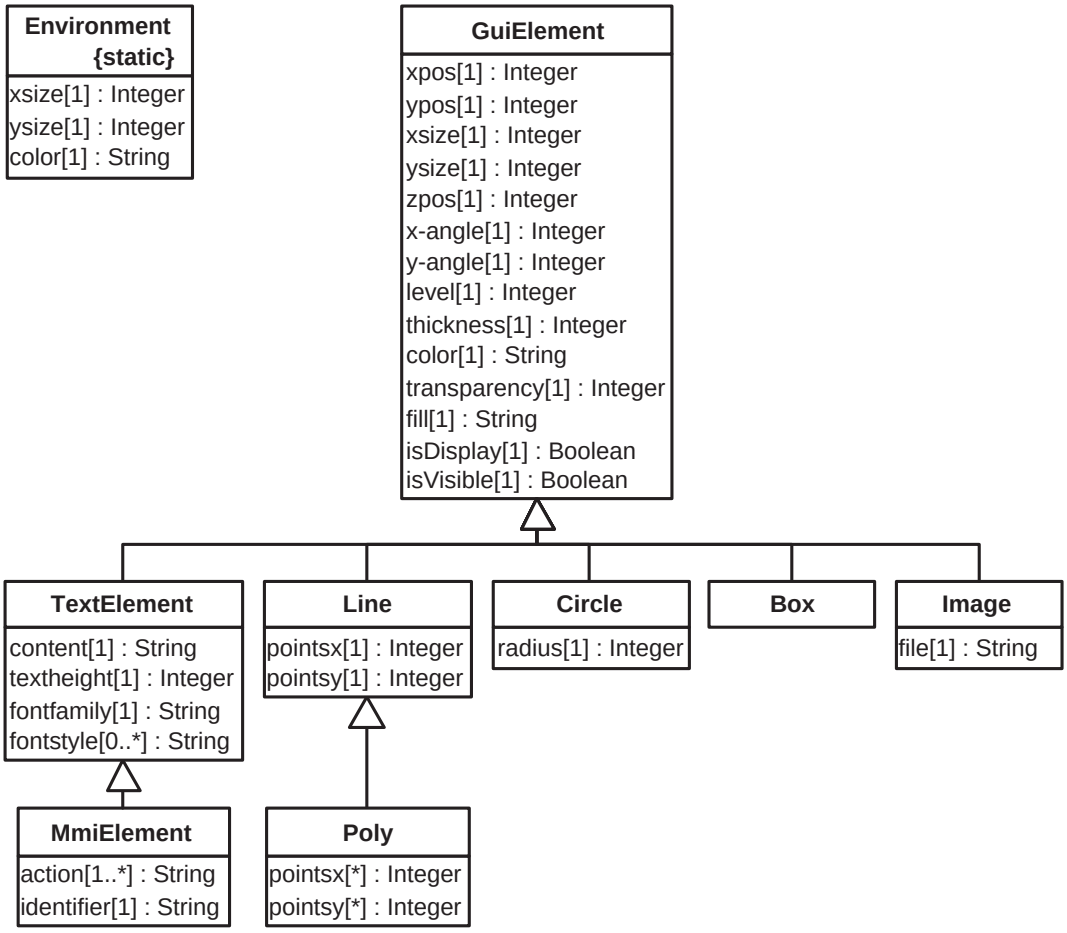

Abbildung 3.3: UML-Diagramm des in dieser Arbeit entwickelten GUI-Beschreibungsformats.

die Angabe des Startpunktes (`xpos`, `ypos` – geerbt von *GuiElement*) sowie des Endpunktes (`pointsx`, `pointsy`) der Linie. Die Unterklasse *Poly* erlaubt die Angabe mehrerer Punktepaare zur Beschreibung von Polygonen durch die Neudefinition der Kardinalität von `pointsx` und `pointsy`. Die Klasse *Circle* dient der Beschreibung von Kreisen mit dem zusätzlichen Attribut `radius`. Rechtecke werden durch die Klasse *Box* beschrieben, die lediglich eine Umbenennung der Basisklasse darstellt. Dadurch können Rechtecke explizit abgefragt werden.

Die Darstellung von Grafiken ist ebenfalls durch die Klasse *Image* mit der Angabe der Grafikdatei möglich. Für eine spätere Bewertung kann jedoch nicht der Inhalt, sondern lediglich die Tatsache der Existenz einer Grafik berücksichtigt werden.

Zur Darstellung von Texten erlaubt die Klasse *TextElement* die Angabe von Schriftart, -typ und -höhe sowie der angezeigten Begriffe.

GUI-Elemente, mit denen der Benutzer das interaktive System bedient (z.B. Tasten oder Drehsteller) werden durch die Klasse *MmiElement* beschrieben. Diese Klasse erfordert die Angabe mindestens einer Aktionsbeschreibung zur Repräsentation von Bedienaktionen. Dazu gehören

z.B. *press* für einen Tastendruck oder *turn_left* und *turn_right* für das Drehen eines Drehknopfes.

Zur Unterscheidung der Hardware- und Displayelemente werden folgende Vorgaben gemacht:

1. Standardmäßig sind alle Elemente Hardwareelemente.
2. Handelt es sich bei einem Hardwareelement um ein Display, muss das Attribut *isDisplay* als „wahr“ (=1)[4] gesetzt sein.
3. Alle GUI-Elemente, die ein Display überdecken (also auf einem grafischen Layer über dem Display liegen), werden automatisch als Displayelement (auf dem Display dargestellte Inhalte) interpretiert, unabhängig davon, ob ihr Attribut *isDisplay* als „wahr“ gesetzt ist oder nicht.

In Abbildung 3.4 ist das GUI eines Mobiltelefons mit der Beschreibung der jeweiligen Elemente beispielhaft gegeben.

Environment, Hintergrundbild und ein Display:

```
• (Environment (xsize 300) (ysize 550) (color #ffffff))
• (Image (xpos 0) (ypos 0) (level 5))
• (Box (xpos 53) (ypos 160) (xsize 175) (ysize 135) (fill
#e2dfdf) (isDisplay 1) (level 7) )
```

Displayelemente (liegen *auf* dem Display – Auszug):

```
• (TextElement (xpos 179) (ypos 271) (xsize 44) (ysize 22)
(textheight 15) (content 'Menü') (color #000000) (isDisplay 0)
(level 9) )
• (TextElement (xpos 108) (ypos 209) (xsize 59) (ysize 30)
(textheight 20) (content 'o2-de') (color #000000) isDisplay 0)
(level 10) )
• (Box (xpos 220) (ypos 246) (xsize 3) (ysize 20) (fill #000000)
(isDisplay 0) (level 11) )
```

Bedienelemente (Auszug):

```
• (MmiElement (xpos 116) (ypos 340) (xsize 50) (ysize 50)
(textheight 10) (content '/\') (action 'press') (isDisplay 0)
(level 3) )
• (MmiElement (xpos 116) (ypos 388) (xsize 50) (ysize 50)
(textheight 10) (content '\/') (action 'press') (isDisplay 0)
(level 1) )
• (MmiElement (xpos 42) (ypos 340) (xsize 50) (ysize 50)
(textheight 10) (content '-') (action 'press') (isDisplay 0)
(level 2) )
```

Abbildung 3.4: Beispiel des Grafikbeschreibungsformats: auf der linken Seite ist das GUI eines Mobiltelefons dargestellt, während rechts auszugsweise die Beschreibungen der jeweiligen Elemente gemäß dem vorgestellten Format abgebildet sind.

Da dieses Format auch zur Simulation eines Prototypen eingesetzt werden soll (s. Abschnitt 3.4.4), ist die Angabe zusätzlicher Informationen z.B. zur Beschreibung von akustischen Signalen nötig. Das Attribut *isVisible* gibt dazu an, ob ein Element auf dem Bildschirm

[4] In Programmiersprachen wird der Ausdruck *falsch* durch die Zahl 0 dargestellt. Jede Zahl ungleich 0 repräsentiert *wahr*, meistens wird dazu jedoch explizit die Zahl 1 benutzt.

angezeigt (also visuell sichtbar) ist. Die Kodierung nicht visueller Elemente ist nicht explizit vorgegeben. So kann beispielsweise die Information „Musik spielt von CD“ durch die Angabe `(TextElement (content 'Musik spielt von CD') ... (isVisible 0) ... )` formuliert werden. Dieses Element wird nicht auf dem Bildschirm angezeigt, ist für eine spätere Bewertung jedoch überprüfbar. Weiterhin sind neben haptischen Eingabemöglichkeiten durch Bedienelemente ebenfalls Spracheingaben modellierbar. Folgendes „Sprach“-Element realisiert die Eingabe des Sprachkommandos „Radio“ („Wechsel zum Radiomenü“): `(MmiElement (content 'sprache') (action 'radio') (isVisible 0))`

Somit ist das Format in der Lage, u.a. haptische und akustische Eingaben sowie visuelle und akustische Ausgaben ohne Einschränkungen zu modellieren. Dadurch lässt sich im Rahmen einer Bewertung des Systemdesigns auf Basis einer Beschreibung der Präsentationskomponente durch dieses Format auch eine Analyse der Ressourcenbelastung von Benutzern durchführen. Bei der Bedienung interaktiver Geräte lassen sich *motorische*, *sensorische*, *akustische* und *kognitive* Ressourcen von Benutzern unterscheiden. Gerade bei der gleichzeitigem Bearbeitung mehrerer oder sicherheitskritischer Aufgaben, z.B. dem Führen eines KFZ und gleichzeitigen Bedienen des Radios, muss auf eine möglichst geringe Belastung der Ressourcen geachtet werden (RASSL 2004; RÖSSGER 2005). Eine Analyse der Ressourcenbelastung kann Hinweise auf die Überlastung eines Ressourcenkanals liefern und somit als Basis für eine Änderung des Systemdesigns dienen (HAMACHER 2003).

3.2 Formalismus zur Beschreibung ergonomischer Erkenntnisse als Regeln

Eine automatische Bewertung erfordert maschinenlesbare Regeln, mit deren Hilfe eine Überprüfung von Systemdaten auf vorgegebene Eigenschaften möglich ist. Dazu müssen die Systemdaten ebenfalls in maschinenlesbarer Form vorliegen (s. vorigen Abschnitt). Die Beschreibung der Systemelemente bildet die Faktenbasis, auf der die Regeln angewendet werden.

Es existieren zwei Formate zur Beschreibung von Guidelines. Beirekdar stellte 2004 die *Guideline Definition Language (GDL)* zur Beschreibung von Guidelines zur Website-Bewertung vor (BEIREKDAR 2004). Shiffman et al. veröffentlichten ebenfalls 2004 das *Guideline Elements Model (GEM)* zur Beschreibung von Guidelines für die Bewertung klinischer Geräte (SHIFFMAN et al. 2004). Beide Formate geben die Art und Weise vor, in welcher Form Attribute von GUI-Elementen abgefragt werden können. So erlaubt z.B. die GDL die Angabe, welches Attribut von welchem GUI-Element (z.B. das Attribut „font color“ des Elementes „td“ (Tabledefinition)) mit welcher Priorität („A“ für niedrige, „AA“ für mittlere und „AAA“ für hohe Priorität) überprüft wird. Allerdings lassen sich dabei keine absoluten Werte abfragen, so ist beispielsweise eine Überprüfung der Farbe nur durch einen direkten Vergleich zweier Farbelemente (z.B. auf Gleichheit) bzw. eines Farbelementes mit einer vorgegeben Farbtabelle, die „gute“ bzw. „websichere“ Farben enthält, möglich. Beide Formate sind sehr stark auf die Anwendungsdomäne der zu bewertenden Systeme ausgerichtet, so dass sie nicht allgemein für

interaktive Systeme einsetzbar sind. Ebenfalls geben sie die Möglichkeiten der Überprüfung von Inhalten der GUI-Elementen restriktiv vor und schränken daher die Möglichkeiten der Bewertung deutlich ein.

Basierend auf diesen Formaten ist im Rahmen dieser Arbeit ein Formalismus zur Beschreibung von Guidelines als Regeln entstanden, der im Folgenden erläutert wird. Die verständlichste Form der Wissensrepräsentation basiert auf Produktionsregeln, da sich herausgestellt hat, dass Experten am ehesten in der Lage sind, ihr Wissen mit Hilfe von Wenn-Dann-Regeln zu formulieren (NEBENDAHL 1987). Die Regeln haben folgende Struktur:

Wenn *P* dann *Q*

P stellt die Vorbedingung (Prämisse[5]) und *Q* die Aktion (Konklusion) dar. Die Prämisse beschreibt eine Situation, in der die Konklusion ausgeführt werden soll (KURBEL 1992).

In der Prämisse werden Aussagen über Eigenschaften von Fakten der Faktenbasis formuliert. Sie basiert auf der Prädikatenlogik[6] erster Ordnung mit der Einschränkung, dass die explizite Benutzung von Quantoren nicht möglich ist. Die Aussagen der Prämisse werden grundsätzlich auf *Wahrheit* bzw. *Erfüllbarkeit* überprüft. Aussagen sind *wahr*, wenn ein Fakt in der Faktenbasis existiert, dessen Attribute die geforderten Eigenschaften *erfüllen*. Sind alle Aussagen erfüllbar, dann trifft die Prämisse zu, die Regel *feuert* und die Konklusion wird ausgeführt.

Die Konklusion entspricht einer Handlungsanweisung für den Fall, dass die Prämisse zutrifft. Gültige Handlungen sind dabei die Generierung einer Ausgabe, das Ändern von Fakten oder der Aufruf von Funktionen (PUPPE 1991).

Im Folgenden werden die Möglichkeiten der Implementierung von Prämisse und Konklusion konzeptuell vorgestellt. Die detaillierte Erläuterung grundlegender Sprachkonstrukte, auf der die Regelsprache basiert, sowie die vollständige BNF[7] der Syntax findet sich in Anhang A.

3.2.1 Implementierung von Regel-Prämissen

Alle Ausdrücke einer Regelprämisse bilden die Grundlage für einen *Mustervergleich (Pattern-Matching)*. Dabei werden alle Fakten der Faktenbasis daraufhin geprüft, ob ihre Attribute die Bedingungen der Prämisse erfüllen. Dieses Vorgehen ist Standard beim Einsatz von Fakten

[5] Für den Wenn-Teil einer Regel existieren zahlreiche synonyme Begriffe. Neben *Vorbedingung (Conditions)* und *Left-Hand-Side (LHS)* wird häufig im Deutschen der Begriff *Prämisse* benutzt, welcher in dieser Arbeit Verwendung findet.

[6] Die Prädikatenlogik erster Ordnung ist zwar in ihren Ausdrucksmittel sehr mächtig, eignet sich jedoch nicht zur Implementation der Prämisse. Durch den Einsatz der Quantoren sind die Prädikate nicht immer entscheidbar, was für eine Anwendung in einem Expertensystem jedoch gefordert ist (EBBINHAUS et al. 1992; STEINMÜLLER 2005). Aus diesem Grund können Quantoren in der Prämisse nicht explizit angegeben werden.

In der Prämisse von Regeln werden diejenigen Eigenschaften von Elementen angegeben, die nötig sind, um die Konklusion ausführen zu können. Die Eigenschaften aller in der Datenbasis vorhandenen Elemente werden durch ein *Pattern-Matching* der Prämisse überprüft. Dazu kommt implizit ein Allquantor $\forall$ zum Einsatz, der für die gesamte Prämisse gilt.

[7] s. Glossar

und Regeln im Rahmen von Expertensystemen (SAVORY 1990). Dementsprechend darf die Prämisse lediglich aus der Angabe von Fakten bzw. deren zur überprüfenden Attributwerte bestehen. Folgender Ausdruck stellt eine korrekte Prämisse dar:

```
(TextElement (xpos 100))
```

Die Regel feuert für jedes `TextElement`, dessen X-Koordinate bei 100 liegt.

Ausdrücke können mit den logischen Verknüpfungen `and`, `or` und `not` verknüpft werden, wobei die Präfix-Notation gilt (vgl. Anhang A). Fehlt eine logische Verknüpfung und sind mehrere Ausdrücke vorhanden, dann werden diese implizit `and`-verknüpft.

Variablen[8] in der Prämisse werden bei ihrem ersten Auftreten mit den in der Faktenbasis vorkommenden Werten belegt. Beim nächsten Vorkommen der Variable wird der entsprechende Wert gefordert. Folgendes Beispiel verdeutlicht diesen Umstand:

```
(TextElement (content Word) (ypos ?y))
(Box (ypos ?y))
```

Die Regel feuert, wenn mindestens ein `TextElement` mit dem Begriff „Word“ und mindestens eine `Box` mit exakt gleicher Y-Koordinate existiert. Beim ersten Auftreten der Variable `?y` wird diese mit der y-Position des Fakts `TextElement` belegt und für das Fakt `Box` gefordert.

Grundsätzlich dürfen Ausdrücke in der Prämisse nur aus der Angabe von Fakten und deren Attributen bestehen. Einzige Ausnahme ist der Aufruf von Funktionen, die einen booleschen Rückgabewert liefern. Im Rahmen des Pattern-Matching schlägt ein Funktionsaufruf fehl, wenn er den Rückgabewert `false` liefert.

3.2.2 Implementierung von Regel-Konklusionen

Die Konklusion einer Regel enthält die Aktionen, die ausgeführt werden sollen, wenn die Prämisse *gefeuert* hat, also alle Ausdrücke der Prämisse erfolgreich überprüft werden konnten.

Die Ausdrücke der Konklusion bestehen ausschließlich aus Anweisungen für Aktionen (im Gegensatz zur Prämisse) in Form von Funktionsaufrufen. Dabei dienen die Funktionen überwiegend zur Veränderung der Faktenbasis sowie zur Ausgabe von Daten. Dabei können Fakten neu angelegt, gelöscht oder verändert werden. Eine Veränderung der Faktenbasis ist sofort wirksam und bedingt eine erneute Abarbeitung bestehender Regeln.

Informationen können – wie in Programmiersprachen üblich – in *Streams* ausgegeben werden. Ein gängiger Stream stellt dabei die Ausgabe auf dem Bildschirm dar. Die Definition eigener Funktionen, die in einer Konklusion aufgerufen werden können, ist ebenfalls möglich.

[8] Variablen werden durch ein führendes „?“ gekennzeichnet. Zu Details der Kodierung s. Anhang A.

3.3 Beschreibung der Begriffe und Dialoghierarchie interaktiver Systeme durch Ontologien

Da in interaktiven Systemen die Displayfläche zur Darstellung von Informationen beschränkt ist, erfolgt der Zugriff auf die Funktionalität des Systems üblicherweise in Form von *Menüs*. Ein Menü stellt eine Anzahl von *Menüoptionen* zur Verfügung, die jeweils eine Systemfunktion oder ein weiteres Menü textuell beschreiben.

Eine Bewertung der benutzten Begriffe (im Rahmen des Informationsdesigns) sowie der Anordnung der Begriffe in der Dialoghierarchie bzw. in einem Menübaum (im Interaktionsdesign) ist durch einen Vergleich mit einem empirisch ermittelten, für eine Domäne allgemein gültigen Begriffsgraphen möglich. Dieser Graph muss alle Begriffe enthalten, die in der Domäne vorkommen, wobei die Verbindungen einer „Verfeinerung“ im Sinne eines Menübaums entsprechen. Ein Unterbegriff verfeinert einen Oberbegriff, wenn er eine Systemfunktion exakter beschreibt. In der FIS-Domäne beschreibt der Oberbegriff *Entertainment* u.a. die Themen *Audio* und *Video*. *Audio* verfeinert damit *Entertainment*.

Zur Beschreibung von Begriffen und ihrer Beziehungen als Graphen in maschinenlesbarer Form haben sich Ontologien durchgesetzt (GRUBER 1993). Zur Abbildung der Begriffe von der obersten bis zur untersten Menüebene genügt eine Darstellung der Ontologie als Begriffsbaum. Beispielhaft ist in Abbildung 3.5 eine Ontologie angegeben, der Begriffe zur Beschreibung der Funktionalität zur Bedienung eines CD-Spielers beinhaltet. Jeder Knoten des Baums stellt dabei potenziell ein Menü dar, dessen Menüoptionen durch die Kinder des Knotens gegeben sind. *idle* bezeichnet den Ausgangszustand (bzw. die Begriffe „Menü“ bzw. „Hauptmenü“) des Systems. Jeder Knoten des Baums enthält synonyme Begriffe, die in Abbildung 3.5 untereinander stehen.

Ontologien zur Beschreibung der Begriffe einer Domäne müssen empirisch erhoben werden und sind allgemein gültig, da sie die Begriffe einer gesamten Domäne (z.B. „Fahrerinformationssysteme“) und nicht nur die Begriffe eines speziellen Systems (z.B. „BMW iDrive“) beinhalten. Zusätzlich zu den hierarchischen Beziehungen in einer Ontologie lassen sich die Wahrscheinlichkeiten angeben, mit der ein Begriff unter einem Oberbegriff erwartet wird. Abbildung 3.6 zeigt beispielhaft einen Teilbaum der Ontologie aus Abbildung 3.5 mit der zusätzlichen Angabe von Wahrscheinlichkeiten zwischen jeweils zwei Begriffen. Sind die Wahrscheinlichkeiten aller Begriffskombinationen bekannt, lässt sich die Erwartungskonformität der tatsächlich in einem System benutzten Begriffen anhand einer Ontologie mit den angegebenen Wahrscheinlichkeiten noch genauer bewerten. Allerdings ist der Aufwand zur Erhebung dieser Wahrscheinlichkeiten aus empirischen Daten sehr hoch. Aus diesem Grund wird auf die Angabe der Wahrscheinlichkeiten in dieser Arbeit verzichtet. Das Vorgehen bei der Erstellung einer Ontologie ist in Anhang C detailliert erläutert.

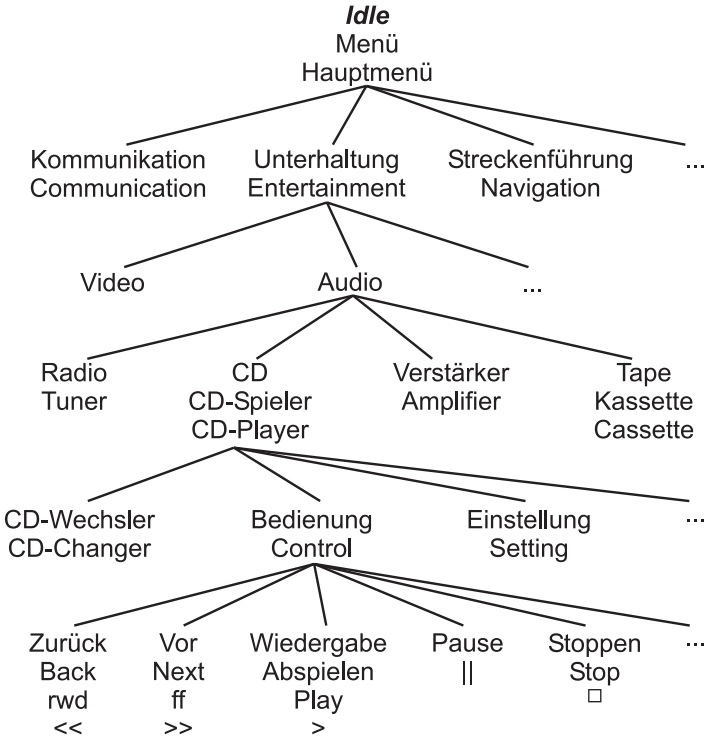

Abbildung 3.5: Beispiel einer Ontologie in Form eines Begriffsbaums mit Begriffen eines Fahrerinformationssystems. Es sind die Begriffe zur Beschreibung der Funktionalität zur Bedienung eines CD-Spielers dargestellt.

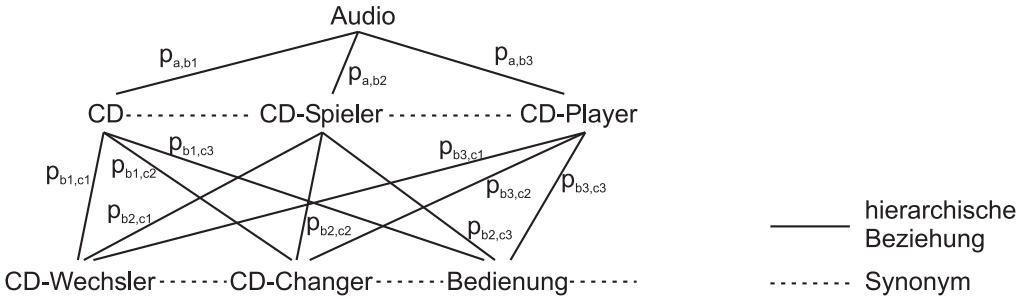

Abbildung 3.6: Beispiel einer Ontologie mit Angabe von Erwartungswahrscheinlichkeit von Begriffen. $p_{i,j}$ bezeichnet die Wahrscheinlichkeit, dass ein Benutzer Begriff j unter Begriff i vermutet.

3.4 Guideline-Bewertung des Designs interaktiver Systeme

Die zur Bewertung des Systemdesigns relevanten Guidelines und Normen sind im Folgenden vorgestellt. Beispielhaft werden konkrete Maße ausgewählter Guidelines mit entsprechenden Beispielen aufgeführt und erklärt. Der Fokus liegt dabei auf der Bewertung von Fahrerinformationssystemen (FIS), deren Eigenschaften in Abschnitt 1.1.1 vorgestellt wurden.

3.4.1 Übersicht über allgemeine ergonomische Guidelines und Normen

Zur Festlegung von Normen existieren zahlreiche Verbände und Organisationen. Zu den wichtigsten gehören die International Organization for Standardization (ISO)[9], das Deutsche Institut für Normung e.V. (DIN)[10] sowie das American National Standards Institute (ANSI)[11].

Designregeln und ergonomische Algorithmen werden i.d.R. exklusiv für eine spezielle Anwendung bzw. ein spezielles Produkt mit meist großem Aufwand entwickelt und aus diesem Grund auch nur selten veröffentlicht (SMITH 1988). Für Fahrerinformationssysteme sind neben zahlreichen Guidelines, die sich mit allgemeinen Darstellungs- und Bedienkonzepten beschäftigen, folgende offiziellen Richtlinien veröffentlicht (eine umfangreiche Auflistung findet sich in (SCHINDHELM et al. 2004)):

- Die Guidelines des US-Department of Defence und des US-Department of Transportation (DEPARTMENT OF TRANSPORTATION 1998; DEPARTMENT OF DEFENSE 1999; GREEN et al. 1995).
- Die Empfehlungen J2364 und J2365 der Society of Automotive Engineers (SAE 2004; SAE 2002).
- Die ISO-Normen 15005 und 15008 (ISO 15005 2003; ISO 15008 2003).
- „Guideline for In-Vehicle Display Systems“ der Japan Automobile Manufactures Association (JAMA 2004).
- Die Richtlinie „Safe and Efficient In-Vehicle Information and Communication Systems“ der Europäische Kommission (EUROPÄISCHE KOMMISSION 2000).
- Die Empfehlungen der Alliance of Automobile Manufacturers (AAM 2003).

3.4.2 Bewertung des Oberflächendesigns

Die Bewertung des Oberflächendesigns interaktiver Systeme betrachtet die Anordnung und Gestaltung der Oberflächenelemente. Darüber hinaus steht ebenfalls die Farbgebung im Fokus einer Überprüfung. Viele Maße basieren auf der Berechnung von Kennzahlen über die Anordnung von Oberflächenelementen, dabei dienen diese Maße vor allem zur Messung der Konsistenz. Die Grundlage der Bewertung bildet die formale Beschreibung des Oberflächendesigns eines Dialogs einer Dialogfolge (vgl. Abb. 3.1). Im Folgenden sind auszugshaft einige Gestaltungsmaße, Kriterien zur Überprüfung der Farbgebung, sowie das Vorgehen zur Berechnung der Konsistenz auf Basis der Gestaltungsmaße erläutert.

[9] www.iso.org
[10] www.din.de
[11] www.ansi.org

3.4.2.1 Maße der Geometrie von Oberflächenelementen

Die Maße zur Bewertung der Lage und Anordnung von Oberflächenelementen stammen überwiegend von Mahajan und Shneiderman, die zahlreiche Maße u.a. aus o.g. Guidelines (s. Abschnitt 3.4.1) operationalisiert haben (MAHAJAN und SHNEIDERMAN 1997).

Display Aspect Ratio (*DAR*)
Erklärung: Gibt das Verhältnis der Breite zur Höhe der benutzten Displays an:

$$DAR = \frac{D^{[h]}}{D^{[w]}} \tag{3.1}$$

Vorgabe: Ein ausgewogenes Verhältnis liegt zwischen 0,5 und 0,8.
Ergebnis: Zahlwert
Alarm (bei Über-/Unterschreitung der Empfehlung)
Quelle: (MAHAJAN und SHNEIDERMAN 1997)

Anzahl der Displayelemente (N_D)
Erklärung: Dieses Maß zählt die Anzahl der im Display dargestellten Displayelemente. Der Wert wird ausschließlich zur Berechnung der Oberflächenkonsistenz benötigt.

$$N_D = |D| \tag{3.2}$$

Ergebnis: Zahlwert
Quelle: (MAHAJAN und SHNEIDERMAN 1997)

linker/rechter/oberer/unterer Rand ($Ma_{l/r/t/b}$)
Erklärung: Diese Maße bestimmen den Abstand zwischen äußerstem Displayelement und Rand des Displays. Der Wert wird ausschließlich zur Berechnung der Oberflächenkonsistenz benötigt. Beispielhaft ist die Berechnung des linken Randes angegeben:

$$Ma_l = d_i^{[x]} - D^{[x]} \quad \text{mit} \quad d_i, d_j \in D, \quad d_i^{[x]} \leq d_j^{[x]} \tag{3.3}$$

Ergebnis: Zahlwert
Quelle: (MAHAJAN und SHNEIDERMAN 1997)

Horizontale/Vertikale Balance (B_h, B_v)
Erklärung: Gibt die horizontale bzw. vertikale Ausgewogenheit der Fläche der auf dem Display befindlichen Displayelement an. Beispielhaft ist die Formel zur Berechnung der horizontalen Balance angegeben:

$$B_h(S) = \frac{\sum_{i \in I} \; d_i^{[w]} \cdot d_i^{[h]}}{\sum_{j \in J} \; d_j^{[w]} \cdot d_j^{[h]}} \tag{3.4}$$

$$\text{mit} \quad I = \{d_i | d_i \in D, d_i^{[x]} \leq D^{[x]} + D^{[w]}/2\}$$
$$J = \{d_j | d_j \in D, d_j^{[x]} > D^{[x]} + D^{[w]}/2\}$$

Vorgabe: Absolute Werte zwischen 0 und 4 entsprechen einem ausgewogenen Verhältnis der Balance. Bei Werten über 4 muss die Balance überprüft werden. Werte bei 10 deuten darauf hin, dass sich die vorhanden Displayelemente nahezu ausschließlich in einer Hälfte des Displays befinden und somit der zur Verfügung stehende Platz nicht gut ausgenutzt ist.
Ergebnis: Zahlwert
Alarm (bei Überschreitung der Empfehlung)
Quelle: (MAHAJAN und SHNEIDERMAN 1997)

Gridedness (G)

Erklärung: Ist ein Maß der *Angeordnetheit* der Displayelemente und wird hauptsächlich zur Konsistenzberechnung benutzt. Es gibt an, in welchem Maße die Displayelemente aneinander ausgerichtet sind. Für alle Displayelemente werden jeweils die linken, rechten, oberen und unteren Kanten aufgenommen; Kanten mit gleichen Werten werden nur einmal gezählt. Die Summe dieser Werte entspricht der Anzahl der vorkommenden Kanten. Die *Gridedness G* des Dialogs *Dlg* einer Dialogfolge berechnet sich anschließend wie folgt:

$$G(S) = \frac{|Dlg|}{|L|+|R|+|T|+|B|} \cdot 100 \tag{3.5}$$

$$\begin{aligned} \text{mit} \quad & L = \{l | l = d^{[x]}, \forall d \in D\} \\ & R = \{r | r = d^{[x]} + d^{[w]}, \forall d \in D\} \\ & T = \{t | t = d^{[y]}, \forall d \in D\} \\ & B = \{b | b = d^{[y]} + d^{[h]}, \forall d \in D\} \end{aligned}$$

Abbildung 3.7 verdeutlicht die Berechnung. Im Dialog D_1 (links) ergeben 4 Displayelemente mit insgesamt 10 Kanten (6 vertikal und 4 horizontal) eine *Gridedness* von $G(D_1) = 40$. Im Dialog D_2 (rechts) sind die Elemente gegeneinander verschoben, was sich in einer deutlich niedrigeren *Gridedness* von $G(D_2) = 28,5$ widerspiegelt.
Elemente, die überdeckt sind, verfälschen die *Gridedness* und sollten daher vor der Berechnung entfernt werden.

Vorgabe: Je größer die Werte im Vergleich, desto mehr sind die Elemente aneinander ausgerichtet. Aufgrund fehlender objektiver Vorgaben eignet sich dieses Maß nur für eine vergleichende Bewertung.
Ergebnis: Zahlwert
Quelle: (MAHAJAN und SHNEIDERMAN 1997)

Widget Density (WD)

Erklärung: Gibt den Grad der Bedeckung eines Displays für einen Dialog einer Dialogfolge *Dlg* an. Die Anzahl aller Displayelemente geht in dieses Maß ein, unabhängig von ihrer tatsächlichen Größe. Der Nenner ist i.d.R. sehr groß, da die Displayfläche üblicherweise in Pixel angegeben ist. Aus diesem Grund ist die

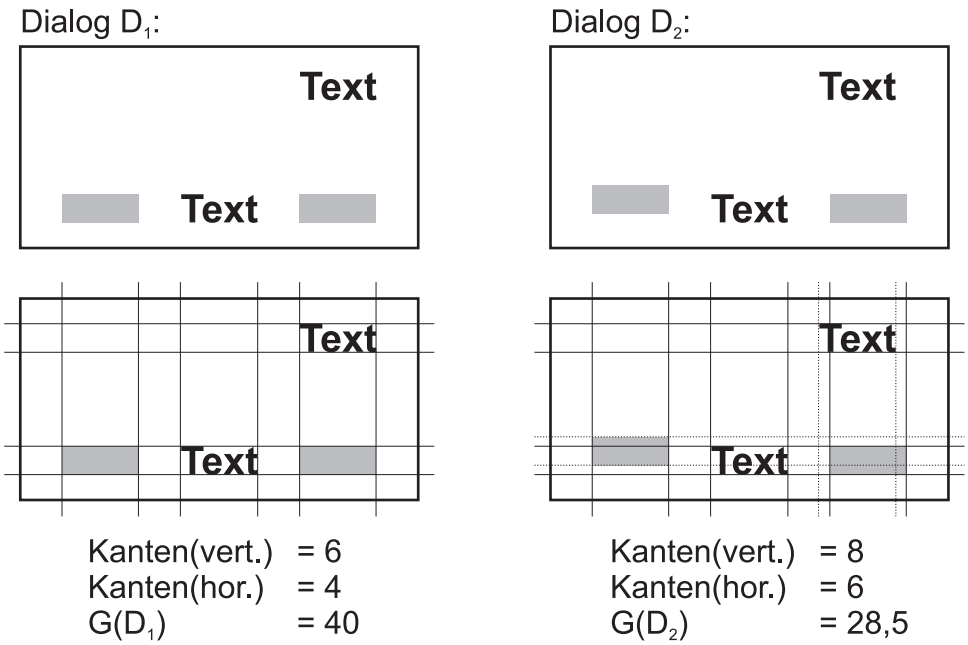

Abbildung 3.7: Beispiel der Berechnung der *Gridedness (Angeordnetheit)* von Displayelementen unter Einbeziehung aller im Bild vorkommenden Kanten. Ein höherer Wert entspricht einem konsistenteren Layout.

Verwendung einer Konstanten k zur Normalisierung hilfreich, für den Mahajan den Wert $k = 100000$ empfiehlt.

$$WD(S) = \frac{|Dlg|}{D^{[w]} \cdot D^{[h]}} \cdot k \tag{3.6}$$

Ergebnis: Zahlwert
Quelle: (MAHAJAN und SHNEIDERMAN 1997)

3.4.2.2 Maße der Farbgebung

Anzahl der Farben (Vordergrund/Hintergrund) ($N_{col_{F/B}}$)

Erklärung: Dieses Maß zählt die Anzahl der benutzten Farben, wobei auch die Hintergrundfarben berücksichtigt werden. Dabei wird bei Displayelementen, die sich überdecken, die Farbe der vorderen Elemente als Vordergrundfarbe und die Farbe des hinteren Elementes als Hintergrundfarbe gezählt. Beispielhaft ist das Vorgehen zum Zählen der Vordergrundfarben dargestellt:

$$N_{col_F} = |D_{col_F}| \tag{3.7}$$
$$\text{mit} \quad D_{col_F} = \{d_i^{[col]} | d_i \in D, \exists d_j \in D : d_i \text{ überdeckt } d_j\}$$

Vorgaben: Die Empfehlungen für die maximale Anzahl benutzter Farben schwankt zwischen 4 und 7. Für gelegentliche Nutzer sind maximal 4, für erfahrene maximal 7 Farben einzusetzen (DEPARTMENT OF TRANSPORTATION 1998).

Ergebnis: Zahlwert
Alarm (bei Über-/Unterschreitung der Empfehlung)
Quelle: (ALBERS 1997; DEPARTMENT OF TRANSPORTATION 1998; IVORY 2001)

ΔE-Farbabstand

Erklärung: Ermittelt den spektralen Abstand jeweils zweier Farben. Die Farben werden beim Design des Systems üblicherweise durch ihre RGB-Werte angegeben. Der RGB-Raum ist jedoch nicht an das menschliche Sehempfinden angepasst. Aus diesem Grund wurde der CIE Yuv-Farbraum entwickelt, in dem sich der mathematische spektrale Abstand am Helligkeitsempfinden des Menschen orientiert (COMMISSION INTERNATIONALE DE L'ECLAIRAGE 2004). Die Formel zur Berechnung des ΔE-Werts zweier Farben i und j ist im Folgenden angegeben:

$$\Delta E(Y_i u_i v_i, Y_j u_j v_j) = \sqrt{\left(155 \cdot \frac{\Delta Y}{Y_{max}}\right)^2 + (367 \cdot \Delta u')^2 + (167 \cdot \Delta v')^2} \tag{3.8}$$

$$\begin{aligned} \text{mit} \quad \Delta Y &= |Y_i - Y_j| \\ Y_{max} &= \max(Y_i, Y_j) \\ \Delta u' &= |u_i - u_j| \\ \Delta v' &= |v_i - v_j| \end{aligned}$$

Abbildung 3.8 stellt beispielhaft Farbkombinationen mit ihren ΔE-Werten dar. Der ΔE-Wert entspricht lediglich dem Abstand der Farben im Yuv-Raum, weitere Aussagen über die Farbgebung, z.B. zur Vermeidung einer Rot-Blau-Kombination, berücksichtigt die Formel nicht. So erzeugt eine Kombination der Farben Rot und Blau durchaus einen guten ΔE-Wert (in Abb. 3.8 links unten).

Vorgabe: ΔE-Werte ≥ 100 weisen auf eine gute Farbkombination hin. Haben zwei Farben einen geringeren Wert, so muss die Farbgebung angepasst werden. Dabei ist zu prüfen, ob die Farben zu Flächen gehören, die aneinander grenzen, oder ob ein derart geringer Farbunterschied aus Designgründen sogar gewollt ist.

Ergebnis: Zahlwert
Alarm (bei Unterschreitung der Empfehlung)
Quelle: (DEPARTMENT OF TRANSPORTATION 1998; COMMISSION INTERNATIONALE DE L'ECLAIRAGE 2004)

Kontrastverhältnis (K)

Erklärung: Gibt den Helligkeitsunterschied zwischen Vordergrundfarbe i und Hintergrundfarbe j an (ebenfalls im CIE Y*u*v*-Farbraum):

$$K(Y_i, Y_j) = \frac{\max(Y_i, Y_j)}{\min(Y_i, Y_j)} \quad \text{für} \quad \min(Y_i, Y_j) \neq 0 \tag{3.9}$$

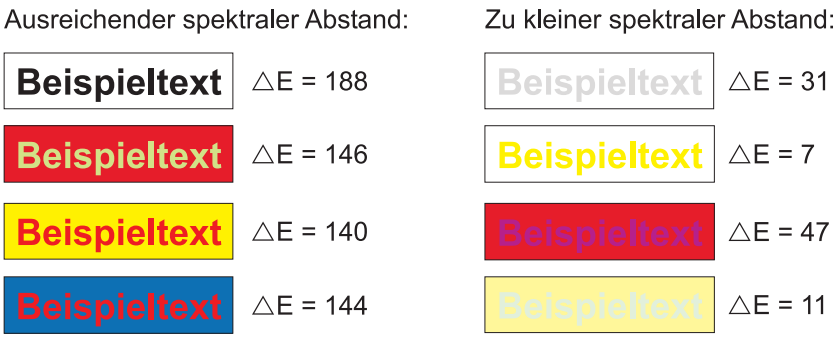

Abbildung 3.8: Farbbeispiele mit zugehörigen ΔE-Werten als Maß der subjektiven Empfindung des Farbkontrastes. Hohe ΔE-Werten (≥ 100) weisen auf einen guten spektralen Abstand hin, niedrige auf einen schlechten.

Vorgabe: Bei Tageslicht ist für junge Menschen mindestens ein Kontrastverhältnis von 1.2, für ältere ein Verhältnis von mindestens 1.4 gefordert. Bei Nacht liegt das geforderte minimale Kontrastverhältnis zwischen 1.6 und 2. Dabei sind die Messwerte grundsätzlich erst ab einer Helligkeit von $Y \geq 50$ aussagekräftig.
Ergebnis: Zahlwert
Alarm (bei Unterschreitung der Empfehlung)
Quelle: (BLACKWELL und BLACKWELL 1971; DEPARTMENT OF DEFENSE 1986; SCHAEFFER 1987; KURSCHEID 2005)

Farb-Gruppen ($N_{col_{text}}$)
Erklärung: Gibt die Anzahl der Textelemente mit gleichen Farben an. Dargestellte Texte sollten grundsätzlich die gleiche Farbe haben. Unterschiedliche Farben sollten nur benutzt werden, um entweder eine Markierung (z.B. ausgewähltes Element einer Liste) oder den *Zustand* einer Information (z.B. eine Funktion ist nicht anwählbar, häufig dargestellt durch eine dunklere Farbe – („ausgegraut“)) anzuzeigen.

$$N_{col_{text}} = |T_{col}| \tag{3.10}$$
$$\text{mit} \quad T_{col} = \{d_i^{[col]} | d_i \in D, d_i \text{ ist Textelement}\}$$

Vorgabe: Eine Farbgruppe mit nur einem Textelement weist ggf. auf einen Fehler in der Farbgebung hin und sollte überprüft werden. Existieren mehrere Farbgruppen mit jeweils mehreren Textelementen kann die Darstellung zu bunt wirken und sollte überprüft werden.
Ergebnis: Zahlwert
Alarm (s. Vorgabe)
Quelle: (DEPARTMENT OF TRANSPORTATION 1998; IVORY 2001)

Spektrale Distanz (Rot-Blau-Kombination)
Erklärung: Beim Farbensehen kommt im Auge der Effekt der chromatischen Aberration zum Tragen. Dabei werden kurze Wellenlängen stärker gebrochen als lange.

Kombinationen mit Farben stark unterschiedlicher Wellenlänge (z.B. rot und blau) sind im Auge nicht gleichzeitig scharf darstellbar, werden daher als unangenehm empfunden, führen schneller zur Ermüdung und sollten vermieden werden.

Vorgabe: Der Einsatz der Farben Rot und Blau bei übereinander liegenden oder benachbarten Displayelemente muss vermieden werden.

Ergebnis: Alarm

Quelle: (OSACA E.V. 1997; GOLDSTEIN 2002)

3.4.2.3 Maße der Beschriftung

Anzahl der Überschriften (N_{header})

Erklärung: Gibt die Anzahl der als Überschrift identifizierten Texte an. Jeder Dialog sollte eine eindeutige Überschrift zur Orientierung bei der Bedienung besitzen. Fehlt eine Überschrift, so muss sich der Benutzer anhand des Dialogkontextes orientieren, was missverständlich sein kann und die Bedienung unnötig erschwert. Mehr als eine Überschrift werden gelegentlich zur Beschriftung von Untermenüs- und -dialogen benutzt, diese sollten jedoch dann ebenfalls unterschiedlich dargestellt werden (z.B. Unterschiede in der Schriftgröße).

$$N_{header} = |H| \quad \text{mit} \quad H = \{d_i^{[col]} | d_i \in D, d_i \text{ ist Überschrift}\} \tag{3.11}$$

Vorgabe: Eine Überschrift ist für jeden Dialog gefordert. Fehlt eine Überschrift muss ein Alarm generiert werden. Bei mehr als einer Überschrift erfolgt ebenfalls ein Alarm, um das Design ggf. zu überprüfen.

Ergebnis: Zahlwert
Alarm (bei Über-/Unterschreitung der Empfehlung)

Quelle: (ISO 9241-110 2004; ISO 9241-12 1998)

Schriftgröße (F)

Erklärung: Gibt die Schriftgrößen der vorhanden Textelemente an.

$$F = \{d_i^{[pt]} | d_i \in D, d_i \text{ ist Textelement}\} \tag{3.12}$$

Vorgabe: Die Schriftgröße sollte mindestens 10 pt[12] betragen.

Ergebnis: Zahlwert
Alarm (bei Unterschreitung der Empfehlung)

Quelle: (DEPARTMENT OF TRANSPORTATION 1998; IVORY und HEARST 2001)

3.4.2.4 Maße der Oberflächenkonsistenz

Konsistenz beschreibt die Ähnlichkeit bzw. Gleichförmigkeit der Gestaltung, wobei für nahezu alle Bereiche des Systemdesigns eine konsistente Gestaltung gefordert wird (STEVENS

[12] *pt* entspricht der Maßeinheit *Punkt (point)*, die vorwiegend zur Angabe der Schriftgröße verwendet wird.

et al. 2004; LUCKHARDT 2005; SHNEIDERMAN und PLAISANT 2005). Im Schwerpunkt einer Systemgestaltung muss daher vor allem die Aufgabenkonsistenz (im Interaktionsdesign) zur gleichförmigen Bedienung bei der Bearbeitung unterschiedlicher Aufgaben sowie die Konsistenz einer Dialogfolge[13] (im Oberflächendesign) für eine einheitliche Oberflächengestaltung stehen (GRUDIN 1989; PRIBEANUS 2000). Eine konsistente Gestaltung wirkt sich wesentlich auf Verbesserung der Erlernbarkeit, Vorhersehbarkeit sowie den Erfolg einer Bedienung und somit auf die Zufriedenheit der Benutzer aus (RHYNE et al. 1987; PARUSH 2001). Alleine durch eine konsistente Gestaltung der Bedienoberfläche kann eine bis zu 25% schnellere Ausführung von Aufgaben erreicht werden (PRIBEANUS 2000). Shneiderman erhebt sogar die Konsistenz zum wichtigsten Ziel in der Systemgestaltung (SHNEIDERMAN und PLAISANT 2005).

Konsistenz in der Oberflächengestaltung ist dadurch messbar, dass unterschiedliche Gestaltungsmaße für einzelne GUI-Konstellationen erhoben werden, die die Farbgebung, die Lage und Größe der Displayelemente sowie die Anordnung der Displayelemente zueinander berücksichtigen. Durch den Vergleich der jeweiligen Werte über verschiedene Dialoge einer Dialogfolge hinweg lassen sich somit Veränderungen in der Gestaltung beobachten (BRAJNIK 2000). Ivory definiert zur Berechnung der Konsistenz einzelner Oberflächenmaße dazu zwei Konsistenzmaße, die im Folgenden vorgestellt werden (IVORY 2001). Basis beider Konsistenzmaße sind die Werte eines Oberflächenmaßes M, die für mehrere zusammenhängende Dialoge einer Aufgabenbearbeitung ermittelt wurden. Weiterhin werden der Erwartungswert des Maßes $E(M)$[14] sowie die Standardabweichung σ_M benötigt.

Z-Score

Erklärung: Die Abweichung des Wertes einer GUI-Konstellation (Dialog) von der Gesamtheit der Werte aller Dialoge einer Dialogfolge ist durch die *Z-Score* dieses Wertes angegeben:

$$\text{Z-Score}(m_i) = \frac{m_i - E(M)}{\sigma_M} \quad \text{für} \quad m_i \in M \tag{3.13}$$

Die Normierung der Abweichung des Wertes vom Erwartungswert durch die Standardabweichung σ_M macht die Z-Score verschiedener Maße miteinander vergleichbar. Sind die Z-Scores zweier unterschiedlicher Maße gleich, so weichen sie auch im gleichen Verhältnis von ihrer Gesamtheit ab, unabhängig von Erwartungswert und Streuung des jeweiligen Maßes.

Vorgabe: Liegt die Z-Score innerhalb des Intervalls $[-1;1]$, dann gilt der jeweilige Wert als konsistent, da er weniger als die Standardabweichung vom Erwartungswert abweicht. Für Werte ausserhalb des Intervalls gilt der Wert als inkonsistent.

[13] In dieser Arbeit wird der Begriff *Dialog* für die GUI-Konstellation kombiniert mit dem entsprechenden Zustand eines Systems zu einem bestimmten Zeitpunkt verwendet. Bei der Bedienung eines Systems präsentiert sich dem Benutzer nach jedem Bedienschritt ein neuer Dialog. Die Informationen, die ein System bei der Aufgabenbearbeitung auf Displays darstellt, lassen sich dadurch eindeutig als *Dialogfolge* darstellen – vgl. Abschnitt 1.1.2.

[14] Die Berechnung des Erwartungswerts E und der Standardabweichung σ für eine Zahlenreihe wird als bekannt vorausgesetzt, s. (BRONSTEIN und SEMENDJAJEW 1991).

Ergebnis: Zahlwert
Quelle: (IVORY 2001)

Coefficient of Variation *CoV*

Erklärung: Der *CoV* gibt die Streuung (Variation) aller Werte eines Maßes über eine komplette Dialogfolge, z.B. für eine Aufgabe, an:

$$\mathrm{CoV}(M) = 100 \cdot \frac{\sigma_m}{E(M)} \tag{3.14}$$

Je kleiner die Streuung der Werte um den Erwartungswert ist, desto kleiner ist der *CoV*. Die Normierung durch den Erwartungswert ermöglicht das Vergleichen mehrerer *CoV* eines Maßes für unterschiedliche Dialogfolgen.

Vorgabe: Der *CoV* ist ein reines Vergleichsmaß, daher muss für ein betrachtetes Maß *M* mindestens der *CoV* von zwei unterschiedlichen Dialogfolgen vorliegen. Der im Vergleich kleinere *CoV* gibt dabei die Dialogfolge mit den konsistenteren Werten des betrachteten Maßes an.

Ergebnis: Zahlwert
Quelle: (IVORY 2001)

Da die *Z-Score* und *CoV* auf statistischen Größen (Erwartungswert und Standardabweichung) basieren, ist deren Berechnung nur für Maße sinnvoll, die pro Dialog einer Dialogfolge aus genau einem objektiv ermittelbaren Zahlenwert (im Gegensatz zu Alarmwerten) bestehen. Maße, die Displayelemente paarweise miteinander vergleichen (z.B. der ΔE-Wert) eignen sich ebenfalls nicht für einen Konsistenzvergleich, da mehrere Werte pro Dialog auftreten können und diese dadurch nicht über mehrere Dialog hinweg vergleichbar sind. Aus den oben vorgestellten Maßen eignen sich zur Berechnung der *Z-Score* und *CoV* z.B. die Maße *Display Aspect Ratio, Anzahl der Displayelemente* N_D, *Anzahl der Farben (Vordergrund/Hintergrund)* $N_{col_{F/B}}$, *Gridedness G, Horizontale/Vertikale Balance* $B_{h/v}$, *unterer/oberer/linker/rechter Rand* $Ma_{l/r/t/b}$ und *Widget Density WD*.

3.4.3 Bewertung des *Wording* im Rahmen des Informationsdesigns

Das Ziel des Informationsdesigns ist die Auswahl geeigneter, aussagekräftiger Begriffe, Icons und Bilder für alle textuellen und grafischen Informationen zur Steigerung der Akzeptanz des Systems bei den Benutzern (VERMEEREN 1999). Eine semantische Analyse von Icons ist bislang noch nicht automatisch durchführbar. Daher fokussiert sich die Bewertung des Informationsdesigns auf das *Wording* eines interaktiven Systems, das die Wahl der Begriffe für Überschriften und Menüeinträge in Displays sowie Beschriftung von Tasten und Reglern umfasst (ROSSON und CARROLL 2002).

Als Maß für die Unterscheidbarkeit dient der *semantische Abstand* dieser Begriffe zueinander, der sich durch zahlreiche Methoden bestimmen lässt (BUDANITSKY und HIRST 2001). Diese Methoden ordnen die Begriffe als Knoten in Graphen an, wobei die Gewichtung der

Kanten zwischen den Knoten den jeweiligen semantischen Abstand zwischen zwei Begriffen repräsentiert. Eine gängige Methode zur Erstellung solcher Begriffsgraphen stellt *Pathfinder* dar, s. (SCHVANEVELDT 1990). Diese Methoden müssen jedoch von Hand durchgeführt werden, da sich eine Untersuchung der Semantik von Begriffen nicht automatisieren lässt (MCDONALD et al. 1986; KIERAS 1999). Dadurch scheidet eine direkte Berechnung des semantischen Abstands zweier gegebener Begriffe im Rahmen dieser Arbeit aus.

Eine Bewertung ist dennoch durch einen Vergleich der eingesetzten Begriffe mit empirisch ermittelten und bereits bewerteten Begriffen möglich. Dabei können die semantischen Beziehungen von Begriffen in Menübäumen sowie die Verwendung von Synonymen[15] bzw. Homonymen[15] überprüft werden. Beide Aspekte des *Wordings* sind im Folgenden erklärt.

3.4.3.1 Bewertung der semantischen Beziehungen von Begriffen in Menübäumen

Begriffe und deren Beziehungen in Menübäumen können mit einer empirisch ermittelten, für eine Domäne allgemein gültigen Ontologie verglichen werden (vgl. Abschnitt 3.3). Bei der Bewertung der benutzten Begriffe darf lediglich deren Lage im Baum betrachtet werden. Ein Begriff (z.B. „CD“), der in der Ontologie unterhalb eines anderen Begriffs (z.B. „Audio“) vorkommt, darf in einem Menü ebenfalls nur unter diesem Begriff („Audio“) eingesetzt werden. Ob in der Ontologie zwischen beiden Begriffen auf weiteren Ebenen zusätzliche Begriffe existieren, ist für den Einsatz in Menüs irrelevant. Ein Menübaum darf also lediglich aus einem Teilbaum einer empirisch ermittelten und vorgegebenen Domänenontologie bestehen, wobei eine beliebige Anzahl von Begriffen wegfallen kann, die Hierarchie jedoch erhalten bleiben muss.

Beispielhaft wird in Abbildung 3.9 ein Teil der Ontologie dargestellt. Dabei sind alle Worte zwischen *idle* und den Begriffen zur Beschreibung der Funktionalität „Abspielen“ aufgeführt. Ein Menübaum, der die Funktion zum Abspielen einer CD beinhaltet, sollte eine Teilmenge dieser Begriffe unter Berücksichtigung der Reihenfolge enthalten. Ein Vergleich der Ontologie mit den entsprechenden Menüs des BMW iDrive und des Audi MMI zeigt, dass diese Vorgaben erfüllt sind und die Fahrerinformationssysteme daher eine erwartungskonforme Bedienung ermöglichen.

Beinhaltet ein Menü einen Begriff, der nicht in der Ontologie enthalten ist (oder dessen Erwartungswahrscheinlichkeit klein ist), so ist davon auszugehen, dass ein potentieller Benutzer Schwierigkeiten beim Verständnis und der Zuordnung des Begriffs in den aktuellen Kontext hat (YOM und FEHRLE 2005).

[15] Synonyme bezeichnen Worte mit der selben Bedeutung (z.B. „Orange - Apfelsine“), während Homonyme gleiche Worte mit verschiedenen Bedeutungen sind (z.B. „Bank“ als Geldinstitut oder Sitzgelegenheit). Das Glossar beinhaltet ausführlichere Definitionen.

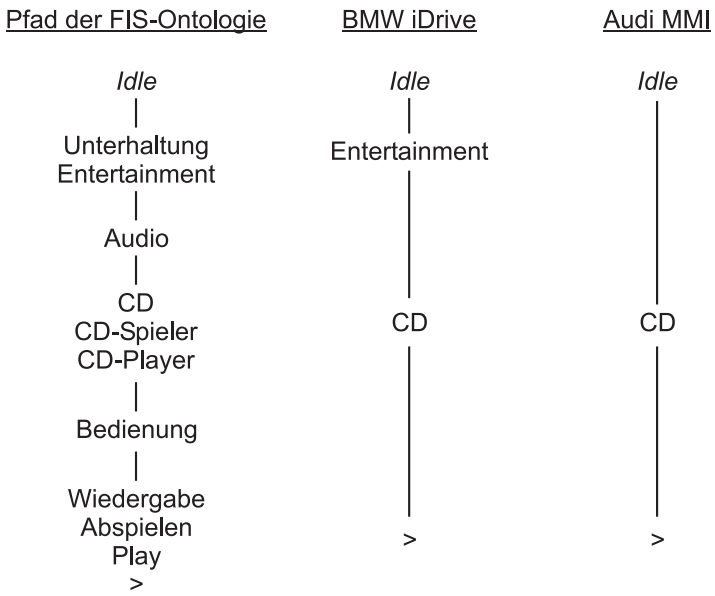

Abbildung 3.9: Worte eines Menüsystems zum Erreichen der Funktion „CD abspielen“. Im Vergleich der Worte der Ontologie aus Abb. 3.5 sind die Begriffe aufgetragen, die im Menü des BMW iDrive und des Audi MMI zum Einsatz kommen.

3.4.3.2 Vermeidung von Synonymen und Homonymen

Durch den Einsatz von Synonymen bei der Darstellung von Begriffen wird die Anzahl der im System eingesetzten Begriffe künstlich erhöht und dadurch die Erlernbarkeit eines Systems erschwert. Aus diesem Grund sollten Synonyme vermieden werden. Homonyme sind nur durch den Kontext eindeutig verständlich. Gerade in Menüsystemen, in denen Begriffe als Schlagworte benutzt werden, können Homonyme zu Missverständnissen führen und sollten daher ebenfalls vermieden werden (ROLLAND und BEN ACHOUR 1998; OMASREITER und METZKER 2004). Diese Vorgaben gelten lediglich für die Darstellung von Begriffen auf dem GUI. Bei der Verarbeitung von Sprachkommandos muss das System die Erkennung von Synonymen dagegen gewährleisten, damit der Benutzer das System durch sein gewohntes Vokabular bedienen kann. Semantische Mehrdeutigkeiten („Abspielen“ einer CD oder einer gespeicherten Staumeldung) lassen sich dabei überwiegend durch die Berücksichtigung des jeweiligen Kontextes auflösen (ANDERSON und LOTTO 2002; AHLERS und OEL 2005)

Die Verwendung von Synonymen ist ebenfalls durch den Einsatz von Ontologien überprüfbar, da eine Begriffsontologie synonyme Begriffe als solche, beispielsweise durch die Angabe mehrerer synonymer Begriffe in einem Knoten der Ontologie, abbilden kann. Beispielhaft ist in Abbildung 3.5 die Beschreibung der Funktionalität „Abspielen“ mit den synonymen Begriffen „Wiedergabe“, „Play“ und dem Zeichen „>“ modelliert. Andere Begriffe sind ebenfalls mit entsprechenden Synonymen aufgeführt, sofern solche existieren bzw. im Kontext eines Menüs in Fahrerinformationssystemen sinnvoll sind.

Eine Überprüfung des Einsatzes synonymer Begriffe ist wie folgt möglich: die verwendeten Begriffe werden gesammelt und daraufhin überprüft, ob ein synonymer Begriff bereits ver-

wendet wurde. In diesem Fall ist das Ergebnis der Bewertung eine Fehlermeldung mit Angabe beider Begriffe bzw. den jeweiligen Menüs, in denen sie vorkommen.

3.4.4 Bewertung des Interaktionsdesigns

Das Interaktionsdesign beinhaltet die Festlegung geeigneter Ein- und Ausgabeelemente (z.B. Tasten, Drehregler, Displays), um eine konsistente, aufgabenangemessene und erwartungskonforme Bedienung gewährleisten zu können (NORMAN 1988).

Bei der Bewertung einer konsistenten Bedienung muss sowohl die Darstellung (Oberflächengestaltung) als auch das Verhalten (Dialogkontrolle) des Systems gleichzeitig betrachtet werden. In modernen Spezifikationswerkzeugen stehen diese Informationen jedoch erst während einer prototypischen Ausführung gleichzeitig zur Verfügung. Die Spezifikation erlaubt lediglich eine statische Betrachtung entweder des GUI oder der Dialogkontrolle, jedoch nicht kombiniert, da das GUI und die Dialogkontrolle in unterschiedlichen Formaten beschrieben und die Benutzungsoberfläche des Systems meist erst zur Laufzeit dynamisch zusammengestellt und vollständig dargestellt wird.

