Modellbasierte Analyse von Nutzerschnittstellen

Sebastian Winter

Technische Universität München

Institut für Informatik
der Technischen Universität München

Modellbasierte Analyse von Nutzerschnittstellen

Sebastian Winter

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzende:	Univ.-Prof. Gudrun J. Klinker, Ph.D.
Prüfer der Dissertation:	
1.	Univ.-Prof. Dr. Dr. h.c. Manfred Broy
2.	Univ.-Prof. Dr. Elisabeth André, Universität Augsburg

Die Dissertation wurde am 30.06.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 07.12.2009 angenommen.

Kurzfassung

Elektronische Geräte werden beständig leistungsfähiger und sind dadurch in der Lage, eine Vielzahl unterschiedlicher Funktionen anzubieten. Gleichzeitig wird deren Größe, ebenfalls beständig, von den Herstellern reduziert. Diese Entwicklung hat eine Auswirkung auf die Nutzerschnittstelle der Geräte. Die hohe Anzahl der zu steuernden Funktionen, die begrenzte Fläche der Anzeigeelemente sowie die begrenzte Menge möglicher haptischer Bedienaktionen erfordern neue Bedienkonzepte. In Folge dessen sind die heutigen interaktiven Geräte sehr komplex: Sie sind multimodal und multifunktional, sie filtern angezeigte Informationen und sie ordnen Bedienaktionen situationsabhängig Wirkungen zu.

Zusätzlich ergeben sich aus der in Bezug auf Nutzermerkmale heterogenen Nutzergruppe, eine beachtliche Menge an Qualitätsanforderungen, welchen das interaktive Gerät gerecht werden muss. Dabei spielt insbesondere das Qualitätsattribut „Gebrauchstauglichkeit“ eine wesentliche Rolle. Jedoch stoßen gängige Evaluationsmethoden – aufgrund der hohen Komplexität der Geräte – an ihre Grenzen.

In dieser Arbeit wird eine Analysemethode entwickelt, die Mensch-Maschine-Systeme auf Basis diskreter mathematischer Modelle beschreibt. Die Modelle werden anhand einer hierarchischen automatenbasierten Beschreibungstechnik dargestellt. Der Schwerpunkt liegt in der strukturierten Beschreibung des Interaktionsverhaltens; die graphische Gestaltung der Nutzerschnittstelle steht nicht im Zentrum.

Wesentliche Konzepte menügetriebener Nutzerschnittstellen werden zu einem konzeptuellen Modell zusammengefasst und in die Beschreibungstechnik integriert. Für die Modelle werden Eigenschaften, die in Bezug zur Gebrauchstauglichkeit stehen, identifiziert und unter Verwendung der Temporalen Logik formalisiert. Mit Hilfe der Technik Model-Checking wird automatisch geprüft, ob die Modelle diese Eigenschaften erfüllen.

Letztlich wird die praktische Anwendbarkeit der entwickelten Methode anhand einer umfangreichen Fallstudie aus dem Automotive-Bereich gezeigt.

Abstract

Electronic devices are constantly becoming more powerful and therefore more capable of providing a variety of different features. At the same time their size is being reduced, also constantly, by the manufacturers. This development has a considerable impact on the user interface of such devices. The high number of functions that have to be controlled, the limited size of the display, as well as the limited set of possible haptic commands require new user-interface concepts. Today's interactive devices are therefore complex: They are multimodal and multifunctional, they filter the displayed information and assign actions to the user's commands according to the situation.

Additionally with respect to user characteristics a heterogeneous user-group is constituted which makes a considerable amount of quality-requirement demands on an interactive device. In particular the quality attribute "usability" plays a significant role. Due to the high complexity of the devices, however, current evaluation methods encounter their limits.

In this thesis, a method of analysis is developed that describes human-machine systems by means of discrete mathematical models. The models are notated by a hierarchical automaton-based description technique. The main focus lies on the structured description of the interaction behaviour and not on the graphical design of the user interface.

The key concepts for the description of menu-driven user interfaces are combined within a conceptual model and integrated into the automaton-based description technique. Significantly, the properties of the models that influence the quality attribute "usability" are identified and formalized by means of temporal logic. With the aid of model checking the conformity of the models to these properties is automatically verified.

The practical applicability of the developed method is shown by means of an extensive case study from the automotive field.

Danksagung

Für die Möglichkeit, am Lehrstuhl Software & Systems Engineering wissenschaftlich tätig zu sein, und für die geduldige Betreuung dieser Arbeit danke ich zuerst Herrn Prof. Dr. Dr. h.c. Manfred Broy. Im gleichen Maße bedanke ich mich bei Frau Prof. Dr. Elisabeth André für die hilfreichen Anmerkungen und für die sofortige Bereitschaft, das Zweitgutachten dieser Arbeit zu übernehmen.

Ich danke den Kollegen aus dem kooperativen Projekt TUMMIC für die interdisziplinäre Zusammenarbeit, die den Anstoß für diese Arbeit gegeben hat, das sind: Markus Ablaßmeier, Chantal Ackermann, Verena Broy, Martin Gründl, Klaus Härtl und Christian Lange.

Für die vielfältige Unterstützung gebührt mein Dank Martin Leucker, Bernhard Schätz und Katharina Spies. Nicht weniger gilt mein Dank Benjamin Hummel, Klaus Lochmann, Patrick Nepper und Stefan Wagner für die Durchsicht des Manuskripts und zahlreiche hilfreiche Kommentare. Auch Sebastian Benz sei an dieser Stelle für die fruchtbaren Diskussionen gedankt.

Allen Kollegen, insbesondere Bastian Best, David Cruz, Florian Deißenböck, Martin Feilkas, Mario Gleirscher, Benjamin Hummel, Elmar Jürgens, Klaus Lochmann, Christian Pfaller und Wassiou Sitou, bin ich für die angenehme Atmosphäre am Lehrstuhl zu Dank verbunden.

Ich danke Jochen Klages, Frank Marschall, Silke Müller, Stefan Wagner und allen anderen Freunden für die Unterstützung und die stetige Ermutigung. Vor allem möchte ich mich bei Céline Laurent für die Geduld, die sie während der Erstellung dieser Arbeit mit mir hatte, bedanken.

Besonderer Dank gebührt meiner Familie Hans, Marcia, Tobias, Christoph und Sarah Winter – ohne die so vieles nicht möglich wäre.

Inhaltsverzeichnis

Kurzfassung **3**

Abstract **5**

1 Einleitung **13**
1.1 Problemstellung 18
1.2 Zielsetzung 19
1.3 Ergebnisse 19
1.4 Verwandte Arbeiten 21
1.5 Gliederung der Arbeit 24

2 Modellierung und Verifikation interaktiver Systeme **27**
2.1 Grundlegende Konzepte 27
2.1.1 Zustandsmaschinen 28
2.1.2 Gezeitete Zustandsmaschinen 29
2.1.3 Temporale Logik 30
2.2 Zustandsübergangsdiagramme 32
2.3 Zustandsbasiertes Systemmodell 37
2.3.1 Funktionen 37
2.3.2 Basisfunktion 38
2.3.3 Parallele Komposition 39
2.3.4 Alternative Komposition 40
2.3.5 Verbergen 40
2.3.6 Feedback 41
2.4 Abbildung der Zustandsübergangsdiagramme 42
2.5 Abbildung des Systemmodells 43
2.6 Werkzeugunterstützung 45
2.7 Zusammenfassung 46

3 Grundlagen der Mensch-Maschine-Interaktion **49**
3.1 Grundlegende Konzepte 49
3.2 Konzeptuelles Modell der Mensch-Maschine-Interaktion 50
3.2.1 Struktur 50
3.2.2 Dialoge und Dialogstränge 51
3.2.3 Menüs, Optionen und Optionsgruppen 53
3.2.4 Dialogschritt 54
3.2.5 Modus und Modi-Konfigurationen 55
3.2.6 Beispiel 55

3.3 Kognitive Fähigkeiten des Menschen . . . 56
3.3.1 Wahrnehmung . . . 57
3.3.2 Aufmerksamkeit . . . 57
3.3.3 Gedächtnis . . . 58
3.4 Grundsätze der Dialoggestaltung . . . 60
3.4.1 Aufgabenangemessenheit . . . 60
3.4.2 Selbstbeschreibungsfähigkeit . . . 61
3.4.3 Steuerbarkeit . . . 61
3.4.4 Erwartungskonformität . . . 62
3.4.5 Fehlertoleranz . . . 63
3.4.6 Individualisierbarkeit . . . 63
3.4.7 Lernförderlichkeit . . . 64
3.5 Zusammenfassung . . . 64

4 Modellbasierte Analyse von Nutzerschnittstellen 67
4.1 Erstellung der Analysemodelle . . . 67
4.1.1 Nutzer . . . 68
4.1.2 Mediator . . . 68
4.1.3 Applikation . . . 70
4.1.4 Kommunikationskanäle und Monitorvariablen . . . 70
4.2 Merkmale der Analysemodelle . . . 71
4.2.1 Struktur der Dialoge . . . 72
4.2.2 Voreingestellte Belegung der Dialogattribute. . . . 72
4.2.3 Voreingestellte Fortsetzung der Dialoge . . . 73
4.2.4 Zusammenspiel der Dialoge mit der Applikation . . . 73
4.2.5 Modi-Konfigurationen . . . 74
4.3 Bezug der Merkmale zu den Grundsätzen der Dialoggestaltung . . . 79
4.4 Beispiel „MP3-Spieler“ . . . 83
4.4.1 Merkmale des Analysemodells . . . 84
4.4.2 Bewertung . . . 91
4.5 Zusammenfassung . . . 92

5 Fallstudie „E60“ 93
5.1 Kernkonzepte des iDrive-Systems . . . 94
5.2 Gewählter Ausschnitt des iDrive-Systems . . . 97
5.3 Merkmale des Analysemodells . . . 97
5.3.1 Struktur der Dialoge . . . 98
5.3.2 Voreingestellte Belegung der Dialogattribute . . . 102
5.3.3 Voreingestellte Fortsetzung der Dialoge . . . 102
5.3.4 Zusammenspiel der Dialoge mit der Applikation . . . 102
5.3.5 Modi-Konfigurationen . . . 104
5.4 Bewertung . . . 107
5.5 Zusammenfassung . . . 110

6 Zusammenfassung und Ausblick 111
6.1 Zusammenfassung . . . 111

6.2 Ausblick 113

Literaturverzeichnis **117**

A Details zum Beispiel „MP3-Spieler“ **125**
A.1 Zustandsübergangsdiagramme 125
A.1.1 Orthogonaler Zustand „Player“ 125
A.1.2 Orthogonaler Zustand „User“ 127
A.1.3 Orthogonaler Zustand „Select“ 129
A.1.4 Zusammengesetzter Zustand „Off“ 130
A.1.5 Zustände „HomeMenu“ und „MusicMenu“ 131
A.1.6 Orthogonaler Zustand „Listen“ 136
A.1.7 Zusammengesetzter Zustand „NowPlaying“ 137
A.2 Tabellen der Modi-Konfigurationen 140
A.3 Übergänge zwischen den Modi-Konfigurationen 143

B Details zur Fallstudie „E60“ **145**
B.1 Zustandsübergangsdiagramme 145
B.1.1 Orthogonaler Zustand „User“ 145
B.1.2 Orthogonaler Zustand „Home“ 148
B.1.3 Orthogonaler Zustand „Navigation“ 150
B.1.4 Orthogonaler Zustand „AssistanceWindow“ 168
B.1.5 Orthogonaler Zustand „CheckControlMessage“ 169
B.1.6 Komponenten der Applikation 171
B.2 Tabellen der Modi-Konfigurationen 176
B.3 Übergänge zwischen den Modi-Konfigurationen 179

1 Einleitung

Im Gebiet Mensch-Maschine-Interaktion[1] wird ein Mensch-Maschine-System (MMS) in die drei Komponenten Mensch, Maschine und Umgebung untergliedert. Diese Komponenten interagieren miteinander, indem sie Informationen austauschen. Der fortlaufende und wechselseitige Austausch von Informationen dient dem Zweck, gemeinsam Aufgaben zu erledigen. Im Folgenden wird der Mensch in einem solchen System auch als *Anwender* oder *Nutzer* und die Maschine als *interaktives Gerät* bezeichnet.

Interaktive Geräte sind demnach in ein übergreifendes Gesamtsystem eingebettet, d. h. sie kommunizieren mit Nutzern, anderen Geräten und sind über Sensoren und Aktuatoren mit ihrer Umgebung verbunden. Aufgrund der Fortschritte im Bereich Mikroelektronik, sind die Geräte sehr leistungsfähig. Dadurch ist die Anzahl sowie die Vielfalt der Funktionen, die dem Nutzer über die *Nutzerschnittstelle*, im Folgenden auch als *Mensch-Maschine-Interface* (MMI) bezeichnet, angeboten werden, beständig gewachsen. Damit der Zugang zu den Funktionen flexibel gestaltet werden kann, verfügen die Geräte in der Regel über eine multimodale Nutzerschnittstelle, d. h. unterschiedliche Sinnesmodalitäten können zum Austausch von Informationen verwendet werden. Die Gestaltung der graphischen Ausgaben und der haptischen Eingaben ist bei Geräten von geringer Größe erschwert: die Einschränkung der Schnittstelle hat einerseits zur Folge, dass dem Nutzer in einer Situation gewisse Informationen vorenthalten werden, und andererseits, dass Eingabeaktionen in Abhängigkeit zur Vorgeschichte unterschiedliche Folgeaktionen bewirken. Diese Eigenschaften sind maßgeblich für die hohe Komplexität dieser Geräte verantwortlich [26, 80]. Damit nachhaltig eine hohe Qualität der Geräte gewährleistet werden kann, ist es notwendig, diese Komplexität zu beherrschen.

Die Norm ISO 9000 [6] bietet eine Terminologie für *Qualitätssicherung* (QS) im Allgemeinen an. Diese definiert ein *Produkt* als das Ergebnis eines Entwicklungsprozesses und die Qualität eines Produkts als den Grad, zu dem dessen inhärente *Merkmale* festgelegten *Anforderungen* genügen. Ein Merkmal ist eine kennzeichnende Eigenschaft und eine Anforderung ist ein Erfordernis, Bedarf oder eine Erwartung an die Merkmale eines Produkts. Die Festlegung einer Anforderung bezeichnet man als *Spezifikation*[2]. Die Anforderungen werden von *Stakeholdern*[3] festgelegt. Erfüllt ein Produkt eine festgelegte Anforderung, dann ist es *konform* zu dieser. Die Nichterfüllung einer festgelegten Anforderung wird als *Mangel* bezeichnet. Die Prüfung, ob ein Produkt die Spezifikation erfüllt, wird mit *Verifikation* bezeichnet. Die Prüfung, ob eine Spezifikation, die tatsächlichen Anforderungen enthält, nennt man *Validierung*.

[1] Der entsprechende englische Begriff ist *Human-Computer Interaction* (HCI).

[2] Die Norm ISO 9000 [6] spricht zwar von „festgelegten Anforderungen“, nicht aber von dem Begriff „Spezifikation“.

[3] Der entsprechende deutsche Begriff ist „Interessenvertreter“.

Die Validierung prüft somit die Adäquatheit der festgelegten Anforderungen. Unter Qualitätssicherung versteht man alle Aktivitäten des Entwicklungsprozesses, die die Qualität eines bereits bestehenden Produkts bestimmen oder ein Produkt in einer Art und Weise herstellen, dass die Erfüllung von Qualitätsanforderungen garantiert ist. Die Gesamtqualität eines Produkts lässt sich in *Qualitätsattribute* zerlegen.

Gebrauchstauglichkeit ist ein zentrales Qualitätsattribut interaktiver Geräte und wird in der Norm ISO 9241-11 [2] als der Grad definiert, zu dem der Nutzer seine Ziele in einem bestimmten Nutzungskontext mit den Maßen Effektivität, Effizienz und Zufriedenstellung erreichen kann. Die Effektivität ist bestimmt durch die Genauigkeit und die Vollständigkeit der Zielerreichung. Die Effizienz ist das Verhältnis der Effektivität zum eingesetzten Aufwand des Nutzers. Zufriedenstellung ist definiert als die „Freiheit von Beeinträchtigungen und positive Einstellung gegenüber der Nutzung des Produkts“ [2]. Diese Definition der Zufriedenstellung trifft keine Aussage darüber, wie die positive Einstellung des Nutzers zu Stande kommt. Die Qualitätsmanagementnorm ISO 9000 [6] ist in diesem Punkt genauer und definiert das Konzept Kundenzufriedenheit als die Wahrnehmung des Kunden zu dem Grad, in dem seine tatsächlichen Anforderungen erfüllt worden sind. Die Norm gibt dazu die Anmerkung, dass die Erfüllung der festgelegten Anforderungen nicht zwangsweise ausreicht, um die Kundenzufriedenheit sicherzustellen.

Die Gebrauchstauglichkeit interaktiver Geräte hat eine direkte Auswirkung auf die Produktivität der Anwender sowie deren Lernbereitschaft und auf die Anzahl der Bedienfehler [71, 95]. Außerdem stellt sie ein Schlüsselkriterium für die Akzeptanz des Geräts dar [68] und steht im Zusammenhang mit der Sicherheit des Anwenders. Hat beispielsweise ein im Kraftfahrzeug verbautes interaktives Gerät Mängel, die die Gebrauchstauglichkeit beeinträchtigen, besteht die Gefahr, dass der Fahrer, wenn er das Gerät beim Fahren benutzt, übermäßig visuell, manuell oder kognitiv beansprucht wird [82]. Diese übermäßige Beanspruchung kann zu einer Ablenkung des Fahrers von seiner Fahraufgabe führen. Ist der Fahrer abgelenkt, so ist die Verkehrssicherheit zu diesem Zeitpunkt beeinträchtigt.

Evaluationsmethoden des Usability-Engineering

Damit ein interaktives Gerät gebrauchstauglich ist, muss es bestimmte Anforderungen erfüllen. Diese Anforderungen werden *Nutzungsanforderungen* genannt. Die Menge der Nutzungsanforderungen enthält sowohl nicht-funktionale als auch funktionale Anforderungen. Drei Aspekte müssen beachtet werden [23]: Erstens ist eine frühzeitige und systematische Erhebung der Nutzungsanforderungen für das System erforderlich, zweitens muss sichergestellt werden, dass die erhobenen Nutzungsanforderungen den tatsächlichen Erwartungen aller Beteiligten entsprechen, und drittens, dass das zu entwickelnde System diese Anforderungen erfüllt.

Deshalb setzen Hersteller für die Erhebung der Nutzungsanforderungen und für die nachhaltige Gewährleistung einer hohen Gebrauchstauglichkeit einen spezifischen Entwicklungsprozess ein, der als *Usability-Engineering-Prozess* bezeichnet wird [81]. Die Aktivitäten dieses Prozesses sind stark interdisziplinär, sie erfordern eine integrierte

Anwendung des Wissens über kognitive Fähigkeiten der Anwender, ergonomischen Richtlinien, Gestaltung und insbesondere Software-Engineering.

	Inspektionsmethoden			Nutzertests		
	Heuristische Evaluierung	**Cognitive Walkthrough**	**Keystroke-Level Analysis**	**Thinking Aloud**	**Feldstudie**	**Fragebogen**
Benötigte Zeit	niedrig	mittel	hoch	hoch	mittel	niedrig
Anzahl der Nutzer	keine	keine	keine	3+	20+	30+
Anzahl der Evaluatoren	3+	3+	1–2	1	1+	1
Benötigte Ausrüstung	niedrig	niedrig	niedrig	hoch	mittel	niedrig
Benötigtes Fachwissen	mittel	hoch	hoch	mittel	hoch	niedrig

Tabelle 1.1: Vergleich gängiger Evaluationsmethoden (Quelle: [59])

Evaluationsmethoden stellen einen wesentlichen Bestandteil des Usability-Engineering-Prozesses dar und sind, nach [59], unterteilt in *Inspektionsmethoden* und *Nutzertests*. Im Folgenden werden gängige Evaluationsmethoden beschrieben (s. Tab. 1.1 für einen Vergleich).

Inspektionsmethoden. Bei Inspektionsmethoden handelt es sich um Verfahren, bei denen Gutachter systematisch prüfen, ob das Produkt festgelegten Anforderungen genügt.[4] Diese Anforderungen sind typischerweise aus grundlegenden Prinzipien oder allgemeinen Richtlinien abgeleitet. Gegenstand einer Inspektion können *Prototypen* unterschiedlicher Stufen oder das fertige Produkt sein. Das Ergebnis einer Inspektion ist eine Mängelliste, auf deren Grundlage Korrekturmaßnahmen vorgenommen werden können.

Damit die Objektivität der Prüfung gewährleistet ist, ist es notwendig, dass mehrere Gutachter – unabhängig von einander – Inspektionen durchführen. Auf Grund der Tatsache, dass für eine Inspektion keine Nutzer benötigt werden, verursachen Inspektionsmethoden in der Regel weniger Kosten als Nutzertests. Allerdings hängt die Effektivität einer Inspektion in einem starken Ausmaß vom Fachwissen der Gutachter sowie der Validität der Anforderungen ab [95].

Inspektionsmethoden unterscheiden sich in der Art und Weise der Durchführung und hinsichtlich dem Abdeckungsgrad der geprüften Abläufe. Zwei gängige Inspektionsmethoden sind *Heuristische Evaluierung* und *Cognitive Walkthrough* [81]. Bei der

[4] Eine Inspektion wird häufig auch als *Expertenevaluation* bezeichnet, da die Gutachter in der Regel Experten der Domäne oder Experten der Mensch-Maschine-Interaktion sind.

Heuristischen Evaluierung sind die Anforderungen durch eine kurze Liste anerkannter Prinzipien, den sog. Heuristiken, gegeben. Bei dem *Cognitive Walkthrough* werden kognitive Prozesse und insbesondere die kognitive Belastung des Nutzers in jedem Schritt bestimmt. Bei einigen Inspektionsmethoden werden Nutzermodelle eingesetzt. Ein Beispiel dafür ist die *Keystroke-Level Analysis*, eine einfache Variante der GOMS-Methode[5], die von Card, Moran und Newell [31] entwickelt wurde. Bei der *Keystroke-Level Analysis* erstellt der Gutachter für typische Aufgaben jeweils eine Sequenz der Bedienaktionen, die der Nutzer zur Erfüllung der Aufgabe durchführen muss. Auf Basis dieser Sequenz und einem kognitiven Modell des Nutzers wird eine Abschätzung der Aufgabenbearbeitungsdauer erstellt.

Nutzertests. Bei Nutzertests wird das Produkt an Probanden, die potentielle Nutzer repräsentieren, getestet und anhand empirischer Daten bewertet. Die Probanden werden gebeten typische Aufgaben mit dem Produkt durchzuführen. Die Tatsache, ob der Proband die Aufgabe fertigstellen konnte und wie lange er dazu gebraucht hat wird aufgezeichnet. Anhand dieser Daten können Mängel der Gebrauchstauglichkeit identifiziert werden. Das Prüfobjekt eines Nutzertests kann ebenfalls entweder ein Prototyp unterschiedlicher Stufe oder das fertige Produkt sein. Nutzertests ermöglichen es, potentielle Nutzer unmittelbar bei der Interaktion mit dem Produkt zu beobachten und sind deshalb eine elementare und unerlässliche Methode für die Prüfung interaktiver Geräte [81]. Sie haben allerdings den Nachteil, dass sie hohe Kosten verursachen.

Es gibt eine Reihe unterschiedlicher Methoden zur Durchführung eines Nutzertests. Bei dem Verfahren *Thinking Aloud* werden beispielsweise die Probanden aufgefordert, während der Interaktion laut mitzudenken. Somit ist ersichtlich, an welchen Stellen der Nutzer Konzepte des Produkts anders als vorgesehen interpretiert [81]. Nutzertests können entweder in einem Labor oder in der tatsächlichen Umgebung der Nutzer durchgeführt werden. Ein Test in der Umgebung des Nutzers wird als *Feldstudie* bezeichnet. Üblicherweise wird dem Probanden, nach Beendigung des Experiments, ein *Fragebogen* ausgehändigt. Fragebogen sind insbesondere dafür geeignet, die subjektive Wahrnehmung sowie die Bedürfnisse des Nutzers zu ermitteln.

Techniken des Software-Engineering

Für die gebrauchstaugliche Gestaltung interaktiver Geräte spielen – neben den Evaluationsmethoden – die Techniken des Software-Engineering eine wichtige Rolle: Getrieben durch den vermehrten Einsatz von sicherheitskritischen softwareintensiven Systemen, gewinnen Techniken, die die Komplexität dieser Systeme beherrschen, zunehmend an Bedeutung. Im Folgenden werden die drei Schlüsselaspekte *Modellbildung*, *Formale Verifikation* und *Werkzeugunterstützung* vorgestellt.

[5] Das Akronym GOMS steht für *Goals*, *Operators*, *Methods* und *Selection Rules*.

Modellbildung. Ein *Modell* ist ein Abbild der realen Welt und zeichnet sich dadurch aus, dass es Informationen, die für den gewählten Ausschnitt nicht relevant sind, weglässt. Die Konzentration auf wesentliche Aspekte sowie die Abstraktion von unwichtigen Details sind daher Grundgedanken bei der Erstellung eines Modells. Der Einsatzzweck eines Modells im Rahmen der Systementwicklung kann ganz unterschiedlich sein. Mit Hilfe eines Modells kann man u. a. Anforderungen an ein System festlegen, Kernkonzepte analysieren und visualisieren oder eine Grundlage für die Kommunikation schaffen. Die Repräsentation eines Modells erfolgt durch das Anwenden einer *Beschreibungstechnik*. In [27] werden Beschreibungstechniken nach dem Grad der Formalität klassifiziert. Eine informelle Beschreibungstechnik charakterisiert ein Modell mit Hilfe der natürlichen Sprache; eine semiformale Beschreibungstechnik besitzt zwar eine formale Syntax, jedoch ist die Semantik der syntaktischen Konstrukte nicht vollständig formal festgelegt; eine formale Beschreibungstechnik besteht aus einer formalen Syntax und einer formalen Semantik, festgelegt durch eine mathematische Notation. Erst die Verwendung einer formalen Beschreibungstechnik bietet den Entwicklern Möglichkeiten, die Komplexität des zu modellierenden Systems nachhaltig zu beherrschen [26]. Sie ermöglicht die Integration verschiedener Sichten auf das Modell, das Anwenden von Modelltransformationen, die gewisse Verfeinerungsbeziehungen garantieren, oder die automatische Quelltextgenerierung.

Formale Verifikation. Die Prüfung, ob ein Produkt eine Spezifikation erfüllt, wird als Verifikation bezeichnet. Liegt das Produkt als formales Modell vor und sind die Anforderungen als formale Eigenschaften des Systems festgelegt, können diese formal überprüft werden. Diesen Vorgang nennt man formale Verifikation. Ein Beispiel für eine formale Verifikationstechnik ist *Model-Checking*, ein Verfahren zur automatischen Verifikation bestimmter Eigenschaften von Modellen. Die Voraussetzung eines Modells für den Einsatz von Model-Checking ist, dass es einen endlichen Zustandsraum besitzt. Ein Model-Checker erzeugt den vollständigen Zustandsraum des zu prüfenden Modells und sucht diesen nach fehlerhaften Abläufen ab. Damit kann man ein Modell auf gewünschte Eigenschaften überprüfen und im Gegensatz zum Testen diese Eigenschaften bei Erfüllung auch garantieren. Ein Ablauf, in dem ein Fehlverhalten auftritt, bezeichnet man als *Gegenbeispiel*. Beispiele für Model-Checker sind SPIN [60], SMV [73] sowie NUSMV [33]. SPIN beschreibt Systeme anhand asynchroner Prozesse, SMV bzw. NUSMV dagegen sind symbolische Model-Checker auf Basis von synchron ausgeführten Modulen. Auf Grund der Tatsache, dass der Zustandsraum exponential mit der Anzahl der Variablen steigt, sind dem Einsatz von Model-Checkern, die auf einem expliziten Zustandsraum basieren, Grenzen gesetzt. Dies wird im Allgemeinen als *State-Explosion-Problem* bezeichnet. Deshalb werden Techniken, wie beispielsweise Binäre Entscheidungsdiagramme (engl.: *binary decision diagrams*, BDDs) [28] oder *Partial-Order-Reduction* [47], eingesetzt, um dem Problem entgegenzuwirken.

Werkzeugunterstützung. Mit dem Einsatz von Werkzeugen werden die Entwickler bei der Durchführung von Aktivitäten des Entwicklungsprozesses, wie beispielsweise

Entwurf, Prüfung oder Wartung, unterstützt. Werden mechanische, sich wiederholende Aufgaben automatisiert, können Flüchtigkeitsfehler der Entwickler vermieden werden. Außerdem wird die Konzentration auf konzeptuelle und inhaltliche Tätigkeiten durch diese Entlastung gefördert. Beispiele dafür sind mechanische Konsistenzüberprüfungen oder die automatische Generierung von Quelltext. Durch diese Unterstützung verspricht man sich eine gesteigerte Produktivität der Entwicklungstätigkeiten. Die Automatisierung mechanischer Tätigkeiten und die Sicherung der Durchgängigkeit benötigt in der Regel eine formale Repräsentation der zu unterstützenden Konzepte, d. h. eine formale Beschreibungstechnik.

1.1 Problemstellung

Bei der Entwicklung interaktiver Geräte werden viele Mängel, die die Gebrauchstauglichkeit beeinträchtigen, erst in den späten Phasen der Entwicklung gefunden, d. h. während der Implementierung, der Integration oder im Betrieb. Es ist bekannt, dass je später Mängel entdeckt werden, desto mehr Kosten verursacht deren Beseitigung [103]. Die Ursachen dafür, dass die Mängel erst spät entdeckt werden, sind im Folgenden aufgeführt:

- *Bei unvollständigen Prototypen können Wechselwirkungen zwischen Funktionen nicht beobachtet werden.* Einzelne Prototypen setzen oft nur isolierte Teile der Funktionalität um. Das Auftreten von Wechselwirkungen zwischen den Funktionen untereinander kann aber nur bei multifunktionalen Prototypen beobachtet werden. Dies kann dazu führen, dass bestimmte Nutzungsanforderungen bezüglich dieser Wechselwirkungen nicht geprüft werden.

- *Zu geringe Diversität der Evaluationsmethoden.* Es gilt die Regel, dass je mehr unterschiedliche Evaluationsmethoden eingesetzt werden, desto mehr Mängel werden aufgedeckt. Konzentriert man sich zu stark auf eine Art der Evaluation, besteht die Gefahr, eine gewisse Klasse an Mängeln zu übersehen. Nutzertests, zum Beispiel, sind eine elementare und unerlässliche Methode für die Prüfung interaktiver Geräte. Sie haben aber den Nachteil, dass sie Inkonsistenzen des Verhaltens der Nutzerschnittstelle häufig nicht entdecken [63].

- *Informelle Beschreibungstechniken.* Ein weiteres Problem besteht darin, dass die eingesetzten Beschreibungstechniken (Aufgabenmodelle, Szenarien etc.) entweder gar nicht oder nur bedingt formal fundiert sind. Beispiele für informelle Aufgabenmodelle sind *Concurrent Task Trees* (CTT) [88] oder *Use Case*-Diagramme der *Unified Modeling Language* (UML) [21]. Somit wird eine Spezifikation der Nutzungsanforderungen erstellt, die zum einen nur schwer auf Widersprüchlichkeit geprüft werden kann und zum anderen dem Softwareentwickler einen großen Interpretationsspielraum gewährt. Dies ist insbesonders für die Sicherung der Gebrauchstauglichkeit ein Problem, da der typischen Softwareentwickler in diesem Gebiet keine umfassende Kenntnisse besitzt.

1.2 Zielsetzung

Kernziel dieser Arbeit ist, durch die Verwendung einer modellbasierten Methode zur Entwicklung konzeptueller MMI-Analysemodelle, die Qualitätssicherung in den frühen Phasen der Produktentwicklung zu verbessern. Dabei liegt der Schwerpunkt auf einem prüfbaren formalen Modell des gesamten Mensch-Maschine-Systems, d. h., die fortlaufende Interaktion zwischen dem Anwender und dem interaktiven Gerät wird strukturiert beschrieben.

Die Erstellung eines solchen Modells ist schon in den frühen Phasen des Entwicklungsprozesses möglich. Somit kann bereits früh im Entwicklungsprozess auf Basis des formalen Modells eine effektive Analyse der Gebrauchstauglichkeit durchgeführt werden, wodurch Kosten im weiteren Verlauf der Entwicklung reduziert werden. Da die Methode auf einem automatischen Verfahren beruht, spart beispielsweise die Wiederholbarkeit der Bewertung Kosten. Das Erstellen eines formalen Modells ermöglicht außerdem eine Objektivierung der Bewertung, die unerlässlich ist, da die Entwicklung von interaktiven Geräten immer die Abwägung zwischen Anforderungen, die im Bezug zur Gebrauchstauglichkeit stehen, und Anforderungen, die im Bezug zu anderen Qualitätsattributen, beispielsweise dem Attribut „Sicherheit“, stehen, beinhaltet.

Der Einsatz formaler Beschreibungstechniken schafft zudem ein Rahmenwerk, in dem die konzeptuellen Modelle der unterschiedlichen Domänen integriert werden können [41, 57]. Präzise Modelle tragen nicht nur zum Verständnis der Nutzungsanforderungen bei, sondern können im späteren Entwicklungsprozess wiederverwendet werden, wie beispielsweise bei der konstruktiven Ableitung von Testfällen [17]. Vor allem für eine effektive Werkzeugunterstützung ist eine ausgereifte Modelltheorie eine unabdingbare Grundlage [26]. Folglich besteht die Notwendigkeit einer durchgängigen Modellbildung zur besseren Integration der Usability-Engineering-Techniken in den Softwareentwicklungsprozess.

1.3 Ergebnisse

Das Ergebnis dieser Arbeit ist eine modellbasierte Methode zur systematischen Entwicklung konzeptueller MMI-Analysemodelle. Die Methode bildet den Rahmen der Arbeit und integriert die folgenden Ergebnisse:

- *Eine automatenbasierte Beschreibungstechnik für die hierarchische Strukturierung interaktiver Systeme.* Für die Beschreibung des Mensch-Maschine-Systems werden Automaten eingesetzt, die mit *gezeiteten Automaten* [22, 52] verwandt sind. Das Automatenmodell der gezeiteten Automaten wird erweitert durch Konzepte von Henzinger [58] und Schätz [93, 94]. Die von Henzinger vorgeschlagene explizite Trennung von Kontroll- und Datenfluss ermöglicht eine modulare, hierarchische Strukturierung der Transitionen. Die von Schätz eingeführten Konzepte ermöglichen eine dienstorientierte Sicht auf das System und erlauben das Beschreiben von partiellem Verhalten. Die in dieser Arbeit durchgeführte

Formalisierung dieser Konzepte auf Basis von gezeiteten Automaten ermöglicht einen direkten Übergang in eine Beschreibung, die durch einen Model-Checker analysiert werden kann, da eine konkrete Operationalisierung gegeben ist.

- *Ein konzeptuelles Modell für die strukturierte Beschreibung von Mensch-Maschine-Systemen.* Das konzeptuelle Modell beschreibt die grundlegenden Konzepte der Mensch-Maschine-Interaktion und deren Beziehungen zueinander. Diese Konzepte sind als explizite Konstrukte in die automatenbasierte Beschreibungstechnik integriert. Die Entwicklung von Mensch-Maschine-Systemen ist getrieben durch die Aufgaben des Nutzers, die mit Hilfe des interaktiven Geräts bearbeitet werden. Das Prinzip Nutzerzentrierung sieht vor, dass sich die Struktur der *Dialoge* an der Struktur dieser Aufgaben orientiert. Die Interaktion zwischen dem Nutzer und einem multifunktionalen interaktiven Gerät stellt in der Regel eine verzahnte Bearbeitung mehrerer Aufgaben dar. Für die Beschreibung solcher nebenläufigen Dialoge, wurde das Konzept der *Dialogstränge* eingeführt. Die automatenbasierte Beschreibungstechnik integriert diese Konzepte und ermöglicht die präzise Beschreibung der verzahnten Aufgabenbearbeitung. Dazu ist eine Strukturierung des Systems in die Komponenten „Nutzer“, „Mediator“ und „Applikation“ vorgegeben. Außerdem ist die Kommunikation zwischen diesen Komponenten auf Basis des Modells kategorisiert. Bei der Entwicklung der Analysemodelle stehen die Konzepte für die Strukturierung der Interaktion, die im konzeptuellen Modell beschrieben sind, im Vordergrund.

- *Ein Analyseverfahren, dass auf der Verifikationstechnik Model-Checking basiert.* Ein weiteres Ergebnis ist ein Analyseverfahren zur Bewertung der erstellten Modelle hinsichtlich des Aspekts Gebrauchstauglichkeit. Dazu sind in der Arbeit, auf der Grundlage des konzeptuellen Modells, eine Reihe an Merkmalen der Dialogstruktur beschrieben und unter Verwendung der Temporalen Logik formalisiert. Diese Merkmale charakterisieren dynamische Eigenschaften der Interaktion. Aus allgemeinen Grundsätze der Dialoggestaltung sind auf Basis dieser Merkmale exemplarisch Prüfkriterien abgeleitet. Somit ist es möglich, die Mensch-Maschine-Interaktion eines Bedienkonzepts auf Konformität zu diesen festgelegten Qualitätskriterien zu prüfen. Erst die strukturierte Beschreibung der Interaktion anhand eines formalen Modells ermöglicht die präzise Festlegung solcher Kriterien. Das Analyseverfahren stellt in der Klassifizierung von [59] eine Inspektionsmethode dar.

- *Eine Fallstudie aus dem Bereich Automotive.* Als Grundlage für die Fallstudie dient das BMW iDrive-System, das in der E60-Fahrzeugreihe verbaut worden ist. Die Fallstudie zeigt anhand einer Reihe ausgewählter Aspekte, die praktische Anwendbarkeit der entwickelten Modellierungs- und Analysemethode. Ferner zeigt die Fallstudie, dass die Methode in der Lage ist, Mängel des Bedienkonzepts – in Bezug zu den abgeleiteten Prüfkriterien – zu identifizieren.

1.4 Verwandte Arbeiten

Im Zentrum dieser Arbeit steht zum einen eine automatenbasierte Beschreibungstechnik für die Modellierung von Mensch-Maschine-Systemen, und zum anderen eine Analysemethode zur Bewertung der Modelle, basierend auf der Verifikationstechnik Model-Checking. Im Folgenden werden Arbeiten herausgestellt, die sich mit der Beschreibung oder mit der Analyse von Mensch-Maschine-Systemen beschäftigen, sowie deren Schwächen in Bezug auf die Zielsetzung dieser Arbeit beschrieben.

Interaktoren. Das Konzept der *Interaktoren* ist in zahlreichen Arbeiten behandelt (z. B. in [43, 78, 72, 37]). Interaktoren sind strukturelle Einheiten und dienen zur Unterteilung einer Nutzerschnittstelle. Sie kapseln einen Zustand und kommunizieren mit ihrer Umgebung mittels Ereignissen. Interaktoren repräsentieren *Widgets* und werden bei der Umsetzung von Nutzerschnittstellen eingesetzt. Das allgemeine Prinzip eines Interaktors ist wie folgt: Wenn ein Interaktor eine Eingabe vom Nutzer bekommt, dann leitet er diese an zwei interne Komponenten weiter. Diese sind dann verantwortlich für die Ausgabe der Darstellung und die Weiterleitung von Daten an weitere Interaktoren, die sich auf einer abstrakteren Ebene befinden. Interaktoren sind ähnlich zum Model-View-Controller-Muster, das initial für die Programmiersprache SMALLTALK eingeführt worden ist [66]: Ein Interaktor (der *Controller*) ist eine Komponente, die einen gekapselten Zustand besitzt. Der Interaktor verfügt über eine Menge an Ausgabekanälen (den *Views*), die zusammen mit den Ausgaben der anderen Interaktoren die Anzeige ergeben. Zudem verfügt der Interaktor über eine Menge an Eingabekanäle, über die er von Interaktoren, die einer abstrakteren Ebene angehören (das *Model*), Daten erhält. Das Konzept wurde in der Literatur mit unterschiedlichen Formalismen beschrieben. *Interaktoren* stellen somit ein schichtenbasiertes Strukturierungsmuster für die konkrete Umsetzung von Nutzerschnittstellen dar. Für die Erstellung von abstrakten Analysemodellen, wie in dieser Arbeit benötigt, ist das Konzept der Interaktoren, da es zu feingranular ist, nicht direkt geeignet.

Zustandsübergangsdiagramme. Zustandsübergangsdiagramme sind ein gängiges Mittel zur Beschreibung von Systemen. Auf Grund der Tatsache, dass Beschreibungen von komplexen Systemen mit einfachen Zustandsübergangsdiagrammen, d. h. ohne Strukturierungskonzepte, schnell unübersichtlich werden, sind eine Reihe von Erweiterungen entwickelt worden. Bekannte Beispiele sind STATECHARTS von Harel [56] oder AUTOFOCUS [62]. Diese Beschreibungstechniken unterstützen den Entwickler anhand von Strukturierungskonzepten, wie beispielsweise zusammengesetzte Zustände bzw. verschachtelte Komponenten. In dieser Arbeit wird eine vereinfachte Variante dieser Beschreibungstechniken verwendet, um die automatische Verifikation zu erleichtern. Wasserman [104] hat eine Erweiterung einfacher Zustandsübergangsdiagramme speziell für die Beschreibung Kommandozeilen-basierter Nutzerschnittstellen entwickelt. Die Erweiterungen von Wasserman sind aber nicht mächtig genug, um multifunktionale, multimodale Nutzerschnittstellen zu beschreiben.

In [61] verwendet Horrocks STATECHARTS zur Realisierung von Nutzerschnittstellen. Horrocks strukturiert dabei die Nutzerschnittstellen mit Hilfe einer speziellen Architektur, die er als *Interface-Control-Model Architecture* bezeichnet. Horrocks stellt durch diese Architektur die Beschreibung der Kontrollstruktur ins Zentrum. Die Architektur dient der konkreten Realisierung von Nutzerschnittstellen; spezielle modellbasierte Analysetechniken werden nicht beschrieben. Im Gegensatz dazu verfolgt die Modellbildung in dieser Arbeit das Ziel, eine Bewertung hinsichtlich der Gebrauchstauglichkeit durchzuführen.

In [100] verwendet Thimbleby ebenfalls STATECHARTS zur Beschreibung von Nutzerschnittstellen. Allerdings wählt er dabei eine deutlich abstraktere Ebene als Horrocks und leitet aus den Zustandsübergangsdiagrammen eine Graphen-basierte Repräsentation der Bedienlogik ab, die er mit Hilfe von Algorithmen, bekannt aus dem Bereich Graphen-Theorie, analysiert. Verschiedene Algorithmen werden vorgestellt und es wird erläutert welche Aussagen im Bezug zur Gebrauchstauglichkeit aus den Ergebnissen der Analyse abgeleitet werden können [101]. Diese Methode hat den Vorteil, dass mit Hilfe gewichteter Kanten Aufwands- und Kostenberechnungen im Bezug auf Nutzeraktivitäten durchgeführt werden können. Der Nachteil dieser Methode ist, dass die Graphen nur für abstrakte Modelle der Nutzerschnittstelle überschaubar und handhabbar sind. Somit kann das Zusammenspiel zwischen der Nutzerschnittstelle und der Applikation durch diese Methode nicht analysiert werden. Denn die Modellierung einer komplexen Applikation führt zu einem umfangreichen Zustandsraum, der, explizit dargestellt, nicht überschaubar ist.

Formale Prüfung von Nutzerschnittstellen. Es gibt eine Reihe von Arbeiten, die sich mit der formalen Prüfung von Nutzerschnittstellen beschäftigen. In Tab. 1.2 sind diejenigen Arbeiten, die im Folgenden betrachtet werden, aufgelistet.

Rushby analysiert in [92] das Verhalten von Autopiloten mit Hilfe der formale Sprache Murϕ [40]. Im Vordergrund stehen bei Rushby die sogenannten *Automation-Surprises*, die genau dann auftreten, wenn ein automatisiertes System sich anders verhält, als es der Nutzer erwartet. Um solche Mängel zu entdecken, modelliert Rushby sowohl den Nutzer als auch das System als Zustandsmaschine und versucht dann Abläufe

Autor	Quelle	Jahr	Modelle
Berstel et al.	[18]	2005	VEG/SPIN
Kistler	[65]	2004	NuSMV
Campos und Harrison	[30]	2001	Interaktoren/SMV
Loer und Harrison	[69]	2001	Ofan/Statecharts/SMV
Paternó und Santoro	[89]	2001	CTT/LOTOS/CADP
Rushby	[92]	2001	Murϕ
d'Ausbourg et al.	[38]	1998	Interaktoren/Lustre
Dwyer et. al.	[45]	1997	SMV
Abowd et al.	[8]	1995	PPS/SMV

Tabelle 1.2: Arbeiten zur formalen Prüfung von Nutzerschnittstellen

zu finden, in denen das mentale Modell des Nutzers nicht mehr mit dem Zustand des Systems übereinstimmt. Der Zustand des Nutzers, in dem er die Wirkung seiner Bedienaktionen nicht mehr vorhersagen kann, wird auch als *Mode-Confusion* bezeichnet.

Degani modelliert in [39] interaktive Geräte mit dem Ziel, die Stellen, an denen *Mode-Confusions* auftreten, zu identifizieren. Als Beschreibungstechnik setzt er Statecharts ein. Die Modelle sind anhand eines Rahmenwerks, das er als *Ofan-Framework* bezeichnet, strukturiert. Allerdings beschreibt Degani kein Verfahren zur formalen Analyse der Modelle. Deswegen erweitert Loer in [69] die Arbeit von Degani dahingehend, dass er eine Methodik beschreibt, die es erlaubt, die Modelle von Degani mit Hilfe des Verfahrens Model-Checking automatisch zu analysieren. Die temporallogischen Eigenschaften, die Loer prüft, sind aus den allgemeinen Mustern von Dwyer [44] abgeleitet und durch die Prinzipien von Nielsen [81] begründet. Die Schwäche in der Arbeit von Loer liegt darin, dass sowohl das *Ofan-Framework* als auch die Muster der temporallogischen Eigenschaften zu allgemein gehalten sind: Auf Basis dieses Ansatzes können beispielsweise keine Aussagen über die Struktur von Menüsysteme getroffen werden und es ist nicht möglich aus den abstrakten Mustern konkreten Vorgaben für die Gestaltung von Menüsystemen abzuleiten.

Kistler modelliert in [65] Nutzerschnittstellen aus dem Automotive-Bereich direkt in der Eingabesprache des Model-Checkers NuSmv [33]. Es werden die graphischen Widgets der Bedienkonzepte abstrakt beschrieben. Dies hat zur Folge, dass sich sowohl die Strukturierung der Interaktion als auch die Eigenschaften an der graphischen Darstellung orientieren. Die Eigenschaften sind als Muster von temporallogischen Formeln angegeben. Allerdings benötigen diese Muster eine Menge an Hilfsvariablen, deren Belegungen im Zusammenhang mit der Bedienlogik nicht klar hervorgehen. Die Beschreibung der Bedienlogik mit Hilfe eines unstrukturierten Automaten, der direkt in der Eingabesprache des Model-Checkers NuSmv kodiert ist, ist für Systeme mit einem realistischen Umfang nicht geeignet.

Es gibt eine Reihe weiterer Arbeiten, die Modelle von Nutzerschnittstellen mit dem Verfahren Model-Checking auf bestimmte Eigenschaften prüfen. Paternò und Santoro [89] formalisieren Concurrent-Task-Trees-Spezifikationen [76] mit Lotos [20] und setzen dann für die Prüfung der Eigenschaften den Model-Checker Cadp [48] ein. Campos [30] bildet Interaktoren auf die Eingabesprache des Model-Checkers Smv [73] ab. Als Beispiel wird das Autopilot-Beispiel aus [92] adaptiert. Ausbourg stellt in [38] einen Ansatz vor, der Interaktoren mit Hilfe des Lustre-Rahmenwerks [32] untersucht. Berstel et al. modellieren in [18] Widgets anhand eines Rahmenwerks, das mit Veg bezeichnet wird. Die Veg-Beschreibungen werden in den Model-Checker Spin [60] überführt und analysiert. In [8] wird die Notation Pps [87] verwendet, um Dialoge zu spezifizieren und mit dem Model-Checkers Smv auf bestimmte Eigenschaften zu prüfen. In [45] werden Abstraktions-Techniken für Menüsysteme vorgestellt. Die genannten Arbeiten haben gemeinsam, dass sie keine direkte Unterstützung für die strukturierte Modellierung und Analyse von menübasierten Nutzerschnittstellen anbieten.

XML-basierte Beschreibungssprachen. Es wurden einige Beschreibungssprachen für Nutzerschnittstellen entwickelt, die auf der Auszeichnungssprache XML basieren. Beispiele für solche Beschreibungssprachen sind UIML [9], XUL [77] oder XAML [74]. Ein Überblick über verschiedene Sprachen ist in [96] zu finden. Diese Beschreibungssprachen sind in der Regel auf spezielle Technologien oder Programmiersprachen zugeschnitten und stellen die Beschreibung der Darstellung in den Vordergrund. Deswegen sind sie als Grundlage für eine konzeptuelle Analyse ungeeignet. In [67] ist ein Vergleich unterschiedlicher Beschreibungssprachen, die im Automotive-Bereich erstellt worden sind, gegeben. Diese Arbeiten, sowie die in [67] erstellte Sprache, besitzen die gemeinsame Schwäche, die Bedeutung der Konstrukte durch eine Abbildung auf objektorientierte Rahmenwerke zu definieren. Somit sind die Konzepte dieser Sprachen, einerseits, sehr konkret auf die objektorientierten Konstrukte zugeschnitten und, andererseits, bleibt die genaue Bedeutung der Konstrukte sowie deren Zusammenhänge unklar.

1.5 Gliederung der Arbeit

Die Arbeit ist in sechs Kapitel unterteilt. Diese sind wie folgt strukturiert.

Kapitel 2 stellt die in dieser Arbeit eingesetzte automatenbasierte Beschreibungssprache vor. Ausgehend vom grundlegenden Systemmodell und den Grundlagen der zustandsorientierten Modellierung interaktiver Systeme wird die Temporale Logik CTL beschrieben. Zusätzlich wird in die Verifikationstechnik Model-Checking eingeführt, indem die Grundprinzipien dieser Technik erklärt werden. Danach wird die graphische Darstellung komplexer Systeme durch verschachtelte Zustandsübergangsdiagramme gezeigt und die Semantik der Diagramme durch eine Abbildung auf das Systemmodell festgelegt. Schließlich wird die Abbildung dieser Beschreibungssprache auf die Eingabesprache des Model-Checkers NUSMV angegeben. Abschließend wird die Werkzeugunterstützung, die entwickelt worden ist, beschrieben.

Kapitel 3 führt in den Bereich der Mensch-Maschine-Interaktion ein und beschreibt das konzeptuelle Modell zur Strukturierung der Interaktion. Die wichtigsten Begriffe der Mensch-Maschine-Interaktion werden definiert und anschließend wird das konzeptuelle Modell vorgestellt. Nach der Einführung in die grundlegenden kognitiven Fähigkeiten des Menschen, werden die Grundsätze der Dialoggestaltung der ISO 9241-110 beschrieben und unter Verwendung weiterer Prinzipien und Richtlinien diskutiert.

Die Methode für die modellbasierte Analyse von Mensch-Maschine-Systemen wird in Kapitel 4 vorgestellt. Zuerst wird gezeigt, auf welche Art und Weise das konzeptuelle Rahmenwerk in die Beschreibungssprache integriert ist. Anschließend werden die identifizierten Merkmale beschrieben und Prüfkriterien aus den Grundsätzen der Dialoggestaltung abgeleitet. Das Kapitel schließt mit der Modellierung und Analyse eines Beispiels.

Kapitel 5 evaluiert die modellbasierte Analyse anhand einer Fallstudie. Die Grundlagen des iDrive-Systems der Firma BMW werden eingeführt und der gewählte Aus-

schnitt des iDrive-Systems wird beschrieben. Darauffolgend wird der gewählte Ausschnitt mit Hilfe der Methode modelliert und analysiert. Die Ergebnisse der Analyse werden angegeben und die Liste der Mängel wird diskutiert.

Die Arbeit endet mit Kapitel 6, das die Ergebnisse zusammenfasst sowie einen Ausblick gibt.

2 Modellierung und Verifikation interaktiver Systeme

Dieses Kapitel beschreibt diejenigen Grundlagen der Systemmodellierung und der Verifikation, auf denen die Arbeit im Weiteren aufbaut. Zuerst wird in die Zustandssicht der Methode FOCUS [27] eingeführt. Dazu werden *Zustandsmaschinen* und eine Variante von diesen, die *gezeiteten Zustandsmaschinen*, vorgestellt. Dann wird eine Art der *Temporalen Logik* vorgestellt mit deren Hilfe in dieser Arbeit Eigenschaften spezifiziert werden. Anschließend wird eine hierarchische Variante von *Zustandsübergangsdiagrammen* zur graphischen Darstellung der Zustandsmaschinen angegeben. Diese ist verwandt mit der Harelschen Notation STATECHARTS [56]. Danach wird ein zustandsbasiertes Systemmodell beschrieben, das sich durch eine explizite Trennung von Kontroll- und Datenfluss auszeichnet. Zum Abschluss des Kapitels werden die Abbildungen zwischen den Notationen beschrieben. Es wird dargestellt, wie die Zustandsübergangsdiagramme in das zustandsbasierte Systemmodell überführt werden können und wie dieses Systemmodell auf die Eingabesprache des Model-Checkers NUSMV [33] abgebildet werden kann.

2.1 Grundlegende Konzepte

FOCUS [27] ist eine Methode für die formale Spezifikation sowie für die schrittweise Entwicklung *interaktiver Systeme*. FOCUS erlaubt die Modellbildung aus unterschiedlichen Sichten und bietet insbesondere die Möglichkeit, zwischen den Sichten zu wechseln. Damit die in dieser Arbeit erstellten Modelle mit Hilfe der Verifikationstechnik Model-Checking geprüft werden können, sind sie in der Zustandssicht beschrieben. Prinzipiell würde die gewählte Art der zustandsorientierten Systembeschreibung eine Abstraktion der Komponentenbeschreibung des Systems zu einem funktionalen Schnittstellenverhalten erlauben [26].

Ein interaktives System besteht aus einer Familie verteilter eigenständiger *Komponenten*. Eine Komponente ist ein interaktives System und besitzt eine Menge an *Eingabe-* und *Ausgabekanälen*. Durch diese gerichteten Kanäle sind die Komponenten miteinander verbunden. Mit Hilfe der Kanäle kommunizieren die Komponenten miteinander, indem sie *Nachrichten* austauschen. Durch einen Kanal wird eine Nachricht an einen Empfänger adressiert. Der Nachrichtenaustausch erfolgt *asynchron*[1], d. h., die Senderkomponente legt unabhängig von der Bereitschaft der Empfängerkomponente eine Nachricht auf den Kanal und setzt ihren *Ablauf* fort. Die Empfängerkomponente verarbeitet entweder die Nachricht oder sie wird vom Kanal zwischengespeichert. Der Ablauf einer Komponente entspricht Strömen von Eingabenachrichten, die Ströme

[1] Den asynchronen Nachrichtenaustausch bezeichnet man auch als *implizites Puffern* oder *nichtblockierendes Senden*.

von Ausgabenachrichten als Resultat erzeugen. Der Ablauf eines Systems ist *zeitsynchron*, d. h., alle Ströme sind in Zeitintervalle eines globalen *Takts* unterteilt. Das gewünschte Verhalten einer Komponente wird durch eine Funktion zwischen den Ein- und Ausgabeströmen spezifiziert. Im Folgenden werden solche stromverarbeitenden Funktionen durch Zustandsmaschinen beschrieben.

2.1.1 Zustandsmaschinen

Ein gängiges Modell für interaktive Systeme sind Zustandsmaschinen[2] [24]. Bei dieser Art der Modellierung wird das Verhalten eines Systems durch die Angabe der Menge der *Zustände* und der bei den Abläufen durchgeführten *Zustandsübergänge* beschrieben. Die Zustände eines Systems werden durch *Belegungen* von *Zustandsattributen*, die durch *Variablen* gegeben sind, angegeben.

Eine Variable v ist ein Bezeichner und wird als Element der Gesamtheit der Variablen $\mathbb{V}$ modelliert. Jede Variable v ist einem Typ, bezeichnet mit $type(v)$, zugeordnet. Sei $V \subseteq \wp(\mathbb{V})$ eine Menge von Variablen. Eine Belegung σ ist eine Abbildung, die jeder Variable aus V einen Wert zuweist:

$$\sigma \,:\, V \to \bigcup_{v \in V} type(v),$$

wobei $\forall\, v \in V :\ \sigma(v) \in type(v)$. Mit $[V]$ wird die Menge aller Belegungen für die Variablen aus V bezeichnet. Mit Σ wird die Menge aller Belegungen bezeichnet. Die Menge der Variablen V für eine Belegung $s \in [V]$ wird mit $vars(x)$ bezeichnet.

Seien V und V' disjunkte Mengen von Variablen mit den Belegungen $s \in [V]$ und $s' \in [V']$, dann ist die direkte Summe der Belegungen wie folgt definiert:

$$(s \oplus s').v \;=\; \begin{cases} s(v) & \text{für } v \in V \\ s'(v) & \text{für } v \in V'. \end{cases}$$

Seien V und V' Mengen von Kanälen und $V \subseteq V'$. Für eine Belegung $s \in [V']$ ist die Beschränkung, bezeichnet mit $s|_V$, wie folgt definiert:

$$\begin{aligned} vars(s|_V) &= V \\ (z \oplus z')|_{vars(z)} &= z\,, \end{aligned}$$

wobei $z' \in [V' \backslash V]$ und $z \in [V]$.

Eine Zustandsmaschine $(\Sigma,\, \sigma_0,\, \Delta)$ ist durch eine Zustandsmenge Σ, einen Startzustand σ_0 und einer Übergangsfunktion $\Delta : \Sigma \to \wp(\Sigma)$ gegeben. Eine Zustandsmaschine erzeugt endliche oder unendliche Ströme von Zuständen, die Abläufe der Zustandsmaschine. Ein Ablauf der Länge $n \in \mathbb{N} \cup \{\infty\}$ ist geben durch die Sequenz

$$\{\ \sigma_i \in \Sigma \mid\ 0 \le i \le n\ \wedge\ \sigma_{i+1} \in \Delta.\sigma_i\ \}.$$

[2] Die Begriffe „Zustandsmaschine“ (engl.: *state machine*) und „Automat“ (engl.: *automaton*) werden in dieser Arbeit synonym verwendet.

Eigenschaften von Zuständen können durch Prädikate beschrieben werden. Ein *Zustandsprädikat* mit der Signatur $P : \Sigma \to \mathbb{B}$ charakterisiert eine Menge von Zuständen und kann durch eine logische Formel spezifiziert werden. Ein Prädikat ist für eine Zustandsmaschine $(\Sigma, \sigma_0, \Delta)$ *stabil*, wenn die Gleichung $P(\sigma) \wedge \sigma' \in \Delta.\sigma \Rightarrow P(\sigma')$ für alle Zustande σ und σ' gilt. Ein stabiles Prädikat ist eine *Invariante* der Zustandsmaschine, wenn zusätzlich $P(\sigma_0)$ gilt.

2.1.2 Gezeitete Zustandsmaschinen mit Ein- und Ausgabenachrichten

Gezeitete Zustandsmaschinen mit Ein- und Ausgabenachrichten stellen eine Variante zur Beschreibung von Systemen dar [22, 53]. Da das System von einen diskreten globalen Takt getrieben ist, ist die Belegung eines Zeitintervalls genau durch einen Übergang beschrieben; das heißt, in jedem Zeitintervall t empfängt die Zustandsmaschine eine endliche Menge an Eingabenachrichten und verschickt eine endliche Menge an Ausgabenachrichten.

Eine gezeitete Zustandsmaschine $(I, O, \Sigma, \sigma_0, \Delta)$ ist durch eine Menge von Eingabevariablen I, eine Menge von Ausgabevariablen O, eine Zustandsmenge Σ, einen Startzustand σ_0 und einer Übergangsfunktion: $\Delta : (\Sigma \times [I]) \to \wp(\Sigma \times [O])$ gegeben.

Ein Zustandsmaschine ist *total*[3], wenn sie für jede Eingabe eine Ausgabe produziert, d., h. die folgende Aussage gilt: $\forall\, \sigma \in \Sigma,\ x \in [I] : \ \Delta(\sigma, x) \neq \emptyset$. Sonst ist die Zustandsmaschine *partiell*.

Eine Zustandsmaschine beschreibt einen Moore-Automaten [75], wenn die Belegung der Ausgabekanäle nur von der Belegung des lokalen Zustands abhängt. Das heißt, für alle $x \in [I]$ und $\sigma \in \Sigma$ gilt die folgende Eigenschaft:

$$(\sigma', y) \in \Delta(\sigma, x) \ \Rightarrow\ \forall\, z \in [I] : \exists\, \sigma'' : \ (\sigma'', y) \in \Delta(\sigma, z).$$

Somit kann sich die Eingabe eines Schritts frühestens im darauffolgenden Schritt auf die Ausgabe auswirken. Diese Eigenschaft garantiert, dass die Komposition von totalen Moore-Automaten – auch bei Rückkopplung – einen totalen Moore-Automaten ergibt [22, 53].

Im Folgenden ist die Komposition zweier Zustandsmaschinen $M_i = (I_i, O_i, \Sigma_i, \sigma_i^0, \Delta_i)$ für $i \in \{1, 2\}$ beschrieben. Die lokalen Variablen der zusammengesetzten Maschine sind $L = L_1 \cup L_2$, wobei $L_1 = (I_2 \cap O_1)$ und $L_2 = (I_1 \cap O_2)$.

Für die Ein- und Ausgabevariablen erhält man $I = (I_1 \cup I_2) \backslash L$ und $O = (O_1 \cup O_2) \backslash L$. Der Zustandsraum ist das Kreuzprodukt der Zustände $\Sigma = \Sigma_1 \times \Sigma_2$. Der Anfangszustand ist $\sigma^0 = (\sigma_1^0, \sigma_2^0)$. Die Übergangsfunktion ist definiert durch die Gleichung:

$$\begin{aligned}\Delta(\,(\sigma_1, \sigma_2), x\,) \ = \ \{\, (\,(\sigma_1', \sigma_2'\,), z|_O\,) \mid \exists\, z \in [I \cup O \cup L] : \ z|_I = x \\ \wedge\ \forall\, i \in \{1, 2\} : \ (\sigma_i', z|_{O_i}) \in \Delta_i(\sigma_i, z|_{I_i})\ \}.\end{aligned}$$

[3]Eine Zustandsmaschine mit dieser Eigenschaft wird auch *input enabled* oder *reactive* genannt.

2.1.3 Temporale Logik

Die Temporale Logik[4] ist eine Erweiterung der klassischen mathematischen Logik. Statt einzelner Zustände werden unendliche Folgen von Zuständen betrachtet. Somit können Eigenschaften interaktiver Systeme mit Hilfe der Temporalen Logik spezifiziert werden. Durch die Entwicklung automatischer Beweisverfahren, das sog. Model-Checking[5], hat die Temporale Logik für die Praxis an Bedeutung gewonnen. Das Verfahren Model-Checking verwendet für die Beschreibung des Verhaltens interaktiver Systeme *Kripkestrukturen* [47].

Sei AP eine endliche Menge von atomaren Aussagen. Eine Kripkestruktur $K = (\Sigma, \Sigma_0, \Delta, L)$ für AP ist durch einen endlichen Zustandsraum Σ, eine Menge an Startzuständen Σ_0, einer Übergangsfunktion $\Delta : \Sigma \to \wp(\Sigma)$ und einer Funktion $L : \Sigma \to \wp(V)$, die jeden Zustand mit der Menge von atomaren Aussagen markiert, die der Zustand erfüllt, gegeben. Sei $p \in AP$ und $\sigma \in \Sigma$, dann gilt $p \in L(\sigma) \Leftrightarrow \sigma(p)$. Es ist in der Regel gefordert, dass Δ total ist, d. h. es muss die folgende Aussage gelten: $\forall\, \sigma \in \Sigma : \exists\, \sigma' \in \Sigma : \sigma' \in \Delta(\sigma)$. Ein *Pfad* π der Kripkestruktur ist ein unendlicher Strom von Zuständen $\langle \sigma_0, \sigma_1, \sigma_2, \ldots \rangle$, sodass $\forall\, i \geq 0 :\ \sigma_{i+1} \in \Delta.\sigma_i$ gilt.

Das Verhalten einer Kripkestruktur für einen Startzustand aus Σ_0 kann durch einen sogenannten Berechnungsbaum dargestellt werden. Die Knoten des Berechnungsbaums sind durch die Funktion m jeweils mit einem Zustand aus Σ markiert, wobei die Wurzel mit dem Startzustand markiert ist. Eine Kante (s, t) ist eine Kante des Baums, wenn $m(t) \in \Delta(m(s))$. Ein unendlicher Pfad durch den Baum, der in der Wurzel beginnt, repräsentiert einen möglichen Ablauf der Kripkestruktur. Die Eigenschaften, welche die Kripkestruktur erfüllen soll, können durch temporallogische Formeln spezifiziert werden. Eine temporallogische Formel charakterisiert ein Verhaltens-Muster der Kripkestruktur, wobei das Verhalten durch den Berechnungsbaum gegeben ist.

Die *Computation Tree Logic* (CTL) [34] ist eine Variante der Temporalen Logik und wird auf Grund ihres baumartigen Zeitmodells als *verzweigte* Logik bezeichnet. CTL ist in ihrer Ausdrucksmächtigkeit gegenüber anderen Spezifikationstechniken zwar eingeschränkt, jedoch findet sie, da effiziente Prüfalgorithmen zur Verfügung stehen, beim Verfahren Model-Checking Verwendung. Aussagen über Pfade können durch *Pfadquantoren* und *temporale Operatoren* getätigt werden: Die Operatoren X (*next*), F (*finally*), G (*globally*) und U (*until*) werden als temporale Operatoren bezeichnet; die Operatoren E (*exists*) und A (*always*) werden als Pfadquantoren bezeichnet. In CTL werden die Operatoren paarweise gebraucht, d. h. A oder E gefolgt von F, G, U oder X.

Im Folgenden wird die Syntax und die Semantik analog zu [34] definiert. Sei p eine atomare Aussage und seien f und g Formeln, dann gilt:

1. Jedes p, $f \wedge g$ und $\neg f$ ist eine Formel.

[4] Der Begriff „Temporale Logik“ geht auf Pnueli [70] zurück.

[5] Der deutsche Begriff ist „Modell-Prüfung“ – in dieser Arbeit wird jedoch durchgängig der englische Begriff verwendet.

2. $\mathsf{EX}\, f$ ist eine Formel, die bedeutet, dass es einen nachfolgenden Zustand gibt, der f erfüllt.

3. $\mathsf{A}\,[\, f\ \mathsf{U}\ g\,]$ ist eine Formel und bedeutet, dass für jeden Pfad gilt: es gibt einen Prefix, der an seiner Endposition g erfüllt und an allen anderen Positionen des Prefixes gilt f.

4. $\mathsf{E}\,[\, f\ \mathsf{U}\ g\,]$ ist eine Formel und bedeutet, dass es mindestens einen Pfad gibt, der einen Prefix enthält, der an seiner Endposition g erfüllt und an allen anderen Positionen gilt f.

Formal sind die beschriebenen Bedeutungen wie folgt festgelegt:

$$
\begin{array}{lcl}
K, \sigma_0 \models p & \Leftrightarrow & p \in L(\sigma_0) \\
K, \sigma_0 \models \neg p & \Leftrightarrow & K, \sigma_0 \not\models p \\
K, \sigma_0 \models f \wedge g & \Leftrightarrow & K, \sigma_0 \models f \text{ und } K, \sigma_0 \models g \\
K, \sigma_0 \models \mathsf{EX}\, f & \Leftrightarrow & \text{es gibt einen Zustand } \sigma_1 \text{ mit } \sigma_1 \in \Delta.\sigma_0 \text{ und } K, \sigma_1 \models f \\
K, \sigma_0 \models \mathsf{A}\,[\, f\ \mathsf{U}\ g\,] & \Leftrightarrow & \text{für alle Pfade } \langle \sigma_0, \sigma_1, \sigma_2, \ldots \rangle \text{ gilt} \\
 & & \exists\, i:\ 0 \leq i\ \wedge\ K, \sigma_i \models g\ \wedge\ \forall\, j:\ 0 \leq j < i\ \rightarrow\ K, \sigma_j \models f \\
K, \sigma_0 \models \mathsf{E}\,[\, f\ \mathsf{U}\ g\,] & \Leftrightarrow & \text{es gibt einen Pfad } \langle \sigma_0, \sigma_1, \sigma_2, \ldots \rangle \text{ , für den gilt} \\
 & & \exists\, i:\ 0 \leq i\ \wedge\ K, \sigma_i \models g\ \wedge\ \forall\, j:\ 0 \leq j < i\ \rightarrow\ K, \sigma_j \models f.
\end{array}
$$

Durch Kombination der definierten Operatoren lassen sich weitere Operatoren erstellen. Diese sind für die praktische Beschreibung von Eigenschaften hilfreich. Es gelten die folgenden Äquivalenzen:

$$
\begin{array}{rcl}
f \vee g & \stackrel{\text{def}}{=} & \neg(\neg f \wedge \neg g) \\
f \rightarrow g & \stackrel{\text{def}}{=} & \neg f \vee g \\
\mathsf{AF}\, g & \stackrel{\text{def}}{=} & \mathsf{A}\,[\,\mathsf{true}\ \mathsf{U}\ g\,] \\
\mathsf{EF}\, g & \stackrel{\text{def}}{=} & \mathsf{E}\,[\,\mathsf{true}\ \mathsf{U}\ g\,] \\
\mathsf{AG}\, g & \stackrel{\text{def}}{=} & \neg\, \mathsf{E}\,[\,\mathsf{true}\ \mathsf{U}\ \neg g\,] \\
\mathsf{EG}\, g & \stackrel{\text{def}}{=} & \neg\, \mathsf{A}\,[\,\mathsf{true}\ \mathsf{U}\ \neg g\,] \\
\mathsf{AX}\, f & \stackrel{\text{def}}{=} & \neg\, \mathsf{EX}\, \neg f \\
\mathsf{A}\,[\, f\ \mathsf{W}\ g\,] & \stackrel{\text{def}}{=} & \neg\, \mathsf{E}\,[\,\neg g\ \mathsf{U}\ (\neg f \wedge \neg g)\,].
\end{array}
$$

Diese Operatoren haben informell die folgenden Bedeutungen und sind zum Teil in Abb. 2.1 visualisiert:

- $\mathsf{AF}\, g$ bedeutet, dass für alle Pfade gilt: es gibt eine Position, die g erfüllt.
- $\mathsf{EF}\, g$ bedeutet, dass es einen Pfad gibt, in dem es eine Position gibt, die g erfüllt.
- $\mathsf{AG}\, g$ bedeutet, dass für alle Pfade gilt: jede Position erfüllt g.

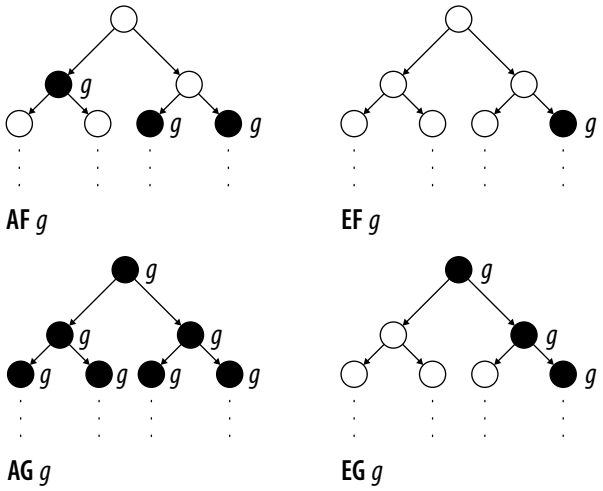


Abbildung 2.1: Visualisierung von $\mathsf{AF}\,g$, $\mathsf{EF}\,g$, $\mathsf{AG}\,g$ und $\mathsf{EG}\,g$ (Quelle: [47])

- $\mathsf{EG}\,g$ bedeutet, dass es einen Pfad gibt, in dem an jeder Position g erfüllt ist.
- $\mathsf{AX}\,g$ bedeutet, dass für alle Pfade gilt: der nachfolgende Zustand erfüllt g.
- $\mathsf{A}[\,f\ \mathsf{W}\ g\,]$ bedeutet, dass für alle Pfade gilt: solange g nicht erfüllt wird, ist f erfüllt.

Die temporale Logik CTL wird in Kapitel 4 verwendet, um dynamische Eigenschaften der Mensch-Maschine-Interaktion zu beschreiben.

2.2 Zustandsübergangsdiagramme

Das Verhalten eines Systems kann durch Zustandsübergangsdiagramme dargestellt werden. In der Grundform stellt ein Zustandsübergangsdiagramm einen gerichteten, markierten Graph dar. Die Knoten repräsentieren *Kontrollzustände* und die Kanten beschreiben Zustandsübergänge, die auch *Transitionen* genannt werden. Diese Übergänge sind mit *Vor-* und *Nachbedingungen* markiert. Die Vorbedingung beschreibt, ob die Transition ausgeführt werden darf, die Nachbedingung setzt die Belegungen der Variablen vor Ausführung des Übergangs mit den Belegungen der Variablen nach Ausführung des Übergangs in Relation.

In dieser Arbeit wird eine hierarchische Variante von Zustandsübergangsdiagrammen zur Beschreibung von Systemen eingesetzt, die im Folgenden mit DIA bezeichnet wird. Diese Notation basiert auf Konzepten von Harel [56] sowie Alur und Grosu [11], ist aber dahingehend eingeschränkt, eine direkte Abbildung auf NUSMV [33] zu ermöglichen. Abb. 2.2 zeigt die Elemente der Notation, die im Folgenden beschrieben sind.

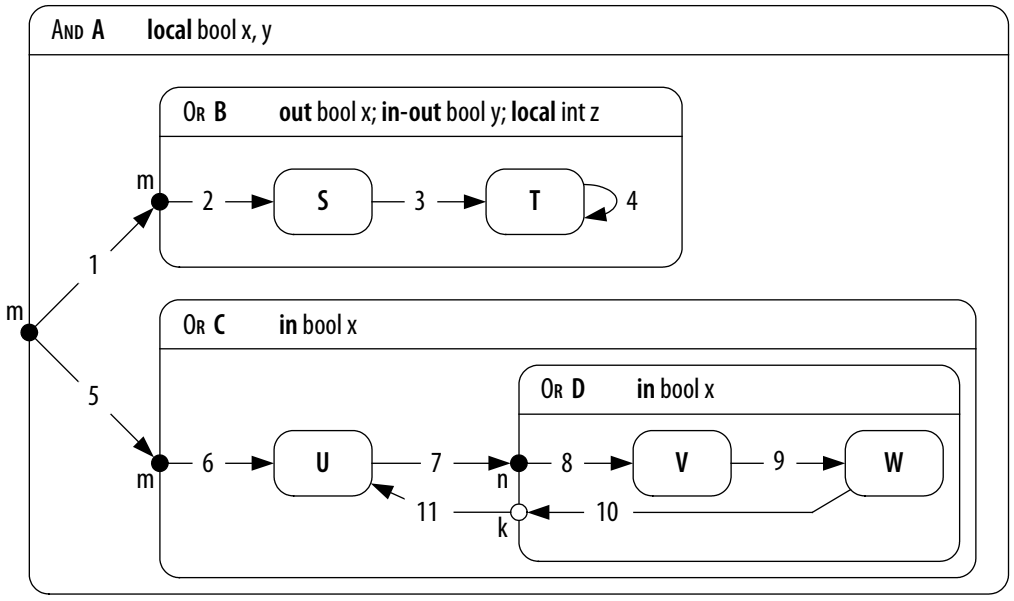


Abbildung 2.2: Elemente der Notation

Hierarchiekonzept. Die Knoten des Diagramms sind entweder Kontrollzustände oder *Stellen*. Die Menge der Stellen ist unterteilt in *Eintritts-* und *Austrittsstellen*. Eintrittsstellen sind durch schwarze Kreise dargestellt, Austrittsstellen durch weiße Kreise. In der Abbildung ist beispielsweise die Stelle „k“ eine Austrittsstelle und die Stelle „n“ eine Eintrittsstelle.

Ein Kontrollzustand kann entweder ein *einfacher* Zustand oder ein *zusammengesetzter* Zustand sein. Ein zusammengesetzter Zustand besitzt eine Menge an *Unterzuständen* und eine Menge an Stellen. Der übergeordnete Zustand der Unterzustände wird auch *Vater* genannt. Die Unterzustand-Beziehung zwischen den Zuständen bildet einen azyklischen Graphen. Ein zusammengesetzter Zustand kann ausschließlich über seine Stellen mit Knoten der Umgebung verbunden werden. In der Abbildung ist z. B. der Zustand „B“ ein zusammengesetzter Zustand und „S“ ein einfacher Zustand. Ein einfacher Kontrollzustand ist *aktiv*, wenn im vorherigen Takt eine Transition ausgeführt wurde, die in diesem Zustand endet; ein zusammengesetzter Zustand ist aktiv, wenn einer seiner Unterzustände aktiv ist.

Ein zusammengesetzter Zustand kann entweder ein Oder-Zustand oder ein Und-Zustand sein. Oder-Zustände sind zusammengesetzte Zustände, dessen Unterzustände abwechselnd aktiv sind. Und-Zustände sind zusammengesetzte Zustände, dessen Unterzustände simultan aktiv sind. Die Unterzustände eines Und-Zustands werden *orthogonale Zustände* genannt. Für jede Stelle eines Und-Zustands muss in jedem Unterzustand eine Stelle mit demselben Bezeichner vorhanden sein.

Transitionen. Die Knoten sind über gerichteten Transitionen miteinander verbunden. Der Knoten, von dem die Transition ausgeht, wird *Startknoten* genannt; der Knoten, in dem die Transition endet, wird *Endknoten* genannt. Mit Hilfe der Stellen

können die Transitionen klassifiziert werden: Eine Transition ist eine *Eintrittstransition*, wenn sie als Startknoten eine Eintrittsstelle besitzt; eine Transition ist eine *Austrittstransition*, wenn sie als Endknoten eine Austrittsstelle besitzt; eine Transition ist eine *lokale Transition*, wenn sie zwei Kontrollzustände miteinander verbindet. Zum Beispiel sind in Abb. 2.2 die Transitionen „1“, „2“ oder „8“ Eintrittstransitionen, die Transitionen „3“ oder „7“ sind lokale Transitionen und die Transition „10“ ist eine Austrittstransition.

Eine *zusammengesetzte Transition* verbindet zwei einfache Zustände miteinander. Die Transitionen einer zusammengesetzten Transition werden als *Segmente* bezeichnet. Eine zusammengesetzte Transition ist ein Pfad, der von einem einfachen Zustand ausgeht und dessen letztes Segment einen einfachen Zustand als Endknoten besitzt.

Variablen. Die Menge der Variablen ist durch die Hierarchie der Zustände strukturiert. Ein zusammengesetzter Zustand deklariert eine Menge an Variablen und stellt einen Gültigkeitsbereich für diese Menge an Variablen dar. Innerhalb eines zusammengesetzten Zustands kann nur auf die deklarierten Variablen zugegriffen werden. Eine Variable ist eine *lokale* Variable (`local`), wenn sie nicht bereits in dem übergeordneten Zustand deklariert worden ist. Eine lokale Variable ist außerhalb des Zustands nicht sichtbar. Ein Unterzustand, der auf die Variablen seines übergeordneten Zustands zugreifen möchte, muss diese erneut deklarieren. Bei der Deaktivierung eines zusammengesetzten Zustands verliert eine lokale Variable ihren Wert.[6]

- Eine Variable ist eine *Eingabevariable* (`in`) eines zusammengesetzten Zustands, wenn sie in seinen Transitionen gelesen werden kann.
- Eine Variable ist eine *Ausgabevariable* (`out`) eines zusammengesetzten Zustands, wenn sie in seinen Transitionen beschrieben werden kann. Eine Ausgabevariable ist eine *gemeinsame* Variable (`shared`) eines orthogonalen Zustands, wenn sie auch von anderen orthogonalen Zuständen beschrieben werden kann.