Im Kontext einer Aufgabe lassen sich diese Angaben in Form einer Dialogfolge generieren. Diese Folge enthält alle Zustände des Systems mit den auf dem GUI dargestellten Informationen sowie die Aktionen, die der Benutzer durchführen muss, um eine Aufgabe erfolgreich bearbeiten zu können (vgl. Abschnitt 1.1.2).

Die Dialogkontrolle ist üblicherweise mit Zustands-Übergangs-Diagrammen spezifiziert, die auf endlichen Automaten basieren (vgl. Abschnitt 1.1.2). Eine isolierte Bewertung der Dialogkontrolle erlaubt lediglich die Anwendung graphentheoretischer Methoden auf die endlichen Automaten z.B. zur Überprüfung der Erreichbarkeit einzelner Knoten oder der Suche nach längsten bzw. kürzesten Wegen (MCDONALD et al. 1986; LOER und HARRISON 2004; BERSTEL et al. 2005). Allerdings bleibt bei diesen Ansätzen die Oberflächengestaltung des Systems unberücksichtigt.

Eine Möglichkeit der automatischen Erstellung einer Dialogfolge besteht in der Bedienung des Systems durch einen interaktiven Agenten. Der Agent stellt dabei einen simulierten Benutzer dar und bedient das System auf Grundlage von Vorwissen. Dazu gehören z.B. Bedienstereotypen sowie eine Ontologie als Modell der Dialoghierarchie bzw. des Menübaums eines interaktiven Systems. Den Aufbau und die Erstellung eines Bedienagenten ist in Kapitel 4 erläutert.

Die Grundlage einer Bewertung des Interaktionsdesigns bildet eine Dialogfolge. Bei der Bewertung können folgende Aspekte betrachtet werden:

Anzahl der Bedienschritte (N_{AC})

Erklärung: Gibt die Anzahl der durchzuführenden Bedienschritte an. Die Anzahl der Bedienschritte sollte möglichst klein sein.

$$N_{AC} = |AC| \tag{3.15}$$

Zuordnung: pro Aufgabe (Dialogfolge)
Ergebnis: Zahlwert

Anzahl Menüebenen (N_{ME})
Erklärung: Gibt die Anzahl der in der Dialogfolge vorkommenden Menüebenen *Me* an. Die Anzahl der Menüebenen sollte möglichst klein sein.

$$N_{ME} = |ME| \tag{3.16}$$

Zuordnung: pro Aufgabe
Ergebnis: Zahlwert

Anzahl Menüoptionen (N_{me_i})
Erklärung: Gibt die Anzahl der Menüoptionen eines Menüs an. Günstig sind 3 bis 12 Menüoptionen pro Menü (NORMAN 1991). Als optimal gelten 7 ± 2 Menüoptionen (MILLER 1981; JACKO et al. 1995; ZAPHIRIS et al. 2002).

$$N_{me_i} = |me_i| \tag{3.17}$$

Zuordnung: pro Dialog der Dialogfolge
Ergebnis: Zahlwert
Alarm (bei Über-/Unterschreitung der Empfehlung)

Menu Aspect Ratio (*MAR*)
Erklärung: Im Rahmen einer Untersuchung konnten Rauch et al. zeigen, dass in Fahrerinformationssystemen eine *breite* Menüstruktur eine schnellere Bedienung erlaubt und lernförderlicher ist als eine *tiefe* (RAUCH et al. 2004). Breite Menüs ordnen mehr Menüoptionen auf einer Menüebene an als tiefe Menüs und haben dadurch weniger Menüebenen. Der Menübaum breiter Menüs erscheint dadurch breit und flach, der Menübaum tiefer Menüs tief und schmal. Das Verhältnis von Tiefe und Breite eines Menübaums lässt sich durch folgende Formel errechnen:

$$MAR = \frac{\sum_i |me_i|}{N_{ME}} \tag{3.18}$$

Die Formel gibt die durchschnittliche Anzahl der Menüoptionen pro Ebene an. Bei gleich bleibender Anzahl von Menüoptionen erzeugt eine flachere Struktur einen höheren Wert. Bei einer vergleichenden Bewertung ist daher ein höherer Wert zu bevorzugen.
Zuordnung: pro Aufgabe
Ergebnis: Zahlwert

Anzahl und Art der benutzten Bedienaktionen (N_{CE})

Erklärung: Die tatsächlich zu tätigenden Aktionen (z.B. Druck einer bestimmten Taste) werden ausgegeben. Durch einen direkten Vergleich mit einer normativen Aktionssequenz können automatisch fehlerhafte Aktionen gefunden werden (HAMACHER 2001).

$$N_{CE} = |CE| \tag{3.19}$$

Zuordnung: pro Dialog der Dialogfolge
Ergebnis: Zahlwert
Liste (Aktionsbeschreibung)

Anzahl und Art der verwendeten Stereotypen (N_{St})

Erklärung: Die zur Bedienung verwendeten Bedienstereotypen erlauben eine Abschätzung des zur erfolgreichen Aufgabenbearbeitung benötigte Vorwissen späterer Benutzer. Dabei erfolgt die Angabe der Stereotypen für jeden Bedienschritt bzw. jeden Dialog der Dialogfolge einzeln. So können Fehler in der Bedienung vorhergesagt und Fehlerursachen eindeutig identifiziert werden. Weiterhin erlaubt die Angabe der verwendeten Stereotypen eine Abschätzung der Gedächtnisbelastung über eine bzw. mehrere Aufgaben hinweg (KIERAS et al. 1997). Je nach dem Erfahrungsgrad der späteren Benutzer kann dadurch die Bedienung des Systems ggf. angepasst und vereinfacht werden.

$$N_{St} = |St| \tag{3.20}$$

Zuordnung: pro Dialog der Dialogfolge / pro Aufgabe
Ergebnis: Zahlwert
Liste (Stereotypenbeschreibung)

Zur Menüorientierung verwendete Begriffe

Erklärung: Die Begriffe der Menüoptionen, die bei einer Aufgabenbearbeitung angewählt werden müssen, ergeben ebenfalls eine Abschätzung der Gedächtnisbelastung bzw. des benötigten Vorwissens späterer Benutzer (STRAYER und JOHNSTON 2001; JAHN et al. 2004). Die Begriffe werden zusammen mit ihren (in der Ontologie) gespeicherten Synonymen ausgegeben und bieten so eine Grundlage zur manuellen Überprüfung.
Zuordnung: pro Dialog der Dialogfolge / pro Aufgabe
Ergebnis: Liste (Begriffe)

Anzahl der gleichzeitig sichtbaren Menüs (N_{VM})

Erklärung: In interaktiven Systemen werden häufig mehrere Menüs (Menüebenen) gleichzeitig dargestellt. Die Schwierigkeit bei der Bedienung liegt dabei im Erkennen, mit welchen Aktionen ein Wechsel von einem Menü in ein anderes erreicht

werden kann. Zusammen mit der verwendeten Aktion und der benötigten Stereotype lassen sich mit der Anzahl der gleichzeitig sichtbaren Menüs Fehler im Menüdesign identifizieren.

$$N_{VM} = |VM| \tag{3.21}$$

Zuordnung: pro Dialog der Dialogfolge
Ergebnis: Zahlwert

Anzahl und Art der Darstellung einer Menüselektion ($N_{M_{SMO}}$)

Erklärung: Bei der Navigation in Menüs muss das aktuell ausgewählte Menüelement grafisch von den restlichen Menüelementen abgehoben werden. Das kann durch eine Änderung einer Eigenschaft des Elementes selber (Änderung der Farbe oder der Größe) oder die zusätzliche Verwendung eines Grafikelementes (Umrahmung mit Rechteck, Unterstreichung durch Linie oder Voranstellung eines kleinen Kreises) erreicht werden. Grundsätzlich muss die Markierung eines Menüelements gleichförmig sein (ISO 9241-12 1998).
Um die Durchgängigkeit der Darstellung einer Menüselektion überprüfen zu können, werden die Anzahl und Art der Darstellung einer Menüselektion pro Dialog der Dialogfolge ausgegeben. Abweichungen können auf diese Weise schnell ausfindig gemacht und das Systemdesign entsprechend angepasst werden.

$$N_{M_{SMO}} = |MS| \tag{3.22}$$

Zuordnung: pro Dialog der Dialogfolge
Ergebnis: Zahlwert
Beschreibung der Darstellungsart

Kapitel 4

Aufbau und Funktion des Bedienagenten

Bei der Bedienung eines interaktiven Systems werden dem Benutzer die einzelnen Systemzustände durch unterschiedliche Konstellationen der GUI präsentiert. Anhand der dargestellten Informationen muss der Benutzer entscheiden, mit welcher Bedienaktion und welchem Bedienelement er sein Ziel bzw. Zwischenziel erreicht. Um eine eindeutige Bedienung zu gewährleisten, dürfen die dargestellten Informationen keinen Interpretationsspielraum haben. So muss dem Benutzer zu jeder Zeit klar sein, welche Funktion sich mit jedem Bedienelement ausführen lässt, mit welchem Begriff innerhalb eines Menüs das Aufgabenziel beschrieben ist bzw. um welchen Typ es sich bei den dargestellten Informationen handelt (z.B. um eine Überschrift, eine Menüoption, einen Funktionswert oder um eine Liste).

Die Systemzustände mit den jeweiligen GUI-Konstellationen und den zur Bedienung benötigten Aktionen bilden eine Dialogfolge (s. Kapitel 1.1.2), die die Basis der Bewertung des Interaktionsdesigns darstellt. Die statische Generierung einer Dialogfolge aus einer formalen Systemspezifikation ist nicht möglich, da die GUI und die Dialogkontrolle in unterschiedlichen Formaten beschrieben und die Benutzungsoberfläche des Systems erst zur Laufzeit dynamisch aufgebaut und vollständig dargestellt wird. Darüber hinaus kommen in formalen Systemspezifikationen üblicherweise Variablen zum Einsatz, deren Werte ebenfalls erst zur Laufzeit bekannt sind (ZIEREN 2000; BISCHOFF et al. 2002; VANDERDONCKT 2005).

Eine weitere Möglichkeit der Generierung einer Dialogfolge liegt in der Simulation der Interaktion eines Benutzers und dem Aufzeichnen der jeweils durchgeführten Aktion mit der Beschreibung der bei jedem Bedienschritt dargestellten GUI-Konstellation. Eine solche simulierte Interaktion kann ein autonomer Softwareagent leisten, dem zur Bedienung Vorwissen zur Verfügung steht. In den folgenden Abschnitten wird der grundlegende Aufbau eines solchen Bedienagenten sowie das Vorgehen bei der Bedienung detailliert erläutert.

4.1 Aufbau eines interaktiven Bedienagenten

Der prinzipielle Aufbau eines Bedienagenten[1] ist in Abbildung 4.1 dargestellt und im Folgenden beschrieben. Die Datengrundlage der Bedienlogik bildet eine formale Beschreibung des GUI des Systems. Aufgrund dieser GUI-Konstellation muss der Agent den aktuellen Systemzustand erkennen. Um den Folgezustand (zur Erreichung des Aufgabenziels) zu erhalten, muss er die dargestellten Informationen nach Typ klassifizieren und den Begriff des (Zwischen-)Ziels identifizieren. Dabei gilt es, die Darstellung von Listen und Menüs sowie die Funktionsbeschreibungen von Softkeys voneinander zu unterscheiden. Weiterhin ist die Detektion der aktuell selektierten Menüoption erforderlich. Nach der Erkennung von Art und Lage der Informationen erfolgt die Auswahl des Bedienelements mit der entsprechenden Bedienaktion, z.B. ein einfacher Tastendruck zur Funktionsauswahl oder die Menübedienung durch einen Dreh-Drück-Knopf. Das erfordert die Berücksichtigung und Bewertung gängiger Bedienstereotypen.

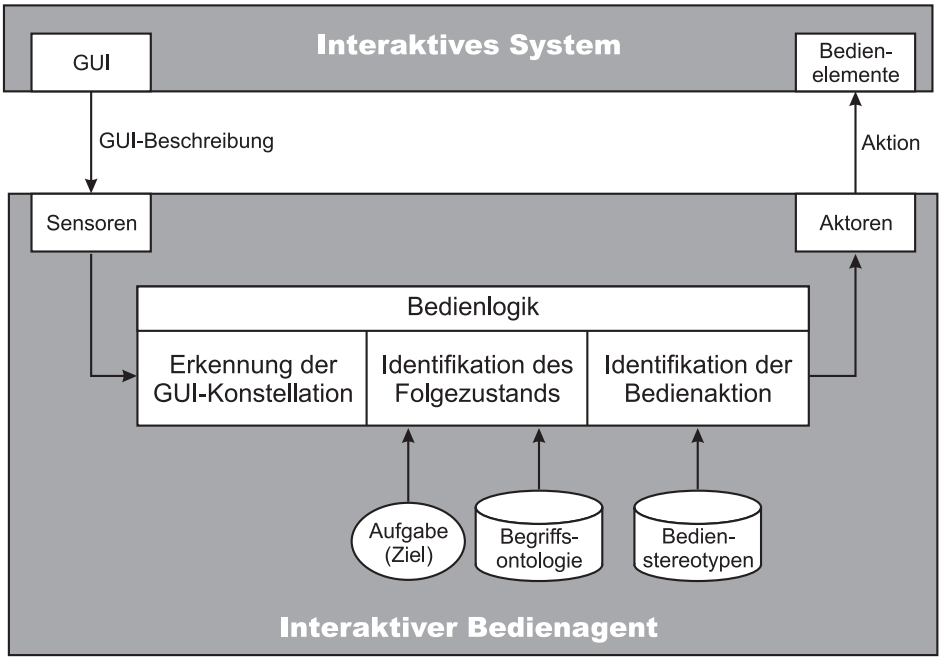

Abbildung 4.1: Prinzipieller Aufbau eines Bedienagenten als simulierter Benutzer zur Bewertung des Interaktionsdesigns interaktiver Systeme.

Zur Klassifizierung der Eigenschaften von Agenten haben Russel und Norvig eine allgemein gültige Taxonomie vorgestellt (RUSSEL und NORVIG 2004). Um erfolgreich mit einem System interagieren zu können, muss ein Agent gemäß dieser Taxonomie *reflexiv*, *modellbasiert* und *zielbasiert* sein sowie die Möglichkeit zum *Lernen*[2] bieten. Diese Eigenschaften sowie

[1] Der allgemeine Aufbau des Agenten orientiert sich an den von Riedl und Amant beschriebenen Schritten zur Menübedienung sowie dem von Eberleh definierten, allgemein gültigen Zyklus der Mensch-Maschine-Interaktion (RIEDL und AMANT 2002; EBERLEH 1989).

[2] *Lernen* wird im Kontext der vorgestellten Agententaxonomie als die Eigenschaft bezeichnet, durch Erkenntnisse im Zeitpunkt t_i auf Erkenntnisse im Zeitpunkt t_{i+1} zu schließen bzw. das Verhalten des Agenten entspre-

ihre Ausprägungen in einem Bedienagenten sind im Folgenden vorgestellt.

- **Reflexivität**
  Der Agent ist in der Lage, Informationen wahrzunehmen, zu verarbeiten und seine Aktionen in Abhängigkeit davon auszuwählen. Solche Aktionen werden auch *bedingte* Aktionen genannt, da sie Bedingungen an die Eingangsinformationen stellen (AMANT und WOOD 2005).

- **Modellbasiertheit**
  Die Aktionen des Agenten basieren nicht nur auf Bedingungen über eingehende Informationen, sondern darüber hinaus auf Vorwissen in Form eines Modells der *Umwelt* (CAWSEY 2003). Der Bedienagent besitzt Vorwissen in der Art der Bedienlogik, in der das Vorgehen zur Bedienung festgelegt ist. Zur Zielfindung dient eine Begriffsontologie, die Begriffe und die Dialoghierarchie interaktiver Systeme abbildet. Zur Auswahl einer Aktion stehen ihm darüber hinaus gängige Bedienstereotypen zur Verfügung.

- **Zielbasiertheit**
  Zur Realisierung eines *agierenden* Agenten – im Gegensatz zu einem *reagierenden*, rein reflexiven oder modellbasierten Agenten – muss ein Ziel zur Bearbeitung einer Aufgabe mit einem interaktiven System vorliegen (GÖRZ et al. 2003). Dieses Ziel besteht bei FIS üblicherweise im Erreichen einer bestimmten Systemfunktionalität. Zur zielgerichteten Bedienung eines Systems steht dem Agenten als Ziel die Beschreibung einer Systemfunktion zur Verfügung.

- **Möglichkeit des Lernens**
  Russel und Norvig bezeichnen mit *Lernen* die Möglichkeit, das Verhalten des Agenten auf Grund von Rückschlüssen aus vorhergehenden Aktionen und der daraus folgenden Reaktion der Umwelt anzupassen (RUSSEL und NORVIG 2004). Häufig wird dies durch eine Bewertung in Form von Gewichten für alternative Aktionsmöglichkeiten erreicht (LUGER 2001).
  Die Darstellung der Zustände interaktiver Systeme auf Displays ist nicht immer eindeutig. So gibt es große Unterschiede und Unbestimmtheiten in Art und Lage der Softkey-Beschriftungen, in Art und Aufbau von Menüs und Listen sowie der Darstellung selektierter Menüoptionen. Um eine größere Sicherheit in der Bedienung zu erreichen, benötigt der Agent eine Bewertung der durchgeführten Bedienschritte. Dieses ist durch eine entsprechende Bewertung von *Bedienhypothesen* möglich: Vor der Durchführung einer Aktion speichert der Agent die Erwartung, wie das System auf die Aktion reagieren wird, in Form einer Hypothese ab. Nach der Durchführung der Aktion wird die Antwort des Systems (formale GUI-Beschreibung) daraufhin überprüft, ob die Erwartungen eingetroffen sind oder nicht. Dadurch kann das Wissen, das zur Auswahl der Aktion zum Einsatz kam (z.B. in Form einer Bedienstereotype), bewertet werden. Dieses Vorgehen ist in Abschnitt 4.4.1 erläutert.

chend anzupassen (RUSSEL und NORVIG 2004).

In den folgenden Abschnitten ist die Realisierung der oben genannten Eigenschaften gemäß der Verarbeitungkette (nach Abbildung 4.1) „*Erkennung der GUI-Konstellation ⇒ Identifikation des Folgezustands ⇒ Identifikation der Bedienaktion*“ erläutert.

4.2 Erkennung des Systemzustands anhand der GUI-Konstellation

Zur Erkennung und Analyse einer GUI-Konstellation auf Basis einer formalen GUI-Beschreibung steht dem Agenten Vorwissen zu Lage, Art und Beschriftung von Bedienelementen[3] zur Verfügung. Auf Basis dieses Vorwissens analysiert der Agent die GUI-Konstellation eines Dialogs, um den aktuellen Zustand des interaktiven Systems zu erkennen. Die Art des Vorwissens sowie das Vorgehen bei der Erkennung der GUI-Konstellation ist in den folgenden Abschnitten erläutert.

4.2.1 Vom Agenten benötigtes Vorwissen zu Art und Lage von Informationen und Bedienelementen

Bedienelemente lassen sich in *Hardkeys, Softkeys* oder *Bedienelemente zur Menübedienung* einteilen. Abbildung 4.2 zeigt beispielhaft alle drei Bedienelementtypen am FIS des VW Phaeton.

Hardkeys sind Bedienelemente mit fester Funktion, deren Funktionsbeschreibung üblicherweise auf oder direkt neben dem Bedienelement angeordnet ist. *Softkeys* sind Tasten mit wechselnder Funktionalität. Beim Einsatz von Softkeys gibt es mehrere Möglichkeiten zur Anzeige der aktuellen Funktionsbeschreibung. Da sich diese während der Systembedienung ändert, wird sie üblicherweise auf Displays dargestellt. Idealerweise ist auf jedem Softkey selber ein Display angebracht. Aus Kostengründen werden Softkeys jedoch meist direkt neben den Displays angeordnet, die die Funktionsbeschreibungen darstellen (vgl. Abb. 4.2). Bei *Bedienelementen zur Menübedienung* bzw. zur Einstellung von Werten handelt es sich um Drehknöpfe, Schieberegler oder Joysticks, die üblicherweise keine Beschriftung haben.

Die Anordnung unterschiedlicher Informationen auf Displays ist ebenfalls als Vorwissen gegeben. Dadurch kann der Agent entscheiden, ob z.B. ein dargestellter Text zu einem Menü gehört oder eine Funktionsbeschriftung eines Softkeys darstellt. Eine Empfehlung zur Aufteilung eines Displays in unterschiedliche Bereiche zur Darstellung von Informationen ist in den Empfehlungen (OSACA E.V. 1997) und (ISO 9241-12 1998) gegeben und in Abbildung 4.3 dargestellt. Funktionsbeschreibungen von Softkeys finden sich demnach üblicherweise im linken, rechten oder unteren Bereich des Displays. Softkeys sollten nicht über einem Display angeordnet sein, da bei der Bedienung das Display (inkl. Funktionsbeschreibung)

[3] Eine Norm, die sich mit der Lage von Bedienteilen und Displays im Fahrzeugcockpit beschäftigt, ist die ISO-Norm 4040 „Personenkraftwagen – Anordnung der Handbedienteile, Anzeige- und Kontrollgeräte“ (ISO 4040 1986).

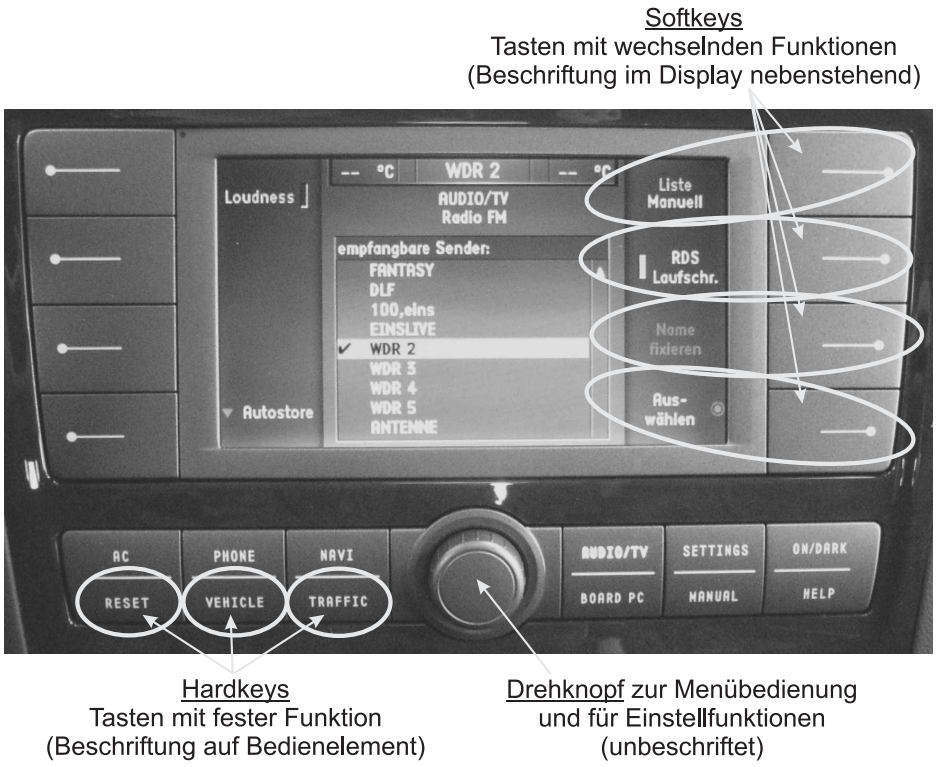

Abbildung 4.2: Hard-, Softkeys und Drehknopf beim VW Phaeton. Die Funktionsbeschriftungen der Softkeys sind im Display, die der Hardkeys auf den Bedienelementen selber angeordnet.

von der bedienenden Hand überdeckt wird. Darüber hinaus ist der *Status- und Meldebereich*, der *Hinweisbereich* und der *Arbeitsbereich* definiert. Die Empfehlung legt die Platzierung der jeweiligen Bereiche fest, wobei nicht alle Bereiche vorkommen müssen. Wird ein Bereich nicht benötigt, steht der dafür vorgesehene Platz somit anderen (vorhandenen) Bereichen zur Verfügung. Zur Verdeutlichung sind in Abbildung 4.3 (rechts) drei Beispiele dargestellt.

Für die Zuordnung von Tasten zu ihren im Display dargestellten Funktionsbeschriftungen wird das Prinzip der *räumlichen Kompatibilität*[4] berücksichtigt. Die räumliche Kompatibilität beschreibt die Zusammengehörigkeit von Anzeigen und Stellteilen aufgrund ihrer Lage (KNAUTH 2004). In Abbildung 4.4 ist jeweils ein Beispiel für eine ein- und mehrdeutige Zuordnung von Softkeys des Audi MMI zu ihren Beschriftungen (in der Anzeige) dargestellt. Die Softkeys sind quadratisch[5] angeordnet. Im linken Beispiel stehen die Beschreibungen der Softkeys („Text1“ bis „Text4“) linear am unteren Displayrand. Die Zuordnung der Texte zu den Tasten ist nicht eindeutig. Offensichtlich gehören „Text1“ und „Text2“ zu den linken beiden Softkeys; mit welchem Text der obere der beiden Softkeys beschriftet ist, lässt sich dagegen nicht eindeutig angeben. Im rechten Beispiel ist die Zuordnung eindeutig, da die quadratische Anordnung der Beschriftungen exakt der Anordnung der Tasten entspricht.

[4]Die *räumliche Kompatibilität* stellt erlerntes Stereotypenwissen dar. Zu diesem Wissen zählt ebenfalls das Prinzip der *Reiz-Reaktions-Kompatibilität*. Beides ist in Abschnitt 4.4.1 vorgestellt.

[5]Die Softkeys sind um einen Dreh-Drück-Steller angeordnet, vgl. Abb. 4.10.

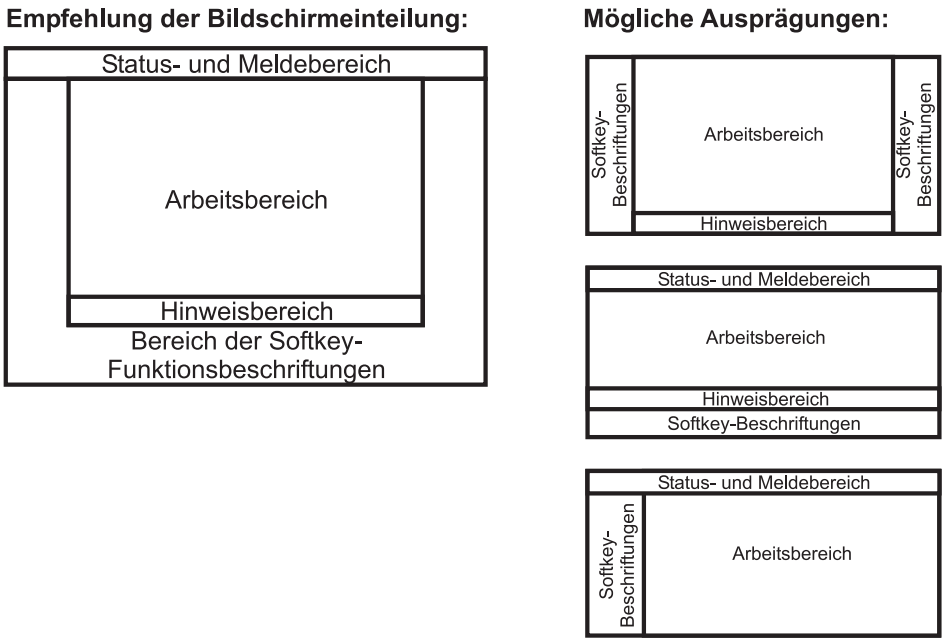

Abbildung 4.3: Empfohlene Einteilung von Displays zur Darstellung unterschiedlicher Informationen (OSACA E.V. 1997). Es müssen nicht alle Bereiche vorhanden sein. Drei mögliche Ausprägungen sind rechts dargestellt.

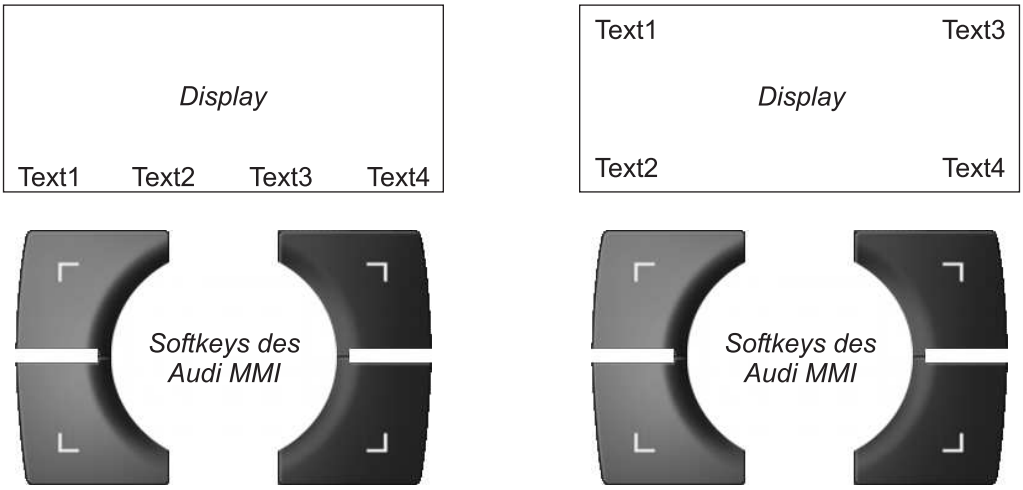

Abbildung 4.4: Mehrdeutige (links) und eindeutige (rechts) Anordnung von Softkeys und Softkey-Beschriftungen am Beispiel der Softkeys des Audi MMI.

Die Darstellung von Listen[6] (z.B. die Senderliste im Radiomenü) oder Menüs[6] erfolgt üblicherweise durch mehrere Textelemente, die vertikal bzw. horizontal angeordnet sind. Darüber hinaus ist das aktuell ausgewählte Menüelement grafisch von den restlichen Menüelementen abgehoben. Das kann durch eine Änderung einer Eigenschaft des Elements selber (Änderung der Farbe oder der Größe) oder die zusätzliche Verwendung eines Grafikelements (Umrah-

[6]Die Darstellung einer Liste unterscheidet sich nicht von der eines Menüs. Aus diesem Grund wird im Folgenden ausschließlich von einer „Menübedienung“ gesprochen. Alle Erklärungen zur einer Menübedienung beziehen sich ohne Einschränkung ebenfalls auf die Bedienung einer Liste.

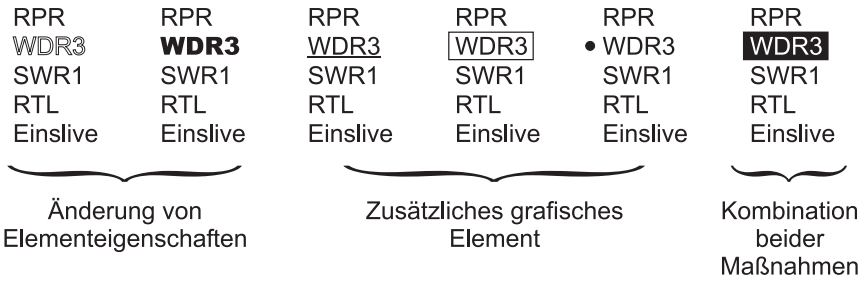

Abbildung 4.5: Darstellungsmöglichkeiten einer Menüselektion. Dabei können die Eigenschaften des selektierten Elementes verändert, zusätzliche grafische Elemente oder eine Kombination beider Maßnahmen zum Einsatz kommen.

mung mit Rechteck, Unterstreichung durch Linie oder Voranstellung eines kleinen Kreises) oder einer Kombination beider Maßnahmen erreicht werden. Abbildung 4.5 verdeutlicht diese Maßnahmen beispielhaft.

4.2.2 Vorgehen bei der Erkennung der GUI-Konstellation

Die dem Agenten vorliegende formale Beschreibung einer GUI-Konstellation beinhaltet alle im GUI dargestellten Objekte. Eine Kategorisierung der GUI-Elemente zur Unterscheidung von Beschriftungen, Überschriften oder Menüs muss durch den Agenten geleistet werden und ist im Folgenden vorgestellt.

Erkennung der Art und Beschriftung von Bedienelementen

Zur Erkennung der Art und Beschriftung von Bedienelementen sind in Abbildung 4.6 die Stufen des Vorgehens dargestellt und im Folgenden erläutert.

Häufig ist die Beschreibung der Bedienelemente nicht auf dem jeweiligen Element selber, sondern darüber oder darunter angebracht, während sich die Beschreibung von Softkeys üblicherweise im Display findet. Diese Zuordnung ist in der formalen GUI-Beschreibung nicht gegeben und muss daher vom Agenten geleistet werden.

Zur Erkennung des Typs der Bedienelemente werden alle Tasten und Druckknöpfe zu einer Gruppe zusammengefasst (die Gruppe der potentiellen Hard- und Softkeys). Alle Bedienelemente, die über einen einfachen Druck hinaus eine rotatorische oder translatorische Bewegung erlauben (z.B. Dreh- oder Schieberegler), werden ebenfalls gruppiert (Gruppe der Bedienelemente zur Menübedienung bzw. Manipulation von Funktionswerten). In der nächsten Stufe erfolgt zur späteren Erkennung von Überschriften oder Softkey-Beschriftungen die Identifizierung aller Textelemente, die auf einem Display dargestellt sind.

Anschließend erfolgt die Kategorisierung der Bedienelemente sowie der Informationen des Displays. Allen Tasten, in deren unmittelbarer Nachbarschaft (z.B. über oder unter dem Taster) ein Text existiert, wird dieser als Funktionsbeschriftung zugeordnet. Diese Tasten ent-

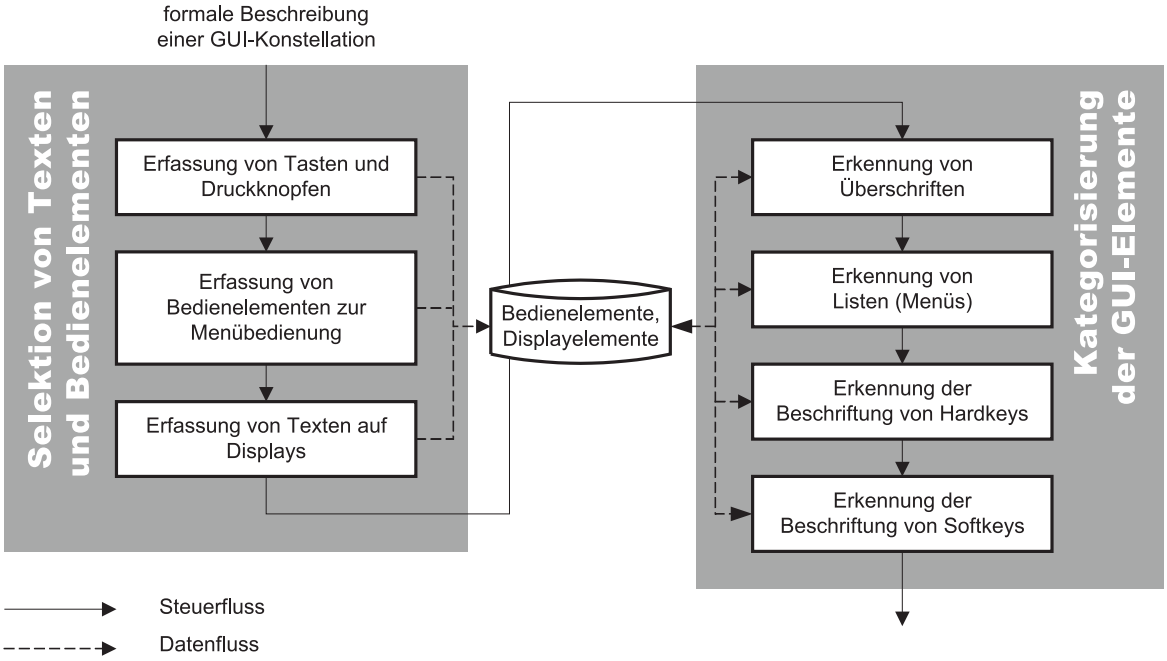

Abbildung 4.6: Stufen zur Erkennung der aktuellen GUI-Konstellation.

sprechen einem *Hardkey*. Tasten, die nicht als Hardkeys erkannt werden konnten, gehören der Gruppe der *Softkeys* an. Für sie muss in einem Display ein Text als Funktionsbeschreibung gefunden werden. Das Vorgehen dazu ist im Folgenden erklärt.

Erkennung der Beschriftung von Softkeys und der Art der auf Displays dargestellten Informationen

Das Vorgehen zur Erkennung der auf Displays dargestellten Informationen orientiert sich an der räumlichen Kompatibilität der Anordnung von Textelementen und Softkeys zueinander (vgl. Kapitel 4.2.1), ist in Abbildung 4.7 dargestellt und im Folgenden erläutert.

Die Analyse der Anordnung der Textelemente auf dem Bildschirm bildet den ersten Schritt der Zuordnung. Dabei werden jeweils zueinander linksbündige und rechtsbündige Texte gruppiert (im Folgenden *Spalten* genannt). Texte mit gleicher horizontalen Ausrichtung bilden weitere Gruppen, im Folgenden mit *Reihen* bezeichnet. Da die Beschriftungen von Softkeys prinzipiell am Displayrand zu erfolgen hat, werden nur die äußerst linke und rechte Spalte sowie die oberste und unterste Reihe betrachtet. Anschließend werden die Gruppen der Textspalten und -reihen daraufhin untersucht, ob sie ein *markiertes*[7] Element beinhalten. Spalten bzw. Reihen mit markierten Elementen stellen potentiell ein Menü mit markierter Menüoption dar.

Zur Zuordnung der Beschriftungen erfolgt zunächst die Betrachtung der Lage der Softkeys. Diese werden ebenfalls in *Spalten* und *Reihen* gruppiert. Wenn mehr als zwei Spalten bzw. Reihen existieren, lassen sich die Beschriftungen nicht mehr eindeutig zuordnen. Das lässt

[7] Zulässige Markierungen sind in Abbildung 4.5 dargestellt.

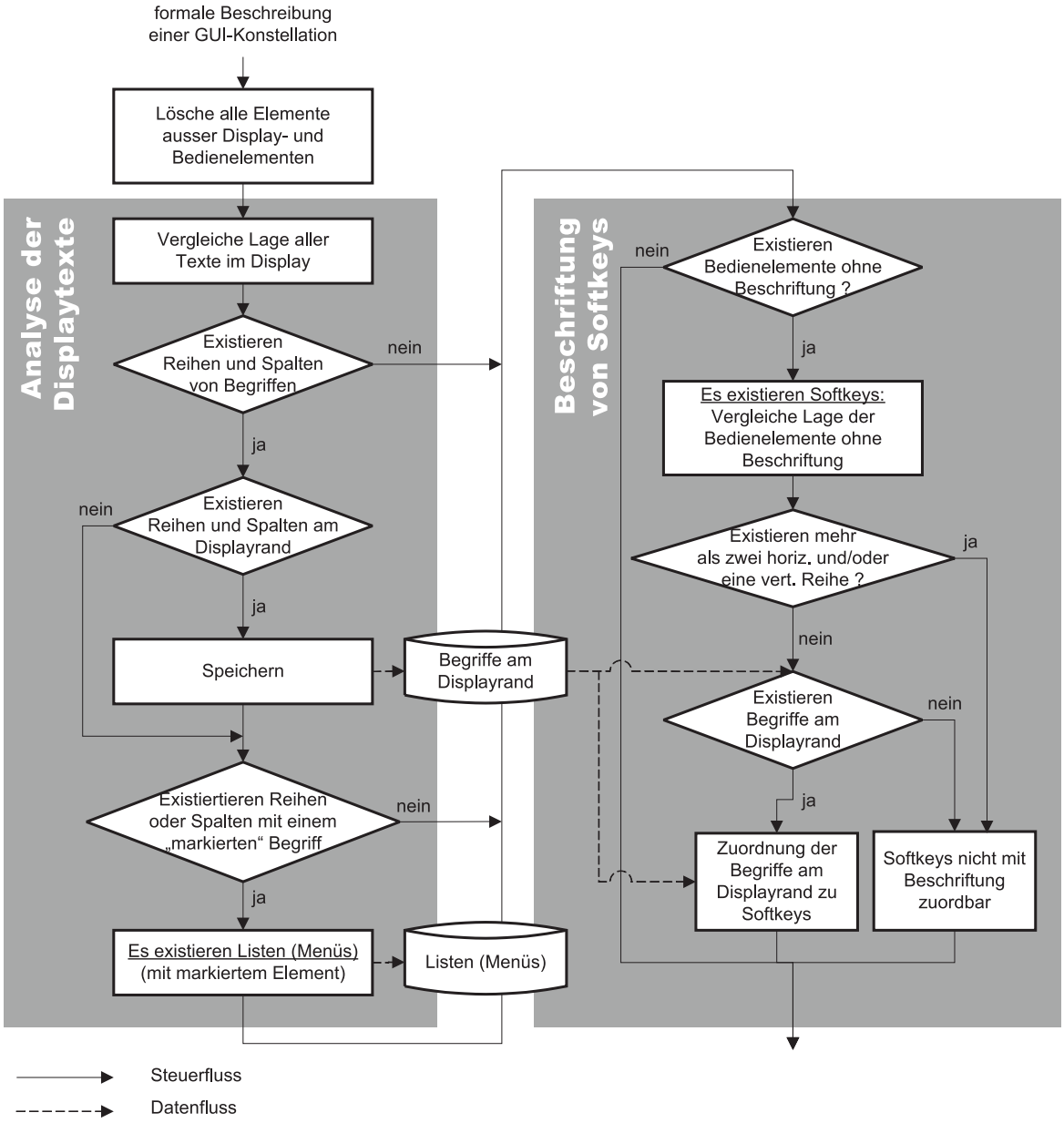

Abbildung 4.7: Schritte der Zuordnung von Funktionsbeschreibungen als Beschriftungen von Softkeys im Agenten.

sich dadurch erklären, dass ein Display lediglich zwei horizontale und zwei vertikale Ränder hat, an denen prinzipiell Beschriftungen von Softkeys stehen können. Existieren jedoch mehr als zwei Spalten oder Reihen von Softkeys, kann die Beschriftung der mittleren Spalte/Reihe nicht mehr am Displayrand liegen, sondern muss weiter im Displayinneren platziert sein. Das widerspricht jedoch der Empfehlung zur Displayaufteilung (vgl. Abb. 4.3) in (OSACA E.V. 1997; ISO 9241-12 1998). In einem solchen Szenario kann nicht mehr zwischen der Beschriftung einer mittleren Spalte/Reihe von Softkeys und auf dem Display dargestellten Informationen bzw. Menüs unterschieden werden. Aus diesem Grund bricht die Verarbeitung der Zuordnung von Softkey-Beschriftungen ab, wobei eine entsprechende Fehlermeldung als

Hinweis auf das schlechte Oberflächendesign erzeugt wird. Existieren dagegen maximal zwei Spalten bzw. Reihen, erfolgt die Zuordnung der gefundenen Texte als Funktionsbeschriftung zu den jeweiligen Softkeys.

Die Zuordnung der Softkeys und deren Beschriftung durch das Prinzip der *räumlichen Kompatibilität* hat den Vorteil, dass Softkeys nicht direkt neben einem Display angeordnet sein müssen (wie beim FIS des VW Phaeton, vgl. Abb. 4.2 auf Seite 77), sondern unabhängig vom Display platziert werden können (wie beim Audi MMI, vgl. Abb. 4.10 auf Seite 84).

4.3 Identifikation des Folgezustands zur Erreichung des Aufgabenziels

Bei komplexen interaktiven Systemen erfolgt der Zugriff auf Systemfunktionen über Menüs. Zur Erreichung eines Aufgabenziels muss der Agent das System über mehrere Menüebenen hinweg bedienen. Zur Orientierung dient ihm dabei ein Begriffspfad, der alle Begriffe einer Ontologie (als Modell einer allgemein gültigen Dialoghierarchie) vom Hauptmenü (*idle*) bis zum Zielbegriff enthält. Die Erstellung eines solchen Begriffspfades und das Vorgehen bei der Navigation und Zielfindung ist in den folgenden Abschnitten erläutert.

4.3.1 Angabe eines Aufgabenziels

Ein allgemein gültiges Modell der Dialoghierarchie bzw. des Menüaufbaus interaktiver Systeme einer Domäne steht dem Agenten in Form einer Begriffsontologie[8] zur Verfügung. Dadurch ist der Agent in der Lage, Funktionen und Informationen innerhalb menübedienter Systeme zu lokalisieren.

Zur Bearbeitung einer Aufgabe muss dem Agenten ein Aufgabenziel angegeben werden. Dieses Ziel besteht bei FIS im Erreichen einer bestimmten Systemfunktionalität. Eine Ontologie enthält alle Funktionsbeschreibungen, die in einem FIS (bzw. in den Menüs eines FIS) vorkommen können. Daher genügt zur Definition eines Ziel die Auswahl eines Knotens einer Ontologie mit der Beschreibung der gewünschten Zielfunktion. Beispielhaft ist in Abbildung 4.8 die Funktion „Abspielen (einer CD)“ als Ziel einer Aufgabe für die in Abbildung 3.5 dargestellten FIS-Ontologie gewählt. Somit ergibt sich vom Startzustand des Systems („*idle*“) bis zum gewählten Zielzustand („Abspielen“) ein Pfad mit Begriffen, anhand derer sich der Agent orientieren und durch die Systemmenüs navigieren kann.

Ontologien stellen Graphen der Begriffe mit Verknüpfungen dar (vgl. Kapitel 3.3). Das Vorgehen bei der Umwandlung eines Begriffsgraphen in einen Begriffsbaum und die Generierung des Begriffspfades ist in Abbildung 4.9 dargestellt und im Folgenden erläutert.

[8] Ontologien sind im Abschnitt 3.3 vorgestellt.

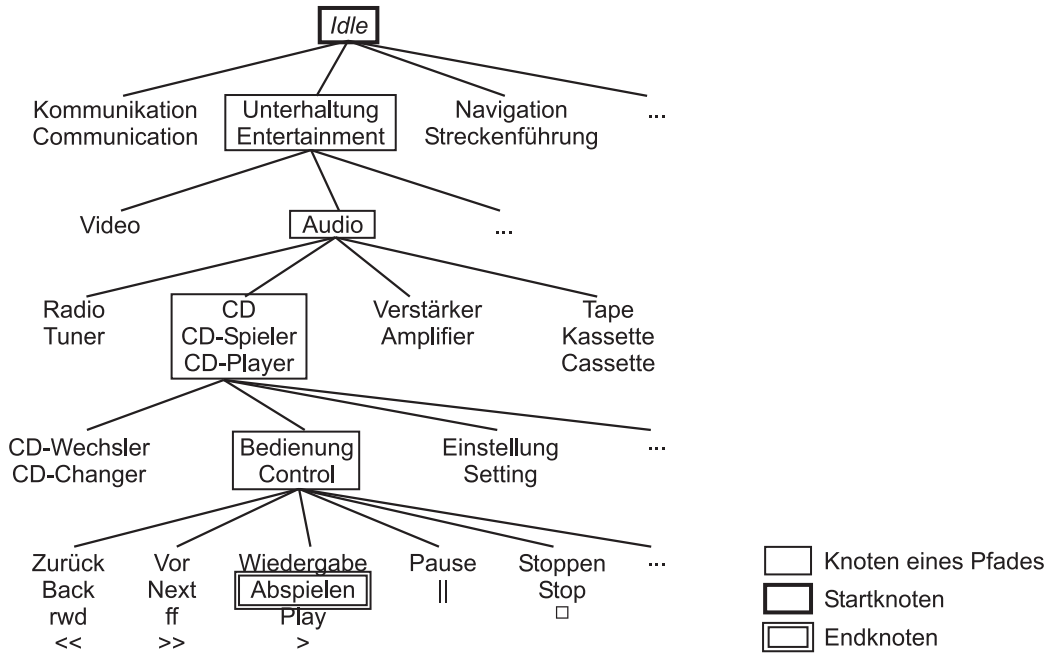

Abbildung 4.8: Beispiel der Zieldefinition in einer Ontologie (vgl. Abb. 3.5). Durch die Angabe der Zielfunktion ergibt sich vom Startknoten ein Pfad durch den Baum.

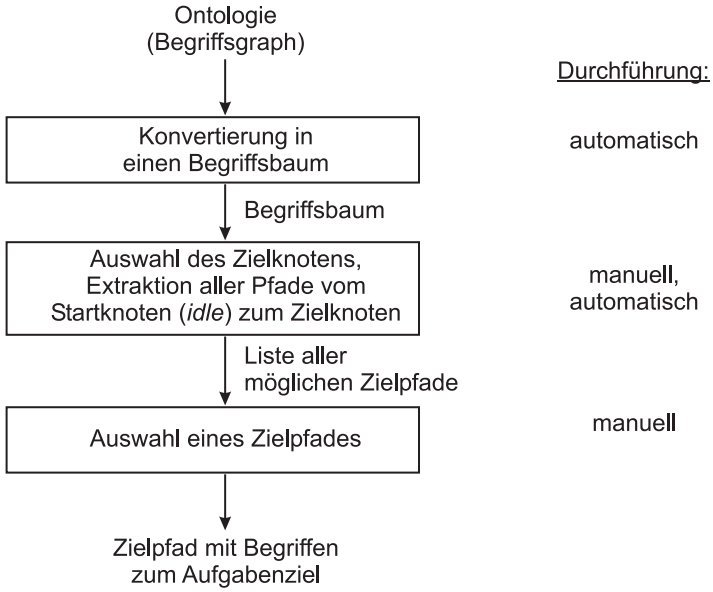

Abbildung 4.9: Schritte zur Auswahl eines Begriffspfades aus einer Begriffsontologie. Die Generierung des Begriffspfades nimmt REVISER halbautomatisch vor.

Die Ontologie lässt sich automatisch durch das Auflösen aller Zyklen in einen Begriffsbaum umwandeln. Den Wurzelknoten stellt dabei *idle* als Repräsentant des Hauptmenüs dar. Zur Generierung des Begriffspfades muss der Begriff, der die zu erreichende Zielfunktion beschreibt, ausgewählt werden. Der Knoten, der in der Ontologie diesen Begriff enthält, repräsentiert den Zielknoten eines Begriffspfades, während der Startknoten durch die Wurzel des Baums (*idle*) festgelegt ist. Anschließend wird aus der Ontologie automatisch der entsprechende Begriffs-

pfad extrahiert. Kommt der Zielbegriff in mehreren Knoten der Ontologie vor, dann erfolgt die automatische Extraktion aller möglichen Begriffspfade. Nach der manuellen Auswahl eines der vorgeschlagenen Begriffspfade steht dem Agenten dieser zur Navigation zur Verfügung.

Die Möglichkeiten der Orientierung in einem Menüsystem mit Hilfe eines Begriffspfades wird im nächsten Abschnitt erläutert.

4.3.2 Vorgehen zum Erreichen des Aufgabenziels

Zur Navigation in Menüs vergleicht der Agent die im GUI dargestellten Begriffe mit dem zur Verfügung stehenden Begriffspfad (vgl. Kapitel 4.3.1). Das Vorgehen bei der Orientierung ist im Folgenden anhand einer beispielhaften Bedienung des Audi MMI erläutert und in den Abbildungen 4.10, 4.11 und 4.12 dargestellt. Dabei soll – ausgehend vom Navigationsmenü – die Anwahl der Funktion „Höhen einstellen“ im Radiomenü erfolgen.

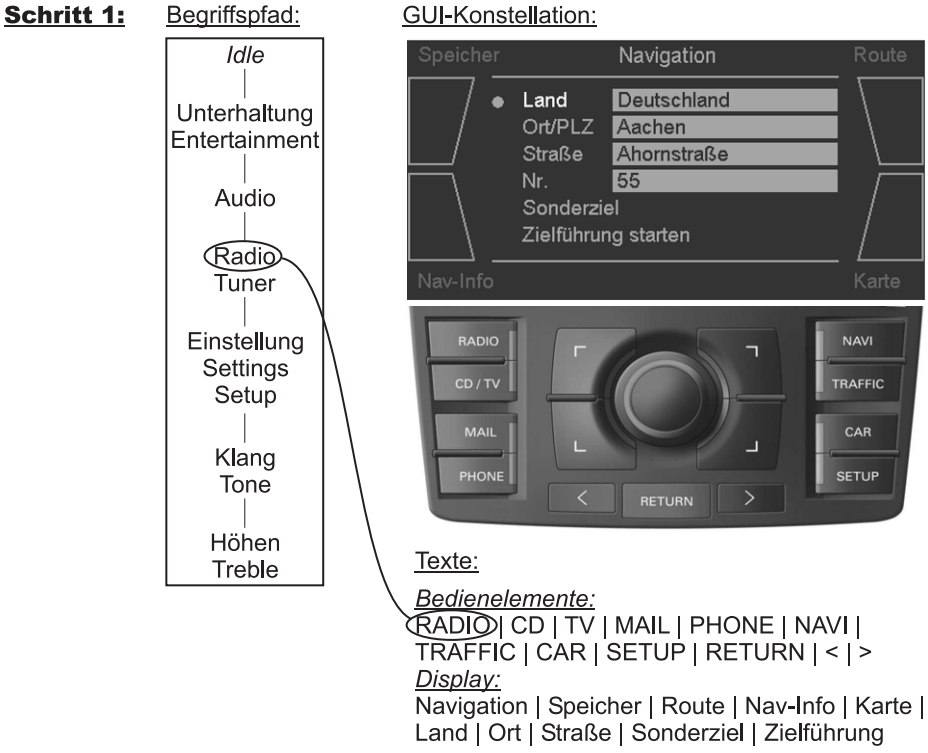

Abbildung 4.10: Beispiel der Orientierung anhand eines Begriffspfades zur Bedienung des Audi MMI. Ziel ist die Anwahl der Funktion „Höhen“. Links ist der zur Orientierung benutzte Begriffspfad dargestellt, rechts sind das Display und die Bedienelemente des Audi MMI abgebildet sowie alle Texte des GUI aufgelistet. Im ersten Schritt wird „Radio“ als Zwischenziel ausgewählt und eine entsprechende Aktion durchgeführt.

Die in einer GUI-Konstellation dargestellten Begriffe werden gesammelt und mit dem Begriffspfad abgeglichen. Das Ergebnis ist der Knoten des Begriffspfades mit dem kürzesten Abstand zum Zielknoten, der einen Begriff aus der aktuellen GUI-Konstellation enthält. Han-

delt es sich dabei um den Endknoten des Begriffspfades, dann ist das Ziel bereits erreicht[9]. Handelt es sich jedoch um einen Knoten innerhalb des Begriffspfades (im Beispiel: der Begriff „Radio“), dann wird dieser Knoten als Zwischenziel markiert und der Agent führt eine Aktion zum Erreichen dieses Zwischenziels aus (im Beispiel: Druck auf die Taste „RADIO“). Anschließend werden aus dem Begriffspfad alle Knoten bis zum Zwischenziel (inklusive) entfernt, da ansonsten im nächsten Schritt ggf. das gleiche Zwischenziel gefunden wird und der Agent in eine Endlosschleife gerät. Die entfernten Knoten des Begriffspfades sind im Beispiel grau hinterlegt.

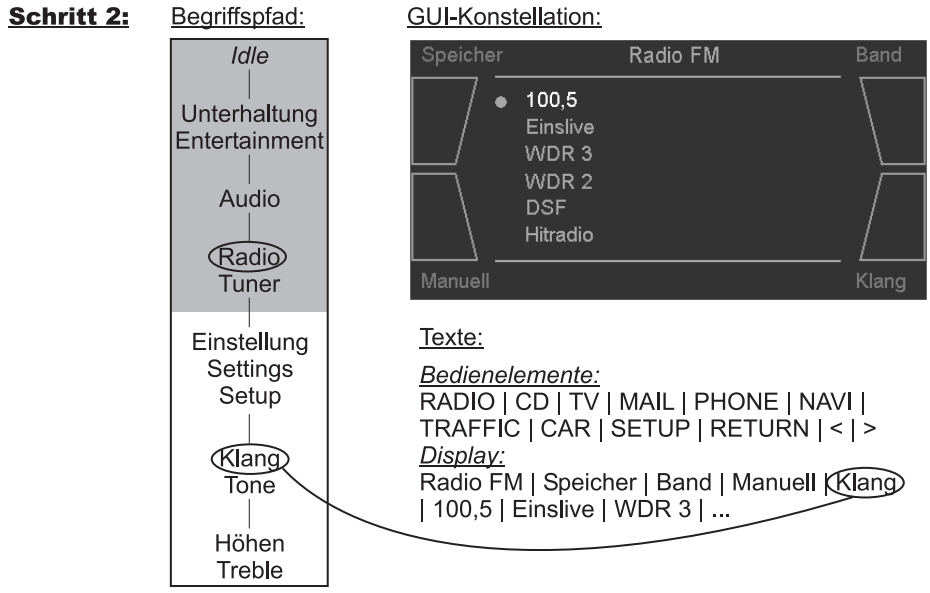

Abbildung 4.11: Beispiel der Orientierung anhand eines Begriffspfades. Im zweiten Schritt wird „Klang“ als Zwischenziel ausgewählt und eine entsprechende Aktion durchgeführt.

Dieses Vorgehen ist auch für die folgenden Schritte stets gleich, so dass im Beispiel der Agent im zweiten Schritt (Abb. 4.11) den Begriff „Klang“ als Zwischenziel findet und eine entsprechende Aktion – Druck des Softkeys „Klang“ (Taste südöstlich des Drehstellers) – ausführt. In Schritt 3 (Abb. 4.12) findet der Agent zwar bereits den Zielbegriff „Höhen“, jedoch ist dieser nicht markiert, die Funktion demnach noch nicht ausgewählt. Daher müssen weitere Aktionen zur Selektion der Menüoption „Höhen“ erfolgen[10].

Eine zusätzliche Sicherheit der Bedienung geben Überschriften von Menüs. Alle Begriffe von Zwischenzielen eignen sich als Überschriften, so dass der Einsatz von Begriffen als Überschriften durch einen Vergleich der letzten Zwischenziele überprüft werden kann (in den Abbildungen 4.11 und 4.12 dargestellt durch die umkreisten, aber grau hinterlegten Begriffe im Begriffspfad).

[9] Allerdings muss der Agent noch feststellen, ob die entsprechende Funktion bereits ausgewählt ist. Das ist der Fall, wenn der entsprechende Zielbegriff z.B. als Überschrift oder markiertes Element eines Menüs dargestellt ist.

[10] Das Vorgehen dazu ist in Abschnitt 4.4.2 mit dem gleichen Beispiel vorgestellt.

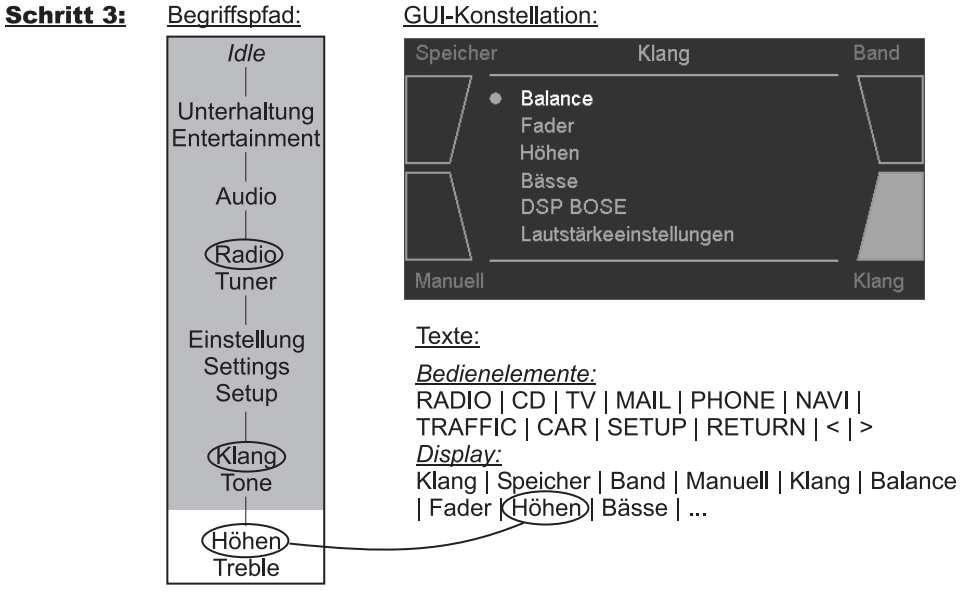

Abbildung 4.12: Beispiel der Orientierung anhand eines Begriffspfades. Im dritten Schritt wird „Höhen“ als Endziel gefunden. Allerdings ist die Funktion nicht angewählt und weitere Aktionen zur Menübedienung müssen erfolgen.

4.4 Bewertung der Anzeige-Bedienelemente-Konstellation zur Auswahl einer Bedienaktion

Nach der Auswahl eines Begriffs, der bei der Menünavigation als Ziel bzw. Zwischenziel erkannt wurde, muss anschließend eine Aktion am interaktiven System durchgeführt werden. Dazu kommen allgemein gültige Bedienstereotypen zum Einsatz. Diese Stereotypen sowie das Vorgehen bei der Auswahl und Bewertung einer Bedienaktion sind in den folgenden Abschnitten erläutert.