Eine lokale Variable ist sowohl eine Eingabevariable als auch eine Ausgabevariable.

Vor- und Nachbedingungen. Eine Transition legt eine Vorbedingung und eine Nachbedingung fest. Die Vorbedingung wird durch eine logische Formel über den Eingabevariablen spezifiziert. Die Nachbedingung setzt die Belegungen der Eingabevariable vor Ausführung des Übergangs mit den Belegungen der Ausgabevariable nach Ausführung des Übergangs in Relation. Dazu wird eine Kopie der Menge der Ausgabevariablen benutzt, deren Elemente als *gestrichene Variablen* bezeichnet werden. Die gestrichene Variante einer Variablen bezieht sich auf den Wert der Variablen nach der Ausführung der Transition.

Die Vorbedingung einer zusammengesetzten Transition ergibt sich durch die Konjunktion der Vorbedingungen aller Segmente. Die Nachbedingung einer zusammengesetzten Transition ergibt sich durch die Konjunktion der Nachbedingungen der Segmente.

[6] Aus diesem Grund kann man die Variablen diese Ansatzes als *volatile* bezeichnen.

In der Abbildung ist z. B. das Segment „3“ eine zusammengesetzte Transition (mit nur einem Segment). Die Segmente „10“ und „11“ sind ein weiteres Beispiel für eine zusammengesetzte Transition.

Zusätzlich zur Einschränkung, dass nur auf deklarierte Variablen zugegriffen werden darf, gelten weitere Einschränkungen, die im Folgenden beschrieben sind:

1. Die Menge der Eingabevariablen, die in einer zusammengesetzten Transition verwendet werden darf, wird durch die einzelnen Segmente bestimmt: Sie enthält alle Eingabevariablen, die die Väter der lokalen Transitionen deklarieren.

2. Die Menge der Ausgabevariablen, die in einer zusammengesetzten Transition verwendet werden darf, wird ebenfalls durch die einzelnen Segmente bestimmt: Sie enthält alle Ausgabevariablen, die die Väter der lokalen Transitionen oder der Eintrittstransitionen deklarieren.

3. Eine zusammengesetzte Transition ist verpflichtet, eine Belegung für die Menge der gültigen Ausgabevariablen zu definieren. Wenn das nicht explizit durch die Nachbedingung geschieht, wird für diese Variable ein *voreingestellter Wert* wie folgt festgelegt: Einer Variablen, die nur als Ausgabevariable deklariert ist, wird der initiale Wert ihres Typs zugewiesen; einer Variablen, die sowohl als Eingabevariable als auch als Ausgabevariable deklariert ist, wird der Wert der Eingabevariable zugewiesen; einer gemeinsamen Variablen wird kein Wert zugewiesen, sondern die Belegung offen gelassen. Es ist zu bemerken, dass Belegungen für gemeinsame Variablen immer explizit festgelegt werden müssen.

4. Für Und-Zustände ist die Besonderheit zu beachten, dass alle Transitionen weder eine Vor- noch eine Nachbedingung besitzen.

Diese Regeln sorgen beispielsweise dafür, dass Eintrittstransitionen lokale Variablen initialisieren und dass Werte für lokale Variablen in Austrittstransitionen nicht festgelegt werden.

Idle-Transition. Für jeden einfachen Zustand wird implizit eine Schleife, d. h. ein Übergang der den Zustand mit sich selbst verbindet, hinzugefügt, die sogenannte Idle-Transition. Die Vorbedingung dieser Transition ist die negierte Konjunktion aller ausgehenden zusammengesetzten Transitionen. Die Nachbedingung dieser Transition belegt alle Ausgabevariablen, die ihr Vater deklariert, mit dem voreingestellten Wert.

Quasiparallele orthogonale Zustände. Die Unterzustände eines Und-Zustands führen ihre Übergänge parallel aus. Es ist aber in einigen Fällen erforderlich, bestimmte Aktionen, beispielsweise das Schreiben einer gemeinsamen Variable, wechselseitig auszuschließen[7].

[7] engl.: *mutual exclusion* abgekürzt mit *mutex*.

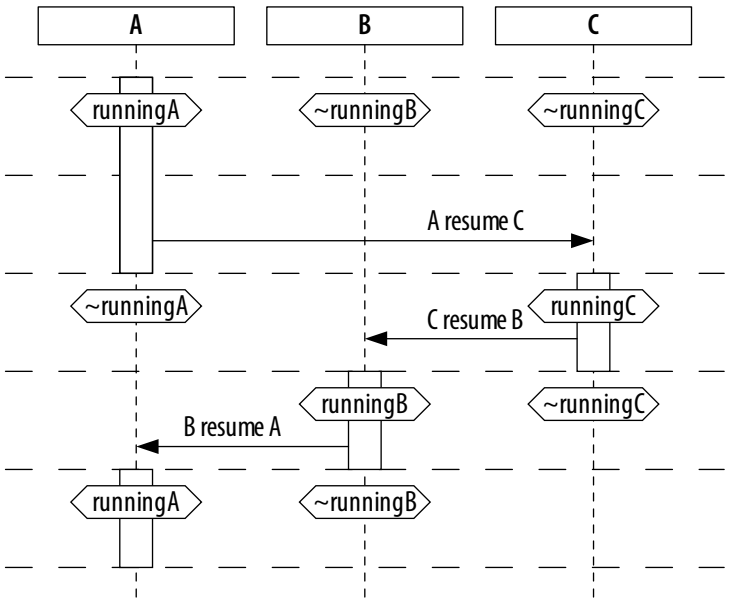


Abbildung 2.3: Muster *quasiparallele orthogonale Zustände*

Deswegen ist das Muster „quasiparallele orthogonale Zustände“ in der Notation realisiert. Ein beispielhafter Ablauf des Musters ist in Abb. 2.3 dargestellt. Jeder orthogonale Zustand des UND-Zustands, der quasiparallel ablaufen soll, erhält eine lokale boolesche Variable `running`. In allen seiner Transitionen und in den Transitionen seiner Unterzustände wird die Vorbedingung mit der Zusicherung, dass `running` gilt, erweitert. Das hat zur Folge, dass wenn `running` nicht gilt, die Idle-Schleife ausgeführt wird. Auf Grund der Tatsache, dass gemeinsame Variablen durch die Idle-Schleife nicht festgelegt werden, können diese in diese in den orthogonalen Zuständen des Musters verwendet werden.

Ein orthogonaler Zustand, dessen Variable `running` nicht gilt, wird als *unterbrochen* bezeichnet. Initial sind alle Zustände, bis auf einen, der als Startzustand festgelegt ist, unterbrochen. Für jeden orthogonalen Zustand X wird festgelegt, welche weiteren orthogonale Zustände er fortsetzen möchte. Jeder dieser Zustände deklariert dann eine boolesche Eingangs-Variable `resume_`X.

Diese Prinzip des expliziten Kontrollwechsels ist auch unter dem Begriff *Coroutine*[8] bekannt. Die orthogonalen Zustände eines UND-Zustands besitzen abwechselnd die sogenannte Kontrolle. Ein orthogonaler Zustand, der die Kontrolle nicht besitzt, verweilt in einem Wartezustand, bis er wieder fortgesetzt wird. Die Aktion, mit der der Zustand, der die Kontrolle besitzt, die Kontrolle an einen anderen Zustand übergibt, wird mit `resume (`X`)` bezeichnet. Das „X“ steht für den Zustand, der die Kontrolle erhält.

[8] Der Begriff „Coroutine“ wurde in [36] durch Conway eingeführt. Er definiert Coroutinen wie folgt: „coroutines are subroutines all at the same level, each acting as if it were the master programm when in fact there is no master program.“

2.3 Zustandsbasiertes Systemmodell

Dieser Abschnitt stellt ein zustandsbasiertes Systemmodell vor, das im Folgenden mit FCT bezeichnet wird. Dieses Systemmodell ist angelehnt an die Arbeiten von Henzinger [58] und Schätz [93, 94]. Die von Henzinger [58] vorgeschlagene explizite Trennung von Kontroll- und Datenfluss ermöglicht eine modulare, hierarchische Strukturierung der Aktionen. Die von Schätz [93, 94] eingeführten *Funktionen* ermöglichen eine dienstorientierte Sicht auf das System und erlauben das Beschreiben von partiellen Verhalten. Diese Aspekte werden aufgegriffen und durch Automaten [22, 52] beschrieben. Die Ausprägung der Automaten wurden so gewählt, dass sie sich einfach auf die Eingabesprache des Modelcheckers NUSMV [33] abbilden lassen.

2.3.1 Funktionen

Ein System besteht aus Funktionen. Eine Funktion kann entweder aus Unterfunktionen zusammengesetzt sein oder eine sogenannte *Basisfunktion* darstellen. Funktionen werden durch die Verwendung von vier Operatoren aus Unterfunktionen zusammengesetzt. Diese sind: *parallele Komposition*, *alternative Komposition*, *Verbergen* und *Feedback*. Die Menge aller Funktionen wird mit $\mathbb{F}$ bezeichnet. Der Baum, der sich induktiv aus den Basisfunktionen und den Operatoren zusammensetzt, ergibt die *Funktionshierarchie*.

Eine Funktion interagiert mit ihrer Umgebung über Variablen. Die Menge der Variablen ist unterteilt in Daten- und in Kontrollvariablen. Die Kontroll-Variablen werden im Folgenden *Locations* genannt. Die Menge aller Datenvariablen wird mit $\mathbb{D}$ bezeichnet; die Menge aller Locations mit $\mathbb{L}$ und es gilt $\forall\, l \in \mathbb{L} : \mathit{type}(l) = \mathbb{B}$ und $\mathbb{D} \cap \mathbb{L} = \emptyset$.

Über die Locations betritt oder verlässt die Kontrolle eine Funktion. Befindet sich die Kontrolle innerhalb einer Funktion, so wird diese Funktion als *aktiviert* bezeichnet. Eine Funktion ohne Kontrolle wird als *deaktiviert* bezeichnet. Das Verhalten einer Funktion in einem Schritt ist dadurch bestimmt, welche ihrer Unterfunktionen aktiviert oder deaktiviert sind. Dies wird auch als die *Konfiguration* einer Funktion bezeichnet.

Im Folgenden sind die Bestandteile einer Funktion angegeben. Die Menge der Variablen einer Funktion ist wie folgt unterteilt:

- Die Menge der Eingabevariablen $V_F^{in} \subseteq \wp(\mathbb{D})$. Die Werte dieser Variablen kann die Funktion lesen.
- Die Menge der Ausgabevariablen $V_F^{out} \subseteq \wp(\mathbb{D})$. Die Werte dieser Variablen kann die Funktion für den nächsten Zustand setzen.
- Die Menge der Eintritts-Locations $V_F^{enter} \subseteq \wp(\mathbb{L})$. Über diese Locations gelangt die Kontrolle in die Funktion.

- Die Menge der Austritts-Locations $V_F^{exit} \subseteq \wp(\mathbb{L})$. Über diese Locations kann die Funktion die Kontrolle zurück an die Umgebung geben.
- Die Menge der lokalen Variablen $V_F^{priv} \subseteq \wp(\mathbb{D} \cup \mathbb{L})$. Diese Variablen werden von der Funktion kontrolliert und sind für die Umgebung nicht sichtbar.

Die abgeleiteten Bestandteile sind:

- $V_F^{ctr} \stackrel{\text{def}}{=} V_F^{priv,\,out,\,exit}$. Die von F kontrollierten Variablen, d. h., die Variablen, deren Belegung von der Funktion bestimmt werden kann.[9]
- $V_F^{intf} \stackrel{\text{def}}{=} V_F^{in,\,enter,\,out,\,exit}$. Die Schnittstelle der Funktion F, d. h., die Variablen, die für die Umgebung sichtbar sind.

Sei s eine Belegung. Das Prädikat *ctr* zeigt, ob eine Location der Belegung die Kontrolle besitzt.

$$ctr(s) \;\equiv\; \exists\, l \in vars(s) \cap \mathbb{L} :\; s(l) = \mathsf{true}.$$

Die Funktion

$$[\![\_]\!] : \mathbb{F} \to (\Sigma \to \wp(\Sigma))$$

bildet eine Funktion auf eine Übergangsfunktion ab. Für eine Funktion $F \in \mathbb{F}$ hat die Übergangsfunktion $[\![F]\!]$ die Form:

$$[\![F]\!] : [V_F^{priv,\,in,\,enter}] \to \wp([V_F^{ctr}]).$$

2.3.2 Basisfunktion

Die Blätter der Funktionshierarchie sind die Basisfunktionen. Eine Basisfunktion

$$(I,\, E,\, O,\, x,\, P,\, Q)$$

ist durch

- eine Menge von Eingabevariablen $I \subseteq \wp(\mathbb{D})$,
- eine Menge von Eintritts-Locations $E \subseteq \wp(\mathbb{L})$,
- eine Menge von Ausgabevariablen $O \subseteq \wp(\mathbb{V})$,
- eine Austritts-Location $x \in \mathbb{L}$,
- eine Zusicherung $P : [I] \to \mathbb{B}$ über den Belegungen der Eingabevariablen und
- eine Zusicherung $Q : [I] \times [O] \to \mathbb{B}$ über den Belegungen der Eingabevariablen und den Belegungen der Ausgabevariablen im Nachfolgezustand

gegeben. Jede Basisfunktion ist Element von $\mathbb{F}$ und für eine Basisfunktion A sind die Bestandteile wie folgt definiert:

- $V_A^{in} = I$, $V_A^{out} = O$, $V_A^{enter} = E$ und $V_A^{exit} = \{x\}$.

[9] Es wird die folgende abkürzende Schreibweise verwendet: $V^{priv,\,out,\,exit} \stackrel{\text{def}}{=} V^{priv} \cup V^{out} \cup V^{exit}$.

- Eine Basisfunktion besitzt keine lokalen Variablen; d. h., $V_A^{priv} = \emptyset$.

Eine Basisfunktion beschreibt eine *Aktion* eines Schritts des Systemablaufs. Sei $A = (I, E, O, x, P, Q)$ und $\sigma \in [I \cup E]$, dann ist

$$[\![(I, E, O, x, P, Q)]\!].\sigma = \{\, \sigma' \in [O \cup \{x\}] \mid (P(\sigma|_I) \wedge ctr(\sigma|_E) \wedge Q(\sigma|_I, \sigma'|_O) \wedge \sigma'(x) = \mathsf{true}) \,\}.$$

Daraus folgt, dass eine Basisfunktion aktiviert werden kann, wenn die Kontrolle sich in σ befindet und die Vorbedingung P für die Eingabewerte gilt. Die Aktion Q beschreibt eine Relation zwischen den Werten der Eingabenachrichten im Zustand σ und den Werten der Ausgabevariablen im Zustand σ'.

2.3.3 Parallele Komposition

Zwei Funktionen können parallel zusammengesetzt werden, wenn sie dieselben Ein- und Austritts-Locations besitzen. Seien F_1 und F_2 Funktionen aus der Menge $\mathbb{F}$. Wenn $V_{F_1}^{enter} = V_{F_2}^{enter}$ und $V_{F_1}^{exit} = V_{F_2}^{exit}$ gilt, dann ist $F_1||F_2$ Element von $\mathbb{F}$.

Die Bestandteile der zusammengesetzten Funktion ergeben sich wie folgt aus den Unterfunktionen:

- Eine Variable ist eine Eingabevariable von $F_1||F_2$, wenn sie eine Eingabevariable von F_1 oder eine Eingabevariable von F_2 ist; d. h., $V_{F_1||F_2}^{in} = V_{F_1}^{in} \cup V_{F_2}^{in}$.
- Eine Variable ist eine Ausgabevariable von $F_1||F_2$, wenn sie eine Ausgabevariable von F_1 oder eine Ausgabevariable von F_2 ist; d. h., $V_{F_1||F_2}^{out} = V_{F_1}^{out} \cup V_{F_2}^{out}$.
- Jede Eintritts-Location von $F_1||F_2$ ist sowohl Eintritts-Location von F_1 als auch Eintritts-Location von F_2; d. h., $V_{F_1||F_2}^{enter} = V_{F_1}^{enter} = V_{F_2}^{enter}$.
- Jede Austritts-Location von $F_1||F_2$ ist sowohl Austritts-Location von F_1 als auch Austritts-Location von F_2; d. h., $V_{F_1||F_2}^{exit} = V_{F_1}^{exit} = V_{F_2}^{exit}$.
- Eine Variable ist eine lokale Variable von $F_1||F_2$, wenn sie eine lokale Variable von F_1 oder eine lokale Variable von F_2 ist; d. h., $V_{F_1||F_2}^{priv} = V_{F_1}^{priv} \cup V_{F_2}^{priv}$.

Sei $F_1||F_2 \in \mathbb{F}$ und $\sigma \in V_{F_1||F_2}^{priv,\,in,\,enter}$, dann ist

$$\begin{aligned}[\![F_1||F_2]\!].\sigma = \{\, \sigma' \in [V_{F_1||F_2}^{ctr}] \mid \\ (\sigma'|V_{F_1}^{ctr} \in [\![F_1]\!].\sigma|V_{F_1}^{priv,\,in,\,enter}) \\ \wedge\, (\sigma'|V_{F_2}^{ctr} \in [\![F_2]\!].\sigma|V_{F_2}^{priv,\,in,\,enter}) \,\}.\end{aligned}$$

Die parallele Komposition erlaubt das Beschreiben von nebenläufigen Aktionen. Es besteht die Einschränkung, dass sowohl die Kontrollübergabe an die Unterfunktionen als auch die Kontrollabgabe gleichzeitig ablaufen muss. Das heißt, die Funktion wird nur dann aktiv, wenn beide Unterfunktionen aktiviert werden können und die

Kontrolle wird nur dann an die Umgebung zurückgegeben, wenn sie von beiden Unterfunktionen gleichzeitig über dieselbe Austritts-Location abgegeben wird. Die parallele Komposition ist assoziativ und kommutativ.

2.3.4 Alternative Komposition

Sind F_1 und F_2 Funktionen, dann ist auch $F_1 + F_2$ eine Funktion. Die Bestandteile der zusammengesetzten Funktion ergeben sich wie folgt aus den Unterfunktionen:

- Eine Variable ist eine Eingabevariable von F_1+F_2, wenn sie eine Eingabevariable von F_1 oder eine Eingabevariable von F_2 ist; d. h., $V^{in}_{F_1+F_2} = V^{in}_{F_1} \cup V^{in}_{F_2}$.
- Eine Variable ist eine Ausgabevariable von $F_1 + F_2$, wenn sie eine Ausgabevariable von F_1 oder eine Ausgabevariable von F_2 ist; d. h., $V^{out}_{F_1+F_2} = V^{out}_{F_1} \cup V^{out}_{F_2}$.
- Eine Location ist ein Eingang von $F_1 + F_2$, wenn sie ein Eingang von F_1 oder ein Eingabe von F_2 ist; d. h., $V^{enter}_{F_1+F_2} = V^{enter}_{F_1} \cup V^{enter}_{F_2}$.
- Eine Location ist ein Ausgang von $F_1 + F_2$, wenn sie ein Ausgang von F_1 oder ein Ausgabe von F_2 ist; d. h., $V^{exit}_{F_1+F_2} = V^{exit}_{F_1} \cup V^{exit}_{F_2}$.
- Eine Variable ist eine lokale Variable von $F_1 + F_2$, wenn sie eine lokale Variable von F_1 oder eine lokale Variable von F_2 ist; d. h., $V^{priv}_{F_1+F_2} = V^{priv}_{F_1} \cup V^{priv}_{F_2}$.

Das Verhalten ist durch eine Abbildung auf einen Automaten gegeben. Sei $F_1+F_2 \in \mathbb{F}$ und $\sigma \in V^{priv,\,in,\,enter}_{F_1+F_2}$, dann ist

$$\begin{aligned}[\![F_1 + F_2]\!].\sigma = \{\, \sigma' \in [V^{ctr}_{F_1+F_2}] \mid \\ (\sigma'|V^{ctr}_{F_1} \in [\![F_1]\!].\sigma|V^{priv,\,in,\,enter}_{F_1} \wedge \neg ctr(\sigma'|(V^{ctr}_{F_1+F_2} \backslash V^{ctr}_{F_1}))) \\ \vee\, (\sigma'|V^{ctr}_{F_2} \in [\![F_2]\!].\sigma|V^{priv,\,in,\,enter}_{F_2} \wedge \neg ctr(\sigma'|(V^{ctr}_{F_1+F_2} \backslash V^{ctr}_{F_2}))) \,\}.\end{aligned}$$

Ein Zustand ist ein möglicher Folgezustand, wenn er entweder ein möglicher Folgezustand von F_1 oder ein möglicher Folgezustand von F_2 ist. Kontrollierte Variablen, die nicht durch die aktive Unterfunktion festgelegt sind, können beliebige Werte annehmen, mit der Einschränkung, dass sich die Kontrolle nur in der aktiven Funktion befinden darf. Die alternative Komposition der Unterfunktionen F_1 und F_2 beschreibt eine Funktion, die sich entweder wie F_1 oder wie F_2 verhält. Man kann sie verwenden, um sequentielle Abläufe von Funktionen zu spezifizieren. Die alternative Komposition ist assoziativ, kommutativ und idempotent.

2.3.5 Verbergen

Eine Variable kann in einer Funktion verborgen werden, wenn sie eine Ausgabevariable oder eine Austritts-Location von dieser ist. Wenn F' eine Funktion ist und $h \in V^{out,\,exit}_{F'}$ gilt, dann ist $F'\backslash h$ eine Funktion. Die Bestandteile von $F'\backslash h$ ergeben sich wie folgt:

- Eine Variable ist eine Eingabevariable von $F'\backslash h$, wenn sie eine Eingabevariable von F' ist; d. h., $V^{in}_{F'\backslash h} = V^{in}_{F'}$.
- Eine Variable ist eine Ausgabevariable von $F'\backslash h$, wenn sie ungleich h und eine Ausgabevariable von F' ist; d. h., $V^{out}_{F'\backslash h} = V^{out}_{F'}\backslash\{h\}$.
- Eine Location ist ein Eingang von $F'\backslash h$, wenn sie ein Eingang von F' ist. d. h., $V^{enter}_{F'\backslash h} = V^{enter}_{F'}$.
- Eine Location ist ein Ausgang von $F'\backslash h$, wenn sie ungleich h und ein Ausgang von F' ist. d. h., $V^{exit}_{F'\backslash h} = V^{exit}_{F'}\backslash\{h\}$.
- Eine Location ist eine lokale Variable von $F'\backslash h$, wenn sie gleich h oder eine lokale Variable von F' ist; d. h., $V^{priv}_{F'\backslash h} = V^{priv}_{F'} \cup \{h\}$.

Die Funktion $F'\backslash h$ verhält sich so wie F', an der Schnittstelle ist aber h nicht mehr sichtbar. Sei $F'\backslash h \in \mathbb{F}$ und $\sigma \in V^{priv,\,in,\,enter}_{F'\backslash h}$, dann ist

$$[\![F'\backslash h]\!].\sigma = [\![F']\!].\sigma.$$

2.3.6 Feedback

Eine Variable kann in einer Funktion rückgekoppelt werden, wenn sie sowohl eine Ausgabevariable und eine Eingabevariable von F' ist. Eine Location, die sowohl Ein- als auch Ausgang von F' ist, kann ebenfalls zurückgekoppelt werden. Sei F' eine Funktion. Wenn $c \in V^{in}_{F'} \cap V^{out}_{F'} \vee c \in V^{enter}_{F'} \cap V^{exit}_{F'}$ gilt, dann ist

$$\mu_c\, F'$$

ein Element von $\mathbb{F}$. Die Bestandteile von $F'\backslash h$ ergeben sich wie folgt:

- Eine Variable ist eine Eingabevariable von $\mu_c\, F'$, wenn sie ungleich c und eine Eingabevariable von F' ist; d. h., $V^{in}_{\mu_c\, F'} = V^{in}_{F'}\backslash\{c\}$.
- Eine Variable ist eine Ausgabevariable von $\mu_c\, F'$, wenn sie ungleich c und eine Ausgabevariable von F' ist; d. h., $V^{out}_{\mu_c\, F'} = V^{out}_{F'}\backslash\{c\}$.
- Eine Location ist ein Eingang von $\mu_c\, F'$, wenn sie ungleich c und ein Eingang von F' ist. d. h., $V^{enter}_{\mu_c\, F'} = V^{enter}_{F'}\backslash\{c\}$.
- Eine Location ist ein Ausgang von $\mu_c\, F'$, wenn sie ungleich c und ein Ausgang von F' ist. d. h., $V^{exit}_{\mu_c\, F'} = V^{exit}_{F'}\backslash\{c\}$.
- Eine Location ist eine lokale Variable von FED, wenn sie gleich c oder eine lokale Variable von F' ist; d. h., $V^{priv}_{\mu_c\, F'} = V^{priv}_{F'} \cup \{c\}$.

Die Funktion $\mu_c\, F'$ verhält sich so wie F', an der Schnittstelle ist aber c nicht mehr sichtbar. Sei $\mu_c\, F' \in \mathbb{F}$ und $\sigma \in V^{priv,\,in,\,enter}_{\mu_c\, F'}$, dann ist

$$[\![\mu_c\, F']\!].\sigma = [\![F']\!].\sigma.$$

2.4 Abbildung der Zustandsübergangsdiagramme

Dieser Abschnitt beschreibt wie die Konzepte der Zustandsübergangsdiagramme auf die Konzepte des Systemmodells abgebildet werden. Eine Abbildung der logischen Formeln ist nicht angegeben, da die Ausdrücke direkt übernommen werden. Im Zentrum steht die Überführung der Zustände.

Es wird angenommen, dass jeder einfache Zustand einer eindeutigen Location zugeordnet ist. Um den Wirkungsbereich eines einfachen Zustands S zu bestimmen, werden alle diejenigen einfachen Zustände, die durch eine zusammengesetzte Transition mit dem Zustand S verbunden sind – sowie S selbst – zu einer Menge B zusammengefasst. Ein zusammengesetzter Zustand wird „LCA für S“[10] genannt, wenn er sich am tiefsten in der Hierarchie befindet und zu allen Zuständen der Menge B übergeordnet ist.

Im Folgenden ist die Abbildung für die Zustände eines Diagramms angegeben, die für einen Zustand S der Notation DIA eine Funktion der Sprache FCT zurück gibt:

- Ist S ein ODER-Zustand, wird für alle seine Unterzustände eine Funktion anhand der Abbildung erstellt. Diese Funktionen werden disjunktiv verknüpft. Alle lokalen Variablen von S werden zurückgeführt und versteckt. Ist S ein LCA für einen einfachen Zustand, dann wird diejenige Location, die diesem einfachen Zustand zugeordnet ist, zurückgeführt und versteckt.

- Ist S ein UND-Zustand, wird für alle seine Unterzustände eine Funktion anhand der Abbildung erstellt. Diese Funktionen werden konjunktiv verknüpft. Alle lokalen Variablen von S werden zurückgeführt und versteckt. Ist S ein LCA für einen einfachen Zustand, dann wird diejenige Location, die diesem einfachen Zustand zugeordnet ist, zurückgeführt und versteckt.

- Ist S ein einfacher Zustand, wird für alle eingehenden zusammengesetzten Transitionen jeweils eine Basisfunktion erstellt, mit den folgenden Bestandteilen: Die Menge der Eingabevariablen ist die Menge der gültigen Eingabevariablen der zusammengesetzten Transition (für die Definition dieser Menge s. Abs. 2.2). Die Menge der Ausgabevariablen ist der Menge der gültigen Ausgabevariablen der zusammengesetzten Transition (s. Abs. 2.2). Die Eintritts-Location ist die Location, die dem Startknoten der zusammengesetzten Transition zugeordnet ist. Die Austritts-Location ist die Location, die S zugeordnet ist. Die Vorbedingung ergibt sich aus der Vorbedingung der zusammengesetzten Transition. Die Nachbedingung ergibt sich aus der Nachbedingung der zusammengesetzten Transition. Die Menge der Basisfunktionen wird disjunktiv verknüpft.

[10] LCA = *Lowest Common Ancestor* (Der niedrigste gemeinsame Vorfahre).

2.5 Abbildung des Systemmodells

Dieser Abschnitt beschreibt wie das Systemmodell auf die Eingabesprache NuSmv abgebildet wird. In dieser Arbeit wird der Model-Checker NuSmv[11] [33] eingesetzt. NuSmv ist eine Reimplementierung des symbolischen Model-Checkers Smv [73]. NuSmv liest Systembeschreibungen, die in der Eingabesprache Smv verfasst sind, und ermöglicht u. a. die automatische Überprüfung von Eigenschaften, festgelegt durch Ctl-Formeln. Die Modelle können simuliert werden und es wird für eine Spezifikation, die nicht erfüllt werden kann, ein Gegenbeispiel erzeugt.

Eingabesprache. Die Eingabesprache des Model-Checkers NuSmv erlaubt die Spezifikation der Übergangsrelationen zustandsbasierter Modelle anhand logischer Formeln. Somit bietet sie die nötige Flexibilität, um das in dieser Arbeit entwickelte Systemmodell auf sie abzubilden. Allerdings ist es durch Angabe inkonsistenter Formeln möglich, Kripkestrukturen zu beschreiben, die nicht total sind. Deshalb ist es sinnvoll, die Kripkestruktur vor der Verifikation auf Totalität zu prüfen, eine Aufgabe, die vom Model-Checker übernommen wird.

Die Übergangsrelation wird durch eine Formel, welche die Belegung der Variablen vor Ausführung des Übergangs mit den Belegungen der Variablen nach Ausführung in Beziehung setzt, beschrieben. Auf die Belegung der Variablen nach der Ausführung wird durch den `next`-Ausdruck Bezug genommen. Wird die Belegung einer Variable für die unmittelbaren Nachfolgezustände nicht durch einen solchen Ausdruck festgelegt, nimmt die Variable einen beliebigen Wert ihres Typs an. Als Beispiel wird die folgende Beschreibung betrachtet:

```
MODULE main
VAR
    button : boolean;
    state : {on, off};

INIT
    !button & state = off;

TRANS
    button & state = off & next(state) = on
    | button & state = on & next(state) = off
    | !button & next(state) = state;

SPEC
    AG (button & state = off  ->  X (state = on));
```

[11] in der Version 2.4.3

Die Beschreibung deklariert im Abschnitt `VAR` die zwei Variablen `button` und `state`. Die Variable `button` ist vom Typ Boolean. Der Typ der Variable `state` ist ein Aufzählungstyp, der aus den Werten „on“ und „off“ besteht.

Der Abschnitt `INIT` legt fest, dass die initiale Belegung der Variable `button` „false“ und die initiale Belegung der Variable `state` „off“ ist.

Die Übergangsrelation wird im Abschnitt `TRANS` beschrieben und spezifiziert das folgende Verhalten: Wenn `button` mit „true“ belegt ist und `state` mit „off“, dann ist im Nachfolgezustand `state` mit „on“ belegt. Wenn `button` mit „true“ belegt ist und state mit „on“, dann ist im Nachfolgezustand `state` mit „off“ belegt. Wenn `button` nicht erfüllt ist ändert sich die Belegung von `state` nicht.

Da die Belegung für die Variable `button` im nächsten Zustand nicht festgelegt ist, kann sie entweder „true“ oder „false“ werden. Somit lässt sich nicht-deterministisches Verhalten beschreiben.

Abbildung. Im Folgenden ist die Abbildung des Systemmodells auf die Eingabesprache SMV angegeben. Es wird davon ausgegangen, dass die folgenden Abbildungen gegeben sind:

- Für eine Variable v bezeichnet $init(v)$ einen SMV-Ausdruck, der die Variable entsprechend ihres Typs auf einen initialen Wert setzt.
- Für ein Prädikat P bezeichnet $t(P)$ den entsprechenden SMV-Ausdruck.

Für diese Abbildungen kommt es darauf an, welche Datentypen und Operatoren man in der Sprache einsetzt. Die Ausdrücke, die in dieser Arbeit verwendet werden, lassen sich direkt in den entsprechenden SMV-Ausdruck umformen. Eine Abbildung für erweiterter Datentypen und deren Operationen ist in [90] zu finden.

Es wird ein Modul angelegt in dem die Variablen entsprechend deklariert werden. Die Variablen werden in dem `INIT`-Block initialisiert. Die initialen Werte der Datenvariablen haben keine Auswirkung auf das Verhalten des Modells. Die tatsächliche Initialisierung erfolgt in dem ersten Schritt des Ablaufs. Das Verhalten wird durch eine logische Formel in dem `TRANS`-Block angegeben.

$$
\begin{array}{ll}
\texttt{MODULE}\ \mathit{Spec} & \\
\texttt{VAR} & \texttt{linit} : \texttt{boolean}; \\
 & \mathit{var}_0 : \mathit{type}_0\,; \\
 & \mathit{var}_1 : \mathit{type}_1\,; \\
 & \quad\vdots \\
 & \mathit{var}_n : \mathit{type}_n\,; \\
\texttt{INIT} & (\bigwedge_{0 \le i \le n} \mathit{var}_i = \mathit{init}(\mathit{var}_i)) \wedge \texttt{linit} = \texttt{TRUE}; \\
\texttt{TRANS} & t(\mathit{root}) \wedge \texttt{next}(\texttt{linit}) = \texttt{FALSE};\ ,
\end{array}
$$

wobei *root* die Wurzel der Funktionshierarchie darstellt und die Abbildung t wie folgt definiert ist:

$$
\begin{aligned}
t(F_1 + F_2) &= (\ t(F_1)\ \wedge\ (\ \bigwedge_{loc \in L_1} \texttt{next}(loc) = \texttt{FALSE}\ )\ ) \\
&\quad \vee (\ t(F_2)\ \wedge\ (\ \bigwedge_{loc \in L_2} \texttt{next}(loc) = \texttt{FALSE}\ )\ ) \\
t(F_1 || F_2) &= t(F_1)\ \wedge\ t(F_2) \\
t(F' \backslash h) &= t(F') \\
t(\mu_c\, F') &= t(F') \\
t((I,\, E,\, O,\, x,\, P,\, Q)) &= t(P) \wedge (\ \bigvee_{e \in E} e = \texttt{TRUE}) \wedge t(Q) \wedge \texttt{next}(x) = \texttt{TRUE}\ ,
\end{aligned}
$$

wobei L_i die Menge $(V^{ctr}_{F_1+F_2} \backslash V^{ctr}_{F_i}) \cap \mathbb{L}$ bezeichnet.

2.6 Werkzeugunterstützung

Die Modellierung von Systemen anhand der in diesem Kapitel vorgestellten Techniken erfordert – aufgrund ihrer Komplexität – die Unterstützung durch eine geeignete Werkzeugkette. In diesem Abschnitt werden die verwendeten Rahmenwerke und die entwickelten Werkzeuge vorgestellt. Die bereitgestellte Unterstützung ermöglicht eine produktive Anwendung der beschriebenen Techniken.

Es wurde eine Entwicklungsumgebung für die Erstellung von Systemmodellen sowie für die Erstellung von hierarchischen Zustandsübergangsdiagrammen entwickelt. Die wichtigsten Aspekte dabei sind die Überprüfung der Modelle auf Konsistenz, eine graphische Visualisierung der Modelle und die entsprechenden Modelltransformationen. Dazu wurde jeweils für die hierarchischen Zustandsübergangsdiagramme und für das Systemmodell eine textuelle Eingabesprache und ein entsprechendes Metamodell erstellt.

Die Entwicklung von Generatoren, die aus einer textuellen Eingabesprache automatisch Quelltext erstellen, ist zeitaufwendig: Man muss die Syntax der Sprache eindeutig festlegen und die Datenstruktur für den abstrakten Syntaxbaum realisieren. Ebenso benötigt man einen Parser, der aus einer Eingabedatei den abstrakten Syntaxbaum erzeugt. Liegt die Spezifikation als abstrakter Syntaxbaum vor, kann man die Prüfung der Konsistenzbedingungen implementieren. Abschließend gilt es, die Erzeugung des Quelltext für die Zielsprache zu programmieren. Für einige dieser Schritte gibt es Rahmenwerke, die für eine effektive Entwicklung sorgen.

Das Rahmenwerk *openArchitectureWare* (OAW) [86] stellt Komponenten und Werkzeuge für ein modellbasiertes Vorgehen zur Verfügung. Diese Werkzeuge und Komponenten sind als Zusatzkomponenten in die Entwicklungsumgebung ECLIPSE [46]

integriert. Für die in diesem Abschnitt beschriebene Werkzeugunterstützung kommen die Komponenten XTENT und XPAND zum Einsatz. Zusätzlich wird der Parser-Generator ANTLR[12] eingesetzt, der nicht zu OAW gehört. Für die Beschreibungssprachen wird jeweils ein entsprechendes Metamodell definiert. ANTLR generiert auf Basis einer Syntaxbeschreibung einen Parser. Die durch ANTLR erstellten Parser werden so erweitert, dass sie eine entsprechende Instanz des Metamodells erzeugen. XTENT ist eine funktionale Programmiersprache zum Transformieren des Metamodells. Damit sind die Konsistenzbedingungen der Sprachen spezifiziert und es werden eine Reihe an Hilfsfunktionen bereitgestellt. XPAND dient der Beschreibung der Quelltextgenerierung.

Unbemerkte Flüchtigkeitsfehler beim Aufschreiben einer Spezifikation können bei der Verifikation durch einen Model-Checker Laufzeitfehler verursachen. Es spart Zeit, wenn diese Fehler von vornherein durch Prüfung von Konsistenzbedingungen ausgeschlossen werden. Deshalb werden die Modelle automatisch hinsichtlich ihrer Konsistenzbedingungen überprüft. Für die graphische Darstellung der hierarchischen Zustandsübergangsdiagramme wird das Werkzeug *yEd*[13] eingesetzt. *yEd* erzeugt anhand einer XML-Datei, die einen Graphen beschreibt, eine graphische Darstellung. Die XML-Datei wird aus der Spezifikation automatisch erzeugt. Die Beschreibung der hierarchischen Zustandsübergangsdiagramme wird durch das Werkzeug ins Systemmodell transformiert. Aus dem Systemmodell wird der Quelltext für den NUSMV-Model-Checker generiert.

2.7 Zusammenfassung

In diesem Kapitel wurden die formalen Techniken beschrieben, die in den folgenden Kapiteln benötigt werden. Ausgehend von gezeiteten Automaten wurde ein zustandsbasiertes Systemmodell angegeben. Zur graphischen Darstellung der Systemmodelle wurde eine Notation eingeführt und der Zusammenhang dieser Notation zum Systemmodell beschrieben. Als Spezifikationstechnik wurde die verzweigte Variante CTL der Temporalen Logik vorgestellt und ihre Syntax und Semantik definiert. Ferner wurde in diesem Kapitel gezeigt, wie das Systemmodell in die Eingabesprache des Model-Checkers NUSMV zu überführen ist.

Auf Grund der Tatsache, dass die Bereitstellung einer durchgängigen Werkzeugunterstützung das Erstellen der Modelle erleichtert, wurde in dieser Arbeit eine Unterstützung basierend auf den Werkzeugen ECLIPSE und OAW erstellt. Für die Beschreibungssprachen wurde mit Hilfe des Rahmenwerks OAW eine Entwicklungsumgebung realisiert. Die Umgebung umfasst einen Editor, eine Konsistenzüberprüfung und einen Generator. Mit dem Generator kann das Systemmodell in die Eingabeformate des Model-Checkers NUSMV transformiert werden. Außerdem wurde sowohl eine Werkzeugunterstützung für die graphische Darstellung der Diagramme als auch eine Werkzeugunterstützung für die Generierung der Übergangstabellen konzipiert.

[12] `http://www.antlr.org/`.
[13] *yEd* Graph Editor der Firma yWorks in der Version 3.0.

Somit stehen Techniken bereit zur (1) strukturierten Beschreibung von interaktiven Systemen, zur (2) eindeutige Spezifikation von bestimmten Eigenschaften und zur (3) automatischen Prüfung dieser Eigenschaften.

3 Grundlagen der Mensch-Maschine-Interaktion

Dieses Kapitel gibt ein konzeptuelles Modell zur Strukturierung der Interaktion an und beschreibt Grundsätze zur Dialoggestaltung. Zuerst werden die Grundlagen der Mensch-Maschine-Interaktion vorgestellt. Im zweiten Abschnitt wird dann das konzeptuelle Modell beschrieben. Im Anschluss daran wird kurz auf die kognitiven Fähigkeiten des Menschen eingegangen; denn das Verständnis dieser Fähigkeiten ist eine wichtige Voraussetzung für die Gestaltung von gebrauchstauglichen interaktiven Geräten. Zuletzt werden die sieben Grundsätze zur Dialoggestaltung der ISO 9241-110 [7] beschrieben und anhand zusätzlicher Quellen erklärt. In den folgenden Kapiteln wird auf diese Grundsätze Bezug genommen.

3.1 Grundlegende Konzepte

Bei einem *Mensch-Maschine-System* wirkt, nach Geiser [49], der Mensch mit einer Maschine mit dem Ziel zusammen, eine selbstgewählte oder vorgegebene *Arbeitsaufgabe* zu bearbeiten. Eine Arbeitsaufgabe gilt als erfüllt, wenn das angestrebte Arbeitsergebnis vorliegt. Dieses angestrebte Arbeitsergebnis wird im Allgemeinen als *Ziel* der Arbeitsaufgabe bezeichnet.

Unter einem *Dialog* versteht [49] den wechselseitigen Informationsaustausch zwischen Mensch und Maschine, der zur Erfüllung einer Aufgabe dient. Die Initiative kann dabei sowohl vom Menschen als auch von der Maschine ausgehen. Der Kontext, in dem ein Dialog stattfindet, wird als *Nutzungskontext* bezeichnet. Nach der ISO 9241-110 [7] gehören die folgenden Elemente zum Nutzungskontext: der Mensch und die ihm zugeordneten Arbeitsaufgaben, die verwendeten Arbeitsmittel (Hardware, Software und Materialien) sowie die physische und soziale Umgebung.

Der Mensch in einem solchen System wird oft als *Nutzer*, *Benutzer* oder *Anwender* bezeichnet. In dieser Arbeit wird die Maschine als *interaktives Gerät* bezeichnet und definiert als eine Kombination von Hardware- und Softwarekomponenten, die Eingaben von einem Benutzer empfängt und Ausgaben zu einem Benutzer übermittelt [1]. Die *Nutzerschnittstelle* eines interaktiven Geräts umfasst genau die Bestandteile, die Ausgaben an den Benutzer übermitteln und Eingaben vom Benutzer empfangen. Die Teile, die Eingaben empfangen, werden als *Bedienelemente* bezeichnet, die Teile, die Ausgaben übermitteln, als *Anzeigeelemente*.

3.2 Konzeptuelles Modell der Mensch-Maschine-Interaktion

Dieser Abschnitt stellt ein konzeptuelles Modell der Mensch-Maschine-Interaktion vor. Die Systematik, die durch dieses Modell beschrieben ist, stellt die Grundlage für das Analyseverfahren (s. Kap. 4) dar.

3.2.1 Struktur

Ausgangspunkt des konzeptuellen Modells ist eine Unterteilung des Gesamtsystems, die auf dem Seeheim-Modell [51] basiert. Diese Struktur ist in Abb. 3.1 schematisch dargestellt.

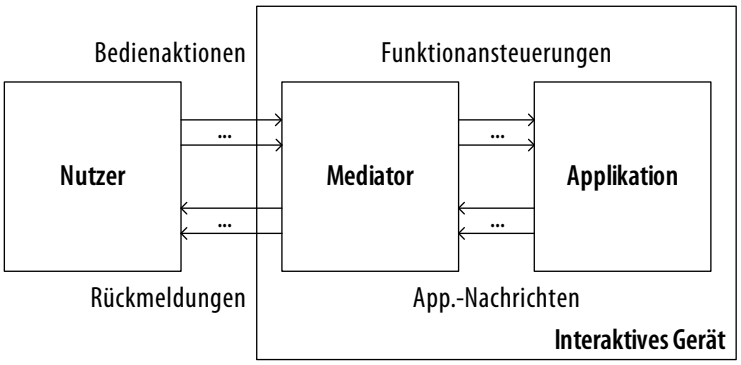


Abbildung 3.1: Komponenten des Gesamtsystems (basierend auf [51])

Das Gesamtsystem ist unterteilt in die Komponenten „Nutzer“ und „interaktives Gerät“. Das interaktive Gerät ist aufgeteilt in die Komponenten „Mediator“ und „Applikation“. Ein Bedienkonzept zeichnet sich durch sein *Look & Feel* aus, d. h. durch die Gestaltung der *Anzeigen* (*Look*) und durch den Entwurf des Verhaltens (*Feel*). Die *Rückmeldungen*, das sind die Nachrichten, die der Mediator an den Nutzer schickt, repräsentieren abstrakte Anzeigen. In diesem Aspekt unterscheidet sich die dargestellte Struktur von dem Seeheim-Modell: für die Darstellung bzw. Präsentation gibt es keine eigene Komponente.

Nutzer. Die Aktion, durch die der Nutzer eine Informationseingabe ins Gerät vollziehen kann, wird als *Bedienaktion* bezeichnet. Die Bedienaktionen des Nutzers sowie die Rückmeldungen des Mediators, sind einer *Modalität* zugeordnet. Es wird zwischen *auditiv*, *visuell* und *taktil* unterschieden. Desweiteren kann zwischen *expliziten* und *impliziten* Bedienaktionen unterschieden werden. Eine explizite Bedienaktion, ist eine Aktion, die der Nutzer bewusst durchführt, um eine Eingabe zu erwirken. Eine implizite Bedienaktion, ist dagegen eine Aktion, die der Nutzer durchführt, die vom Nutzer nicht als Eingabe intendiert ist – aber vom Mediator als solche verwendet wird. Ein Beispiel für eine implizite Bedienaktion ist ein *Timeout*.

Mediator. Der Mediator koordiniert eine Kommunikationsverbindung zwischen dem Nutzer und der Applikation. Über diese Verbindung kann der Nutzer durch Bedienaktionen Funktionen der Applikation ansteuern und bekommt Rückmeldungen angezeigt. Die Applikation kann über diese Verbindungen dem Nutzer Informationen anzeigen bzw. vom Nutzer Informationen anfordern und die Antworten des Nutzers entgegennehmen. Es ist zu bemerken, dass der Mediator, der für die Kommunikation zwischen dem Nutzer und der Applikation verantwortlich ist, in der Literatur (beispielsweise in [79]) oft als Nutzerschnittstelle bezeichnet wird.

Applikation. Die Ansteuerung der Applikation entspricht einer Folge von Interaktionen, ausgetauscht über die Kanäle „Funktionsansteuerung“ und „Applikationsnachrichten“. Diese Folge dient dazu, die Parameter der gewünschten Funktion zu setzen und im Anschluss daran die Funktion anzustoßen. Die Funktionen der Applikation werden unterteilt in *widerrufbar* und *permanent* sowie in *nicht-schädlich* und *schädlich*. Eine Funktion ist widerrufbar, wenn der Mediator eine Ansteuerung zu einem späteren Zeitpunkt rückgängig machen kann, sonst gilt die Funktion als permanent. Ein Beispiel für eine permanente Funktion ist das Versenden einer E-Mail. Eine Funktion wird als schädlich bezeichnet, wenn sie in der Lage ist Arbeitsergebnisse zu zerstören oder der Umgebung Schaden zuzufügen.

Die Nachrichten, die dazu dienen, Parameter eine Funktion zu setzen, werden unterteilt in *Objekte* und *Operationen*. Objekte sind beispielsweise Dateien, Musiktitel, Kontakte, Fotos etc; Operationen stehen für Aktionen, die ausgeführt werden können, z. B. das Löschen von Dateien, das Abspielen von Musiktitel, das Anrufen eines Kontakts oder das Anzeigen eines Fotos. In der Regel werden Objekte in der Anzeige durch Substantive und Operationen durch Verben bezeichnet.

3.2.2 Dialoge und Dialogstränge

In dieser Arbeit werden Mediatoren untersucht, die mit Hilfe von *Dialogsträngen* und *Dialogen* die Kommunikation zwischen dem Nutzer und der Applikation strukturieren. Ein Dialogstrang verwaltet eine Menge an Dialogen und jeder Dialog ist genau einem Strang zugeordnet. Ein Dialog kann eine Menge an Unterdialogen besitzen. Ein Dialog besitzt eine Menge an *Attributen*. Die Belegung dieser Attribute stellt den *Dialogzustand* dar.

Abb. 3.2 zeigt die verschiedenen Zustände, die ein Dialog einnehmen kann. Die Begriffe basieren bzw. erweitern die Terminologie aus [4]. Ein Dialog ist entweder *geöffnet* oder *geschlossen*:

1. **geschlossen.** Ein geschlossener Dialog wartet darauf, geöffnet zu werden. Das Öffnen eines Dialogs ist in der Regel mit dem Ziel verbunden, bestimmte Funktionen der Applikation anzusteuern. Beim Öffnen eines Dialogs werden seine Attribute mit einem voreingestellen Wert belegt.

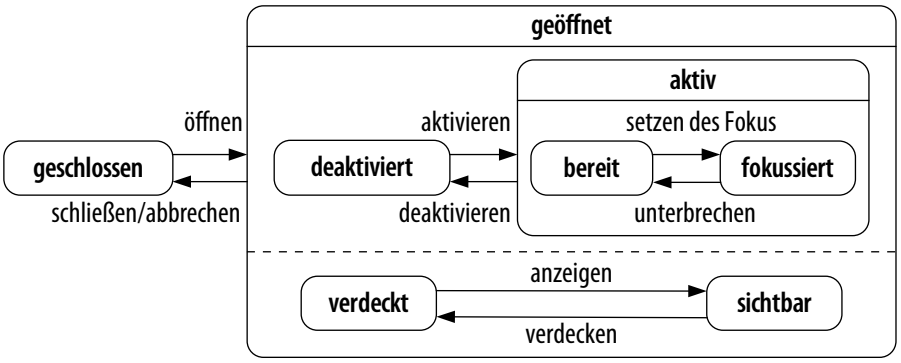


Abbildung 3.2: Zustände der Dialoge

2. **geöffnet.** Ein geöffneter Dialog kann geschlossen bzw. abgebrochen werden. Der Unterschied ist wie folgt: Wenn das Ziel eines Dialogs erfüllt ist, wird der Dialog *geschlossen*; ist hingegen das Ziel nicht erfüllt, wird der Dialog *abgebrochen*. Beim Schließen bzw. beim Abbrechen eines Dialogs wird die Belegung der Attribute gelöscht. Ein geöffneter Dialog ist entweder *deaktiviert* oder *aktiv*:

   a) **deaktiviert.** Ein deaktivierter Dialog wartet darauf, aktiviert zu werden. Die Aktivierung eines Dialogs wird typischerweise durch eine Bedienaktion angestoßen.

   b) **aktiv.** Ein aktiver Dialog kann deaktiviert werden. Beim Deaktivieren eines Dialogs wird dahingehend unterschieden, ob der Dialog, der stattdessen aktiviert wird, in der Hierarchie übergeordnet, untergeordnet oder gleichgestellt ist. Der *Sprung*[1] zu einem übergeordneten Dialog, wird als *Rückwärts-Sprung* bezeichnet. Der Sprung zu einem untergeordneten Dialog, wird als *Vorwärts-Sprung* bezeichnet. Der Sprung zu einem gleichgestellten Dialog, wird als *Tabulator-Sprung*[2] bezeichnet. Das Deaktivieren eines Dialogs bedeutet in der Regel, dass er zu einem späteren Zeitpunkt fortgesetzt werden soll. Deshalb wird die Belegung seiner Attribute gespeichert. Ein aktiver Dialog ist entweder *bereit* oder *fokussiert*:

      i. **bereit.** Ein Dialog im Zustand „bereit“ wartet darauf, den *Fokus* zu erhalten. Solange der Dialog bereit ist, kann er nicht auf Bedienaktionen reagieren.

      ii. **fokussiert.** Ein Dialog im Zustand „fokussiert“ reagiert auf Bedienaktionen des Nutzers. Ein fokussierter Dialog kann unterbrochen werden. Das Unterbrechen eines Dialogs entspricht dem Wechsel des Fokus, d. h. es wird zwischen den Dialogsträngen navigiert. Dieser Sprung wird als *Strang-Wechsel-Sprung* bezeichnet.

Für Stränge und Dialoge bestehen in dieser Arbeit die folgenden Einschränkungen: Zu einem Zeitpunkt ist genau ein Dialog fokussiert. Die Unterdialog-Beziehung zwischen

[1] Die Begriffe „Sprung“ und „Navigation“ werden in dieser Arbeit synonym verwendet.
[2] Dieser Begriff wurde gewählt, da die Tabulator-Taste in der Regel diese Art der Navigation auslöst.

den Dialogen bildet eine Hierarchie. Ein Unterdialog gehört zum selben Strang wie der seines übergeordneten Dialogs. In der Menge der geöffneten Dialoge eines Strangs gibt es höchstens einen Dialog, der aktiv ist. Ist ein Dialog geöffnet, dann ist auch sein übergeordneter Dialog geöffnet.

Ein Dialog ist entweder *sichtbar* oder *verdeckt*. Bevor ein Dialog Informationen an den Nutzer übermitteln kann, muss ein entsprechendes Anzeigeelement zur Verfügung stehen. Die Verfügbarkeit wird durch den Zustand „sichtbar“ ausgedrückt.[3]

Einen Fokus kann man sich als Sperre für einen Eingabekanal vorstellen. Eine Sperre für einen Eingabekanal ist notwendig, wenn sich mehrere Dialogstränge einen Eingabekanal teilen. Die Sperre dient dann als Koordinationsmittel, das dafür sorgt, dass immer nur ein Dialogstrang auf Bedienaktionen des Eingabekanals reagiert. In dieser Arbeit wird davon ausgegangen, dass es nur einen Eingabekanal gibt. Es ist allerdings durchaus möglich, das Konzept durch weitere Eingabekanäle und somit auch durch weitere Fokusse zu erweitern.

Ein Dialog mit der folgenden Eigenschaft wird als *modal*[4] bezeichnet: Wenn sich ein modaler Dialog im Zustand „fokussiert“ und „sichtbar“ befindet, ist keine Zustandsänderung durch den Nutzer möglich, außer ein Schließen des Dialogs. Modale Dialog werden eingesetzt, um beispielsweise bei einer Fehlermeldung sicherzustellen, dass der Nutzer diese zur Kenntnis genommen hat. Allerdings muss der Einsatz modaler Dialoge genau geprüft werden, da das Prinzip „Steuerbarkeit“ (s. Abs. 3.4) in einem starken Ausmaß eingeschränkt wird. Denn die Einschränkung, dass ein modaler Dialog nicht unterbrochen werden kann, hat zur Folge, dass Bedienaktionen, die einen Fokus-Wechsel auslösen, keine Wirkung haben.[5] Der Nutzer kann – solange er den Dialog nicht schließt – ausschließlich mit diesem interagieren.

Die Sprünge innerhalb eines Dialogstrangs kann man sich als Operationen auf einem Stapel vorstellen. Derjenige Dialog, der oben auf dem Stapel liegt, ist der aktive Dialog, die restlichen Dialoge des Stapels sind deaktiviert. Bei einem Vorwärts-Sprung wird ein Unterdialog des obersten Dialogs auf den Stapel gelegt, wobei dieser Unterdialog zuvor geöffnet werden muss, wenn er geschlossen ist. Bei einem Rückwärts-Sprung wird der oberste Dialog vom Stapel genommen und gegebenenfalls geschlossen oder abgebrochen. Bei einem Tabulator-Sprung wird der oberste Dialog vom Stapel genommen und ein anderer Unterdialog auf den Stapel gelegt.

3.2.3 Menüs, Optionen und Optionsgruppen

In dieser Arbeit werden *Menüs* als spezielle Dialoge aufgefasst. Diese Art der Dialogführung wird als *menübasiert* bezeichnet. Ein Menü enthält entweder eine Menge

[3] Um dieses Konzept auf unterschiedliche Modalitäten zu erweitern, kann man anstelle des Zustands „sichtbar“ einen Zustand „wahrnehmbar“ (entweder „sichtbar“, „hörbar“ oder „fühlbar“) einführen.

[4] Für eine Beschreibung der praktischen Anwendung siehe z. B. das Konzept „*modal panel*“ in [102].

[5] Ein Beispiel dafür ist der sog. Komfortaufruf des iDrive-Konzepts (s. Kap. 5).

an *Optionen* oder eine Menge an *Optionsgruppen*. Eine Optionsgruppe ist ein Untermenü und enthält eine Menge an Optionen. Typischerweise kann zwischen den Optionsgruppen eines Menüs direkt navigiert werden. Eine solche Navigation ist, da die Optionsgruppen derselben Ebene zugeordnet sind, eine Tabulator-Navigation. Die Navigationsbeziehung (Rückwärts, Vorwärts, Tabulator und Strang-Wechsel) zwischen den Dialogen bildet einen Graphen, der bei einer menübasierten Dialogführung auch als *Menü-Netz* bezeichnet wird. Die graphische Darstellung eines Menüs wird als *Menüfeld* bezeichnet.

Durch Bedienaktionen kann auf das Menü zugegriffen werden, d. h., der Nutzer kann eine oder mehrere Optionen *auswählen*[6] und diese dann *ausführen*. Der Zugriff kann dabei durch unterschiedliche Modalitäten erfolgen. Können mehrere Optionen eines Menüs ausgewählt werden, dann spricht man von einer *Mehrfachauswahl*. Wird eine Option ausgeführt, werden die für die Option festgelegten Aktionen vom Mediator durchgeführt.

Positionszeiger[7] dienen im Allgemeinen zur Anzeige von Dialogattributen. Bei der visuellen Darstellung von Menüs wird in der Regel die zur Auswahl stehende Option durch einen Positionszeiger graphisch hervorgehoben.

Auf Basis des Dialogmodells können Spezialfälle von Menüs beschrieben werden. Ein *flüchtiges* Menü ist ein Menü, das weder im Zustand „deaktiviert“ noch im Zustand „bereit“ sein kann. Das bedeutet, dass wenn ein flüchtiges Menü unterbrochen bzw. deaktiviert wird, dann wird es gleichzeitig auch geschlossen bzw. abgebrochen. Das hat zur Folge, dass man solche Menüs nicht fortsetzen kann. Ein Beispiel für flüchtige Menüs sind in der Regel die sogenannten *Pulldown-* oder *Popup*-Menüs. Auf Grund der Tatsache, dass beim Öffnen eines Dialogs, die Attribute kontextsensitiv gesetzt werden können, kann fälschlicher Weise der Eindruck entstehen, dass ein flüchtiges Menü seinen Zustand gespeichert hat.

Anhand der Interaktion der Menüs mit der Applikation können diese – wenn auch nur unscharf – nach dem Schema aus [10] kategorisiert werden: Ein Menü, dass nicht mit der Applikation interagiert, wird *Übergangsmenü* genannt; ein Menü, dass vorderrangig Parameternachrichten an die Applikation schickt, wird *Einstellmenü* genannt; ein Menü, dass vorderrangig Nachrichten der Applikation dem Nutzer anzeigt wird *Verweilmenü* genannt. Beispiele für Verweilmenüs sind Menüs, die Musiktitel abspielen oder die Zielführung anzeigen. In der Regel sind diese Menüs nicht eindeutig voneinander abzugrenzen: Ein Menü, das Musiktitel abspielt erlaubt oft auch das Einstellen der Lautstärke.

3.2.4 Dialogschritt

Ein *Dialogschritt* beginnt mit einer Bedienaktion des Nutzers. Diese Aktion zieht eine Reihe an internen Aktionen des Geräts nach sich. Der Nutzer tätigt keine weiteren Eingaben – er reagiert ausschließlich auf Rückmeldungen des Mediators. Wenn

[6] synonym wird für „auswählen“ auch das Wort „selektieren“ verwendet.
[7] engl.: *cursor*.

der Mediator keine weiteren internen Aktionen durchführen möchte, schickt er eine Rückmeldung an den Nutzer. Die Rückmeldung stellt einen *Prompt* dar, d. h. eine Aufforderung eine Bedienaktion durchzuführen. Die Anzahl der Takte von der Bedienaktion bis einschließlich zur Rückmeldung ist die *Antwortzeit* des Mediators. Schickt der Mediator eine Rückmeldung an den Nutzer, dann antwortet dieser immer im nächsten Takt mit einer Bedienaktion. Das Verstreichen von Zeit wird durch Timeout-Bedienaktionen des Nutzers modelliert.

3.2.5 Modus und Modi-Konfigurationen

Abhängig vom Zustand des Geräts kann eine Bedienaktion, wenn sie ausgeführt wird, eine unterschiedliche Wirkung haben. Ein *Modus* einer Bedienaktion fasst alle Zustände eines Geräts zusammen, in denen die Bedienaktion die gleiche Wirkung hat [100, 99]. In dieser Arbeit wird als Wirkung einer Bedienaktion die Ereignisfolge, die während eines Dialogschritts auftritt, verstanden. Es werden die folgenden Ereignisse betrachtet: Öffnen eines Dialogs, Schließen eines Dialogs, Abbrechen eines Dialogs, Rückwärts-Sprung, Vorwärts-Sprung, Tabulator-Sprung, Dialogstrang-Wechsel-Sprung und Ansteuerung der Applikation. Eine *Modi-Konfiguration* beschreibt für einen Zustand die Modi der einzelnen Bedienaktionen.

3.2.6 Beispiel

Abb. 3.3 zeigt eine Folge von Anzeigen und beschreibt den Zustand der Dialoge für die jeweilige Anzeige. Das Beispiel besteht aus den drei Dialogsträngen „Start“, „Navigation“ und „Assistenz“. Der Dialogstrang „Start“ verwaltet das Menü „Startmenü“. Die Menüs „Navigations-Liste“ und „Zielliste“ werden vom Dialogstrang „Navigation“ verwaltet. Der Dialogstrang „Assistenz“ besitzt das Menü „Assistenzmenü“.

1. Zu Beginn ist das Startmenü fokussiert und reagiert somit auf Bedienaktionen. Das Assistenzmenü ist aktiv, besitzt aber nicht den Fokus. Es ist daher in dem Zustand „bereit“. Sowohl die Navigations-Liste als auch die Zielliste sind geschlossen.

2. Durch eine Bedienaktion wird zur Navigations-Liste gewechselt, die sich dann im Zustand „fokussiert“ befindet. Dieser Zustand ist in der Anzeige durch den hellen Hintergrund-Farbton des Menüfeldes zu erkennen. Das Startmenü bleibt aktiv, besitzt aber nicht mehr den Fokus. Es ist im Zustand „bereit“ und nicht mehr sichtbar.

3. Durch eine weitere Bedienaktion springt der Nutzer zum Menü „Zielliste“. Die Navigations-Liste wird deaktiviert, bleibt aber geöffnet.

4. Im nächsten Schritt, wechselt der Nutzer zum Assistenzmenü. Die Zielliste bleibt aktiv und gibt den Fokus ab. An dieser Stelle ist ein Schwachpunkt der Darstellung auszumachen: Die Menüs des Dialogstrangs „Navigation“ befinden sich nicht im gleichen Zustand (die Navigations-Liste ist deaktiviert; die Zielliste

ist aktiv und im Zustand „bereit“). Jedoch wird dieser Unterschied durch die Anzeige nicht übermittelt und somit kann der Nutzer anhand der Anzeige nicht vorhersagen, welche genaue Wirkung ein Wechsel zurück zur Navigation hat.

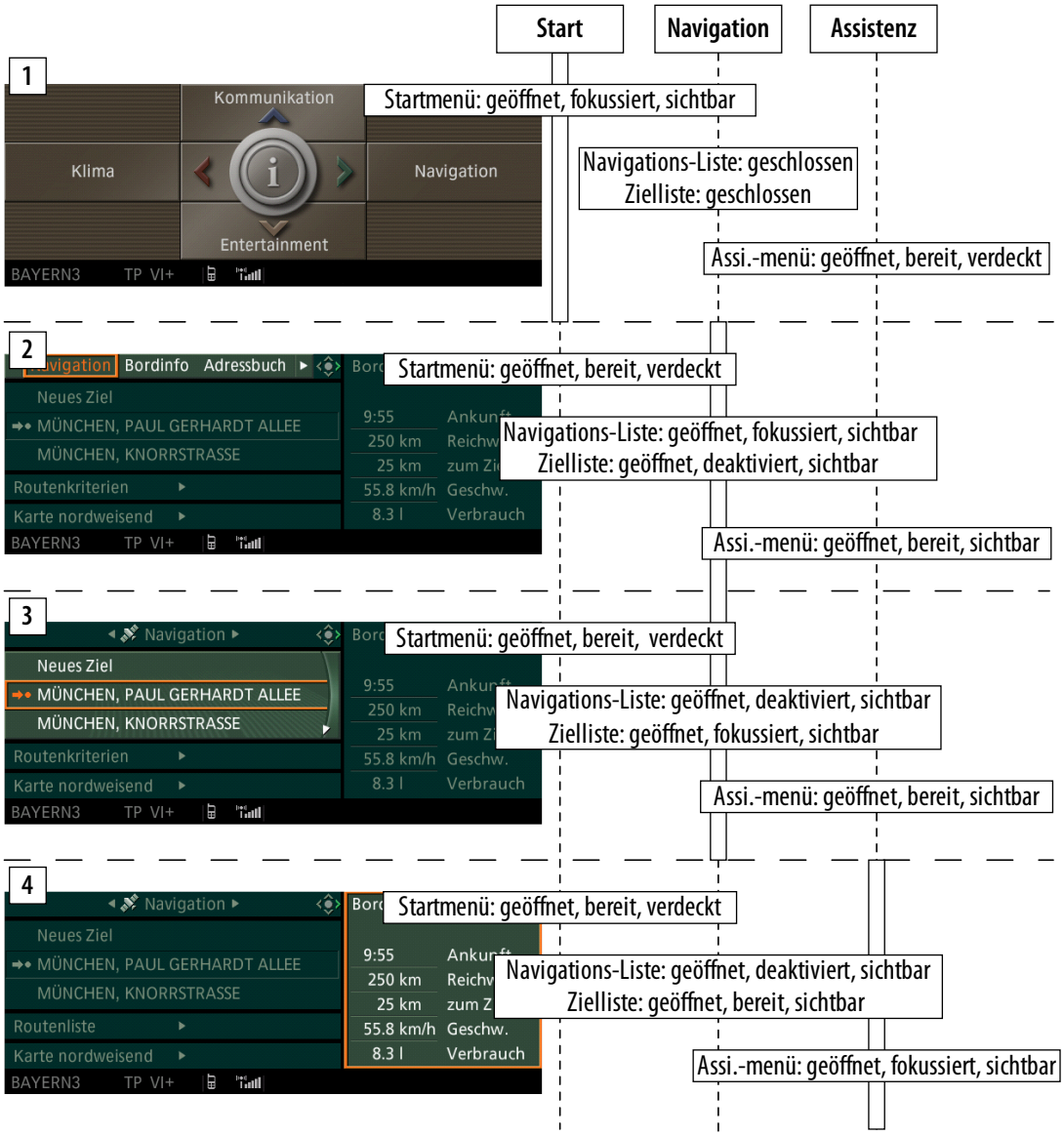


Abbildung 3.3: Folge von Zuständen der Dialoge

3.3 Kognitive Fähigkeiten des Menschen

Unter dem Begriff „Kognition“ werden im Allgemeinen die Informationsverarbeitungsprozesse des Menschen verstanden. Die kognitive Psychologie liefert Theorien, die die Fähigkeiten und Grenzen der Informationsverarbeitungsprozesse des Menschen aufzeigen. Die grundlegenden Prinzipien für die Gestaltung von Nutzerschnittstellen

(s. Abs. 3.4) basieren auf diesen Theorien. Deshalb werden im Folgenden die zentralen Konzepte „Wahrnehmung“, „Aufmerksamkeit“ und „Gedächtnis“ vorgestellt.

3.3.1 Wahrnehmung

Unter Wahrnehmung versteht man nicht nur das Erfassen von Informationen, welche die Sinnesorgane des Menschen erreichen, sondern auch eine erste Interpretation dieser Informationen. Das heißt, die Wahrnehmung der Außenreize ist kein passiver Prozess sondern ein aktiver. Zum Beispiel ist die visuelle Wahrnehmung dafür verantwortlich, aus der Lichtenergie der Umgebung eine Repräsentation der in der Umgebung enthaltenen Objekte zu bilden und höheren kognitiven Verarbeitungsprozessen zur Verfügung zu stellen.

Dabei werden nach [12] die folgenden Schritte durchgeführt: Zuerst findet eine Merkmalsextraktion statt. Dieser Prozess erkennt Kanten und Balken in der Umgebung. Der nächste Schritt ergänzt diese sogenannte Primärskizze mit Informationen über Oberflächen und Tiefen. Im darauffolgenden Schritt werden aus diesen Informationen Objekte, also zusammenhängende Einheiten, gebildet. Die Verwendung von zusätzlicher Kontextinformation führt dann zur Erkennung und der Identifizierung der Objekte.

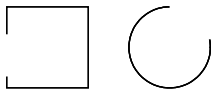


Abbildung 3.4: Unvollständige Formen

Die Erkennung der Objekte wird erleichtert, wenn die Darstellung der Informationen auf den Anzeigen den *Gestaltgesetzen* Rechnung trägt. Abb. 3.4 zeigt, dass unvollständige Formen als eigenständige Objekte wahrgenommen werden. Diese Tatsache ist durch das Gestaltgesetz „Geschlossenheit“ beschrieben. Aufgrund der Neigung, unvollständige Formen zu komplettieren, erkennt der Mensch geschlossene Formen schneller als offene.

Für jeden der verschiedenen Sinneskanäle steht ein Kurzzeitspeicher zur Verfügung. Zum Beispiel kann der visuell-sensorische Kurzzeitspeicher die gesamte wahrgenommenen visuelle Information zwischenspeichern. Allerdings geht diese Information ohne Aufmerksamkeitszuweisung schnell verloren.

3.3.2 Aufmerksamkeit

Die kognitiven Ressourcen, die dem Menschen zur Verfügung stehen, um Informationen zu verarbeiten, sind beschränkt [12]. Damit wichtige Informationen bevorzugt

behandelt werden können, müssen weniger wichtige Informationen entsprechend vernachlässigt werden. Dies wird durch eine Verteilung der kognitiven Ressourcen erreicht. Das Thema Aufmerksamkeit beschäftigt sich mit dem menschlichen Aufmerksamkeitsprozess, dessen Kernaufgabe die optimale Verteilung der kognitiven Ressourcen ist.

Eine Frage, mit der sich die Forschung befasst, ist, ob und wieviel Ressourcen denjenigen Informationsquellen zugeteilt werden, die sich nicht im Zentrum der Aufmerksamkeit befinden. Ein erster Standpunkt, den man einnehmen könnte, ist der Folgende: konzentriert man sich auf eine Informationsquelle, dann werden die Informationen der anderen Quellen herausgefiltert und somit nicht verarbeitet. Allerdings zeigen Untersuchungen, dass alle Informationsquellen semantisch verarbeitet werden – auch wenn die Aufmerksamkeit auf einer Informationsquelle liegt. Dies ermöglicht es, auf bestimmte Informationen, wie beispielsweise ein bekannter Name, entsprechend zu reagieren. Die unbeachteten Informationen werden aber weniger gründlich verarbeitet als diejenigen Informationen, die im Zentrum der Aufmerksamkeit stehen.

Der Aspekt „Automatisiertheit“ spielt eine entscheidende Rolle bei der Verarbeitung von Informationen. Es ist bekannt, dass ein Prozess, der intensiv geübt worden ist, deutlich weniger Aufmerksamkeit erfordert, als ein Prozess, der nicht geübt worden ist. Deshalb werden kognitive Prozesse in die zwei Klassen *automatische Prozesse* und *kontrollierte Prozesse* eingeteilt [12]. Automatische Prozesse laufen ohne bewusste Kontrolle ab und erfordern sehr wenig Aufmerksamkeit; kontrollierte Prozesse hingegen erfordern sowohl eine bewusste Kontrolle als auch das Zentrum der Aufmerksamkeit. Automatisierte Prozesse werden auch als eine *Gewohnheit* (engl. *habit*) bezeichnet.

Für die Gebrauchstauglichkeit eines interaktiven Geräts ist der Grad, zu dem das Gerät die Bildung von Gewohnheiten erlaubt bzw. fördert, von Bedeutung. Raskin unterstreicht dies in [91]:

> *The ideal humane interface would reduce the interface component of a user's work to benign habituation. Many of the problems that make products difficult and unpleasant to use are caused by human-maschine design that fails to take into account the helpful and injurious properties of habit formation.*

3.3.3 Gedächtnis

Dieser Abschnitt beschäftigt sich mit den Prozessen, die Informationen ins Gedächtnis des Menschen *aufnehmen*, *behalten* und *abrufen*. Das Modell des Gedächtnis ist in die zwei Bereiche „Arbeitsspeicher“ und „Langzeitspeicher“ unterteilt.

Der Arbeitsspeicher vergisst einen Inhalt, wenn dieser Inhalt nicht in kurzen Abständen aktiviert wird. Die Inhalte des Arbeitsspeicher werden zu Einheiten zusammengefasst, den sogenannten *Chunks*. Die Bildung dieser Chunks hängt stark von den charakteristischen Eigenschaften der jeweiligen Person ab. Je nach dem Vorwissen

und den vorhandenen Strategien zur Bildung von Chunks, kann die Person Teile der Information zu Chunks zusammenfassen.

Die Anzahl der Chunks, die ein Mensch zu einem Zeitpunkt in seinem Arbeitsspeicher *memorieren* kann, ist beschränkt. Für die Gestaltung interaktiver Geräte lassen sich daraus zwei Punkte ableiten: Zum einen sollte vermieden werden, dass der Arbeitsspeicher des Nutzer überladen wird, das heißt, dass das Gerät diejenigen Informationen, die der Nutzer zu einem Zeitpunkt benötigt, ihm auf der Anzeige bereitstellt; zum anderen sollte die Bildung von Chunks durch das Gerät gefördert werden. Eine nach ergonomischen Gesichtspunkten strukturierte Darstellung erleichtert dem Nutzer das Zusammenfassen von logisch zusammengehörigen Informationen zu Chunks.

Der Langzeitspeicher behält seine Inhalte permanent. Allerdings muss entsprechender Aufwand getrieben werden, dass eine Information vom Arbeits- in den Langzeitspeicher transferiert wird. Die Bearbeitung der Information während des Aufnehmens wird als *Kodierung* bezeichnet. Bei dem Aufnehmen der Information, gibt es eine Reihe von Faktoren, die das Behalten begünstigen: Zum einen die Verbindung mehrerer Sinneskanäle (z. B. auditiv-visuelle Kodierung); zum anderen die Verankerung der Information im Vorwissen (*Assimilation*) und die Vernetzung der Information. Eine solche verstärkt *elaborative* Kodierung führt zu einem besseren Behalten. Unter Elaboration versteht man im Allgemeinen das Ausschmücken der Information, die man behalten möchte, mit zusätzlicher Information. Der Grad, zu dem die gespeicherten Inhalte kognitiv miteinander verknüpft sind, wird als die *Tiefe der Verarbeitung* bezeichnet.

Ein Kernproblem ist, dass sich der Nutzer an Informationen, die er bereits in den Langzeitspeicher aufgenommen hat, nicht mehr erinnern kann. Dabei können zwei verschiedene Ursachen vorliegen: Die erste Ursache ist, dass sich die Informationen nicht mehr im Gedächtnis befinden, die zweite, dass die Informationen nicht abrufbar sind – obwohl sie sich im Gedächtnis befinden. Der erste Fall trifft oft nur scheinbar zu: Experimente belegen, dass „vergessene“ Inhalte sich in der Regel im Gedächtnis befinden [12].

Die *Abfrage* von Inhalten aus dem Langzeitspeicher besteht aus den zwei Phasen *Erinnern* und der eigentliche *Abruf*. Um sich an einen Gedächtnisinhalt erinnern zu können, muss der entsprechende Bereich der kognitiven Struktur aktiviert werden, damit eine *Rekonstruktion* möglich ist. Die Rekonstruktion basiert auf emotionalen Aspekten und der Einstellung der Person zum Inhalt sowie semantischen Querbezügen. Diese nehmen die Rolle als Stichwörter (engl. *cues*) für die Rekonstruktion ein. Eine solche aktive Rekonstruktion ist deutlich schwieriger als eine Rekonstruktion durch direktes Wiedererkennen. Dazu kommt, dass die aktive Rekonstruktion durch sogenannte *Interferenzen* gestört werden kann: Das zusätzliche Lernen von Assoziationen zu einem Stimulus kann das Vergessen alter Assoziationen zur Folge haben [12].

3.4 Grundsätze der Dialoggestaltung

Auf Basis der Theorien der kognitiven Psychologie sowie Erkenntnissen aus dem Gebiet der Mensch-Maschine-Interaktion sind eine Reihe von allgemeinen und grundlegenden Prinzipien für die Gestaltung von Nutzerschnittstellen festgelegt worden. Diese Arbeit orientiert sich an den Grundsätzen der Norm ISO 9241-110 [7].

Die ISO 9241-110 beschreibt unabhängig von einer bestimmten Dialogtechnik sieben Grundsätze der Dialoggestaltung. Die in dieser Arbeit identifizierten Merkmale von Nutzerschnittstellen stehen im Bezug zu diesen Grundsätzen, die im Folgenden, unter Verwendung der *acht goldenen Regeln zur Gestaltung von Nutzerschnittstellen* von Shneiderman [95], den Richtlinien zur Gestaltung von menübasierten Dialogen der ISO 9241-14 [3] sowie der Quellen [50, 42, 99, 102, 91], beschrieben sind. Auf Grund der Tatsache, dass die in dieser Arbeit verwendete Fallstudie (s. Kap. 5) aus dem Automotive-Bereich entnommen ist, sind die Erklärungen mit Richtlinien aus [5] und [35] ergänzt. Diese Richtlinien adressieren interaktive Geräte, die in Fahrzeugen verbaut sind.

3.4.1 Aufgabenangemessenheit

> *Ein interaktives Gerät ist aufgabenangemessen, wenn es den Nutzer unterstützt, seine Arbeitsaufgabe* effektiv *und* effizient *durchzuführen.*

Ein interaktives Gerät soll keine Bedienaktionen vom Nutzer fordern, die nicht nötig sind. Ferner soll sich die Funktionalität des Geräts aus den charakteristischen Eigenschaften der Arbeitsaufgabe ergeben, anstatt aus der zur Aufgabenerledigung eingesetzten Technologie [95].

Bei einer menübasierten Dialogführung soll z. B., wenn die Aufgabe es verlangt, die Möglichkeit bestehen, dass der Nutzer direkt zwischen zwei verschachtelten Menüs springen kann und nicht den Weg über die übergeordneten Menüs nehmen muss. Ist es bei einem Menü-Netz sinnvoll die verschiedenen Ebenen durch verschiedene Pfade zu erreichen, dann soll dies dem Nutzer auch ermöglicht werden [3].

Wenn sich beispielsweise aus der Aufgabe sinnvolle voreingestellte Belegungen für Attribute der Dialoge oder für Attribute der Applikation ableiten lassen, soll diese dem Benutzer so angeboten werden, dass keine Bedienaktionen des Nutzers erforderlich sind. Diese Werte sollen vom Nutzer durch andere Belegungen ersetzt werden können. Das Setzen von voreingestellten Werten bedeutet, dass das Gerät in Abhängigkeit der Aufgabe Annahmen über die Bedürfnisse des Nutzers ableitet und somit gewisse Aktionen automatisch durchführt. Ein solches automatisches Verhalten muss jedoch mit dem Grundsatz „Erwartungskonformität“ (s. Abs. 3.4.4) abgestimmt werden. Grundsätzlich geht aus diesem Grundsatz hervor, dass an bestimmten Stellen Kontextsensitivität erforderlich ist.

3.4.2 Selbstbeschreibungsfähigkeit

> *Aus der Interaktion soll zu jedem Zeitpunkt hervorgehen, in welchem Dialog, an welcher Stelle im Dialog der Nutzer sich befindet und welche Bedienaktionen möglich sind.*

Der Nutzer soll über den gegenwärtigen Dialogzustand, der für die Arbeitsaufgabe relevant ist, informiert werden, d. h. aus den Rückmeldungen soll die Information in welchem Dialog, an welcher Stelle und welche Bedienaktionen möglich sind hervorgehen. Rückmeldungen sollen Rückschlüsse auf den Zustand des Geräts erlauben. Sind beispielsweise bei einem hierarchischen Menü gleichzeitig mehrere Menüfelder angezeigt, die unterschiedlichen Ebenen bzw. Dialogsträngen zugeordnet sind, dann sollen die Beziehungen zwischen den Feldern offensichtlich sein.