4.4.1 Bedienstereotypen zum Abbilden erlernten Bedienwissens

Bei der Bedienung interaktiver Geräte greifen Benutzer auf Bedienwissen zurück, das sich im Laufe der Jahre aufgrund von Erfahrung gebildet hat. Ein Teil dieses Wissens besteht aus *Stereotypen*[11], die kompatible Zuordnungen beschreiben. Dazu gehört die *räumliche Kompatibilität* (z.B. die räumliche Zuordnung von Bedienelementen zu Funktionsbeschreibungen – s. Kapitel 4.2.1) sowie die *Reiz-Reaktions-Kompatibilität* (die Zuordnung der Bedienung von Stellteilen und daraus resultierenden Reaktionen eines Systems – im Folgenden *Bewegungsstereotypen* genannt) (COULSON et al. 2003; KNAUTH 2004). Erst mit Hilfe systemübergreifender Stereotypen ist ein Transfer von Bedienwissen zwischen unterschiedlichen Systemen möglich (MCDONALD et al. 1986). Da solche Stereotypen auf Erfahrungen beruhen, können

[11] Andere gängige Bezeichnungen für Stereotypen in diesem Zusammenhang sind *(Bedien-)Metaphern* bzw. *mentale Modelle*, s. Glossar.

sie sehr stark ausgebildet sein und müssen beim Interaktionsdesign unbedingt berücksichtigt werden (NORMAN 1982; MCDONALD et al. 1986; NORMAN 2002; SZAMEITAT 2003).

Bewegungsstereotypen (Reiz-Reaktions-Kompatibilität) beschreiben die erwartete, kompatible Reaktion eines Systems auf eine Aktion (Reiz) des Benutzers (Bewegung eines Stellteils, z.B. Schieben eines Reglers). So muss bei der Bedienung eines Stellteils eine zur Bewegung kompatible Systemreaktion erfolgen. Die Systemreaktion kann sich dem Benutzer dabei auf unterschiedliche Weise darstellen, z.B. durch die Veränderung einer Skalenanzeige. Beispielhaft wird im Folgenden der Zusammenhang einer Stereotype „Drehen eines Drehreglers im Uhrzeigersinn (UZS)“ mit der Veränderung eines Wertes einer Systemfunktion vorgestellt.

Bei der Benutzung eines Drehreglers zur Bedienung von Alltagsgeräten ohne zusätzliche Anzeige des einzustellenden Wertes gilt grundsätzlich die Stereotype (HOFFMANN 1997): „Drehen eines Drehreglers im UZS bewirkt eine Erhöhung (z.B. „wärmer“, „lauter“) der Stellgröße“. Beispielhaft sind dazu in Abbildung 4.13 auf der linken Seite einige Drehregler von Alltagsgeräten mit den entsprechenden Reaktionen dargestellt.

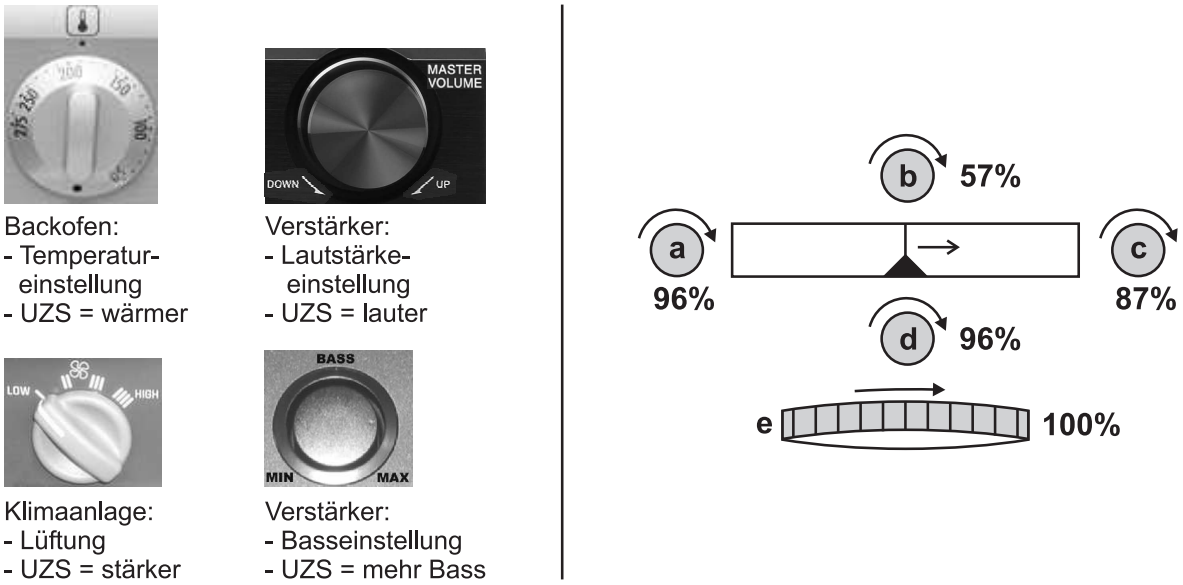

Abbildung 4.13: Beispiele für die Stereotype „Drehen im Uhrzeigersinn (UZS)“ eines Drehreglers. Links sind die erwarteten Reaktionen von Alltagsgeräten dargestellt. Rechts ist die stereotype Veränderung einer Skalenanzeige in Abhängigkeit der Lage von Skala und Stellteilen abgebildet. Die Prozentwerte geben die Häufigkeit der Erwartung der Reaktion „Markierung nach Rechts“ an.

Wird bei der Betätigung des Drehreglers die Veränderung eines Wertes durch eine Skalenanzeige dargestellt, dann ist die Bewegungsrichtung der Skalenmarkierung abhängig von der Lage des Displays und des Stellteils zueinander (WICKENS et al. 1984). In Abbildung 4.13 ist auf der rechten Seite beispielhaft eine Skala mit fünf Drehreglern dargestellt. Hoffmann untersuchte experimentell die Stärke der Stereotype: „Drehen des Drehreglers im UZS bewirkt eine Verschiebung der Skalenmarkierung nach rechts“. Dabei kamen mehrere Drehsteller in unterschiedliche Lagen (a-d) zum Einsatz. Die aus der Untersuchung resultierenden Erwar-

tungswerte sind in Abb. 4.13 als Prozentwerte angegeben. Lage (a), (c) und (d) ermöglichen eine erwartungskonforme Bedienung. Bei einem Drehregler in Lage (b) erwarten lediglich 57% der Probanden eine Bewegung der Skalenmarkierung nach rechts. Aus diesem Grund sollte auf eine Platzierung eines Drehreglers an dieser Stelle verzichtet werden (HOFFMANN 1997). Eine noch höhere Erwartungskonformität kann durch den Einsatz eines Drehrades (e) zur Bedienung einer linearen Skalenanzeige erreicht werden (KRAISS 2005).

Bei der Betrachtung der Anordnung von Bedienteilen und Displays im Raum muss auch die Ebene der translatorischen bzw. rotatorischen Bewegung und Anzeige berücksichtigt werden. Beispielhaft sind zwei Anzeige-Bedienelemente-Konstellationen des Audi MMI in Abbildung 4.14 dargestellt und im Folgenden beschrieben.

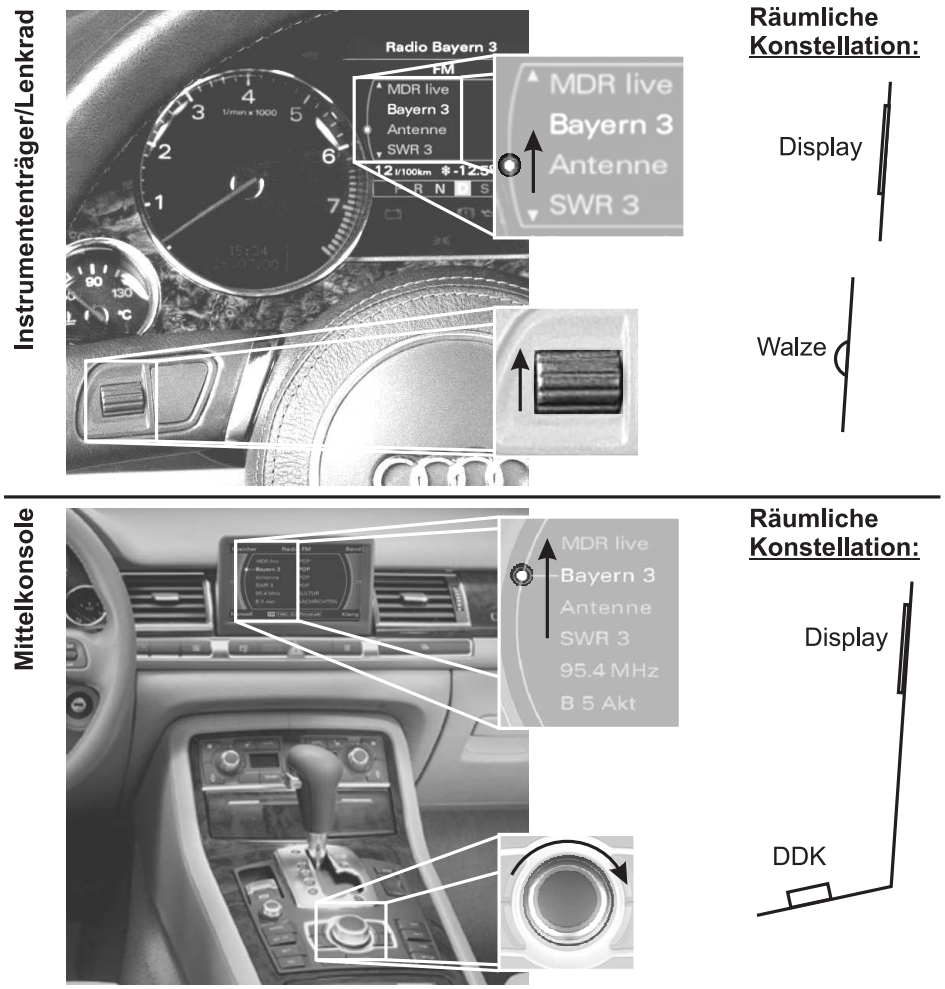

Abbildung 4.14: Zwei Stereotypen zur Listenbedienung im Audi MMI. Im oberen Bild ist die Bedienung der Senderliste mit einer Walze im Lenkrad dargestellt. Unten ist die Bedienung mit dem Dreh-Drück-Knopf (DDK) der Mittelkonsole abgebildet. Die räumliche Konstellation von Bedienelement und Display ist ebenfalls aus der seitlichen Perspektive dargestellt.

Das obere Bild zeigt im Display des Instrumententrägers eine Senderliste, die durch das Drehen einer im Lenkrad angebrachten Walze bedient wird. Dadurch ist eine erwartungskonforme Bedienung gewährleistet, da sich aus der Sicht des Fahrers sowohl die Drehrichtung der Wal-

ze als auch die Bewegungsrichtung der Listenmarkierung in der gleichen räumlichen Ebene befindet. Im unteren Bild von Abbildung 4.14 ist das Display und das Bedienelement der Mittelkonsole dargestellt. Die Bedienung der Liste erfolgt hier durch das DDK, wobei zwei Inkompatibilitäten auftreten. Zum Einen bewirkt die rotatorische Bewegung des DDK eine translatorische Bewegung der Listenmarkierung, zum Zweiten findet die Bewegung des DDK und die Bewegung der Listenmarkierung auf unterschiedlichen räumlichen Ebenen statt. Diese beiden Beispiele zeigen deutlich, dass Bedienstereotypen, in denen z.B. die Bewegungsrichtungen und -ebenen von Bedienelementen und Anzeigen beschrieben sind, sehr genau beachtet werden müssen. In der Literatur finden sich zahlreiche Untersuchungen der Stärke von Bewegungsstereotypen unterschiedlicher Stellteile und Skalenanzeigen sowie deren zwei- und dreidimensionaler Anordnungsmöglichkeiten, z.B. (SCHMIDTKE 1981; WICKENS et al. 1984; HOFFMANN 1997; FISCHER 1999; MACLEOD 2005).

Zur Definition von Bewegungsstereotypen im Rahmen dieser Arbeit ist die Angabe der Art des Bedienelementes (z.B. „Drehregler“ oder „Taste“) sowie die Aktion (z.B. „press“ für einen Tastendruck, „turn [right,left]“ zur Definition einer Drehrichtung oder „slide [north,west,...]“ zur Angabe einer Schieberichtung) nötig. Darüber hinaus muss die erwartete Reaktion (z.B. „up/down“ oder „select“ für eine Menübedienung) angegeben werden. Ein zusätzliches Rating zur Angabe, wie stark die Stereotype ausgebildet ist und somit während der Bedienung berücksichtigt werden muss, vervollständigt die Definition einer Stereotype.

Zur Bedienung eines Lautstärkereglers lässt sich beispielhaft folgende Stereotypen zum Drehen im Uhrzeigersinn (UZS) bzw. gegen den Uhrzeigersinn (GUZS) definieren (ohne Rating):

```
Stereotype: Element=Drehregler, Aktion=Drehen-UZS, Reaktion=lauter
Stereotype: Element=Drehregler, Aktion=Drehen-GUZS, Reaktion=leiser
```

Zur Bedienung eines interaktiven Systems muss der Agent in der Lage sein, sein Verhalten auf Grund von Rückschlüssen aus vorhergehenden Aktionen und den daraus folgenden Reaktionen der Umwelt anzupassen (RUSSEL und NORVIG 2004). Häufig wird dies durch eine Bewertung in Form von Gewichten für alternative Aktionsmöglichkeiten erreicht (LUGER 2001). Im Bedienagenten ist dies durch eine entsprechende Bewertung von Bedienhypothesen möglich: Vor dem Durchführen einer Aktion speichert der Agent die Erwartung, wie das System auf die Aktion reagieren wird, in Form einer Hypothese ab. Nach Durchführung der Aktion wird die Antwort des Systems (formale GUI-Beschreibung) daraufhin überprüft, ob die Erwartungen eingetroffen sind oder nicht. Dadurch kann das Wissen, das zur Auswahl der Aktion zum Einsatz kam (z.B. in Form einer Bedienstereotype) bewertet werden. Dazu eignet sich ein Ranking der Stereotypen, das bei Bestätigung einer Hypothese erhöht bzw. bei Widerlegung einer Hypothese vermindert wird. Dieses Vorgehen ist beispielhaft anhand einer Menübedienung in Abbildung 4.15 dargestellt.

4.4.2 Vorgehen zur Auswahl einer geeigneten Bedienaktion

Der Agent orientiert sich ausschließlich über die im GUI dargestellten Begriffe. Dazu bedient er sich einer Begriffsontologie. Allerdings enthält eine Ontologie keine Angaben darüber, was

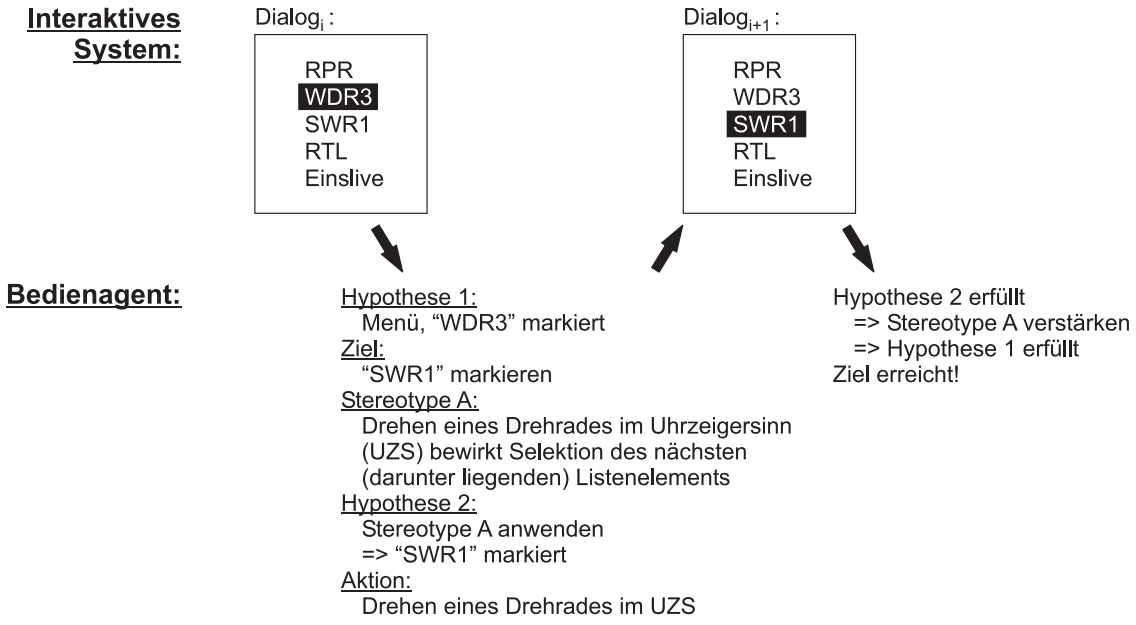

Abbildung 4.15: Beispiel der Bewertung einer Bedienhypothese im Bedienagenten zur Verstärkung von benutzten Bedienstereotypen.

die Begriffe im GUI eines interaktiven Systems beschreiben. So ist nicht angegeben, ob ein Begriff die Beschriftung einer Taste oder ein Element eines Menüs darstellt (z.B. die Taste „RADIO“ und das Menüelement „Sonderziel“ des Audi MMI in Abb. 4.10).

Grundsätzlich lassen sich die Bedienelemente eines interaktiven Systems in *Hardkeys*, *Softkeys* und *Bedienelemente zur Menübedienung* unterscheiden. Bei einer Menübedienung sind Aktionen zur Auswahl (z.B. „nach oben“ bzw. „nach unten“) und Selektion (z.B. „Drücken“) von Menüoptionen nötig. Dazu kommen Bedienelemente wie Dreh-Drück-Steller, Joysticks oder Navigationswippen zum Einsatz, die üblicherweise keine Beschriftung haben.

Hard- und Softkeys besitzen normalerweise als Beschriftung eine Funktions- bzw. Menübeschreibung und erlauben durch die Aktion „Drücken“ die Anwahl dieser Funktion bzw. dieses Menüs. Findet der Agent bei der Navigation (s. vorhergehender Abschnitt) einen Begriff als Ziel bzw. Zwischenziel, der gleichzeitig eine Beschreibung eines Hard- oder Softkeys darstellt, dann führt er als Aktion das Drücken der jeweiligen Taste durch.

Handelt es sich bei einem Zielbegriff (bzw. Begriff eines Zwischenziels) um einen Begriff, der nicht direkt einer Taste zugeordnet werden kann, muss eine Menübedienung erfolgen. Das Vorgehen der Selektion und Ausführung einer Aktion zur Menübedienung ist in Abbildung 4.16 dargestellt und im Folgenden erläutert. Dabei dient als Beispiel[12] die Bedienung eines Menüs mit den Menüelementen „Balance“, „Fader“ und „Höhen“ mit Hilfe eines Dreh-Drück-Stellers. Das Bedienelement sowie Stereotypen zur Bedienung sind ebenfalls in Abb. 4.16 (rechts oben) abgebildet.

[12]Das Beispiel beschreibt die fehlenden Bedienschritte des Beispiels in Abschnitt 4.3.2.

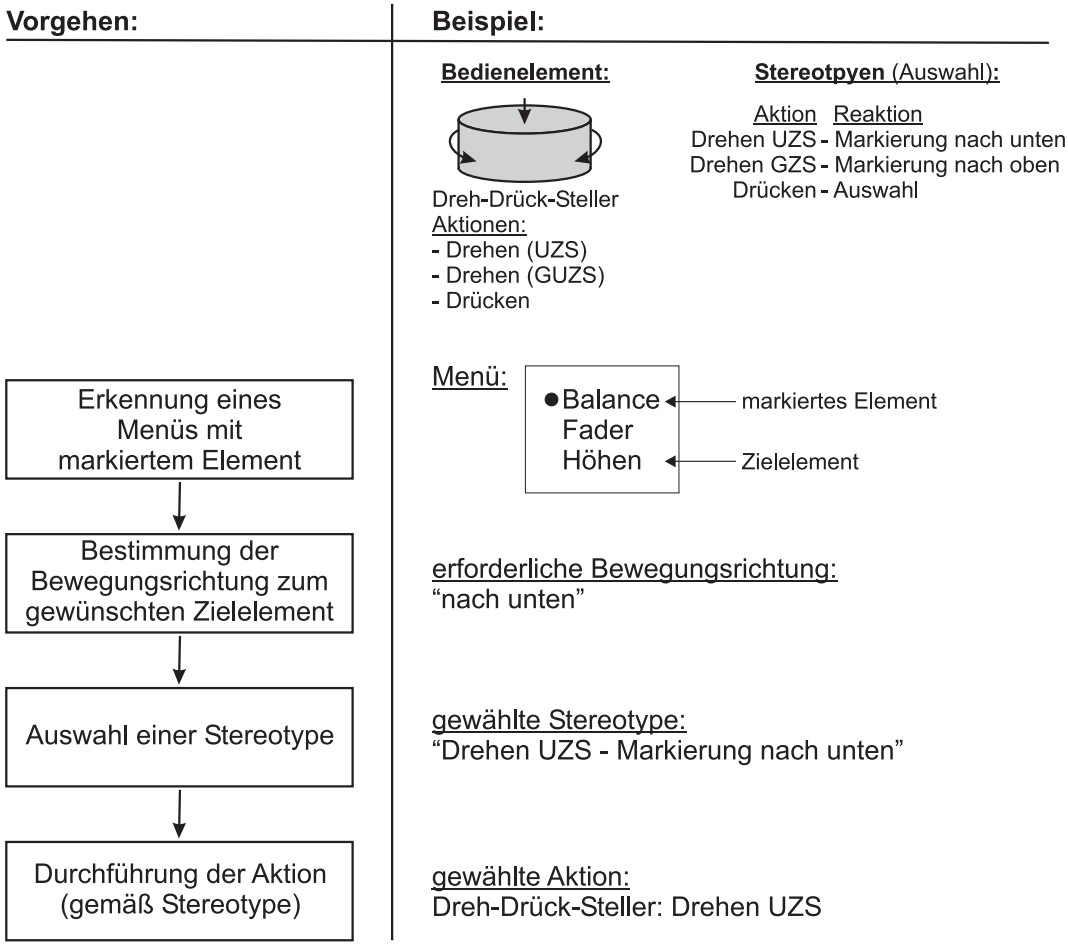

Abbildung 4.16: Schritte zur Auswahl einer Bedienaktion. Auf der rechten Seite ist beispielhaft die Bedienung eines Menüs dargestellt.

Im ersten Schritt benötigt der Agent das Wissen, welche Begriffe zu Menüs zusammengefasst sind, und welches Element markiert ist (vgl. Abschnitt 4.2). Bedienelemente, denen mehr als eine Aktion (z.B. „Drehen im Uhrzeigersinn“, „Drehen gegen den Uhrzeigersinn“ und „Drücken“ bei einem Dreh-Drück-Steller) zugeordnet sind, klassifiziert der Agent automatisch als Bedienelement zur Menübedienung.

Ist das aktuell markierte Element des Menüs nicht gleichzeitig auch das gewünschte Ziel, dann ergibt sich aufgrund der linearen Anordnung der Menüelemente automatisch die Richtung, in die die Markierung zum Zielelement verschoben werden muss (Schritt 2).

Die Definition von Bedienstereotypen enthält die Angabe einer Aktion eines Bedienelementes sowie der erwarteten Reaktion (vgl. Abschnitt 4.4.1). Grundsätzlich berücksichtigt der Agent nur solche Stereotypen, die Aussagen zur Kompatibilität tatsächlich vorhandener Bedienelemente enthalten. Eine Stereotype mit der Definition einer Schieberichtung wird nur dann benötigt, wenn auch ein Bedienelement vorhanden ist, dass eine Schieberichtung erlaubt.

Da die gewünschte Bewegungsrichtung bekannt ist (Schritt 2) muss im dritten Schritt eine Ste-

reotype gewählt werden, die eine entsprechende kompatible Bewegungsrichtung beschreibt. Kommen dazu mehrere Stereotypen in Frage, erfolgt die Auswahl anhand des zu jeder Stereotype angegebenen Ratings. Das Ergebnis dieses Schrittes ist die Stereotype mit dem höchsten Rating; bei Stereotypen mit gleichem Rating entscheidet die Reihenfolge, in der die Stereotypen im Agenten gespeichert sind. Im Beispiel existiert nur eine Stereotype, die die Bewegungsrichtung „nach unten“ enthält. Durch die in der ausgewählten Stereotype definierte Aktion ergibt sich im letzten Schritt die durchzuführende Bedienaktion.

Die Ergebnisse dieser Aktionsauswahl werden (wie in Abschnitt 4.4.1 erklärt und in Abbildung 4.15 dargestellt) in Form einer Hypothese gespeichert, um im nächsten Schritt die Richtigkeit der angewandten Stereotype bestätigen oder widerlegen und entsprechend das Rating der Stereotype anpassen zu können.

Kapitel 5

Aufbau und Realisierung des Werkzeugs REVISER zur kriterienorientierten Bewertung interaktiver Systeme

In den vorangegangenen Kapiteln wurden die theoretischen Grundlagen und Anforderungen zur Entwicklung eines Werkzeugs zur automatischen kriterienorientierten Bewertung interaktiver Systeme erläutert. Dieses Kapitel befasst sich mit der Entwicklung des Werkzeugs REVISER (Rapid EValuation of Interactive Systems using ExpeRt knowledge)[1]. Zunächst wird in Abschnitt 5.1 ein Überblick über die Komponenten und die Architektur des Werkzeugs gegeben, dessen einzelne Module in den darauf folgenden Abschnitten beschrieben sind. Abschnitt 5.2 stellt den Kriterieneditor des Werkzeugs vor, der die Eingabe von Regeln ermöglicht. Die Ablaufsteuerung der Bewertung eines Systems durch eine Regelüberprüfung ist in Abschnitt 5.3 erläutert.

5.1 Überblick über das Expertensystem REVISER

Ein Expertensystem bildet das Spezialwissen und die Schlussfolgerungsfähigkeit von Fachleuten programmtechnisch ab. Das Spezialwissen besteht dabei aus Fakten des Fachgebietes, in dem das Expertensystem eingesetzt werden soll. Die Schlussfolgerungsfähigkeit wird in Form von Wenn-Dann-Regeln implementiert (PUPPE 1991).

Im Allgemeinen besteht ein Expertensystem aus einem Steuersystem und einer Wissensbasis. Die Wissensbasis besteht aus den anzuwendenden Fakten und Regeln, während das Steuersystem Module zur Ein- und Ausgabe von Wissen sowie die Problemlösungskomponente enthält. Die Beziehung dieser Module zueinander ist in Abbildung 5.1 dargestellt und im Folgenden erläutert (nach (PUPPE 1991)).

[1] Weitere Informationen zu REVISER finden sich unter http://www.hamacher-online.com/reviser

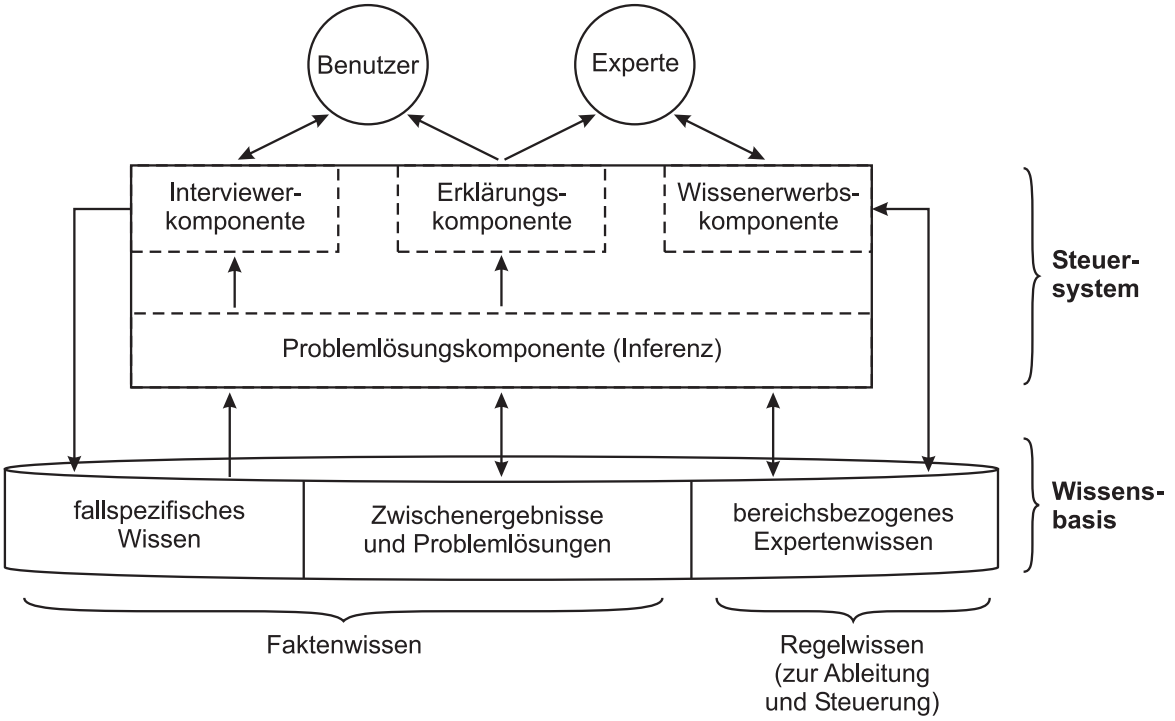

Abbildung 5.1: Grundsätzlicher Aufbau eines Expertensystems nach (PUPPE 1991) mit allen Modulen zur Wissensein- und -ausgabe, zur Durchführung der Inferenz sowie dem benötigten Fakten- und Regelwissen.

Im Rahmen des Steuersystems interpretiert die *Problemlösungskomponente* das Expertenwissen durch die Anwendung der Regeln auf die Fakten der Wissensbasis mit Hilfe eines Inferenzalgorithmus (KURBEL 1992). Die *Wissenserwerbskomponente* ermöglicht Experten, Wissen in Regelform in das Expertensystem einzugeben und zu ändern. Die *Interviewerkomponente* führt den Dialog mit dem Benutzer und liest automatisch erhobene Messdaten ein. Zur Darstellung der Vorgehensweise des Expertensystems sowie zur Ausgabe von Ergebnissen der Inferenz dient die *Erklärungskomponente*.

Die Wissensbasis enthält *fallspezifisches Wissen* (z.B. erhobene Messdaten) sowie *(Zwischen-)Ergebnisse der Problemlösung*, die die Faktenbasis darstellen. Das vom Experten eingegebene *bereichsbezogene Wissen* sind Regeln, die den Gebrauch des Faktenwissens angeben (*Ableitungsregeln*) und die Ableitung selber steuern (Steuerregeln).

Zur formalen Beschreibung von Expertenwissen in Regelform hat sich neben Prolog vor allem die Sprache Lisp durchgesetzt (GOTTLOB et al. 1990; KURBEL 1992). Prolog wurde 1973 von Colmerauer entwickelt und ist eine logische Sprache, die auf dem Lambda-Kalkül basiert und die Formulierung von Anfragen über eine Wissensbasis erlaubt (COLMERAUER et al. 1973). Lisp (1960 von McCarthy erstmals vorgestellt) ermöglicht die Beschreibung von Expertenwissen in Form von Funktionen (MCCARTHY 1960). Der Fokus imperativer Sprachen (wie C oder Java) liegt in der Angabe von Berechnungsvorschriften, so dass sie sich zwar zur Durchführung arithmetischer Berechnungen, aber nicht zur Formalisierung von Expertenwissen eignen (NEBENDAHL 1987). Darüber hinaus haben Untersuchungen gezeigt,

dass die Programmierung von Regeln und Anweisungen in Lisp signifikant schneller gelernt und durchgeführt werden kann als in C, C++ oder Java (GAT 2000).

5.1.1 Aufbau des Werkzeugs REVISER

Ausgehend von dem allgemeinen Aufbau eines Expertensystems ist der grundlegende Aufbau des Werkzeugs REVISER in Abbildung 5.2 dargestellt. Das Werkzeug dient als Unterstützung eines Experten bei der Bewertung interaktiver Systeme. Aus diesem Grund ist eine Trennung zwischen Experte und Benutzer nicht sinnvoll. Ein solches System wird auch *eingebettetes* Expertensystem genannt (PUPPE 1991). Die Basis der Bewertung bildet die formale GUI-Beschreibung[2] eines interaktiven Systems, die als *fallspezifisches Wissen* in der Wissensbasis zur Verfügung steht. Das *bereichsbezogene Expertenwissen* besteht in REVISER aus den Bewertungsregeln, die der Experte in die Wissensbasis ablegt. Die Steuerung der Inferenz wird durch Steuerregeln übernommen, die in einer eigenen Wissensbasis abgelegt sind. Ein System, das eine Fakten- und eine Regelbasis sowie einen Regelinterpreter (Inferenzalgorithmus) und ein Kontrollsystem (Ablaufsteuerung) enthält, wird als „Produktionssystem“ bezeichnet.

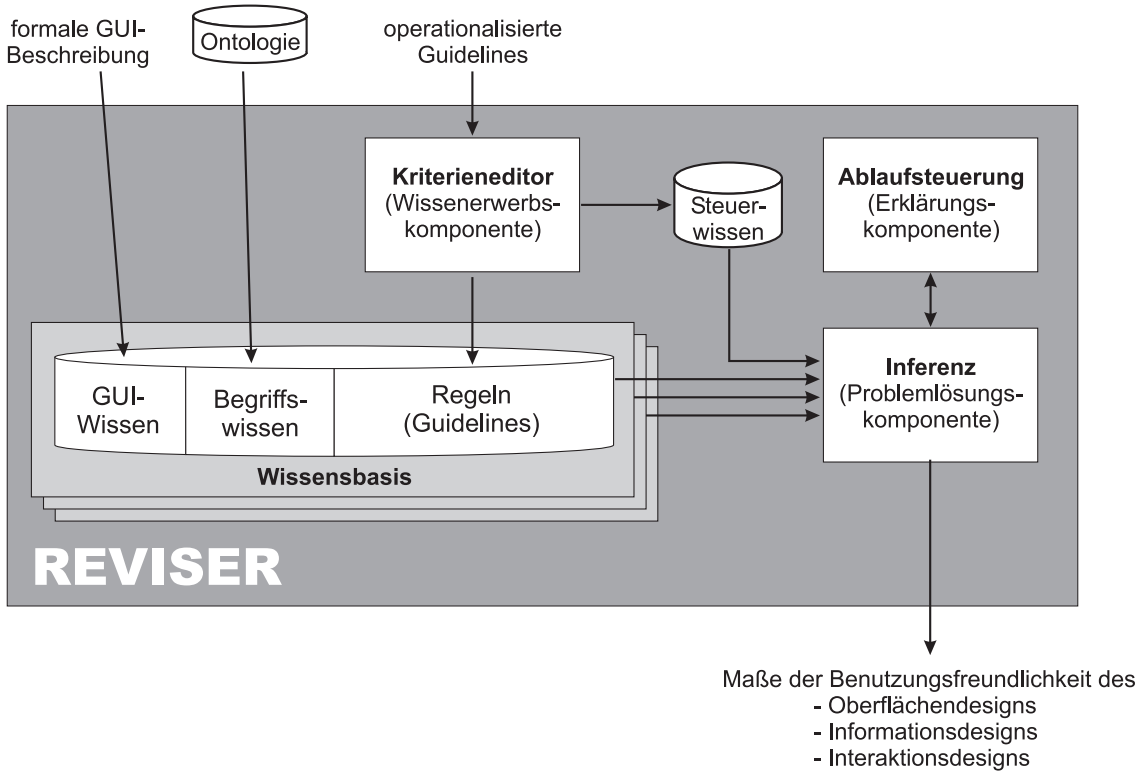

Abbildung 5.2: Systemarchitektur zur kriterienorientierten Bewertung interaktiver Systeme. Der dunkelgrau hinterlegte Bereich kennzeichnet das Werkzeug REVISER.

[2]Die GUI-Beschreibung muss in dem in Kapitel 3.1 vorgestellten Format vorliegen. Dieses Format zur Beschreibung von GUIs ist allgemein gültig. Aus diesem Grund ist eine Konvertierung einer GUI-Beschreibung aus anderen Formate in das in dieser Arbeit entwickelte Beschreibungsformat problemlos möglich und daher nicht weiter berücksichtigt.

Mit Hilfe des *Kriterieneditors* werden in REVISER die operationalisierten Guidelines in Form von Regeln implementiert. Die Basis der Bewertung bildet die formale GUI-Beschreibung eines interaktiven Systems. Zur Bewertung des Informationsdesigns sowie zur Menü-Navigation im Rahmen der agentenbasierten Bewertung des Interaktionsdesigns bildet eine Begriffsontologie die Basis für das benötigte Begriffswissen.

Die Berechnung der meisten Bewertungsmaße erfolgt in mehreren Schritten (vgl. Abschnitt 3.4). Beispielsweise werden zur Berechnung der Gridedness[3] im ersten Schritt die Menge aller Kanten (oben, unten, links, rechts) bestimmt, bevor im zweiten Schritt die Gridedness berechnet werden kann. Da funktionale Sprachen (im Gegensatz zu imperativen Sprachen) keine Angabe der Ausführungsreihenfolge von Regeln erlauben, lassen sich in REVISER Regeln in unterschiedlichen Wissensbasen gruppieren. Die Steuerung der Bearbeitungsreihenfolge dieser Wissensbasen bei der Inferenz lässt sich durch Steuerregeln angeben.

Üblicherweise werden die Regeln durch den Inferenzalgorithmus einmal auf die Fakten einer Wissensbasis angewendet. Zur Erzeugung einer Dialogfolge zur Bewertung des Interaktionsdesigns kommt ein Bedienagent zum Einsatz (vgl. Abschnitt 4), der zur Analyse der Dialogfolge und Aktionenfindung nach jedem Bedienschritt die gleichen Verarbeitungsstufen durchläuft. Ein solcher Dialogschritt entspricht einem in Expertensystemen üblichen *Match-Execute-Zyklus*. Dabei wird im Rahmen des *Match*-Schritts der Zustand des interaktiven Systems aufgrund der GUI-Beschreibung des Dialogs erkannt, der Folgezustand anhand der Ontologie und Angabe des Aufgabenziels identifiziert und ein Bedienelement zur Durchführung der benötigten Aktion aufgrund der Stereotypen festgelegt. Der *Execute*-Schritt besteht in der Ausführung der identifizierten Aktion. Das Prinzip ist in Abbildung 5.3 dargestellt. Jeder *Match*- bzw. *Execute*-Schritt kann selber wieder durch mehrere Inferenzen realisiert sein.

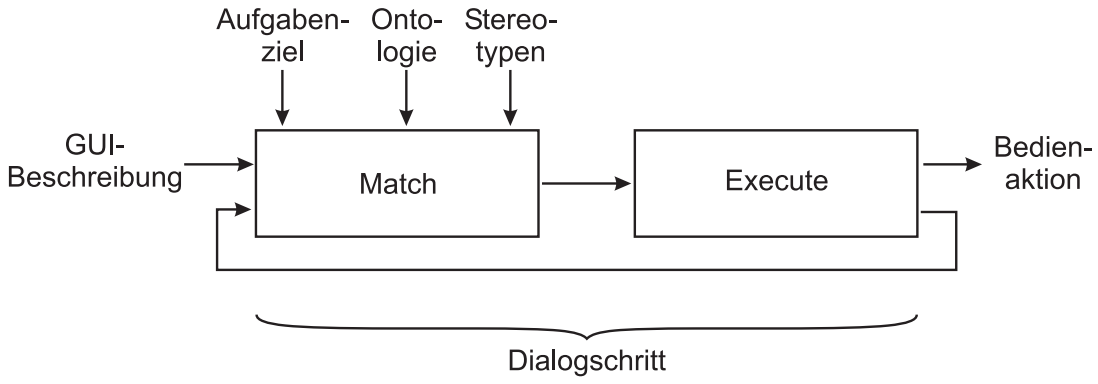

Abbildung 5.3: Implementierung eines Dialogschritts des Bedienagenten als Match-Execute-Zyklus.

Zur Generierung von Dialogfolgen durch den Agenten sind mehrere Dialogschritte in Form von *Match-Execute-Zyklen* nötig. Ein solcher Zyklus ist in REVISER in mehreren Wissensbasen implementiert, deren Inferenzreihenfolge durch die Steuerregeln angegeben ist. Die dazu

[3]Die Gridedness ist ein Maß für die *Angeordnetheit* der GUI-Objekte und ist in Abschnitt 3.4.2.1 detailliert vorgestellt.

benötigte Funktionalität des interaktiven Bedienagents aus Kapitel 4 ist in REVISER als Projekt realisiert. Daher beschränkt sich die Bedienung auf das Laden und Starten des Agenten und ist nicht weiter ausgeführt.

5.1.2 Realisierung des Werkzeugs REVISER

Zur Entwicklung von Expertensystemen existieren heutzutage einige frei verfügbare Programmbibliotheken, die zur Angabe von Regeln eine Regelsprache sowie einen Inferenzalgorithmus zur Abarbeitung von Regeln mit einer Wissensbasis enthalten. Zu den gängigsten Programmbibliotheken gehören CLIPS[4] und JESS[5], die beide die funktionale Programmiersprache Lisp zur Implementierung der Regeln verwenden.

Das in dieser Arbeit entwickelte Expertensystem REVISER basiert auf JESS, das für die Inferenz den RETE-Algorithmus[6] benutzt und in Java implementiert ist. Da die weitere Funktionalität von REVISER ebenfalls in Java implementiert wurde, kann das Werkzeug dadurch eine weitestgehende Plattformunabhängigkeit gewährleisten.

Die Hauptansicht von REVISER ist in Abbildung 5.4 dargestellt[7]. Die Angabe der Daten der einzelnen Wissensbasen, die in *Projekten* verwaltet werden, erfolgt im Kriterieneditor. Der Zugriff auf die einzelnen Wissensbasen sowie der Lade- und Speicherfunktionalität von Projekten ist im linken Bereich der Hauptansicht von REVISER dargestellt. Dabei lassen sich die Wissensbasen in einem Baum frei anordnen. Die Anordnung bezieht sich jedoch ausschließlich auf die Darstellung im Projektbaum und hat keinen Einfluss auf die Ausführungsreihenfolge. Diese Reihenfolge wird im Bereich *Routing* angegeben und im weiteren Verlauf dieses Kapitels vorgestellt.

Die Wissensbasen sind mit einem Namen im Baum bezeichnet. Hinter dem Namen stehen in eckigen Klammern die Namen der Wissensbasis, die in der Ausführungsreihenfolge der jeweiligen Wissensbasis folgen (im Beispiel folgt „DataOutput“auf die angewählte Wissensbasis „DeltaE“). Es existieren unter dem Projektbaum Buttons zum Verschieben der Basen im Baum, sowie zum Löschen und Erstellen.

Rechts neben dem Projektbaum erfolgt die Darstellung der aktuell im Projektbaum angewählten Wissensbasis („DeltaE“ im Beispiel). In der Mitte der Hauptansicht sind die in der Wissensbasis definierten Regeln und rechts daneben die Faktendefinitionen dargestellt. Die Bedienung dieser Komponenten ist im nachfolgenden Abschnitt 5.2 erläutert.

[4] Das *C Language Integrated Production System (CLIPS)* ist ein weit verbreitetes Expertensystem, in dem Regeln in Lisp implementiert werden und das die Integration von C-Code erlaubt. Weitere Informationen finden sich unter http://www.ghg.net/clips/CLIPS.html

[5] Die *Java Expert System Shell (JESS)* ist ein in Java implementiertes Expertensystem, das im Aufbau und der Bedienung dem Expertensystem CLIPS entspricht. Die offizielle Webseite ist unter folgendem Link zu finden: http://herzberg.ca.sandia.gov/jess

[6] Der RETE-Algorithmus ist ausführlich in Abschnitt 5.3.1 sowie im Glossar vorgestellt.

[7] Die in dieser Arbeit entwickelte Bewertungsmöglichkeit ist in dem Werkzeug REVISER erstmalig umgesetzt. Bei der Realisierung lag der Fokus daher primär in der korrekten Umsetzung der geforderten Funktionalität und erst sekundär in der Entwicklung eines gefälligen Dialogdesigns.

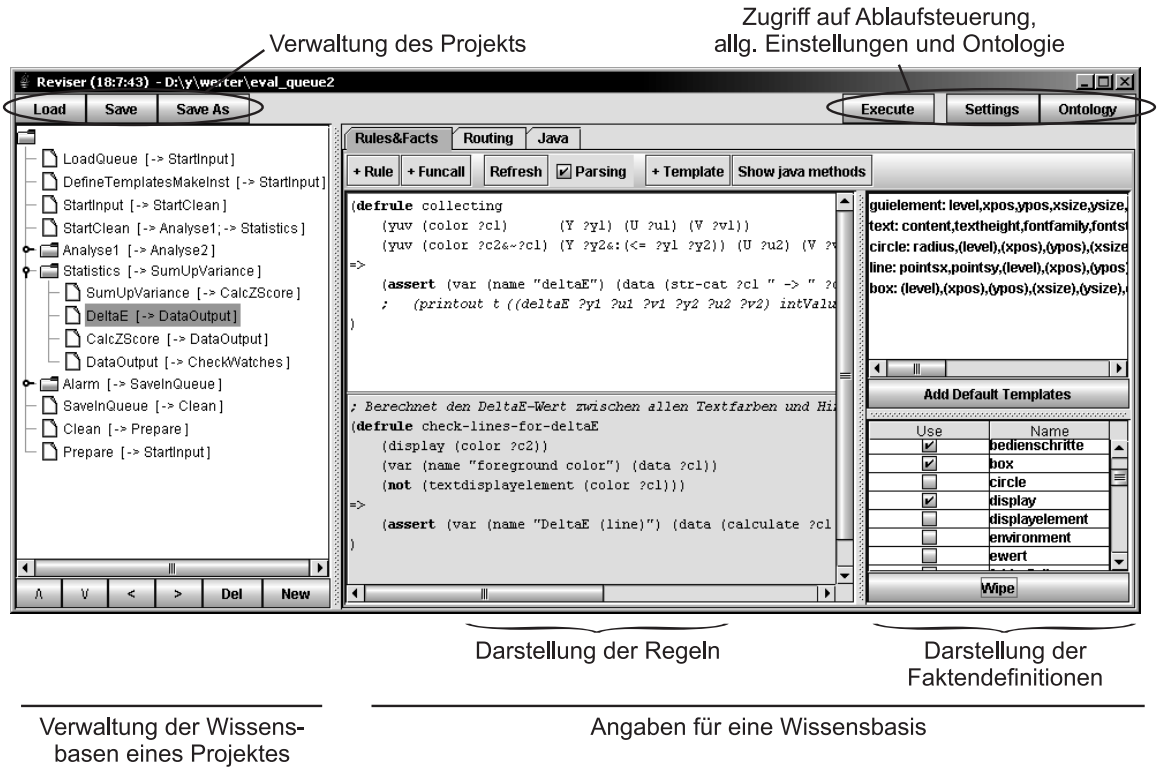

Abbildung 5.4: Hauptansicht von REVISER. Im linken Bereich erfolgt die Verwaltung der Wissensbasen. Der Zugriff auf die Regeln einer Wissensbasis ist im mittleren Bereich gegeben, die Verwaltung der Faktendefinitionen im rechten Bereich der Hauptansicht. Der Import einer Ontologie sowie der Aufruf der Ausführungskomponente lassen sich durch entsprechende Buttons oben rechts aktivieren.

Die Buttons für den Zugriff auf allgemeine Einstellungen, die Ontologie sowie die Ausführungskomponente sind im Dialog in der oberen rechten Ecke zu finden. Zum Import einer Ontologie ist die Angabe einer .owl-Datei erforderlich (s. Anhang C). Nach dem erfolgreichen Laden der Ontologie wird sie in einem eigenen Dialog in Baumform dargestellt. Beispiele der Darstellung einer Ontologie in REVISER findet sich in Anhang C.

5.2 Implementierung von Regeln zur Überprüfung von Guidelines

Zur Überprüfung des Systemdesigns erlaubt REVISER die Implementierung von Guidelines als Regeln im Kriterieneditor, dessen Zuordnung in REVISER in Abbildung 5.5 dargestellt ist. Um Regeln in REVISER implementieren zu können, müssen die zugehörigen Guidelines operationalisiert sein (vgl. Abschnitt 2.2). Im Folgenden ist eine Übersicht der Funktionalität des Kriterieneditors gegeben (Abschnitt 5.2.1). Anschließend erfolgt die Vorstellung der in REVISER enthaltenen Dialoge zur Verwaltung der Wissensbasen (Abschnitt 5.2.2), zur Regeleingabe (Abschnitt 5.2.3) sowie die Möglichkeit zur Definition von Fakten in Abschnitt 5.2.4.

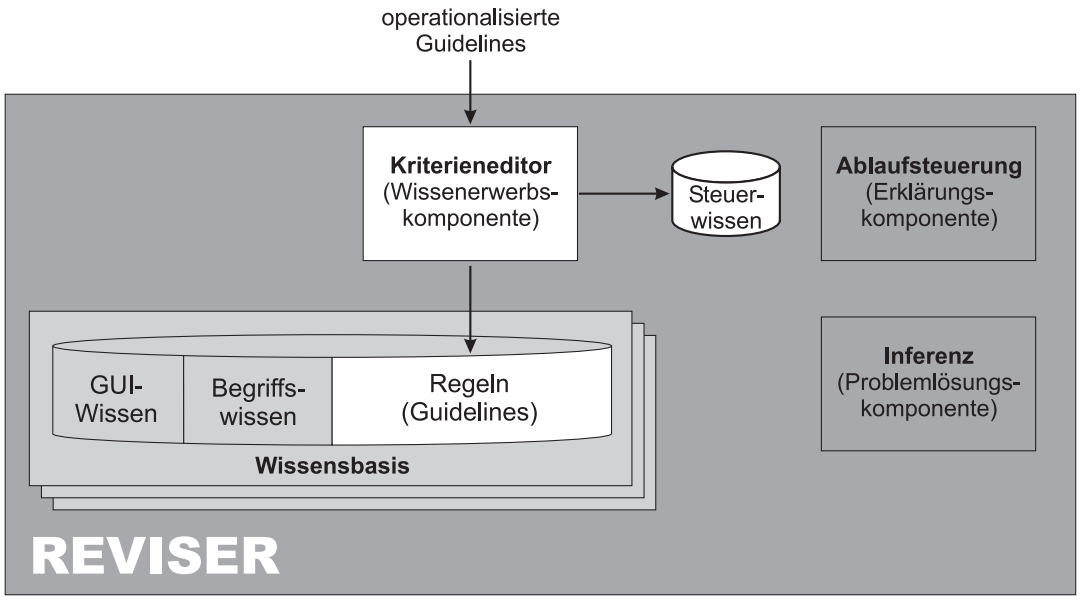

Abbildung 5.5: Schematische Darstellung des *Kriterieneditors* zur Erstellung von Regeln sowie dem Import von Java-Sourcecode im Rahmen des Werkzeugs REVISER.

5.2.1 Übersicht über die Funktionalität des Kriterieneditors

REVISER unterstützt die Regeleingabe in dem in Kapitel 3.2 vorgestellten Format. Dabei haben Regeln grundsätzlich einen Wenn-Dann-Aufbau. Im Wenn-Teil (Regelprämisse) erfolgt eine Abfrage von Fakten des GUI-Formates bzw. Attribute einzelner GUI-Elemente, während im Dann-Teil (Regelkonklusion) Fakten modifiziert, gelöscht, neu erzeugt oder Funktionen aufgerufen werden können. Alle Daten, die der Experte im Kriterieneditor eingibt, werden in REVISER in Form eines *Projekts* gespeichert.

Die Möglichkeiten zur Eingabe von Regeln in Lisp sowie die Erweiterung der Regelsprache im Rahmen dieser Arbeit werden im Folgenden vorgestellt. Die Syntax und die BNF der Regelsprache ist Bestandteil von Anhang A.

Die Eingabe der Regeln kann sowohl im Freitext als auch dialoggestützt erfolgen. Die Freitexteingabe erlaubt die maximale Flexibilität bei der Regeleingabe, bei der alle Möglichkeiten der funktionalen Regelsprache genutzt werden können. Der Eingabedialog unterstützt darüber hinaus die Eingabe der einzelnen Regelkomponenten (Regelname, Kommentare, Bedingungen der Prämisse, Anweisungen der Konklusion) und ermöglicht die Implementierung einer Regel ohne detaillierte Kenntnisse der Regelsprache.

Häufig müssen bei der Formulierung des Expertenwissens auch umfangreiche arithmetische Berechnungen durchgeführt werden (z.B. die Berechnung der ΔE-Werte, vgl. Abschnitt 3.4.2). Während sich zur Regeleingabe die funktionale Sprache Lisp eignet, kommen zur Implementierung dieser Berechnungen überwiegend imperative Programmiersprachen wie C oder Java zum Einsatz. Daher ist die Kombination einer funktionalen mit einer imperativen Sprache zur Implementierung der Regeln sinnvoll. Aufgrund der großen Verbrei-

tung sowie der Plattformunabhängigkeit bietet sich Java als imperative Sprache an. REVISER ermöglicht die Eingabe bzw. den Import von Prozeduren in Java, die in der Regelkonklusion aufgerufen werden können. Das Vorgehen beim Import des Java-Sourcecodes sowie der Registrierung der Java-Prozeduren für den Aufruf in Lisp ist in Anhang B detailliert erklärt.

Zur Steuerung der Inferenz erlaubt REVISER die Angabe der Reihenfolge, in der die Wissensbasen inferiert werden sollen. Dieses Steuerwissen lässt sich ebenfalls im Kriterieneditor angeben und wird in REVISER zentral gespeichert.

5.2.2 Realisierung des Kriterieneditors

Der Zugriff auf die Daten einer Wissensbasis ist in den drei Bereichen *Regeln und Faktendefinition (Rules&Facts)*, *Ausführungsreihenfolge (Routing)* und *Java-Sourcecode (Java)* möglich. Alle drei Bereiche sind in Abbildung 5.6 dargestellt und im Folgenden erläutert.

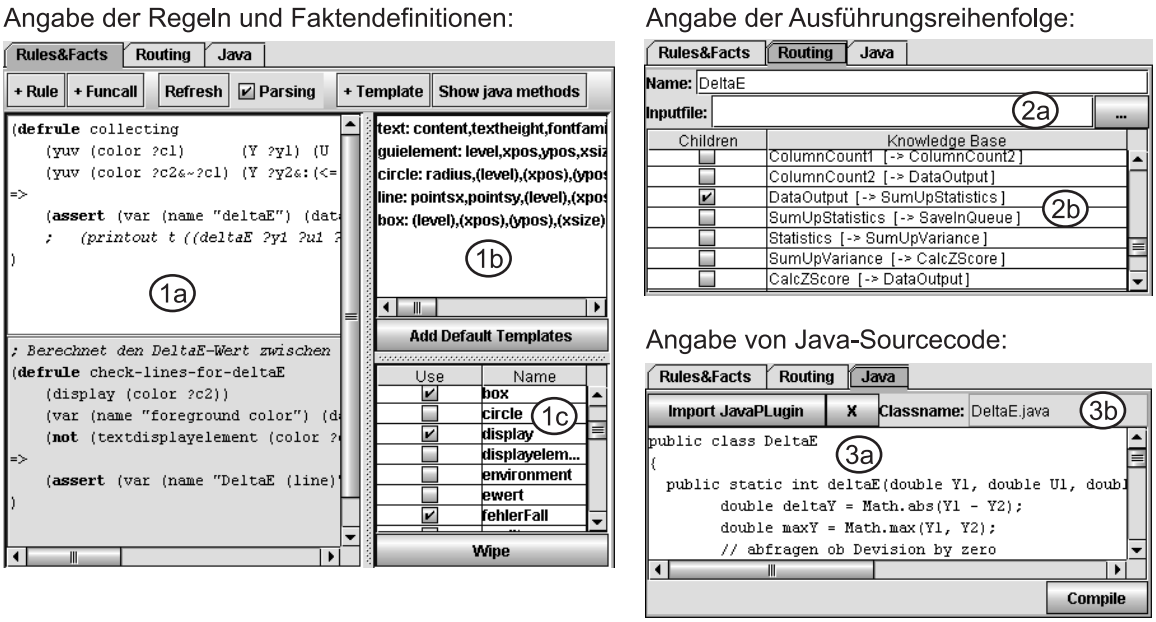

Abbildung 5.6: Darstellung der Informationen einer Wissensbasis in drei Bereichen innerhalb von REVISER. Der erste Bereich (links) dient der Darstellung und Erstellung von Regeln und Faktendefinitionen. Der zweite Bereich (rechts oben) dient der Angabe der Ausführungsreihenfolge, während im dritten Bereich (rechts unten) die Möglichkeit gegeben ist, Java-Sourcecode anzugeben.

Angabe der Regeln und Faktendefinitionen

Der erste Bereich der Datenverwaltung dient der Darstellung und Angabe von Regeln und Faktendefinitionen. Im linken Teil (1a) sind die Regeln aufgelistet, die in der Wissensbasis implementiert sind. Regeln (bzw. Funktionen) lassen sich in Freitext oder dialoggestützt

eingeben. Ist das „Parsing“ aktiviert (oben mittig liegende Checkbox), so erfolgt die Eingabe dialoggestützt, wobei neue Regeln (bzw. Funktionen) durch den Button „+Rule“ (bzw. „+Funcall“) eingegeben und Änderungen bestehender Regeln durch Doppelklick auf die entsprechende Regel im Regelfenster durchgeführt werden können. Der Dialog zur Regeleingabe ist in Abschnitt 5.2.3 vorgestellt. Aktive Regeln sind mit einem weißen, inaktive (also „auskommentierte“ Regeln) mit einem grauen Hintergrund dargestellt. Ist „Parsing“ deaktiviert, dann wandelt sich der Bereich der Regeln (1a) zu einem Editorfenster, in dem Regeln in Freitext eingegeben werden können. Die Java-Methoden, die als Funktionen der Regelsprache registriert sind (vgl. Abschnitt B), lassen sich durch den Button „Show java methods“ anzeigen.

Da Fakten erst zur Laufzeit, z.B. durch den Import einer formalen GUI-Beschreibung, erzeugt werden, erfolgt im Kriterieneditor die Definition, welche Fakten in der aktuellen Wissensbasis zur Verfügung stehen sollen. Dabei lassen sich neue Fakten erzeugen (1b) bzw. die Fakten der in der Ausführungsreihenfolge vorhergehenden Wissensbasis importieren (1c). Der Dialog zur Angabe neuer Fakten ist in Abschnitt 5.2.4 vorgestellt.

Angabe der Ausführungsreihenfolge

Zur Angabe der Ausführungsreihenfolge sind alle im Projektbaum angegebenen Wissensbasen aufgelistet (s. Abb. 5.6 rechts oben – 2b). Alle Basen, die direkt nach der aktuellen Wissensbasis ausgeführt werden sollen, lassen sich entsprechend markieren. Im Beispiel ist die Wissensbasis „DataOutput“ als Nachfolger gewählt (entsprechend der Darstellung im Projektbaum sind die Nachfolger der Basen ebenfalls in Klammern angegeben).

Weiterhin ist es in diesem Bereich möglich, die Quelle einer formalen GUI-Beschreibung anzugeben (2a). Dabei werden beim Faktenimport externe Quellen von Fakten (z.B. Datei) genauso behandelt wie interne (Wissensbasen). Es erfolgt bei der Ausführung lediglich der Import derjenigen Fakten, die (im Bereich 1b bzw. 1c) explizit für einen Import vorgesehen sind. Alle weiteren Fakten werden ignoriert.

Angabe von Java-Sourcecode

Java-Sourcecode lässt sich in einem Editor eingeben (s. Abb. 5.6 rechts unten – 3a). Der Sourcecode wird automatisch unter dem Namen der Wissensbasis gespeichert, durch Anwahl des Buttons „Compile“ in Bytecode übersetzt und die in Java genutzten Methodennamen als Funktionen[8] der Regelsprache zur Verfügung gestellt (vgl. Abschnitt B). Im Beispiel ist die Methode „deltaE“ dargestellt, die problemlos als Funktion registriert werden kann, da sie *public* und *static* ist.

[8] Lisp erlaubt als logische Sprache keinen Zugriff auf die Objekte einer objektorientierte Sprache. Dennoch ist der Zugriff auf große objektorientiert Java-Bibliotheken durch den Einsatz einer *Wrapper*-Klasse möglich, die den Zugriff realisiert und in REVISER implementiert werden kann.

Bereits übersetzter Bytecode (abgelegt in .class-Dateien) lässt sich ebenfalls einlesen und umwandeln. Die Angabe der .class-Datei erfolgt im Textfeld (3b). Es lässt sich pro Wissensbasis eine Klasse[9] (mit beliebig vielen Methoden) verwalten. Diese Klasse kann entweder eingeladen oder im Editor programmiert werden, eine Kombination von beidem ist nicht möglich.