Auf jede Eingabe des Nutzers soll das Gerät mit einer Rückmeldung antworten. In Abhängigkeit davon, welche Wirkung die folgenden Eingaben des Nutzers im Gerät erzielen werden, soll die Deutlichkeit der Antwort ausfallen.

Ein Prinzip, das stark in Bezug zur Selbstbeschreibungsfähigkeit steht, ist die *Vorhersagbarkeit* (engl. *predictability*) [50]. Die Vorhersagbarkeit bezieht sich auf die Bedienaktionen, die der Nutzer in Reaktion auf eine Rückmeldung ausführt. Das Prinzip adressiert die Frage, ob die gegenwärtige Rückmeldung ausreicht, die Wirkung der möglichen Bedienaktionen vorherzusagen. In [42] wird dieses Problem als „*gone away for a cup of tea problem*“ bezeichnet. Es beschreibt die Situation, in der man die Interaktion mit einem Gerät wieder aufnimmt, wobei nur die aktuelle Rückmeldung bekannt ist.

Die Verwendung einer Vielzahl an Modi kann die Selbstbeschreibungsfähigkeit einer Nutzerschnittstelle erschweren, da eine Vielzahl an Modi es dem Nutzer teilweise unmöglich macht, den Zustand eines Geräts intuitiv zu erfassen [91]. Ein Modusfehler tritt dann auf, wenn der Nutzer eine Bedienaktion ausführt, dessen Wirkung er nicht intendiert hat. Norman gibt in [83] drei Maßnahmen an, um Modusfehler zu reduzieren: Keine Modi verwenden; Modi erkennbar hervorheben, d. h. sicherstellen, dass dem Benutzer der Zustand des Geräts klar und deutlich angezeigt wird; sicherstellen, dass die Bedienaktionen der verschiedenen Modi nicht dieselben sind, damit eine Bedienaktion, die im falschen Modus ausgeführt wird, keine Probleme verursacht.

3.4.3 Steuerbarkeit

> *Der Nutzer soll in der Lage sein, einen Dialog zu initiieren und sowohl den Ablauf als auch die Geschwindigkeit des Dialogs beeinflussen zu können.*

Grundlegend ist, dass der Nutzer das Verhalten des Geräts jederzeit steuern kann. Vom Gerät initiierte Zustandswechsel, die den Nutzer überraschen, seine Eingaben außer Kraft setzen oder die Wirkungen bereits getätigter Eingaben rückgängig machen, sollen vermieden werden. Der Nutzer soll selbst bestimmen können, wann ein Dialog

vollständig bearbeitet ist oder abgebrochen werden soll. Bei einer menübasierten Dialogführung soll es beispielsweise zu jedem Zeitpunkt möglich sein, mit Bedienaktionen zum Ausgangsmenü zurückzukehren. Das Erreichen der nächsthöheren Ebene einer hierarchischen Menüstruktur soll ebenfalls jederzeit durch Bedienaktionen möglich sein.

Dieser Grundsatz ist insbesondere für interaktive Geräte, die im Fahrzeug verbaut sind, von Bedeutung. Der Fahrer soll einen unterbrochenen Dialog fortsetzen können und zwar an der Stelle, an der der Dialog unterbrochen worden ist, d. h. die Wirkung bereits getätigter Eingaben wird nicht gelöscht. Sonst könnte der Fahrer sich veranlasst fühlen, einen Dialog nicht zu unterbrechen, obwohl die Verkehrssituation genau dies erfordern würde. Aus diesem Grund sollen implizite Timeout-Bedienaktionen bisherige Eingaben des Fahrers nur dann löschen, wenn es sich bei den Eingaben um einen kurzen Dialog handelt oder eine hinreichend lange Zeitüberschreitung erfolgt ist. Prinzipiell „sollte das Gerät den Fahrer nicht veranlassen, Eingaben unter Zeitdruck vorzunehmen“ [35]. Dazu soll die Möglichkeit bestehen, einen Dialog zu unterbrechen und diesen zu einem späteren Zeitpunkt fortzusetzen. Auch Bengler et al. [16] kommen zu dem Schluss, dass nicht der schnellste, unmittelbarste Dialog der beste ist, sondern der Dialog, den man am leichtesten unterbrechen und fortsetzen kann.

3.4.4 Erwartungskonformität

> *Das Verhalten eines Dialogs soll die Belange des Nutzer, die aus dem Nutzungskontext ableitbar sind, und relevante allgemeine Konventionen erfüllen.*

Die Gestaltung soll die ganzen Bandbreite der Anforderungen, die sich durch die unterschiedlichen Nutzer ergibt, berücksichtigen. Sie soll sowohl gezielte Unterstützung für Anfänger bieten, wie auch den Erwartungen und Belange geübter Nutzer gerecht werden.

Die *Konsistenz* der Nutzerschnittstelle spielt dabei eine entscheidende Rolle. In ähnlichen Situationen sollen die benötigten Eingabeaktionen konsistent zueinander sein, d. h. die Überführung des Geräts in einen neuen Zustand wird auf einheitliche Art und Weise herbeigeführt. Die Menge der Begriffe, die in Aufforderungen, in Menüs oder in der Hilfe verwendet werden, das Layout der Anzeigen sowie die verwendeten Schriften und deren Größe und Farbe sollen ebenfalls konsistent sein. Das folgende Beispiel einer Richtlinie beschreibt das einheitliche Verhalten eines Bedienkonzepts bei Betätigung der Return-Taste [55]:

> *Durch Betätigung der Return-Taste bewegt man sich in der Menühierarchie immer einen Schritt nach oben. D. h., es erfolgt in keinem Fall*

> *ein Rücksprung in ein vorher aktiviertes Menü eines parallelen Softkey-Menüastes[8] oder in eine andere Hauptgruppe[9]. Aktive Funktionen[10] (z. B. Sender abspeichern, offene Drop-down-box) werden abgebrochen [...]. In der obersten Ebene der Hauptgruppen hat die Betätigung von Return keine Funktion.*

Eine weitere Maßnahme zur Verbesserung der Konsistenz ist die sog. Substantiv-Verb-Interaktion [91]. Diese Maßnahme schreibt vor, dass beim Setzen der Parameter einer Applikations-Funktion zuerst das Objekt gesetzt wird und dann die Operation. Diese Maßnahme ist wie folgt begründet: Grundsätzlich wird angenommen, dass eine Funktion von der Applikation unmittelbar ausgeführt wird, wenn die beiden Parameter „Objekt“ und „Operation“ vorliegen. Wird bei einer Verb-Substantiv-Interaktion eine Operation gesetzt, dann ändert sich der Modus der Aktion „Objekt auswählen“ dahingehend, dass wenn ein Objekt ausgewählt wird, die Operation ausgeführt wird. Wenn der Nutzer sich nicht bewusst ist, dass eine Operation gesetzt ist und er wählt ein Objekt aus, wird fälschlicherweise eine Operation ausgeführt – es wird also eine weitere Möglichkeit für den Nutzer geschaffen, einen Modusfehler zu begehen.

3.4.5 Fehlertoleranz

> *Auch wenn der Nutzer fehlerhafte Eingaben tätigt, soll das geplante Arbeitsergebnis mit dem kleinstmöglichen Korrekturaufwand zu bewerkstelligen sein.*

Zentral für die Fehlertoleranz ist die Fehlervermeidung: das Gerät soll, wann immer möglich, verhindern, dass die Eingaben des Nutzers zu einem Fehler führen. Wenn eine fehlerhafte Eingabe getätigt wurde, soll das Gerät eine einfache, konstruktive und konkrete Anweisung geben, wie weiter zu verfahren ist. Zum Fehlermanagement gehört das Rückgängig machen von bereits getätigten Aktionen: Wann immer möglich, soll das Gerät dies ermöglichen. Denn das hat zur Folge, dass die Sorge des Nutzers einen Fehler zu machen reduziert wird und der Nutzer ermutigt wird, auch unbekannte Optionen auszuführen [102].

3.4.6 Individualisierbarkeit

> *Dem Nutzer soll die Möglichkeit geboten werden, die Darstellung der Informationen an seine Belange anzupassen.*

Dieser Grundsatz trägt der Tatsache Rechnung, dass die Nutzer eines interaktiven Geräts in der Regel unterschiedliche Anforderungen an das Gerät stellen. Deshalb ist es erforderlich, dass das Gerät eine Anpassung – innerhalb bestimmter Grenzen – des Verhaltens erlaubt. Die Begrenzung der Anpassung ist notwendig, um zu verhindern,

[8] Dialogstrang.
[9] Gruppe von Dialogsträngen.
[10] Dialoge.

dass durch Änderungen des Nutzers Beeinträchtigungen entstehen. Der Nutzer soll die Möglichkeit haben, Parameter, welche die Geschwindigkeit von Abläufen bestimmen, an seine persönlichen Anforderungen anzupassen.

3.4.7 Lernförderlichkeit

> *Der Nutzer soll vom Gerät bei dem Lernen der Bedienung unterstützt und geleitet werden.*

Das Prinzip der *Abgeschlossenheit* ist für die Unterstützung des Nutzers von zentraler Bedeutung [99]. Dieses Prinzip fordert eine Struktur der Dialoge, d. h. die Aktionen des Nutzer sollen in zusammenhängende Einheiten gefasst werden; diese Einheiten sollen einen Beginn, einen Mittelteil und ein Ende aufweisen und die Art und Weise, wie das Gerät auf die Eingaben des Nutzers antwortet, soll dieser Struktur entsprechen.

Ferner ist es für die Unterstützung des Nutzers hilfreich, sein Arbeitsgedächtnis zu entlasten: die Menge an Informationen, die der Nutzer in seinem Arbeitsgedächtnis halten muss, soll so klein wie möglich sein. Dazu sollen die Anzeigen einfach gestaltet sein und wichtige Informationen durchgängig und zusammengefasst zur Verfügung stellen.

Der Grundsatz Lernförderlichkeit schließt mit ein, ob das Gerät die Bildung von Gewohnheiten erlaubt bzw. fördert [91]. Gewohnheiten sind automatisierte Prozesse und diese benötigen deutlich weniger kognitive Ressourcen als kontrollierte Prozesse.

Die Konsistenz der Bedienlogik trägt entscheidend zur Lernförderlichkeit des Geräts bei. Ein konsistentes Verhalten erlaubt es dem Nutzer, durch einen Transfer bereits gelernte Konzepte in einer neuen Situation anzuwenden.

3.5 Zusammenfassung

In diesem Kapitel wurde ein konzeptuelles Modell für die Strukturierung der Interaktion beschrieben. Das Kernprinzip ist die Unterteilung der Nutzerschnittstelle in Dialogstränge, die eine Menge an Dialogen verwalten. Die Interaktion mit einem Dialogstrang ist strukturiert durch den Lebenszyklus der Dialoge, die diesem Dialogstrang zugeordnet sind. Der Wechsel zwischen den Dialogsträngen erfolgt dadurch, dass dem aktiven Dialog der Fokus entzogen wird und dem aktiven Dialog eines anderen Dialogstrangs übergeben wird. Dieses Konzept ist Vergleichbar mit der Bearbeitung von Prozessen. Der Unterschied ist, dass bei den Dialogsträngen der Nutzer in der Regel bestimmt, welcher Strang den Fokus besitzt bzw. unterbrochen ist.

Ferner wurde ausgehend von den kognitiven Fähigkeiten des Menschen, die sieben Grundsätze der Dialoggestaltung aus der Norm ISO 9241-110 [7] angegeben. Diese

Grundsätze wurden jeweils unter Verwendung relevanter Literatur und dem konzeptuellen Modell erklärt. Das konzeptuelle Modell erleichtert die Erklärung der Grundsätze, da eine klare Terminologie herangezogen werden kann.

4 Modellbasierte Analyse von Nutzerschnittstellen

Dieses Kapitel beschreibt eine modellbasierte Methode zum Entwurf von Analysemodellen interaktiver Bedienkonzepte. Mit Hilfe dieser Analysemodelle ist eine frühe Sicherung der Gebrauchstauglichkeit möglich. Das Fundament bildet ein formales, zustandsbasiertes Modell des gesamten Mensch-Maschine-Systems. Ein Mensch-Maschine-System wird als verteiltes, interaktives System aufgefasst, d. h. eigenständige Einheiten kommunizieren miteinander, um gemeinsam Aufgaben zu erledigen. Das Modell beschreibt die Kommunikation zwischen den Komponenten „Nutzer“, „Mediator“ und „Applikation“. Neben der formalen Analyse ermöglicht diese Modellbildung die Anwendung weiterer Techniken zur Bewertung der Gebrauchstauglichkeit und bietet somit eine Grundlage für die effektive Integration dieser Techniken in den Entwicklungsprozess.[1] Der Schwerpunkt liegt in der Beschreibung des Interaktionsverhaltens; die konkrete Gestaltung der Darstellung steht nicht im Zentrum dieser Arbeit.

Zu Beginn des Kapitels wird die Notation, in der die Analysemodelle verfasst werden, beschrieben. Insbesondere wird dargestellt, wie die Konzepte der menübasierten Dialogführung anhand dieser Notation modelliert werden. Anschließend werden Merkmale der Analysemodelle beschrieben und zum Teil mit Hilfe der Temporalen Logik formalisiert. Auf Basis dieser Merkmale werden Prüfkriterien aus den allgemeinen Grundsätze der Dialoggestaltung abgeleitet. Den Abschluss bildet ein Fallbeispiel, ein Bedienkonzept eines MP3-Spielers, zur Veranschaulichung der beschriebenen Methode.

4.1 Erstellung der Analysemodelle

Dieser Abschnitt beschreibt die Erstellung der Analysemodelle. Dazu wird das konzeptuelle Modell (s. Abs. 3.2) in die Notation der Zustandsübergangsdiagramme (s. Abs. 2.2) integriert.

Das Gesamtsystem wird durch einen Und-Zustand dargestellt. Die orthogonalen Unterzustände des Gesamtsystems sind dem Nutzer, dem Mediator oder der Applikation zugeordnet.

[1] Es besteht zum Beispiel die Möglichkeit die Analysemodelle dahingehend zu verfeinern, dass graphische Prototypen erstellt werden können. Diese Prototypen können dann mit Nutzertests empirisch evaluiert werden.

4.1.1 Nutzer

Die Bedienaktionen des Nutzers werden durch nicht-deterministische Transitionen modelliert. Für jede Rückmeldung des Mediators sind entsprechenden Reaktionen angegeben. Eine Rückmeldung besteht u. a. aus einem Indikator für eine Modi-Konfiguration (s. Abs. 3.2). Durch diesen Indikator wird dem Nutzer genau mitgeteilt welche Bedienaktionen im nächsten Schritt möglich sind und welchen Effekt eine Bedienaktion auslöst. Es werden diejenigen Bedienaktionen, die für die empfangene Modi-Konfiguration erlaubt sind, nichtdeterministisch festgelegt.

Eine Rückmeldung mit dem Indikator Eins stellt einen speziellen Fall dar. Sie teilt dem Nutzer mit, dass im folgenden Dialogschritt keine Bedienaktion einen Effekt bewirkt. Der Nutzer reagiert in einem solchen Fall mit einer Timeout-Bedienaktion.

4.1.2 Mediator

Der orthogonale Zustand, der den Mediator repräsentiert, ist ein zusammengesetzter Und-Zustand. Abb. 4.1 zeigt die Konstrukte zur Strukturierung des Mediators. Diese sind im Folgenden beschrieben.

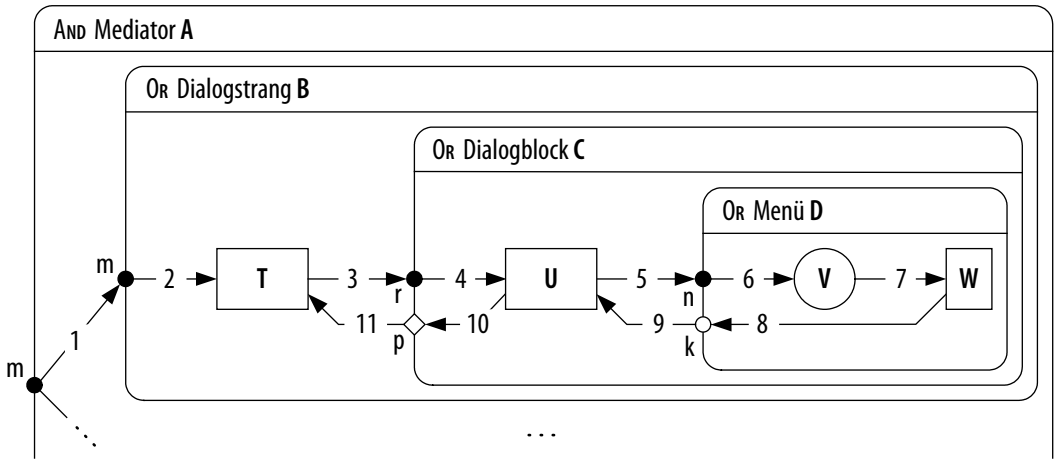


Abbildung 4.1: Konstrukte zur Strukturierung des Mediators

Dialogstränge. Die orthogonalen Unterzustände des Mediators stellen typischerweise Dialogstränge dar. Der Ablauf der Dialogstränge wird koordiniert durch das Muster „quasiparallele orthogonale Zustände“ (s. Abs. 2.2). Die konzeptuelle Sperre „Fokus“ wird durch die Variablen mit der Bezeichnung „running“ modelliert. Zur Erinnerung: Ist die boolesche Variable „running“ eines orthogonalen Zustands gesetzt, spricht man davon, dass dieser orthogonale Zustand die Kontrolle besitzt. Der aktive Dialog eines Dialogstrangs S befindet sich im Zustand „fokussiert“, wenn der orthogonale Zustand, der S modelliert, die Kontrolle besitzt. Der aktive Dialog von S befindet sich im Zustand „bereit“, wenn der orthogonale Zustand, der S modelliert,

nicht die Kontrolle besitzt. Ein Sprung zwischen zwei Dialogsträngen wird mit Hilfe der Aktion „resume“ durchgeführt.

Dialogblöcke. Das Öffnen und Schließen bzw. Abbrechen von Dialogen, wird durch *Dialogblöcke* modelliert. Ein Dialogblock ist ein zusammengesetzter Zustand, dessen Aktivierung das Öffnen einer Menge an Dialogen darstellt. Die Deaktivierung eines Dialogblocks stellt das Schließen bzw. Abbrechen der Dialoge dar. Eine Stelle eines Dialogblocks kann als *Abbruchsstelle* markiert werden. Abbruchsstellen werden als Diamanten dargestellt. Im Beispiel ist die Stelle „p“ eine Abbruchsstelle. Das Deaktivieren des Dialogblocks über diese Stelle bedeutet das Abbrechen der Dialoge des Blocks. Die Variablen eines Dialogblocks modellieren die Attribute der Dialoge. Die Tatsache, dass die Variablen ihre Belegungen verlieren, wenn der Dialogblock deaktiviert wird, entspricht dem konzeptuellen Verhalten beim Schließen bzw. beim Abbrechen eines Dialogs.

Menüs. Ein Menü ist ein zusammengesetzter Zustand, dessen Aktivierung das Überführen eines Dialogs in den Zustand „aktiv“ bedeutet. Typischerweise verweist der Name des Zustands auf einen Dialog. Das heißt, wird der Zustand aktiviert, wird konzeptuell der zugeordnete Dialog auf den Stapel gelegt; wird der Zustand deaktiviert, wird der Dialog vom Stapel entfernt. Der konzeptuelle Dialog-Stapel wird somit durch die verschachtelten Menü-Zustände repräsentiert.

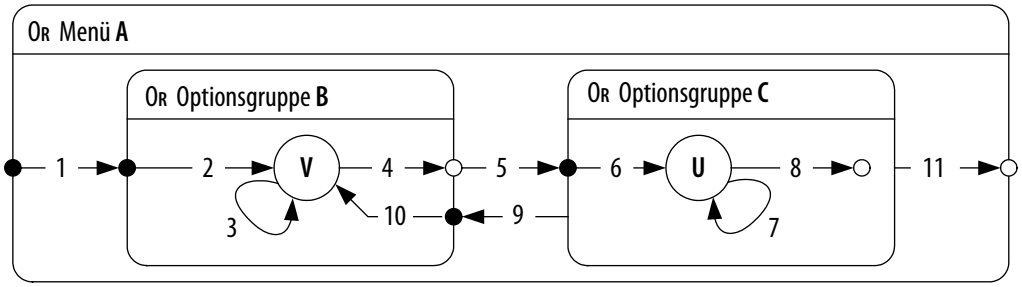


Abbildung 4.2: Konstrukt „Optionsgruppe“

Optionsgruppen. Abb. 4.2 zeigt das Konstrukt „Optionsgruppe“. Eine Optionsgruppe ist ein spezielles Menü und wird durch einen zusammengesetzten Zustand modelliert. Dieser zusammengesetzte Zustand ist ein Unterzustand eines Menüs, das den Zustand „fokussiert“ nicht einnehmen kann. Das Aktivieren einer Optionsgruppe repräsentiert das Überführen des zugeordneten Dialogs in den Zustand „aktiv“. Eine zusammengesetzte Transition, die zwei einfache Zustände zweier unterschiedlicher Optionsgruppen miteinander verbindet, modelliert einen Tabulator-Sprung.

Prompts. Ein einfacher Zustand mit der Eigenschaft, dass sein Vater ein Menü oder eine Optionsgruppe darstellt und dass jede eingehende zusammengesetzte Transition eine Rückmeldung verschickt, wird als Prompt bezeichnet. Prompts werden im

Diagramm als Kreise dargestellt. Im Beispiel stellt der Zustand „V“ einen Prompt dar.

Intermediates. Einfache Zustände, die dem Mediator untergeordnet sind und deren eingehenden zusammengesetzten Transitionen jeweils keine Rückmeldung verschicken, werden als *Intermediates* bezeichnet. Intermediates werden im Diagramm durch Rechtecke dargestellt. In Abb. 4.1 sind die Zustände „T“, „U“ und „W“ Intermediates. Ein Intermediate wird beispielsweise als Endzustand für eine Funktionsansteuerung der Applikation verwendet.

4.1.3 Applikation

Die orthogonalen Zustände, die der Applikation zugeordnet sind, modellieren die Funktionalität der Applikation. Die Funktionalität der Applikation wird aus den Aufgaben des Nutzers abgeleitet. Die Nachrichten, die der Mediator an die Applikation schickt, sind gemäß des konzeptuellen Modells klassifiziert. Es wird beispielsweise bei einer Nachricht, die einen Parameter setzt, unterschieden, ob dieser Parameter ein *Objekt* oder eine *Operation* darstellt. Außerdem wird festgestellt, ob der Effekt der Ansteuerung wieder rückgängig gemacht werden kann und ob der Effekt potentiell einen Schaden verursachen kann.

4.1.4 Kommunikationskanäle und Monitorvariablen

Dieser Abschnitt beschreibt die Kommunikationskanäle der orthogonalen Zustände. Ferner wird beschrieben, welche Monitorvariablen zum Einsatz kommen.

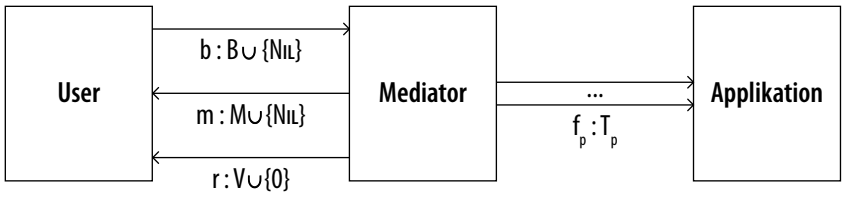


Abbildung 4.3: Kommunikationskanäle

Kommunikationskanäle. Die Kommunikationskanäle sind in Abb. 4.3 dargestellt. Der Nutzer signalisiert über die Variable b das Durchführen einer Bedienaktion. Die Menge aller Bedienaktionen wird mit B bezeichnet. Die Wertemenge der Variable b ist $B \cup \{\text{Nil}\}$. Der Mediator signalisiert seine Rückmeldung über die Variablen r und m. Die Rückmeldung besteht aus einem Indikator (r), der die Modi-Konfiguration anzeigt, und einem Bezeichner eines Menüfeldes (m) desjenigen Dialogs, der sich im Zustand „fokussiert“ befindet. Die Menge aller Modi-Konfigurationen wird mit V bezeichnet, wobei V eine endliche Teilmenge von $\mathbb{N}$ ist. Die Wertemenge der Variable

r ist $V \cup \{0\}$. Die Menge aller Menüfelder wird mit M bezeichnet. Die Wertemenge der Variable m ist $M \cup \{\text{NIL}\}$. Der Mediator steuert die Funktionen der Applikation anhand einer Reihe von Variablen. Die Menge aller Ansteuerungskanäle wird mit F bezeichnet.

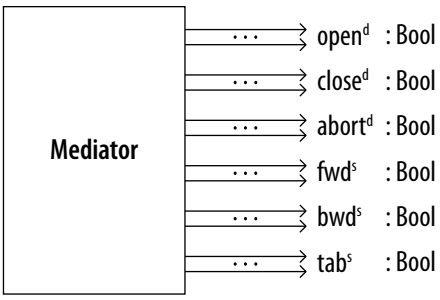


Abbildung 4.4: Monitorvariablen

Monitorvariablen. Abb. 4.4 zeigt die verwendeten Monitorvariablen. Die Monitorvariablen werden vom Mediator bei der Durchführung eines Dialogschritts mit den entsprechenden Werten belegt. Im Folgenden sei $d \in D$ und $s \in S$, wobei D die Menge der Dialogblöcke bezeichnet und S die Menge der orthogonalen Zustände des Mediators. Zur Beschreibung der Modi-Konfigurationen, werden die booleschen Variablen open^d, close^d, abort^d, fwd^s, bwd^s und tab^s verwendet.

Wenn ein Dialogblock d aktiviert wird, dann wird die Variable open^d gesetzt. Wenn ein Dialogblock d geschlossen wird, d. h. der Dialogblock wird über eine normale Stelle deaktiviert, dann wird die Variable close^d gesetzt. Wenn ein Dialogblock d abgebrochen wird, d. h. der Dialogblock wird über eine Abbruchsstelle deaktiviert, dann wird die Variable abort^d gesetzt. Das Ereignis, dass ein Dialogblock d entweder geschlossen oder abgebrochen wird, wird mit $\mathsf{ca}^d = \mathsf{close}^d \vee \mathsf{abort}^d$ bezeichnet. Wenn in einem orthogonalen Zustand s ein Menü aktiviert wird, wird die Variable fwd^s gesetzt. Wenn in einem orthogonalen Zustand s ein Menü deaktiviert wird, wird die Variable bwd^s gesetzt. Wenn in einem orthogonalen Zustand s eine Optionsgruppe deaktiviert und eine andere Optionsgruppe aktiviert wird, wird die Variable tab^s gesetzt.

4.2 Merkmale der Analysemodelle

In diesem Abschnitt sind Merkmale, also kennzeichnende Eigenschaften, der Analysemodelle beschrieben. Der Zusammenhang dieser Merkmale zu den Dialog-Grundsätzen wird im nächsten Abschnitt hergestellt. Für einen Teil der Merkmale sind temporallogische Formeln, die geeignet sind, Ausprägungen der Merkmale festzulegen, angegeben. Somit besteht die Möglichkeit, die Modelle mit Hilfe eines Model-Checkers automatisch auf bestimmte Merkmalsausprägungen zu prüfen.

Die Merkmale, die in dieser Arbeit im Zentrum stehen, sind in die fünf Bereiche

1. Struktur der Dialoge,
2. Voreingestellte Belegung der Dialogattribute,
3. Voreingestellte Fortsetzung der Dialoge,
4. Zusammenspiel der Dialoge mit der Applikation und
5. Modi-Konfigurationen

unterteilt und werden im Folgenden beschrieben.

4.2.1 Struktur der Dialoge

Dieser Bereich umfasst die Merkmale der Dialogstruktur. Dazu gehört die Art und Weise wie der Mediator in Dialogstränge unterteilt ist. Diese Aufteilung zeigt, welche Dialoge als nebenläufig aufgefasst werden und welche Möglichkeiten es gibt, zwischen den Dialogsträngen zu springen. Bei den Sprüngen zwischen Dialogsträngen ist einerseits zu beachten, welche Bedienaktionen solche Wechsel auslösen und andererseits, welchen Wiederaufnahmezustand der unterbrochene Dialog einnimmt. Es ist deshalb zu unterscheiden, ob ein Dialog, wenn er den Fokus verliert, abgebrochen bzw. geschlossen wird oder unverändert bleibt. Ferner ist zu unterscheiden, ob die Belegung der Dialogattribute geändert wird oder nicht.

Tidwell unterscheidet in [102] zwischen einem freien Wechsel der Dialogstränge und einem Muster, das als *Hub-and-Spoke* bezeichnet wird. In diesem Muster wird ein Dialogstrang als Hauptstrang, der Hub, festgelegt; die restlichen Dialogstränge werden als Spokes bezeichnet. Die Navigation wird so eingeschränkt, dass nur Sprünge zwischen dem Hub und den Spokes erlaubt sind – nicht aber zwischen den Spokes.

Weitere Merkmale ergeben sich aus der Verschachtelung der Dialoge innerhalb eines Dialogstrangs. Dazu werden die Dialoge unterteilt in Übergangs-, Einstell- und Verweilmenüs (s. Abs. 3.2.3). Besonders der Typ des Anfangsdialogs und der Typ der Blätter charakterisieren die Dialogstruktur.

Die Sprünge zwischen den Dialogen können als Graph dargestellt werden. Die Menge der Knoten ist durch die Menge M, die Menge aller Dialogfeld-Bezeichner, gegeben. Die Kanten repräsentieren mögliche Sprünge. Die Relation $R : M \times M$ ist durch die folgende temporallogische Formel gegeben: Für alle $x, y \in M$ gilt

$$(x, y) \in R \;\Leftrightarrow\; \neg\,\mathsf{AG}\,(\,m = x \;\rightarrow\; \mathsf{AX}\,(\,m = \textsc{Nil} \wedge \neg\,\mathsf{E}\,[\,m = \textsc{Nil}\;\mathsf{U}\;m = y\,]\,)).$$

4.2.2 Voreingestellte Belegung der Dialogattribute.

Wenn ein Dialog geöffnet wird, sind seine Attribute mit einem initialen Wert zu belegen. Diese initiale Belegung wird als *voreingestellte* Belegung bezeichnet, und kann – in Abhängigkeit zum gegenwärtigen Kontext – unterschiedlich ausfallen. Deshalb

ist zuerst zu unterscheiden, ob die Voreinstellung vom Kontext abhängig oder unabhängig ist. Da ein zusammengesetzter Zustand, der einen Dialogblock darstellt, die Attribute der Dialoge deklariert, ist die Voreinstellung explizit an den Eintrittstransitionen zu erkennen. Desweiteren ist festzustellen, ob bei den Zustandsübergängen „aktivieren“ oder „setzen des Fokus“ (s. Abs. 3.2.2) die Belegungen der Dialog-Attribute zurückgesetzt oder anderweitig geändert werden. Für einen Dialog ist somit kennzeichnend, auf welche Art und Weise seine Attribute voreingestellt und im weiteren Verlauf geändert werden.

4.2.3 Voreingestellte Fortsetzung der Dialoge

Das Analysemodell legt fest, welcher Dialog zu aktivieren ist, wenn ein Dialog geschlossen bzw. abgebrochen wird. Dieser Dialog wird als *Nachfolgedialog* bezeichnet. Das Fortsetzen des übergeordneten Dialogs, wenn ein untergeordneter Dialog geschlossen wird, ist typisch; es besteht aber auch die Möglichkeiten, den geschlossenen Dialog erneut zu öffnen und zu aktivieren oder mehrere neue Dialoge zu öffnen und einen davon zu aktivieren. Die Auswahl des Nachfolgedialogs, wird als die *voreingestellte Fortsetzung* des Dialogs bezeichnet. Die Beschriftungen der Schaltflächen stellen für den Nutzer Indikatoren dar, um zu bestimmen, welcher Nachfolgedialog zu erwarten ist.

4.2.4 Zusammenspiel der Dialoge mit der Applikation

Damit das Zusammenspiel der Dialoge mit der Applikation bestimmt werden kann, sind für die Dialoge Eintritts- und Austrittsbedingungen, die jeweils einer Aufgabe zugeordnet sind, festzulegen. Das heißt, für einen Dialog d wird eine Menge an Aufgaben definiert; diese Menge wird mit T_d bezeichnet. Eine Aufgabe t besteht aus den drei temporallogischen Formeln p^t, q^t und z^t, wobei

- p die Eintrittsbedingung des Dialogs,
- q die Bedingung, die gelten muss, wenn der Dialog geschlossen wird, und
- z die Bedingung, die gelten muss, wenn der Dialog abgebrochen wird,

darstellt. Es sind die folgenden Eigenschaften mit Hilfe des Model-Checkers zu prüfen:

1. Wenn p erfüllt wird, dann gibt es einen Ablauf, der an einer Stelle das entsprechende q erfüllt:
$$\bigwedge_{d \in D} \bigwedge_{t \in T_d} \mathsf{AG}\,(\,p^t \;\rightarrow\; \mathsf{EF}\,q^t\,).$$
2. Wenn ein Dialog geöffnet wird, dann erfüllt dieser Zustand mindestens ein p der Dialog-Aufgaben:
$$\bigwedge_{d \in D} \mathsf{AG}\,(\,\mathsf{open}^d \;\rightarrow\; \bigvee_{t \in T_d} p^t\,).$$

3. Wird ein Dialog in einem Zustand geöffnet, der ein p der Dialog-Aufgaben erfüllt, dann erfüllt dieser Zustand bei einem Abbruch des Dialogs das entsprechende z:

$$\bigwedge_{d \in D} \bigwedge_{t \in T_d} \mathsf{AG}\,(\,\mathsf{open}^d \wedge p^t \;\rightarrow\; \neg\,\mathsf{E}\,[\,\neg\mathsf{ca}^d \;\mathsf{U}\; \mathsf{abort}^d \wedge \neg z^t\,]\,).$$

4. Wird ein Dialog in einem Zustand geöffnet, der ein p der Dialog-Aufgaben erfüllt, dann erfüllt der Zustand der Schließung das entsprechende q:

$$\bigwedge_{d \in D} \bigwedge_{t \in T_d} \mathsf{AG}\,(\,\mathsf{open}^d \wedge p^t \;\rightarrow\; \neg\,\mathsf{E}\,[\,\neg\mathsf{ca}^d \;\mathsf{U}\; \mathsf{close}^d \wedge \neg q^t\,]\,).$$

Ein Dialog kann entweder geschlossen oder abgebrochen werden. Der Nutzer bricht einen Dialog ab, wenn er feststellt, dass er die Aufgaben, die mit dem Dialog verbunden sind, nicht bearbeiten möchte.

Für einen Dialog ist kennzeichnend, ob der Mediator dem Nutzer die Möglichkeit bietet, einen geöffneten Dialog abzubrechen. Das Prädikat U zeigt für einen Dialog $d \in D$, ob der Dialog, nachdem er geöffnet worden ist, vom Nutzer abgebrochen werden kann:

$$U(d) \;\Leftrightarrow\; \mathsf{AG}\,(\,\mathsf{open}^d \;\rightarrow\; \mathsf{A}\,[\,r \neq 0 \wedge r \neq 1 \;\rightarrow\; \mathsf{E}\,[\,\neg\mathsf{close}^d \;\mathsf{U}\; \mathsf{abort}^d\,]\;\mathsf{W}\;\mathsf{ca}^d\,]\,).$$

Die Formel beschreibt, dass, wenn ein Dialog geöffnet wurde, gilt: Solange der Dialog geöffnet ist, kann der Nutzer, wenn er aufgefordert wird, einen Abbruch herbeiführen.

Ein weiterer Aspekt dieses Bereichs ist die Ansteuerung der Applikation. Die Steuersignale, die Parameter setzen, sind unterteilt in Objekte und in Operationen. Für die Ansteuerung einer Applikationsfunktion ist die Reihenfolge, in der die Parameter gesetzt werden, bezeichnend. Sei so ein Signal, das ein Objekt repräsentiert und sc ein Signal, das eine Operation repräsentiert. Das Signal sf steht für das Ausführen der Funktion. Die folgende Formel beschreibt, dass aus dem Setzen des Operationsparameters folgt, dass bis zur Ausführung der Funktion der Parameter für das Objekt nicht mehr verändert wird.

$$\mathsf{AG}\,(\,sc \;\rightarrow\; \mathsf{A}\,[\,\neg so \;\mathsf{W}\; sf\,]).$$

Festzustellen sind auch diejenigen Aktionen, die eine schädliche oder permanente Ansteuerung auslösen. Außerdem sind die Effekte der impliziten Bedienaktionen im Bezug auf die Ansteuerung der Applikation festzuhalten.

4.2.5 Modi-Konfigurationen

Eine Modi-Konfiguration setzt sich aus einer Reihe an Modus-Beschreibungen zusammen. Eine Modus-Beschreibung legt die Wirkung einer Bedienaktion fest (s. Abs. 3.2.5). Kennzeichnend für das Analysemodell ist die Anzahl der Modi-Konfigurationen. In

Modi-Konfiguration	Bedienaktion	Öffnen (O)	Schließen (C)	Abbruch (A)	Vorwärts (D)	Rückwärts (U)	Tabulator (T)	Sprung (J)	Ansteuerung (F)	Wiederholbar (W)	Stabilität (S)
n	a_0	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet/s$	$\circ/\bullet/f$	$\circ/\bullet$	$\bullet$
	a_1	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet$	$\circ/\bullet/s$	$\circ/\bullet/f$	$\circ/\bullet$	$\bullet$
					...						

Abbildung 4.5: Form der Tabellen zur Beschreibung der Modi-Konfigurationen

dieser Arbeit werden Modi-Konfigurationen mit Hilfe von Tabellen festgelegt. Die Form dieser Tabellen zeigt Abb. 4.5.

Beschreibung der Spalten

Im Folgenden sind die Eigenschaften des Analysemodells, die jeweils durch eine Spalte spezifiziert werden, anhand temporallogischer Formeln angegeben.

In der Tabelle bezeichnet n den Indikator der Modi-Konfiguration und a eine Bedienaktion des Nutzers. Eine Zeile der Tabelle legt einen Modus für eine Bedienaktion fest und ordnet diesen Modus einer Modi-Konfiguration zu. Der Nutzer kann, wenn er auf einen Indikator reagiert, die durch die Modi-Konfiguration definierte Bedienaktionen ausführen:

$$\mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{EX}\,(\,b = a\,)\,).$$

Öffnen. $O^\bullet$ gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n ausgeführt wird, die Bedienaktion a das Öffnen eines Dialog bewirkt. Damit $O^\bullet_k$ erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung ausgeführt wird, die Bedienaktion a das Öffnen eines Dialogs. Alle Prädikate, die mit „k“ indiziert sind, werden später für die Spalte „Stabilität“ benötigt.

$$\begin{aligned}
O^\bullet(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{W}\;\bigvee_{d \in D} \mathsf{open}^d\,]\,)) \\
O^\bullet_k(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,b = \textsc{Nil}\;\mathsf{U}\;(b \neq \textsc{Nil} \wedge (b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{W}\;\bigvee_{d \in D} \mathsf{open}^d\,]\,)\,]\,)).
\end{aligned}$$

O° gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n ausgeführt wird, die Bedienaktion a *nicht* das Öffnen eines Dialog bewirkt. Damit O°_k

erfüllt wird, muss gelten: wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung ausgeführt wird, die Bedienaktion a *nicht* das Öffnen eines Dialogs.

$$
\begin{aligned}
O^{\circ}(n,\, a) \;\Leftrightarrow\; & \mathsf{AG}\,(\, r = n \;\rightarrow\; \mathsf{AX}\,(\, b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\, \bigwedge_{d \in D} \neg \mathsf{open}^{d} \;\mathsf{U}\; (\, \bigwedge_{d \in D} \neg \mathsf{open}^{d}) \;\wedge\; r \neq 0)\,]\,)) \\
O_{k}^{\circ}(n,\, a) \;\Leftrightarrow\; & \mathsf{AG}\,(\, r = n \;\rightarrow\; \mathsf{AX}\,(\, b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\, b = \textsc{Nil} \;\mathsf{U}\; (b \neq \textsc{Nil} \;\wedge\; (b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\, \bigwedge_{d \in D} \neg \mathsf{open}^{d} \;\mathsf{U}\; (\, \bigwedge_{d \in D} \neg \mathsf{open}^{d}) \;\wedge\; r \neq 0)\,]\,))\,]\,)).
\end{aligned}
$$

Exemplarisch wird im Folgenden die temporallogische Formel des Prädikats $O^{\bullet}$ beschrieben. Für alle Pfade gilt: Jede Position erfüllt die folgende Aussage: Aus der Rückmeldung n folgt, dass für alle Pfade gilt: Der nachfolgende Zustand erfüllt die folgende Aussage: Wenn der Nutzer mit der Bedienaktion a antwortet, dann gilt für alle Pfade: Der nachfolgende Zustand erfüllt die folgende Aussage: Solange kein Dialog geöffnet wird, ist der Dialogschritt noch nicht zu Ende, wobei das Aktivieren eines Dialogblocks durch die Variable open^{d} signalisiert wird.