5.2.3 Dialog zur Eingabe von Regeln

Der Dialog zur Regeleingabe bietet Eingabefelder für die unterschiedlichen Komponenten einer Regel und ist in Abbildung 5.7 dargestellt. Zur Definition einer Regel erlaubt der Dialog die Angabe eines Namens und eines Kommentars. Ein vollständiges „Auskommentieren“ von Regeln ist durch Anwahl der Checkbox „disable“ möglich.

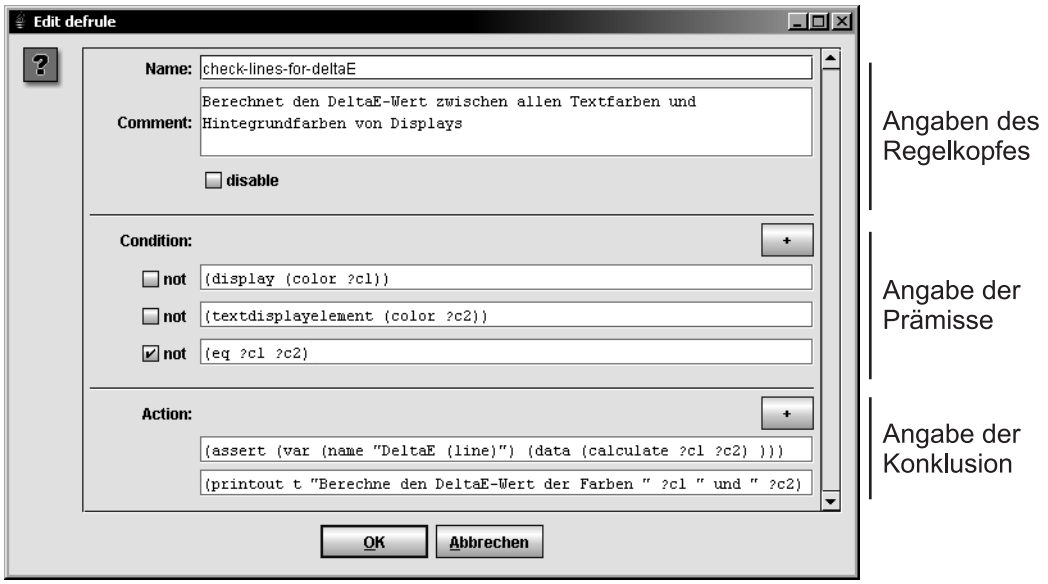

Abbildung 5.7: Dialog zur Eingabe der Regel. Für jede Bedingung der Prämisse und Anweisung der Konklusion existiert jeweils ein Eingabefeld.

Innerhalb der Prämisse lässt sich jede Bedingung (Condition) in einem eigenen Feld eingeben. Alle Bedingungen werden implizit „und“-verknüpft. Bedingungen, die *nicht* eintreten dürfen, können durch die Aktivierung der Checkbox „not“ negiert werden. „Oder“-Verknüpfungen müssen explizit im Eingabefeld angegeben werden. Die Anwahl des Button „+“ erzeugt ein neues Feld für eine weitere Bedingung. Soll ein Feld gelöscht werden, so genügt das Löschen des Feldinhalts. REVISER parst die Regeln automatisch und löscht daher leere Felder selbstständig. Für jede Aktion der Konklusion ist ebenfalls ein eigenes Feld vorgesehen.

Beispielhaft ist in Abbildung 5.7 eine Regel zur Berechnung der ΔE-Werte[10] der Farben von Texten und des dahinter liegenden Displays implementiert. Dazu werden alle Kombinationen

[9] In Java beinhaltet eine .class-Datei den Sourcecode genau einer Klasse.

[10] Ein ΔE-Wert gibt den spektralen Farbabstand zweier Farben an. Die Berechnung ist in Abschnitt 3.4.2 vorgestellt.

der Farben von Displays (erste Bedingung) und Textelementen (zweite Bedingungen) miteinander verglichen, solange diese Farben nicht ohnehin gleich sind (dritte Bedingung). Die ΔE-Werte der jeweiligen Farben werden ausgerechnet und das Ergebnis als Fakt in die Faktenbasis gespeichert (erste Aktion). Darüber hinaus erfolgt eine Ausgabe auf den Bildschirm (zweite Aktion).

5.2.4 Dialog zur Eingabe von Faktendefinitionen

Vor der Benutzung von Fakten muss deren Struktur definiert sein. Das dient der Angabe derjenigen Fakten, die in eine Wissensbasis übernommen werden sollen (s.o.). Eine Definition der Faktenstruktur – im Folgenden *Template* genannt – beinhaltet den Namen sowie eine Anzahl von *Slots* zur Beschreibung von Attributen eines Fakts. Ein Slot wird durch einen *Slotnamen* eindeutig benannt und enthält einen Wert.

Die Definition der Templates erfolgt in REVISER im *Templateeditor*, der in Abbildung 5.8 dargestellt ist. Fakten können objektorientiert aufgebaut sein und somit Slots von anderen Fakten erben, wobei das Prinzip der Einfachvererbung gilt. Die Angabe eines *Vaters*, dessen Slots geerbt werden sollen, ist ebenfalls im Templateeditor möglich. Weiterhin lassen sich bei der Angabe von Slots Vorgabewerte (defaults) definieren.

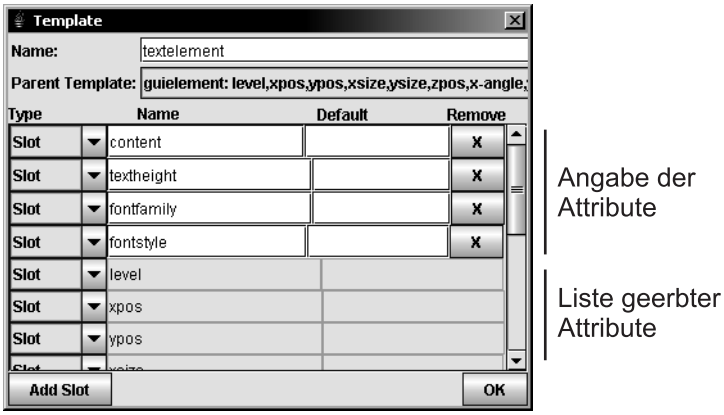

Abbildung 5.8: Dialog zur Definition von Fakten (Templates). Fakten haben einen Namen sowie *Slots* zur Angabe der Attribute. Slots können von anderen Templates geerbt werden. Darüber hinaus ist die Angabe von Vorgabewerten für Slots möglich.

Beispielhaft ist in Abbildung 5.8 das Fakt zur Beschreibung eines Textelements des in dieser Arbeit entwickelten formalen GUI-Beschreibungsformat dargestellt (s. Abschnitt 3.1). „textelement“ erbt alle Slots (Attribute) von „guielement“ und definiert zusätzlich die vier Slots „content“, „textheight“, „fontfamily“ und „fontstyle“, jeweils ohne Vorgabewerte.

5.3 Durchführung und Steuerung einer Bewertung der Gebrauchstauglichkeit

Zur Durchführung und Steuerung einer Bewertung ist innerhalb von REVISER eine Problemlösungskomponente (Inferenz) integriert, um die Bewertungsregeln auf die Fakten anzuwenden. Die an der Regelausführung beteiligten Komponente von REVISER sind in Abbildung 5.9 dargestellt. Die Realisierung der Inferenz sowie die Steuerung der Ausführungsreihenfolge der Wissensbasen ist im Folgenden vorgestellt.

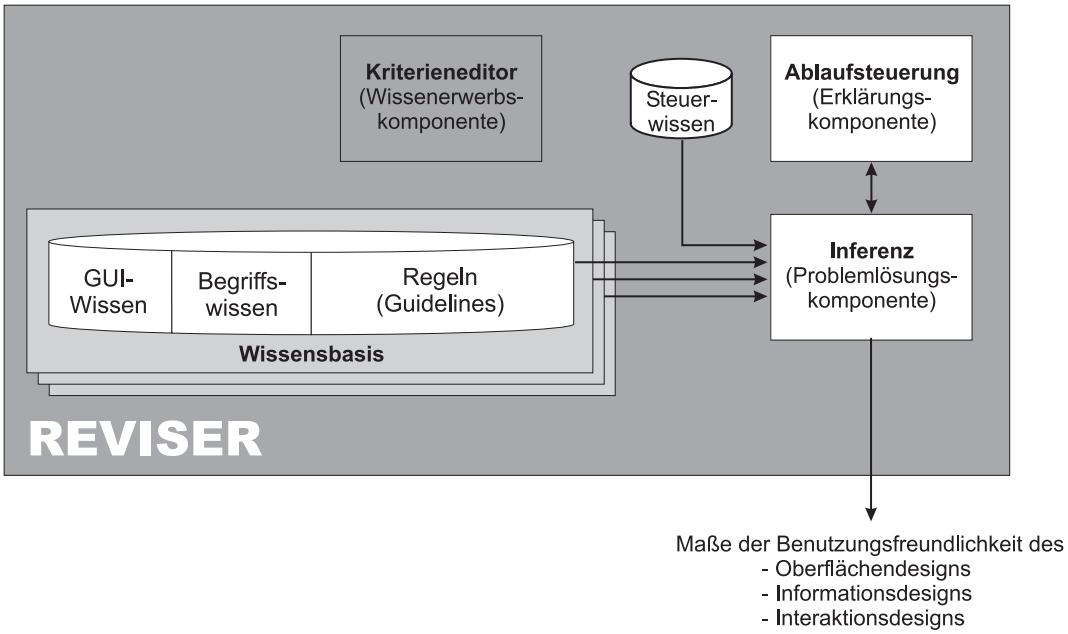

Abbildung 5.9: Die Ablaufsteuerung in REVISER kontrolliert die Bearbeitung der Regeln und Fakten der Wissensbasen durch einen Inferenzalgorithmus sowie die Durchführung einer Bediensimulation durch den Bedienagenten und stellt die Ergebnisse dar.

5.3.1 Inferenz zur Auswertung der Wissensbasis

Die Problemlöse- bzw. Schlussfolgerungskomponente eines Expertensystems wendet mit Hilfe eines Inferenzalgorithmus die Regeln auf die Fakten der Wissensbasis an. Dabei werden die (Zwischen-)Ergebnisse meist ebenfalls in Faktenform in der Wissensbasis gespeichert (vgl. Abbildung 5.1). Die Aufgabe des Inferenzalgorithmus ist die Auswertung von Regeln sowie die Entscheidung der Reihenfolge der Regelauswertung (KURBEL 1992).

Die Abarbeitung der Regeln erfolgt dabei entweder *vorwärtsverkettet* (datengetrieben) oder *rückwärtsverkettet* (zielgetrieben). Bei der Vorwärtsverkettung wird aufgrund bestehender Fakten auf neue Daten geschlossen, mögliche Zielzustände sind nicht bekannt, die Suche ist somit nicht zielgerichtet und daher zum Teil nutzlos. Bei der Rückwärtsverkettung ist das gewünschte Ziel vorgegeben und es sollen Fakten ermittelt werden, die die Regeln zum Er-

reichen des Ziels erfüllen. Falls nicht alle benötigten Fakten vorhanden sind, werden Regeln gesucht, die wiederum die benötigten Fakten erzeugen (RUSSEL und NORVIG 2004).

Die Auswertung der Regeln kann sehr ineffizient sein, da nach jeder Änderung der Wissensbasis (z.B. Ändern oder Hinzufügen eines Fakts) erneut alle Regeln überprüft werden müssen. Die Inferenz lässt sich jedoch in zweierlei Hinsicht optimieren:

1. In der Regelmenge der Wissensbasis existieren Abhängigkeiten zwischen den Prämissen und Konklusionen von Regeln. Ist die Konklusion einer Regel gleich mit einer Prämisse einer anderen Regel, dann lässt sich mit diesen Regeln eine Regelhierarchie bilden (KRAISS 2005). Die Beziehungen aller Regeln einer Regelmenge bilden einen Regelgraphen. Kann bei der Inferenz eine Regel ausgeschlossen werden, so erfolgt ohne weitere Prüfung gleichzeitig der Ausschluss aller Regeln, die die gleiche Bedingung in der Prämisse haben.

2. Bei Änderung der Faktenbasis braucht der Inferenzalgorithmus nicht alle, sondern lediglich die Regeln neu auszuwerten, auf die sich die Änderung auswirkt. Alle anderen Regeln müssen nicht mehr neu ausgewertet werden.

Einer der populärsten Inferenzalgorithmen, der beide Verbesserungen berücksichtigt, ist der RETE-Algorithmus[11], der 1982 von Forgy zum ersten Mal vorgestellt wurde und in vielen Expertensystemen Verwendung findet (FORGY 1982). In der Literatur finden sich zahlreiche Informationen zum RETE-Algorithmus – z.B. (ALBERT 1989; KURBEL 1992; FRIEDMAN-HILL 2003), weshalb an dieser Stelle nicht weiter darauf eingegangen wird.

In dieser Arbeit kommt für die Inferenz (im Rahmen der Programmbibliothek JESS) eine Implementierung des RETE-Algorithmus mit der Möglichkeit der Vorwärts- und Rückwärtsverkettung zum Einsatz. Diese Inferenz wird in der *Ablaufsteuerung* gestartet, die ebenfalls Informationen zur Inferenz sowie die Ergebnisse der Regelauswertung darstellt.

5.3.2 Auswahl der Bearbeitungsreihenfolge der Wissensbasen

Zur Steuerung des Bearbeitung der Wissensbasen ist die Ausführungskomponente verantwortlich, die die Bearbeitung der einzelnen Wissensbasen durch den Inferenzalgorithmus in der jeweils angegebenen Reihenfolge steuert.

Dabei sorgt sie dafür, dass für die Inferenz der einzelnen Wissensbasen die benötigten Fakten aus vorhergehenden Wissensbasen übernommen werden. Da REVISER eine Anordnung der Wissensbasen in Form eines gerichteten Graphen erlaubt, muss die Ausführungskomponente dabei Zyklen erkennen und entsprechend abarbeiten. Dabei muss sie bei der Bearbeitung einer Wissensbasis die Aktualität der Fakten gewährleisten. Die auszuführenden Wissensbasen werden in einer Queue gespeichert. Das Vorgehen ist in Abbildung 5.10 dargestellt und wird im Folgenden beispielhaft anhand der Bearbeitung eines Graphen mit den Wissensbasen *a* bis

[11] Der Name leitet sich aus dem Lateinischen „rete“ für „Netz“ in Bezug zum Regelgraph ab.

e erklärt (s. Abb. 5.11, oben). Die Pfeile des Graphen markieren die Datenflüsse zwischen den jeweiligen Wissensbasen.

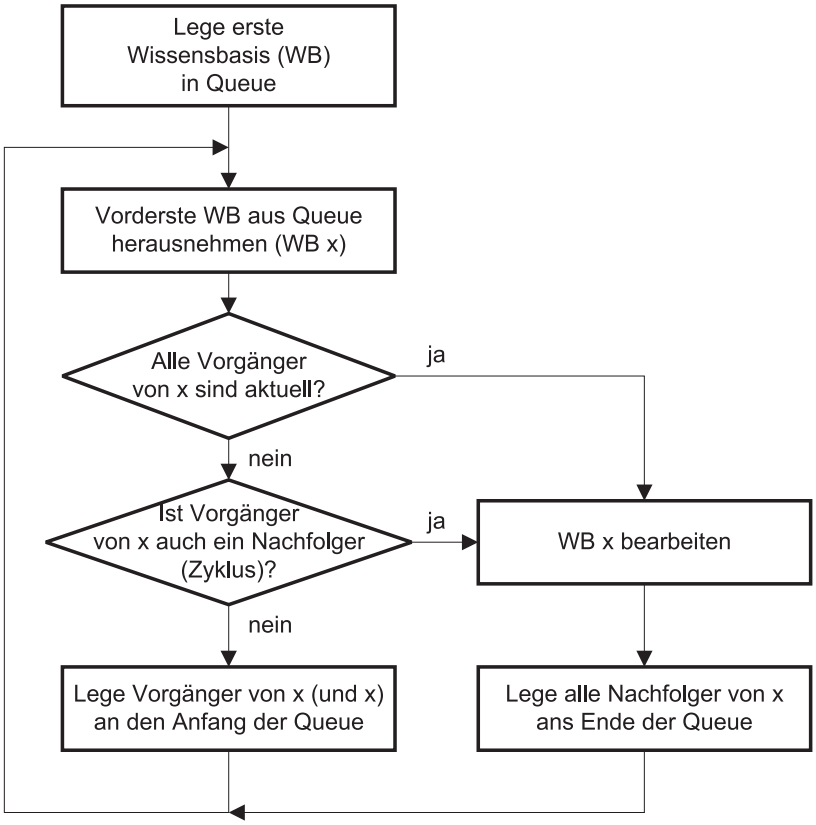

Abbildung 5.10: Vorgehen bei der Abarbeitung eines Wissensbasis-Graphen in REVISER. Dazu müssen Zyklen erkannt und die Aktualität der Fakten bei der Bearbeitung einer Wissensbasis gewährleistet werden.

Die Ausführung soll bei der Wissensbasis (WB) *a* starten. Diese benötigt jedoch Daten aus WB *b*. Bei der ersten Ausführung erkennt die Ausführungskomponente automatisch, dass sich *a* und *b* in einem Zyklus befinden. Aus diesem Grund wird *a* als erste bearbeitet (Schritt 1 (*s*1) in Abb. 5.11), ohne Daten aus *b* zu importieren. Nach der Ausführung von *a* werden die nachfolgenden Wissensbasen *b* und *c* in der Queue gespeichert.

WB *b* benötigt nicht nur Daten aus der bereits bearbeiteten WB *a*, sondern ebenfalls aus WB *c*, die noch nicht bearbeitet ist und daher noch keine Daten liefern kann. Aus diesem Grund werden WB *b* und *c* in der Queue getauscht und in Schritt 2 (*s*2) WB *c* ausgeführt. Anschließend sollte die Ausführungskomponente den Nachfolger von *c* in die Queue legen; allerdings ist *b* bereits vorhanden und die Queue ändert sich demnach nicht.

Im dritten Schritt (*s*3) erfolgt die Ausführung von *b* und das Hinzufügen der Nachfolger *d* und *a* in die Queue. WB *a* kann auf aktuelle Daten von *b* zurückgreifen und wird daher ausgeführt (*s*4), woraufhin die nachfolgenden Wissensbasen von WB *a* (*b* und *c*) in die Queue eingefügt werden.

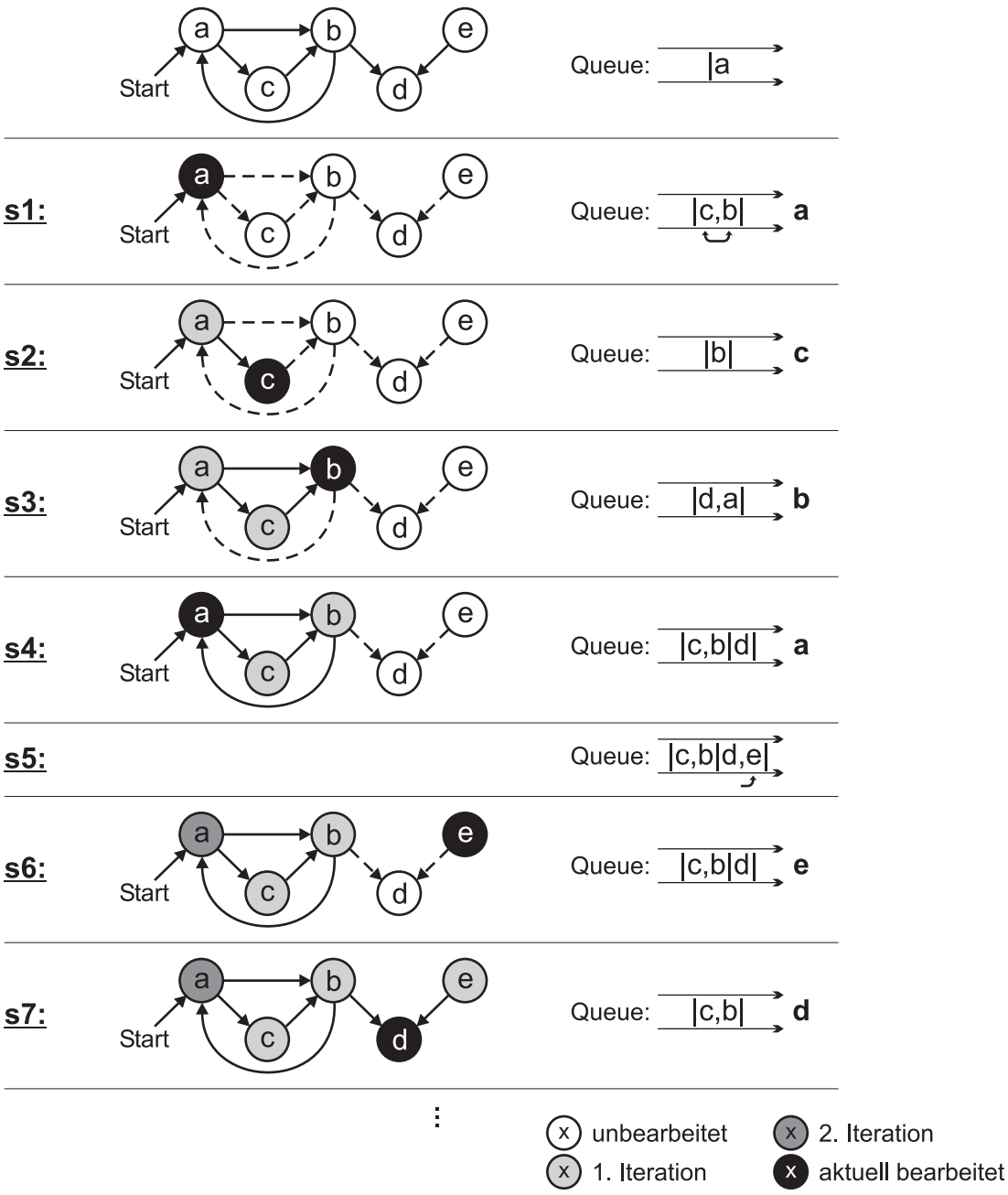

Abbildung 5.11: Beispiel der Auswahl einer Reihenfolge der zu bearbeitenden Wissensbasen. Die Wissensbasen sind als gerichteter Graph dargestellt (oben). Die in der Queue gespeicherten Wissensbasen werden ausgeführt und deren Nachfolger wieder in die Queue gelegt.

In Schritt 5 (*s*5) sollte als Nächstes WB *d* zur Bearbeitung kommen. Diese WB benötigt jedoch Daten aus WB *e*, die noch nicht bearbeitet wurde. Daher wird WB *e* vor *d* in die Queue gelegt und im sechsten Schritt (*s*6) bearbeitet. Anschließend kommt in Schritt 7 (*s*7) WB *d* zur Ausführung.

Dass die zweite Bearbeitung von WB *a* vor der ersten Ausführung von WB *d* erfolgt, ist unbedenklich, da keine direkten Abhängigkeiten zwischen diesen Knoten im Graphen definiert sind und daher das beschriebene Vorgehen eine korrekte Ausführungsreihenfolge erzeugt.

5.3.3 Realisierung der Ablaufsteuerung

Zur Ablaufsteuerung der Wissensbasen in REVISER dient das Modul *Execute*. Der Dialog wird im Hauptfenster von REVISER aufgerufen, ist in Abbildung 5.12 dargestellt und im Folgenden erläutert.

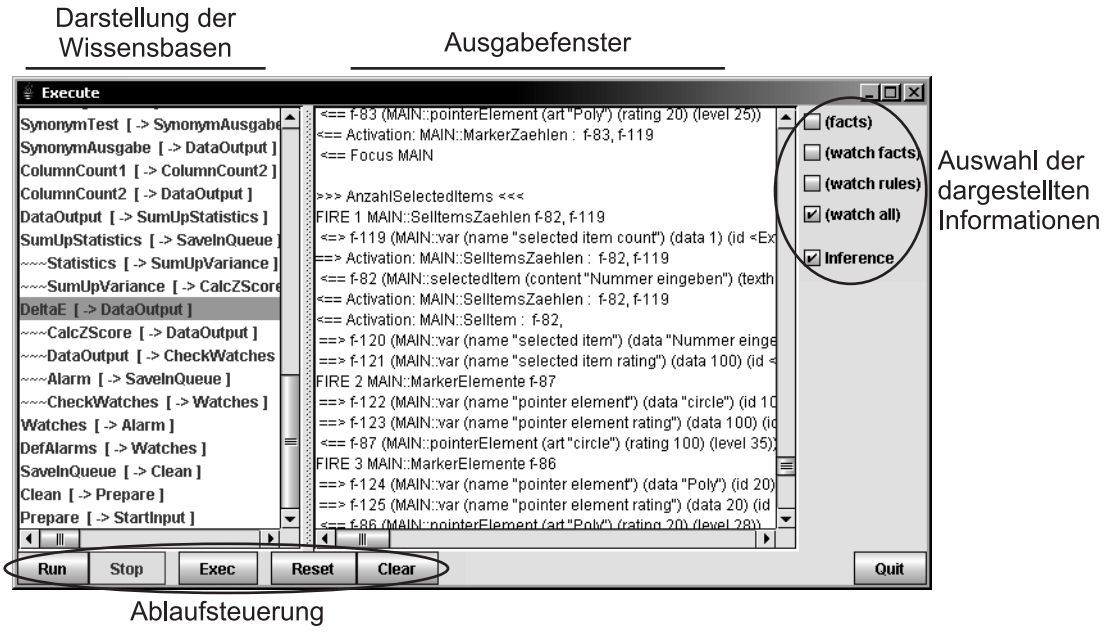

Abbildung 5.12: Dialog zur Ablaufsteuerung in REVISER. Im linken Bereich sind die Wissensbasen dargestellt, während der rechte Bereich der Informationsausgabe dient. Die Art der angezeigten Informationen lässt sich am rechten Rand des Dialogs auswählen. Die Ausführung wird mit den Buttons am unteren Rand gesteuert.

Alle Wissensbasen eines Projektes werden im linken Bereich des Dialogs dargestellt. Die aktuell bearbeitete Wissensbasis ist dabei markiert. Wissensbasen, deren Ausführung verhindert wird (z.B. durch den Befehl „setIgnoreStep“ – s. Abschnitt A.1.2.3), sind mit Tilden (∼∼∼) gekennzeichnet.

Im Ausgabefenster sind Informationen dargestellt, die bei der Regelbearbeitung der aktuellen Wissensbasis ausgegeben werden. Darüber hinaus lassen sich weitere Informationen darstellen, deren Auswahl im rechten Bereich des Dialogs erfolgt. Dabei kann sich der Benutzer bei der Bearbeitung einer Wissensbasis die gesamte Faktenbasis (facts), die Fakten, die sich bei der Regelausführung ändern (watch facts), die Regeln, die gerade ausgeführt werden (watch rules) sowie die gesamten Aktivitäten der Inferenz (watch all) ausgeben lassen. Die Ausgabe allgemeiner Angaben über die Ausführung (z.B. Anzeige der Namen der ausgeführten Wissensbasen) lässt sich mit „Inference“ aktivieren.

Die Ausführung wird mit den Buttons am unteren Dialograndgesteuert. Dabei kann der Benutzer zwischen automatischer und manueller Ausführung wählen. Bei der automatischen Ausführung (kontrolliert mit „Run“ und „Stop“) bearbeitet die Ablaufsteuerung alle Wissensbasen automatisch der Reihenfolge nach. Dabei ergibt sich die Ausführungsreihenfolge nach

dem in Abschnitt 5.3.2 beschriebenen Verfahren. Erwartet eine Wissensbasis eine manuelle Eingabe des Benutzers (z.B. Angabe des Dateinames einer formalen GUI-Beschreibung), dann wird die Ausführung sebstständig gestoppt. Die automatische Ausführung eignet sich z.B. zur mehrfachen Durchführung einer Bewertung bzw. zur Ausführung des Bedienagenten.

Bei der manuellen Ausführung wird die Bearbeitung einer Wissensbasis gemäß der Ausführungsreihenfolge einzeln durch den Button „Exec“ gestartet. Dadurch kann der Benutzer die Bearbeitung schrittweise durchführen und hat eine bessere Übersicht über die dargestellten Informationen. Somit lassen sich die Inferenz bzw. die eingesetzten Fakten der Faktenbasis besser kontrollieren.

Der Button „Clear“ löscht alle Informationen im Bildschirm, während „Reset“ die Ausführung neu startet. Dabei wird der gesamte Faktenspeicher gelöscht und die Ausführungsreihenfolge neu berechnet.

5.3.4 Darstellung und Speicherung von Bewertungsergebnissen

Die Ergebnisse der Inferenz sowie Informationen über die Ausführungsreihenfolge der Wissensbasen werden ebenfalls in der Ablaufsteuerung dargestellt (vgl. Abb. 5.12). Darüber hinaus lassen sich sowohl einzelne Fakten als auch die gesamte Faktenbasis einer Wissensbasis in einem Datenspeicher ablegen. Dieser Datenspeicher kann zur Laufzeit gespeichert bzw. geladen werden und gewährleistet den Datenaustausch zwischen mehreren Projekten. So können (Teil-)Ergebnisse einer Bewertung in anderen Projekten weiterverarbeitet bzw. Ergebnisse aus unterschiedlichen Projekten (z.B. Bewertung mehrerer FIS) miteinander verglichen werden.

Die Darstellung der Ergebnisse einer Bewertung (bzw. einer Inferenz) erfolgt textuell in einem Fenster der Ablaufsteuerung bzw. im Datenspeicher. Diese Ergebnisse lassen sich sich in Programmen wie „Microsoft Excel“ oder „OpenOffice Calc“[12] geeignet visualisieren. Darüber hinaus ist es durch die Einbindung von Java-Sourcecode bzw. -Bibliotheken möglich, geeignete Darstellungsklassen[13] in Java zu implementieren, in REVISER einzubinden und in einer Regel zu verwenden.

[12] http://de.openoffice.org/product/calc.html

[13] Eine mächtige freie Klassenbibliothek zur Darstellung großer Datenmengen mit unterschiedlichen Diagrammtypen ist *JFreeChart*: http://www.jfree.org/jfreechart/

Kapitel 6

Automatische Bewertung von Fahrerinformationssystemen

Die Funktionsweise des in dieser Arbeit entwickelten Werkzeugs REVISER wird anhand einer Bewertung zweier FIS aufgezeigt. Bei den untersuchten FIS handelt es sich um das Audi MMI und das BMW iDrive. Diese Systeme sind in Kapitel 6.1 vorgestellt. In Kapitel 6.2 erfolgt ein Vergleich der Ergebnisse einer automatischen, kriterienorientierten Bewertung durch REVISER mit Ergebnissen zweier empirischer Versuche. Dabei handelt es sich um eine im Rahmen dieser Arbeit durchgeführten empirische Bewertung der beiden FIS sowie um eine Bewertung der Firma SirValUse, deren Ergebnisse zum Vergleich vorlagen. Die Details der automatischen Bewertung durch REVISER sind in Kapitel 6.3 erläutert, während die Durchführung und Ergebnisse der im Rahmen dieser Arbeit durchgeführten empirischen Versuche detailliert in Kapitel 6.4 beschrieben wird.

6.1 Fahrerinformationssysteme Audi MMI und BMW iDrive als Anwendungsbeispiel

Fahrerinformationssysteme (FIS) bieten den Zugriff auf alle Komfortfunktionen eines KFZ und bestehen üblicherweise aus einem in der Mittelkonsole zentral angebrachten Display mit entsprechenden Bedienelementen. Ein zusätzliches Display im Instrumententräger[1] sowie Bedienelemente auf dem Lenkrad ergänzen häufig diese FIS[2] (BISCHOFF et al. 2002).

Nahezu alle aktuellen KFZ der Oberklasse verfügen heutzutage über ein FIS, wobei sie sich in der Platzierung der Bedienelemente und Displays sowie der Bedienung und Präsentation der Informationen auf den Displays signifikant unterscheiden. Ein Überblick über die aktuellen FIS sowie deren Eigenschaften ist in (HAMACHER und HÄHNEL 2002) gegeben.

[1] Bei dem *Instrumententräger* handelt es sich um den Bereich hinter dem Lenkrad, in dem fahrrelevante Informationen wie Geschwindigkeit und Drehzahl angezeigt werden.

[2] FIS wurden ausführlich in Kapitel 1.1.1 vorgestellt,

In diesem Abschnitt werden die FIS Audi MMI und BMW iDrive vorgestellt. Die im Rahmen dieser Arbeit entstandene FIS-Ontologie zur Bewertung des Informationsdesigns und zur Zielfindung des autonomen Bedienagenten ist im Anschluss erklärt. Hiernach erfolgt die Definition geeigneter Aufgaben zur Bewertung des Systemdesigns der FIS.

6.1.1 Beschreibung der zu bewertenden FIS

Beide FIS besitzen einen Monitor in der Mittelkonsole im blickgünstigen (peripheren) Sichtbereich. Die Bedienelemente beider FIS sind an der Armlehne, im optimalen Greifbereich, angeordnet. Da beide Systeme das zentrale Display sowie die Bedienelemente an der gleichen Stelle haben, eignen sie sich besonders für eine vergleichende Bewertung des Oberflächen- und Informationsdesigns der auf den Displays dargestellten Informationen sowie des Interaktionsdesigns.

Da beide Systeme nicht im Original zur Verfügung standen, wurden sie im Rahmen dieser Arbeit nachgebildet. Dazu kam das Prototyping-Werkzeug STATESHADE[3] zum Einsatz. Dieses Werkzeug erlaubt die textuelle Spezifikation der Dialogkontrolle durch Zustands-Übergangs-Diagramme, wobei Zustände hierarchisch angeordnet sein können. Darüber hinaus ist die Angabe des GUI für jeden Zustand in einem grafischen GUI-Editor möglich. STATESHADE ermöglicht die prototypische Ausführung des Systems, wobei ebenfalls eine Schnittstelle zum Datenaustausch (formale GUI-Beschreibung sowie Aktionen) mit REVISER zur Laufzeit integriert ist.

Zur Durchführung einer empirischen Bewertung zur Evaluierung der Ergebnisse von REVISER stand eine Sitzkiste[4] zur Verfügung. Die Bedienelemente der Systeme wurden zusammen mit einem entsprechendem Display an die jeweiligen Positionen eingebaut. Somit boten sich die Systeme den Probanden nahezu originalgetreu dar.

Die Eigenschaften der simulierten Systeme sowie der Vergleich zu den jeweiligen Originalsystemen ist im Folgenden erklärt.

Audi MMI

Das im aktuellen Audi A8 eingebaute FIS *Audi MMI* ist im Vergleich zu dem in dieser Arbeit erstellten Versuchsaufbau in Abbildung 6.1 dargestellt.

Für den Versuchsaufbau stand ein originales Bedienpanel zur Verfügung. Die Bedienung erfolgt über Hard- und Softkeys sowie einen Dreh-Drück-Knopf; eine Zuordnung der Tasten zu den Inhalten des Bildschirms ist in Abbildung 6.2 gegeben. Mit den Hardkeys (HK) erfolgt die Anwahl der Hauptmenüs (Das „Car“-Menü im Beispiel), während die zweite Menüebene

[3] http://www.hamacher-online.com/stateshade

[4] Als *Sitzkiste* wird üblicherweise ein Versuchsaufbau bezeichnet, der die Sitze und das Cockpit eines Fahrzeugs beinhaltet. Zur Durchführung der Versuche im Rahmen dieser Arbeit stand die Fahrgastzelle eines BMW 7er (ohne Motorraum und Rückbank) zur Verfügung.

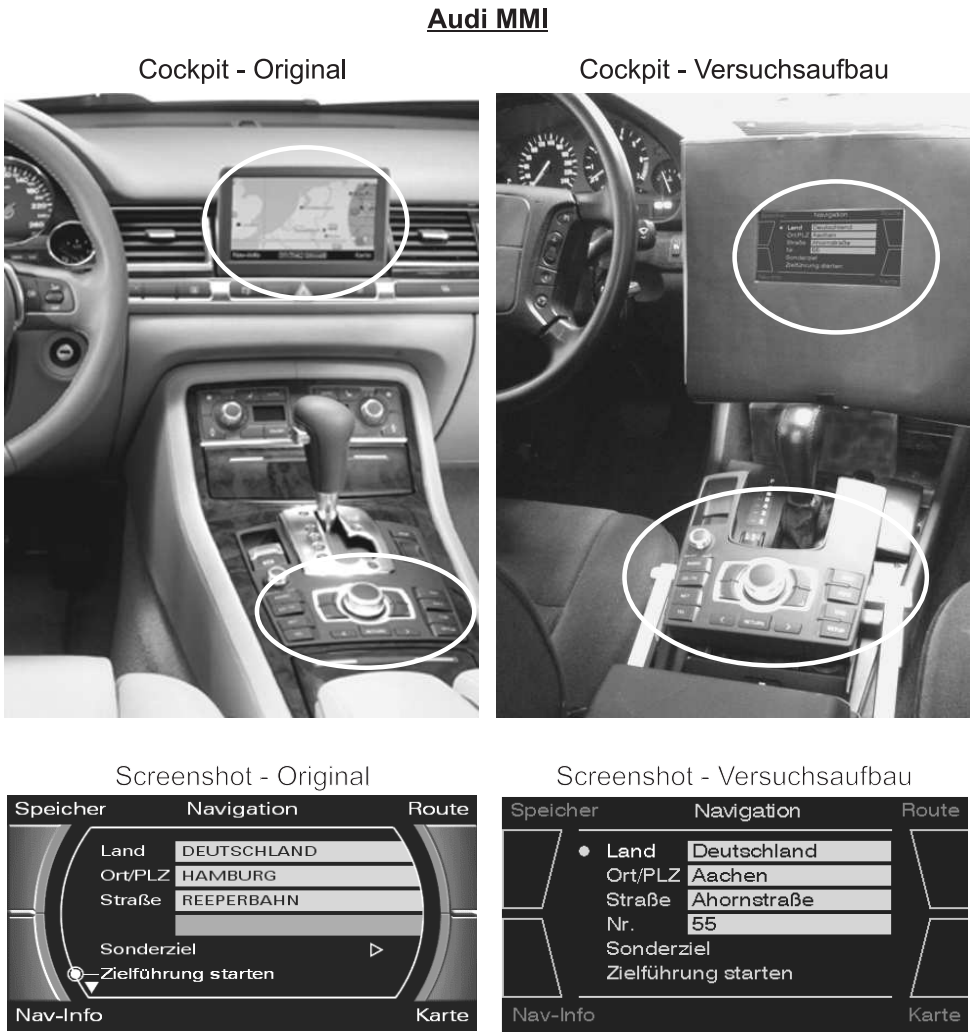

Abbildung 6.1: Das FIS *Audi MMI* im Original (links) und im Versuchsaufbau (rechts). Ein Beispiel für die Darstellung von Informationen auf dem zentralen Display ist ebenfalls dargestellt.

durch die Softkeys (SK) ausgewählt wird. Die Beschriftungen der quadratisch angeordneten Softkeys erfolgt in den Ecken des Displays. Die Bedienung nachfolgender Menüebenen sowie die Einstellung von Funktionswerten erlaubt der Dreh-Drück-Knopf (DDK).

BMW iDrive

Das im aktuellen 7er BMW eingebaute FIS *BMW iDrive* ist im Vergleich zu dem in dieser Arbeit erstellten Versuchsaufbau in Abbildung 6.3 dargestellt. Die Bedienung erfolgt über einen zentralen Dreh-Drück-Knopf (DDK – genannt „iDrive-Controller“), der sich ebenfalls in acht Richtungen schieben lässt. Die Anwahl der ersten Menüebene erfolgt über das Schieben des DDK. Alle weiteren Menüs werden ebenso wie die Einstellung von Funktionswerten durch Drehen und anschließendes Drücken ausgewählt. Das Hauptmenü ist normalerweise nicht sichtbar und wird erst durch zweimaliges Schieben in die gleiche Richtung dargestellt. Beng-

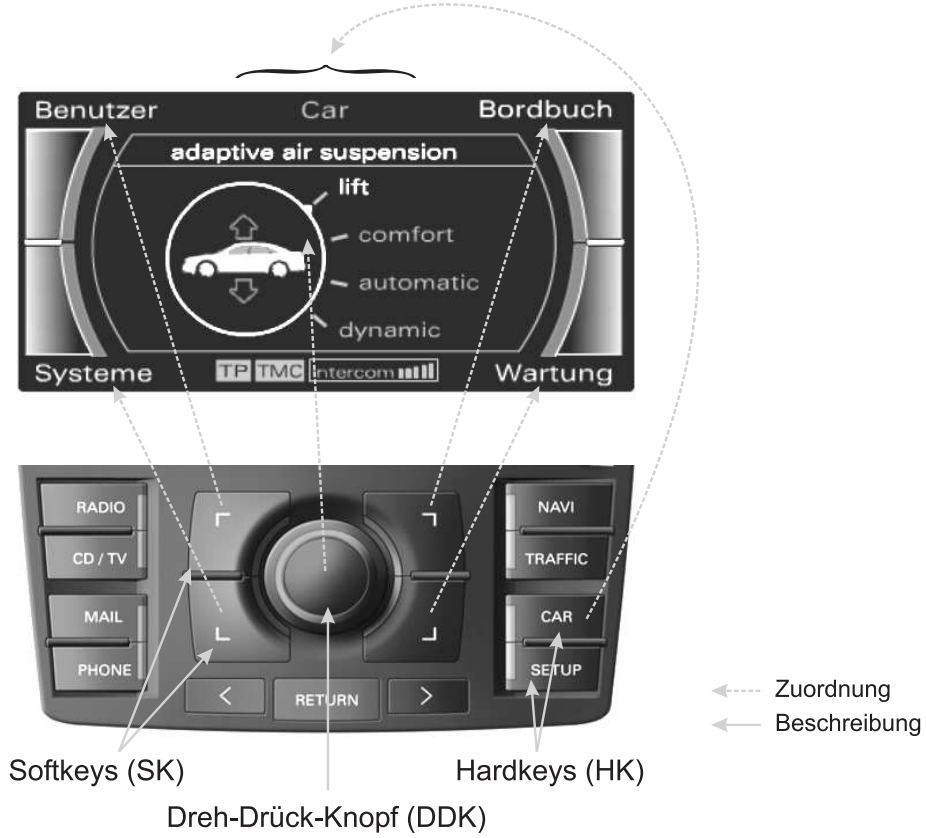

Abbildung 6.2: Das Bedienpanel des Audi MMI mit der Zuordnung der Bedienelemente zu Informationen auf dem Display.

ler et al. rechtfertigen dies dadurch, dass sich der Fahrer die Zuordnung der acht Hauptmenüs zu den Schieberichtungen „nach kurzer Lernphase“ merken kann (BENGLER et al. 2002).

Für den Versuchsaufbau stand ein Bedienelement zur Verfügung, dass das Schieben in vier Richtungen erlaubt und einen Dreh-Drück-Knopf beinhaltet. Dadurch ist die Funktionalität des originalen iDrive-Controllers in allen Aktionsmöglichkeiten abgedeckt. Beide Bedienelemente sind in Abbildung 6.4 dargestellt.

Die Darstellung von Menüs erfolgt beim iDrive auf der linken Seite. Häufig werden Listen bzw. Untermenüs zusätzlich auf der rechten Seite angezeigt. Die Bedienung erfordert ein Drehen des DDK im Uhrzeigersinn, bis sich die Markierung ausgehend vom linken Menü auf der rechten Seite befindet.

6.1.2 Auswahl repräsentativer Aufgaben zur Bewertung

Die vorgestellten FIS stellen interaktive Systeme dar, bei denen die Initiierung des Dialogs immer vom Anwender ausgeht. Daher eignen sich diese Systeme zur automatischen Generierung von Dialogfolgen durch den autonomen Bedienagenten im Rahmen von REVISER.

Zur Bewertung der FIS sowie zur Generierung der Dialogfolgen werden Aufgaben festgelegt,

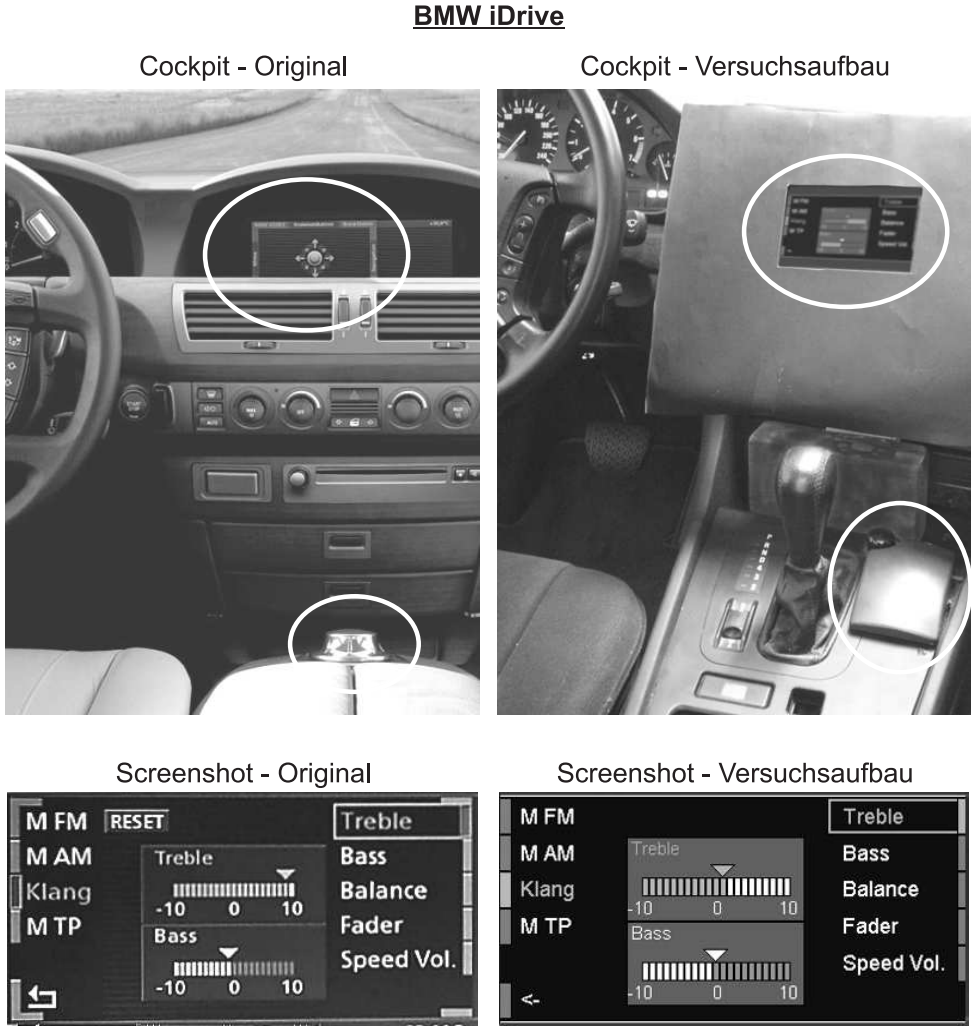

Abbildung 6.3: Das FIS *BMW iDrive* im Original (links) und im Versuchsaufbau (rechts). Ein Beispiel der Darstellung von Informationen auf dem zentralen Display ist ebenfalls abgebildet.

die einen Querschnitt hinsichtlich der Bedienung und Benutzerführung darstellen. Bei diesen exemplarischen Aufgaben handelt es sich um Folgende:

1. Radiosender „WDR3“ einstellen.
   (sehr häufig benutzte Aufgabe – wenige Bedienschritte/Menüebenen – Listenauswahl)

2. Menü zur Einstellung der Klanghöhen aufrufen.
   (selten benutzte Aufgabe – Einstellung eines Funktionswertes – Einstellungsmenü)

3. Zweites Navigationsziel (Aachen, Rütscherstr. 83) aus der Zielliste auswählen.
   (häufig benutzte Aufgabe – viele Bedienschritte/Menüebenen – Listenauswahl)

Die Dialogfolgen für alle sechs Aufgaben sind in Anhang B vorgestellt.

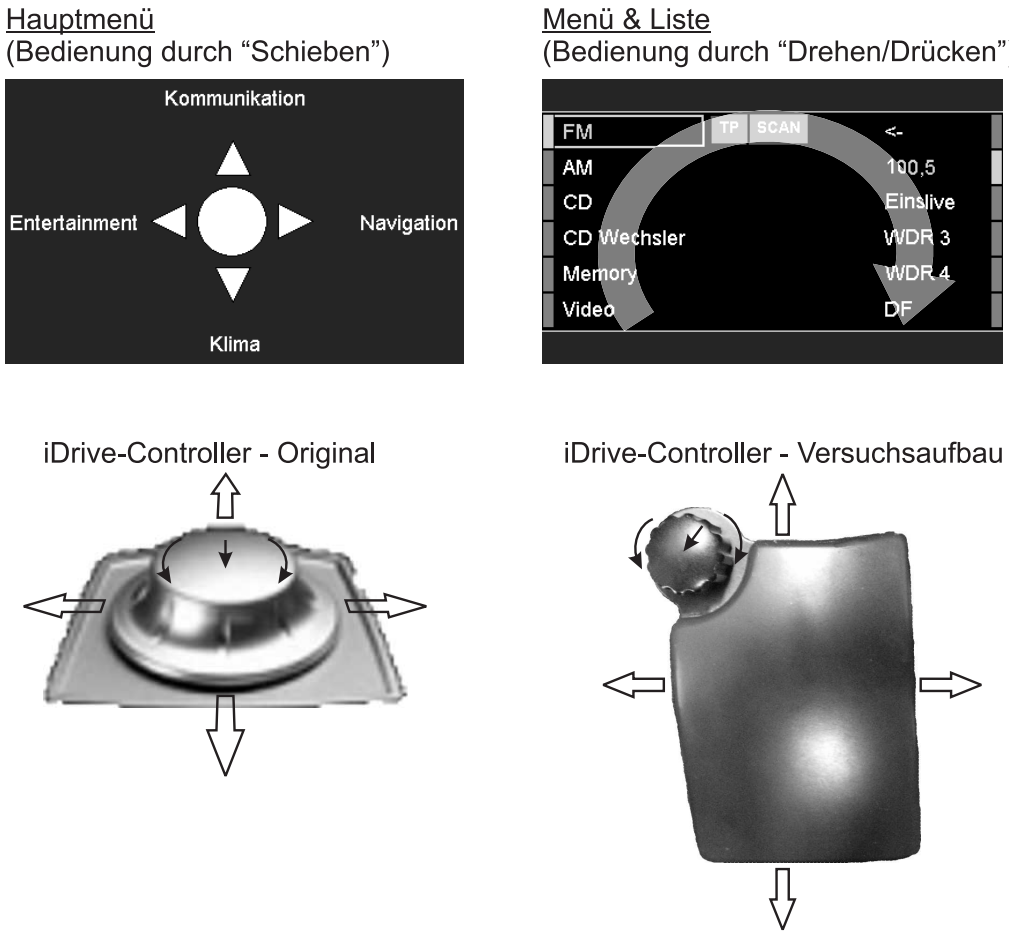

Abbildung 6.4: Das Bedienprinzip zweier Menüdarstellungen des BMW iDrive. Der originale iDrive-Controller im Vergleich zu dem in dieser Arbeit benutzten Bedienelement ist ebenfalls dargestellt.

6.2 Vergleich der Ergebnisse einer kriterienorientierten und zweier empirischer Bewertungen von FIS

Bei der automatischen kriterienorientierten sowie der im Rahmen dieser Arbeit durchgeführten empirischen Evaluierung basiert die Bewertung auf der Bearbeitung dreier Aufgaben mit zwei FIS. Eine Übersicht über die wichtigsten Ergebnisse ist unten gegeben, während die Details der Bewertung in den Kapiteln 6.3 und 6.4 vorgestellt und diskutiert sind. Ein Vergleich der Ergebnisse zeigt, dass bei beiden Bewertungen die gleichen Auffälligkeiten identifiziert wurden.

Die Ergebnisse lassen sich mit einer weiteren Studie vergleichen, die 2003 die Firma SirValUse Consulting GmbH durchführte (SIRVALUSE 2003). Dabei wurden die Bedienkonzepte von fünf Fahrzeugen der Oberklasse (u.a. der Audi A8 und der 7er BMW) empirisch verglichen. Insgesamt führten 13 Probanden an je zwei Fahrzeugen fünf Aufgaben durch. Zu den Aufgaben gehörte u.a. die Auswahl eines Sender im Radiomenü sowie die Anwahl eines Ziels und das Starten der Zielführung im Navigationsmenü (SIRVALUSE 2004).

Die wesentlichen Ergebnisse der in dieser Arbeit durchgeführten kriterienorientierten und empirischen Bewertungen für das Audi MMI und BMW iDrive, zusammen mit den Ergebnissen der SirValUse-Studie, beinhalten die Tabellen 6.1 und 6.2. Dabei sind die Ergebnisse der Bewertung des Oberflächendesigns (inkl. Menüdarstellung), der Bedienung der Bedienelemente und der Aufgabenbearbeitung dargestellt, da sich diese Ergebnisse über alle drei Bewertungen vergleichen lassen.

Tabelle 6.1: Vergleich der Bewertungsergebnisse der automatischen kriterienorientierten, der empirischen und der von der Firma SirValUse durchgeführten Bewertung des Audi MMI.

Audi MMI	REVISER	emp. Bewertung	SirValUse
Oberflächen-design	konsistent / Farbgebung ist noch gut (zu viele Farben)	gut / Farbgebung ist noch gut	gut
Menü-darstellung	eindeutig / je eine eindeutige Überschrift pro Menü	eindeutig	eindeutig
Bedienung der Bedienele-mente	problemlos	problemlos / Zuordnung der SK zu Beschriftungen anfangs ungewohnt	problemlos / Zuordnung der SK zu Beschriftungen anfangs ungewohnt
Aufgaben-bearbeitung	problemlos	problemlos	„die Bedienerführung kann von den Testpersonen schnell erfasst werden, nach kurzer Zeit ist eine sichere Bedienung möglich.“

Anhand der Ergebnisse zeigt sich, dass die automatische, kriterienorientierte Bewertung durch REVISER Designfehler der beiden FIS identifizieren konnte. Die Generierung zahlreicher Maße zur Bewertung des Oberflächendesigns sowie die Berechnung der Konsistenzmaße erlaubt eine gute Abschätzung der subjektiven Meinung späterer Benutzer. Der Einsatz einer Begriffsontologie ermöglicht eine Bewertung der Güte der eingesetzten Begriffe im Rahmen des *Wording*. Vor allem eignet sie sich zur Identifikation schwer (oder nicht) verständlicher Begriffe. Die Bewertung des Interaktionsdesigns durch den Einsatz des autonomen Bedienagenten ergab eine gute Vorhersage der Probleme späterer Benutzer und Fehler im Systemdesign.

Die Bewertung mit REVISER ergab eine eindeutige Präferenz für den Audi MMI. Bei der empirischen Untersuchung empfanden die Probanden ebenfalls das Audi MMI als deutlich besser. Bei der SirValUse-Studie ist das Audi MMI im Vergleich der deutliche Gewinner mit der Note 2 (gemittelter Wert von Schulnoten 1 bis 6) gegenüber dem BMW iDrive als Verlierer des Tests (Note 3,5).

Tabelle 6.2: Vergleich der Bewertungsergebnisse der automatischen kriterienorientierten, der empirischen und der von der Firma SirValUse durchgeführten Bewertung des BMW iDrive.

BMW iDrive	REVISER	emp. Bewertung	SirValUse
Oberflächen-design	inkonsistent / Farbgebung gut	weniger gut / Farbgebung gut	„Die Menüs sind unübersichtlich gestaltet“
Menü-darstellung	„rechtes“ Menü wird bei Aufforderung nur angezeigt und nicht aktiviert / der Begriff „Memory“ sollte unbedingt ersetzt werden/ schlecht: keine Überschriften	die Darstellung des „rechten“ Menüs war verwirrend	„Es ist völlig unklar, wann ein Menüwechsel bzw. wann ein Bereichswechsel innerhalb eines Menüs erfolgt.“
Bedienung der Bedienele-mente	Schieben des Controllers nicht erwartungskonform	Schieben des Controllers zur Bedienung der Untermenüs führte zu großer Verwirrung	„Vielen Testpersonen ist nicht klar, wann gedreht, gedrückt oder geschoben werden muss.“
Aufgaben-bearbeitung	Wording-Fehler („Memory“) sowie Verletzung der Stereotype zum Schieben des DDK führte je zum Aufgabenabbruch	insgesamt 15 *Trial&Error*-Phasen, 7 Aufgabenab-brüche (teilweise mit starken Aggressionen)	„Das System gibt den Testpersonen das Gefühl, die Kontrolle über die Bedienung zu verlieren, da sie auch nach einiger Zeit immer noch nicht wissen, wie es zu bedienen ist. Einige macht die weitere Verwendung des iDrive sogar aggressiv.“

6.3 Automatische Bewertung mit REVISER

Die in Kapitel 3.4 vorgestellten Maße wurden in REVISER als Regeln implementiert und als Projekt abgespeichert. Anschließend erfolgte die Bewertung der beiden FIS Audi MMI und BMW iDrive mit REVISER.

6.3.1 Durchführung der automatischen kriterienorientierten Bewertung

Zusätzlich zu den implementierten Maßen erfolgt zur automatischen Bewertung im Rahmen dieser Arbeit die Integration einer Ontologie für die in FIS gebräuchlichen Begriffen sowie die Erstellung eines autonomen Bedienagenten zur Generierung einer Dialogfolge im Werkzeug REVISER. Die Details zu den einzelnen Bereichen sind im Folgenden vorgestellt.

Maße des Oberflächendesigns

Zur Bewertung des Oberflächendesigns wurden Maße der Lage und Anordnung von GUI-Elementen sowie der Farbgebung errechnet. Für alle Maße, die als Ergebnis einen Zahlenwert besitzen, erfolgt zusätzlich die Berechnung eines entsprechenden Konsistenzwertes (Z-Score und *CoV* – vgl. Kapitel 3.4.2.4).

Maße des *Wordings* im Rahmen des Informationsdesigns

Zur Bewertung des *Wordings* erfolgte im Rahmen dieser Arbeit die Erstellung einer Ontologie für Begriffe der Domäne „Fahrerinformationssysteme (FIS)“ (s. Kapitel 3.4.3). Dazu standen 13 Personen[5] zur Verfügung, die durchschnittlich 13 Jahre Fahrerfahrung haben und mit der Bedienung von Autoradios sowie Navigationssystemen vertraut sind.

Mit diesen Personen erfolgte gruppenweise ein Brainstorming zum Sammeln relevanter Begriffe, die mit der Funktionalität von FIS in Zusammenhang stehen. Insgesamt wurden 613 unterschiedliche Begriffe für die Domäne FIS identifiziert. Anschließend ordneten die Personen diese Begriffe einander in hierarchischer Form zu. Insgesamt wurden so 356 Begriffe in der Hierarchie verwendet und 257 Synonyme zugeordnet. Details zum Vorgehen bei der Ontologie-Erstellung sowie ein Auszug der FIS-Ontologie sind im Anhang C gegeben.

Implementierung des Bedienagenten und Maße des Interaktionsdesigns

Die Maße des Interaktionsdesigns gemäß Kapitel 3.4.4 wurden ebenso wie der autonome Agent zur Erstellung einer Dialogfolge für eine Aufgabe mit der in Kapitel 4 beschriebenen Funktionalität in REVISER implementiert. Die Aufgabenziele der sechs Aufgaben stehen dem Agenten zu Beginn jeder Bedienung in Form eines Ontologieknotens mit dem entsprechenden Zielbegriff zur Verfügung. Darüber hinaus sind die gängigen Bedienstereotypen für die benutzten Bedienelemente zur Aktionsfindung (s. Abschnitt 4.4) in REVISER integriert.

Der Agent speichert die GUI-Beschreibung sowie das benutzte *Wissen* (z.B. die erkannte *selektierte Menüoption (SMO)* bzw. benutzte Stereotypen) während der Bedienung zusammen mit der am Prototypen durchgeführten Aktion. Anschließend erfolgte die Auswertung der Dialogfolge mit den Maßen des Oberflächen- und Informationsdesigns für jeden Dialog sowie mit den Maßen des Interaktionsdesigns über die komplette Sequenz

[5] Bei den Personen zur Erstellung der FIS-Ontologie handelte es sich nicht um die Probanden zur Durchführung der empirischen Bewertung

6.3.2 Ergebnisse der automatischen kriterienorientierten Bewertung des Audi MMI

Bei der Bearbeitung des Audi MMI war der Agent in der Lage, für jede Aufgabe das Ziel zu erreichen. Die Ergebnisse der Aufgabenbearbeitung mit den entsprechenden Werten der jeweiligen Maße sowie aller Alarme sind im Folgenden erläutert.

Bewertung des Oberflächendesigns

Einzelmaße/Konsistenzmaße:

Die Dialogfolgen aller Aufgaben haben ein durchweg gleichförmiges Design. Weiterhin gibt es keine Auffälligkeiten bei den Einzelmaßen.

Beispielhaft sind in Abbildung 6.5 die Werte der Oberflächenmaße für alle fünf Bedienschritte (S.1 - S.5) der Aufgabe „Höhen einstellen“ aufgeführt (die Ergebnisse aller Aufgaben finden sich in Anhang E). Die entsprechenden Z-Score-Werte für die einzelnen Bedienschritte sowie der *CoV*-Wert der Aufgabe sind ebenfalls dargestellt. Z-Score-Werte, die eine Inkonsistenz anzeigen (Wert ist > 1 bzw. < -1), sind in der Tabelle grau markiert. Der erste Dialog der Aufgaben weicht in manchen Werten von den übrigen ab, was daran liegt, dass die Aufgabenbearbeitung in einem anderen Menü startet (Telefonmenü), das sich von der Farbgebung und der Anzahl der dargestellten GUI-Elemente vom Radio- und Navigationsmenü unterscheidet.
Die etwas schlechteren Konsistenzwerte in der Navigationsaufgabe erklären sich dadurch, dass das Navigationsmenü die Adresse des aktuell eingegebenen Ziels enthält. Das wirkt sich durch eine deutlich erhöhte Anzahl von GUI-Elementen aus. Die Darstellung dieser Elemente innerhalb dieses Menüs ist durchgängig gleichförmig.

Audi - "Höhen einstellen"	Ergebnisse					Z-Scores					CoV
Maß	S.1	S.2	S.3	S.4	S.5	S.1	S.2	S.3	S.4	S.5	
background color count	2,00	2,00	2,00	2,00	2,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	3,00	3,00	3,00	3,00	3,00	0,00	0,00	0,00	0,00	0,00	0,00
display aspect ratio	0,50	0,50	0,50	0,50	0,50	0,00	0,00	0,00	0,00	0,00	0,00
foreground color count	4,00	4,00	4,00	4,00	4,00	0,00	0,00	0,00	0,00	0,00	0,00
gridedness	36,17	39,13	42,55	42,55	42,55	-1,72	-0,57	0,76	0,76	0,76	6,35
horizontal area-balance	-1,30	-17,60	-11,70	-10,80	-11,60	1,77	-1,33	-0,21	-0,04	-0,19	-49,52
margin east	-1,00	-1,00	-1,00	-1,00	-1,00	0,00	0,00	0,00	0,00	0,00	0,00
margin north	1,00	1,00	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
margin south	6,00	6,00	6,00	6,00	6,00	0,00	0,00	0,00	0,00	0,00	0,00
margin west	1,00	1,00	0,00	0,00	0,00	1,22	1,22	-0,82	-0,82	-0,82	122,47
non widget area	59,39	70,29	70,12	70,12	70,12	-2,00	0,53	0,49	0,49	0,49	6,34
selected item count	1,00	1,00	1,00	1,00	1,00	0,00	0,00	0,00	0,00	0,00	0,00
text widget count	10,00	11,00	11,00	11,00	11,00	-2,00	0,50	0,50	0,50	0,50	3,70
vertical area-balance	-2,70	-23,80	-25,40	-24,60	-25,60	1,99	-0,38	-0,56	-0,47	-0,58	-43,50
widget count	17,00	18,00	20,00	20,00	20,00	-1,58	-0,79	0,79	0,79	0,79	6,66
widget density	14,70	15,62	17,30	17,30	17,30	-1,60	-0,76	0,79	0,79	0,79	6,62
word count	12,00	14,00	12,00	12,00	12,00	-0,50	2,00	-0,50	-0,50	-0,50	6,45

Abbildung 6.5: Ergebnisse, Z-Scores und *CoV* der Oberflächenmaße aller Bedienschritte für die Aufgabe „Höhen einstellen“ beim Audi MMI. Z-Score-Werte, die auf Inkonsistenzen hinweisen, sind grau hinterlegt.

Alarme: Fehlende Gleichförmigkeit der Textfarben.

Es trat der Alarm „Main Color Changed“ auf, der aussagt, dass sich die Farbe der meisten Textelemente verändert hat. Der Grund liegt darin, dass jeder Themenbereich im Audi MMI (z.B. „Telefon“ oder „Navigation“) die Textelemente in einer anderen Farbe darstellt. Dadurch leidet die Gleichförmigkeit der Farbdarstellung.

Farbgebung:

Es kommen (zu) viele Farben zum Einsatz.

Der Agent registrierte durchgehend in jedem Dialog sechs Farben. Bei der Farbgebung der Textelemente konnte der Agent zwei unterschiedliche Gruppen unterscheiden: die Beschriftungen der Softkeys sowie die Begriffe zur Menüdarstellung (inkl. zusätzlicher Farben zur Markierung der selektierten Menüoption – SMO). Die Anzahl der Farben liegt noch im akzeptablen Bereich. Die Farbabstände (ΔE) und Kontraste zwischen allen Farben sind gut.

Darstellung der selektierten Menüoption (SMO):

Die Darstellung des SMO ist eindeutig und durchgängig.

Der Agent erkannte eine doppelte Markierung der selektierten Menüoptionen in Form einer Farbänderung und der Voranstellung eines kleinen Kreises (vgl. Abb. 6.1). Die Art der Markierung war in allen Dialogen der Dialogfolge gleich.

Beschriftung der Softkeys:

Eindeutige Beschriftung der Softkeys.

Die Softkeys liegen zwar nicht in direkter räumlicher Nähe zur Beschriftung, über die quadratische Anordnung war der Agent dennoch in der Lage, die Beschriftungen im Display (in den Ecken des Displays) zu finden und eindeutig zuzuordnen. Die Zuordnung wurde durch eine eigene Farbkodierung der Begriffe verstärkt (s.o.) und änderte sich während der Aufgabenbearbeitung nicht.

Darstellung von Informationen:

Die Lage der Informationen sowie Beschriftungen von Softkeys entsprechen den jeweiligen Empfehlungen.

Schriftgröße:

Die Schriftgröße aller dargestellten Begriffe ist ausreichend groß gewählt.

Bewertung des *Wording* im Rahmen des Informationsdesigns

Alarme: Unbekannter Begriff („DSP Bose“) entdeckt.

Im Radiomenü konnte der Begriff „DSP Bose“ nicht in der Ontologie gefunden werden und erzeugte einen Alarm. Zur Beschreibung dieser Spezialfunktion sollte daher ein gängigerer Begriff gewählt werden. Der Begriff wird jedoch nicht zur Bearbeitung der gestellten Aufgaben benötigt und stellt daher kein Hindernis dar.

Überschrift:

In allen Menüs konnte eine eindeutige Überschrift identifiziert werden, die sich der Tiefe des Menübaums anpasst. Dadurch ist eine Möglichkeit der Orientierung – vor allem nach einer Unterbrechung der Aufgabenbearbeitung – gewährleistet.

Synonyme:

Es wurden keine Synonyme entdeckt.

Bewertung des Interaktionsdesigns

Anzahl Bedienaktionen:

über alle Aufgaben: 18 (Senderauswahl: 5, Höheneinstellung: 7, Navigation: 6)
(Im Vergleich weniger als beim BMW iDrive)

Anzahl Menüebenen:

Senderwahl: 1, Höheneinstellung: 3, Navigation: 4
(Im Vergleich weniger als beim BMW iDrive)

Bedienkonsistenz/Stereotypen:

Die Bedienung des Systems ist erwartungskonform.

Es fand keine Verletzung einer Stereotype statt. Bei allen drei Aufgaben erfolgt durchgängig die erste Aktion durch einen Hardkey. Die zweite Menüebene wurde (bei Bedarf) durch einen Softkey ausgewählt. Die Auswahl eines Elementes aus einer Liste sowie die Bedienung der weiteren Menüebenen erfolgt mit Hilfe des Dreh-Drück-Knopfes. Ausnahmen davon konnten nicht entdeckt werden.

Alarme: Es wurden keine Alarme produziert.

6.3.3 Ergebnisse der automatischen kriterienorientierten Bewertung des BMW iDrive

Bei der Bedienung war der Agent nicht in der Lage, alle Aufgaben zu beenden. An den jeweiligen Stellen erzeugte er entsprechende Alarme und brach die Bearbeitung ab. Erst nach einer manuellen Modifikation des dem Agenten zur Verfügung stehenden Wissens konnten alle Aufgaben erfolgreich bearbeitet werden. Die Werte der jeweiligen Maße, die Alarme und die Modifikationen sind im Folgenden erläutert.

Bewertung des Oberflächendesigns

Einzelmaße/Konsistenzmaße:

Durchgängig schlechte Konsistenz des GUI-Aufbaus über alle Dialoge und Aufgaben.

Beispielhaft sind in Abbildung 6.6 die Werte der Oberflächenmaße für fünf (S.8 - S.12) der insgesamt 19 Bedienschritte der Aufgabe „Höhen einstellen“

aufgeführt (die Ergebnisse aller Aufgaben finden sich in Anhang E). Die entsprechenden Z-Score-Werte für die einzelnen Bedienschritte sowie der *CoV*-Wert der Aufgabe sind ebenfalls dargestellt. Bei den Werten der Z-Score zeigt sich, dass sich das Menüdesign häufig stark ändert (s. Darstellung der Dialogfolgen in Anhang B). Im ersten Dialog ist das Hauptmenü mit der Angabe der Schieberichtungen für die Untermenüs dargestellt. In den weiteren Dialogen der Dialogfolge werden teilweise ein Menü (zweite Menüebene) oder zwei Menüs (folgende Menüebenen) dargestellt. Das Menü zur Auswahl eines Navigationsziels aus der Zielliste ist wiederum vollständig anders aufgebaut (horizontale statt einer vertikalen Navigation wie in den anderen Menüs). Diese großen Unterschiede der verschiedenen Menüdesigns schlagen sich in entsprechend schlechten Konsistenzwerten der einzelnen GUI-Konstellationen nieder.

BMW "Höhen einstellen"	Ergebnisse						Z-Scores						CoV
Maß	S.8	S.9	S.10	S.11	S.12	…	S.8	S.9	S.10	S.11	S.12	…	
background color count	3,00	3,00	3,00	3,00	3,00		0,00	0,00	0,00	0,00	0,00		0,00
column count	2,00	1,00	1,00	1,00	2,00		0,34	-1,29	-1,29	-1,29	0,34		0,00
display aspect ratio	0,48	0,48	0,48	0,48	0,48		-0,24	-0,24	-0,24	-0,24	-0,24		0,00
foreground color count	2,00	1,00	1,00	1,00	2,00		0,43	-2,31	-2,31	-2,31	0,43		0,00
gridedness	55,56	35,71	45,45	45,45	52,63		0,72	-2,20	-0,77	-0,77	0,29		6,35
horizontal area-balance	-2,50	-100,00	-100,00	-100,00	-9,13		0,36	-2,26	-2,26	-2,26	0,18		-49,52
margin east	41,00	335,00	335,00	335,00	43,00		-0,42	2,30	2,30	2,30	-0,40		0,00
margin north	6,00	6,00	6,00	6,00	6,00		-0,24	-0,24	-0,24	-0,24	-0,24		0,00
margin south	8,00	8,00	70,00	70,00	8,00		-0,34	-0,34	2,92	2,92	-0,34		0,00
margin west	19,00	19,00	19,00	19,00	19,00		0,24	0,24	0,24	0,24	0,24		122,47
non widget area	72,10	72,10	89,78	90,27	90,27		-1,21	-1,21	1,40	1,48	1,48		6,34
selected item count	1,00	1,00	1,00	1,00	1,00		0,24	0,24	0,24	0,24	0,24		0,00
text widget count	15,00	5,00	5,00	5,00	10,00		1,12	-1,51	-1,51	-1,51	-0,19		3,70
vertical area-balance	12,74	-38,69	-22,12	-61,40	-51,89		1,49	-0,42	0,20	-1,26	-0,91		-43,50
widget count	15,00	5,00	5,00	5,00	10,00		1,14	-1,72	-1,72	-1,72	-0,29		6,66
widget density	19,74	6,58	6,58	6,58	13,16		1,15	-1,64	-1,64	-1,64	-0,25		6,62
word count	18,00	8,00	9,00	8,00	13,00		1,00	-1,43	-1,19	-1,43	-0,22		6,45

Abbildung 6.6: Ergebnisse, Z-Scores und *CoV* der Oberflächenmaße einiger Bedienschritte für die Aufgabe „Höhen einstellen“ beim BMW iDrive. Z-Score-Werte, die auf Inkonsistenzen hinweisen, sind grau hinterlegt.

Alarme: Es wurden keine Alarme generiert.

Farbgebung:

Gute Farbwahl mit guter Anzahl an Farben.

Der Agent registrierte pro Menü durchschnittlich vier Farben (im Hauptmenü zwei Farben) zur Darstellung der Informationen. Die Farbabstände (ΔE) und Kontraste zwischen allen Farben sind sehr gut. Die Farbgebung ist über alle Aufgaben konstant.

Darstellung der selektierten Menüoption (SMO):

Die Darstellung des SMO ist eindeutig und durchgängig.

Der Agent erkannte eine doppelte Markierung der selektierten Menüoptionen in Form einer Farbänderung und der Voranstellung eines Vierecks (vgl. Abb. 6.3). Die Art der Markierung war in allen Dialogen der Dialogfolge gleich.

Darstellung von Informationen:

Der Menüaufbau ist nicht einheitlich.

Im Display werden teilweise zwei Menüebenen gleichzeitig angezeigt. Die Darstellung erfolgt jeweils am linken bzw. rechten Rand des Displays. Bei der Aufgabe der Höheneinstellung ist zusätzlich in der Mitte des Displays eine Möglichkeit der Einstellung gegeben. Daher entspricht die Lage der Informationen nur bedingt der Empfehlung zur Bildschirmaufteilung.

Schriftgröße:

Die Schriftgröße aller dargestellten Begriffe ist ausreichend groß gewählt.

Bewertung des *Wording* im Rahmen des Informationsdesigns

Alarme: „Memory“ ist als Oberbegriff von „Klanghöhen“ nicht erwartungskonform.

Das Menü zur Einstellung der Klanghöhen erscheint durch die Anwahl der Menüoption „Memory“ im Radiomenü (s. Abbildung D.5). Dieser Zusammenhang ist in der Ontologie nicht gegeben. Daher bricht der Agent bei der Aufgabenbearbeitung an dieser Stelle mit einem Alarm ab. Erst durch das Hinzufügen von „Memory“ an der entsprechenden Stelle in der Ontologie ist der Agent in der Lage, die Aufgabe erfolgreich zu bearbeiten. Es sollte an dieser Stelle ein anderer Begriff (z.B. „Einstellungen“) benutzt werden.

Überschrift:

Es existieren keine Überschriften. Dadurch wird die (Neu-)Orientierung bei der Aufgabenbearbeitung deutlich erschwert.

Synonyme:

Es wurden keine Synonyme entdeckt.

Bewertung des Interaktionsdesigns

Anzahl Bedienaktionen:

über alle Aufgaben: 44 (Senderwahl: 9, Höheneinstellung: 13, Navigation: 22)
(Im Vergleich mehr als beim Audi MMI)

Anzahl Menüebenen:

Senderwahl: 1, Höheneinstellung: 4, Navigation: 4
(Im Vergleich mehr als beim Audi MMI)

Bedienkonsistenz/Stereotypen:

Die Bedienung des Systems war nicht erwartungskonform. Der Agent war lediglich in der Lage, das System beginnend vom Hauptmenü zu bedienen.

Startet die Bedienung in einem anderen Menü (z.B. Navigationsmenü), dann gibt es keine Hinweise (textueller Art), wie ein Wechsel in einen anderen Themenbereich (z.B. ins Radiomenü) vorgenommen werden kann, was eine gravierende Schwäche in der Selbstbeschreibungsfähigkeit des Systems darstellt. Um dieses Problem zu umgehen, musste die Aufgabenbearbeitung des Agenten im Hauptmenü des iDrive starten.

Das Schieben des Controllers ist nicht erwartungskonform.

Die Schieberichtungen des iDrive-Controllers sind nicht beschriftet (im Sinne eines Hardkeys), daher geht der Agent von einer wechselnden Funktionsaktivierung durch das Schieben im Sinne eines Softkeys aus. Um die Funktionsbeschreibung der jeweiligen Schieberichtung zu ermitteln, erfolgt daher die Überprüfung der Kompatibilität der „Lage“ der Schieberichtungen und entsprechender Begriffe im Display. Im Hauptmenü können entsprechende Begriffe gefunden und auch eindeutig zugeordnet werden. Die Auswahl des entsprechenden Themenbereichs durch ein Schieben des DDK ist daher problemlos möglich. Ebenso erwartungskonform verlief die Bedienung der Menüs auf der linken Displayseite.
In den weiteren Menüs lassen sich jedoch keine Begriffe als Funktionsbeschreibungen den Schieberichtungen des Controllers zuordnen, da solche Begriffe nicht vorhanden sind. Daher klassifiziert der Agent die Schieberichtungen als „Aktionen zur Menübedienung“. Der Wechsel von einem Menü auf der linken Seite in ein Menü auf der rechten Seite durch das Schieben des DDK ist ebenfalls nicht möglich (vgl. Abb. 6.7), was die entsprechende Bedienstereotype verletzt. Ein Menüwechsel ist nur durch das Drehen des DDK im Uhrzeigersinn möglich. Die Darstellung der Menüs gibt jedoch keinen Hinweis darauf – die Anzeige ist zu den geforderten Aktionen nicht kompatibel.

Alarme: Eine Bedienaktion beim Menüwechsel ist überflüssig.

Das Menü für die Einstellfunktionen des Radios ist auf der linken Displayseite dargestellt. Nach Anwahl der Menüoption „Klang“ erscheint das Klangmenü auf der rechten Seite (vgl. Abb. 6.7). Allerdings wechselt das System nicht automatisch in dieses Menü, sondern es bleibt „Klang“ selektiert. Der Benutzer muss durch Drehen des DDK erst in das nun sichtbare Klangmenü wechseln.
Der Bedienagent erkennt an dieser Stelle ebenfalls keinen Zustandswechsel des Systems, da die selektierte Menüoption gleich bleibt, und generiert einen Alarm, da er einen Zustandswechsel erwartet hat. An dieser Stelle sollte bei Anwahl der Menüoption „Klang“ automatisch ein Wechsel ins Klangmenü erfolgen. Das Gleiche gilt für die Auswahl der Zielliste im Navigationsmenü; dort wird ebenfalls ein Alarm generiert.

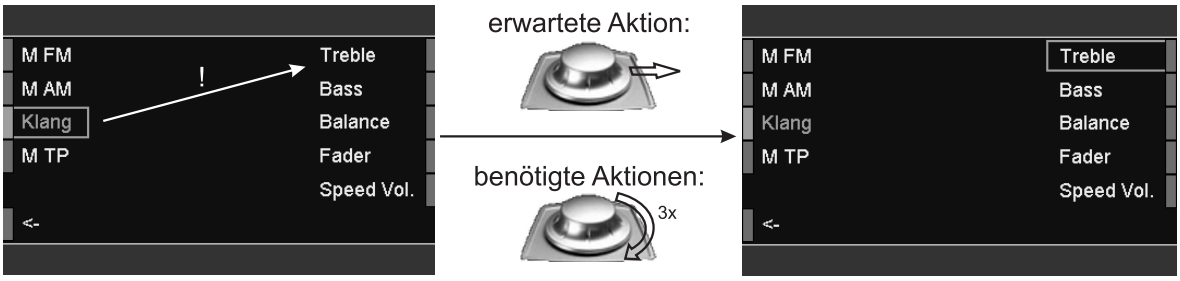

Abbildung 6.7: Verletzung einer Bedienstereotype beim Menüwechsel im BMW iDrive.

6.3.4 Vergleich der Bewertungsergebnisse für das Audi MMI und BMW iDrive

Ein Vergleich der Ergebnisse der automatischen kriterienorientierten Bewertung der beiden FIS Audi MMI und BMW iDrive zeigt deutlich, dass das Audi MMI ein durchgängig besseres Systemdesign besitzt. Dabei handelt es sich bei den meisten von REVISER ermittelten Bewertungsergebnissen um Aussagen des allgemeinen Systemdesigns und sind auch auf andere Aufgaben übertragbar.

Zum Vergleich der Oberflächenkonsistenz ist in Abbildung 6.8 das Konsistenzmaß *CoV* für alle Werte der jeweiligen Oberflächenmaße sowohl für jede Aufgabe als auch über alle Aufgaben gemittelt dargestellt. Die *CoV* wird zur vergleichenden Konsistenzbewertung eingesetzt, wobei ein kleinerer Wert eine größere Konsistenz für das jeweilige Maß angibt. In Abbildung 6.8 stehen die jeweils ermittelten Werte des Audi MMI und des BMW iDrive nebeneinander, wobei der kleinere von beiden hellgrau („konsistenter“) und der größere schwarz („inkonsistenter“) eingefärbt ist. Bei nahezu allen errechneten Oberflächenmaßen zeigt sich beim Audi eine deutlich höhere Konsistenz der Darstellung.

	Senderwahl		Navigationsziel		Klanghöhen		Über alle Aufgaben	
Maß / CoV	Audi	BMW	Audi	BMW	Audi	BMW	Audi	BMW
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	0,00	43,30	12,50	44,72	0,00	34,30	8,39	41,00
display aspect ratio	0,00	0,00	0,00	7,71	0,00	6,94	0,00	77,17
foreground color count	0,00	0,00	15,59	22,22	0,00	19,79	13,74	18,84
gridedness	0,58	15,91	5,76	6,39	6,35	13,43	6,23	18,47
horizontal area-balance	-49,01	74,82	-50,21	-92,49	-49,52	-232,76	-54,98	-193,99
margin east	0,00	31,60	0,00	132,32	0,00	124,90	0,00	131,47
margin north	0,00	5,40	0,00	4,11	0,00	3,69	0,00	4,24
margin south	0,00	0,00	0,00	87,32	0,00	130,99	0,00	113,92
margin west	0,00	24,74	122,47	17,88	122,47	15,85	86,60	18,53
non widget area	0,58	6,82	7,52	13,80	6,34	8,42	11,34	11,66
selected item count	0,00	37,80	0,00	26,73	0,00	23,57	0,00	27,74
text widget count	0,58	26,71	17,31	30,36	3,70	35,48	11,19	33,28
vertical area-balance	-52,18	964,57	-55,35	-38,39	-43,50	-98,04	-61,39	-94,49
widget count	0,58	13,91	8,38	25,06	6,66	31,77	7,58	27,92
widget density	0,58	19,28	8,38	26,80	6,62	32,99	7,55	29,76
word count	0,58	28,66	16,18	32,61	6,45	29,62	17,29	33,00

Abbildung 6.8: Die *CoV*-Werte aller automatisch berechneten Bewertungsmaße für das Oberflächendesign. Der jeweils bessere Wert ist hellgrau, der schlechtere schwarz hinterlegt.

Die Farbgestaltung ist beim BMW iDrive aufgrund der geringeren Anzahl der Farben sowie der durchgängig gleichen Verwendung der Textfarbe besser. Beide Systeme verwenden zur Markierung einer selektierten Menüoption ein zusätzliches geometrisches Element sowie eine Farbänderung der Option. Diese Maßnahme erlaubt eine eindeutige Zuordnung und ist in beiden Systemen durchgängig realisiert.

Die benutzten Begriffe beider FIS sind verständlich und eindeutig. Die Anordnung der Be-

griffe in einem Menübaum ist erwartungskonform. Der Begriff „DSP Bose“ im Menü des Audi MMI war für den Agenten nicht verständlich und sollte überprüft werden. Deutlich gravierender wirkte sich der Begriff „Memory“ zur Beschreibung des Klangmenüs aus. Diese Zuordnung ist nicht verständlich, ist jedoch Teil der Aufgabe und führt daher zum Aufgabenabbruch.

Die Bedienung des Audi erfolgte in allen Menüstufen erwartungskonform und ohne Verletzung einer Bedienstereotype. Aus jedem Menü des Systems sind intuitiv alle Themenbereiche des Audi MMI beschrieben und erreichbar. Jedes Menü besitzt eine deskriptive Überschrift, was die (Neu)-Orientierung während der Bedienung unterstützt.

Die Bedienung des BMW iDrive stellt höhere Ansprüche an den Benutzer, da zahlreiche Aspekte nicht erwartungskonform gelöst sind. Die Schieberichtungen des iDrive-Controllers sind mit der Auswahl jeweils genau eines Menüs im Sinne eines Hardkeys verbunden. Jedoch fehlt dazu jegliche Beschriftung am Controller. Die Bedienung weiterer Menüs ist durch das Schieben nicht möglich. Eine einfache Lösung zur Umgehung dieses Konflikts besteht in der entsprechenden Beschriftung des iDrive-Controllers. Bei der Auswahl einer Menüoption eines Menüs auf der linken Displayseite wird lediglich ein weiteres Menü (bzw. eine Liste) auf der rechten Displayseite dargestellt. Der Wechsel in dieses Menü erfolgt erst durch weitere Bedienaktionen. Dabei handelt es sich um das Drehen des Controllers im UZS, was ebenfalls durch den Menüaufbau nicht ersichtlich ist.

6.4 Empirische Bewertung

Zur Durchführung der empirischen Bewertung wurden die FIS Audi MMI und BMW iDrive nachgebaut und in einer Sitzkiste integriert (s. Abschnitt 6.1). Ziel war die Bearbeitung der in Abschnitt 6.1.2 festgelegten Aufgaben durch Probanden.

Durch den Vergleich der Ergebnisse der automatischen Bewertung mit Ergebnissen der empirischen Bewertung sollen folgende Punkte nachgewiesen werden:

- Das Verhalten der Probanden wird durch das Vorgehen des Agenten realistisch abgebildet.
- Alle von den Probanden genannten Auffälligkeiten der Systeme sind durch die vom Agenten gelieferten Messwerte ebenfalls identifizierbar.
- Das von den Probanden subjektiv als besser empfundene System wird auch vom Agenten besser bewertet.

6.4.1 Versuchsvorbereitung und -durchführung

Die Versuche wurden mit insgesamt 13 Probanden im Alter von 23 bis 47 Jahren und einer durchschnittlichen jährlichen Fahrleistung von 29.500 km pro Jahr durchgeführt. Diese

Stichprobe ist groß genug, um aussagekräftige Ergebnisse zu erzielen (NIEDERMAIER 2002). Alle Personen verfügen über den Führerschein Klasse B (bzw. 3) und sind farbtüchtig (als Voraussetzung der Bewertung der Farbgebung).

Keiner der Probanden ist Dauernutzer eines der Systeme oder hat bereits anderweitig vertiefte Kenntnisse, die zu Verfälschungen der Ergebnisse führen könnten. Alle Probanden sprechen muttersprachlich Deutsch und verfügen über das benötigte technische Vokabular in üblichem Umfang, was begrifflichen Verständnisschwierigkeiten vorbeugt.

Die Einweisung in die Fahrerinformationssysteme erfolgte mündlich. Der Versuchsleiter stellte die Funktionsweise der Bedienelemente vor. Es wurden keine weiteren Hilfestellungen gegeben, damit die intuitive Bedienbarkeit der FIS unverfälscht überprüfbar bleibt. Die Reihenfolge der Aufgaben war immer gleich, um einen Lerneffekt nachverfolgen zu können.

Vor dem Versuch beantwortete jeder Proband einen Fragebogen mit persönlichen Daten, woraufhin er nacheinander die drei Aufgaben durchführte, wobei vor und nach jedem Bedienschritt seine Intention, Erwartung und Einschätzung der Systemreaktion protokolliert wurde. Nach jeder Aufgabenbearbeitung erfolgte die Bewertung der Aufgabe durch den Probanden in Form eines Debriefings mit Hilfe semantischer Differenziale. Die subjektive Meinung des Probanden bzgl. des Systemdesigns und -bedienung fand abschließend mittels eines Fragebogens statt (KURSCHEID 2005).

6.4.2 Ergebnisse der empirischen Bewertung

Bei der Bedienung des Audi MMI waren alle Probanden in der Lage, die Aufgaben erfolgreich zu beenden. Die Bedienung des BMW iDrive stellt die Probanden vor ein deutlich größeres Problem. Dabei kam es häufig vor, dass ein Proband bei der Bedienung die Orientierung verlor und wahllos das System bediente, um wieder einen bekannten Systemzustand zu erreichen. Diese Phase ist im Folgenden mit *Trial&Error* bezeichnet.

Eine statistische Darstellung der aufgetretenen Fehler bietet Abbildung 6.9. Das linke Diagramm zeigt das durchschnittliche Verhältnis falscher (innerhalb der Dialogfolge) zu richtigen Bedienschritten. *Trial&Error*-Phasen, die naturgemäß ausschließlich aus „falschen“ Schritten (im Sinne der Dialogfolge) bestehen, sind dabei nicht berücksichtigt. Das rechte Diagramm gibt eine Übersicht, wie viele Probanden bei den jeweiligen Aufgaben in eine *Trial&Error*-Phase gerieten bzw. die Aufgabenbearbeitung abbrachen.

Beim Audi MMI traten in der ersten Aufgabe vermehrt Fehler auf, da einigen Probanden die Zuordnung der Softkeys zu den Funktionsbeschriftungen nicht von vornherein klar war. Sie erkannten das Prinzip jedoch schnell durch Ausprobieren, so dass alle Probanden nach Beendigung der Versuche diese Zuordnung als „Designprinzip“ benennen konnten. Die Aufgabe zur Einstellung der Höhen wurde nahezu fehlerfrei gelöst. Bei der Navigation vermuteten zwei Probanden die Liste der gespeicherten Navigationsziele nicht unter dem Begriff „Speicher“, weshalb sich bei dieser Aufgaben wieder ein schlechteres Fehlerverhältnis ergibt.

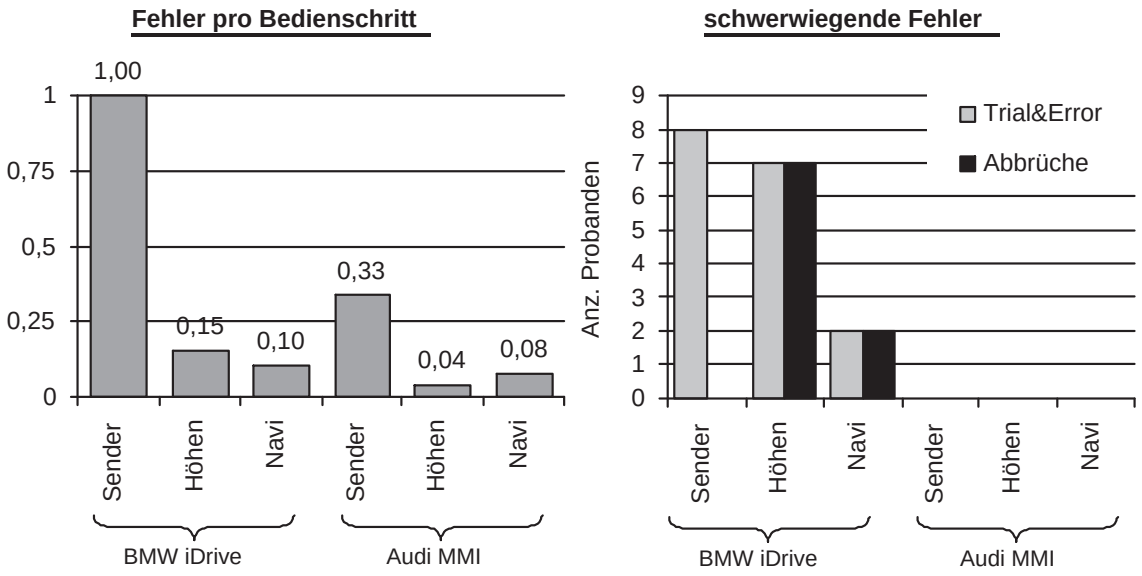

Abbildung 6.9: Verhältnis falscher zu korrekten Bedienschritte bei der Bedienung der FIS sowie Anzahl der schwerwiegenden Fehler (*Trial&Error*-Phasen und Abbrüche).

Beim BMW iDrive war die Bedienung des Hauptmenüs überwiegend problemlos möglich. Allerdings vermuteten 12 der 13 Probanden, dass ein Wechsel aus dem Radiomenü (am linken Displayrand) in die Senderliste (am rechten Displayrand) ebenfalls durch das Schieben des iDrive-Controllers nach rechts erreicht werden kann. Tatsächlich wechselt das System jedoch in das Navigationsmenü, was bei den Probanden zu starker Verwirrung führte. Lediglich 4 der 12 Probanden konnten das System auf Anhieb bedienen, 8 Probanden erkannten erst nach einer *Trial&Error*-Phase, dass ein Wechsel von einem Menü im linken Displaybereich in ein Menü im rechten Displaybereich lediglich durch Drehen des DDK erreicht werden kann.

In den folgenden Aufgaben bereitete die Bedienung des Controllers weniger Probleme, was sich ebenfalls in dem besseren Fehlerverhältnis ausdrückt (Abb. 6.9). Allerdings vermuteten bei der Bearbeitung der zweiten Aufgabe die meisten Probanden die Funktion zur Einstellung der Klanghöhen nicht unter dem Begriff „Memory“. Nach einer *Trial&Error*-Phase brachen 7 Probanden die Aufgabenbearbeitung (teilweise mit starken Aggressionen) ab (KURSCHEID 2005). Bei der Bearbeitung der Navigations-Aufgabe waren ebenfalls zwei Probanden nicht in der Lage, die Aufgabe zu beenden.

Die subjektive Meinung der Probanden wurde nach jeder Aufgabenbearbeitung durch einen Fragebogen mit semantischen Differenzialen ermittelt. Dabei geben Probanden auf Adjektivskalen an, wie sehr gegebene Aussagen über Geräteeigenschaften zutreffen. Detailliertere Erklärungen über semantische Differenziale finden sich zusammen mit den Ergebnissen der jeweiligen Aufgaben in Anhang F. In Abbildung 6.10 sind die Ergebnisse der jeweiligen Aufgaben vergleichend für die beiden betrachteten FIS dargestellt. Die Bewertung einer Eigenschaft ist von -3 (negativ) bis 3 (positiv) möglich. Die bewerteten Eigenschaften sind stichwortartig angegeben und finden sich detailliert in den einzelnen Diagrammen in Anhang F.

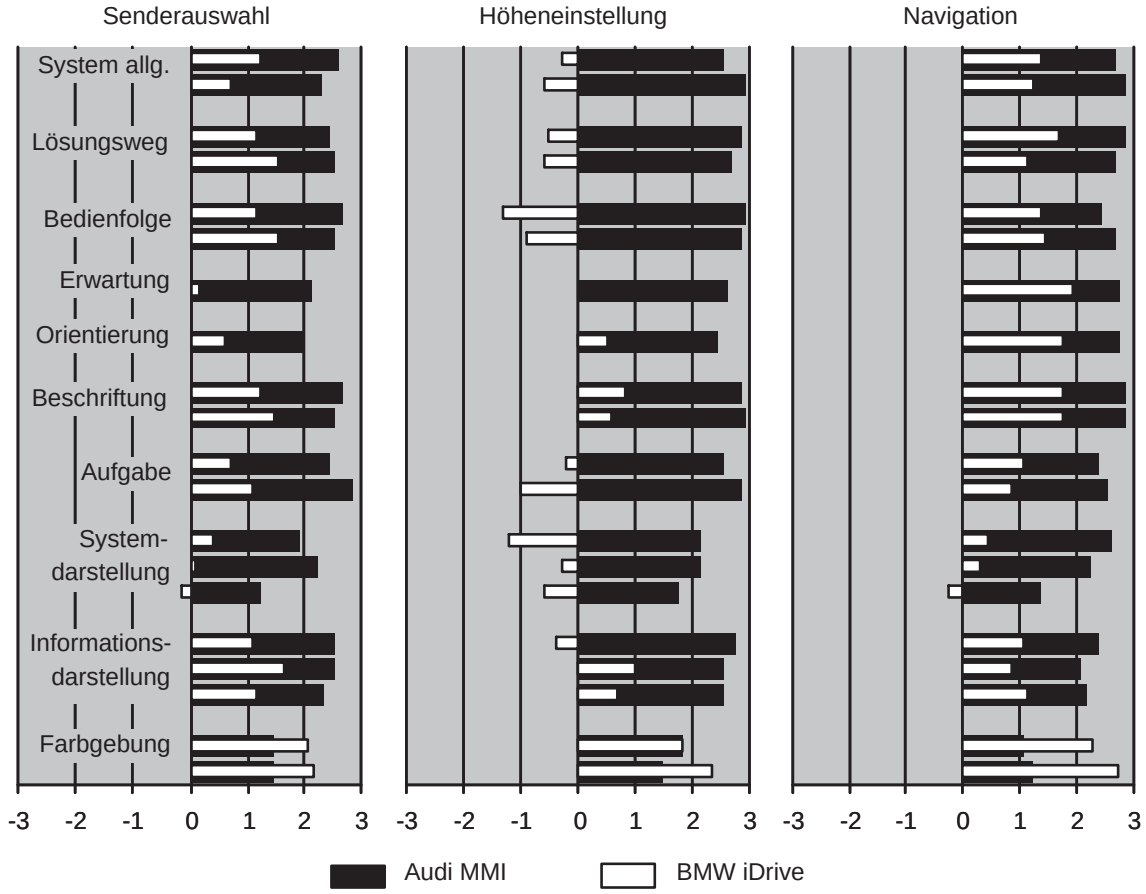

Abbildung 6.10: Die gemittelten Ergebnisse der semantischen Differenziale jeweils für das Audi MMI und das BMW iDrive.

Es wird beim Audi MMI durch die vollständig auf der positiven Seite liegenden Mittelwerte deutlich, dass von einer guten Akzeptanz der Informationsauswahl ausgegangen werden kann. Die Informationen wurden als übersichtlich, verständlich und hilfreich bewertet. Im Vergleich ist das Audi MMI in jeder Aufgabe und nahezu jeder Eigenschaft besser bewertet als das BMW iDrive. Lediglich in der Farbgebung (untersten zwei Balken) wird das BMW iDrive als besser empfunden. Die besonders schlechten Werte des BMW in der zweiten Aufgabe (Höheneinstellung) sind darin begründet, dass 7 Probanden nicht in der Lage waren, die Aufgabe zu beenden und das System entsprechend schlecht bewertet haben. In den beiden anderen Aufgaben des BMW iDrive (Senderauswahl und Navigation) geben die Probanden ebenfalls an, dass sie sich mehr Unterstützung vom System und Sicherheit bei der Bedienung gewünscht hätten.

Nach der Bearbeitung aller Aufgaben nahmen die Probanden eine vergleichende Bewertung hinsichtlich vorgegebener Kriterien vor. Dabei wurden Aspekte der Eindeutigkeit und Konsistenz der Informationsdarstellung, der Farbgebung und der Sicherheit der Bedienung aufgeführt. Für jedes einzelne Kriterium konnten die Probanden angeben, ob ihnen das Audi MMI oder das BMW iDrive besser gefallen hat (Enthaltungen waren erlaubt).

Über alle Punkte hinweg bewerteten durchschnittlich 11 der 13 Probanden das Audi MMI besser als das BMW iDrive (durchschnittlich 0.65 Stimmen, 1.35 Enthaltungen). Die Ergebnisse sind in Abbildung 6.11 dargestellt und zeigen deutlich, dass das Audi MMI durchgängig ein besseres Systemdesign bietet als das BMW iDrive. Lediglich in der Farbgestaltung erhielt der BMW mehr Stimmen, da der Audi MMI als zu bunt empfunden wurde.

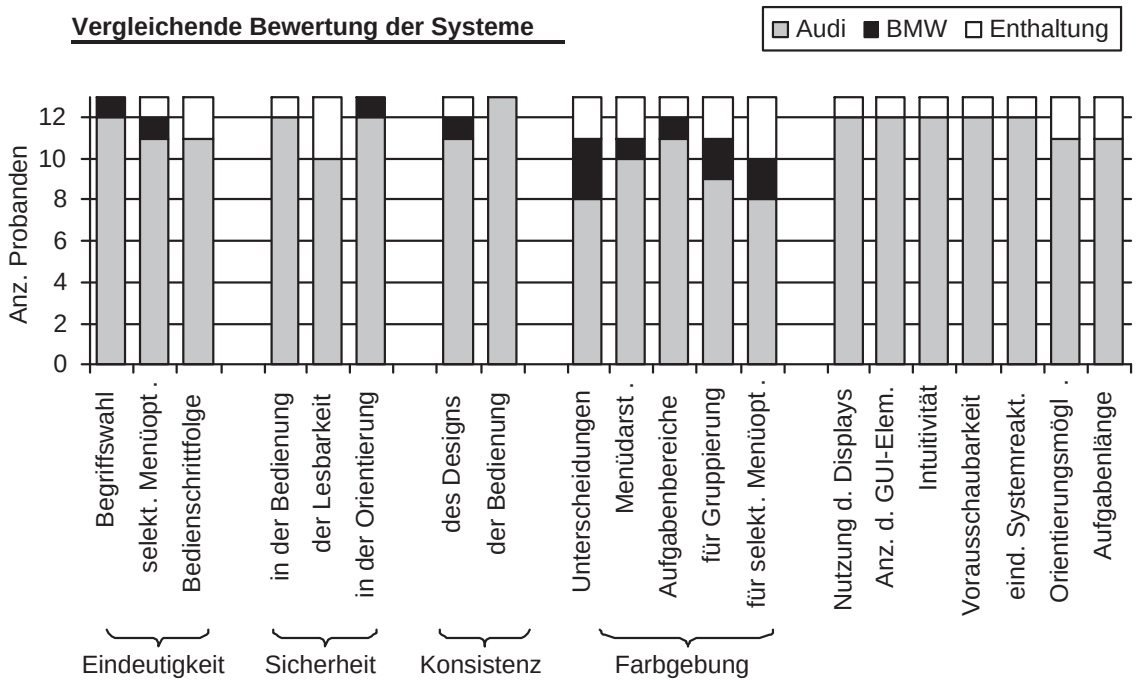

Abbildung 6.11: Ergebnis der vergleichende Bewertung aufgrund der subjektiven Meinung der Probanden.

Kapitel 7

Zusammenfassung und Ausblick

Immer kürzere Entwicklungszeiten für neue interaktive Systeme und steigender Wettbewerbsdruck führen dazu, dass entwicklungsbegleitende Werkzeuge zur Bewertung der Gebrauchstauglichkeit eine immer größere Bedeutung erlangen. Aufgrund der stetig wachsenden Anzahl von Vorgaben, z.B. in Form von Normen oder Styleguides, nimmt der Aufwand und die Fehler bei der von Experten manuell durchgeführten Berücksichtigung dieser Vorgaben zu. Eine automatische Überprüfung des Systemdesigns auf Konformität mit den Vorgaben stellt eine große Hilfe bei der Systementwicklung dar. Es existieren dazu einige Werkzeuge, die jedoch in Umfang und Art der Bewertung sehr beschränkt sind. Diese Werkzeuge bewerten dabei überwiegend Aspekte wie Lage oder Anordnung der GUI-Objekte im Rahmen des Oberflächendesigns. Wenige Werkzeuge generieren Aussagen über die im System benutzten Begriffe, während eine aussagekräftige Bewertung der Art der Bedienung sowie des erwartungskonformen Einsatzes unterschiedlicher Bedienelemente bislang von keinem Werkzeug geleistet wird.

Um diese Problematik zu beheben, wurde in dieser Arbeit das Werkzeug REVISER (Rapid EValuation of Interactive Systems using ExpeRt knowledge) konzipiert und realisiert, das die Bewertung aller Designbereiche (Gestaltung der Oberfläche bzw. GUI, Verwendung aussagekräftiger Begriffe, intuitive Interaktion durch erwartungskonforme Nutzung geeigneter Bedienelemente) eines interaktiven Systems durch eine Konformitätsprüfung bestehender Guidelines bietet. REVISER ist ein Expertensystem, das die Eingabe von Kriterien in Regelform ermöglicht. Als Anwendungsfelder sind menügestützte, interaktive Systeme des täglichen Gebrauchs, wie Mobiltelefone oder Fahrerinformationssysteme, zu sehen.

Zur Bewertung des Oberflächendesigns werden im Rahmen dieser Arbeit zahlreiche Maße identifiziert und vorgestellt, die Angaben über Farbgebung, Lage und Anordnung von GUI-Elementen, sowie Aussagen über die Konsistenz der Darstellung als Prädiktor der Zufriedenheit und Belastung der Benutzer liefern. Die Basis dieser Bewertung bildet eine formale Beschreibung des System-GUI. Um unabhängig von proprietären Beschreibungsformaten spezieller Spezifikationswerkzeuge zu sein, ist in dieser Arbeit ein allgemein gültiges GUI-Format entwickelt worden, in das sich jedes andere Beschreibungsformat konvertieren lässt.

Zur Bewertung der eingesetzten Begriffe (*Wording*) kommt eine Ontologie zum Einsatz, die die semantischen Beziehungen von Begriffen in Menüsystemen abbildet. Ein Vergleich der tatsächlich benutzten mit den empirisch ermittelten Begriffen der Ontologie gibt Aufschluss über die Güte der eingesetzten Begriffe sowie der Verwendung von Synonymen.

Die Bewertung der Bedienung und Benutzung von Bedienelementen im Rahmen des Interaktionsdesigns basiert auf einer Dialogfolge, die im Rahmen einer Aufgabenbearbeitung die GUI-Konstellationen der einzelnen Bedienschritte sowie die dazugehörigen Aktionen beinhaltet. Aus einer formalen Systembeschreibung ist die statische Generierung dieser Dialogfolge nicht möglich. Aus diesem Grund wurde im Rahmen dieser Arbeit ein autonomer Bedienagent entwickelt, der mit Hilfe von Vorwissen ein interaktives System bedienen und dadurch eine Dialogfolge erstellen kann. Der Agent analysiert in jedem Bedienschritt die dargestellten Informationen zur Erkennung des aktuellen Zustands des Systems. Durch die Ontologie, deren Struktur einem generischen Menübaum entspricht, ist er in der Lage, den Folgezustand zu erkennen. Anschließend muss der Agent die zum Erreichen des Folgezustands benötigte Bedienaktion auswählen. Dazu ist die Berücksichtigung gängiger Bedienstereotypen erforderlich, die eine erwartungskonforme Zuordnung von Benutzeraktionen (z.B. Druck einer Taste) und Systemreaktion (z.B. Aktivierung einer Funktion) beschreiben. Der Einsatz eines autonomen Bedienagenten zur Bewertung des Interaktionsdesigns ist erstmalig in dieser Arbeit beschrieben und umgesetzt worden.

Zur Demonstration der Funktions- und Leistungsfähigkeit des Werkzeugs erfolgt die Bewertung zweier Fahrerinformationssysteme (Audi MMI und BMW iDrive). Die Systeme unterscheiden sich in Art und Aufbau der Menüstruktur und Bedienelemente sowie in der Darstellung der Informationen und wurden im Rahmen dieser Arbeit originalgetreu nachgebildet. Bei der Bewertung identifizierte REVISER mehrere Fehler in den Designs beider Systeme. Im Systemdesign des BMW iDrive waren dabei zwei Fehler so gravierend, dass sie zu Aufgabenabbrüchen des Agenten führten. Ein Fehler basierte auf einer nicht erwartungskonformen Benutzung eines Bedienelements, der andere trat aufgrund eines missverständlichen Begriffs auf. Darüber hinaus konnte das Werkzeug viele Vorschläge und Hinweise zur Verbesserung beider Systeme liefern. Ein Vergleich der Bewertungsergebnisse zeigt das durchgehend bessere Design des Audi. Sowohl der Aufbau des GUI als auch der Einsatz der zur Verfügung stehenden Bedienelemente waren beim Audi MMI konsistenter und erwartungskonformer als beim BMW iDrive. Lediglich die Farbgebung des BMW iDrive wurde von REVISER besser bewertet.

Eine empirische Bewertung der beiden Fahrerinformationssysteme bestätigte nahezu alle von REVISER identifizierten Auffälligkeiten. Die Aufgabenbearbeitung der meisten Probanden am BMW iDrive scheiterte ebenfalls aus genau den gleichen Gründen wie der Bedienagent. Die von den Probanden erkannten Designfehler entsprachen ebenfalls den Ergebnissen von REVISER. Die durch Fragebögen ermittelte Meinung bzw. Zufriedenheit der Benutzer konnte mittels der in REVISER implementierten Regeln und Konsistenzmaße bis hin zu den Aussagen über die Farbgebung zuverlässig bestätigt werden. Die Ergebnisse einer von der Firma SirValUse durchgeführten empirischen Bewertung bestätigen ebenfalls die Ergebnisse der au-

tomatischen kriterienorientierten Bewertung durch REVISER.

Für die zukünftige Nutzung lässt sich das Werkzeugs im Wesentlichen in zweierlei Hinsicht erweitern. Die erste Erweiterungsmöglichkeit betrifft den Ausbau der Funktionalität. Zu nennen sind u.a. die Erweiterung der Regelbasis zur Bewertung des Systemdesigns. Eine Betrachtung des aktuellen Aufgabenkontextes zur Bewertung der benutzen Begriffe (im Rahmen des *Wordings*) erlaubt eine noch genauere Abschätzung der Benutzerakzeptanz.

Weiterhin ist eine Erweiterung der Bedienlogik sowie der Bedienstereotypen des autonomen Agenten zur Bedienung von Menüs mit Hilfe von Joysticks o.ä. (z.B. in modernen Mobiltelefonen) bzw. der Bedienung von Systemen mit Touchscreens sinnvoll. Durch die dynamische Verstärkung häufig benutzter Bedienstereotypen im Agenten wird der Lernprozess von Benutzern simuliert. Eine zusätzliche Komponente, die die Verstärkungen zeitabhängig zurücknimmt, ist in der Lage, ein *Vergessen* dieses Wissens realistisch nachzubilden.

Verbale Aussagen bzw. Guidelines sind zu großen Teilen nicht exakt und dadurch nur schwer mathematisch fassbar. Sachverhalte werden häufig durch Attribute wie „viel“ oder „wenig“ ausgedrückt. Daher ist eine Erweiterung der bestehenden Regelsprache um Fuzzy-Sets sinnvoll, die es erlauben, solche „unscharfen“ Aussagen in Regelform anzugeben. Dadurch lassen sich in Abhängigkeit der Schwere des Fehlers Alarme in unterschiedlichen Stufen generieren. Beispielsweise können so Maße zur Bewertung der Oberflächengestaltung, die die Lage der GUI-Elemente bislang pixelgenau bewerten, realistischer an das Empfinden der Benutzer angepasst werden, da diese meistens einen Unterschied von wenigen Pixeln nicht wahrnehmen.

Da interaktive Systeme in zunehmenden Maße Icons und Grafiken zur Navigation benutzen, stellt die Bewertung und Erkennung solcher Grafiken ebenfalls eine vielversprechende Erweiterung des Werkzeugs dar. Über ein Bildverarbeitungsmodul können Grafiken erkannt und anschließend bewertet werden. Analog zur Bewertung von Begriffen bietet sich dazu der Einsatz einer Ontologie an, die eine Beschreibung gängiger Icons enthält. Mit Hilfe weiterer Bildverarbeitungsalgorithmen wäre eine Bewertung komplexerer GUIs möglich, die verstärkt mit Überblendeffekten und zusammengesetzten Dialogen arbeiten. Kontinuierliche Veränderungen auf dem GUI, z.B. auf Grund von Animationseffekten oder kontinuierlicher Bedienung, lassen sich dabei in Form eines Films aufnehmen und Veränderungen durch einen Vergleich der einzelnen Frames erkennen und bewerten.

Die zweite Erweiterungsmöglichkeit berücksichtigt die Tatsache, dass weitere Werkzeuge zur Spezifikation und Erstellung interaktiver Systeme existieren. Durch den Einsatz eines allgemein gültigen GUI-Formats innerhalb von REVISER lässt sich das Werkzeug an beliebige Spezifikations- und Prototyping-Werkzeuge anbinden. So ist eine permanente Überprüfung des entwickelten interaktiven Systems automatisch während der gesamten Systementwicklung möglich. Dabei erlaubt REVISER die Angabe der Regeln domänenunabhängig, so dass ein großer Teil des Regelwissens auch für unterschiedliche interaktive Systeme gültig bleibt und wiederverwendet werden kann. Ein Modul zur Verwaltung solcher Regelbibliotheken könnte somit die Erstellung und Verteilung auch großer Regelbasen unterstützen.

Die Berücksichtigung der zahlreichen existierenden Guidelines während der Systementwick-

lung ist eine nicht zu unterschätzende Aufgabe, die bisher überwiegend manuell durchgeführt werden musste. Daher stellt die Implementierung automatisch messbarer Kriterien in einem Bewertungswerkzeug und die Integration dieses Werkzeugs in den Entwicklungsprozess eine große Hilfe dar, um einerseits die Benutzungsfreundlichkeit der interaktiven Systeme zu erhöhen und andererseits den zeitlichen und finanziellen Aufwand für die Systementwicklung zu reduzieren. Das entwickelte Werkzeug REVISER bildet einen fundierten Rahmen, um dieser Problematik gerecht zu werden und ist das erste Werkzeug, das eine automatische Bewertung aller Bereiche des Systemdesigns bietet.

Literaturverzeichnis

AAM (2003). *Statement of Principles, Criteria and Verification Procedures on Driver Interactions with Advanced In-Vehicle Information and Communication Systems*. Alliance of Automobile Manufacturers, Washington, D.C.

ABDULKHAIR, M. (2004). *A Multilingual Automated Web Usability Evaluation Agent*. Doktorarbeit, Department of Computer Science – University of Sheffield.

AHLERS, J. und P. OEL (2005). *Design von Sprachbediensystemen*. automotive, 07-08:36–38. Carl Hanser Verlag, München.

ALBERS, J. (1997). *Interaction of Color – Grundlegung einer Didaktik des Sehens*. DuMont Verlag, Köln.

ALBERT, L. (1989). *Average Case Complexity Analysis of RETE Pattern-Match Algorithm and Average Size of Join in Databases*. Technischer Bericht, Institut National de Recherche en Informatique et en Automatique (INRIA).

ALLENSBACHER BERICHTE (2002). *Handy - hat bald jeder eines?*. Institut für Demoskopie, Allensbach.

AMANT, R. ST. und A. B. WOOD (2005). *Tool Use for Autonomous Agents*. In: *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, S. 184–189.

ANDERSON, K. und U. LOTTO (2002). *A Communication Model for Voice Control of Service Gateways*. Diplomarbeit, Göteborg University.

APPLE (2005). *Human Interface Guidelines*. Apple Computer Inc.

AVOURIS, N.M. (2001). *An Introduction to Software Usability*. In: *Proc. 8th Panhellenic Conference on Informatics*, Bd. 2, S. 514–522. Livanis Publ., Athens.

BARWISE, J. und J. ETCHEMENDY (1992). *The Language of First-Order Logic*. Stanford University Center for the Study.

BAUMGARTNER, N. (2004). *Der Effekt von Polarität der Textdarstellung und Umgebungsbeleuchtung auf die Performanz und psychophysiologische Beanspruchung bei Bildschirmarbeit*. Doktorarbeit, Heinrich-Heine Universität, Düsseldorf.

BEIREKDAR, A. (2004). *A Methodology for Automating Guideline Review of Web Sites*. Doktorarbeit, Institut d'Informatique, Namur.

BEIREKDAR, A., M. KEITA, M. NOIRHOMME, F. RANDOLET, J. VANDERDONCKT und C. MARIAGE (2005). *Flexible Reporting for Automated Usability and Accessibility Evaluation of Web Sites*. In: COSTABILE, M. F. und F. PATERNÒ., Hrsg.: *Tenth IFIP TC13 International Conference on Human-Computer Interaction*, Lecture Notes in Computer Science. Springer Verlag, Berlin.

BEIREKDAR, A., J. VANDERDONCKT und M. NOIRHOMME-FRAITURE (2002). *A Framework and Language for Usability Automatic Evaluation of Websites by static Analysis of HTML Sourcecode*. In: *4th International Conference on Computer-Aided Design of User*, S. 337–348. Kluwer Academics Publishers, Dordrecht.

BEIREKDAR, A., J. VANDERDONCKT und M. NOIRHOMME-FRAITURE (2003). *Kwaresmi – Knowledge-based Web Automated Evaluation with Reconfigurable Guidelines Optimization*. In: STEPHANIDIS, C., Hrsg.: *2nd International Cenference on Universal Access in Human-Computer Interaction*, Bd. 4, S. 1504–1508. Lawrence Erlbaum Associates, Mahwah.

BENGLER, K., M. HERRLER und H. KÜNZNER (2002). *Usability Engineering bei der Entwicklung von iDrive*. it + ti Informationstechnik und Technische Informatik, 44(3):145–152.

BENYON, D., P. TURNER und S. TURNER (2005). *Designing Interactive System*. Addison-Wesley Professional, Boston.

BERSTEL, J., S. REGHIZZI, G. ROUSSEL und P. S. PIETRO (2005). *A Scalable Formal Method for Design and Automatic Checking of User Interfaces*. In: *ACM Transactions on Software Engineering and Methodology (Tosem)*, Bd. 14, S. 124–167. ACM Press, New York.

BISCHOFF, J., E. COCHLOVIUS und M. T. TRAN (2002). *Flexible Software-Architekturen für Embedded Multimediasysteme*. Elektronik Automotive, 2:63–69.

BLACK, J. (2002). *Usability Is Next to Profitability*. BusinessWeek-Online. http://www.businessweek.com/technology/content/dec2002/tc2002124_2181.htm.

BLACKWELL, O.M. und H. BLACKWELL (1971). *Visual Performance Data for 156 Normal Observers of Various Ages*. Journal of the Illuminating Engineering Society, 61:3–13.

BLEICH, H. (2005). *Web-Fehler*. c't - Magazin für Computertechnik, 9:92–93.

BLUM, E.J. und G. NIRSCHL (2000). *MMI-Prüfliste – Verfahren und Werkzeug zur Bewertung von Mensch-Maschine-Systemen im Kraftfahrzeug*. In: TIMPE, K.-P., H.-P. WILLUMEIT und H. KOLREP, Hrsg.: *Bewertung von Mensch-Maschine-Systemen - 3. Berliner Werkstatt*, Bd. 22 d. Reihe *Fortschritt-Berichte*, S. 139–150. VDI Verlag, Düsseldorf.

BLUME, M., D. STOKAR und F. SEEWALD (2005). *Usability-Evaluation: Egal wer's macht? Ein schweizer Fallbeispiel*. i-com, 3:69–71.

BONATO, M. (1990). *Wissensstrukturierung mittels Struktur-Lege-Techniken - Eine graphentheoretische Analyse von Wissensnetzen*. Peter Lang Verlag, Frankfurt am Main.

BONNER, J. V. H. (1997). *Developing Advanced Interface Design Guidelines from Survey Based and Empirical Research*. In: *International Conference on Design and Technology Educational Research and Curriculum Development*. Loughborough University of Technology, Department of Design & Technology.

BORCHERS, J. (2001). *A Pattern Approach to Interaction Design*. John Wiley & Sons.

BOWEN, J. A. (2005). *Formal Specification of User Interface Design Guidelines*. Doktorarbeit, University of Waikato, New Zealand.

BRAJNIK, G. (2000). *Automatic Web Usability Evaluation: What Needs to be Done?*. In: *Proc. Human Factors and the Web, 6th Conference*, Austin - Texas.

BRONSTEIN, I.N. und K. SEMENDJAJEW (1991). *Taschenbuch der Mathematik*. B.G. Teubner Verlagsgesellschaft, Stuttgart.

BUDANITSKY, A. und G. HIRST (2001). *Semantic Distance in WordNet: An Experimental, Application-Oriented Evaluation of Five Measures*. In: *Workshop on WordNet and Other Lexical Resources, Second Meeting of the North American Chapter of the Association for Computational Linguistics*, Pittsburgh.

BULLINGER, H.-J. und G. WASSERLOOS (1989). *Die Entwicklung praxisgerechter Expertensysteme*. Moderne Industrie Verlag, Landsberg.

BURMESTER, M. (2001). *Optimierung der Erlern- und Benutzbarkeit von Benutzungsschnittstellen interaktiver Hausgeräte auf der Basis der speziellen Anforderung älterer Menschen*. Nr. 664 in *Fortschritts-Berichte*. VDI Verlag, Düsseldorf.

BURNS, J. T. (2004). *Evaluation*. De Montfort University, Bedford. Lecture Notes.

CARD, S.K., T. MORAN und A. NEWELL (1983). *The Psychology of Human-Computer Interaction*. Lawrence Erlbaum Associates, New Jersey, Hillsdale, NJ, USA.

CAWSEY, A. (2003). *Künstliche Intelligenz im Klartext*. Pearson Studium.

CLAUS, V. und A. SCHWILL (2003). *Duden Informatik – Ein Fachlexikon für Studium und Praxis*. Dudenverlag, Mannheim.

COLMERAUER, A., H. KANOUI, R. PASERO und P. ROUSSEL (1973). *Un Système de communication Homme-Machine en Francais*. Technischer Bericht, Groupe Intelligence Artificielle – Université d'Aix-Marseille II.

COMMISSION INTERNATIONALE DE L'ECLAIRAGE (2004). *Colorimetry*. CIE 15:2004.

CORDES, R. E. (1982). *Software Guideline Development: Proposed Methodology*. In: *Proceedings of the Conference on Human Factors in Computing Systems*, S. 82–84. ACM Press, New York.

COSTABILE, M. F. und M. MATERA (2001). *Proposing Guidelines for Usability Inspection*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 283–292. Springer Verlag, London.

COULSON, T., C. SHAYO, L. OLFMAN und C. E. T. ROHM (2003). *ERP Training Strategies: Conceptual Training and the Formation of Accurate Mental Models*. In: *Proceedings of the 2003 SIGMIS conference on Computer personnel research: Freedom in Philadelphia–leveraging differences and diversity in the IT workforce*, S. 87–97, Philadelphia.