Schließen, Abbruch, Vorwärts und Rückwärts. Die Prädikate der Spalten „Schließen“, „Abbruch“, „Vorwärts“ und „Rückwärts“ werden analog zur Spalte „Öffnen“ gebildet.

Sprung. J^{s} gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n ausgeführt wird, die Bedienaktion a die Rückmeldung des Feldes s bewirkt. Damit J_{k}^{s} erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung ausgeführt wird, die Bedienaktion a die Rückmeldung des Feldes s.

$$
\begin{aligned}
J^{s}(n,\, a) \;\Leftrightarrow\; & \mathsf{AG}\,(\, r = n \;\rightarrow\; \mathsf{AX}\,(\, b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\, m = \textsc{Nil} \;\mathsf{U}\; m = s\,]\,)) \\
J_{k}^{s}(n,\, a) \;\Leftrightarrow\; & \mathsf{AG}\,(\, r = n \;\rightarrow \\
& \qquad \mathsf{AX}\,(\, b = a \;\rightarrow\; \mathsf{AX}\;\mathsf{A}\,[r = 0 \;\mathsf{U}\; r \neq 0 \;\wedge\; \mathsf{AX}\,(\, b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\, m = \textsc{Nil} \;\mathsf{U}\; m = s\,]\,)\,]\,)).
\end{aligned}
$$

$J^{\bullet}$ gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n des Feldes s ausgeführt wird, die Bedienaktion a eine Rückmeldung eines Feldes ungleich s bewirkt. Damit $J_{k}^{\bullet}$ erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion

auf die darauf folgende Rückmeldung des Feldes s ausgeführt wird, die Bedienaktion a eine Rückmeldung eines Feldes ungleich s.

$$
\begin{aligned}
J^{\bullet}(n,\,a) \;\Leftrightarrow\; & \bigwedge_{s \in M} \mathsf{AG}\,(\,r = n \;\wedge\; m = s \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,m = \textsc{Nil}\;\mathsf{U}\;m \neq s \;\wedge\; m \neq \textsc{Nil}\,]\,)) \\
J_k^{\bullet}(n,\,a) \;\Leftrightarrow\; & \bigwedge_{s \in M} \mathsf{AG}\,(\,r = n \;\rightarrow \\
& \quad \mathsf{AX}\,(\,b = a \;\rightarrow\; \mathsf{AX}\;\mathsf{A}\,[r = 0\;\mathsf{U}\;r \neq 0 \;\wedge\; (\,m = s \;\rightarrow \\
& \quad \mathsf{AX}\,(\,b = a \;\rightarrow\; \mathsf{AX}\;\mathsf{A}\,[\,m = \textsc{Nil}\;\mathsf{U}\;m \neq s \;\wedge\; m \neq \textsc{Nil}\,]\,)\,]\,)\,]\,)).
\end{aligned}
$$

J° gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n des Feldes s ausgeführt wird, die Bedienaktion a die Rückmeldung des Feldes s bewirkt. Damit J_k° erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung des Feldes s ausgeführt wird, die Bedienaktion a die Rückmeldung des Feldes s.

$$
\begin{aligned}
J^{\circ}(n,\,a) \;\Leftrightarrow\; & (\,\bigwedge_{s \in M} \mathsf{AG}\,(\,r = n \;\wedge\; m = s \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,m = \textsc{Nil}\;\mathsf{U}\;m = s\,]\,))) \\
J_k^{\circ}(n,\,a) \;\Leftrightarrow\; & (\,\bigwedge_{s \in M} \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{U}\;r \neq 0 \;\wedge\; (\,m = s \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,m = \textsc{Nil}\;\mathsf{U}\;m = s\,]\,)\,]\,))).
\end{aligned}
$$

Ansteuerung. F^f gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n ausgeführt wird, die Bedienaktion a die Ansteuerung f bewirkt. Damit F_k^f erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung ausgeführt wird, die Bedienaktion a die Ansteuerung f.

$$
\begin{aligned}
F^f(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{W}\;\mathsf{fct}^f\,]\,)) \\
F_k^f(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,b = \textsc{Nil}\;\mathsf{U}\;(b \neq \textsc{Nil} \;\wedge\; (b = a \;\rightarrow \\
& \quad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{W}\;\mathsf{fct}^f\,]\,)\,]\,)).
\end{aligned}
$$

$F^{\bullet}$ gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n ausgeführt wird, die Bedienaktion a eine Ansteuerung bewirkt. Damit $F_k^{\bullet}$ erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung

ausgeführt wird, die Bedienaktion a eine Ansteuerung.

$$
\begin{aligned}
F^{\bullet}(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{W}\;\bigvee_{f \in F} \mathsf{fct}^{f}\,]\,)) \\
F_k^{\bullet}(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,b = \textsc{Nil}\;\mathsf{U}\;(b \neq \textsc{Nil} \wedge (b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{W}\;\bigvee_{f \in F} \mathsf{fct}^{f}\,]\,))\,]\,)).
\end{aligned}
$$

F° gilt, wenn in jedem Dialogschritt, der als Reaktion auf die Rückmeldung n ausgeführt wird, die Bedienaktion a *nicht* eine Ansteuerung bewirkt. Damit F_k° erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung ausgeführt wird, die Bedienaktion a *nicht* eine Ansteuerung.

$$
\begin{aligned}
F^{\circ}(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,\bigwedge_{f \in F} \neg\mathsf{fct}^{f}\;\mathsf{U}\;(\,\bigwedge_{f \in F} \neg\mathsf{fct}^{f}) \wedge r \neq 0)\,]\,)) \\
F_k^{\circ}(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,b = \textsc{Nil}\;\mathsf{U}\;(b \neq \textsc{Nil} \wedge (b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,\bigwedge_{f \in F} \neg\mathsf{fct}^{f}\;\mathsf{U}\;(\,\bigwedge_{f \in F} \neg\mathsf{fct}^{f}) \wedge r \neq 0)\,]\,))]\,)).
\end{aligned}
$$

Wiederholbar. Damit $W^{\bullet}$ erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, kann er die darauf folgende Aufforderung ebenfalls mit der Bedienaktion a beantworten.

Damit W° erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, ist in der darauf folgende Aufforderung die Bedienaktion a nicht möglich, d. h. die Bedienaktion a kann nicht erneut ausgeführt werden. Wenn eine Bedienaktion für eine Modi-Konfiguration nicht definiert ist, stellt dies den Fall dar, dass die Bedienaktion keine Wirkung hat. Eine Bedienaktion, die keine Wirkung hat, wird nicht als fehlerhafte Eingabe aufgefasst. Beispiele für ein solches Verhalten sind das Drücken der „Weiter“-Taste am Ende einer Liste oder das Drücken der „Home“-Taste im Anfangsmenü.

$$
\begin{aligned}
W^{\bullet}(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{U}\;(\,r \neq 0 \wedge (\,r \neq 1 \;\rightarrow\; \mathsf{EX}\,(\,b = a\,))\,)\,]\,)) \\
W^{\circ}(n,\,a) \;\Leftrightarrow\; & \mathsf{AG}\,(\,r = n \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow \\
& \qquad \mathsf{AX}\;\mathsf{A}\,[\,r = 0\;\mathsf{U}\;(\,r \neq 0 \wedge \mathsf{AX}\,(\,b \neq a\,))\,]\,)).
\end{aligned}
$$

Stabilität. Damit $S^{\bullet}$ erfüllt wird, muss gelten: Wenn der Nutzer die Aufforderung n mit der Bedienaktion a beantwortet, bewirkt in jedem Dialogschritt, der als Reaktion auf die darauf folgende Rückmeldung ausgeführt wird, die Bedienaktion a den gleichen Effekt, der in der Zeile festgelegt ist. Das heißt, die mit „k“ indizierten Prädikate sind erfüllt. Intuitiv bedeutet diese Eigenschaft, dass sich die Wirkung einer Bedienaktion bei einer erneuten Ausführung nicht ändert.

Ist eine Bedienaktion nicht stabil, bedeutet das, dass sich nach der Ausführung der Bedienaktion der Modus dieser Bedienaktion ändert, d. h. ein sog. Modus-Wechsel vollzogen wird. Insbesondere die Modus-Wechsel dieser Art sind kennzeichnend für ein Analysemodell.

Visualisierung der Übergänge zwischen den Modi-Konfigurationen

Die Übergänge zwischen den Modi-Konfigurationen können durch einen Graphen dargestellt werden. Die Menge der Knoten ist durch die Menge der Modi-Konfigurationen V gegeben. Die Kanten repräsentieren mögliche Übergänge. Die Relation $R : V \times V$ kann durch die folgende temporallogische Formel angegeben werden: Für alle x, $y \in V$ gilt

$$(x, y) \in R \;\Leftrightarrow\; \neg\,\mathsf{AG}\,(\,r = x \;\rightarrow\; \mathsf{AX}\,(\,r = 0 \;\wedge\; \neg\,\mathsf{E}\,[\,r = 0 \;\mathsf{U}\; r = y\,]\,)).$$

Die Übergänge zwischen den Modi-Konfigurationen können mit derjenigen Bedienaktion markiert werden, die diesen Übergang auslöst. Die Relation $R : V \times B \times V$ ist durch die folgende temporallogische Formel gegeben: Für alle x, $y \in V$, $a \in B$ gilt

$$(x, a, y) \in R \;\Leftrightarrow\; \neg\,\mathsf{AG}\,(\,r = x \;\rightarrow\; \mathsf{AX}\,(\,b = a \;\rightarrow\; \neg\,\mathsf{E}\,[\,r = 0 \;\mathsf{U}\; r = y\,]\,)).$$

Zusammenhang der Modi-Konfiguration zu den Anzeigen

Der Zusammenhang der Modi-Konfigurationen zu den Anzeigen ist kennzeichnend für ein Analysemodell. Die Informationen, die dem Nutzer zu einer Modi-Konfiguration angezeigt werden, sind schematisch festzuhalten.

4.3 Bezug der Merkmale zu den Grundsätzen der Dialoggestaltung

In diesem Abschnitt werden die identifizierten Merkmale den Grundsätzen der Dialoggestaltung zugeordnet. Ziel ist zu zeigen, dass diese Merkmale im Zusammenhang mit der Gebrauchstauglichkeit des modellierten Bedienkonzepts stehen.

Zusätzlich werden aus den Grundsätzen der Dialoggestaltung exemplarisch Prüfkriterien für die Merkmale der Analysemodelle abgeleitet und argumentativ begründet. Für konkrete Projekte ist es erforderlich, dass diese Prüfkriterien im jeweiligen Projekt-Umfeld zusätzlich validiert werden. Ferner besteht bei dieser Erhebung kein

Anspruch auf Vollständigkeit. Die Herleitung der Anforderungen soll verdeutlichen, dass anhand der Merkmale der Analysemodelle konkrete, prüfbare Kriterien festgelegt werden können.

Aufgabenangemessenheit

Struktur der Dialoge. Die Struktur, also die Verschachtelung, der Dialoge im Verbund mit dem jeweiligen Typ des Dialogs, steht im Zusammenhang mit der Aufgabenangemessenheit des Systems. Stimmt die Strukturierung mit der Struktur der Aufgaben des Nutzers nicht überein, ist die Effizienz der Aufgabenerledigung beeinträchtigt, da zusätzlicher Navigationsaufwand entsteht. Die Einschränkung der Sprünge zwischen den Dialogen, kann dagegen einen positiven Einfluss auf die Effizienz ausüben: Ein eingeschränktes Menü-Netz verringert ungewollte Sprünge, indem es dem Nutzer weniger Möglichkeiten darbietet und somit auch übersichtlicher ist.

Voreingestellte Belegung der Dialogattribute. Je nachdem wie die Voreinstellung der Dialogattribute ausfällt, kann die Effizienz der Aufgabenerledigung verbessert oder verschlechtert werden. Entspricht die Voreinstellung den Bedürfnissen des Nutzers, werden dadurch die nötigen Bedienaktionen des Nutzers reduziert. Widerspricht die Voreinstellung den Wünschen des Nutzers, so muss dieser Aufwand betreiben, um die gewünschte Belegung herzustellen.

Voreingestellte Fortsetzung der Dialoge. Die Voreinstellung, wie der Gesamtdialog fortgesetzt wird, wenn ein Unterdialog am Ende ist, beeinflusst die Effizienz der Dialogausführung, da – analog zur Voreinstellung der Dialogattribute – ein Zusammenhang zur Aufgabenstruktur des Nutzers gegeben ist.

Zusammenspiel der Dialoge mit der Applikation. Die Funktionalität der Applikation steht in einem engen Bezug zu den Aufgaben des Nutzers. Deshalb ist die geeignete Kopplung der Dialoge mit der Applikation entscheidend für eine effektive und effiziente Durchführung dieser Aufgaben.

Selbstbeschreibungsfähigkeit

Voreingestellte Fortsetzung der Dialoge. Wird bei der Fortsetzung eines Dialogs ein Sprung über mehrere Dialogebenen vollzogen, hat das einen Einfluss auf die Selbstbeschreibungsfähigkeit des Bedienkonzepts: Ist sich der Nutzer über den Sprung sowie über die Struktur der Dialoge nicht im Klaren, dann weiß der Nutzer im Folgedialog nicht mehr, an welcher Stelle im Gesamtdialog er sich befindet.

> [Prüfkriterium 1] Der Nachfolgedialog sollte entweder ein übergeordneter Dialog des Vorgängers sein, oder ein unmittelbarer Unterdialog eines übergeordneten Dialogs.

Da die Beschriftungen der Schaltflächen für den Nutzer als Indikator dafür dienen, welcher Nachfolgedialog aktiviert wird, beeinträchtigt die Aktivierung eines Nachfolgedialogs, der nicht mit der Beschriftungen übereinstimmt, die Selbstbeschreibungsfähigkeit des Bedienkonzepts.

> [Prüfkriterium 2] Der Nachfolgedialog soll aus den Beschriftungen der Schaltflächen hervor gehen.

Modi-Konfigurationen. Die Tatsache, ob die aktuelle Modi-Konfiguration aus der Anzeige ersichtlich ist, hat einen Einfluss auf die Vorhersagbarkeit des Dialogs. Die Vorhersagbarkeit ist ein Prinzip, das die Selbstbeschreibungsfähigkeit eines Dialogs unterstützt, d. h. es geht zu jedem Zeitpunkt hervor, welche Bedienaktionen möglich sind und welche Wirkungen diese haben. Insbesonders können unbemerkte Modus-Übergänge die Selbstbeschreibungsfähigkeit beeinträchtigen. Allgemein gilt: Je komplexer die Modi-Struktur ist und je unklarer die Anzeige diese vermittelt, desto schwieriger wird es für den Nutzer, den gegenwärtigen Zustand des Dialogs intuitiv zu erfassen.

> [Prüfkriterium 3] Die Anzahl der Modi, die einer Bedienaktion zugeordnet sind, sollte so gering wie möglich sein. Es ist zu prüfen, ob die Möglichkeit besteht, Modi durch eine veränderte Gestaltung zu vermeiden.

> [Prüfkriterium 4] Die Wirkung der jeweiligen Bedienaktionen der gegenwärtigen Modi-Konfiguration soll aus der Anzeige klar und deutlich hervor gehen. Das heißt, die Anzeige soll die Modi-Konfiguration widerspiegeln.

Steuerbarkeit

Struktur der Dialoge. Die Art und Weise in der die Interaktion in Dialogstränge zerlegt ist hat einen Einfluss auf die Steuerbarkeit des Systems. Damit der Nutzer den Ablauf und die Geschwindigkeit der Interaktion beeinflussen kann, muss er die Möglichkeit besitzen, einen Dialog zu unterbrechen und diesen zu einem späteren Zeitpunkt wieder aufzunehmen. Das Konzept der Dialogstränge zeigt explizit die Umsetzung des Unterbrechungsverhaltens auf.

> [Prüfkriterium 5] Es soll die Möglichkeit bestehen, zwischen den Dialogsträngen zu wechseln.

> [Prüfkriterium 6] Ein unterbrochener Dialog soll die Belegung seiner Attribute – und damit die bisherige Eingabehistorie des Nutzers – bis zur Wiederaufnahme speichern.

Erwartungskonformität

Voreingestellte Belegung der Dialogattribute. Wenn die Voreinstellungen kontextabhängig gewählt werden, kann das die Erwartungskonformität des Systems beeinträchtigen und zwar in dem Fall, dass das Verhalten des Dialogs nicht die tatsächlichen Erwartungen des Nutzers in dem gegebenen Kontext erfüllt. Nachdem der

Nutzer grundsätzlich ein konsistentes Systemverhalten im Bezug auf die Nutzungssituation erwartet [54], ist auch für die Voreinstellungen Konsistenz gefordert.

> [Prüfkriterium 7] Das Voreinstellen der Attribute soll, in Bezug auf das gesamte Bedienkonzept, konsistent sein.

Zusammenspiel der Dialoge mit der Applikation. Im Zusammenhang zur Erwartungskonformität ist es wichtig, dass bei Dialogen, die abgebrochen werden, keine impliziten Nebenwirkungen bestehen bleiben; denn diese können im weiteren Verlauf des Dialogs zu Verhalten führen, das der Nutzer nicht erwartet.

> [Prüfkriterium 8] Die Eintrittsbedingung eines Dialogs soll zusätzlich beim Abbruch des Dialogs gelten.

Die Reihenfolge in der die Parameter bei der Ansteuerung der Applikationsfunktionen gesetzt werden besitzt einen Einfluss auf die Erwartungskonformität (s. Abs. 3.4.4).

> [Prüfkriterium 9] Nach dem Setzen des Operationsparameters soll der Parameter für das Objekt bis zur Ausführung der Funktion nicht mehr verändert werden.

Modi-Konfigurationen. Die Modi-Konfigurationen haben einen starken Zusammenhang zur Konsistenz des Interaktionsverhalten und somit auch zur Erwartungskonformität. Die ISO 9241-14 [3] empfiehlt: „Wenn die Menüstruktur hierarchisch ist, dann sollte ein einfaches und einheitliches Mittel verfügbar sein, um die nächsthöhere Ebene der Menüstruktur erreichen zu können“.

> [Prüfkriterium 10] Es sollte eine Bedienaktion geben, die in allen Modi-Konfigurationen, in denen sie einen Effekt bewirkt, einen Rückwärts-Sprung auslöst, das heißt, die Eigenschaften „Rückwärts“ und „Sprung“ werden erfüllt und die Eigenschaft „Vorwärts“ wird ausgeschlossen.

Außerdem unterstützt die Eigenschaft „Stabilität“ – wenn man davon ausgeht, dass der Nutzer beim Wiederholen von Bedienaktionen, Kontinuität in Bezug zur Wirkung erwartet – die Erwartungskonformität: Bei einer wiederholten Ausführung einer Bedienaktion bleibt die Wirkung gleich.

Fehlertoleranz

Zusammenspiel der Dialoge mit der Applikation. Das Zusammenspiel der Dialoge mit der Applikation beeinflusst insbesondere die Fehlertoleranz des Systems. Wenn man davon ausgeht, dass ein abgebrochener Dialog die bisherige Ansteuerung rückgängig macht, ist es erforderlich, dass diese Möglichkeit für den Nutzer besteht. Dadurch ist er in der Lage, auf eine einfache Art und Weise, fehlerhafte Eingaben zu korrigieren.

[Prüfkriterium 11] Der Nutzer soll immer die Möglichkeit besitzen, einen geöffneten Dialog abzubrechen.

Lernförderlichkeit

Struktur der Dialoge. Ein eingeschränktes Menü-Netz verringert ungewollte Sprünge, indem es dem Nutzer weniger Möglichkeiten darbietet und somit übersichtlicher ist. Zusätzlich wird der Nutzer darin unterstützt, Gewohnheiten zu bilden [91]. Der Nutzer wird in dem Verständnis einer komplexen Dialogstruktur unterstützt, wenn die einzelnen Dialoge einen klaren Anfang, Mittelteil und Schluss aufweisen [99]. Dies geht aus den Dialogblöcken des Analysemodells hervor, die aufzeigen, zu welchem Zeitpunkt Dialoge geöffnet bzw. geschlossen werden.

Modi-Konfigurationen. Die Eigenschaft „Stabilität“ einer Bedienaktion steht im Zusammenhang zur Lernförderlichkeit. Ein solches Verhalten ermöglicht es dem Nutzer, eine schnelle Folge von Bedienaktionen auszuführen, da er nicht befürchten muss, dass sich die Wirkung der Aktion ändert. Das hat einen positiven Einfluss auf die Lernförderlichkeit des Dialogs, da solche automatisierten Eingabefolgen den Nutzer darin unterstützen, Gewohnheiten zu bilden [91].

4.4 Beispiel „MP3-Spieler“

Dieser Abschnitt beschreibt ein Beispiel, das sich an dem interaktiven Gerät *iPod mini* [14] der Firma Apple orientiert. Es ist zu beachten, dass die Bedienlogik des Beispiels nicht genau mit diesem Gerät übereinstimmt. Das Beispiel dient zur Demonstration der Methode. Deshalb wurde der Umfang so gewählt, dass die Darstellung überschaubar bleibt.

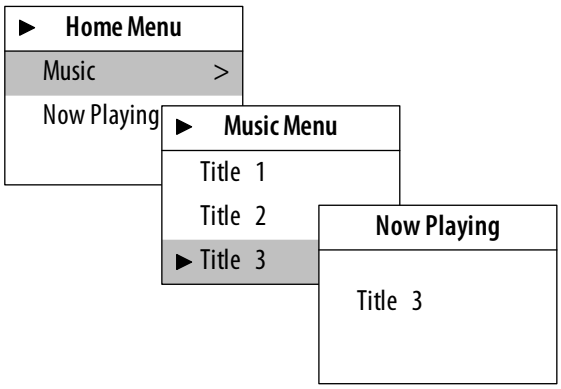


Abbildung 4.6: Anzeigen des Beispiels „MP3-Spieler“

Abb. 4.6 zeigt die Anzeigen des Beispiels. Der gewählte Ausschnitt des Bedienkonzepts ermöglicht dem Nutzer die Wiedergabe von Musiktiteln. Dazu wird zuerst das

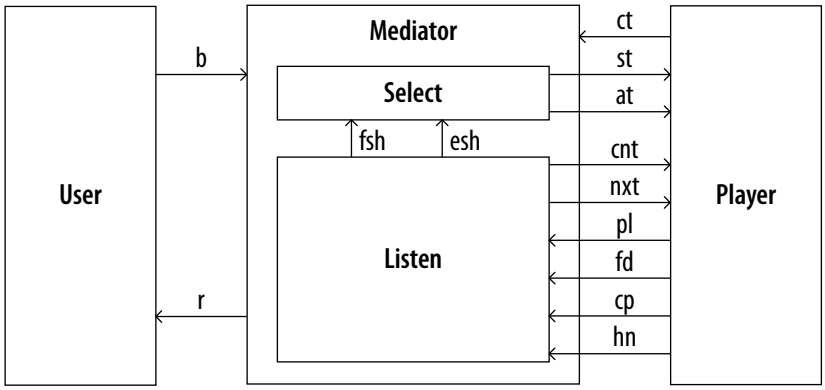


Abbildung 4.7: Struktur des Beispiels „MP3-Spieler“

Gerät durch eine beliebige Bedienaktion eingeschaltet. Der Nutzer kann dann durch Ausführen der Menüoption „Music ›“ zum Untermenü „Music Menu“ springen. Die Auswahl der Menüoptionen erfolgt durch Blättern mit dem sog. *Click Wheel*. Das Ausführen einer Option wird durch Drücken der Taste „Auswählen“ angestoßen.

Das Menü „Music Menu“ zeigt eine Liste von Musiktiteln an. Der Nutzer kann den entsprechenden Titel auswählen und ausführen. Dadurch wechselt er zur Anzeige „Now Playing“. Das Drücken der Taste „Nächster Titel“ ermöglicht es dem Nutzer, den nächsten Titel der Liste abzuspielen; das Drücken der Taste „Menu“ erlaubt das Zurückkehren zum vorherigen Menü. Wenn das Gerät einen Musiktitel wiedergibt, besteht für den Nutzer die Möglichkeit, im Menü „Home Menu“ die Option „Now Playing ›“ zu nutzen, um direkt zur Anzeige „Now Playing“ zu wechseln.

4.4.1 Merkmale des Analysemodells

Für eine detaillierte Beschreibung des Analysemodells sei auf Anhang A.1 verwiesen. Das Analysemodell besteht insgesamt aus 9 zusammengesetzten Zuständen, wobei 4 Zustände orthogonale Zustände sind. Der Mediator besitzt 2 Dialogstränge, die insgesamt 4 Dialogblöcke und 4 Menüs besitzen. Die Anzahl der einfachen Zustände ist 17 mit 4 Prompts und 8 Intermediates. Die Zustände sind durch 148 Transitionen verbunden. Die Kardinalität der Menge der Zustände ist $2^{83,3188}$, von diesen Zuständen sind 4 343 erreichbar.

Abb. 4.7 zeigt die Struktur des Analysemodells, das sich aus den Komponenten „User“, „Mediator“ und „Player“ zusammensetzt. Für eine Beschreibung der Kanäle siehe Anhang A.1. Die Struktur der Dialoge ist im Mediator modelliert. Dieser enthält die zwei Dialogstränge „Select“ und „Listen“. Durch den Einsatz des Musters „quasiparallele orthogonale Zustände“ (s. Abs. 2.2) reagiert in einem Dialogschritt nur einer dieser zwei orthogonalen Zustände auf Eingaben des Nutzers mit einer Rückmeldung. Der orthogonale Zustand, der mit einer Rückmeldung reagiert, ist im Besitz der Kontrolle bzw. im Besitz des Fokus. Auf Grund der Tatsache, dass man

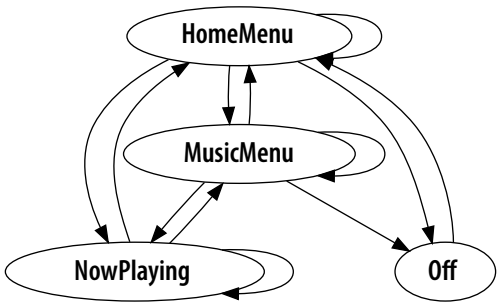


Abbildung 4.8: Übergänge der Anzeigen

den Begriff „Kontrolle“ ebenfalls im Zusammenhang mit der Aktivierung der verschachtelten Zustände verwendet, wird im Folgenden für das Muster ausschließlich der Begriff „Fokus“ eingesetzt.

Im Folgenden werden für die Merkmale, die in Abs. 4.2 definiert sind, die spezifischen Ausprägungen des Beispiels „MP3-Spieler“ beschrieben.

Struktur der Dialoge

Der Mediator ist unterteilt in zwei Dialogstränge. Die Anzeigen „Off“, „HomeMenu“ und „MusicMenu“ werden vom Dialogstrang „Select“ verschickt, die Anzeige „NowPlaying“ vom Dialogstrang „Listen“. Abb. 4.8 zeigt alle Übergänge zwischen den Anzeigen.

Initial besitzt der Dialogstrang „Select“ den Fokus und zeigt die Anzeige „Off“ an. Führt der Nutzer die Bedienaktion Skip[2], Select[3] oder Menu[4] aus, wird zur Anzeige „HomeMenu“ gesprungen. Wenn das Gerät keinen Musiktitel wiedergibt und der Nutzer führt die implizite Bedienaktion Timeout[5] aus, wird zurück zur Anzeige „Off“ gesprungen, d. h. das Gerät wird ausgeschaltet.

In der Anzeige „HomeMenu“ wird bei Durchführung der Bedienaktion Select entweder zur Anzeige „MusicMenu“ (Option „Music ›“) oder zur Anzeige „NowPlaying“ (Option „Now Playing ›“) gesprungen, wobei der Sprung zur Anzeige „NowPlaying“ einen Wechsel des Dialogstrangs darstellt.

In der Anzeige „MusicMenu“ springt der Nutzer bei Ausführung der Bedienaktion Select zur Anzeige „NowPlaying“. Die Bedienaktion Menu veranlasst einen Sprung zurück zur Anzeige „HomeMenu“. Wenn das Gerät keinen Musiktitel wiedergibt und der Nutzer führt die implizite Bedienaktion Timeout aus, wird – analog zur Anzeige „HomeMenu“ – zurück zur Anzeige „Off“ gesprungen.

In der Anzeige „NowPlaying“ kann der Nutzer durch die Bedienaktion Menu zurück zu der Anzeige springen, die zuvor angezeigt wurde. Wenn das Gerät den letzten

[2]Die Bedienaktion „Nächster Titel“.
[3]Das Drücken der Taste „Auswählen“.
[4]Das Drücken der Taste „Menu“.
[5]Modelliert das Verstreichen von Zeit.

Titel der Liste wiedergibt und der Nutzer führt die Bedienaktion SKIP aus, wird zur Anzeige „HomeMenu“ gesprungen. Ist die Wiedergabe des letzten Titels beendet und der Nutzer führt die implizite Bedienaktion TIMEOUT durch, wird ebenfalls zur Anzeige „HomeMenu“ gesprungen.

Voreingestellte Belegung der Dialogattribute

Der Dialog „HomeMenu“ besitzt das Attribut „h“, dessen Belegung anzeigt, welche Option („Music ›“ oder „Now Playing ›“) gegenwärtig ausgewählt ist. Beim Öffnen des Dialogs „HomeMenu“ wird das Attribut mit dem Wert belegt, der beim letzten Schließen bzw. Abbrechen gespeichert wurde – bis auf die folgenden zwei Ausnahmen:

1. Beim Anschalten des Geräts ist die Option „Music ›“ ausgewählt.
2. Wird der Dialog „HomeMenu“ geöffnet, weil der Dialog „NowPlaying“ geschlossen bzw. abgebrochen wurde, ist die Option „Music ›“ ausgewählt.

Der Dialog „MusicMenu“ besitzt das Attribut „o“, dessen Belegung anzeigt, welcher Titel (1–3) gegenwärtig ausgewählt ist. Beim Öffnen des Dialogs „MusicMenu“ wird das Attribut mit dem Wert belegt, der beim letzten Schließen bzw. Abbrechen gespeichert wurde. Analog zum Attribut „h“ gibt es die folgenden zwei Ausnahmen:

1. Beim Anschalten ist die erste Option der Liste ausgewählt.
2. Wird der Dialog „HomeMenu“ geöffnet, weil der Dialog „NowPlaying“ geschlossen bzw. abgebrochen wurde, wird die erste Option der Liste ausgewählt.

Voreingestellte Fortsetzung der Dialoge

Schließt der Nutzer den Dialog „MusicMenu“, wird im selben Dialogschritt der Dialog „HomeMenu“ geschlossen und der Fokus dem Dialogstrang „Listen“ überreicht. Bei der Wiederaufnahme, d. h. der Dialogstrang „Listen“ gibt den Fokus zurück, wird entweder der Dialog „MusicMenu“ oder der Dialog „HomeMenu“ fortgesetzt. Die Bedienaktion MENU bewirkt die Fortsetzung des Dialogs „MusicMenu“, in allen anderen Fällen wird mit dem Dialog „HomeMenu“ fortgesetzt.

Bricht der Nutzer den Dialog „MusicMenu“ durch die Bedienaktion MENU ab, wird der Dialog „HomeMenu“ fortgesetzt. Bricht der Nutzer den Dialog „MusicMenu“ ab, indem er diejenige Option ausführt, deren Titel gegenwärtig wiedergegeben wird, wird im selben Dialogschritt der Dialog „HomeMenu“ abgebrochen und der Fokus dem Dialogstrang „Listen“ überreicht. Die Wiederaufnahme verhält sich analog zum Schließen des Dialogs „MusicMenu“.

Bricht der Nutzer den Dialog „HomeMenu“ ab, indem er die Option „Now Playing ›“ ausführt, wird der Fokus dem Dialogstrang „Listen“ überreicht. Bei der Wiederaufnahme wird der Dialog „HomeMenu“ fortgesetzt.

	Öffnen	**Schließen**	**Abbruch**
HomeMenu			
1	true	MusicMenu = Close	true
MusicMenu			
2	true	$at \neq 0$	$at = 0$
NowPlaying			
3	Player@On	Player@Off	Player@Off

Tabelle 4.1: Zusammenspiel der Dialoge mit der Applikation

Bricht der Nutzer den Dialog „HomeMenu“ bzw. „MusicMenu“ durch die implizite Bedienaktion TIMEOUT ab, wird der Dialog „Off“ fortgesetzt. Beim Schließen des Dialogs „Off“, wird der Dialog „HomeMenu“ fortgesetzt.

Der Nutzer kann den Dialog „NowPlaying“ durch Ausführen der Bedienaktion SKIP abbrechen. Spielt das Gerät den letzten Titel, wird mit dem Dialog „HomeMenu“ fortgesetzt; spielt das Gerät einen anderen Titel, wird der Dialog „NowPlaying“ für den nächsten Titel geöffnet.

Wenn die Wiedergabe eines Titels beendet ist, schließt der Player den Dialog „NowPlaying“ und setzt entweder mit dem Dialog „HomeMenu“ oder mit dem Dialog „NowPlaying“ fort. Es wird mit dem Dialog „HomeMenu“ fortgesetzt, wenn es keinen weiteren Titel in der Liste gibt.

Zusammenspiel der Dialoge mit der Applikation

Zur Beschreibung des Zusammenspiels der Dialoge mit der Applikation, werden für die Dialoge Eintritts- und Austrittsbedingungen festgelegt. Tab. 4.1 stellt die Bedingungen, die für das Beispiel „MP3-Player“ festgelegt wurden, dar. Im Folgenden werden diese Bedingungen beschrieben:

1. Wenn der Dialog „HomeMenu“ geschlossen wird, wird auch der Dialog „MusicMenu“ geschlossen.
2. Wenn der Dialog „MusicMenu“ geschlossen wird, wird der ausgewählte Titel an die Komponente „Player“ verschickt; Wenn der Dialog „MusicMenu“ abgebrochen wird, wird keine Nachricht verschickt.
3. Wenn der Dialog „NowPlaying“ geöffnet wird, ist der Player im Zustand „On“, das bedeutet, es wird zum Zeitpunkt des Öffnens ein Titel wiedergegeben; Wenn der Dialog geschlossen wird, ist der Player im Zustand „Off“; Wenn der Dialog abgebrochen wird, ist der Player im Zustand „On“.

Der Mediator bietet dem Nutzer die Möglichkeit, alle Dialoge – bis auf den Dialog „Off“ – abzubrechen. Die Menge der Signale zur Ansteuerung der Komponente „Player“ umfasst *st*, *at* und *nxt*.

1	Die Wiedergabe des Musiktitels ist beendet
2	Das Gerät ist ausgeschaltet
3	Der Dialog „HomeMenu“ wird angezeigt; Es wird kein Musiktitel wiedergegeben; Die Option „Music›“ ist ausgewählt
4	Der Dialog „HomeMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Es existiert ein weiterer Musiktitel in der Liste; Die Option „Music›“ ist ausgewählt
5	Der Dialog „HomeMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Es existiert kein weiterer Musiktitel in der Liste; Die Option „Music›“ ist ausgewählt
6	Der Dialog „HomeMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Es existiert ein weiterer Musiktitel in der Liste; Die Option „Now Playing›“ ist ausgewählt
7	Der Dialog „HomeMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Es existiert kein weiterer Musiktitel in der Liste; Die Option „Now Playing›“ ist ausgewählt
8	Der Dialog „MusicMenu“ wird angezeigt; Es wird kein Musiktitel wiedergegeben; Die erste Option ist ausgewählt
9	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Die erste Option ist ausgewählt; Der wiedergegebene Titel ist nicht ausgewählt; Es existiert ein weiterer Musiktitel in der Liste
10	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Die erste Option ist ausgewählt; Der wiedergegebene Titel ist nicht ausgewählt; Es existiert kein weiterer Musiktitel in der Liste
11	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Die erste Option ist ausgewählt; Der wiedergegebene Titel ist ausgewählt
12	Der Dialog „MusicMenu“ wird angezeigt; Es wird kein Musiktitel wiedergegeben; Weder die erste noch die letzte Option ist ausgewählt
13	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Weder die erste noch die letzte Option ist ausgewählt; Der wiedergegebene Titel ist nicht ausgewählt; Es existiert ein weiterer Musiktitel in der Liste
14	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Weder die erste noch die letzte Option ist ausgewählt; Der wiedergegebene Titel ist nicht ausgewählt; Es existiert kein weiterer Musiktitel in der Liste
15	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Weder die erste noch die letzte Option ist ausgewählt; Der wiedergegebene Titel ist ausgewählt
16	Der Dialog „MusicMenu“ wird angezeigt; Es wird kein Musiktitel wiedergegeben; Die letzte Option ist ausgewählt
17	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Die letzte Option ist ausgewählt; Der wiedergegebene Titel ist nicht ausgewählt; Es existiert ein weiterer Musiktitel in der Liste
18	Der Dialog „MusicMenu“ wird angezeigt; Es wird ein Musiktitel wiedergegeben; Die letzte Option ist ausgewählt; Der wiedergegebene Titel ist ausgewählt
19	Der Dialog „NowPlaying“ wird angezeigt; Es existiert ein weiterer Musiktitel in der Liste
20	Der Dialog „NowPlaying“ wird angezeigt; Es existiert kein weiterer Musiktitel in der Liste

Tabelle 4.2: Beschreibung der Modi-Konfigurationen

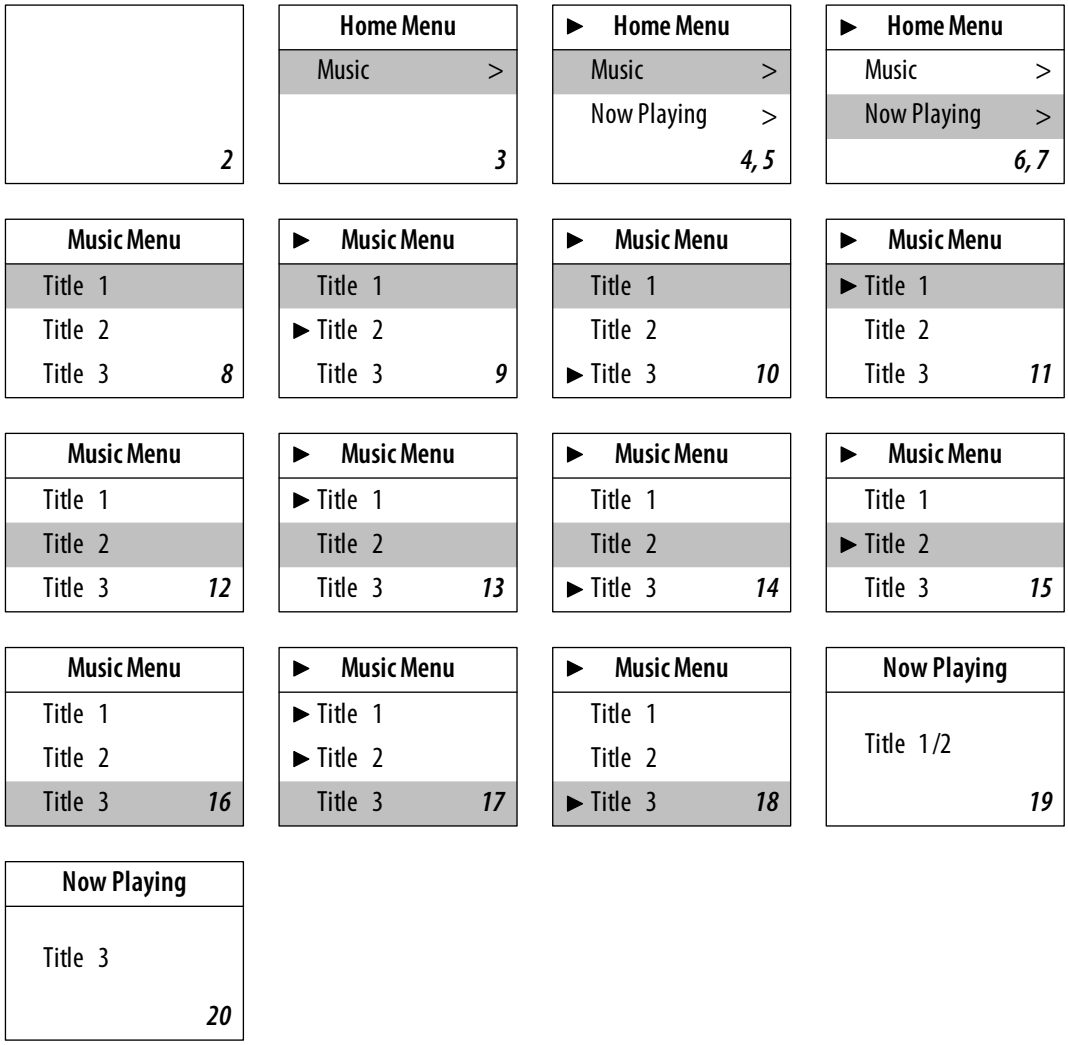


Abbildung 4.9: Zuordnung der Modi-Konfigurationen zu den Anzeigen

Modi-Konfigurationen

Eine Modi-Konfiguration ordnet Bedienaktionen jeweils einem Modus zu. Für das Analysemodell sind 20 Modi-Konfigurationen festgelegt, die in Tab. 4.2 beschrieben sind. Die Festlegung der Modi ist in Anhang A.2 angegeben. Abb. 4.9 ordnet den Modi-Konfigurationen eine entsprechende Anzeige zu. Die Übergänge zwischen den Modi-Konfigurationen sind in Anhang A.3 angegeben.

Tab. 4.3 zeigt eine Zusammenfassung der Modi-Konfigurationen. Das Zeichen “–” bedeutet, dass in allen Modi-Konfigurationen das Nichteintreten des Effekts der Spalte festgelegt worden ist. Eine Zahl sagt aus, in wieviel Prozent der Modi-Konfigurationen das Auftreten des Effekts festgelegt worden ist. Zum Beispiel hat in 93 % der Modi-Konfigurationen die Bedienaktion Skip den Effekt, einen Abbruch eines Dialogs auszulösen.[6]

[6]Die Ausnahme stellt das Anschalten des Geräts dar: der Dialog „Off“ wird durch die Bedienaktion Skip geschlossen.

Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
Skip	66 %	6 %	93 %	66 %	100 %	–	13 %	93 %	53 %	60 %
Timeout	21 %	5 %	84 %	21 %	89 %	–	84 %	–	78 %	21 %
Select	70 %	52 %	58 %	70 %	82 %	–	100 %	47 %	23 %	76 %
Menu	21 %	7 %	78 %	21 %	85 %	–	100 %	–	0 %	85 %
Plus	–	–	–	–	–	–	–	–	40 %	100 %
Minus	–	–	–	–	–	–	–	–	33 %	100 %

Tabelle 4.3: Zusammenfassung der Modi-Konfigurationen: Im Grundwert der Prozentangaben sind diejenigen Konfigurationen enthalten, die für die jeweilige Bedienaktion eine Wirkung definieren

Wird für eine Bedienaktion der Wert 0 % angegeben, so bedeutet das, dass in allen Modi-Konfiguration weder das Auftreten noch das Nichteintreten festgelegt worden ist.

Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
(Plus)	○	○	○	○	○	○	○	○	○	
(Minus)	○	○	○	○	○	○	○	○	○	
(Skip)					●	○			○	
(Select)						○	●		○	
(Timeout)						○		○	○	
(Menu)						○	●	○	○	

Tabelle 4.4: Globale Wirkungen der Bedienaktionen

Aus den Modi-Konfigurationen lässt sich eine Tabelle ableiten, welche die Wirkung einer Bedienaktion unabhängig von der gegenwärtigen Modi-Konfiguration beschreibt. Diese als *global* bezeichneten Wirkungen sind in Tab. 4.4 dargestellt. Diejenigen Bedienaktionen, die nicht in jeder Modi-Konfiguration definiert sind, sind in Klammern gesetzt. Zum Beispiel kann die erste Zeile wie folgt gelesen werden: Führt der Nutzer die Bedienaktion Plus durch, dann wird kein Dialog geöffnet, kein Dialog geschlossen, kein Unterdialog aktiviert und keine Nachricht an die Applikation gesendet. Für die Bedienaktion Menu kann beispielsweise garantiert werden, dass ein Sprung durchgeführt wird.

4.4.2 Bewertung

Auf Basis der Merkmale des Beispiels „MP3-Spieler“ und den Prüfkriterien (s. Abs. 4.3) lassen sich nun Mängel des Bedienkonzepts identifizieren. Ein Mangel tritt genau dann auf, wenn ein Merkmal ein Prüfkriterium nicht erfüllt. Im Folgenden werden beispielhaft zwei Mängel des Bedienkonzepts beschrieben.

Mangel 1. *[Zu Prüfkriterium 4]* Das Bedienkonzept „MP3-Spieler“ verletzt das Kriterium 4, das fordert, dass aus den Anzeigen die jeweilige Modi-Konfiguration hervorgehen soll. Das ist bei den Modi-Konfigurationen 4 und 5, 6 und 7 sowie 19 und 20 nicht der Fall: Ob der letzte Titel der Liste wiedergegeben wird oder nicht, geht aus den Anzeigen nicht hervor (für die Modi-Konfiguration 20 wird dabei angenommen, dass der Nutzer nicht weiß, welcher Titel der Letzte der Liste ist).

Somit kann der Nutzer z. B. im Fall der Modi-Konfigurationen 19 und 20 nicht vorhersehen, ob im nächsten Schritt ein neuer Titel gespielt wird oder zurück ins Menü „HomeMenu“ gesprungen wird. Hinzu kommt, dass bei einem Sprung zurück ins Menü „HomeMenu“ die Attribute der Dialoge „HomeMenu“ und „MusicMenu“ zurückgesetzt werden und somit die Möglichkeit besteht, dass der Nutzer eine zuvor getroffene Auswahl verliert. Eine weitere Konsequenz ist, dass sich der Modus der Bedienaktion Skip bei einer wiederholten Durchführung ändert – während die Anzeige gleich bleibt.

Als Korrekturmaßnahme würde sich anbieten, das Zeichen „▶“, im Fall, dass der letzte Titel der Liste wiedergegeben wird, zu verändern, beispielsweise in „▶|“.

Mangel 2. *[Zu Prüfkriterium 10]* Das Kriterium 10 verlangt, dass es eine Bedienaktion gibt, die in allen Modi-Konfigurationen einen Rückwärts-Sprung auslöst. Tab. 4.4 macht deutlich, dass es eine solche Bedienaktion in dem Bedienkonzept nicht gibt. Für die Bedienaktion Menu ist lediglich garantiert, dass ein Sprung stattfindet. Der Grund dafür ist, dass im Bedienkonzept „MP3-Spieler“ die Bedienaktion Menu zusätzlich die folgenden zwei Aufgaben übernimmt:

1. Das Anschalten des Geräts und

2. das Wechseln vom Dialogstrang „NowPlaying“ zum Dialogstrang „Select“.

Legt man die Rahmenbedingung zugrunde, dass das Bedienkonzept mit den Bedienaktionen Menu, Select, Skip, Plus und Minus auskommen muss, können keine unmittelbaren Korrekturmaßnahmen angeboten werden. Mangel 2 macht deutlich, dass die Abwägung zwischen Anforderungen eine zentrale Aufgabe bei der Entwicklung von interaktiven Geräten darstellt. Das Erstellen eines formalen Analysemodells und die damit verbundene Objektivierung der Bewertung stellt eine hilfreiche Grundlage für den Abwägungsprozess dar.

4.5 Zusammenfassung

In diesem Kapitel wurden die zentralen Techniken dieser Arbeit vorgestellt. Es wurde beschrieben auf welche Art und Weise die Konzepte einer MMI durch Zustandsübergangsdiagramme dargestellt werden können, so dass sie als Analysemodelle Verwendung finden. Darauf aufbauend wurden Merkmale der Analysemodelle identifiziert sowie strukturiert. Dazu wurden die Merkmale in die fünf Bereiche „Struktur der Dialoge“, „Voreingestellte Belegung der Dialogattribute“, „Voreingestellte Fortsetzung der Dialoge“, „Zusammenspiel der Dialoge mit der Applikation“ und „Modi-Konfigurationen“ unterteilt. Ein Teil der Merkmale wurde mit Hilfe der Temporalen Logik formalisiert. Der Bezug der Merkmale zur Gebrauchstauglichkeit wurde erläutert und aus anerkannten Dialoggrundsätzen wurden exemplarisch Prüfkriterien abgeleitet.

Ein Beispiel, die Analyse der MMI eines MP3-Spielers, verdeutlicht das Vorgehen. Es zeigt, dass sich anhand der strukturierten Modellierung gezielte Änderungsvorschläge für das Bedienkonzept ableiten lassen. Ein Hilfsmittel stellt dabei das konzeptuelle Modell dar, das, aufgrund der definierten Terminologie, es erlaubt, die Eigenschaften des Analysemodells klar zu diskutieren.

5 Fallstudie „E60“: Modellbasierte Analyse des BMW iDrive-Systems

Dieses Kapitel beschreibt eine Fallstudie zur Demonstration der Anwendbarkeit der entwickelten Analysetechnik. Die Fallstudie zeigt zum einen die Erstellung der Modelle und zum anderen deren Analyse. Gegenstand der Studie ist das iDrive-System „E60“ der Firma BMW. Dieses System wurde 2001 zum ersten Mal in die Fahrzeuge der 7er-Reihe eingebaut. Die Modellierung erfolgt auf Basis eines Demonstrators[1], der dem Verhalten des tatsächlichen Systems weitestgehend entspricht.

Das iDrive-System wurde entwickelt, um die Anzahl der Bedienelemente im Fahrzeug zu reduzieren. Hierbei ging es vorrangig um die Reduzierung der *tertiären Bedienelemente*. Das sind – nach der Klassifikation von Bubb [29] – die Bedienelemente, die nicht mit der Fahraufgabe in Verbindung stehen, sondern die Komfort-, Unterhaltungs- oder Informationsfunktionen steuern.

Insgesamt steuert das iDrive-System ungefähr 700 Funktionen, die in die Gruppen Navigation, Kommunikation, Entertainment, Klima und Einstellungen unterteilt sind. Neben der Trennung von Fahr- und Komfortfunktionen wurden bei der Entwicklung des iDrive-Konzepts die folgenden Ziele festgelegt [16]:

- *Eine Strukturierung der Interaktion, die für den Fahrer klar und nachvollziehbar ist.* Heutige Fahrzeuge zeichnen sich durch ihre Funktionsvielfalt aus; der Schlüssel zum Erlangen eines gebrauchstauglichen Bedienkonzepts ist laut Bengler et al. [16] diese Funktionsvielfalt klar und nachvollziehbar strukturiert darzubieten.

- *Eine Positionierung der Anzeige- und Bedienelemente, die den Anforderungen der Ergonomie Rechnung trägt:*

    1. *Ablesbarkeit.* Das Anzeigeelement soll auf eine Art und Weise angebracht werden, welche die Akkomodation zwischen Fahrbahn und Anzeigeelement erleichtert und bei einer Blickzuwendung zum Anzeigeelement weiterhin eine periphere Wahrnehmung der Fahrbahn erlaubt.

    2. *Erreichbarkeit.* Das Bedienelement soll so positioniert sein, dass es mit einer natürlichen Armstellung und ohne Blickzuwendung bedient werden kann, und dass es auch dem Beifahrer möglich ist, das Bedienelement zu betätigen.

[1] Demonstrator *E60 MÜ*, Version März 2007, erstellt von der Firma Usaneers GmbH.

- *Eine Gestaltung der MMI, die eine Bedienung während der Fahrt erlaubt.* Die visuelle und auditive MMI soll den spezifischen Anforderungen, die sich aus der Fahrsituation ergeben, gerecht werden.

5.1 Kernkonzepte des iDrive-Systems

Im Folgenden wird auf die wichtigsten Konzepte des iDrive-Systems eingegangen, und zwar auf die Trennung von Anzeige und Bedienung und auf die menügetriebene Dialogführung.

Trennung von Anzeige und Bedienung. Aufgrund der Zielsetzung, den Konflikt zwischen den beiden Anforderungen „Erreichbarkeit“ und „Ablesbarkeit“ zu lösen, trennt das iDrive-Konzept Anzeige und Bedienung. Die Funktionen werden daher über ein zentrales Bedienelement gesteuert, das auf der Mittelkonsole angebracht ist. Abb. 5.1 zeigt diese Bedienelement, das auch als *Dreh-Drück-Steller* bezeichnet wird.

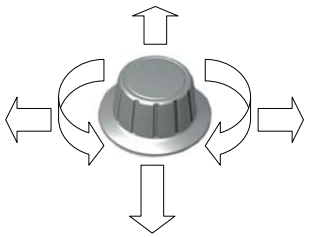


Abbildung 5.1: Dreh-Drück-Steller

Mit Hilfe dieses Bedienelements lassen sich drei Bedienaktionen durchführen: *Schieben*, *Drücken* und *Drehen*, wobei das Schieben in vier Richtungen, das Drehen sowohl im als auch gegen den Uhrzeigersinn möglich ist. Zusätzlich ist unterhalb des zentralen Bedienelements eine Menü-Taste angebracht. Alle Bedienaktionen – außer dem Drehen – sind in zwei Aktionen zu unterteilen: dem Drücken bzw. Schieben und dem Loslassen. Diese Unterscheidung durchgängig zu berücksichtigen ist sinnvoll, da es zu unterscheiden gilt, ob eine Funktion beim Drücken bzw. Schieben oder beim Loslassen ausgeführt wird. Die Unterscheidung ermöglicht auch zeitbehaftete Aktionen zu erkennen, wie beispielsweise den sogenannten *Komfortaufruf*, der ausgeführt wird nachdem die Schiebeaktion zwei Sekunden lang durchgeführt wurde. Für die Darstellung von Informationen verfügt das iDrive-System über ein Flüssigkristallanzeigeelement (engl.: *liquid crystal display*, LCD).

Die menügetriebene Dialogführung des iDrive-Systems. Die Dialogführung des Systems ist menügetrieben und wird im Folgenden vorgestellt. Ausgangspunkt der Menüstruktur ist das Startmenü, zu dem der Fahrer durch Drücken der Menü-Taste

gelangt. Vom Startmenü aus kann in die Gruppen Navigation, Kommunikation, Entertainment, Klima oder Einstellungen gesprungen werden. Der oben erwähnte Komfortaufruf erlaubt dem Fahrer, zwischen diesen Gruppen zu wechseln. Schiebt dieser beispielsweise das Bedienelement für zwei Sekunden nach rechts, gelangt er zur Gruppe Klima.

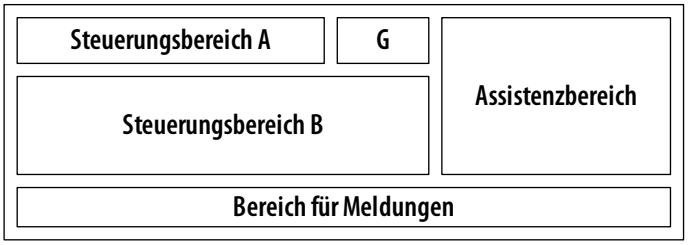


Abbildung 5.2: Bereiche der Anzeige

Die Anzeige des iDrive-Systems ist in der Regel in fünf Bereiche, die in Abb. 5.2 dargestellt sind, unterteilt. In jedem dieser Bereiche können Menüfelder angezeigt werden. In den Steuerungsbereichen sind üblicherweise Menüs mit Bezug zur Aufgabe, die der Fahrer gegenwärtig bearbeitet, dargestellt; der Assistenzbereich zeigt zusätzliche Menüfelder an, die nicht im Bezug zu dieser Aufgabe stehen müssen. Der mit „G“ bezeichnete Bereich zeigt an, in welcher Gruppe man sich befindet. Informationen über den Zustand des Systems werden im Bereich für Meldungen angezeigt. Die dort angezeigten Informationen umfassen u. a. den gewählten Radiosender, die Uhrzeit oder die hergestellten Verbindungen. Das fokussierte Menüfeld ist gegenüber den Feldern, die deaktiviert oder bereit sind, graphisch hervorgehoben. Das fokussierte Menüfeld reagiert auf Bedienaktionen des Fahrers.

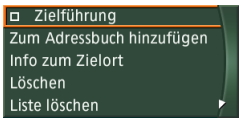
Element	Bezeichnung
	Anzeigefeld
	Schaltfläche
	Horizontale Liste
	Vertikale Liste
	Markierungen

Tabelle 5.1: Menüfelder der iDrive-Anzeigen

Tab. 5.1 zeigt die zentralen Menüfelder der Anzeigen im Überblick. Anzeigenfelder stellen Informationen dar und reagieren typischerweise nicht auf Bedienaktionen des Fahrers. Schaltflächen können ausgeführt werden, dienen also der Eingabe. Die in der Abbildung dargestellte Schaltfläche löst im iDrive-System die „Zurück“-Aktion aus. Eine Liste zeigt dem Fahrer eine Menge an Einträgen an, die horizontal oder vertikal angeordnet sind. Zur Darstellung des Dialogzustandes sind diese Einträge

unterschiedlich ausgezeichnet. Der Rahmen kennzeichnet denjenigen Eintrag der Liste, auf den der Positionszeiger zeigt; die Farbveränderung (die Farbe der Schrift ist Orange, statt Weiß) eines Eintrags kennzeichnet ihn als ausgewählt. Durch weitere Auszeichnungen, wie beispielsweise Kontrollkästchen, wird Zusatzinformation an den Fahrer übermittelt. Der Positionszeiger wird durch Drehen des Bedienelements auf den gewünschten Eintrag gesetzt. Dieser kann dann durch Drücken ausgewählt und ausgeführt werden. Zwischen den angezeigten Menüfelder kann durch Schieben des Bedienelements gesprungen werden.

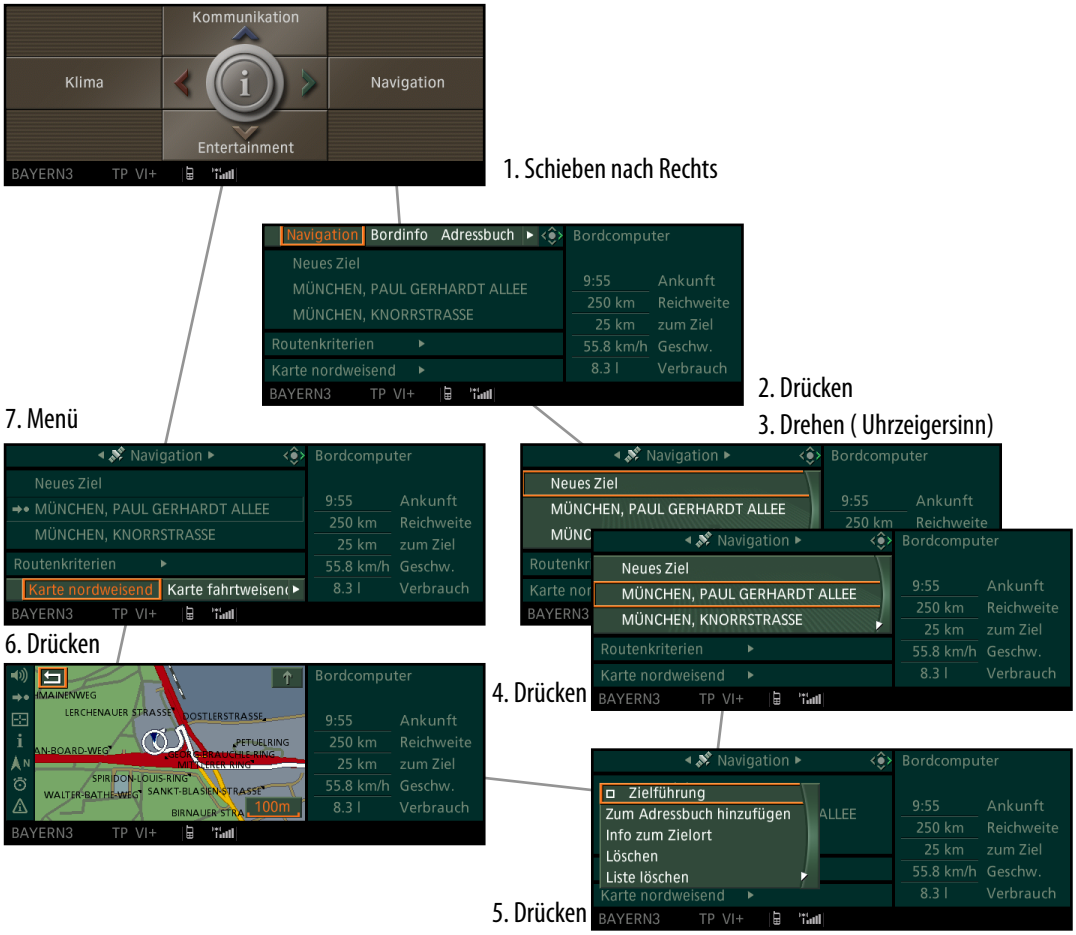


Abbildung 5.3: Zielführung aktivieren

Szenario „Zielführung aktivieren“. Im Folgenden wird ein Szenario zur Verdeutlichung der Interaktion beschrieben. Abb. 5.3 zeigt die Anzeigenfolge des Szenarios. Das Ziel des Fahrers ist es, die Zielführung zu aktivieren. Der Fahrer möchte zur Paul-Gerhardt-Allee.

1. Dazu schiebt er das Bedienelement nach rechts und bekommt als Folge das Navigationsmenü angezeigt.
2. Der Eintrag „Navigation“ ist bereits ausgewählt, sodass der Fahrer durch ein erneutes Drücken zum Zielmenü gelangt.

3. In diesem Menü selektiert der Fahrer den Eintrag, der mit seinem Ziel, der Paul-Gerhardt-Allee, übereinstimmt, und
4. führt diesen aus. Dadurch springt er in ein Pop-Up-Menü, das die möglichen Funktionen auf Zielen anbietet.
5. Der Fahrer führt die Option „Zielführung“ aus und gelangt dadurch direkt ins Zielführungsmenü. Dort ist die Schaltfläche „Zurück“ bereits ausgewählt.
6. Durch Drücken gelangt er zum Kartenmenü und
7. mit dem Drücken der Menü-Taste springt er zurück ins Startmenü.

5.2 Gewählter Ausschnitt des iDrive-Systems

Der Fahrer soll mit Hilfe des Systems die folgenden Aufgaben bearbeiten können. Es soll die Möglichkeit bestehen, aus einer Liste von Zielen ein Ziel auszuwählen und anschließend die Zielführung für dieses Ziel zu aktivieren. Wenn die Zielführung aktiv ist, soll der Fahrer die Zielführung abbrechen können, indem er dasjenige Ziel, zu dem geführt wird, auswählt und im Pulldown-Menü die entsprechende Operation ausführt. Ferner soll der Fahrer ein neues Ziel zur Liste von Zielen hinzufügen können, wobei ein Ziel in dieser Fallstudie ausschließlich aus einer Straße besteht. Außerdem soll der Fahrer die Zielführung anhand einer Kartendarstellung verfolgen können. Zudem ist es die Aufgabe des Fahrers, Meldungen des Systems zur Kenntnis zu nehmen und zu bestätigen.

5.3 Merkmale des Analysemodells

Die Diagramme sowie die Spezifikation der Übergänge sind im Anhang B beschrieben. Im Folgenden werden für die Merkmale, die in Abs. 4.2 definiert sind, die spezifischen Ausprägungen der Fallstudie „E60“ vorgestellt.

Das gesamte Modell der Fallstudie besteht aus 31 zusammengesetzten Zuständen. Darin enthalten sind 10 orthogonale Zustände, 8 Dialoge, 10 Menüs und 10 Optionsgruppen. Die Anzahl der explizit deklarierten Variablen ist 16. Insgesamt gibt es 58 einfache Zustände. Von diesen Zuständen sind 20 Prompts und 22 Intermediates. Die Anzahl der Transitionen ist 344. Die Zustände sind über 95 Stellen miteinander verbunden. Sieben dieser Stellen repräsentieren das Schließen von Dialogen und 9 davon ein Abbrechen.

Das Modell hat $2^{139,734}$ verschiedene Zustände, davon sind 11 721 erreichbar. Die Zeitdauer für eine Prüfung der Modi-Konfigurationen durch den Model-Checker NuSmv beträgt für dieses Modell ungefähr vier Stunden.[2]

[2] Auf einem Laptop, der über einen Intel Dual-Core-Prozessor (2 GHz) und 2 GB RAM verfügt.

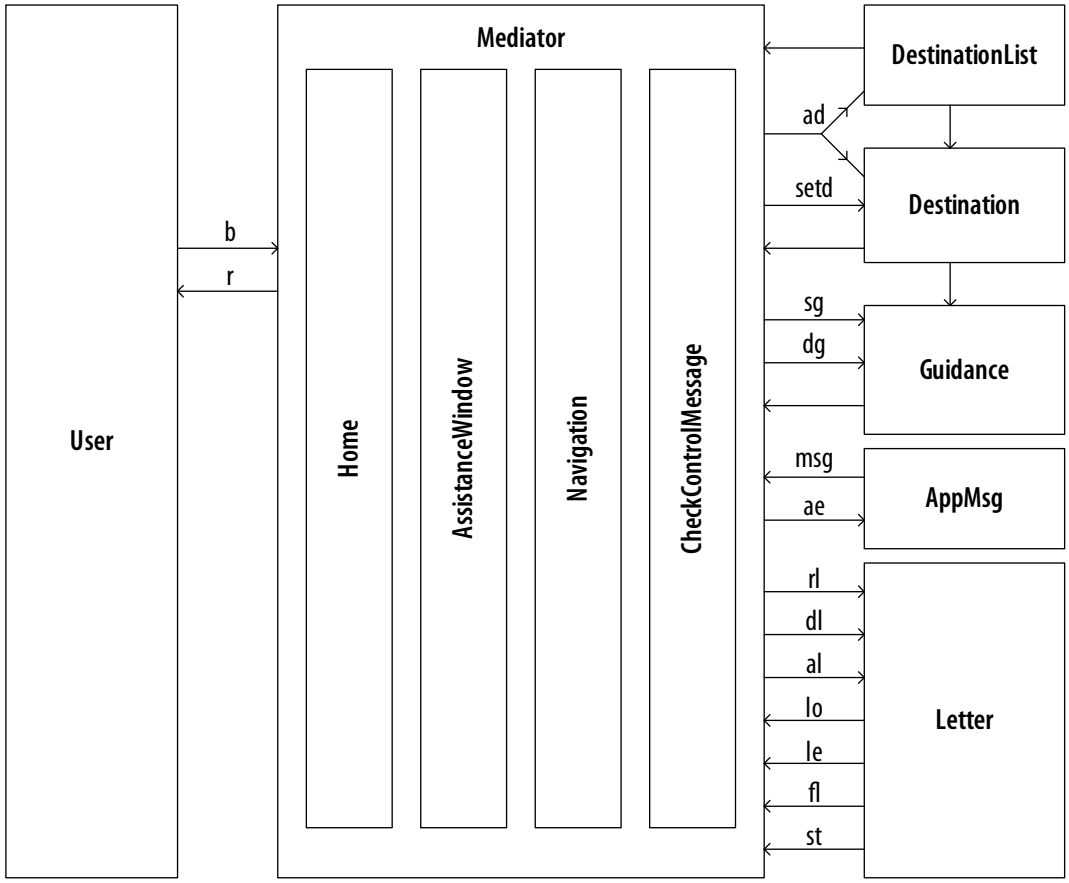


Abbildung 5.4: Struktur der Fallstudie „E60“

Abb. 5.4 zeigt die Struktur des Gesamtsystems. Das System ist unterteilt in die Komponenten „User“, „Mediator“ und in die Komponenten der Applikation. Diese sind „DestinationList“, „Destination“, „Guidance“, „AppMsg“ und „Letter“. Für eine Beschreibung der Kanäle siehe Anhang B.1.

Die Zielliste wird durch die Komponente „DestinationList“ repräsentiert. Die Komponente „Destination“ stellt das gegenwärtig ausgewählte Ziel dar. Die Komponente „Guidance“ modelliert die Zielführung. Applikationsmeldungen werden von der Komponente „AppMsg“ verwaltet. Die Zeichenkette, die bei der Zieleingabe die Straße bestimmt, wird modelliert durch die Komponente „Letter“.

5.3.1 Struktur der Dialoge

Die Komponente „Mediator“ ist unterteilt in die Komponenten „Home“, „AssistanceWindow“, „Navigation“ und „CheckControlMessage“. Diese Komponenten laufen quasiparallel ab und stellen – bis auf die Komponente „CheckControlMessage“ – Dialogstränge dar.

Abb. 5.5 zeigt die Anzeigen des gewählten Ausschnitts. Die Anzeige „HomeMenu“ wird vom Dialogstrang „Home“ angezeigt. Die Anzeige „AssiMenu“ wird vom Dia-

Abbildung 5.5: Anzeigen der Fallstudie „E60“

logstrang „AssistanceWindow“ angezeigt. Die Anzeigen „Msg“ und „BackMsg“ werden von der Komponente „CheckControlMessage“ angezeigt. Die restlichen Anzeigen werden vom Dialogstrang „Navigation“ angezeigt.

Abb. 5.6 zeigt die Übergänge zwischen den Anzeigen. Zu Beginn besitzt der Dialogstrang „Home“ den Fokus. Deswegen bekommt der Fahrer zuerst das Startmenü („HomeMenu“) angezeigt. Durch ein Schieben des Dreh-Drück-Stellers nach rechts (RIGHT) wird das Navigationsmenü aufgerufen. Der Fahrer bekommt eine Liste von Funktionen der Navigation angezeigt („NavList“).

Durch ein Drücken (PRESS) oder ein Schieben nach unten (DOWN) des Dreh-Drück-Stellers kann der Fahrer ins Untermenü „DestMenu“ springen. Dieses Menü ist unterteilt in die zwei Optionsgruppen „DestList“ und „MapList“. Voreingestellt bekommt der Fahrer die Zielliste („DestList“) angezeigt. Die Zielliste zeigt dem Fahrer diejenigen Ziele an, die zur Auswahl stehen. Ist die Zielführung aktiv, dann ist das Ziel, zu dem geführt wird, durch einen Marker gekennzeichnet. Aktiviert der Fahrer ein Ziel, dann wird das Pulldown-Menü („PullMenu“), ein Untermenü von „DestList“, aufgerufen. Das Menü „PullMenu“ wird, wenn der Fahrer eine gewisse Zeitspanne keine Bedienaktion durchführt, abgebrochen. Sonst bietet das Menü dem Fahrer die Option an, die Zielführung zu aktivieren bzw. abzubrechen.

Vom Menü „DestList“ kann der Fahrer durch die Bedienaktion DOWN zur Options-

gruppe „MapList“ gelangen. Aktiviert er in diesem Menü eine Option, dann wird ins Untermenü „MapViewMenu“ gesprungen. Der Fahrer kann durch die Bedienaktion Press zum Menü „MapList“ zurückspringen.

Die erste Option des Menüs „DestList“, die Option „neues Ziel“, ermöglicht dem Fahrer ein neues Ziel hinzuzufügen. Aktiviert der Fahrer die Option „neues Ziel“, dann wird das Menü zur Zieleingabe („SelectInputMenu“) aufgerufen. Dieses Menü ist unterteilt in die Optionsgruppen „InputList“ und „BackSelect“. Voreingestellt springt der Fahrer ins Menü „InputList“. Von dort kann der Fahrer, durch ein Schieben nach oben, zur gleichgestellten Optionsgruppe „BackSelect“ gelangen. Aktiviert der Fahrer die einzige Option durch ein Drücken des Dreh-Drück-Stellers, dann springt er zurück zur Zielliste. Durch ein Drücken im Menü „InputList“ wird das Untermenü „NewDestMenu“ aufgerufen.

Im Menü „NewDestMenu“ kann der Fahrer durch Aktivieren derjenigen Option, welche die gegenwärtige Straße anzeigt, ein neues Ziel erstellen. In diesem Fall springt der Fahrer ins Menü „NewStreetMenu“ zur Straßeneingabe. Das Menü „NewStreetMenu“ ist unterteilt in die Optionsgruppen „LetterList“, „BackStreet“, „DeleteLetter“ und „StreetList“. Voreingestellt bekommt der Fahrer die Optionsgruppe „LetterList“ angezeigt. Diese Liste mit Zeichen kann der Fahrer nutzen, um eine Zeichenkette zu erzeugen. Mit Hilfe dieser Zeichenkette wird in der Optionsgruppe „StreetList“ eine Straße ausgewählt (die erste Straße, für die die Zeichenkette ein Präfix darstellt). Das ermöglicht es dem Fahrer, sich in der umfangreichen Straßenliste zurechtzufinden. Aktiviert der Fahrer eine Straße, so wird diese Straße als neues gegenwärtiges Ziel gesetzt. Durch ein Schieben nach unter kann der Fahrer aus der Optionsgruppe „LetterList“ in die Optionsgruppe „StreetList“ springen.

Liegt eine Meldung, eine sog. *Check Control Message*, für den Fahrer vor, dann wird ins Menü „Msg“ gesprungen. Dies ist ein modales Menü, das heißt, ein Wechsel der Dialogstränge ist nicht mehr möglich. Der Fahrer kann nur durch ein Schieben nach oben ins Menü „BackMsg“ springen. In diesem Menü kann er durch ein Drücken den Empfang der Nachricht bestätigen und es wird zum übergeordneten Dialog zurückgesprungen.

Wenn sich die Nutzerschnittstelle nicht in einem modalen Dialog befindet, kann der Fahrer durch Drücken der Taste „Menü“ immer den Dialogstrang wechseln und zum Menü „HomeMenu“ springen. Dabei wird der gegenwärtige Dialog unterbrochen, d. h. er speichert in der Regel seinen Zustand und legt einen Wiederaufnahmepunkt fest. Wenn der Fahrer das Bedienelement nach rechts schiebt und für zwei Sekunden hält (Komfortaufruf), springt er an die Stelle im Dialogstrang „Navigation“ zurück, von der er den Dialog verlassen hat.

Der Fahrer kann im Dialogstrang „Navigation“ – bis auf eine Ausnahme – durch ein Schieben nach rechts zum Dialogstrang „AssistanceWindow“ wechseln. Von diesem Dialogstrang wechselt der Fahrer mit einem Schieben nach links wieder zurück zum Dialogstrang „Navigation“. Die Ausnahme stellt das Menü „LetterList“ dar: schiebt der Fahrer dort den Dreh-Drück-Steller nach rechts und die Zeichenkette ist nicht leer, dann springt er zur Optionsgruppe „DeleteLetter“.

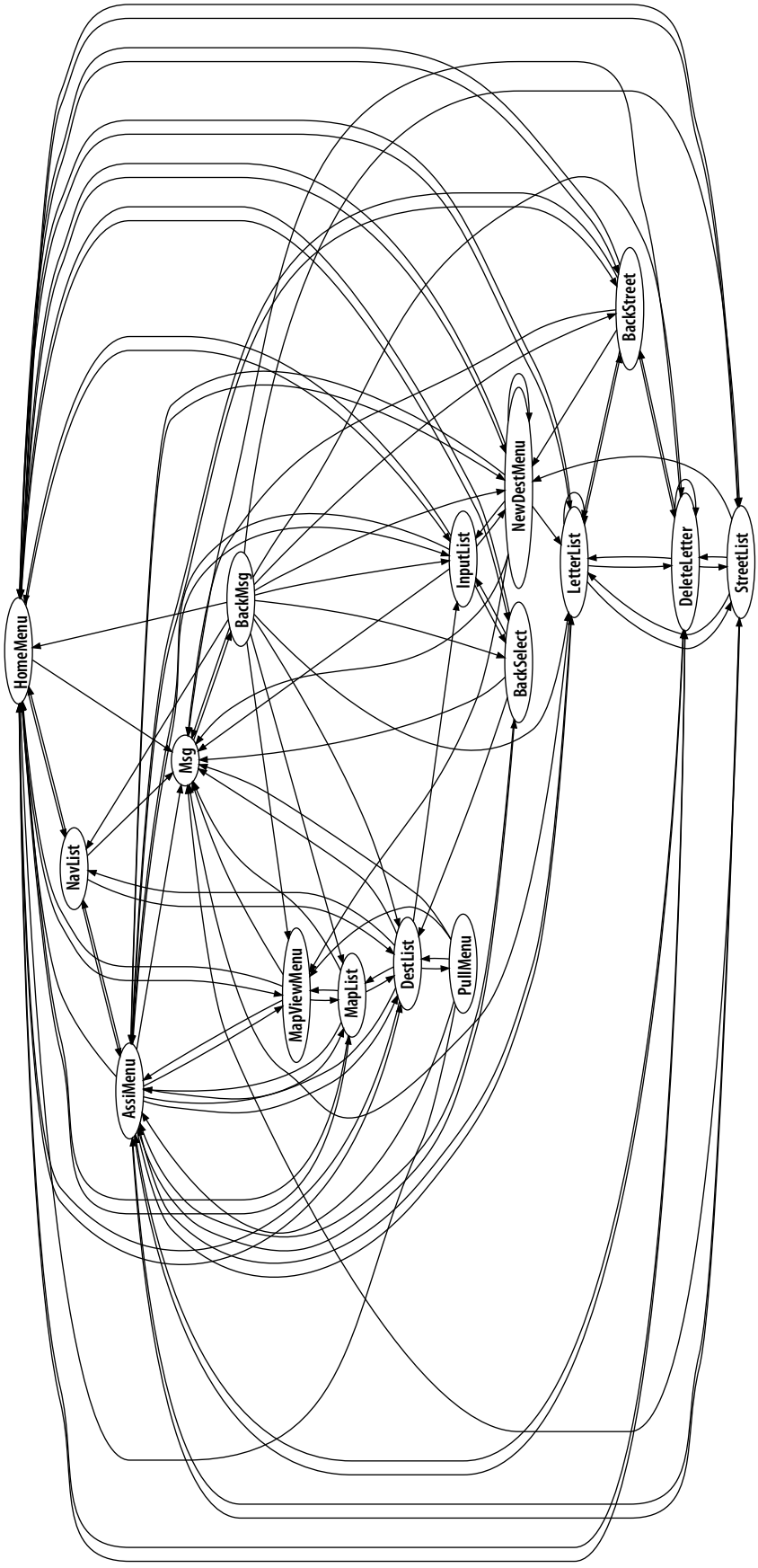


Abbildung 5.6: Übergangsgraph der Anzeigen

5.3.2 Voreingestellte Belegung der Dialogattribute

Im Folgenden wird die Voreinstellung der Belegung der Zeichenkette beschrieben. Dazu wird angenommen, dass gegenwärtig eine Optionsgruppe des Menüs „NewStreetMenu“ fokussiert ist und die Zeichenkette nicht leer ist. Wenn der Fahrer die Bedienaktion MENU durchführt, wird die Zeichenkette zurückgesetzt und zum Dialogstrang „Home“ gewechselt. Dabei ist zu bemerken, dass lediglich ein Wechsel des Dialogstrangs durchgeführt wurde, d. h. der Dialog „NewStreetMenu“ wurde nicht abgebrochen. Wenn der Fahrer hingegen die Bedienaktion RIGHT durchführt, dann wird die Zeichenkette *nicht* zurückgesetzt und zum Dialogstrang „AssistanceWindow“ gewechselt.

5.3.3 Voreingestellte Fortsetzung der Dialoge

Im Folgenden sind die Sprünge beschrieben, bei denen Dialoge geschlossen bzw. abgebrochen werden. Wenn das Menü „PullMenu“ fokussiert ist und der Fahrer aktiviert die Zielführung, dann wird als Nachfolgedialog die Kartenanzeige („MapViewMenu“) aufgerufen und der Fahrer bekommt dieses Menü angezeigt. Wenn das Menü „NewDestMenu“ fokussiert ist und der Fahrer aktiviert die Option „Zur Zielliste hinzufügen“, dann wird als Nachfolgedialog der gleiche Dialog („NewDestMenu“) angezeigt. Allerdings ist dann die Option „Zur Zielliste hinzufügen“ nicht mehr aktivierbar. Wenn das Menü „NewDestMenu“ fokussiert ist und der Fahrer aktiviert die Option „Zielführung starten“, dann wird als Nachfolgedialog die Kartenanzeige („MapViewMenu“) aufgerufen und angezeigt. Wenn das Menü „StreetList“ fokussiert ist und der Fahrer aktiviert eine Option (eine Straße), dann ist der Nachfolgedialog der übergeordnete Dialog „NewDestMenu“. Es erfolgt also ein Rücksprung zum Menü „NewDestMenu“.