COUTAZ, J., D. SALBER und S. BALBO (1993). *Towards Automatic Evaluation of Multimodal User Interfaces*. In: *Proceedings of the 1st international conference on Intelligent user interfaces*, S. 201–208. ACM Press, New York.

DAHM, M., C. FELKEN, M. KLEIN-BÖSING, G. ROMPEL und R. STROICK (2005). *Zur Gebrauchstauglichkeit von Handys - Breite Untersuchung und aktueller Stand*. i-com, 1:26–33.

DATECH (2004). *DATech-Prüfhandbuch Gebrauchstauglichkeit – Leitfaden für die ergonomische Evaluierung von Software auf Grundlage von DIN EN ISO 9241, Teile 10 und 11*. DATech – Deutsche Akkreditierungsstelle Technik GmbH, Frankfurt. Version 3.3.

DEPARTMENT OF DEFENSE (1986). *Military Specification Displays, Airborne, Electronically/Optically Generated*. Technischer Bericht MIL-D-87213A, U.S. Government Printing Office, Washington, DC.

DEPARTMENT OF DEFENSE (1999). *Design Criteria Standard - Human Engineering*. Technischer Bericht MIL-STD-1472F, U.S. Army Aviation and Missile Command, Redstone.

DEPARTMENT OF TRANSPORTATION (1998). *Human Factors Design Guidelines*. US Department of Transportation, Washinton D.C.

EBBINHAUS, H.-D., J. FLUM und W. THOMAS (1992). *Einführung in die Mathematische Logik*. BI-Wissenschafts-Verlag, Mannheim.

EBERLEH, E. (1989). *Beschreibung, Klassifikation und mentale Repräsentation komplexer Mensch-Computer-Interaktionen*. S. Roderer, Regensburg.

EHRLENSPIEL, K. (2002). *Integrierte Produktentwicklung*. Hanser Fachbuchverlag, München.

EUROPÄISCHE KOMMISSION (2000). *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*. Dokument Nr. C(1999) 4786.

FARENC, C. (1997). *ERGOVAL: une Méthode de Structuration des Règles Ergonomiques Permettant l'Évaluation Automatique d'Interfaces Graphiques*. Doktorarbeit, Université Toulouse.

FARENC, C. und P. PALANQUE (1999). *A Generic Framework based on Ergonomics Rules for Computer Aided Design of User Interface*. In: PUERTA, A., Hrsg.: *Computer-Aided Design of User Interfaces*, Dordrecht. Kluwer Academics.

FAULK, S.R. (1997). *Software Requirements: A Tutorial*. In: THAYER, R. und M. DORFMAN, Hrsg.: *Software Requirements Engineering*, Bd. 2. IEEE Computer Society Press, New Jersey.

FENSEL, D. (2004). *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer Verlag, Berlin Heidelberg, 2 Aufl.

FISCHER, A. R. H. (1999). *Intuitive Interfaces: A Literature Review of the Natural Mapping Principle and Stimulus Response Compatibility*. Technischer Bericht, Technische Universität Eindhoven.

FITTS, P.M. (1951). *Human Engineering for an Effective Air Navigation and Traffic Control System*. Technischer Bericht, Ohio State University, Columbus, OH, USA.

FITTS, P.M. (1954). *The Information Capacity of the Human Motor System in Controlling the Amplitude of Movement*. Journal of Experimental Psychology, 47:381–391.

FITZPATRICK, R. (1999). *Strategies for Evaluating Software Usability*. Technischer Bericht, Department of Mathematics, Statistics and Computer Science, Dublin Institute of Technology.

FOLEY, J. D., V. L. WALLACE und C. PEGGY (1984). *The Human Factors of Computer Graphics Interaction Techniques*. IEEE Computer Graphics & Applications, 11:13–48.

FORGY, C. (1982). *Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem*. Artificial Intelligence, 19:17–37. Elsevier.

FOWLER, M. und K. SCOTT (2000). *UML Distilled – A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley Professional, Boston, 2 Aufl.

FREISE, A. (2003). *Der Nutzen gut bedienbarer Produkte*. Siemens – Pictures of the Future, 2:65. Siemens AG, München.

FRIEDMAN-HILL, E. (2003). *Jess in Action*. Manning Publications, Greenwich.

GAT, E. (2000). *Point of View: Lisp as an Alternative to Java*. Intelligence, 11(4):21–24. ACM Press, New York.

GEDIGA, G. und K.-C. HAMBORG (2002). *Ergonomische Evaluation von Software: Methoden und Modelle im Software-Entwicklungsprozess*. Zeitschrift für Psychologie, 1:40–57.

GEISER, G. (1998). *Mensch-Maschine-Kommunikation: Warum entstehen „benutzungsfeindliche“ Geräte?*. ITG-Fachbericht: Technik für den Menschen – Gestaltung und Einsatz benutzerfreundlicher Produkte, 154:7–14.

GIERSICH, M., T. HEIDER und T. KIRSTE (2005). *Multimodale Gerätekooperation basierend auf expliziten Zielen: Konzepte & Potentiale*. i-com, 3:15–21.

GOLDSTEIN, E. B. (2002). *Wahrnehmungspsychologie*. Spektrum Akademischer Verlag, 2 Aufl.

GOMAA, M., A. SALAH und S. RAHMAN (2005). *Towards A Better Model Based User Interface Development Environment: A Comprehensive Survey*. In: *Proceedings of the 38th Annual Midwest Instruction and Computing Symposium*.

GOTTLOB, G., T. FRÜHWIRTH und W. HORN (1990). *Expertensysteme*. Springer Verlag, Wien.

GRAF, N., E. WETZENSTEIN und H. WANDKE (2004). *GUIDEAS - Meta-Assistenz zur Entwicklung benutzerzentrierter Assistenzfunktionen*. Useware 2004: Nutzergerechte Gestaltung Technischer Systeme - VDI-Berichte, 1873:85–92.

GRAMMENOS, D., D. AKOUMIANAKIS und C. STEPHANIDIS (2000). *Integrated Support for Working with Guidelines: the Sherlock Guideline Management System*. Interacting with Computers, 12(3):281–311.

GRAMMENOS, D., D. AKOUMIANAKIS und C. STEPHANIDIS (2001). *Sherlock: A Tool Towards Computer-Aided Usability Inspection*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 87–98. Springer Verlag, London.

GREEN, P., W. LEVISON, G. PAELKE und C. SERAFIN (1995). *Preliminary Human Factors Design Guidelines for Driver Information Systems*. Technischer Bericht FHWA-RD-94-087, U.S. Department of Transportation.

GRUBER, T.R. (1993). *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*. Technischer Bericht KSL-93-04, Knowledge Systems Laboratory, Stanford University.

GRUDIN, J. (1989). *The Case against User Interface Consistency*. Communications of the ACM, 32(10):1164–1173. ACM Press, New York.

GÖRZ, G., C.-R. ROLLINGER und J. SCHNEEBERGER (2003). *Handbuch der Künstlichen Intelligenz*. Oldenbourg Verlag, München.

GÖTZE, R. (1994). *Dialogmodellierung für multimediale Benutzerschnittstellen*. Doktorarbeit, Universität Oldenburg.

HAMACHER, N. (2001). *Combining Formal and Empirical Evaluation Methods Regarding the Usability of Interactive Systems*. ERCIM NEWS, 46:38–39.

HAMACHER, N. (2003). *TREVIS ein Werkzeug für die formale Bewertung von integrierten Mensch-Maschine-Schnittstellen*. In: SPATH, D., Hrsg.: *Vehicle Interaction Summit. Interaktive Systeme im Fahrzeug*. Fraunhofer- Gesellschaft, Stuttgart.

HAMACHER, N. und M. HÄHNEL (2002). *Formale Bewertung unterschiedlicher Fahrer-Informationssysteme*. In: *IIR-Fachkonferenz*, Bd. C2336. IIR GmbH München, Nürtingen.

HAMACHER, N. und K.-F. KRAISS (2004). *Expertensystem zur kriterienorientierten Bewertung der Gebrauchsfähigkeit von Dialogsystemen*. In: *VDE-Kongress „Innovationen für den Menschen“*, Bd. 1, S. 193–198, Berlin. VDE, VDE Verlag GmbH.

HAMACHER, N., K.-F. KRAISS und J. MARRENBACH (2002). *Einsatz formaler Methoden zur Evaluierung der Gebrauchsfähigkeit interaktiver Geräte*. it + ti Informationstechnik und Technische Informatik, 44(1):49–55.

HAMACHER, N. und J. MARRENBACH (2001). *Analytical Evaluation of Interactive Systems regarding the Ease of Use*. In: STEPHANIDIS, C., Hrsg.: *Proceedings of HCI International 2001. Universal Access in HCI. Towards an Information Society for All*, Bd. 3, S. 585–589, New Orleans. Lawrence Erlbaum Associates.

HAREL, D. (1987). *Statecharts: A Visual Formalism for Complex Systems*. In: SINTZOFF, M., Hrsg.: *Science of Computer Programming*, S. 231–274. North-Holland, Amsterdam, The Netherlands.

HEGNER, M. (2003). *Methoden zur Evaluation von Software*. Technischer Bericht 29, Informationszentrum Sozialwissenschaften der Arbeitsgemeinschaft Sozialwissenschaftlicher Institute e.V., Bonn.

HENNINGER, S. (2001). *A Methodology and Tools for Applying Context-Specific Usability Guidelines to Interface Design*. Interacting with Computers, 12(3):225–243.

HENNINGER, S., K. HAYNES und M. W. REITH (1995). *A Framework for Developing Experience-Based Usability Guidelines*. In: *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, & Techniques*, S. 43–53. ACM Press, New York.

HIX, D. und R. HARTSON (1993). *Developing User Interfaces, Ensuring Usability Through Product & Process*. John Wiley & Sons, New York.

HOFFMANN, E. R. (1997). *Strength of component principles determining direction of turn stereotypes-linear displays with rotary controls*. Ergonomics, 40(2):199–222.

HOFSTÄTTER, P. R. und D. WENDT (1974). *Quantitative Methoden der Psychologie I*. Springer Verlag, Heidelberg.

HOLLNAGEL, E. (1989). *The Reliability of Expert Systems*. Ellis Horwood Ltd., Chichester.

HOLYER, A. (1993). *Methods for Evaluating User Interfaces*. Technischer Bericht, School of Cognitive and Computing Sciences, University of Sussex.

HOLZ AUF DER HEIDE, B. (1993). *Welche software-ergonomischen Evaluationsverfahren können was leisten?*. In: RÖDIGER, K.-H., Hrsg.: *Software-Ergonomie '93, Von der Benutzeroberfläche zur Arbeitsgestaltung: Fachtagung des German Chapter of the ACM*, S. 157–171. B. G. Teubner, Wiesbaden.

HOLZINGER, A. (2005). *Usability Engineering Methods for Software Developers*. Communications of the ACM, 48(1):71–74. ACM Press, New York.

HORROCKS, I. (1999). *Constructing the User Interface with Statecharts*. Addison-Wesley, Harlow.

HOWARD, S. und D. MURRAY (1987). *A Taxonomy of Evaluation Techniques for HCI*. In: SHACKEL, H. J. BULLINGERAND, Hrsg.: *Proceedings of INTERACT 87*, S. 453–459. IFIP, Elsevier Science Publishers B.V. (North-Holland).

HOYOS, C. G. (1974). *Kompatibilität*. In: SCHMIDTKE, H., Hrsg.: *Ergonomie II. Gestaltung von Arbeitsplatz und Arbeitsumwelt*, S. 93–112. Hanser Verlag, München.

HUHNS, M. N. und M. P. SINGH (1997). *Ontologies for Agents*. IEEE Internet Computing, 1(6):81–83.

HUTCHINS, E. (1989). *Methaphors for interface design.*. In: TAYLOR, M. M., F. NEEL und D. G. BOUWHUIS, Hrsg.: *The Structure of Multimodal Dialoque*, S. 11–28. Elsevier Science B.V., Amsterdam.

IBM (1992). *Object-Oriented Interface Design - IBM Common User Access Guidelines*. Que Corporation, Carmel, 1 Aufl.

ISO 15005 (2003). *Ergonomische Aspekte von Fahrerinformations und -assistenzsystemen – Grundsätze und Prüfverfahren des Dialogmanagements*. Deutsches Institut für Normung e.V.

ISO 15008 (2003). *Ergonomische Aspekte von Fahrerinformations- und Assistenzsystemen – Anforderungen und Bewertungsmethoden der visuellen Informationsdarstellung im Fahrzeug*. Deutsches Institut für Normung e.V.

ISO 4040 (1986). *Personenkraftwagen – Anordnung der Handbedienteile, Anzeige- und Kontrollgeräte*. Deutsches Institut für Normung e.V.

ISO 9241 (1997). *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 1-17*. Deutsches Institut für Normung e.V.

ISO 9241-10 (1996). *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 10: Grundsätze der Dialoggestaltung*. Deutsches Institut für Normung e.V.

ISO 9241-11 (1998). *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 11: Anforderungen an die Gebrauchstauglichkeit*. Deutsches Institut für Normung e.V.

ISO 9241-110 (2004). *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 110: Grundsätze der Dialoggestaltung*. Deutsches Institut für Normung e.V.

ISO 9241-12 (1998). *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 12: Informationsdarstellung*. Deutsches Institut für Normung e.V.

ISO9126 (2003). *Software Engineering – Product Quality (Teil 1-4)*. International Organization for Standardization.

IVORY, M. und A. CHEVALIER (2002). *A Study of Automated Web Site Evaluation Tools*. Technischer Bericht UW-CSE-02-10-01, University of Washington.

IVORY, M., M. HEARST und S. RASHMI (2001). *Empirically Validated Web Page Design Metrics*. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, S. 53–60. ACM Press, New York.

IVORY, M. Y. (2001). *An Empirical Foundation for Automated Web Interface Evaluation*. Doktorarbeit, Computer Science Division, UC Berkeley.

IVORY, M. Y. und M. A. HEARST (2001). *The State of the Art in Automating Usability Evaluation of User Interfaces*. ACM Computing Surveys, 33(4):470–516.

JACKO, J. A., G. SALVENDY und R. J. KOUBEK (1995). *Modelling of Menu Design in Computerized Work*. Interacting with Computers, 7:304–330.

JAHN, G., A. OEHME, D. RÖSLER und J. F. KREMS (2004). *Kompetenzerwerb im Umgang mit Fahrerinformationssystemen*. Nr. 47 in *Berichte der Bundesanstalt für Straßenwesen*. Wirtschaftsverlag NW, Bremerhaven.

JAMA (2004). *Guideline for In-Vehicle Display Systems – Version 3*. Japan Automobile Manufactures Association, Tokyo.

JONES, T.O. und W. SASSER (1995). *Why Satisfied Customers Defect?*. Harvard Business Report, 6(1):88–99.

KAINDL, H., B. LUTZ und P. TIPPOLD (1998). *Methodik der Softwareentwicklung: Vorgehensmodell und State of the Art der professionellen Praxis*. Vieweg Verlag, Wiesbaden.

KARAT, C.-M. (1994). *A Comparison of User Interface Evaluation Methods*. In: NIELSEN, J. und R. L. MACK, Hrsg.: *Usability Inspection Methods*, S. 203–233. John Wiley & Sons.

KIERAS, D. (2004). *GOMS Models for Task Analysis*. In: DIAPER, D. und N. STANTON, Hrsg.: *The Handbook of Task Analysis for Human-Computer Interaction*, S. 83–116. Lawrence Erlbaum Associates.

KIERAS, D. E., S. D. WOOD und D. E. MEYER (1997). *Predictive Engineering Models Based on the EPIC Architecture for a Multimodal High-Performance Human-Computer Interaction Task*. ACM Transactions on Computer-Human Interaction, 4(3):230–275.

KIERAS, D.E. (1999). *A Guide to GOMS Model Usability Evaluation Using GOMSL and GLEAN3*. Technischer Bericht, Electrical Engineering and Computer Science Department, University of Michigan.

KILEY, D. (2004). *Driven: Inside BMW, the Most Admired Car Company in the World*. John Wiley and Sons.

KIM, W. und J. FOLEY (1993). *Providing High-level Control and Expert Assistance in the User Interface Presentation Design*. In: *Proceedings of Interchi'93*, S. 430–437. ACM Press, New York.

KNAUTH, P. (2004). *Steuerstandgestaltung*. Institut für Industriebetriebslehre und Industrielle Produktion (IIP), Universität Kaiserslautern.

KOSTKA, M., K. DAHMEN-ZIMMER, I. SCHEUFLER, W. PIECHULLA und A. ZIMMER (2001). *Fahrfehlerbeobachtung durch Experten: Eine Methode zur Evaluierung von Belastung und Ablenkung des Fahrers.*. In: ZIMMER, A.C. und K. LANGE, Hrsg.: *Experimentelle Psychologie im Spannungsfeld von Grundlagenforschung und Anwendung*. Universitätsbibliothek, Regensburg.

KÖPPEN, N., J. NITSCHKE, H. WANDKE und M. VAN BALLEGOOY (2001). *Guidelines for Developing Assistive Components for Information Appliances – Developing a Framework for a Process Model*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 67–76. Springer Verlag, London.

KRAFTFAHRT-BUNDESAMT (2006). *Neuzulassungen von Personenkraftwagen nach Segmenten und Modellreihen im Dezember 2005*. Technischer Bericht, Flensburg.

KRAISS, K.-F. (1995). *Modellierung von Mensch-Maschine Systemen*. In: WILLUMEIT, H.-P. und H. KOLREP, Hrsg.: *ZMMS-Spektrum, Band 1, Verlässlichkeit von Mensch-Maschine-Systemen*, S. 15–35. Pro Universitate Verlag, Berlin.

KRAISS, K.-F. (2005). *Mensch-Maschine Systeme I*. Lehrstuhl für Technische Informatik, RWTH Aachen. Skript zur Vorlesung.

KURBEL, KARL (1992). *Entwicklung und Einsatz von Expertensystemen – Eine anwendungsorientierte Einführung in wissensbasierte Systeme*, Bd. 2. Springer Verlag.

KURSCHEID, J. (2005). *Entwicklung und Implementierung eines autonomen Agenten zur effizienten Bewertung von interaktiven Systemen*. Diplomarbeit, RWTH Aachen.

LACY, L. W. (2005). *Owl: Representing Information Using The Web Ontology Language*. Trafford Publishing, Oxford.

LANC, O. (1975). *Ergonomie - Psychologie der technischen Welt*. Verlag Kohlhammer, Stuttgart.

LANDAUER, T. K. (1988). *Research Methods in Human-Computer Interaction*. In: HELANDER, M., Hrsg.: *Handbook of Human-Computer Interaction*, S. 905–929. North-Holland, Amsterdam.

LIBUDA, L. und N. HAMACHER (2005). *Komfort als Kriterium für die Systementwicklung*. In: GRANDT, M. und A. BAUCH, Hrsg.: *47 Fachausschusssitzung Anthropotechnik - Komfort als Entwicklungskriterium in der Systemgestaltung*, Bd. DGL-Bericht 2005-05, S. 19–39, Wolfsburg. DGLR e.V., Bonn.

LICHTER, HORST (2005). *Objektorientierte Softwarekonstruktion: Modellierung von Begriffen*. Lehr- und Forschungsgebiet Informatik III, RWTH Aachen. Unterlagen zur Vorlesung.

LIMBOURG, Q. und J. VANDERDONCKT (2001). *Completing Human Factor Guidelines by Interactive Examples*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 357–373. Springer Verlag, London.

LINDGAARD, G., B. TSUJI und S. KHAN (2005). *Ecological Validity and Behavioural Measures in the Usability Testing of New Applications: A Workshop in Reality Testing*. In: *Proceedings of the 19th British HCI Group Annual Conference*, Edinburgh. Springer Verlag.

LOER, K. und M. HARRISON (2004). *Analysing and Modelling Context in Mobile Systems to Support Design*. Technischer Bericht CS-TR-876, School of Computing Science, University of Newcastle upon Tyne.

LOWE, D. B., A. J. BUCKNELL und R. G. WEBBY (1999). *Improving Hypermedia Development: a Reference Model-Based Process Assessment Method*. In: *HYPERTEXT '99: Proceedings of the tenth ACM Conference on Hypertext and Hypermedia : Returning to our Diverse Roots*, S. 139–146. ACM Press, New York.

LUCKHARDT, H.-D. (2005). *Virtuelles Handbuch Informationswissenschaft - Kriterien für das Webpublishing*. Universität des Saarlandes.

LUGER, G. (2001). *Künstliche Intelligenz – Strategien zur Lösung komplexer Probleme*. Pearson Studium.

LÖWGREN, J., U. MALINOWSKI und K. NAKAKOJI (1996). *Knowledge-Based Support for the User-Interface Design Process - A CHI '95 Workshop*. ACM SIGCHI Bulletin, 28(1):43–47.

LÖWGREN, J. und T. NORDQVIST (1992). *Knowledge-based Evaluation as Design Support for Graphical User Interfaces*. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, S. 181–188, Monterey, California.

LÖWGREN, J. und E. STOLTERMAN (1999). *Methods & Tools: Design Methodology and Design Practice*. Interactions, 6(1):13–20. ACM Press, New York.

MACLEOD, D. (2005). *Cognitive Ergonomics - Making Sense with Design*. Technischer Bericht, Society for Work Science of the Institute of Industrial Engineers, Norcross.

MAGKANARAKI, A., G. KARVOUNARAKIS, T. T. ANH, V. CHRISTOPHIDES und D. PLEXOUSAKIS (2002). *Ontology Storage and Querying*. Technischer Bericht 308, Foundation for Research and Technology Hellas Institute of Computer Science Information Systems Laboratory, Hellas.

MAHAJAN, R. und B. SHNEIDERMAN (1997). *Visual and Textual Consistency Checking Tools for Graphical User Interfaces*. IEEE Transactions on Software Engineering, 23(11):722–735.

MARIAGE, C., J. VANDERDONCKT und C. PRIBEANU (2005). *State of the Art of Web Usability Guidelines*, Kap. 12. Lawrence Erlbaum Associates, New Jersey.

MARRENBACH, J. (2001). *Werkzeug-basierte Evaluierung der Benutzungsfreundlichkeit interaktiver Endgeräte mit normativen Benutzermodellen*. Doktorarbeit, RWTH Aachen, Aachen.

MAYHEW, D.J. (1999). *The Usability Engineering Life Cycle*. Morgan Kaufmann Publishers, San Francisco, CA, USA.

MCCARTHY, JOHN L. (1960). *Recursive Functions of Symbolic Expressions and Their Computation by Machine, Part I*. Communications of the ACM, 3(4):184–195.

MCDONALD, J. E., D. W. DEARHOLT, K. R. PAAP und R. W. SCHVANEVELDT (1986). *A Formal Interface Design Methodology Based on User Knowledge*. In: *Proceedings of the SIGCHI conference on Human factors in computing systems*, S. 285–290.

MICROSOFT (2001). *Windows XP Visual Guidelines*. Online: http://www.microsoft.com/whdc/Resources/windowsxp/default.mspx.

MILLER, D. P. (1981). *The Depth/Breadth Tradeoff in Hierarchical Computer Menus*. In: *Proceedings of the Human Factors Society*.

MOUROUZIS, A., D. GRAMMENOS, P. KARAMPELAS und C. STEPHANIDIS (2004). *Users Requirements Towards Guidelines and Design Support Tool and Their Evolution*. Technischer Bericht IST-2000-26089 - D2.3, University of Nottingham.

MOUSSA, F., C. KOLSKI und M. RIAHI (2000). *A Model Based Approach to Semi-automated User Interface Generation for Process Control Interactive Applications*. Interacting with Computers, 12(3):245–279.

MOUSSA, F. und M. RIAHI (2001). *Use of Formalized Guidelines for Semi-Automated Generation of GUI: the Ergo-Conceptor+ Tool*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 237–246. Springer Verlag, London.

MOUSSA, F., M. RIAHI, C. KOLSKI und M. MOALLA (2002). *Interpreted Petri Nets used for Human-Machine Dialogue*. Integrated Computer-Aided Engineering, 9(1):87–98. IOS Press, Amsterdam.

NEBENDAHL, D., Hrsg. (1987). *Expertensysteme – Einführung in Technik und Anwendung*. Siemens Verlag, Berlin.

NEWMAN, M. W. und J. A. LANDAY (2000). *Sitemaps, Storyboards, and Specifications: A Sketch of Web Site Design Practice*. In: *Proceedings of the Conference on Designing Interactive Systems: Processes, Practices, Methods, and Techniques*, S. 263 – 274, Berkeley, CA, USA. ACM Press, New York.

NIEDERMAIER, F. B. (2002). *Entwicklung und Bewertung eines Rapid-Prototyping Ansatzes zur multimodalen Mensch-Maschine-Interaktion im Kraftfahrzeug*. Doktorarbeit, TU München.

NIELSEN, J. (1994). *Usability Engineering*. Morgan Kaufmann Publishers, San Francisco.

NIELSEN, J. (2000). *The Mud-Throwing Theory of Usability*. Online. http://www.useit.com/alertbox/20000402.html.

NIELSEN, J. und R. L. MACK (1994). *Usability Inspection Methods*. J. Wiley & Sons, Inc., New York.

NITSCHKE, J. und H. WANDKE (2002). *GUIDEAS – Guidance for Developing Assistance*. Humbold Spektrum 1, Humboldt Universität, Berlin. pp. 34?-38.

NORMAN, D. A. (1982). *Steps Toward a Cognitive Engineering: Design Rules Based on Analyses of Human Error*. In: *Proceedings of the 1982 Conference on Human Factors in Computing Systems*, S. 378–382.

NORMAN, D. A. (1988). *The Psychology of Everyday Things, Basic Books, New York, 1988.*. Basic Books, New York.

NORMAN, D. A. (2002). *The Design of Everyday Things*. 2. Basic Books, New York.

NORMAN, D. A. und J. NIELSEN (2003). *Usability Return on Investment*. Technischer Bericht, Nielsen Norman Group.

NORMAN, K. L. (1991). *The Psychology of Menu Selection: Designing Cognitive Control at the Human/Computer Interface*. Greenwood Publishing Group Inc., Westport, CT, USA.

NOY, N. F. und D. L. MCGUINNESS (2001). *Ontology Development 101: A Guide to Creating Your First Ontology*. Technischer Bericht KSL-01-05, Knowledge Systems Laboratory, Stanford University.

OED, R., A. BECKER und E. WETZENSTEIN (2001). *Welche Unterstützung wünschen Softwareentwickler beim Entwurf von Bedienoberflächen?*. In: OBERQUELLE, H., R. OPPERMANN und J. KRAUSE, Hrsg.: *Mensch & Computer 2001: 1. Fachübergreifende Konferenz*, S. 355–364. B. G. Teubner, Wiesbaden.

OERTEL, K. und W. LACHERMEIER (2005). *Spracherkennung bei der BMW Group: „Es gilt das gesprochene Wort“*. automotive, 09-10:16–21. Carl Hanser Verlag, München.

O'HARA, J. (1993). *The Development and Evaluation of Human Factors Guidelines for the Review of Advanced Human-System Interfaces*. In: *Conference Record for 1992 IEEE Fifth Conference on Human Factors and Power Plants*, S. 260–265. IEEE, New Jersey.

OMASREITER, H. und E. METZKER (2004). *A Context-Driven Use Case Creation Process for Specifying Automotive Driver Assistance Systems*. In: *Requirements Engineering Conference, 12th IEEE International (RE'04)*, S. 334–339.

OPPERMANN, R. und H. REITERER (1997). *Software Evaluation using the 9241 Evaluator*. Behaviour & Information Technology, 16:232–245.

OSACA E.V. (1997). *Style Guide Werkzeugmaschinen: Ein Handbuch zur Gestaltung von Benutzungsoberflächen für Werkzeugmaschinen*. Fraunhofer IRB Verlag, Stuttgart.

OSGOOD, C. E., G. SUCI und P. TANNENBAUM (1957). *The Measurement of Meaning*. University of Illinois Press, Urbana.

PARUSH, A. (2001). *A Data Base Approach to Building and Using Online Human Computer Interaction Guidelines*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 77–86. Springer Verlag, London.

PAUSCH, R., N. R. YOUNG und R. DELINE (1991). *SUIT: the Pascal of User Interface Toolkits*. In: *UIST '91: Proceedings of the 4th annual ACM symposium on User interface software and technology*, S. 117–125. ACM Press ACM Press, New York.

PAWAR, S. A. (2004). *A Common Software Development Framework for Coordinating Usability Engineering and Software Engineering Activities*. Diplomarbeit, Virginia Polytechnic Institute, Virginia.

PFAFF, G. E., Hrsg. (1983). *User Interface Management Systems: Proceedings of the Seeheim Workshop*, Berlin. Springer Verlag.

PFISTERER, C. (2002). *A Semantic Description Language for Platform-Independent Graphical User Interfaces*. Diplomarbeit, Institut für Visualisierung und Interaktive Systeme, Universität Stuttgart.

PONSARD, C., N. BALYCH, P. MASSONET, A. VAN LAMSWEERDE und J. VANDERDONCKT (2005). *Goal-oriented Design of Domain Control Panels*. In: *Proceedings of the 12th International Workshop on Design, Specification and Verification of Interactive Systems*, Newcastle (UK).

POSCH, T., K. BIRKEN und M. GERDOM (2004). *Basiswissen Softwarearchitektur*. dpunkt-Verlag, Heidelberg.

PREECE, J., Y. ROGERS und H. SHARP (2002). *Interaction Design: Beyond Human-Computer Interaction*. John Wiley & Sons, Washington D.C.

PRIBEANUS, C. (2000). *Tools for Evaluating the Visual Consistency Of Graphical Human-Computer Interfaces*. Studies in Informatics and Control, 9(1).

PRICE, H.E. (1985). *The Allocation of Functions in Systems*. Human Factors, 27(1):33–45. Human Factors and Ergonomics Society (HFES), Santa Monica.

PUPPE, F. (1991). *Einführung in Expertensysteme*, Bd. 2 d. Reihe *Studienreihe Informatik*. Springer Verlag.

RASKIN, J. (2000). *The Humane Interface*. Addison-Wesley Professional, Boston.

RASSL, R. (2004). *Ablenkungswirkung tertiärer Aufgaben im Pkw - Systemergonomische Analyse und Prognose*. Doktorarbeit, Technische Universität München.

RAUCH, N., I. TOTZKE und H.-P. KRÜGER (2004). *Kompetenzerwerb für Fahrerinformationssysteme: Bedeutung von Bedienkontext und Menüstruktur*. In: VEREIN DEUTSCHER INGENIEURE, Hrsg.: *Integrierte Sicherheit und Fahrerassistenzsysteme*, Bd. VDI-Berichte Nr. 1864, S. 303–322, Düsseldorf. VDI-Verlag.

RAUTERBERG, M. (1992). *Lässt sich die Gebrauchstauglichkeit interaktiver Software messen? Und wenn ja, wie?*. Ergonomie & Informatik, 16:3–18.

RAUTERBERG, M. (1996). *Usability Engineering*. Technischer Bericht, ETH Zürich, Zürich.

RAUTERBERG, M. (1997). *Methoden zur Normenkonformitätsprüfung im Rahmen von ISO 9241*. In: KERN, P., Hrsg.: *Die EU-Bildschirmrichtlinie in der Praxis*, S. 239–258. Institut für Arbeitswissenschaft und Technologiemangement IAT, Universität Stuttgart.

REITERER, H. (1995). *Die IDA-Entwicklungsumgebung: Einsatz von objekt-orientierten, multimedialen und wissensbasierten Unterstützungswerkzeugen zur ergonomischen Gestaltung von Benutzungsoberflächen*. Informatik - Forschung und Entwicklung, 10(4):180–196. Springer Verlag, Berlin.

REITERER, H. (2001). *Tools for Working with Guidelines in Different Interface Design Approaches*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 225–236. Springer Verlag, London.

RESEARCH INTERNATIONAL DEUTSCHLAND (2004a). *Das Handy, ein Mysterium für Grey Consumers?*. Technischer Bericht, Hamburg.

RESEARCH INTERNATIONAL DEUTSCHLAND (2004b). *Kaufblockaden und wie sie zu therapieren sind*. Technischer Bericht, Hamburg.

RÖSSGER, P. (2005). *Die Ablenkung von Fahrern durch Fahrer-Informations-Systeme: Stand der Diskussion und Maßnahmen*. In: URBAS, L., Hrsg.: *Zustandserkennung und Systemgestaltung*, Reihe 22 - Fortschritt-Berichte. VDE Verlag.

RHYNE, J., R. EHRICH, J. BENNETT, T. HEWETT, J. SIBERT und T. BLESER (1987). *Tools and Methodology for User Interface Development*. Computer Graphics, 21(2):78–87.

RIEDL, M O. und R. S. AMANT (2002). *Toward Automated Exploration of Interactive Systems*.

ROLLAND, C. und C. BEN ACHOUR (1998). *Guiding the Construction of Technical Use Case Specifications*. Data & Knowledge Engineering Journal, 25(1-2):125–160.

ROSSON, M. B. und J. M. CARROLL, Hrsg. (2002). *Usability Engineering: Scenario-Based Development of Human-Computer Interaction*. The Morgan Kaufmann Series in Interactive Technologies. Morgan Kaufmann Publishers, San Fransisco.

RUSHBY, J. (2002). *Using Model Checking to Help Discover Mode Confusions and Other Automation Surprises*. Reliability Engineering and System Safety, 75(2):167–177.

RUSSEL, S. und P. NORVIG (2004). *Künstliche Intelligenz*. Prentice Hall Studium, München.

SAE (2002). *Recommended Practice Calculation of the Time to Complete In-Vehicle Navigation and Route Guidance Tasks (SAE J2365)*. Society of Automotive Engineers, Warrendale.

SAE (2004). *Recommended Practice Navigation and Route Guidance Function Accessibility While Driving (SAE 2364)*. Society of Automotive Engineers, Warrendale.

SAP (2004). *R/3 Style Guide*. SAP AG, Walldorf.

SAVORY, S. E. (1990). *Grundlagen von Expertensystemen*. Oldenbourg Verlag, München, 2 Aufl.

SCAPIN, D., C. LEULIER, J. VANDERDONCKT, C. M. C. BASTIEN, C. FARENC, P. PALANQUE und R. BASTIDE (2001a). *A Framework for Organizing Web Usability Guidelines*. In: KORTUM, PH. und E. KUDZINGER, Hrsg.: *Proceedings of 6th Conference on Human Factors and the Web*. online. available at http://www.tri.sbc.com/hfweb/scapin/Scapin.html.

SCAPIN, D., J. VANDERDONCKT, C. FARENC, R. BASTIDE, C. BASTIEN, C. LEULIER, C. MARIAGE und P. PALANQUE (2001b). *Transferring Knowledge of User Interfaces Guidelines to the Web*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 293–304. Springer Verlag, London.

SCHAEFFER, M.S. (1987). *Hologram luminance study*. Technischer Bericht HAC IDC Ref. No. 061187.MSS, Hughes Aircraft Company, El Segundo, CA.

SCHINDHELM, R., C. GELAU, A. KEINATH, K. BENGLER, H. KUSSMANN, P. KOMPFER, C. P. CACCIABUE und M. MARTINETTO (2004). *Report on the Review of the Available Guidelines and Standards*. Technischer Bericht IST-1-507674-IP, AIDE - Adaptive Integrated Driver-Vehicle Interface.

SCHMIDTKE, H. (1981). *Lehrbuch der Ergonomie*. Carl Hanser Verlag, München, 2 Aufl.

SCHVANEVELDT, R. W., Hrsg. (1990). *Pathfinder Associative networks - Studies in Knowledge Organization*. 2. Ablex Publishing Corporation, New Jersey.

SEARS, A. (1993a). *AIDE: A Tool to Assist in the Design and Evaluation of User Interfaces*. Technischer Bericht, Interactive Systems Research Center, Baltimore.

SEARS, A. (1993b). *Layout Appropriateness: A Metric for Evaluating User Interface Widget Layout*. IEEE Transactions on Software Engineering, 19(7):707–719. IEEE Press, New Jearsey.

SEARS, A. (1995). *AIDE: a Step Toward Metric-Based Interface Development Tools*. In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, S. 101–110. ACM Press, New York.

SEIFERT, K. (2002). *Evaluation multimodaler Computer-Systeme in frühen Entwicklungsphasen*. Doktorarbeit, TU Berlin.

SHIFFMAN, R. N., G. MICHEL, A. ESSAIHI und E. THORNQUIST (2004). *Bridging the Guideline Implementation Gap: A Systematic, Document-Centered Approach To Guideline Implementation*. JAMIA - Journal of the American Medical Informatics Association, 11(5):418–426. Published online: http://www.jamia.org/cgi/reprint/11/5/418.pdf.

SHNEIDERMAN, B. und C. PLAISANT (2005). *Designing the User Interface: Strategies for effective Human-Computer Interaction*. Addison Wesley, 4. Aufl.

SIRVALUSE (2003). *Usability-Test der Bedienkonzepte in Oberklasse-Limousinen*. Technischer Bericht, SirValUse Consulting GmbH, Hamburg.

SIRVALUSE (2004). *Best Practice - Mittelklasse in der Oberklasse*. Usability-Compass, 1:2–4. Sirvaluse Consulting, Hamburg.

SMITH, S. L. (1988). *Standards Versus Guidelines for Designing User Interface Software*. In: HELANDER, M., Hrsg.: *Handbook of Human-Computer Interaction*, S. 877–889. North-Holland, Amsterdam.

SMITH, S. L. und J. N. MOSIER (1986). *Guidelines for Designing User Interface Software*. ESD-TR-86-278, The MITRE Corporation, Bedford, Massachusetts, USA.

SNYDER, C. (2003). *Paper Prototyping*. Morgan Kaufmann Publishers, San Fransisco.

SONG, X. (2002). *Systematic Integration of Design Methods*. Software, IEEE, 14(2):107–117.

SOUZA, F. und N. BEVAN (1990). *The Use of Guidelines in Menu Interface Design: Evaluation of a Draft Standard*. In: *IFIP INTERACT'90: Human-Computer Interaction*, S. 435–440. IFIP.

STANTON, N. und C. BABER (1996). *Methods and Techniques of Usability Evaluation*. In: JORDAN, P. W., B. THOMAS, B. A. WEERDMEESTER und I. L. MCCLELLAND, Hrsg.: *Usability Evaluation in Industry*, London. Taylor and Francis.

STARY, C., T. RIESENECKER-CABA und J. FLECKER (1995). *EU-konforme Bewertung von Bildschirmarbeit – Schritte zur Operationalisierung*. vdf Hochschulverlag AG an der ETH Zürich, Zürich.

STATISTISCHES BUNDESAMT (2004). *Ausstattung privater Haushalte mit Informations- und Kommunikationstechnik, Ergebnis der Einkommens- und Verbrauchsstichprobe 1998 und 2003*. Wiesbaden.

STEINMÜLLER, J. (2005). *Expertensysteme*. Lehrstuhl für Künstliche Intelligenz, TU-Chemnitz. Skript zur Vorlesung.

STEINWEG, C. (2005). *Management der Software-Entwicklung*. Vieweg Verlag, 6. Aufl.

STEVENS, A., A. QUIMBY, A. BOARD, T. KERSLOOT und P. BURNS (2004). *Design Guidelines for Safety of In-Vehicle Information Systems*. Technischer Bericht PA3721/01, Transport Research Laboratory.

STONEBRAKER, M., M. HEARST und S. POTAMIANOS (1989). *A commentary on the POSTGRES rules system*. # j-SIGMOD#, 18(3):5–11.

STRAYER, D.L. und W. A. JOHNSTON (2001). *Driven to Distraction: Dual-Task Studies of Simulated Driving and Conversing on a Cellular Telephone*. Psychological Science, 12(2):462–466. Blackwell Publishing, Oxford.

SZAMEITAT, S. (2003). *Gestaltung von Auswertungssystemen für sicherheitskritische Ereignisse in Industrieanlagen mit hohem Gefährdungspotential*. Doktorarbeit, Technischen Universität Berlin.

TATZLAFF, L. und D. R. SHWARTZ (1991). *The Use of Guidelines in Interface Design*. In: ROBERTSON, S. P., G. OLSEN und J. S. OLSON, Hrsg.: *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, S. 329–333. ACM Press, New York.

THALHEIM, B. (2000). *Entity-Relationship Modeling*. Springer, Berlin.

TIEDTKE, T., C. MÄRTIN und N. GERTH (2002). *AWUSA - A Tool for Automated Website Usability Analysis*. In: FORBRIG, P., Q. LIMBOURG, B. URBAN und J. VANDERDONCKT, Hrsg.: *9th International Workshop on Design, Specification, and Verification of Interactive Systems, DSV-IS 2002*, Bd. 2545 d. Reihe *Lecture Notes in Computer Science*. Springer Verlag.

TIESCHKY, S., W. HAMBERGER, J. SCHRÖDER und G. MAUTER (2002). *Audi MMI - von der Idee zum Produkt*. Elektronik Automtotive, S. 10–18.

TIMPE, K. P., H. KOLREP und T. JÜRGENSOHN (2002). *Mensch-Maschine-Systemtechnik – Konzepte, Modellierung, Gestaltung, Evaluation*. Symposion Publishing GmbH, Düsseldorf.

TROMP, J., R. EASTGATE, S. NICHOLS, H. PATEL, M. DCRUZ, A. MOUROUZIS, D. GRAMMENOS und J. WILSON (2004). *Development of Guidelines and Interactive Design Support Tool - Code of Good Practice*. Technischer Bericht IST-2000-26089 - D8.1, University of Nottingham.

TSCHÖPE, G. J. und J. NITSCHKE (2002). *Instruktionen und Usability: Empirische Untersuchung anhand eines Tools für Entwickler*. In: HERCZEG, M. und W. PRINZ, Hrsg.: *Mensch & Computer 2002: Vom interaktiven Werkzeug zu kooperativen Arbeits- und Lernwelten*, S. 235–244. B. G. Teubner, Stuttgard.

TULLIS, T. (1988). *A System for Evaluating Screen Formats*. In: HARTSON, R. und D. HIX, Hrsg.: *Advances in Human-Computer Interaction*. Ablex Publishing, New Jersey.

TUMA, T. (1997). *Der programmierte Frust.*. Der Spiegel, 48. Springer-Verlag, Hamburg.

VAN WELIE, M. und G. C. VAN DER VEER (1999). *Ontologies and Methods in Interdisciplinary Design*. In: *Computer Science Education: Challenges for the New Millenium*, S. 143–158.

VANDERDONCKT, J. (2005). *A MDA-Compliant Environment for Developing User Interfaces of Information Systems.*. In: PASTOR, O. und J. F. CUNHA, Hrsg.: *CAiSE*, Bd. 3520 d. Reihe *Lecture Notes in Computer Science*, S. 16–31. Springer Verlag, Berlin.

VANDERDONCKT, J. und A. BEIREKDAR (2005). *Automated Web Evaluation by Guideline Review.*. Journal of Web Engineering, 4(2):102–117. Rinton Press.

VDE/ITG (1995). *ITG-Prüfverfahren zur Bewertung der Benutzungsfreundlichkeit von Telekommunikationssystemen und -endgeräten – ITG-Richtlinie 9.4.1.1*. VDE Verlag, Offenbach, Berlin.

VERMEEREN, A. P. O. S. (1999). *Designing Scenarios and Tasks for User Trials of Home Electronic Devices*. In: GREEN, W. S. und P. W. JORDAN, Hrsg.: *Human Factors in Product Design – Current Practices and Future Trends*, S. 47–55. Fracis & Taylor, London.

VICENTE, K. J., C. M. BURNS und W. S. PAWLAK (1997). *Muddling through Wicked Design Problems*. Ergonomics in Design, 5(1):25–30.

VOGT, T. (2001). *Difficulties in Using Style Guides for Designing User Interfaces*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 197–208. Springer Verlag, London.

WANDKE, H. und J. HÜTTNER (2001). *Completing Human Factor Guidelines by Interactive Examples*. In: VANDERDONCKT, J. und C. FARENC, Hrsg.: *Tools for Working with Guidelines*, S. 99–106. Springer Verlag, London.

WANDKE, H., R. OED, E. METZKER, M. VAN BALLEGOOY und J. NITSCHKE (2001). *Die Entwicklung von User Interfaces als arbeitswissenschaftlicher Prozess und seine Unterstützung durch Software-Tools*. Zeitung für Arbeitswissenschaften, 2. Gesellschaft für Arbeiswissenschaft e.V., Dortmund.

WANDMACHER, J. (1993). *Software-Ergonomie*. Mensch Computer Kommunikation, Grundwissen Bd.2. de Gruyter, Berlin.

WHITEFIELD, A., F. WILSON und J. DOWELL (1991). *A framework for human factors evaluation*. Behaviour and Information Technology, 10:65–79.

WICKENS, C. D., M. VIDULICH und D. SANDRY-GARZA (1984). *Principles of S-C-R Compatibility with Spatial and Verbal Tasks: The Role of Display-Control Location and Voice-Interactive Display-Control Interfacing*. Human Factors, 26(5):533–543.

WISSELMANN, D., K. GRESSER, H. SPANNHEIMER, K. BENGLER und A. A. HUESMANN (2004). *ConnectedDrive - ein methodischer Ansatz für die Entwicklung zukünftiger Fahrerassistenzsysteme*. In: *Aktive Sicherheit durch Fahrerassistenzsysteme*. TÜV Akademie GmbH.

YOM, M. und D. FEHRLE (2005). *Wording-Studie 2005? Nutzungsfreundliche Bezeichnung von Navigationselementen im Internet!*. Technischer Bericht, eResult GmbH, Göttingen.

ZAPHIRIS, P., B. SHNEIDERMAN und K. L. NORMAN (2002). *Expandable Indexes vs. Sequential Menus for Searching Hierarchies on the World Wide Web*. Behaviour and Information Technology, 21(3):201–207.

ZIEGLER, J. und R. ILG, Hrsg. (1993). *Benutzergerechte Software-Gestaltung*. Oldenbourg Verlag, München.

ZIEREN, J. (2000). *Automatische Generierung normativer Benutzermodelle aus SDL-Spezifikationen*. Diplomarbeit, Lehrstuhl für Technische Informatik, RWTH Aachen.

ZIMBARDO, P. G. und J. GERRIG (1999). *Psychologie*. Springer Verlag, Berlin.

Abbildungsverzeichnis

Tabellenverzeichnis

Anhang A

Detaillierte Beschreibung der Regelsprache

In diesem Abschnitt ist die Syntax der in Kapitel 3.2 beschriebenen Regelsprache vorgestellt. Dabei werden in Abschnitt A.1 ausgehend von einer Erläuterung der grundlegenden Sprachkonstrukte die Möglichkeiten der Implementierung von Bewertungsregeln erklärt. Zusätzlich ist eine Auflistung der wichtigsten Funktionen zur Ansteuerung des Datenspeichers sowie des Prototyping-Werkzeugs STATESHADE gegeben. Die vollständige BNF der Regelsprache ist in Abschnitt A.2 vorgestellt.

A.1 Formalismus zur Beschreibung ergonomischer Erkenntnisse als Regeln

Regeln bestehen grundsätzlich aus einem Wenn-Dann-Konstrukt:

Wenn *P* dann *Q*

P stellt die Vorbedingung (Prämisse) und *Q* die Aktion (Konklusion) dar. Die Prämisse beschreibt eine Situation, in der die Konklusion ausgeführt werden soll, wobei Elemente des zu bewertenden Systems daraufhin überprüft werden, ob sie bestimmte, in der Prämisse angegebene Eigenschaften besitzen bzw. erfüllen. Trifft die Prämisse zu, dann *feuert* die Regeln und die Aktionen der Konklusion werden ausgeführt.

Im Folgenden werden die Möglichkeiten der Implementierung von Prämisse und Konklusion detailliert vorgestellt, angeführt von einer Erläuterung grundlegender Konstrukte, auf der die Regelsprache basiert.

A.1.1 Grundlegende Konstrukte der Regelsprache

Die in dieser Arbeit benutzte Regelsprache basiert auf der Implementierung von LISP im Rahmen der Expertensystem-Bibliothek *JESS*. Zum besseren Verständnis sind die grundlegenden Sprachkonstrukte im Folgenden vorgestellt, eine ausführliche Erklärung der Sprache findet sich in (FRIEDMAN-HILL 2003).

Token

Die grundlegenden Elemente der Sprache sind *Token*. Token sind Zeichenketten und werden klassifiziert als *Symbole*, *Nummern*, *Strings* und *Kommentare*. Nummern sind Ganzzahlen (Integer) bzw. Fließkommazahlen (Float). Strings bestehen aus Zeichenketten, die durch Hochkommas eingeschlossen sind. Kommentare sind beliebige Zeichenketten, denen ein Semikolon vorangestellt ist. Symbole sind Zeichenketten zur Beschreibung von Funktionen (z.B. Schlüsselwörter) oder Variablen. Symbole können jedoch auch interpretationsfrei, also ohne Verbindung zu einer Funktion oder Variable, existieren. Beispielhaft ist daher eine Variablenbelegung sowohl mit *HelloWorld* als auch *„Hello World“* möglich, im ersten Fall wäre die Variable vom Typ *Symbol*, im zweiten Fall vom Typ *String*.

Variablen

Variablen entsprechen Containern zur Aufnahme eines Wertes. Variablen sind grundsätzlich untypisiert und müssen daher nicht deklariert werden. Sie werden durch das Präfix *?* markiert. Gültige Variablennamen sind z.B. `?x`, `?_temp`.

Ausdrücke und Funktionen als Liste

Die Struktur eines Ausdrucks entspricht einer Liste von Symbolen. Die Liste ist geklammert und Symbole mit einem Leerzeichen voneinander getrennt. Regeldefinitionen, Funktionen und mathematische Ausdrücke folgen diesem Prinzip. Für mathematische Ausdrücke entspricht die Darstellung der Präfixnotation[1]. Folgende Beispiele verdeutlichen den grundsätzlichen Aufbau:

`(+ 3 4)` entspricht dem mathematischen Ausdruck $(3+4)$.

`(printout shell 'hello world')` ruft die Funktion `printout` mit zwei Parametern auf. Der erste Parameter gibt den Ausgabestream, der zweite den Ausgabetext an.

`(eine Liste)` entspricht einer einfachen Liste mit zwei Symbolen, sofern `eine` nicht als Funktion mit einem Parameter definiert ist.

[1] s. Glossar

Parameter von Funktionen sind Token oder wiederum Listen, wobei grundsätzlich bei verschachtelten Listen die Auswertung bei der innersten Liste beginnt

`(printout shell (+ 3 4))` ergibt die Ausgabe `7` in der Shell.

A.1.2 Implementierung von Regeln

Regeln haben grundsätzlich folgenden Aufbau:

```
( defrule <Prämisse> => <Konklusion> )
```

Das Schlüsselwort `defrule` leitet die Regeln ein. Prämisse und Konklusion sind durch „=>“ voneinander getrennt und in den folgenden Abschnitten erläutert. Eine vollständige BNF der Syntax ist in Abschnitt A.2 gegeben.

A.1.2.1 Aufbau der Regel-Prämissen

Alle Ausdrücke einer Regelprämisse bilden die Grundlage für ein Pattern-Matching. Dementsprechend darf die Prämisse lediglich aus der Angabe von Fakten bzw. deren Attributwerten bestehen. Bei der Ausführung der Regeln wird versucht, passende Fakten aus der Faktenbasis zu finden, die die angegebenen Werte besitzen. Aus diesem Grund brauchen nur die Attribute angegeben zu werden, deren Werte zu überprüfen sind. Folgender Ausdruck stellt eine korrekte Prämissen dar: `(TextElement (xpos 100))`
Die Regel feuert für jedes `TextElement`, dessen X-Position bei 100 liegt.

Ausdrücke können mit den logischen Verknüpfungen `and`, `or` und `not` verknüpft werden, wobei ebenfalls die Präfix-Notation gilt. Fehlt eine logische Verknüpfung und sind mehrere Ausdrücke vorhanden, dann werden diese implizit `and`-verknüpft.

Freie Variablen werden in der Prämisse bei ihrem ersten Auftreten mit einem in der Faktenbasis vorkommenden Wert belegt. Beim nächsten Vorkommen dieser Variable erfolgt die Abfrage, ob das Attribut des angegeben Fakts den Wert der Variable enthält. Folgendes Beispiel verdeutlicht diesen Umstand:

```
(TextElement (content Word) (ypos ?x))
(Box (ypos ?x))
```

Die Regel feuert, wenn mindestens ein `TextElement` mit dem Begriff „Word“ und mindestens einer `Box` mit exakt gleicher Y-Position existiert. Beim ersten Auftreten der Variable `?x` wird diese mit dem Wert des Fakts `TextElement` belegt und für das Fakt `Box` gefordert.

Grundsätzlich dürfen Ausdrücke in der Prämisse nur aus der Angabe von Fakten und deren Attribute bestehen. Einzige Ausnahme ist der Aufruf von Funktionen, die einen booleschen Rückgabewert liefern. Im Rahmen des Pattern-Matching schlägt ein Funktionsaufruf fehl, wenn er den Rückgabewert `false` liefert.

Die Kombinationsmöglichkeiten von Ausdrücken in einer Prämisse demonstriert das Beispiel aus Abbildung 2.5 (Seite 28), das in Abbildung A.1 in korrekter Syntax dargestellt ist. Die Funktion `eq` testet zwei Token auf Gleichheit, während die Funktion `nachbar` die Platzierung der GUI-Elemente (repräsentiert durch ihren eindeutigen `level`) überprüft und als Rückgabewert `true` gibt, wenn beide Elemente benachbart sind oder sich überlappen.

```
(defrule
     (TextElement (level ?level1) (color ?farbe1))
     (GuiElement (level ?level2) (color farbe2))
     (or
           (and (test (eq ?farbe1 blau))
                (test (= ?farbe2 rot)))
           (and (test (eq ?farbe1 rot))
                (test (= ?farbe2 blau)))
     )
     (test (nachbar ?level1 ?level2))
=>
     (assert (var 'Farbproblem') (v1 ?level1) (v2 ?level2))
     (printout shell 'Fehlermeldung')
)
```

Abbildung A.1: Implementierung einer Regel zur Überprüfung des Farbkontrastes zweier GUI-Elemente in JESS.

A.1.2.2 Aufbau der Regel-Konklusionen

Die Konklusion einer Regeln enthält die Aktionen, die ausgeführt werden sollen, wenn die Prämisse *gefeuert* hat, also alle Ausdrücke der Prämisse erfolgreich überprüft werden konnten.

Die Ausdrücke der Konklusion bestehen ausschließlich aus Anweisungen für Aktionen (im Gegensatz zur Prämisse) in Form von Funktionsaufrufen. Dabei dienen die Funktionen überwiegend zur Veränderung der Faktenbasis sowie zur Generierung von Ausgaben.

Die Veränderung der Faktenbasis wird über eine Modifikation der Fakten erreicht. Dabei können Fakten neu angelegt (`(assert ...)`, gelöscht (`(retract ...)` und modifiziert (`(modify ...)` werden. Eine Veränderung der Faktenbasis ist sofort wirksam und bedingt eine erneute Abarbeitung bestehender Regeln. Beispielhaft wird in Abbildung A.1 im Fehlerfall ein Fakt `Farbproblem` der Faktenbasis hinzugefügt, das den Level der beiden Elemente zur eindeutigen Identifizierung enthält.

Zur Ausgabe von Informationen dient die Funktion (`printout ...`) mit der Angabe des Ausgabeortes (als ersten Parameter) sowie der auszugebenden Informationen (als folgende Parameter). Eine Ausgabe ist ebenfalls in der Konklusion in Abbildung A.1 dargestellt.

Selbst definierte Funktionen, die durch `(deffunction ...)` spezifiziert oder in Java implementiert sind, lassen sich ebenfalls aufrufen.

A.1.2.3 Zusätzliche Funktionen zum Aufruf in der Konklusion

Die zusätzlichen Funktionen, die der Regelsprache zur Verfügung stehen, sind im Folgenden auszugshaft vorgestellt. Dabei handelt es sich überwiegend um Funktionen für den Zugriff auf den Datenspeicher sowie zur Steuerung der Ausführungsreihenfolge der Wissensbasen.

- `setIgnoreSteps (string knowledgebase, boolean)` – Ignoriert die angegeben Wissensbasis während der Ausführung.

Funktionen zur Ansteuerung des Datenspeichers (Queue):

Die Queue speichert die Fakten in Form von *Szenen*. Eine Szene kann mehrere Fakten (z.B. alle Elemente einer GUI-Konstellation), eine Queue mehrere Szenen enthalten. So ist es möglich, die Bewertungsergebnisse für jede GUI-Konstellation einer Aufgabenbearbeitung in der Queue zu speichern. Der Zugriff auf die Queue in einer Regelkonklusion (angegeben im Kriterieneditor) ist durch folgende Funktionen gewährleistet:

- `QLoadQueue(string name)` – Lädt die Queue unter dem angegebenen Namen.
- `QSaveQueue(string name)` – Speichert die Queue unter dem angegebenen Namen.
- `QInfoQueue()` – Gibt die gesamte Queue auf dem Bildschirm aus.
- `QClear()` – Löscht alle Daten der Queue.
- `QAddFact(fact)` – Legt einen Fakt (mit beliebigem Format) in die aktuelle Szene innerhalb der Queue.
- `QNewScene()` – Legt eine neue Szene an, in der Fakten gespeichert werden können.
- `QGetScene(int,knowledgebase)` – Fügt alle Fakten der angegebenen Szene in die Faktenbasis der Wissensbasis ein, in der diese Funktion aufgerufen wurde.
- `QGetFirstScene(knowledgebase)` – Fügt alle Fakten der ersten (vordersten) Szene in die Faktenbasis der Wissensbasis ein, in der diese Funktion aufgerufen wurde.
- `QGetLastScene(knowledgebase)` – Fügt alle Fakten der letzten (hintersten) Szene in die Faktenbasis der Wissensbasis ein, in der diese Funktion aufgerufen wurde.
- `QRemoveFirstScene()` – Löscht die erste (vorderste) in der Queue gespeicherte Szene.

- `QRemoveLastScene()` – Löscht die letzte (hinterste) in der Queue gespeicherte Szene.

Funktionen zur Ansteuerung von STATESHADE:

STATESHADE[2] ist ein Werkzeug zum Rapid-Prototyping und erlaubt die Beschreibung der GUI während der Ausführung eines Prototypen in dem in dieser Arbeit entwickelten GUI-Format. Den Zugriff auf STATESHADE aus den Regeln von REVISER gewährleisten folgende Funktionen:

- `loadProject(string name)` – Lädt in STATESHADE das unter dem Dateinamen gespeicherte STATESHADE-Projekt.
- `getScreenShot(knowledgebase)` – Fügt alle Elemente, die aktuell in STATESHADE dargestellt werden, als Fakten in die Faktenbasis der aktuellen Wissensbasis ein.
- `perfomAction(string action)` – Übergibt STATESHADE eine Aktion zur Bedienung des Prototypen.
- `perfomActionGetScreen(string action, knowledgebase)` – Übergibt STATESHADE eine Aktion zur Bedienung des Prototypen und fügt alle Elemente, die anschließend in STATESHADE dargestellt werden, als Fakten in die Faktenbasis der aktuellen Wissensbasis ein.
- `guiViewerSetVisible(boolean)` – Zeigt die Prototyping-Komponenten von STATESHADE auf dem Bildschirm an oder entfernt sie.
- `boolean guiViewerVisible()` – Gibt an, ob die Prototyping-Komponenten von STATESHADE aktuell angezeigt wird oder nicht.

A.2 BNF des Regelformats

Die BNF des in dieser Arbeit benutzten Regelformats entspricht bis auf wenige Details der von JESS (in der Version 6.03) und ist im Folgenden vollständig gegeben. Native Datentypen wie `integer`, `float` oder `string` sind nicht aufgelistet. Eine Definition dieser Datentypen findet sich in (FRIEDMAN-HILL 2003).

```
(e = empty)

<JESS>        ::= <defs> | e
<defs>        ::= <def> <defs>
<def>         ::= <atom> | <string> | <variable> | <multivariable> |
                  <float> | <integer> | <funcall> | <defrule> |
```

[2] http://www.hamacher-online.com/stateshade

```
                  <defquery> | <deffacts> | <deftemplate> |
                  <deffunction> | <defglobal>

<defrule>     ::= ( defrule <atom> <docString> <declaration> <LHS> =>
                  <actions> )
<docString>   ::= <string> | e
<declaration> ::= ( declare <valuePair> <valuePairs> ) | e
<valuePairs>  ::= <valuePair> <valuePairs> | e
<valuePair>   ::= ( <atom> <liste> )
<liste>       ::= <atom> | <string> | <variable> | <multivariable> |
                  <float> | <integer> | <funcall>
<funcall>     ::= ( <functor> <arguments> )
<functor>     ::= <atom> | = | <variable>
<argument>    ::= <atom> | <string> | <variable> | <multivariable> |
                  <float> | <integer> | <nestedFuncalls>
<arguments>   ::= <argument> <arguments> | e
<nestedFunclass> ::= ( assert <fact> ) | ( modify <valuePair> ) |
                  ( <funcall> )
<fact>        ::= ( <atom> <factSlot> <factSlots> )
<facts>       ::= <fact> <facts> | e
<factSlot>    ::= <orderedSlot> | <notOrderedSlot>
<factSlots>   ::= <orderedSlot> factSslots> | <notOrderedSlot>
                  <factSlots> | e
<orderedSlot> ::= <slot> <slots>
<notOrderedSlot> ::= ( <atom> <slot> <slots> )
<slot>        ::= <atom> | <string> | <variable> | <multivariable> |
                  <float> | <integer> | <funcall> | = <funcall>
<slots>       ::= <slot> <slots> | e
<LHS>         ::= <pattern> <LHS> | <variable> <- <pattern> <LHS> | e
<actions>     ::= <funcall> <actions> | e

<pattern>     ::= ( <patternValue> )
<patterns>    ::= <pattern> <patterns> | e
<patternValue> ::= <group> <patterns> | <test> <funcall> | <otherCE>
<group>       ::= not | and | or | exists | unique | explicit
<test>        ::= test
<otherCE>     ::= <atom> <slotCE>
<slotCE>      ::= <patternSlotDatas> | <orderedSlotData> |
                  ( <unorderedSlotData> ) | e
<patternSlotDatas> ::= <orderedSlotData> <orderedSlotDatas> |
                  ( <unorderedSlotData> ) <unorderedSlotDatas> | e
<orderedSlotDatas> ::= <orderedSlotData> <orderedSlotDatas> | e
<unorderedSlotDatas> ::= ( <unorderedSlotData> ) <unorderedSlotDatas> | e
<orderedSlotData> ::= <slotData>
<unorderedSlotData> ::= <atom> <slotData>
<slotData>    ::= <not> <slotData2> <and>
<slotData2>   ::= <variable> | <multivariable> | <atom> <column> |
                  <string> <funcall> | <float> <funcall> | <integer>
                  <funcall> | = <funcall>
<not>         ::= ~ | e
<and>         ::= & | e
```

```
<or>          ::= '|' | e
<column>      ::= : <funcall> | <funcall>

<defquery>    ::= ( defquery <atom> <docString> <declaration> <LHS> )
<deffacts>    ::= ( deffacts <atom> <docString> <facts> )

<deftemplate> ::= ( deftemplate <atom> <extends> <docString> <slotDescs> )
<extends>     ::= extends <atom> | e
<slotDescs>   ::= ( <slotType> ) <slotDesc>
<slotDesc>    ::= ( <slotType> ) <slotDesc> | e
<slotType>    ::= slot <atom> <slotQualifier> <slotQualifiers> |
  multislot <atom> <multiSlotQualifier> <multiSlotQualifiers>
<slotQualifier> ::= <multiSlotQualifier> | ( type <token> )
<slotQualifiers> ::= <slotQualifier> <slotQualifiers> | e
<multiSlotQualifier> ::= ( default <slotValue> ) | ( default-dynamic
                         <slotValue> )
<multiSlotQualifiers> ::= <multiSlotQualifier> <multiSlotQualifiers> | e
<slotValue>   ::= <atom> | <string> | <float> | <integer> | <funcall>

<deffunction> ::= ( deffunction <atom> <docString> ( <parameters> )
                  <docString> <values> )
<parameter>   ::= <variable> | <multiVariable>
<parameters>  ::= <parameter> <parameters> | e

<defglobal>   ::= ( defglobal * <variable> * = <values> )
<value>       ::= <atom> | <string> | <variable> | <multivariable> |
                  <float> | <integer> | <funcall>
<values>      ::= <value> <values> | e
```

Anhang B

Import von Java-Sourcecode in REVISER

Die Regelsprache zur Formulierung von Kriterien in REVISER basiert auf der funktionalen Sprache LISP. Zur Implementierung arithmetischer Berechnungen eignen sich vor allem imperative Programmiersprachen wie C++ oder Java. Aus diesem Grund ist in REVISER die Möglichkeit realisiert, Java-Sourcecode bzw. Java-Bytecode[1] einzubinden und die Methoden in der Regelsprache aufzurufen. Das prinzipielle Vorgehen dazu ist in Abbildung B.1 dargestellt und im Folgenden erläutert.

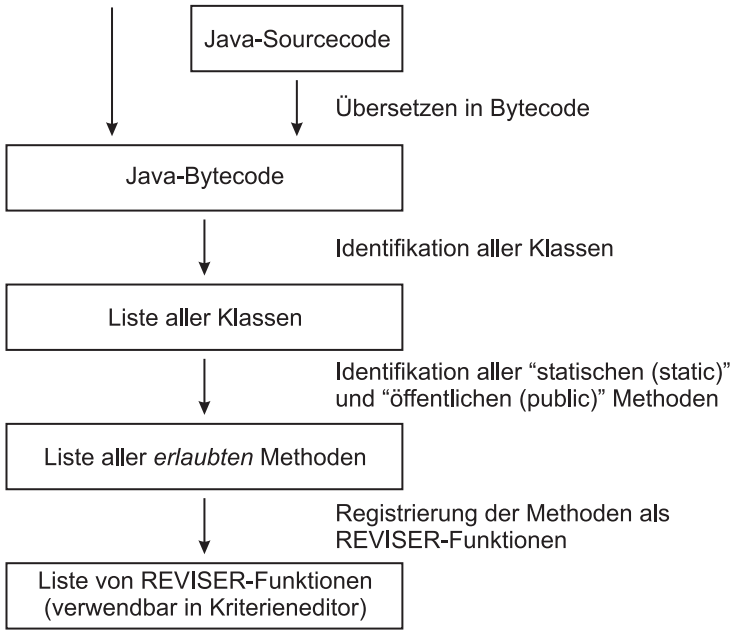

Abbildung B.1: Schritte zum Importieren von Java-Sourcecode und Registrieren der Java-Methoden im *Kriterieneditor* von REVISER.

Aus dem Java-Bytecode werden alle Klassen extrahiert und in einer Liste gespeichert. Anschließend erfolgt die Auswahl aller Methoden der gelisteten Klassen ebenfalls in Listen-

[1] Java-Sourcecode (gespeichert in „.java“-Dateien) wird standardmäßig durch einen Compile-Vorgang in einen plattformunabhängigen „Bytecode“ (gespeichert in „.class“-Dateien) übersetzt. Dieser Code lässt sich auf unterschiedlichen Plattformen direkt ausführen.

form. Dabei werden nur „statische (static)“ Methoden berücksichtigt, da per Definition nur auf diese Methoden ein direkter Zugriff erlaubt ist, ohne eine Instanziierung der Klasse zu fordern. Weiterhin müssen diese Methoden „öffentlich (public)“ sein, da nur bei öffentlichen Methoden ein Zugriff von ausserhalb der Klasse explizit erlaubt ist und somit das Prinzip der Datenabstraktion der in Java programmierten Funktionalität erhalten bleibt.

Als Einschränkung für dieses Vorgehen ist die Eindeutigkeit der Methodennamen gefordert. Jeder Name einer Methode darf lediglich einmal vorkommen, unabhängig davon, aus welcher Klasse die Methode stammt oder ob sie überladen[2] wurde. Die extrahierten Methoden werden im letzten Schritt als *Funktionen* in der Regelsprache registriert und stehen für einen Aufruf in der Konklusion zur Verfügung.

Durch dieses Vorgehen lassen sich neben umfangreichen Berechnungen auch ganze Programmbibliotheken nutzen und ansprechen. Die gesamte Funktionalität der Java-Bibliotheken steht somit zur Verfügung, der Zugriff muss lediglich über eine zu registrierende *public, static*-Methode gekapselt sein. Java-Bytecode kann durch Angabe der class-Dateien in REVISER eingebunden werden. Zur Realisierung einfacher Methoden bietet der Kriterieneditor darüber hinaus eine Eingabemöglichkeit für Java-Sourcecode, der innerhalb des Werkzeugs automatisch in Bytecode umgewandelt und in der Regelsprache zur Verfügung gestellt wird.

[2] *Überladen* bezeichnet die Mehrfachbenutzung eines Methodennamens. Die Wahl, welche Methode zur Laufzeit benutzt und ausgeführt wird hängt dabei ausschließlich von der Art und Anzahl der angegebenen Parameter ab.

Anhang C

Ontologien

Dieser Anhang stellt die in dieser Arbeit entstandenen Ontologie vor. Dazu erfolgt zunächst eine Einführung in die Grundlagen von Ontologien. Anschließend ist erläutert, wie die Ontologie von Begriffen und deren Hierarchie für Fahrerinformationssysteme erstellt wurde. Abschließend ist der Inhalt der Ontologie beispielhaft dargestellt.

C.1 Grundlagen der Ontologien

Ontologien sind formale und maschinenlesbare Spezifikationen von Objekten bestimmter Anwendungsbereiche und deren semantischer Beziehungen zueinander (GRUBER 1993). Anwendungsgebiete sind vor allem Softwareagenten und die semantische Suche. In beiden Anwendungen werden vor allem unterschiedliche Begriffe und deren semantische Zusammenhänge mit Hilfe von Ontologien festgelegt (HUHNS und SINGH 1997). Die Beziehungen müssen empirisch erhoben und ggf. durch *automatisches Schließen* erweitert werden (MAGKANARAKI et al. 2002).

C.1.1 Bestandteile einer Ontologie

Die formale Beschreibung von Ontologien besteht aus *Klassen*, *Beziehungen*, *Eigenschaften*, *Einschränkungen* und *Instanzen*, die im Folgenden vorgestellt werden.

Klassen (auch Konzept, Begriff, Entität) repräsentieren eine Gruppe von individuellen Objekten, die konzeptuelle Gemeinsamkeiten aufweisen. So ist ein Burgunder- und ein Frascati-Wein beides *Wein*.

Beziehungen (auch Relations) ermöglichen das Verhältnis von Klassen zueinander anzugeben. Dazu gehören die Beziehungen (ein *Wein* wird aus einer oder mehreren *Traubensorten* hergestellt) sowie die Vererbung, bei der eine (Unter-)Klasse Eigenschaften von

einer (Ober-)Klasse erbt, sie durch die Angabe weiterer Eigenschaften jedoch zusätzlich einschränkt. So verfeinert die Klasse *Qualitätswein* die Klasse *Wein* durch die Angabe, dass die Trauben in einem einzigen Anbaugebiet geerntet wurden. In der Klasse *Qualitätswein* muss jedoch keine Angabe mehr gemacht werden, dass es sich bei der Flüssigkeit um Traubensaft handelt, diese Eigenschaft ist von *Wein* geerbt.

Eigenschaften (auch Attribut, Slot, Role, Property) beschreiben bzw. verfeinern Klassen. So ist „*rot*“ und „*Alkoholgehalt*“ eine Eigenschaft von *Wein*.

Einschränkungen (auch Facette, Nebenbedingung) sind Restriktionen der Eigenschaften. Sie geben u.a. Auskunft über die Kardinalität oder den Typ einer Eigenschaft. So muss bei *Qualitätswein* das Anbaugebiet (im Typ String oder als Instanz einer eigenen Klasse) angegeben werden. Die Angabe der Süße des Weins ist dagegen optional.

Instanz (auch Objekt, Individuum, Exemplar) stellt ein individuelles Element einer Klasse dar. Während der *Instanziierung* werden den Klassen und ihren Attributen konkrete Werte zugewiesen.

C.1.2 Schritte bei der Modellierung von Ontologien

Eine in der Wissensrepräsentation bewährte Methode zur Modellierung einer Ontologie sieht eine strukturelle Unterteilung in sieben Schritte vor. Dabei sind diese Schritte nicht streng seriell abzuarbeiten, sondern können Iterationen enthalten. Gerade die Schritte vier und fünf sind kaum getrennt voneinander durchführbar und liefern erst nach mehrfachem Bewerten ein brauchbares Ergebnis (NOY und MCGUINNESS 2001).

1. **Fachgebiet und Reichweite der Ontologie festlegen**
   Zu Beginn muss die Domäne, für die die Ontologie erstellt werden soll, mit allen relevanten Begriffen genau festgelegt werden.

2. **Benutzung existierender Ontologien abwägen**
   Im Lauf der Zeit sind zahlreiche Ontologien für verschiedene Anwendungszwecke erstellt worden. Eine Wiederverwendung einer bestehenden Ontologie spart Zeit und Geld.

3. **Zentrale Begriffe identifizieren**
   In diesem Schritt werden die für eine Ontologie wesentlichen Begriffe identifiziert und gesammelt. Dazu kommen u.a. Kreativtechniken wie Brainstorming zum Einsatz (BONATO 1990; LÖWGREN und STOLTERMAN 1999).

4. **Klassen und Klassenhierarchien bilden**
   Aus den in Schritt 3 ermittelten Begriffen werden diejenigen ausgewählt, die weder eine Ausprägung noch ein Merkmal beschreiben, Synonyme sind oder eine Beziehungen zwischen Begriffen darstellen (LICHTER 2005). Die so erhaltenen Begriffe sind

potentielle Kandidaten für Klassen, die die Struktur der Ontologie definieren. Eine Definition von Vererbungen und Klasse-Oberklasse Beziehungen schließt diesen Schritt ab.

5. **Eigenschaften der Klassen festlegen**
   Die Klassen werden durch Eigenschaften (Attribute) verfeinert und beschrieben. Solche Eigenschaften stellen ebenfalls Begriffe (z.B. Adjektive) dar.

6. **Einschränkungen der Eigenschaften definieren**
   Vorgaben und Einschränkungen von Eigenschaften werden benutzt, um diese zu formalisieren und auf diese Weise maschinenlesbar zu machen.

7. **Instanzen erschaffen**
   Der letzte Schritt dient der Generierung der eigentlichen Objekte als Instanzen einer Ontologie (im Gegensatz zu den Klassen als Strukturdefinition). Dazu müssen entsprechende Begriffe für die Klassen und die Eigenschaften unter Berücksichtigung der Vorgaben und Einschränkungen angegeben werden. Eine gute Grundlage dazu bilden die im dritten Schritt bereits ermittelten Begriffe (BONATO 1990).

C.2 Erstellung der in dieser Arbeit entwickelten Ontologie

Gemäß dem in Abschnitt C.1.2 beschriebenen Vorgehen wurde im Rahmen dieser Arbeit eine Ontologie für Begriffe eines Fahrerinformationssystems (FIS) erstellt. Ziel der Erstellung war die semantische Anordnung dieser Begriffe in Form einer Begriffshierarchie. Ein *Oberbegriff* wird dabei von *Unterbegriffen* semantisch spezialisiert. Solche Beziehungen finden zur Beschreibung von Gerätefunktionen als Menüoptionen in interaktiven Geräten Benutzung. So ist in einem Mobiltelefon die Funktion, die durch den Begriff *Rufumleitung* beschrieben ist, unter dem Begriff *Anrufeinstellungen*, der wiederum unter *Einstellungen* eingeordnet.

Im Folgenden ist das entwickelte Format der Ontologie (als Ergebnis der Schritte 1-6 aus Abschnitt C.1.2) sowie das Vorgehen zur Generierung von Begriffen (als Ergebnis aus Schritt 7) beschrieben.

C.2.1 Festlegung des Formats der FIS-Ontologie

Da in dieser Arbeit lediglich Begriffe in semantische Abhängigkeiten gebracht und durch Synonyme erweitert werden sollen, reicht eine Klassenstruktur bestehend aus zwei Klassen aus, die im Folgenden beschrieben und deren Aufbau als Entity-Relationship-Diagram[1] in Abbildung C.1 dargestellt ist.

Die Klasse `WordClass` kapselt einen Begriff und dessen Attribute. `id` ist der eindeutige Identifier der Klasse. `name` stellt den zu fassenden Begriff dar (z.B. *CD-Spieler, CD-Player*),

[1] s. Glossar

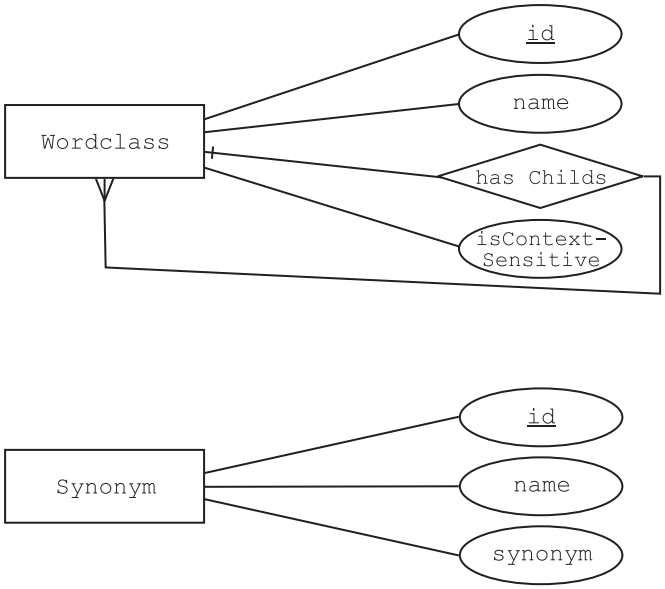

Abbildung C.1: Entity-Relationship-Diagram der Struktur der eingesetzten Ontologie.

wobei ein Begriff nicht zwangsläufig aus lediglich einem Wort bestehen muss. Im Gegensatz zur Definition des Wortes *Begriff* ist hier ein sinnhafter Begriff auch aus mehreren Worten wie z.B. *Navigation starten* gemeint. `hasChilds` verbindet wiederum Begriffe, die semantisch unterhalb dieses Begriffs angeordnet werden, diesen also weiter einschränken (z.B. *CD-Wechsler, CD abspielen, Eject*). `isContextSensitive` beschreibt, ob eine Begriffsmenge für sich genommen seinen Kontext definieren kann oder nicht (*CD-Spieler* ist eindeutig, *abspielen* nicht, da dieses Wort auch für Musikkassetten und MP3-Dateien gelten kann).

`Synonym` beschreibt ein Synonym zu einem Begriff. Eine Instanz von `Synonym` wird ebenfalls durch einen eindeutigen Identifier `id` beschrieben. `name` gibt den Begriff an, zu dem `synonym` einen synonymen Begriff definiert. Die Synonymdefinition ist bewusst nicht in `WordClass` integriert, da so eine größere Flexibilität erreicht wird.

C.2.2 Experimentelle Erfassung der Begriffe der FIS-Ontologie

Für den Aufbau der Ontologie standen 13 Personen[2] zwischen 25 und 40 Jahren zur Verfügung. Alle verfügen über einen Führerschein (mind. Klasse 3) und haben im Durchschnitt 13 Jahre Fahrerfahrung. Das Bedienen von Autoradios sowie Navigationssystemen ist allen Personen geläufig. Drei Personen stufen sich als Anfänger, fünf als Fortgeschrittene und fünf als Experten im Umgang mit solchen Systemen ein.

Die Erfassung der Begriffe gliedert sich in zwei Schritte, der Begriffsidentifikation und der Hierarchiebildung, die mit jeder Gruppe durchgeführt wurde.

Die Implementierung der Ontologie erfolgt in dem Werkzeug *Protégé* der Universität Stan-

[2]Bei den Personen zur Erstellung der FIS handelte es sich nicht um die Probanden zur Durchführung der empirischen Bewertung.

ford[3], das sich als De-Facto-Standard durchgesetzt hat (VAN WELIE und VAN DER VEER 1999). Die Ontologie wird dabei in der *Ontology Web Language (OWL)*[4] formal beschrieben. Die OWL erlaubt eine XML-konforme Angabe von Begriffsklassen mit den entsprechenden Eigenschaften sowie Vererbungsbeziehungen zu weiteren Klassen. Die Sprachkonstrukte sind sehr umfangreich und werden daher in dieser Arbeit nicht vorgestellt.

Zur Bewertung des Informationsdesigns sowie zur Navigation in menügestützten interaktiven Systemen genügt statt Begriffsgraphen ein Begriffsbaum. Das allgemeine Vorgehen bei der Erstellung einer Ontologie sowie das in dieser Arbeit entwickelte Ontologieformat ist im Anhang C.1.2 gegeben. Bei der Umwandlung eines Begriffsgraphen in einen Begriffsbaum kann es vorkommen, dass Knoten mehrmals vorkommen. Das ist jedoch zur Bewertung des Informationsdesigns nicht hinderlich und stellt daher keinen Nachteil dar.

Begriffsidentifikation mittels Brainstorming

Die 13 Personen wurden in fünf Gruppen zu je zwei bis drei Personen eingeteilt. In jeder dieser Gruppe erfolgte ein Gruppenbrainstorming von unbeschränkter Länge zum reinen Sammeln der Begriffe. Ziel war die Identifikation von Begriffen, die mit der Funktionalität von FIS in Zusammenhang stehen. Dazu gehören ebenfalls gängige Zeichen (Icons) sowie Synonyme bereits vorhandener Begriffe. Insgesamt wurden 613 Begriffe identifiziert.

Erstellung einer Hierarchie der erfassten Begriffe

Nach dem reinen Sammeln erfolgte in den jeweiligen Gruppen die semantische Zuordnung der Begriffe in hierarchischer Form (s.o.). Jeder im Brainstorming identifizierte Begriff musste dabei in die Hierarchie eingefügt werden. In diesem Schritt erfolgte gleichzeitig die Zuordnung synonymer Begriffe. Insgesamt wurden 356 Begriffe in der Hierarchie verwendet und 257 Synonyme zugeordnet.

C.3 Inhalt der in dieser Arbeit erstellten Ontologie für Begriffe von Fahrerinformationssystemen

Im Folgenden ist die in dieser Arbeit entwickelte Ontologie in Form eines Baumes vorgestellt. Synonyme Begriffe sind durch Semikolon getrennt. In Abbildung C.2 ist der Wurzelknoten *idle* mit der ersten Ebene der Begriffe eines FIS dargestellt. Aufgrund des Umfangs der Ontologie ist beispielhaft der Teilbaum *Entertainment* in den Abbildungen C.3, C.4 und C.5 dargestellt.

[3] http://protege.stanford.edu
[4] s. Glossar

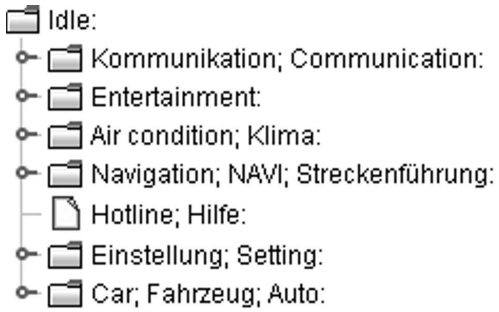

Abbildung C.2: Darstellung der Ontologie für ein FIS als Baum (Wurzelknoten *idle* und erste Ebene).

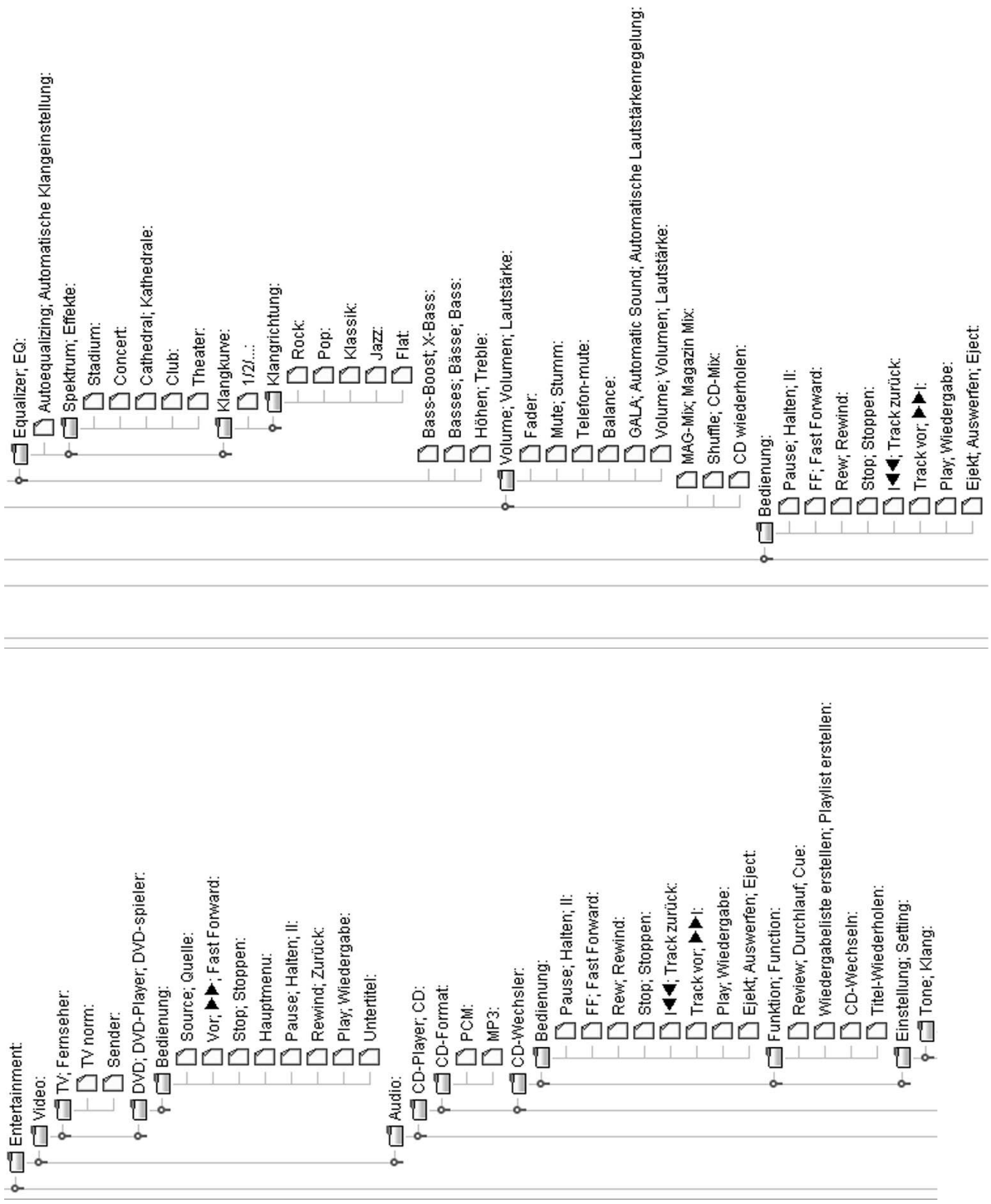

Abbildung C.3: Darstellung des Teilbaums *Entertainment* der Ontologie für ein FIS.

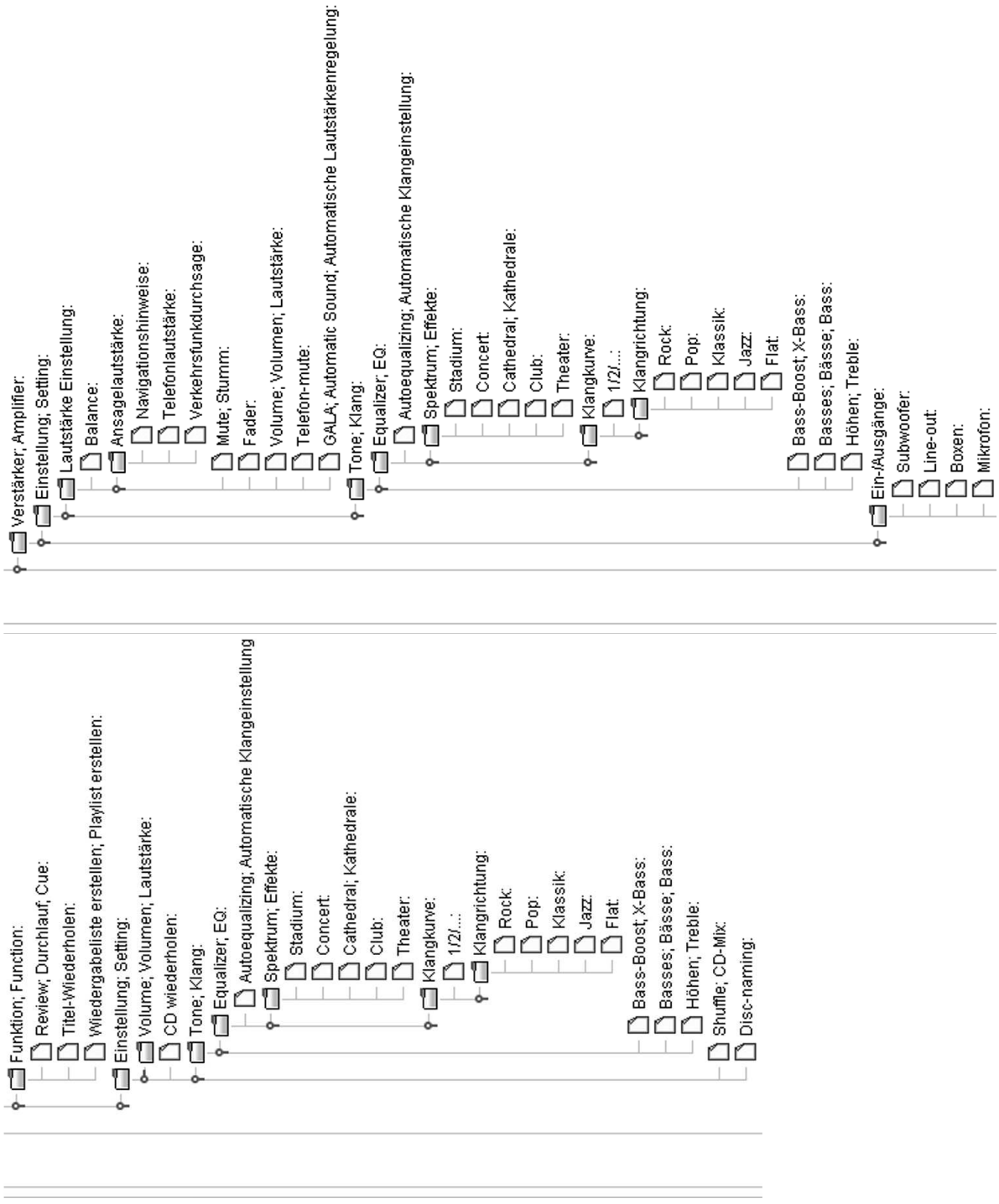

Abbildung C.4: Darstellung des Teilbaums *Entertainment* der Ontologie für ein FIS (Fortsetzung).

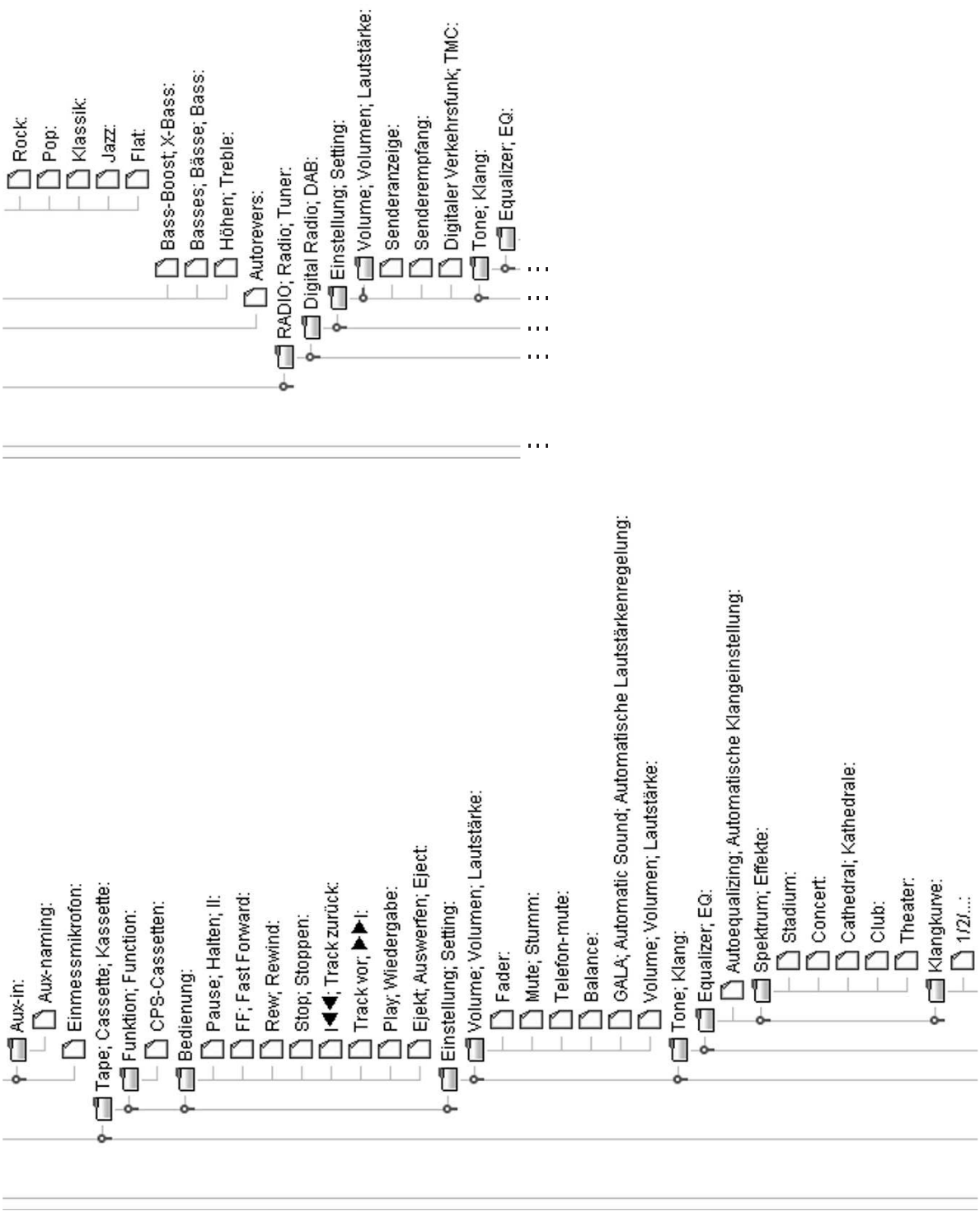

Abbildung C.5: Darstellung des Teilbaums *Entertainment* der Ontologie für ein FIS (Fortsetzung).

Anhang D

Dialogfolgen der Aufgaben zur Bewertung der FIS Audi MMI und BMW

Zur Evaluierung von REVISER werden in Kapitel 6 die beiden Fahrerinformationssysteme Audi MMI und BMW iDrive anhand der Darstellung und Bedienung jeweils dreier Aufgaben bewertet. Diese drei Aufgaben sind:

- Sender „WDR3“ einstellen
- Menü zur Einstellung der Klanghöhen aufrufen
- Zweites Navigationsziel aus Zielliste auswählen

In diesem Abschnitt sind die Dialogfolgen der Aufgaben vorgestellt. Die Dialogfolgen enthalten die GUI-Konstellation eines jeden Systemzustands sowie die Aktionen zur Bedienung des Systems. Jeder dargestellte Dialogschritt entspricht der Darstellung eines neuen Menüs nach einem Menüwechsel; Änderungen innerhalb eines Menüs sind nicht extra dargestellt.

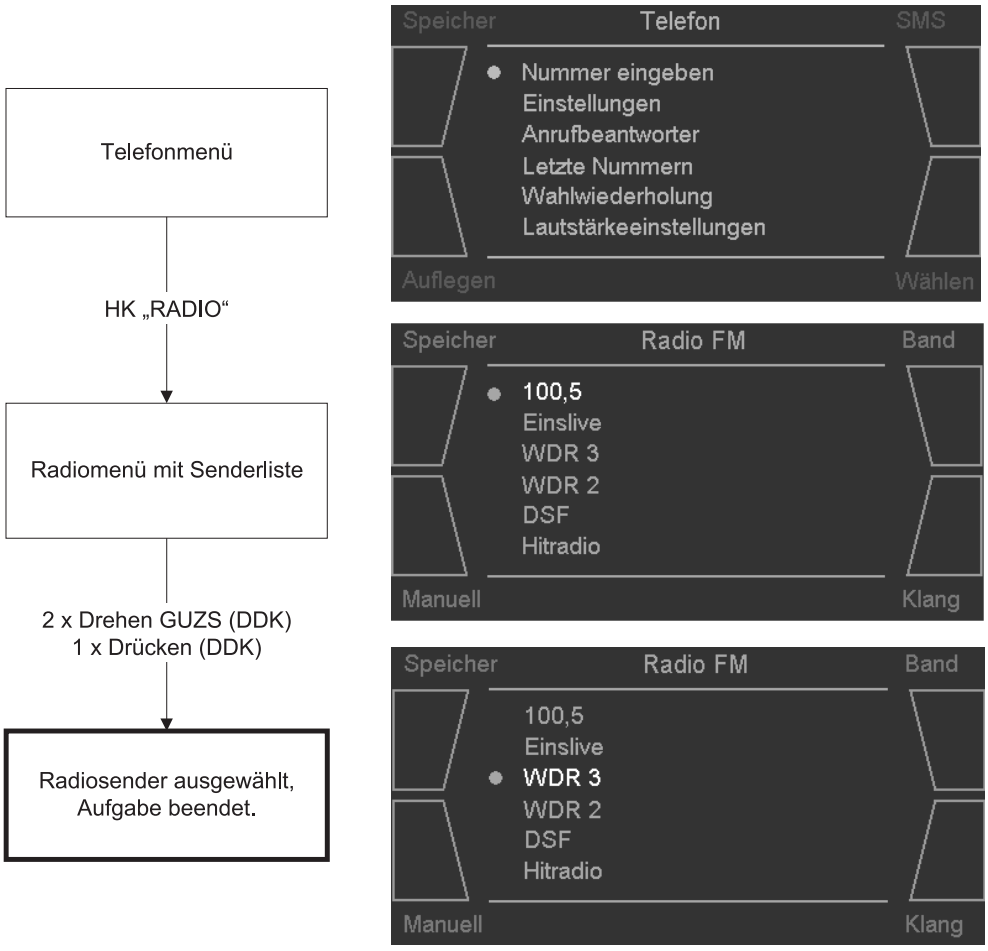

Abbildung D.1: Dialogfolge für die Aufgabe „Sender ‚WDR3‘ einstellen für das Audi MMI.

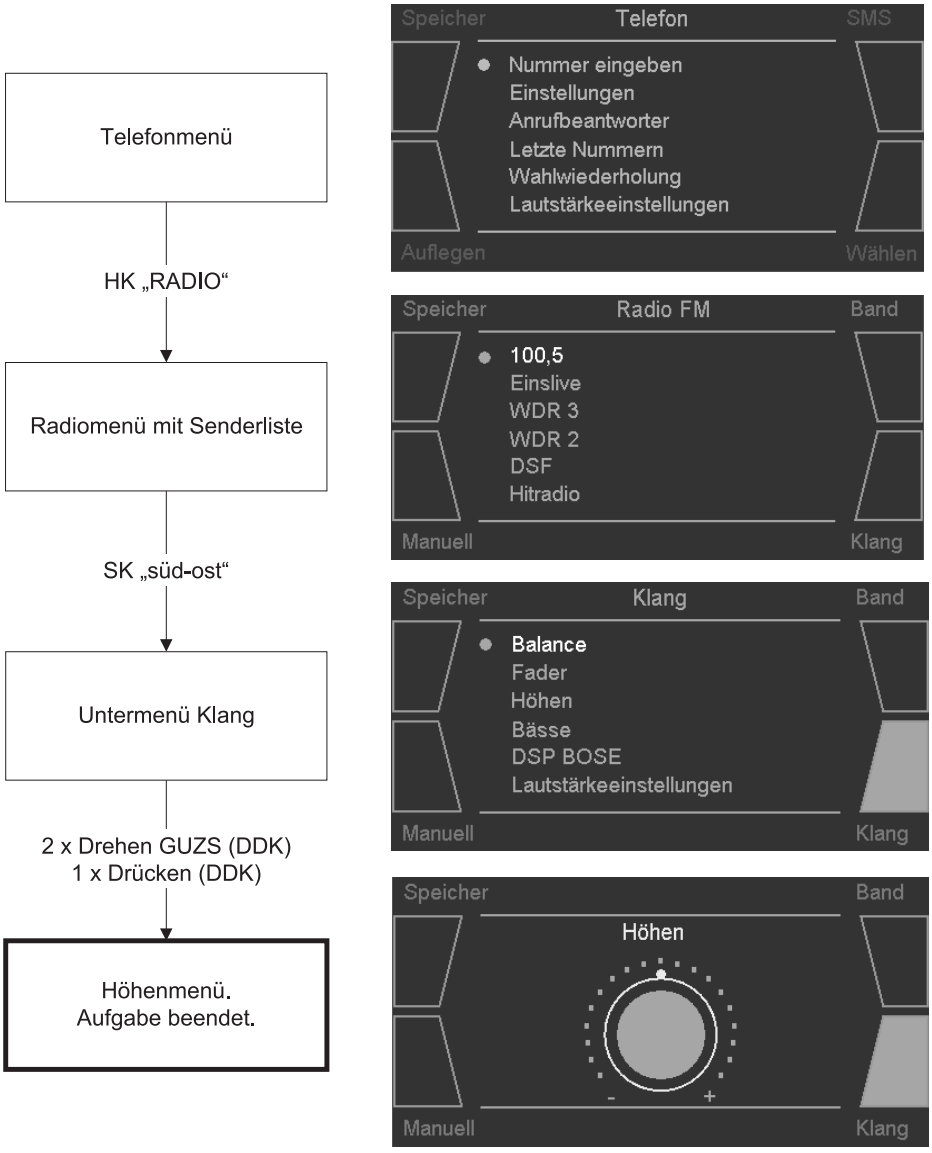

Abbildung D.2: Dialogfolge für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das Audi MMI.

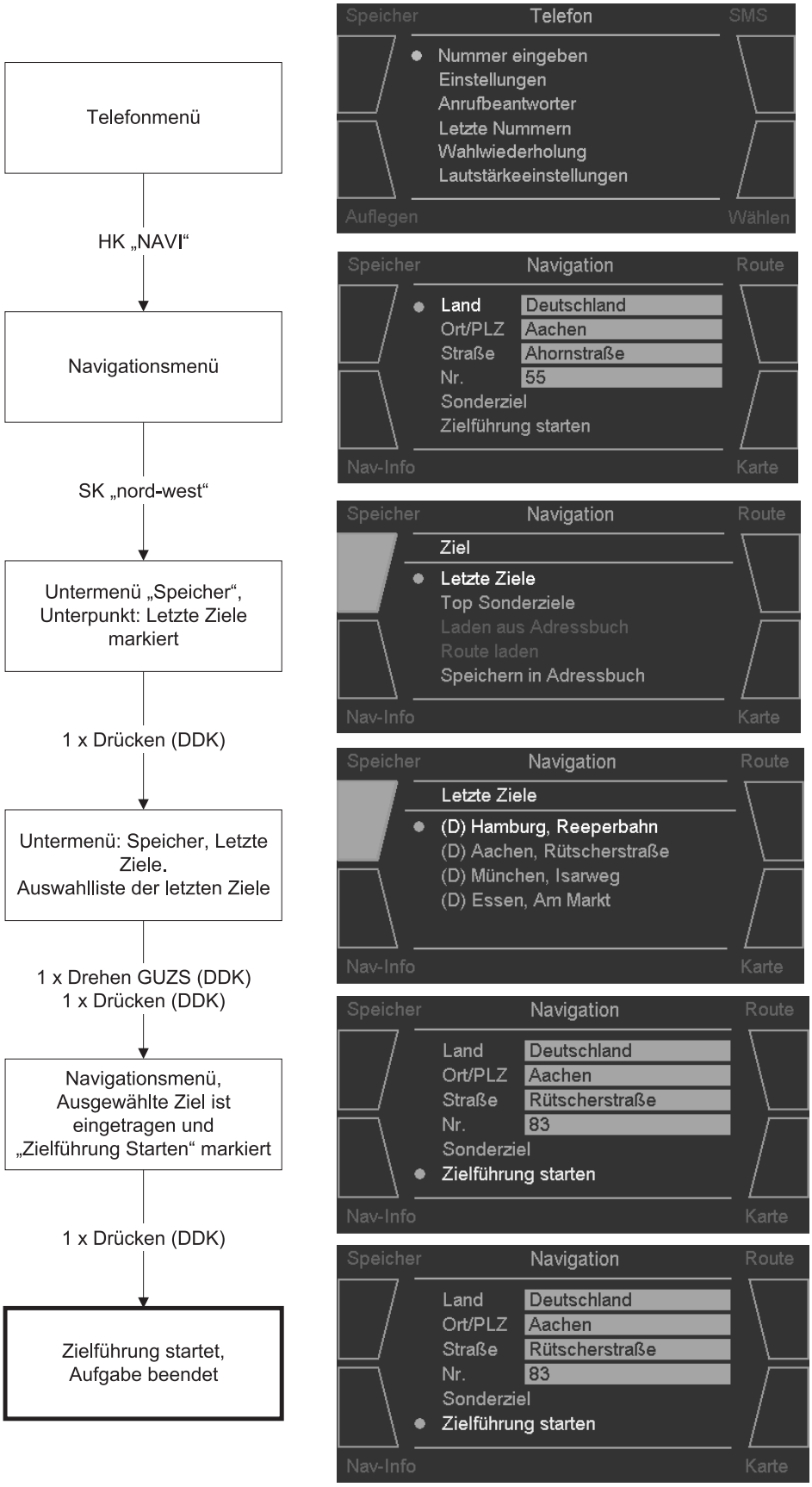

Abbildung D.3: Dialogfolge für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das Audi MMI.

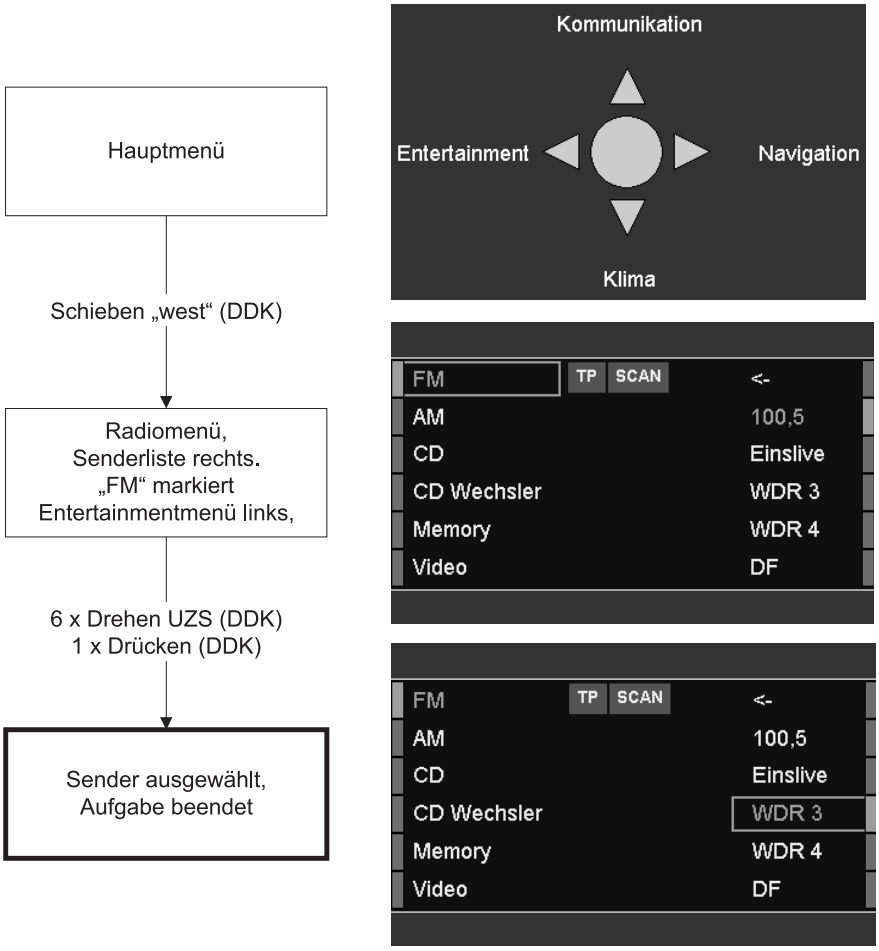

Abbildung D.4: Dialogfolge für die Aufgabe „Sender ‚WDR3‘ einstellen“ für das BMW iDrive.

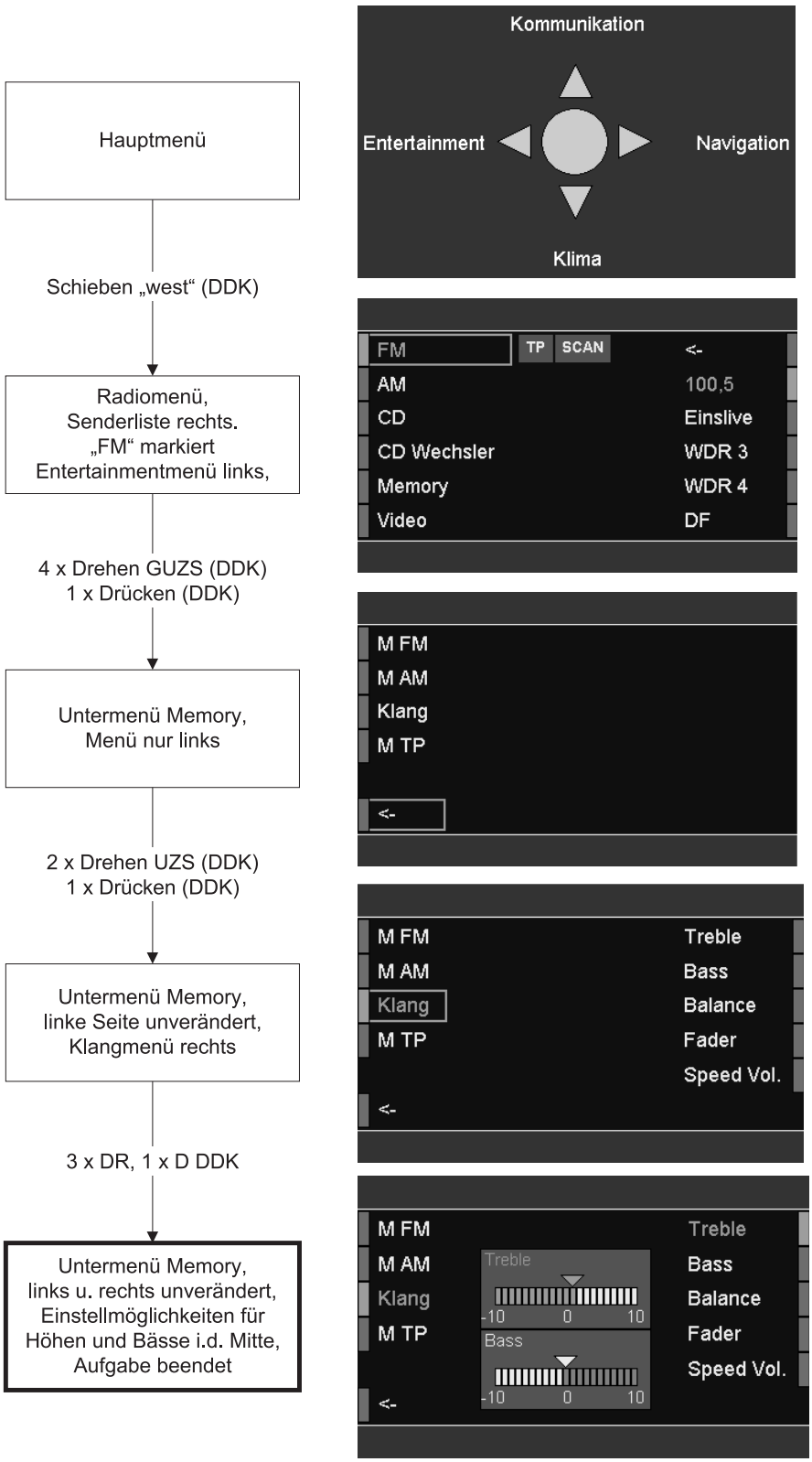

Abbildung D.5: Dialogfolge für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das BMW iDrive.

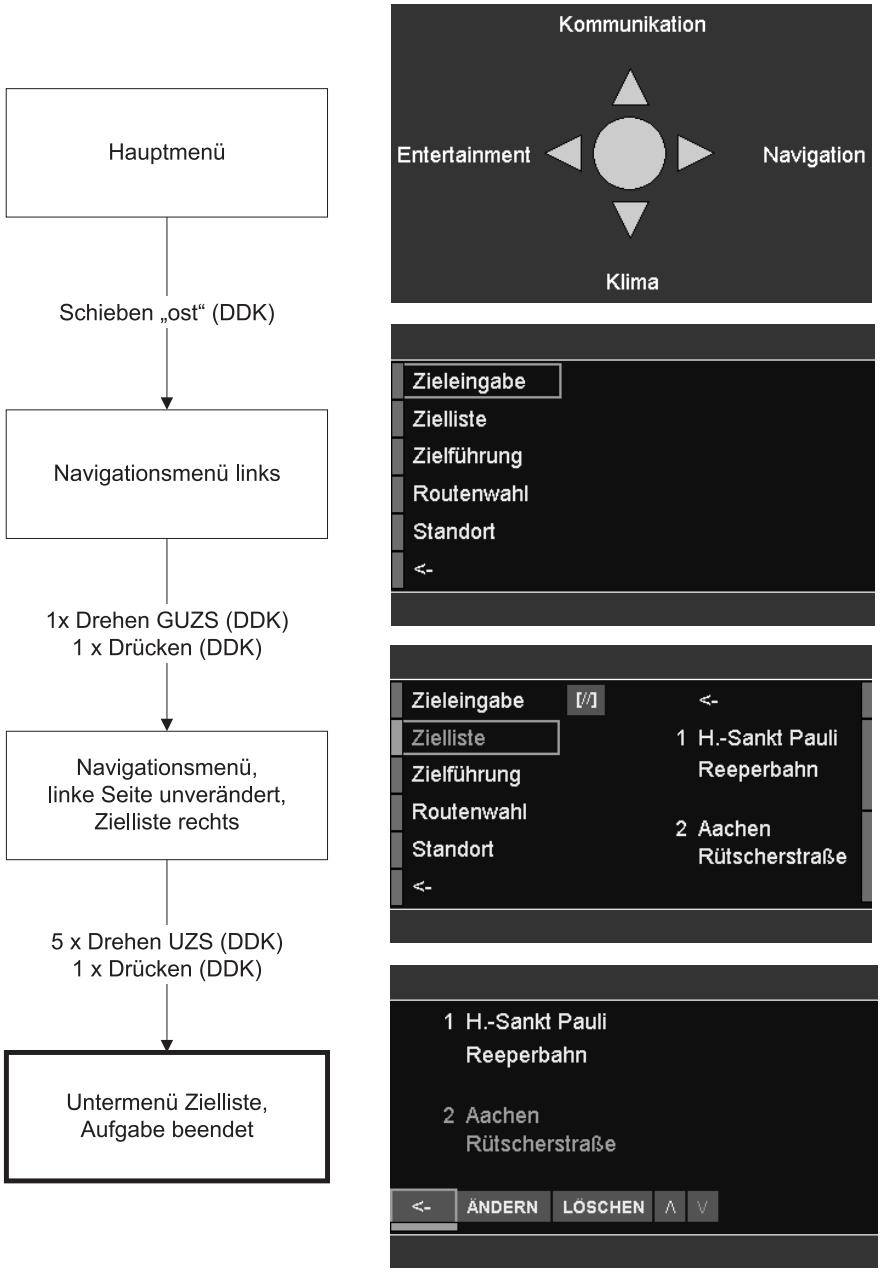

Abbildung D.6: Dialogfolge für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das BMW iDrive.

Anhang E

Ergebnisse der automatischen Bewertung des Oberflächendesigns

Während der automatischen Bewertung der beiden FIS Audi MMI und BMW iDrive durch REVISER wurden die in Kapitel 3.4.2 vorgestellten Maße erhoben. Alle Maße mit einem Zahlenwert als Ergebnis sind jeweils für die Dialogfolgen im Folgenden vorgestellt. Anschließend erfolgt die tabellarische Darstellung der zu den jeweiligen Maßen berechneten Konsistenzmaße *Z-Score* und die *CoV* (s. Kapitel 3.4.2.4).

E.1 Ergebnisse der Maße des Oberflächendesigns

In den folgenden Abbildungen sind die Ergebnisse der Bewertung des Oberflächendesigns der FIS Audi MMI und BMW iDrive für jede Dialogfolge tabellarisch dargestellt.

Maß	Schritt 1	Schritt 2	Schritt 3	Schritt 4
background color count	2,00	2,00	2,00	2,00
column count	3,00	3,00	3,00	3,00
display aspect ratio	0,50	0,50	0,50	0,50
foreground color count	4,00	4,00	4,00	4,00
gridedness	36,17	39,13	39,13	39,13
horizontal area-balance	-2,73	-23,16	-23,16	-23,16
margin east	-1,00	-1,00	-1,00	-1,00
margin north	1,00	1,00	1,00	1,00
margin south	6,00	6,00	6,00	6,00
margin west	1,00	1,00	1,00	1,00
non widget area	59,39	70,29	70,29	70,29
selected item count	1,00	1,00	1,00	1,00
text widget count	10,00	11,00	11,00	11,00
vertical area-balance	-1,30	-17,65	-17,65	-17,65
widget count	17,00	18,00	18,00	18,00
widget density	14,76	15,63	15,63	15,63
word count	12,00	14,00	14,00	14,00

Abbildung E.1: Werte der Oberflächenmaße für die Aufgabe „Sender ‚WDR3‘ einstellen“ für das Audi MMI.

Maß	Schritt 1	Schritt 2	Schritt 3	Schritt 4	Szene 5
background color count	2,00	2,00	2,00	2,00	2,00
column count	3,00	3,00	3,00	3,00	3,00
display aspect ratio	0,50	0,50	0,50	0,50	0,50
foreground color count	4,00	4,00	4,00	4,00	4,00
gridedness	36,17	39,13	42,55	42,55	42,55
horizontal area-balance	-1,30	-17,60	-11,70	-10,80	-11,60
margin east	-1,00	-1,00	-1,00	-1,00	-1,00
margin north	1,00	1,00	1,00	1,00	1,00
margin south	6,00	6,00	6,00	6,00	6,00
margin west	1,00	1,00	0,00	0,00	0,00
non widget area	59,39	70,29	70,12	70,12	70,12
selected item count	1,00	1,00	1,00	1,00	1,00
text widget count	10,00	11,00	11,00	11,00	11,00
vertical area-balance	-2,70	-23,80	-25,40	-24,60	-25,60
widget count	17,00	18,00	20,00	20,00	20,00
widget density	14,70	15,62	17,30	17,30	17,30
word count	12,00	14,00	12,00	12,00	12,00

Abbildung E.2: Werte der Oberflächenmaße für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das Audi MMI.

Maß	Schritt 1	Schritt 2	Schritt 3	Schritt 4	Schritt 5
background color count	2,00	2,00	2,00	2,00	2,00
column count	3,00	4,00	3,00	3,00	3,00
display aspect ratio	0,50	0,50	0,50	0,50	0,50
foreground color count	4,00	4,00	6,00	5,00	5,00
gridedness	36,17	42,31	37,04	37,25	37,25
horizontal area-balance	-2,73	-20,52	-26,67	-14,55	-14,51
margin east	-1,00	-1,00	-1,00	-1,00	-1,00
margin north	1,00	1,00	1,00	1,00	1,00
margin south	6,00	6,00	6,00	6,00	6,00
margin west	1,00	1,00	0,00	0,00	0,00
non widget area	59,39	48,04	59,66	56,18	56,18
selected item count	1,00	1,00	1,00	1,00	1,00
text widget count	10,00	15,00	11,00	10,00	10,00
vertical area-balance	-1,30	-25,75	-28,58	-45,22	-45,47
widget count	17,00	22,00	20,00	19,00	19,00
widget density	14,76	19,10	17,36	16,49	16,49
word count	12,00	15,00	17,00	19,00	19,00

Abbildung E.3: Werte der Oberflächenmaße für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das Audi MMI.

Maß	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	-2,31	0,00	1,15	1,15	0,00	0,00	0,00	0,00
display aspect ratio	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
foreground color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
gridedness	-2,58	0,57	0,10	0,10	0,57	0,11	0,57	0,57
horizontal area-balance	-1,87	-0,83	-0,50	-0,23	0,22	1,05	1,12	1,05
margin east	-2,65	0,38	0,38	0,38	0,38	0,38	0,38	0,38
margin north	2,65	-0,38	-0,38	-0,38	-0,38	-0,38	-0,38	-0,38
margin south	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
margin west	-2,65	0,38	0,38	0,38	0,38	0,38	0,38	0,38
non widget area	2,63	-0,52	-0,11	-0,45	-0,39	-0,39	-0,39	-0,39
selected item count	-2,65	0,38	0,38	0,38	0,38	0,38	0,38	0,38
text widget count	-2,63	0,42	0,42	0,42	0,42	0,14	0,42	0,42
vertical area-balance	-1,57	-0,22	0,06	-0,29	-0,03	1,50	-0,95	1,50
widget count	-2,61	0,45	0,45	0,45	0,45	-0,06	0,45	0,45
widget density	-2,63	0,43	0,43	0,43	0,43	0,05	0,43	0,43
word count	-2,63	0,38	0,38	0,38	0,38	0,16	0,38	0,59

Abbildung E.4: Werte der Oberflächenmaße für die Aufgabe „Sender ‚WDR3‘ einstellen“ für das BMW iDrive.

Maß	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8	S. 9	S. 10
background color count	3,00	3,00	3,00	3,00	3,00	3,00	3,00	3,00	3,00	3,00
column count	0,00	2,00	3,00	2,00	2,00	2,00	2,00	2,00	1,00	1,00
display aspect ratio	0,63	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48
foreground color count	2,00	2,00	2,00	2,00	2,00	2,00	2,00	2,00	1,00	1,00
gridedness	30,00	55,56	51,72	55,56	55,56	55,56	55,56	55,56	35,71	45,45
horizontal area-balance	-3,61	3,42	5,59	3,42	3,42	3,42	-8,30	-2,50	-100,00	-100,00
margin east	6,00	41,00	41,00	41,00	41,00	41,00	41,00	41,00	335,00	335,00
margin north	7,00	6,00	6,00	6,00	6,00	6,00	6,00	6,00	6,00	6,00
margin south	8,00	8,00	8,00	8,00	8,00	8,00	8,00	8,00	8,00	70,00
margin west	6,00	19,00	19,00	19,00	19,00	19,00	19,00	19,00	19,00	19,00
non widget area	88,19	88,19	72,10	74,22	72,10	72,10	72,10	72,10	72,10	89,78
selected item count	0,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
text widget count	4,00	15,00	15,00	15,00	15,00	15,00	15,00	15,00	5,00	5,00
vertical area-balance	-10,91	-0,88	1,20	-0,88	-0,88	-0,88	17,93	12,74	-38,69	-22,12
widget count	9,00	15,00	15,00	15,00	15,00	15,00	15,00	15,00	5,00	5,00
widget density	9,00	19,74	19,74	19,74	19,74	19,74	19,74	19,74	6,58	6,58
word count	4,00	18,00	18,00	18,00	18,00	18,00	19,00	18,00	8,00	9,00

Maß	S. 11	S. 12	S. 13	S. 14	S. 15	S. 16	S. 17	S. 18	S. 19
background color count	3,00	3,00	3,00	3,00	3,00	3,00	3,00	3,00	3,00
column count	1,00	2,00	2,00	2,00	2,00	2,00	2,00	2,00	2,00
display aspect ratio	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48	0,48
foreground color count	1,00	2,00	2,00	2,00	2,00	2,00	2,00	2,00	2,00
gridedness	45,45	52,63	52,63	52,63	52,63	52,63	52,63	52,63	52,63
horizontal area-balance	-100,00	-9,13	-9,13	-9,13	-9,13	-9,13	12,53	12,53	12,53
margin east	335,00	43,00	43,00	43,00	43,00	43,00	43,00	43,00	43,00
margin north	6,00	6,00	6,00	6,00	6,00	6,00	6,00	6,00	6,00
margin south	70,00	8,00	8,00	8,00	8,00	8,00	8,00	8,00	8,00
margin west	19,00	19,00	19,00	19,00	19,00	19,00	19,00	19,00	19,00
non widget area	90,27	90,27	81,90	81,90	81,90	81,90	81,90	81,90	80,48
selected item count	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
text widget count	5,00	10,00	10,00	10,00	10,00	10,00	10,00	10,00	10,00
vertical area-balance	-61,40	-51,89	-51,89	-51,89	-51,89	-51,89	-52,53	-52,53	-52,53
widget count	5,00	10,00	10,00	10,00	10,00	10,00	10,00	10,00	10,00
widget density	6,58	13,16	13,16	13,16	13,16	13,16	13,16	13,16	13,16
word count	8,00	13,00	13,00	14,00	14,00	14,00	14,00	13,00	13,00

Abbildung E.5: Werte der Oberflächenmaße für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das BMW iDrive.

Maß	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8
background color count	3,00	3,00	3,00	3,00	3,00	3,00	3,00	3,00
column count	0,00	1,00	1,00	1,00	2,00	2,00	2,00	2,00
display aspect ratio	0,63	0,48	0,48	0,48	0,48	0,48	0,48	0,48
foreground color count	2,00	1,00	1,00	1,00	2,00	2,00	2,00	2,00
gridedness	30,00	37,50	37,50	37,50	38,71	38,71	38,71	38,71
horizontal area-balance	-3,61	-100,0	-100,0	-100,0	-34,86	-34,86	-34,86	-37,56
margin east	6,00	286,00	286,00	286,00	30,00	30,00	30,00	30,00
margin north	7,00	6,00	6,00	6,00	6,00	6,00	6,00	6,00
margin south	8,00	39,00	39,00	39,00	8,00	8,00	8,00	8,00
margin west	6,00	19,00	19,00	19,00	19,00	19,00	19,00	19,00
non widget area	88,19	81,23	81,23	81,23	70,04	70,04	70,04	70,04
selected item count	0,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
text widget count	4,00	6,00	6,00	6,00	12,00	12,00	12,00	12,00
vertical area-balance	-10,91	-34,80	-34,80	-30,34	-51,35	-51,35	-51,35	-53,36
widget count	9,00	6,00	6,00	6,00	12,00	12,00	12,00	12,00
widget density	9,00	7,89	7,89	7,89	15,79	15,79	15,79	15,79
word count	4,00	6,00	6,00	6,00	13,00	13,00	13,00	13,00

Maß	S. 9	S. 10	S. 11	S. 12	S. 13	S. 14	S. 15
background color count	3,00	3,00	3,00	3,00	3,00	3,00	3,00
column count	3,00	3,00	2,00	2,00	3,00	3,00	3,00
display aspect ratio	0,48	0,48	0,48	0,48	0,48	0,48	0,48
foreground color count	2,00	2,00	2,00	2,00	2,00	2,00	2,00
gridedness	38,71	38,71	38,71	38,71	42,42	37,84	37,84
horizontal area-balance	-30,43	-30,43	-25,85	-25,85	3,00	-0,45	-0,45
margin east	30,00	30,00	30,00	30,00	30,00	22,00	22,00
margin north	6,00	6,00	6,00	6,00	6,00	6,00	6,00
margin south	8,00	8,00	8,00	8,00	8,00	8,00	8,00
margin west	19,00	19,00	19,00	19,00	19,00	19,00	19,00
non widget area	73,49	73,49	68,98	68,98	55,89	54,13	54,13
selected item count	1,00	1,00	1,00	1,00	1,00	1,00	1,00
text widget count	12,00	12,00	12,00	12,00	14,00	14,00	14,00
vertical area-balance	-48,04	-48,04	-48,27	-48,27	-60,13	-14,44	-14,44
widget count	12,00	12,00	12,00	12,00	14,00	14,00	14,00
widget density	15,79	15,79	15,79	15,79	18,42	18,42	18,42
word count	13,00	13,00	13,00	13,00	16,00	15,00	15,00

Abbildung E.6: Werte der Oberflächenmaße für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das BMW iDrive.

E.2 Werte der Konsistenz für die erhobenen Oberflächenmaße

In den folgenden Abbildungen sind die Konsistenzmaße *Z-Score* und *Cov* für die Werte des Oberflächendesigns der FIS Audi MMI und BMW iDrive für jede Dialogfolge tabellarisch dargestellt. Werte, deren Betrag der Z-Score größer als 1 ist, gelten als inkonsistent und sind in den Tabellen grau hinterlegt.

Maß / Z-Score	Schritt 1	Schritt 2	Schritt 3	Schritt 4	CoV
background color count	0,00	0,00	0,00	0,00	0,00
column count	0,00	0,00	0,00	0,00	0,00
display aspect ratio	0,00	0,00	0,00	0,00	0,00
foreground color count	0,00	0,00	0,00	0,00	0,00
gridedness	-1,73	0,58	0,58	0,58	0,58
horizontal area-balance	1,73	-0,58	-0,58	-0,58	-49,01
margin east	0,00	0,00	0,00	0,00	0,00
margin north	0,00	0,00	0,00	0,00	0,00
margin south	0,00	0,00	0,00	0,00	0,00
margin west	0,00	0,00	0,00	0,00	0,00
non widget area	-1,73	0,58	0,58	0,58	0,58
selected item count	0,00	0,00	0,00	0,00	0,00
text widget count	-1,73	0,58	0,58	0,58	0,58
vertical area-balance	1,73	-0,58	-0,58	-0,58	-52,18
widget count	-1,73	0,58	0,58	0,58	0,58
widget density	-1,73	0,58	0,58	0,58	0,58
word count	-1,73	0,58	0,58	0,58	0,58

Abbildung E.7: *Z-Scores* und *CoV* der Oberflächenmaße für die Aufgabe „Sender ‚WDR3' einstellen" für das Audi MMI.

Maß / Z-Score	Schritt 1	Schritt 2	Schritt 3	Schritt 4	Schritt 5	CoV
background color count	0,00	0,00	0,00	0,00	0,00	0,00
column count	0,00	0,00	0,00	0,00	0,00	0,00
display aspect ratio	0,00	0,00	0,00	0,00	0,00	0,00
foreground color count	0,00	0,00	0,00	0,00	0,00	0,00
gridedness	-1,72	-0,57	0,76	0,76	0,76	6,35
horizontal area-balance	1,77	-1,33	-0,21	-0,04	-0,19	-49,52
margin east	0,00	0,00	0,00	0,00	0,00	0,00
margin north	0,00	0,00	0,00	0,00	0,00	0,00
margin south	0,00	0,00	0,00	0,00	0,00	0,00
margin west	1,22	1,22	-0,82	-0,82	-0,82	122,47
non widget area	-2,00	0,53	0,49	0,49	0,49	6,34
selected item count	0,00	0,00	0,00	0,00	0,00	0,00
text widget count	-2,00	0,50	0,50	0,50	0,50	3,70
vertical area-balance	1,99	-0,38	-0,56	-0,47	-0,58	-43,50
widget count	-1,58	-0,79	0,79	0,79	0,79	6,66
widget density	-1,60	-0,76	0,79	0,79	0,79	6,62
word count	-0,50	2,00	-0,50	-0,50	-0,50	6,45

Abbildung E.8: *Z-Scores* und *CoV* der Oberflächenmaße für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen" für das Audi MMI.

Maß / Z-Score	Schritt 1	Schritt 2	Schritt 3	Schritt 4	Schritt 5	CoV
background color count	0,00	0,00	0,00	0,00	0,00	0
column count	-0,50	2,00	-0,50	-0,50	-0,50	12,50
display aspect ratio	0,00	0,00	0,00	0,00	0,00	0,00
foreground color count	-1,07	-1,07	1,60	0,27	0,27	15,59
gridedness	-0,84	1,97	-0,44	-0,34	-0,34	5,76
horizontal area-balance	1,65	-0,60	-1,37	0,16	0,16	-50,21
margin east	0,00	0,00	0,00	0,00	0,00	0,00
margin north	0,00	0,00	0,00	0,00	0,00	0,00
margin south	0,00	0,00	0,00	0,00	0,00	0,00
margin west	1,22	1,22	-0,82	-0,82	-0,82	122,47
non widget area	0,83	-1,87	0,90	0,07	0,07	7,52
selected item count	0,00	0,00	0,00	0,00	0,00	0,00
text widget count	-0,62	1,96	-0,10	-0,62	-0,62	17,31
vertical area-balance	1,73	0,22	0,04	-0,98	-1,00	-55,35
widget count	-1,48	1,60	0,37	-0,25	-0,25	8,38
widget density	-1,48	1,60	0,37	-0,25	-0,25	8,38
word count	-1,66	-0,53	0,23	0,98	0,98	16,18

Abbildung E.9: *Z-Scores* und *CoV* der Oberflächenmaße für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das Audi MMI.

Maß / Z-Score	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8	CoV
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	-2,31	0,00	1,15	1,15	0,00	0,00	0,00	0,00	43,30
display aspect ratio	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
foreground color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
gridedness	-2,58	0,57	0,10	0,10	0,57	0,11	0,57	0,57	15,91
horizontal area-balance	-1,87	-0,83	-0,50	-0,23	0,22	1,05	1,12	1,05	74,82
margin east	-2,65	0,38	0,38	0,38	0,38	0,38	0,38	0,38	31,60
margin north	2,65	-0,38	-0,38	-0,38	-0,38	-0,38	-0,38	-0,38	5,40
margin south	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
margin west	-2,65	0,38	0,38	0,38	0,38	0,38	0,38	0,38	24,74
non widget area	2,63	-0,52	-0,11	-0,45	-0,39	-0,39	-0,39	-0,39	6,82
selected item count	-2,65	0,38	0,38	0,38	0,38	0,38	0,38	0,38	37,80
text widget count	-2,63	0,42	0,42	0,42	0,42	0,14	0,42	0,42	26,71
vertical area-balance	-1,57	-0,22	0,06	-0,29	-0,03	1,50	-0,95	1,50	964,57
widget count	-2,61	0,45	0,45	0,45	0,45	-0,06	0,45	0,45	13,91
widget density	-2,63	0,43	0,43	0,43	0,43	0,05	0,43	0,43	19,28
word count	-2,63	0,38	0,38	0,38	0,38	0,16	0,38	0,59	28,66

Abbildung E.10: *Z-Scores* und *CoV* der Oberflächenmaße für die Aufgabe „Sender ‚WDR3‘ einstellen“ für das BMW iDrive.

Maß / Z-Score	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8	S. 9	S. 10
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	-2,92	0,34	1,97	0,34	0,34	0,34	0,34	0,34	-1,29	-1,29
display aspect ratio	4,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24
foreground color count	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	-2,31	-2,31
gridedness	-3,04	0,72	0,15	0,72	0,72	0,72	0,72	0,72	-2,20	-0,77
horizontal area-balance	0,33	0,52	0,58	0,52	0,52	0,52	0,21	0,36	-2,26	-2,26
margin east	-0,75	-0,42	-0,42	-0,42	-0,42	-0,42	-0,42	-0,42	2,30	2,30
margin north	4,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24
margin south	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	2,92
margin west	-4,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24
non widget area	1,17	1,17	-1,21	-0,90	-1,21	-1,21	-1,21	-1,21	-1,21	1,40
selected item count	-4,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24
text widget count	-1,77	1,12	1,12	1,12	1,12	1,12	1,12	1,12	-1,51	-1,51
vertical area-balance	0,61	0,99	1,06	0,99	0,99	0,99	1,69	1,49	-0,42	0,20
widget count	-0,57	1,14	1,14	1,14	1,14	1,14	1,14	1,14	-1,72	-1,72
widget density	-1,13	1,15	1,15	1,15	1,15	1,15	1,15	1,15	-1,64	-1,64
word count	-2,40	1,00	1,00	1,00	1,00	1,00	1,24	1,00	-1,43	-1,19
Maß / Z-Score	S. 11	S. 12	S. 13	S. 14	S. 15	S. 16	S. 17	S. 18	S. 19	CoV
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	-1,29	0,34	0,34	0,34	0,34	0,34	0,34	0,34	0,34	34,30
display aspect ratio	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	6,94
foreground color count	-2,31	0,43	0,43	0,43	0,43	0,43	0,43	0,43	0,43	19,79
gridedness	-0,77	0,29	0,29	0,29	0,29	0,29	0,29	0,29	0,29	13,43
horizontal area-balance	-2,26	0,18	0,18	0,18	0,18	0,18	0,77	0,77	0,77	-232,76
margin east	2,30	-0,40	-0,40	-0,40	-0,40	-0,40	-0,40	-0,40	-0,40	124,90
margin north	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	-0,24	3,69
margin south	2,92	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	-0,34	130,99
margin west	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	15,85
non widget area	1,48	1,48	0,24	0,24	0,24	0,24	0,24	0,24	0,03	8,42
selected item count	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	0,24	23,57
text widget count	-1,51	-0,19	-0,19	-0,19	-0,19	-0,19	-0,19	-0,19	-0,19	35,48
vertical area-balance	-1,26	-0,91	-0,91	-0,91	-0,91	-0,91	-0,93	-0,93	-0,93	-98,04
widget count	-1,72	-0,29	-0,29	-0,29	-0,29	-0,29	-0,29	-0,29	-0,29	31,77
widget density	-1,64	-0,25	-0,25	-0,25	-0,25	-0,25	-0,25	-0,25	-0,25	32,99
word count	-1,43	-0,22	-0,22	0,03	0,03	0,03	0,03	-0,22	-0,22	29,62

Abbildung E.11: *Z-Scores* und *CoV* der Oberflächenmaße für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das BMW iDrive.

Maß / Z-Score	S. 1	S. 2	S. 3	S. 4	S. 5	S. 6	S. 7	S. 8
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	-2,24	-2,24	-1,12	-1,12	-1,12	0,00	0,00	0,00
display aspect ratio	3,74	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27
foreground color count	0,50	-2,00	-2,00	-2,00	0,50	0,50	0,50	0,50
gridedness	-3,30	-3,30	-0,21	-0,21	-0,21	0,28	0,28	0,28
horizontal area-balance	0,98	-1,83	-1,83	-1,83	0,06	0,06	0,06	-0,01
margin east	-0,70	2,00	2,00	2,00	-0,47	-0,47	-0,47	-0,47
margin north	3,74	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27
margin south	-0,50	2,00	2,00	2,00	-0,50	-0,50	-0,50	-0,50
margin west	-3,74	0,27	0,27	0,27	0,27	0,27	0,27	0,27
non widget area	1,79	1,79	1,07	1,07	1,07	-0,07	-0,07	-0,07
selected item count	-3,74	-3,74	0,27	0,27	0,27	0,27	0,27	0,27
text widget count	-2,06	-2,06	-1,44	-1,44	-1,44	0,41	0,41	0,41
vertical area-balance	1,89	0,34	0,34	0,63	-0,74	-0,74	-0,74	-0,87
widget count	-0,73	-0,73	-1,81	-1,81	-1,81	0,36	0,36	0,36
widget density	-1,38	-1,38	-1,67	-1,67	-1,67	0,39	0,39	0,39
word count	-2,00	-2,00	-1,46	-1,46	-1,46	0,41	0,41	0,41
Maß / Z-Score	S. 9	S. 10	S. 11	S. 12	S. 13	S. 14	S. 15	CoV
background color count	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
column count	0,00	1,12	1,12	0,00	0,00	1,12	1,12	44,72
display aspect ratio	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27	7,71
foreground color count	0,50	0,50	0,50	0,50	0,50	0,50	0,50	22,22
gridedness	0,28	0,28	0,28	0,28	0,28	1,81	-0,07	6,39
horizontal area-balance	0,19	0,19	0,33	0,33	1,17	1,17	1,17	-92,49
margin east	-0,47	-0,47	-0,47	-0,47	-0,47	-0,54	-0,54	132,32
margin north	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27	-0,27	4,11
margin south	-0,50	-0,50	-0,50	-0,50	-0,50	-0,50	-0,50	87,32
margin west	0,27	0,27	0,27	0,27	0,27	0,27	0,27	17,88
non widget area	-0,07	0,28	0,28	-0,18	-0,18	-1,52	-1,70	13,80
selected item count	0,27	0,27	0,27	0,27	0,27	0,27	0,27	26,73
text widget count	0,41	0,41	0,41	0,41	0,41	1,03	1,03	30,36
vertical area-balance	-0,52	-0,52	-0,54	-0,54	-1,31	-1,31	-1,31	-38,39
widget count	0,36	0,36	0,36	0,36	0,36	1,09	1,09	25,06
widget density	0,39	0,39	0,39	0,39	0,39	1,08	1,08	26,80
word count	0,41	0,41	0,41	0,41	0,41	1,21	0,94	32,61

Abbildung E.12: *Z-Scores* und *CoV* der Oberflächenmaße für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das BMW iDrive.

Anhang F

Ergebnisse der empirischen Bewertung: semantische Differentiale

Die subjektive Meinung der Probanden wurde nach jeder Aufgabenbearbeitung durch einen Fragebogen mit den aus der Bedeutungsanalyse bekannten *semantischen Differenzialen* ermittelt. Bei den semantischen Differenzialen hat die Versuchsperson die Aufgabe, das betrachtete System gegenüber einer Reihe beschreibender, bipolarer Adjektivskalen zu bewerten. Dazu werden sieben Stufen als mögliche Einteilung der Skalen gewählt, so dass es möglich ist, Aussagen wie „ziemlich“, „etwas“ oder „extrem“ zu treffen. Damit jede Adjektivskala für sich bewertet wird, werden den Versuchspersonen die Skalen in ihrer Semantik ungeordnet angeboten, d.h. positive Seite mal links, mal rechts, mal in der Mitte. Dies führt zu einer deutlichen Steigerung der Aufmerksamkeit beim Ausfüllen des Fragebogens. Weitere Informationen zu semantischen Differenzialen finden sich in (OSGOOD et al. 1957; HOFSTÄTTER und WENDT 1974).

Bei der Auswertung erfolgt Umrechnung der Skalenstufen in Zahlenwerte, wobei schlechte Eigenschaften negativen Zahlen und positiven Zahlen zugeordnet werden. Anschließend erfolgt eine Sortierung der Werte (positive rechts, negative links). Die gemittelten Ergebnisse der semantischen Differenziale für jede Aufgabe sind im Folgenden abgebildet.

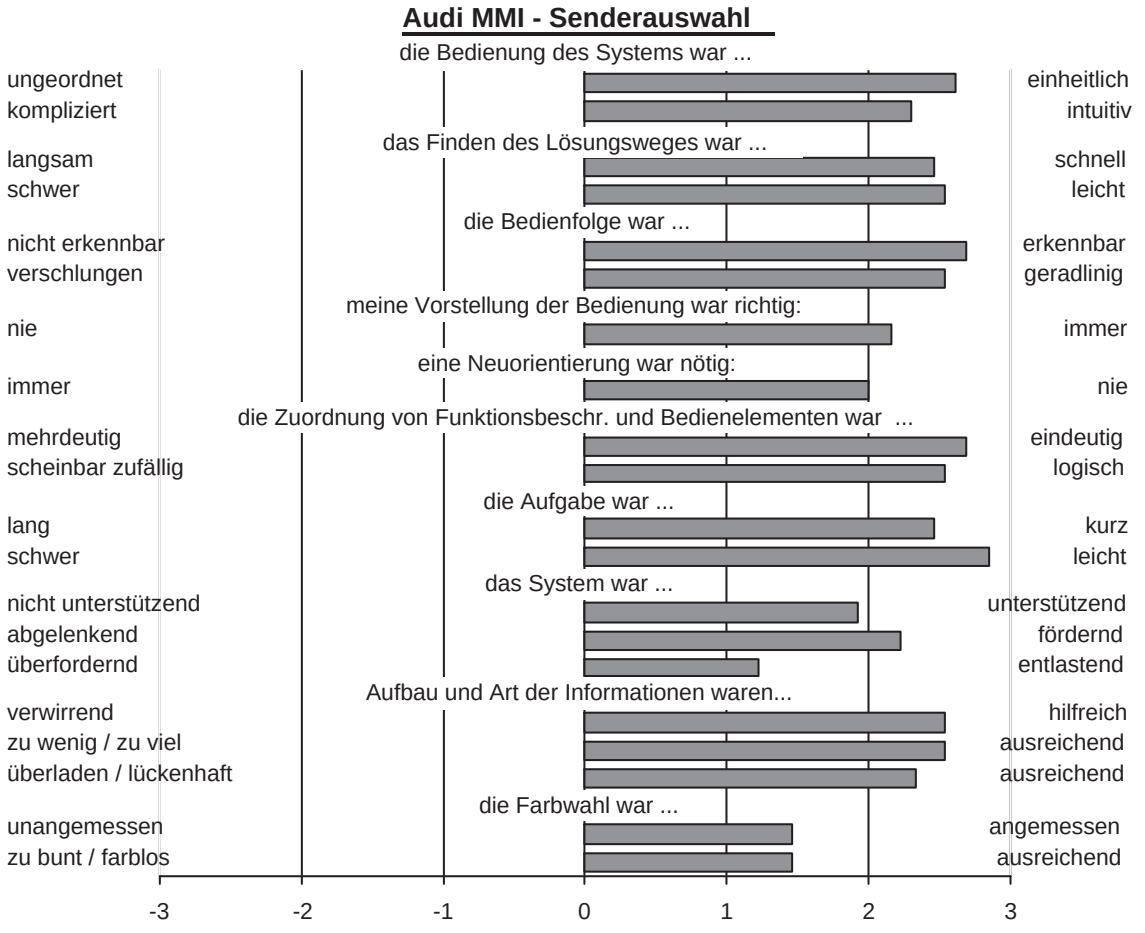

Abbildung F.1: Auswertung der semantischen Differenziale für die Aufgabe „Sender ‚WDR3' einstellen" für das Audi MMI.

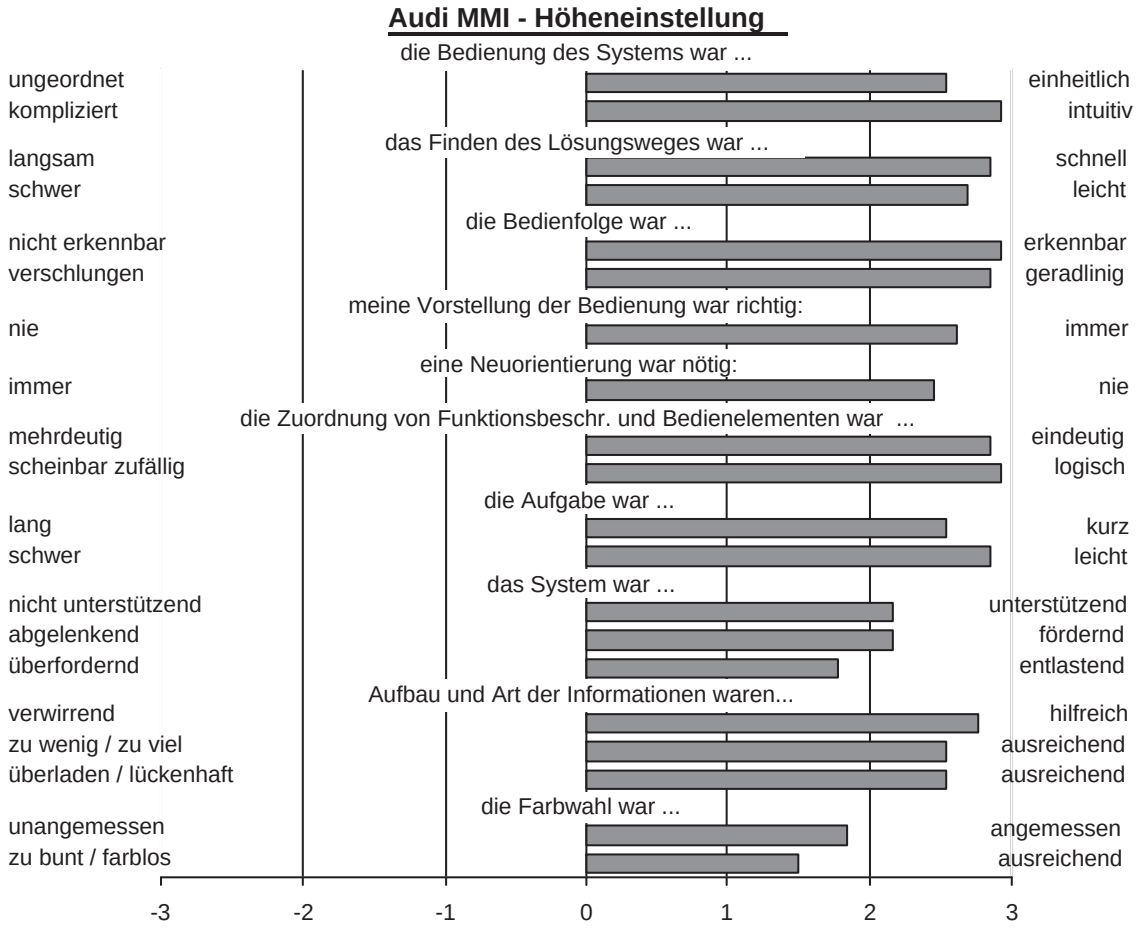

Abbildung F.2: Auswertung der semantischen Differenziale für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das Audi MMI.

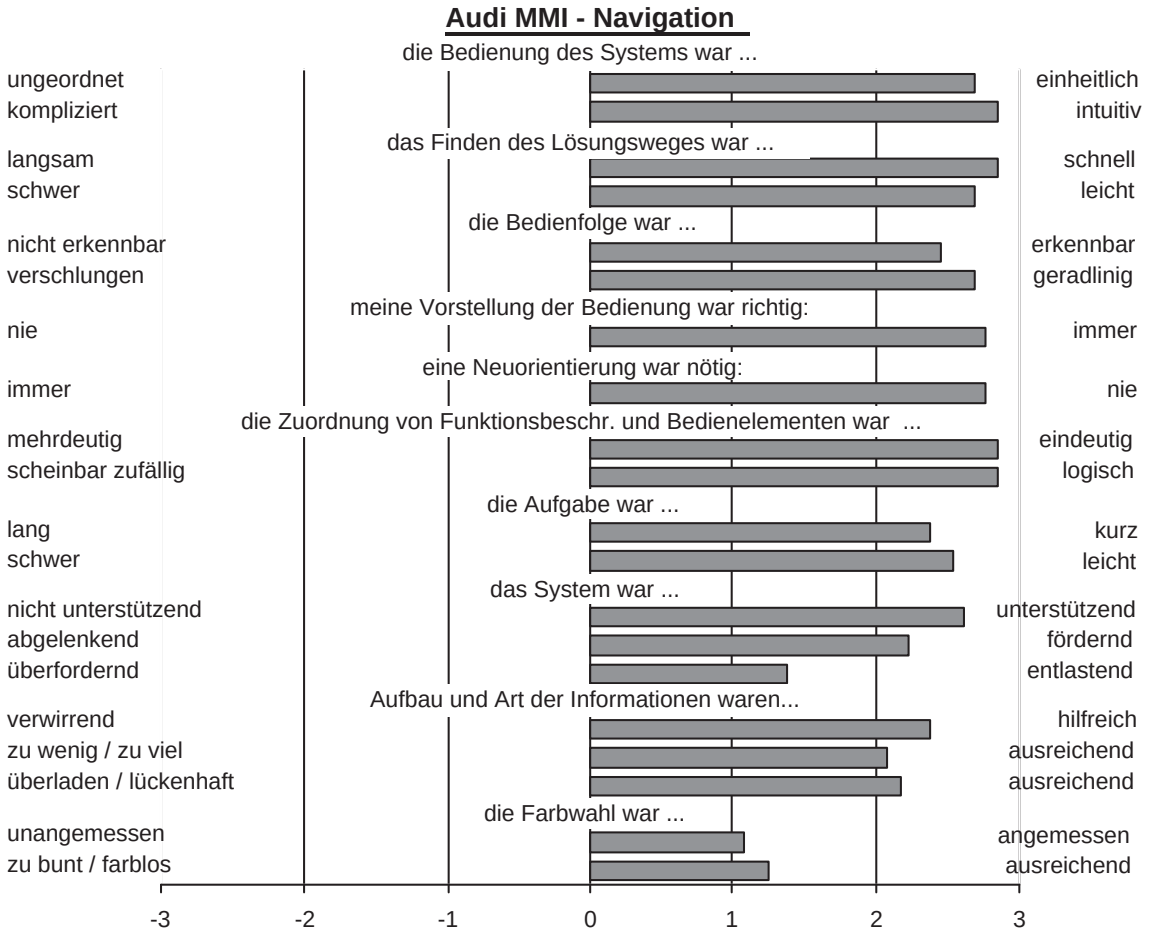

Abbildung F.3: Auswertung der semantischen Differenziale für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das Audi MMI.

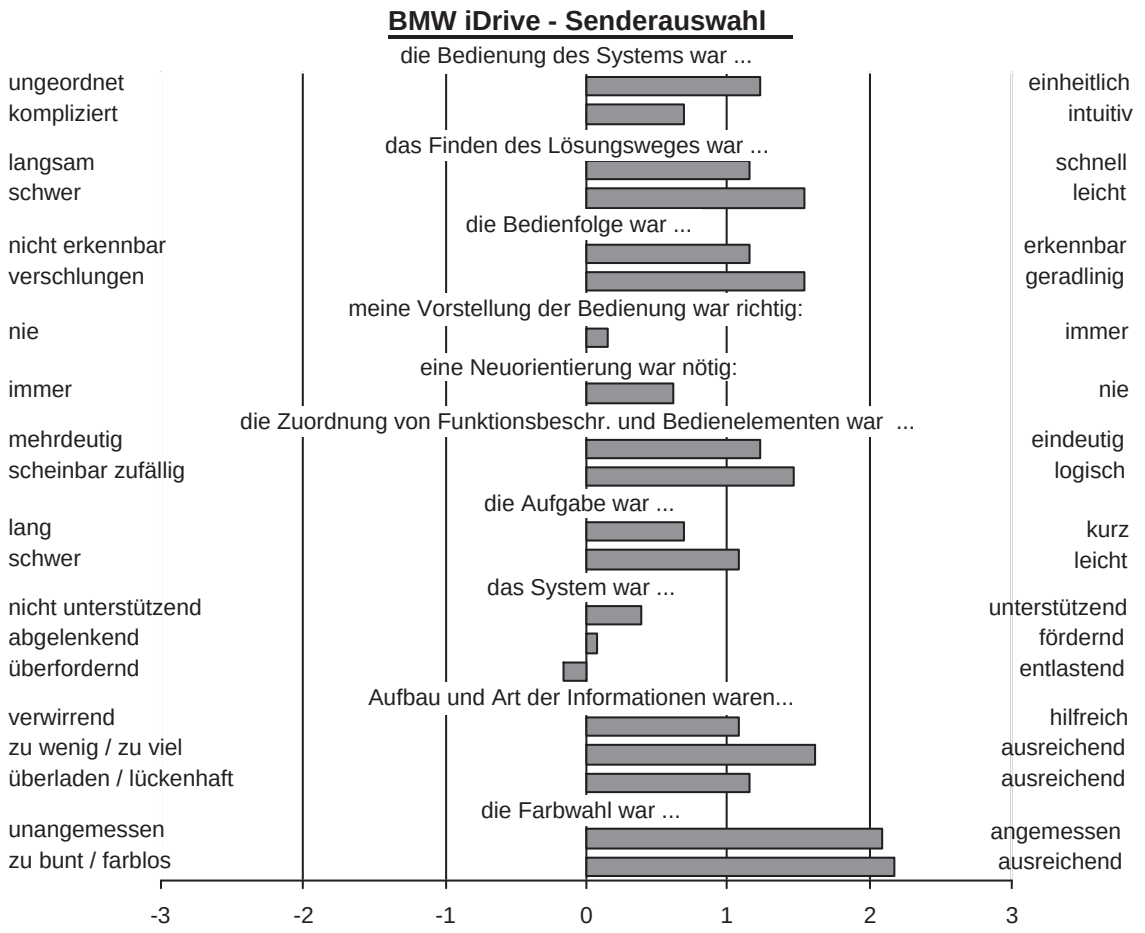

Abbildung F.4: Auswertung der semantischen Differenziale für die Aufgabe „Sender ‚WDR3' einstellen" für das BMW iDrive.

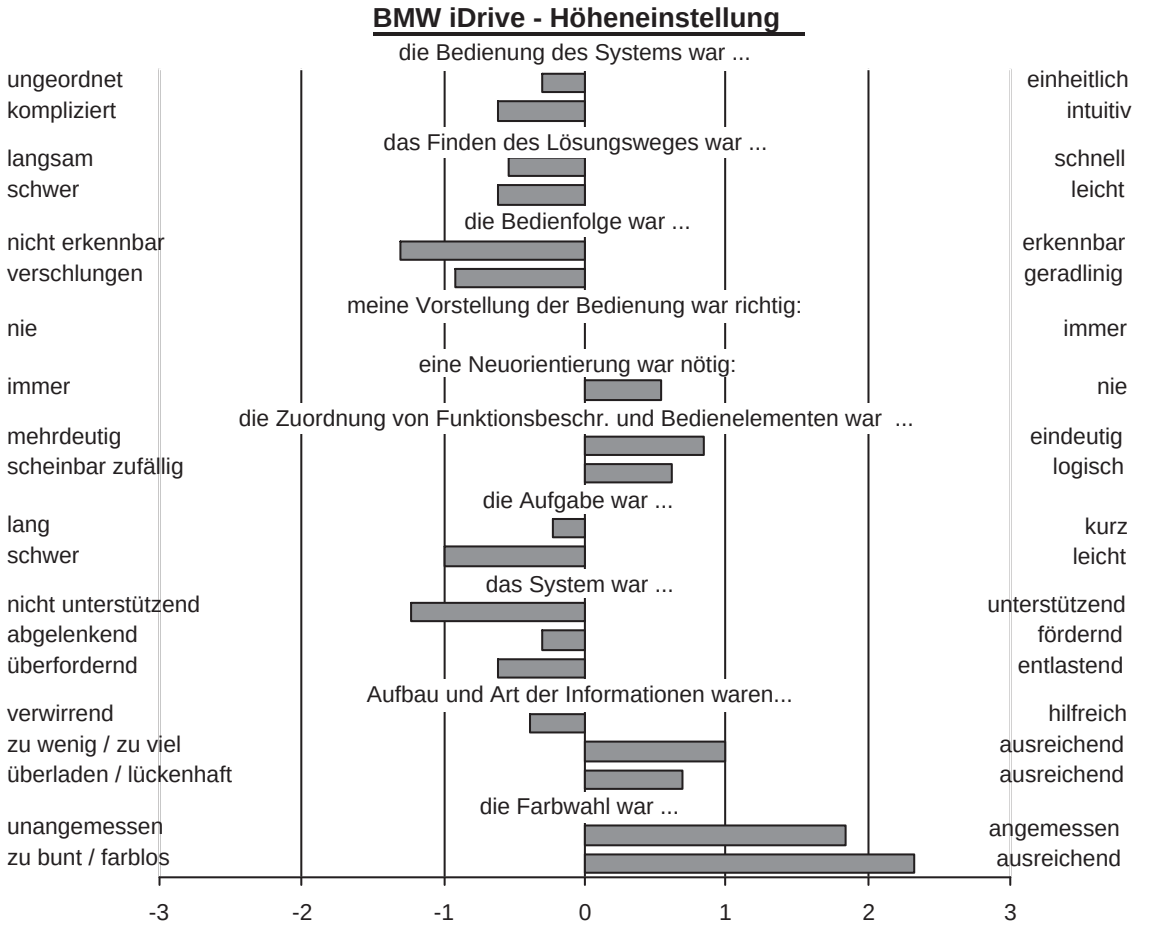

Abbildung F.5: Auswertung der semantischen Differenziale für die Aufgabe „Menü zur Einstellung der Klanghöhen aufrufen“ für das BMW iDrive.

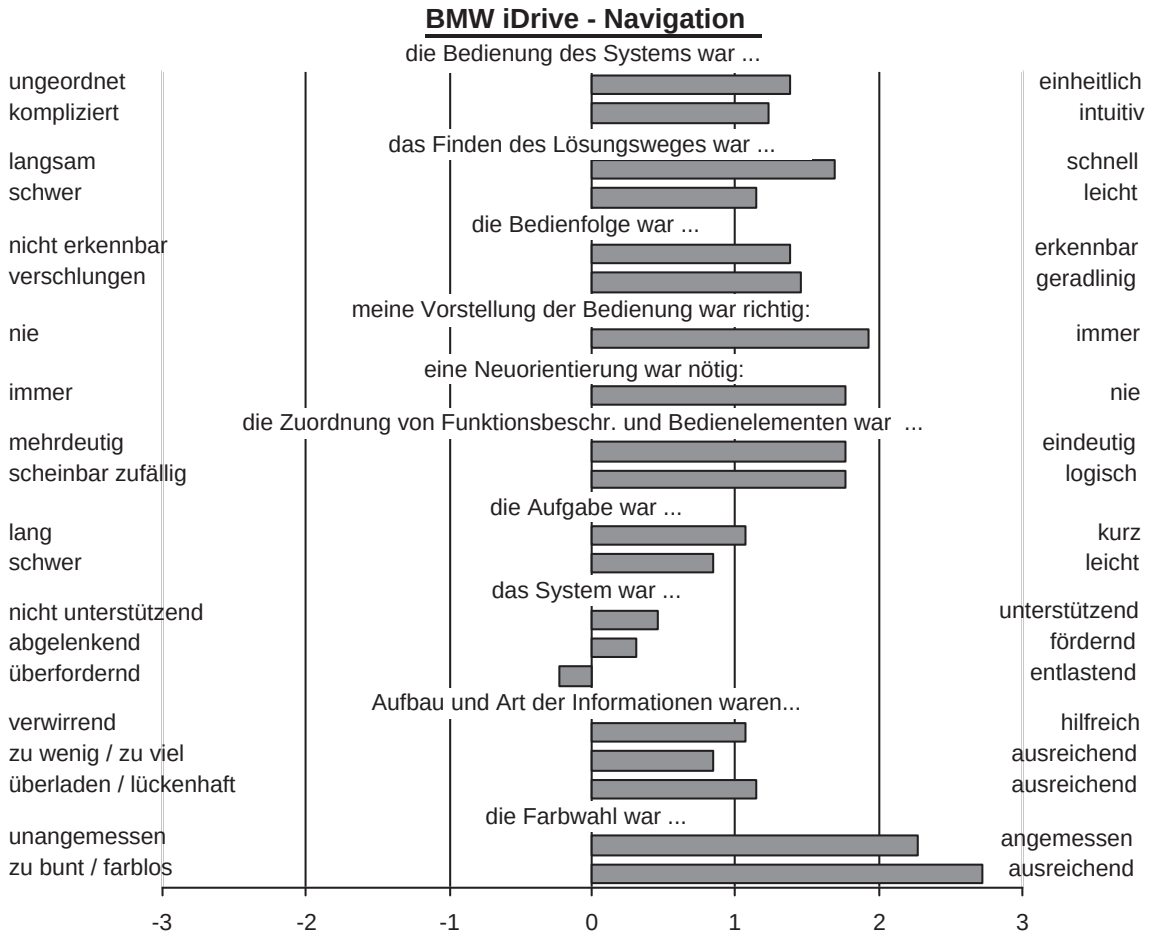

Abbildung F.6: Auswertung der semantischen Differenziale für die Aufgabe „Zweites Navigationsziel aus Zielliste auswählen“ für das BMW iDrive.

Glossar

Das Glossar ist lexikografisch sortiert. Verweise zu weiteren erklärten Schlagworten sind mit „[→]“ gekennzeichnet.

ActiveX-DLL

Softwarekomponenten (Dynamik Link Library - DLL), die miteinander in Netzwerkumgebungen interagieren, und zwar unabhängig von der Sprache, in der die Komponenten erstellt wurden. Die Schnittstellen sind von Microsoft definiert und fest vorgegeben. ActiveX-DLLs sind ausschließlich in Microsoft Windows anwendbar.

Siehe http://support.microsoft.com/default.aspx?scid=kb;de;154544

Aussagenlogik

Stellt den ältesten Zweig der formalen Logik dar. Die Aussagen sind grundsätzlich *wahr* oder *falsch* und können durch die Junktoren *und* und *oder* sowie die Verneinung *nicht* zu neuen Aussagen verknüpft werden (BARWISE und ETCHEMENDY 1992; GOTTLOB et al. 1990). Die Aussagenlogik entspricht der Schaltalgebra bzw., den dort angegebenen booleschen Ausdrücken (CLAUS und SCHWILL 2003).

Benutzungsfreundlichkeit

Ausmaß, in dem ein Produkt durch bestimmte Operateure in einem bestimmten Nutzungskontext genutzt werden kann, um bestimmte Ziele effektiv, effizient und mit Zufriedenheit zu erreichen (ISO 9241-11 1998).

Backus-Naur-Form (BNF)

Beschreibungsform kontextfreier Grammatiken und damit der Syntax vieler Programmiersprachen. Sie verwendet Ersetzungsregeln, die eine linke und eine rechte Seite besitzen und in denen Terminal und Nichtterminalzeichen vorkommen. Nichtterminalzeichen werden durch <...> gekennzeichnet und können mit Hilfe weiterer Ersetzungsregeln ausgetauscht werden.

Benutzungsoberfläche, Bedienoberfläche

[→] (Graphical) User Interface

Branch and Bound-Algorithmus

Methode zur Bestimmung der Lösung eines Optimierungsproblems, dass schrittweise in Teilprobleme zerlegt wird (*branch*). Die einzelnen Teilprobleme bilden einen

Baum mit Knoten, denen jeweils ein Wert als Schranke (*bound*) für den Wert der Zielfunktion zugeordnet wird. Es werden nun diejenigen Zweige bearbeitet, die den höchsten Wert der Zielfunktion erwarten lassen. Zweige, deren Werte bereits geringer sind, werden entfernt. Dadurch lässt sich der Rechenaufwand deutlich reduzieren (CLAUS und SCHWILL 2003).

Dialogfolge

Bezeichnet in dieser Arbeit die Folge der GUI- und Systemzustände inkl. der benötigten Bedienaktionen in Abhängigkeit einer Aufgabenbearbeitung.

Endlicher Automat

Mathematisches Modell für Automaten, die Informationen Zeichen für Zeichen einlesen, das Zeichen sofort weiterverarbeiten und eine Ausgabe erzeugen. Ein Endlicher Automat besitzt eine endliche Menge von Zuständen und keinen zusätzlichen Speicher (CLAUS und SCHWILL 2003).

Entity-Relationship Diagram (ERD)

Dient zur Darstellung von *Objekten* (dargestellt als Rechteck), deren *Eigenschaften* (dargestellt als Ovale) und den *Beziehungen* (dargestellt als Raute) zu anderen Objekten. ERDs haben sich vor allem als Standard zur Visualisierung von Datenbankstrukturen durchgesetzt. Weitergehende Informationen finden sich bei (THALHEIM 2000).

Ergonomie

Eine angewandte wissenschaftliche Disziplin der Analyse und Optimierung menschlicher Tätigkeiten und Leistungen unter Einbeziehung subjektiver und objektiver Faktoren und Prozesse. Das Ziel liegt in der Anpassung der Arbeitsbedingungen und Werkzeuge an den individuellen Menschen (WANDMACHER 1993).

Expertensystem

Programmsystem, das „Wissen“ über eine spezielle Domäne speichert und ansammelt, aus dem Wissen Schlussfolgerungen zieht und zu konkreten Problemen der Domäne Lösungen anbietet (CLAUS und SCHWILL 2003). Das Gebiet der Expertensysteme sowie die Möglichkeiten der Anwendung sind sehr umfangreich. Folgende Literatur bietet weitergehende Informationen: (HOLLNAGEL 1989; BULLINGER und WASSERLOOS 1989; KURBEL 1992; STEINMÜLLER 2005)

Fitts Gesetz

Formel zur Berechnung der benötigten Zeit t in Millisekunden, um einen Cursor (Mauspfeil) zu einem Ziel, etwa einem Button oder einem Textlink, zu bewegen. Dabei ist die Entfernung d der Abstand in gerader Linie vom Startpunkt der Bewegung zum nächsten Punkt des Ziels und s die Größe des Ziels entlang dieser geraden Linie der Bewegung (FITTS 1954). Dann gilt:

$$t = a + b \cdot \log(2 \cdot d/s) \tag{F.1}$$

a und *b* sind experimentell bestimmte Konstanten (FITTS 1954). Raskin empfiehlt als grobe Näherung $a = 50$ und $b = 150$ (RASKIN 2000).

Funktionale Sprache

Programmiersprache, die Eingabedaten in Ausgabedaten abbildet. Die Berechnung einer Funktion steht bei der Programmierung im Vordergrund und nicht die Reihenfolge der Ausführung wie bei [→] prozeduralen Sprachen.

Gerätemodell

Beschreibt die interne Funktionalität des Systems in Form einer formalen Spezifikation, z.B. durch Schaltpläne, Algorithmen oder [→]Zustandsübergangsdiagramme. Zusätzlich werden die Eigenschaften der Ein- und Ausgabeelemente des [→]User Interfaces und ihre Beziehungen untereinander festgelegt.

GOMS-Modell

Eine der gängigsten formalen Bewertungsmethoden ist das GOMS-Modell, wobei GOMS für die Komponenten des Modells steht: Ziele (Goals), Operatoren (Operators), Methoden (Methods) und Auswahlregeln (Selection Rules). Goals sind die Ziele, die der Anwender mittels der Aufgabenbearbeitung erreichen will. Die Operatoren sind die sensorischen, kognitiven und motorischen Aktionen, durch die der Anwender mit dem interaktiven Endgerät interagiert. Die Methoden repräsentieren Teilziele und setzen sich aus einer Abfolge von Operatoren und weiteren Methoden zusammen. Die Auswahlregeln werden verwendet, wenn durch das System mehrere Methoden alternativ zum Erreichen des Ziels möglich sind. Durch eine Analyse von GOMS-Modellen können sowohl qualitative Aussagen hinsichtlich der Umsetzung der Funktionalität als auch quantitative Vorhersagen zur Bedienbarkeit, z.B. Ausführungs- und Lernzeiten, gemacht werden (CARD et al. 1983).

Hardkey

Taste zur Aktivierung fester Funktionen – im Gegensatz zu einem [→] Softkey. Die Funktionsbeschreibungen bzw. -namen sind meist auf die Taste gedruckt und ändern sich nicht.

Homonym

Wort, das für verschiedene Begriffe stehen kann. Umgangssprachlich werden solche mehrdeutigen Bezeichnungen auch *Teekesselchen* genannt. Beispiele sind „Schloss“ (Verriegelung oder Gebäude), „Bank“ (Geldinstitut oder Sitzgelegenheit) bzw. „modern“ (neu oder verwesen). Ihre jeweilige Bedeutung erhalten die Worte über den Kontext, in dem sie benutzt werden.

Das Gegenteil eines Homonym ist ein [→] Synonym.

Inferenz

Schlussfolgerungsprozess durch zielgerichtete Auswertung der in einer Wissensbasis enthaltenen Regeln und Fakten. Grundsätzlich wird bei der Inferenz zwischen

Vorwärtsverkettung (datengetrieben) und Rückwärtsverkettung (zielgetrieben) unterschieden. Bei der Vorwärtsverkettung wird aufgrund bestehender Fakten auf neue Daten geschlossen. Mögliche Zielzustände sind nicht bekannt, die Suche ist somit nicht zielgerichtet und daher zum Teil nutzlos. Bei der Rückwärtsverkettung ist das gewünschte Ziel vorgegeben und es sollen die Fakten ermittelt werden, die die Regeln zum Erreichen des Ziels erfüllen. Falls nicht alle Fakten dazu vorhanden sind, werden Regeln gesucht, die wiederum die benötigten Fakten erzeugen (RUSSEL und NORVIG 2004).

Interaktive Systeme
Geräte, Produkte und Software, die mit dem Benutzer interagieren, also auf Benutzeraktionen reagieren und dem Benutzer Informationen zur Verfügung stellen (BENYON et al. 2005). Interaktive Systeme werden auch als *reaktive Systeme* bezeichnet (MARRENBACH 2001).

JESS Java Expert System Shell – Programmpaket sowie Programmiersprache zur Implementierung von Expertensystemen in Java (FRIEDMAN-HILL 2003). Die offizielle Webseite ist unter folgendem Link zu finden: http://herzberg.ca.sandia.gov/jess/

Logik s. [→]Aussagenlogik bzw. [→]Prädikatenlogik

Ontology Web Language (OWL)
Format zur Beschreibung von Ontologien, das Anfang 2004 vom [→] W3C vorgestellt wurde, und aktuell die zukunftsträchtigste Ontologiesprache darstellt (FENSEL 2004). OWL ist eine Erweiterung des *Resource Description Framework Schema (RDFS)* und erweitert die gängige *DARPA Agent Markup Language (DAML)* in Verbindung mit dem *Ontology Inference Layer (OIL)*.
Weiterführende Informationen gibt Lacy in (LACY 2005) bzw. die offizielle Webseite: http://www.w3.org/2004/OWL/

Petri-Netz
Modell zur Beschreibung und Analyse von Abläufen mit nebenläufigen Prozessen und nichtdeterministischen Vorgängen, welches 1962 von C.A. Petri entwickelt wurden. Ein Petri-Netz ist ein gewichteter Graf, bestehend aus *Knoten* (ursprünglich auch als *Stellen* bezeichnet), als Zwischenablage für Daten (in Form von *Marken*), und *Transitionen*, als Beschreibung für die Verarbeitung von Daten. Kanten verbinden Transitionen und Knoten, dürfen also nicht Knoten mit Knoten oder Transitionen mit Transitionen verbinden. Eine Transition schaltet, wenn jeder eingehende Knoten mindestens eine Marke enthält. Schaltet eine Transition, dann wird aus jedem eingehenden Knoten eine Marke entfernt und zu jedem ausgehenden Knoten eine Marke hinzugefügt (CLAUS und SCHWILL 2003).

Prädikatenlogik
(auch **Logik erster Ordnung**) Erweitert die [→]Aussagenlogik um Variablen und die Quantoren ∀ („für alle“) und ∃ („es gibt mindestens ein“). Sie dient zur For-

malisierung und zum Beweis von Eigenschaften von Programmen (CLAUS und SCHWILL 2003). Die Prädikatenlogik ist durch den Einsatz der Quantoren nicht mehr in jedem Fall entscheidbar, was den Einsatz in Expertensystemen unmöglich macht (STEINMÜLLER 2005).

Präfix-Notation

(auch **Polnische-Notation** oder **Lukasiewicz-Notation**) Notation zur Darstellung mathematischer Ausdrücke, bei der grundsätzlich der Operator vor den Operanden steht (CLAUS und SCHWILL 2003). Diese Notation ist eindeutig und benötigt keine Klammern. Das Beispiel „$(4+9):(17+2)$“ in der gängigen Infixnotation entspricht dem Ausdruck „ : + 4 9 + 17 2 “ in Präfixnotation.

Die Präfixnotation ist allgemein für zwei Operanden definiert, lässt sich jedoch auch für jegliche Funktionsaufrufe anwenden, deren Operanden-/Parameteranzahl eindeutig festgelegt ist.

Prozedurale Sprache

(auch **imperative Sprache**) Programmiersprache zur Implementierung von Befehlsfolgen. Auf Grundlage der von-Neumann-Architektur von Systemen steht die Ausführungsreihenfolge und nicht logische Funktionen wie bei [→] funktionalen Sprachen im Fokus.

RETE-Algorithmus

Pattern-Matching-Algorithmus zur effizienten Überprüfung von Fakten und Regeln in Expertensystemen. Der Algorithmus wurde 1982 von Forgy in (FORGY 1982) zum ersten Mal vorgestellt und wird in vielen populären [→]Expertensystemen wie z.B. *Clips* oder [→]*JESS* eingesetzt. Der Name leitet sich aus dem Lateinischen „rete“ für „Netz“ ab, da jede Bedingung einer Prämisse als Knoten eines Netzes dargestellt wird. Dadurch werden Abhängigkeiten von Regeln abgebildet und eine Überprüfung der Faktenbasis kann sehr effizient erfolgen. Weitere Informationen zum RETE-Algorithmus bieten u.a. (ALBERT 1989; FRIEDMAN-HILL 2003).

Seeheim-Modell

Softwaretechnologisch orientiertes Modell von Benutzungsschnittstellen bestehend aus drei Komponenten: Präsentationskomponente, Dialogkontrolle und Anwendungsschnittstelle – s. auch Abschnitt 1.1 bzw. (PFAFF 1983).

SDL - Specification and Description Language

Formale Beschreibungssprache für die Spezifikation interaktiver Systeme.

Softkey Im Gegensatz zu einem [→] Hardkey ist ein Softkey eine Taste zur Aktivierung mehrerer unterschiedlicher Funktionen. Die jeweils zu aktivierende Funktion wird in einem Display angezeigt. Um die Funktionsbeschreibung dem Softkey zuordnen zu können, muss sie in direkter Nähe dargestellt werden. Dazu ist es hilfreich, Softkeys direkt neben, ober- oder unterhalb einem Display anzuordnen. Es existieren ebenfalls Softkeys mit einem eigenen, in die Taste eingebauten Display.

(Bedien-) Stereotyp

Erwartungskonforme Bedienung von Stellteilen. Die Anwendung solcher Stereotypen ist gelernt und gesellschaftsabhängig.
Synonyme Begriffe sind *Metapher*, *mentales Modell*, *Kompatibilität* oder *natural Mapping*, wobei die Begriffe nicht scharf definiert bzw. voneinander abgegrenzt sind (HOYOS 1974; LANC 1975). In der Psychologie wird teilweise zwischen (Bedien-)Metapher und (Bedien-)Stereotypen unterschieden. Eine Metapher stellt dabei eine allgemeine erwartungskonforme Bedienung eines Stellteils, z.B. „Drehen eines Drehknopfes im Uhrzeigersinn bedeutet *mehr* im Sinne von *lauter* oder *wärmer*“, dar (ZIMBARDO und GERRIG 1999). Eine Stereotype hingegen beschränkt sich auf ein konkretes Stellteil, z.B. „Drücken eines Lichtschalters (im Gegensatz zu Schaltern im Allgemeinen) in der oberen Hälfte schaltet das Licht aus“.

Weiterhin werden häufig unterschiedliche Abstraktionsgrade angegeben (HUTCHINS 1989). Da in dieser Arbeit solche Unterscheidungen nicht sinnvoll sind, wird ausschließlich der Begriff *Stereotyp* zur Angabe des Zusammenhangs zwischen einer Bedienaktion und einer erwarteten Reaktion des Systems verwendet.

Styleguide

Sammlung von Richtlinien zur Gestaltung einer grafischen Benutzungsoberfläche mit Fokus auf dem Oberflächendesign; meist zur Vereinheitlichung bzw. Sicherstellung der *Corporate Identity*.

Synonym

Wort, das die gleiche Bedeutung hat wie ein anderes Wort. Die ursprüngliche Definition stammt von Aristoteles: Zwei Dinge sind synonym, wenn sie dieselbe Bezeichnung und dieselbe Definition aufweisen. Im Kontext von synonymen Begriffen wird jedoch ausschließlich die Semantik betrachtet. Beispiele sind „Orange - Apfelsine“ oder „Streichholz - Zündholz“. Im alltäglichen Sprachgebrauch werden fremdsprachliche Begriffe und ihre deutschen Übersetzungen ebenfalls synonym verwendet, z.B. „abspielen - play“.

Das Gegenteil eines Synonyms ist ein [→] Homonym.

Token (engl. für *Zeichen, Marke*) Kleinste sinngebende Informationseinheit. Dabei kann ein *Ausgabetoken* z.B. den Zustand bzw. das angewählte Element einer Combobox[1] angeben. In Programmiersprachen bzw. im Compilerbau bezeichnet ein Token die lexikalische Grundeinheit, welches einem Terminalsymbol (vgl. [→] BNF) einer Grammatik entspricht.

Unified Modeling Language (UML)

Ein Standard zur Beschreibung von Software und Datenstrukturen mit Hilfe von unterschiedlichen Diagrammtypen (FOWLER und SCOTT 2000).

[1] Eine Combobox ist ein GUI-Konstrukt, dass mehrere Element zur Auswahl bietet.

(Graphical) User Interface - (G)UI

Der Teil bzw. die Schnittstelle eines Systems, mit dem der Benutzer in Kontakt kommt. Die Schnittstelle benötigt Mechanismen, damit der Benutzer Instruktionen oder Informationen in das System eingeben kann sowie um Feedback oder Informationen auszugeben (BENYON et al. 2005). Synonyme Begriffe sind *Benutzungsoberfläche* und *Bedienoberfläche*.

W3C Das World Wide Web Consortium (W3C) ist ein Gremium zur Standardisierung das World Wide Web betreffender Techniken. Gründer und Vorsitzender des W3C ist Tim Berners-Lee (der auch als der Erfinder des World Wide Web gilt). Das W3C wurde 1994 gegründet, die offizielle Internetadresse lautet: http://www.w3c.org

Zustandsübergangsdiagramme/Statecharts

Grafische Notation zur Definition und Modellierung komplexer, ereignisgesteuerter Systeme. Zustandsübergangsdiagramme erweitern die [→]Endlichen Automaten um Hierarchie und Orthogonalität.

Index

Lebenslauf des Verfassers

Name:	Nicolas Alexander Hamacher
Geburtstag:	15. Mai 1971
Geburtsort:	Rheinbach
Kontakt:	dissertation@hamacher-online.com

Schulbesuch:

1978 - 1982	Grundschule am Hagelkreuz, Nitterscheid
1982 - 1991	St. Michael-Gymnasium, Bad Münstereifel
15. Juni 1991	Erlangung der allgemeinen Hochschulreife

Zivildienst:

08/1991 - 09/1992	Ev. Kirchengemeinde, Bad Münstereifel

Studium:

10/1992 - 07/2000	Studium der Informatik an der RWTH Aachen, Abschluss: Diplom
10. Juli 2000	Datum des Diplomzeugnisses
08/2000 - 03/2006	Promotion am Lehrstuhl für Technische Informatik, RWTH Aachen
21. Juli 2006	Datum der wissenschaftlichen Aussprache

Praktische Erfahrungen:

05/1991 - 11/1996	Freier Mitarbeiter bei der Fosbel GmbH, Euskirchen
07/1993 - 08/1997	Freier Mitarbeiter bei der BB-medica GmbH, Aachen
04/1996 - 08/2000	Freier Mitarbeiter beim Verlag für Büchersammler – Radtke, Aachen
06/1996 - 03/1997	Studentische Hilfskraft am Lehrstuhl für Informatik V, RWTH Aachen
07/1997 - 05/2000	Studentische Hilfskraft am Lehrstuhl für Technische Informatik, RWTH Aachen

Berufstätigkeit:

08/2000 - 10/2005	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Technische Informatik, RWTH Aachen
seit 04/2006	IVU Traffic Technologies AG, Aachen