5.3.4 Zusammenspiel der Dialoge mit der Applikation

Das Zusammenspiel der Dialoge mit der Applikation zeigt Tabelle 5.2 und ist im Folgenden beschrieben:

1. Wird der Dialog „NewStreetMenu“ geöffnet und es ist gegenwärtig kein Ziel gesetzt, dann ist, wenn der Dialog geschlossen wird, ein Ziel gesetzt, das entweder in der Zielliste ist oder nicht. Wenn der Dialog abgebrochen wird, ist kein Ziel gesetzt.

2. Wird der Dialog „NewStreetMenu“ geöffnet und es ist gegenwärtig ein Ziel gesetzt, das sich in der Zielliste befindet, dann ist, wenn der Dialog geschlossen wird, ein Ziel gesetzt, das entweder in der Zielliste ist oder nicht. Wenn der Dialog abgebrochen wird, dann wird das Ziel nicht verändert und ist somit weiter in der Zielliste.

	Öffnen	Schließen	Abbruch
NewStreetMenu			
1	Dest.@NoDest	Dest.@DestInList ∨ Dest.@DestNotInList	Dest.@NoDest
2	Dest.@DestInList	Dest.@DestInList ∨ Dest.@DestNotInList	Dest.@DestInList
3	Dest.@DestNotInList	Dest.@DestInList ∨ Dest.@DestNotInList	Dest.@DestNotInList
NavList			
4	true	PullMenu = Close ∨ SelectInputMenu = Close	true
PullMenu			
5	Guidance@GOff	Guidance@GOn	Guidance@GOff
6	Guidance@GOn ∧ Navigation@StopG	Guidance@GOff	Guidance@GOn
7	Guidance@GOn ∧ Navigation@StartG	Guidance@GOn	Guidance@GOn
SelectInputMenu			
8	Dest.@NoDest	Dest.@DestInList	Dest.@NoDest ∨ Dest.@DestNotInList
9	Dest.@DestInList	Dest.@DestInList	Dest.@DestInList ∨ Dest.@DestNotInList
10	Dest.@DestNotInList	Dest.@DestInList	Dest.@DestNotInList
CCMenu			
11	AppError@AE2	AppError@AE1	AppError@AE2

Tabelle 5.2: Zusammenspiel der Dialoge mit der Applikation

3. Wird der Dialog „NewStreetMenu“ geöffnet und es ist gegenwärtig ein Ziel gesetzt, das sich nicht in der Zielliste befindet, dann ist, wenn der Dialog geschlossen wird, ein Ziel gesetzt, das entweder in der Zielliste ist oder nicht. Wenn der Dialog abgebrochen wird, dann wird das Ziel nicht verändert und ist somit weiter nicht in der Liste.

4. Wenn der Anfangsdialog „NavList“ geschlossen wird, dann wird zeitgleich entweder der Dialog „PullMenu“ oder der Dialog „SelectInputMenu“ geschlossen.

5. Wenn der Fahrer den Dialog „PullMenu“ öffnet und die Zielführung ist deaktiviert, dann ist in demjenigen Takt, in dem der Dialog geschlossen wird, die Zielführung aktiv. Wenn der Fahrer den Dialog „PullMenu“ öffnet und die Zielführung ist deaktiviert, dann ist in demjenigen Takt, in dem der Dialog abgebrochen wird, die Zielführung deaktiviert.

6. Wenn der Fahrer den Dialog „PullMenu“ öffnet und zu dem ausgewählten Ziel wird gegenwärtig geführt, dann ist in demjenigen Takt, in dem der Dialog geschlossen wird, die Zielführung deaktiviert. Wenn der Fahrer den Dialog „PullMenu“ öffnet und zu dem ausgewählten Ziel wird gegenwärtig geführt, dann

ist in demjenigen Takt, in dem der Dialog abgebrochen wird, die Zielführung unverändert.

7. Wenn der Fahrer den Dialog „PullMenu“ öffnet und zu dem ausgewählten Ziel wird nicht gegenwärtig geführt, dann ist in demjenigen Takt, in dem der Dialog geschlossen wird, die Zielführung aktiv. Wenn der Fahrer den Dialog „PullMenu“ öffnet und zu dem ausgewählten Ziel wird nicht gegenwärtig geführt, dann ist in demjenigen Takt, in dem der Dialog abgebrochen wird, die Zielführung unverändert.

8. Wenn der Fahrer den Dialog „SelectInputMenu“ öffnet und es ist gegenwärtig kein Ziel gesetzt, dann ist, wenn der Dialog geschlossen wird, ein Ziel gesetzt, das sich in der Zielliste befindet. Wenn der Dialog abgebrochen wird, ist entweder kein Ziel gesetzt oder ein Ziel gesetzt, das sich nicht in der Zielliste befindet.

9. Wenn der Fahrer den Dialog „SelectInputMenu“ öffnet und es ist gegenwärtig ein Ziel gesetzt, das sich in der Zielliste befindet, dann ist, wenn der Dialog geschlossen wird, ein Ziel gesetzt, das sich in der Zielliste befindet. Wenn der Dialog abgebrochen wird, ist ein Ziel gesetzt, das sich entweder in der Zielliste befindet oder nicht in der Zielliste befindet.

10. Wenn der Fahrer den Dialog „SelectInputMenu“ öffnet und es ist gegenwärtig ein Ziel gesetzt, das sich nicht in der Zielliste befindet, dann ist, wenn der Dialog geschlossen wird, ein Ziel gesetzt, das sich in der Zielliste befindet. Wenn der Dialog abgebrochen wird, ist ein Ziel gesetzt, das sich nicht in der Zielliste befindet.

11. Wenn die Applikation den Dialog „CCMenu“ öffnet und es liegt eine Meldung vor, dann liegt in demjenigen Takt, in dem der Dialog geschlossen wird, keine Meldung vor.

Der Mediator bietet dem Nutzer die Möglichkeit, alle Dialoge, die vom Nutzer geöffnet werden, abzubrechen. Die Menge der Ansteuerungskanäle umfasst al, dl, sg, ad, dg, $setd$ und rl.

5.3.5 Modi-Konfigurationen

Die Fallstudie zeichnet sich durch 23 Modi-Konfigurationen aus. Die Tabellen der Modi-Konfigurationen sind im Anhang B.2 zu finden.

Tabelle 5.3 zeigt eine Zusammenfassung der Modi-Konfigurationen. Das Zeichen “–” bedeutet, dass in allen Modi-Konfigurationen das Nichteintreten des Effekts der Spalte festgelegt worden ist. Eine Zahl sagt aus, in wieviel Prozent der Modi-Konfigurationen das Auftreten des Effekts festgelegt worden ist. Wird für eine Bedienaktion der Wert 0 % angegeben, so bedeutet das, dass in allen Modi-Konfiguration weder das Auftreten noch das Nichteintreten festgelegt worden ist.

Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
Up	–	–	6 %	–	33 %	66 %	100 %	–	40 %	66 %
Down	–	–	7 %	15 %	7 %	76 %	100 %	–	30 %	76 %
Left	–	–	25 %	–	25 %	50 %	100 %	–	0 %	75 %
Right	5 %	–	10 %	5 %	10 %	10 %	100 %	0 %	15 %	84 %
Timeout	0 %	–	50 %	0 %	50 %	–	100 %	–	0 %	50 %
LongRight	–	–	5 %	–	5 %	15 %	100 %	35 %	100 %	55 %
Press	31 %	26 %	15 %	42 %	42 %	10 %	73 %	57 %	94 %	10 %
Menu	–	–	5 %	–	5 %	15 %	100 %	36 %	–	100 %

Tabelle 5.3: Zusammenfassung der Modi-Konfigurationen: Im Grundwert der Prozentangaben sind diejenigen Konfigurationen enthalten, die für die jeweilige Bedienaktion eine Wirkung definieren

Aus den Modi-Konfigurationen lässt sich eine Tabelle ableiten, welche die Wirkung einer Bedienaktion unabhängig von der gegenwärtigen Modi-Konfiguration beschreibt. Diese als globalen Wirkungen sind in Tabelle 5.4 dargestellt, wobei diejenigen Bedienaktionen, die nicht in jeder Modi-Konfiguration definiert sind, in Klammern gesetzt sind. Für die Bedienaktion Menu kann beispielsweise garantiert werden, dass das Menüfeld „HomeMenu“ als Reaktion angezeigt wird.

Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
(Up)	○	○		○			●	○		
(Down)	○	○					●	○		
(Left)	○	○		○			●	○		
(Right)		○					●			
(Timeout)		○				○	●	○		
(LongRight)	○	○		○			●		●	
(Menu)	○	○		○			HomeMenu		○	●

Tabelle 5.4: Globale Wirkungen der Bedienaktionen

Abb. 5.7 ordnet den Modi-Konfigurationen eine entsprechende Anzeige zu. Menüfelder, die nicht aktiviert werden können, sind gestrichelt dargestellt. Die Übergänge zwischen den Modi-Konfigurationen sind in Anhang B.3 angegeben.

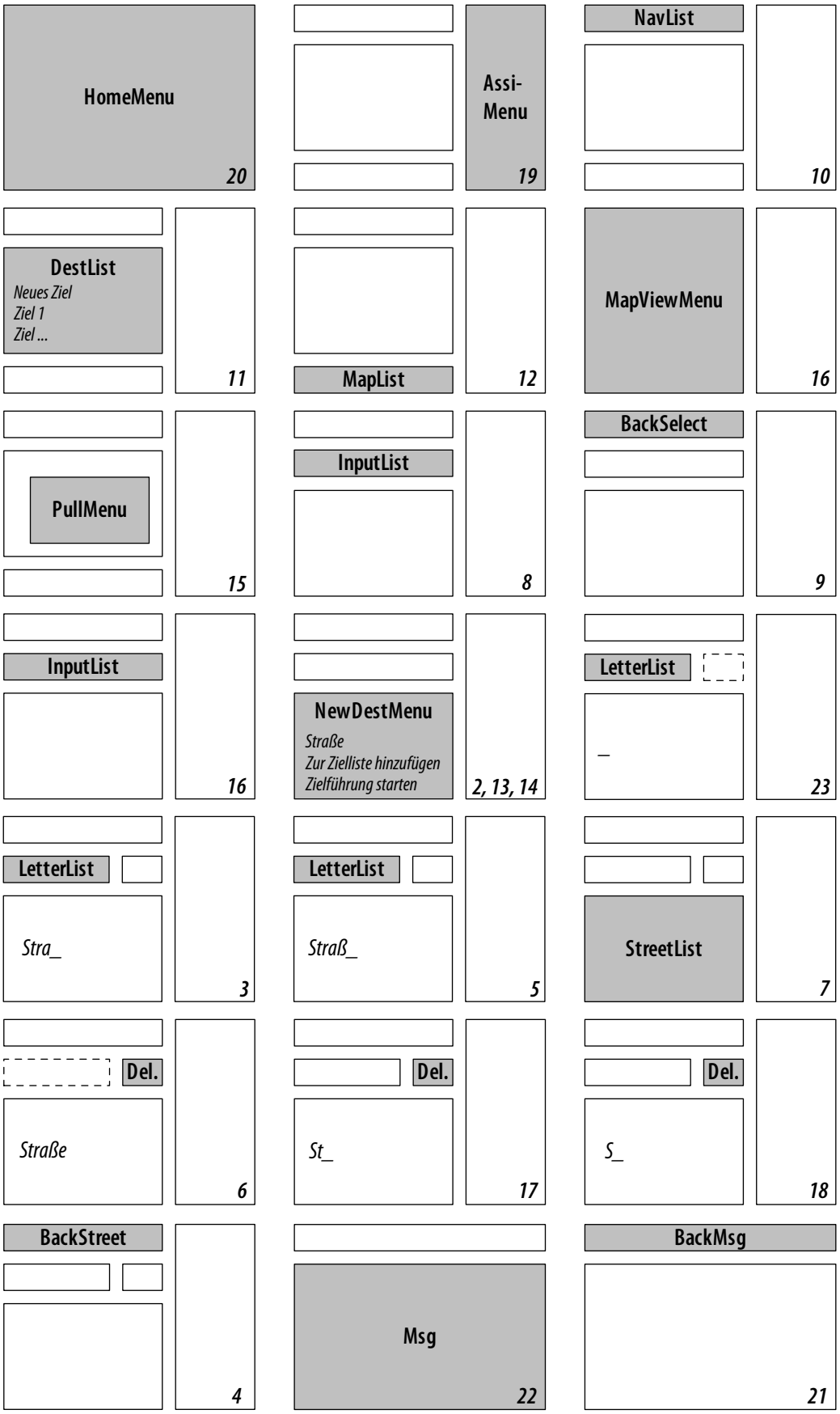


Abbildung 5.7: Zuordnung der Anzeigen zu den Modi-Konfigurationen

5.4 Bewertung

Anhand des strukturierten Modells, der identifizierten Merkmale und der abgeleiteten Prüfkriterien (PKs) (s. Abs. 4.3) ist es nun möglich das Bedienkonzept zu bewerten. Die Bewertung des Bedienkonzepts „E60“ erfolgte anhand einer manuellen Inspektion der Merkmale. Tab. 5.5 listet die entdeckten Mängel auf, die im Folgenden – anhand der Dialoggrundsätze strukturiert – beschrieben sind.

Selbstbeschreibungsfähigkeit

Mangel 1. *[Zu PK 1]* Das Menü „MapViewMenu“ ist der Nachfolgedialog der Dialoge „PullMenu“ und „NewDestMenu“. Auf Grund der Tatsache, dass sich das Menü „MapViewMenu“ inmitten der verschachtelten Dialogstruktur befindet, hat die Fortsetzung durch das Menü „MapViewMenu“ zwangsweise zur Folge, dass ein verschachtelter Sprung vollzogen werden muss. Solche Sprünge nachzuvollziehen, ist für den Fahrer schwierig.

Das Menü „MapViewMenu“ stellt nach der Klassifikation (s. Abs. 3.2) ein Verweilmenü dar und ist ein wesentliches Menü der Navigation. Als mögliche Alternative zur bisherigen Dialoggestaltung bietet es sich an, für Verweilmenüs eigene Dialogstränge einzuführen. Somit wäre der Sprung für den Nutzer, zum einen, leichter nachzuvollziehen (ein Wechsel zwischen Dialogsträngen) und, zum anderen, leichter rückgängig zu machen (für ein Beispiel s. Abs. 4.4).

Mangel 2. *[Zu PK 2]* Die Fortsetzung des Dialogs „SelectInputMenu“ stellt einen Mangel dar. Wie in Abs. 5.3 beschrieben, wird dieser Dialog, wenn er geschlossen wird, erneut geöffnet. Der Nachfolgedialog ist somit erneut der Dialog „SelectInputMenu“. Die Beschriftung der Schaltfläche beschreibt die Aufgabe des Dialogs mit „neues Ziel“ – die voreingestellte Fortsetzung realisiert jedoch die Aufgabe „neue Ziele“.

#	Dialoggrundsatz	PK	Kurzbeschreibung
1	Selbstbeschreibungsfähigkeit	1	Sprung zum Nachfolgedialog „MapViewMenu“
2	Selbstbeschreibungsfähigkeit	2	Fortsetzung des Dialogs „SelectInputMenu“
3	Selbstbeschreibungsfähigkeit	3	Das Menu „PullMenu“
4	Selbstbeschreibungsfähigkeit	4	Bedienaktion MENU
5	Selbstbeschreibungsfähigkeit	4	Bedienaktion PRESS
6	Selbstbeschreibungsfähigkeit	4	Bedienaktion UP
7	Selbstbeschreibungsfähigkeit	4	Verhalten der Zieleingabe
8	Steuerbarkeit	6	Unterbrechung der Zieleingabe
9	Steuerbarkeit	5, 6	Komfortfunktion
10	Erwartungskonformität	8	Abbruch des Dialogs „SelectInputMenu“
11	Erwartungskonformität	10	Zurück zum übergeordneten Menü

Tabelle 5.5: Mängel des Bedienkonzepts

Mangel 3. *[Zu PK 3]* Der Grund, warum eine Reihe von Bedienaktionen das Abbrechen des Menüs „PullMenu“ bewirken, ist das Fehlen einer Bedienaktion, die ausschließlich einen Rücksprung bewirkt. Dadurch handelt sich das Bedienkonzept, zum Beispiel, einen zusätzlichen Modi für die Bedienaktion DOWN ein: Im Menü „PullMenu“ bewirkt DOWN das Abbrechen eines Dialogs und den Sprung ins übergeordnete Menü „DestList“. Üblicherweise bewirkt DOWN einen Sprung zu einem gleich- oder unterrangigen Menü (s. Tab. 5.3).

Mangel 4. *[Zu PK 4]* Beim Wechsel vom Dialogstrang „Navigation“ zum Dialogstrang „Home“ durch die Bedienaktion MENU wird der aktive Dialog des Dialogstrangs „Navigation“ unterbrochen und der Fokus dem Dialogstrang „Home“ übergeben. Als Wiederaufnahmepunkt hält der Dialogstrang zunächst seinen gegenwärtigen Zustand. Bei einem Wechsel vom Dialogstrang „Home“ zurück zum Dialogstrang „Navigation“ durch die Bedienaktion RIGHT wird der Dialogstrang „Navigation“ jedoch zurückgesetzt. Das Problem in diesem Verhalten liegt darin, dass die Möglichkeit besteht, dass der Nutzer fälschlicherweise die Bedienaktion MENU für das Rücksetzen verantwortlich macht. Das hat zur Folge, dass der Effekt der Bedienaktion LONG-RIGHT den Nutzer überrascht, da, in diesem Fall, kein Rücksetzen durchgeführt wird. Aus er Interaktion geht nicht klar hervor, dass die Bedienaktion MENU als Unterbrechung und die Bedienaktion RIGHT als ein Zurücksetzen mit Dialogstrangwechsel aufzufassen sind.

Eine alternative Bedienlogik dazu wäre, erstens, eine Bedienaktion einzuführen, die ausschließlich ein Zurücksetzen bewirkt, d. h. es wird kein Wechsel des Dialogstrangs durchführt, sondern zum Anfangsdialog des Strangs gesprungen; und zweitens, für jeden Dialogstrang eine Bedienaktion zu definieren, durch diese der Dialogstrang direkt angesprungen und der gegenwärtige unterbrochen wird.

Mangel 5. *[Zu PK 4]* Die Tatsache, dass eine Bedienaktion wie PRESS unterschiedliche Modi besitzt, ist durchaus üblich. Allerdings sollte der gegenwärtige Modus aus der Anzeige hervorgehen. Dies ist in den Modi-Konfigurationen 11 und 15 nicht der Fall. In der Konfiguration 11 bewirkt das Ausführen der Option „Ziel n“ durch die Bedienaktion PRESS das Öffnen des untergeordneten Dialogs „PullMenu“. In diesem Menü (Konfiguration 15) bewirkt PRESS die Ansteuerung einer Applikationsfunktion und das Schließen von Dialogen.

Zur Korrektur könnte man beispielsweise die Option „Ziel n“ entsprechend kennzeichnen. Damit aus der Anzeige die Wirkung einer Bedienaktion vorhersehbar ist, verlangt die Richtlinie von Apple [13], die Menüeinträge entsprechend zu kennzeichnen: Verzweigt eine Menüoption in ein untergeordnetes Menü, dann ist die Option mit dem Zeichen „›“ zu versehen. Dadurch ist für den Nutzer immer klar, ob das Ausführen einer Menüoption einen Funktionsaufruf nachsichzieht, oder einen Sprung in ein untergeordnetes Menü.

Mangel 6. *[Zu PK 4]* Auf der Anzeige des iDrive-Systems sind mehrere Menüfelder gleichzeitig dargestellt. Die Beziehung zwischen diesen Menüfeldern kann je nach Situation unterschiedlich sein und geht aus der graphischen Darstellung nicht hervor. Dies beeinträchtigt die Selbstbeschreibungsfähigkeit der Anzeigen und erschwert es dem Fahrer, die Struktur des Menüs zu lernen. Im Menü „NewDestMenu“ (Konfiguration 2, 3 und 14), zum Beispiel, bewirkt Up einen Sprung ins übergeordnete Menü „InputList“. In diesem Menü (Konfiguration 8) bewirkt Up einen Sprung ins gleichrangige Menü „BackSelect“. Dieser Unterschied erschließt sich nicht intuitiv aus der Anzeige.

Mangel 7. *[Zu PK 4]* Die Optionsgruppe „LetterList“ zeigt dem Nutzer eine Liste von Zeichen an, aus der er ein Zeichen auswählen und bestätigen kann. Dieses Zeichen wird an die Zeichenkette angehängt. Da alle möglichen Straßennamen dem System bekannt sind, zeigt die Liste, um die Anzahl der benötigten Bedienaktionen des Nutzers zu reduzieren, nur diejenigen Zeichen an, die, angehängt an die Zeichenkette, eine bekannte Straße ergeben. Wenn der Nutzer ein Zeichen hinzufügt und es in der Folge nicht mehr möglich ist, ein weiteres Zeichen hinzuzufügen, dann wird automatisch in die Optionsgruppe „StreetList“ gesprungen. Das Problem hierbei ist, dass der Nutzer vor dem Hinzufügen eines Zeichens anhand der Anzeige nicht weiß, ob er in der Optionsgruppe bleibt oder ein Wechsel vollzogen wird. Somit ist das Verhalten des Dialogs an dieser Stelle nicht vorhersehbar. Die Alternative wäre, den Sprung vom Menü „LetterList“ zum Menü „StreetList“ durch eine explizite Bedienaktion auszulösen.

Ferner kann die Optionsgruppe „DeleteLetter“ nur angesprungen werden, wenn die Zeichenkette nicht leer ist. Somit kann es vorkommen, dass ein Nutzer, der zur Optionsgruppe „DeleteLetter“ wechseln will, fälschlicherweise zum Dialogstrang „AssistanceWindow“ wechselt.

Steuerbarkeit

Mangel 8. *[Zu PK 6]* Wird ein Unterdialog des Dialogs „NewStreetMenu“ unterbrochen, löscht der Mediator die Belegung der Zeichenkette und aktiviert das Menü „LetterList“ Somit werden bisherige Eingaben des Fahrers gelöscht.

Mangel 9. *[Zu den PKs 5 & 6]* Der Komfortaufruf ist Fahrern, die wenig Wissen über das iDrive-System besitzen, typischerweise unbekannt. Das liegt daran, dass der Komfortaufruf ein Schieben und ein Halten von zwei Sekunden erfordert – eine Bedienaktion, die sich nicht intuitiv aus der Anzeige ableiten lässt. Für solche Fahrer ist folglich, dem Grundsatz der Steuerbarkeit zuwider, ein direktes Wechseln zwischen Dialogsträngen – und damit auch das Unterbrechen und Fortsetzen eines Dialogs – nicht möglich.

Erwartungskonformität

Mangel 10. *[Zu PK 8]* Der Dialog „SelectInputMenu“ verletzt das Kriterium, das fordert, dass die Eintrittsbedingung eines Dialogs bei einem Abbruch gilt. Wird dieser Dialog durch die Option des Menüs „BackSelect“ abgebrochen, wird das ausgewählte Ziel nicht zurückgesetzt. Das verhindert die Möglichkeit, die Wirkung des Dialogs rückgängig zu machen.

Mangel 11. *[Zu PK 10]* Tab. 5.4 macht deutlich, dass es keine Bedienaktion gibt, die einheitlich einen Sprung zum übergeordneten Dialog bewirkt. In Folge dessen ist der Sprung “Zurück zum übergeordneten Menü” nicht konsistent im Bedienkonzept realisiert. In einigen Situationen gelangt man durch ein Schieben des Bedienelements ins übergeordnete Menü; in anderen Situationen muss man zuerst zu einem Menü mit einer Schaltfläche „Zurück“ navigieren und diese ausführen, um ins übergeordnete Menü zu gelangen.

5.5 Zusammenfassung

In diesem Kapitel wurde die Fallstudie aus dem Automotive-Bereich vorgestellt. Zuerst wurden die Kernkonzepte des iDrive-Systems beschrieben und anschließend der gewählte Ausschnitt des iDrive-Systems mit Hilfe der in dieser Arbeit entwickelten Methodik modelliert. Auf Basis des Modells wurden die Eigenschaften des Bedienkonzepts bestimmt. Ausgehend von der strukturierten Modellierung und der Bestimmung der Eigenschaften, wurden Mängel des Bedienkonzepts beschrieben. Die Mängel ergeben sich aus denjenigen Eigenschaften des Modells, die die festgelegten Prüfkriterien verletzen. Die Prüfung erfolgte einerseits auf Basis formal geprüfter dynamischer Eigenschaften des Analysemodells, und andererseits auf Basis struktureller Eigenschaften der verschachtelten Dialoge.

Die Mängel 1–3 und 8–11 stützen die Kritik von Norman [85], der erläutert, dass die Struktur der Dialoge nicht mit der Struktur der Aufgaben des Fahrers im Einklang steht:

> *This is the problem with BMW's original design for the iDrive. The iDrive provided a logical, sensible organization of the automobile's controls and displays. But it failed to support activity patterns. The correct approach to the support of behavior is activity-based classification.*

Ferner stützen die Mängel 4–7 die Kritik von Wolf [106], der darlegt, dass es für den Fahrer an vielen Stellen unklar ist, welche Bedienaktion, die von ihm gewünschte Wirkung erzielt. Die Fallstudie zeigt Eigenschaften eines Bedienkonzepts auf, die entstehen, wenn man die Strukturierung der Interaktion im Vergleich zur Gestaltung der graphischen Darstellung vernachlässigt.

6 Zusammenfassung und Ausblick

In dieser Arbeit wurde eine Methode entwickelt, die Mensch-Maschine-Systeme anhand eines diskreten mathematischen Modells beschreibt. Die Modellbildung stellt die Interaktion, also den Dialog, in den Vordergrund und ermöglicht eine objektive und automatische Analyse von Merkmalen, die im Zusammenhang mit der Gebrauchstauglichkeit stehen. Im Folgenden werden die Kernaspekte der Methode zusammengefasst und im Anschluss daran wird ein Ausblick über weiterführende Arbeiten gegeben.

6.1 Zusammenfassung

Die Methode basiert auf einer automatenorientierten Beschreibung des Systems. Eine Menge von Automaten führen, getrieben durch einen globalen Takt, synchron (im *Lock-Step*) Aktionen aus, wobei auf eine Aktion erst im nächsten Schritt reagiert werden kann. Hierdurch wird die Tatsache berücksichtigt, dass die Durchführung einer Aktion Zeit benötigt. Die Kommunikation zwischen den Automaten erfolgt asynchron und ohne explizites Zwischenspeichern der Nachrichten. Asynchrone Kommunikation bedeutet, dass ein Automat in jedem Takt ohne zu blockieren Nachrichten verschickt. Der Empfänger hat im folgenden Schritt die Wahl, die Nachricht entweder zu verarbeiten oder zu verwerfen.

Da Nachrichten nicht explizit zwischengespeichert werden, können Analysetechniken eingesetzt werden, die einen endlichen Zustandsraum erfordern, wie z. B. Model-Checking. Im Zusammenhang mit Model-Checking bietet der Takt-synchrone Ablauf den Vorteil, dass der Zustandsraum nicht durch ein nichtdeterministisches Scheduling vergrößert wird. Darauf aufbauend wurde in dieser Arbeit ein Systemmodell definiert, das Konzepte aus den Arbeiten [58, 93, 94] integriert. Ferner wurde für dieses Systemmodell eine Abbildung auf die Eingabesprache des Model-Checkers NuSmv [33] definiert. Somit wurde die Möglichkeit geschaffen, die Systembeschreibungen für eine gegebene Eigenschaft automatisch zu verifizieren.

Mit diesem Systemmodell als Grundlage wurde eine Notation definiert, welche die Automaten als hierarchische Zustandsübergangsdiagramme darstellt. Diese Notation ist mit Statecharts [56] und mit Mode-Diagrammen aus [11] verwandt. Beispielsweise wurde das Konzept der Gültigkeitsbereiche für Variablen übernommen, d. h. jede Variable eines zusammengesetzten Zustands besitzt einen Gültigkeitsbereich und speichert einen Ausschnitt der Interaktionshistorie. Wird ein zusammengesetzter Zustand deaktiviert, verlieren seine lokalen Variablen ihre Belegung und die von diesen Variablen gespeicherte Historie geht verloren – ein Effekt, der für die Modellierung der verschachtelten Dialoge benötigt wird.

In dieser Arbeit wurden wesentliche Konzepte zur Beschreibung von menügetriebenen MMIs zu einem konzeptuellen Modell zusammengefasst. Das Gesamtsystem wurde zerlegt in die Teile „Nutzer“, „Mediator“ und „Applikation“. Die Interaktion zwischen dem Nutzer und dem interaktiven System wurde in dieser Arbeit anhand von Dialogsträngen und Dialogen strukturiert. Ein wichtiger Aspekt in diesem Zusammenhang ist der festgelegte Lebenszyklus der Dialoge, der die möglichen Zustände und Übergänge eines Dialogs genau beschreibt. Ein Menü wird in dieser Arbeit als ein spezifischer Dialog aufgefasst. Ein Dialogschritt beginnt mit einer Bedienaktion des Nutzers, gefolgt von einer Reihe an Aktionen des Mediators und der Applikation. Diese Aktionen sind vor dem Nutzer versteckt – er kann sie nicht beobachten. Der Dialogschritt endet mit einer Rückmeldung des Mediators. Im nächsten Dialogschritt reagiert der Nutzer mit seiner Bedienaktion auf diese Rückmeldung.

Typischerweise sind die meisten Bedienaktionen als Reaktion auf eine Rückmeldung des Mediators aufzufassen; dann stellt die Rückmeldung eine Aufforderung, einen sogenannten Prompt, dar. Diesem Prinzip kommt jedoch in heutigen Bedienkonzepten mit graphischen Anzeigen weniger Beachtung zu als bei Kommandozeilen-orientierten Dialogen. Ein weiteres wichtiges Konzept ist die Unterteilung in explizite und implizite Bedienaktionen des Nutzers. Dadurch lassen sich Annahmen über Nutzerintentionen explizit modellieren und deren Wirkung analysieren. Für eine detaillierte Betrachtung dieses Aspekts siehe [25].

Ausgehend vom konzeptuellen Modell wurden die Wirkungen der Bedienaktionen klassifiziert. Somit können für ein gegebenes Bedienkonzept die verschiedenen Modi einer Bedienaktion eindeutig festgelegt werden. Das Konzept der Modi-Konfigurationen umfasst eine Menge von Bedienaktionen und ordnet diese jeweils einem Modus zu. Bei der Bestimmung der Wirkung einer Bedienaktion wird die Ansteuerung der Applikation mit berücksichtigt. Generell spielt der Zusammenhang zwischen der Dialogstruktur und der Funktionalität der Applikation eine wichtige Rolle.

Die hierarchischen Zustandsübergangsdiagramme wurden dahingehend erweitert, dass die identifizierten Konzepte einer MMI explizit modelliert werden können. Die hierarchischen, zusammengesetzten Zustände spiegeln die Dialogstruktur wider, beispielsweise entspricht die Aktivierung eines zusammengesetzten Zustands der Aktivierung eines Dialogs. Somit wurde eine Technik geschaffen, welche die Erstellung von strukturierten MMI-Modellen erlaubt. Dabei stand die Beschreibung der Kontrollstruktur – im Gegensatz zur graphischen Gestaltung der Anzeigen – im Vordergrund.

Anschließend wurden für diese Modelle kennzeichnende Eigenschaften identifiziert und zum Teil und mit Hilfe der Temporalen Logik CTL [34] formalisiert. Die Norm ISO 9241-110 [7] beschreibt sieben Grundsätze zur Gestaltung von Dialogen, die in dieser Arbeit anhand zusätzlicher Quellen erklärt wurden. Diese Grundsätze sind Aufgabenangemessenheit, Selbstbeschreibungsfähigkeit, Steuerbarkeit, Erwartungskonformität, Fehlertoleranz, Individualisierbarkeit und Lernförderlichkeit.

Damit das übergreifende Ziel, die Gebrauchstauglichkeit zu sichern, erfüllt werden kann, ist es notwendig, diejenigen Merkmale zu bestimmen, die einen positiven Einfluss auf die Gebrauchstauglichkeit haben. Dazu wurden exemplarisch Prüfkriterien

aus den Grundsätzen abgeleitet. Der Kernbeitrag ist die Schaffung der Möglichkeit, Merkmale, die sich positiv auf die Gebrauchstauglichkeit auswirken, eindeutig anhand eines mathematischen Modells festzulegen. Grundlegend dafür ist die domänenspezifische Struktur des Modells. Die eindeutig festgelegten und automatisch prüfbaren Merkmale stellen ein objektives Mittel zur Bewertung von Nutzerschnittstellen dar. Die Verifikation dieser Merkmale erfolgt durch den Einsatz des Model-Checkers NuSmv.

Zur Demonstration der praktische Anwendbarkeit der entwickelten Modellierungs- und Analysemethode, wurde eine umfangreiche Fallstudie aus dem Bereich Automotive durchgeführt. Die Fallstudie zeigt, dass die Methode in der Lage ist, Mängel – in Bezug auf die Prüfkriterien – eines Bedienkonzepts zu identifizieren. Die identifizierten Mängel stützen zudem die von Experten geübte Kritik an dem Bedienkonzept.

6.2 Ausblick

Die in dieser Arbeit vorgestellten Techniken bieten die Möglichkeit für Erweiterungen in unterschiedliche Richtungen.

Nutzermodellierung. Die Modellierung des Nutzers im Analysemodell dieser Arbeit ist geradlinig: der Nutzer reagiert nichtdeterministisch auf eine Rückmeldung mit einer Bedienaktion im nächsten Schritt – der kognitive Entscheidungsprozess wird dabei nicht berücksichtigt. Eine Möglichkeit das Modell zu erweitern ist daher, das Verhalten des Nutzers detaillierter zu erfassen. Dazu ist es notwendig, Nutzer- und Aufgabenmodelle aus dem Bereich Mensch-Maschine-Interaktion in das konzeptuelle Modell zu integrieren. Kandidaten sind die Konzepte der Goms-Methode [64] oder das Modell von Norman [84], das die Aktionen des Nutzers in sieben Abstraktionsebenen strukturiert. Diese Modelle beschreiben und erklären kognitive und motorische Aktionen des Nutzers und erlauben eine Abschätzung der Aufgabendauer sowie eine Abschätzung der kognitiven Belastung. Da der Zustandsraum des Analysemodells mit einem komplexeren Nutzerverhalten deutlich umfangreicher ausfällt, als mit dem Nutzermodell, das in dieser Arbeit verwendet wird, ist als Analyseverfahren eine Simulation des Modells besser geeignet, als das Verfahren Model-Checking.

Multimodalität. Es besteht in der Praxis der Bedarf, Bedienkonzepte zu entwickeln, deren Verhalten durch parallele oder verzahnte Ausführung multimodaler Bedienaktionen gesteuert wird [97]. Der Aspekt Multimodalität ist in dem konzeptuellen Modell dieser Arbeit zwar enthalten, es ist jedoch aus dem genannten Grund erstrebenswert, diesen Aspekt noch stärker auszubauen. Im Bezug zur Multimodalität kann das Konzept der Sichtbarkeit der Dialoge verfeinert werden und es können weitere Merkmale identifiziert werden.

Erlebbare Prototypen. Der Schwerpunkt dieser Arbeit liegt in der Beschreibung des Interaktionsverhaltens – die konkrete Gestaltung der Darstellung steht im Hintergrund. Um das modellierte Bedienkonzept für potentielle Nutzer erlebbar darzustellen, bietet sich eine Verfeinerung der Analysemodelle dahingehend an, dass detaillierte Prototypen mit auditiven, visuellen und taktilen Bedien- und Anzeigeelementen beschrieben werden.

Formalisierung der Beschreibungssprachen aus dem Automotive-Bereich. Im Automotive-Bereich werden eine Reihe von Beschreibungssprachen eingesetzt, um Bedienkonzepte zu beschreiben [67]. Diese Sprachen basieren auf XML und besitzen die Schwäche, dass sie sehr konkret auf objektorientierte Sprachen zugeschnitten sind und die genaue Bedeutung der Konstrukte sowie deren Zusammenhänge unklar bleiben. Hier besteht die Möglichkeit, diesen Sprachen – anhand des in dieser Arbeit vorgestellten Systemmodells – eine formale Bedeutung zuzuordnen.

Die in dieser Arbeit verwendeten Prüfkriterien wurden aus den Dialoggrundsätzen der ISO 9241-110 [7] abgeleitet. Die Erstellung eines umfassenden Qualitätsmodells stellt einen nächsten Schritt dar. Die Grundlagen eines solchen Qualitätsmodells sind bereits in [105] gelegt. Zusätzlich besteht die Möglichkeit mit Hilfe empirischer Nutzertests eine zusätzliche Validierung der Prüfkriterien durchzuführen.

Bounded-Model-Checking. Erweitert man die Analysemodelle in ihrem Umfang, steigt der Ressourcen- und Zeitbedarf des Model-Checkers stark an. Für die Analyse umfangreicher, komplexer Bedienkonzepte gibt es die Möglichkeit entweder Bounded-Model-Checking (BMC) [19] oder Runtime-Verifikation [15] im Verbund mit einer Simulation einzusetzen. BMC unterscheidet sich vom klassischen Model-Checking darin, dass Abläufe nur über eine begrenzte Anzahl von Takten betrachtet werden. Auf Grund der Tatsache, dass die Interaktion zwischen dem Nutzer und dem System in Dialogschritte zergliedert wird, deren Länge ebenfalls eingeschränkt werden könnte, ist insbesondere zu untersuchen inwiefern BMC sich eignet, um Eigenschaften eines Dialogschritts zu prüfen. Bei dem Verfahren Runtime-Verifikation, werden neben der Simulation des Bedienkonzepts die spezifizierten Eigenschaften geprüft. Der Ressourcenverbrauch ist im Vergleich mit dem Verfahren Model-Checking wesentlich geringer, da die Prüfung nicht alle Abläufe mit einschließt. Eine weitere Möglichkeit umfangreiche Modelle in den Griff zu bekommen ist dadurch gegeben, die Abbildung des Systemmodells auf die Sprache NUSMV bezüglich der Effizienz zu optimieren.

Abbildung auf F#. Generell kann untersucht werden, auf welche weiteren Sprachen sich eine Abbildung des Systemmodells als sinnvoll erweist. Für die Erstellung eines Simulations-Rahmenwerk bietet beispielsweise F# [98], eine Funktionale Sprache auf Basis von .NET, interessante Konzepte. Insbesondere das Konzept der asynchronen Ausdrücke zur Beschreibung von Nebenläufigkeit könnte sich als geeignetes Implementierungskonstrukt herausstellen. Schließlich bietet die Werkzeugunterstützung, die in dieser Arbeit entwickelt wurde, neben der Verbesserung der Robustheit, zahlreiche

Möglichkeiten für Erweiterungen. Beispielsweise könnte man die Methoden aus [93] zur Identifikation von Konflikten, wenn zusammengesetzte Zustände synchron kompositioniert werden, integrieren.

Literaturverzeichnis

[1] Norm EN ISO 13407 1999. *Benutzer-orientierte Gestaltung interaktiver Systeme*

[2] Norm EN ISO 9241-11 1999. *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 11: Leitsätze*

[3] Norm EN ISO 9241-14 1999. *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten – Teil 14: Dialogführung mittels Menüs*

[4] Siedersleben, Johannes (Hrsg.): *Quasar: Die sd&m Standardarchitektur – Teil 2.* 2003

[5] Norm EN ISO 15005 2003. *Straßenfahrzeuge – Ergonomische Aspekte von Fahrerinformations- und -assistenzsystemen – Grundsätze und Prüfverfahren des Dialogmanagements*

[6] Norm EN ISO 9000 2005. *Qualitätsmanagementsysteme: Grundlagen und Begriffe*

[7] Norm EN ISO 9241-110 2006. *Ergonomie der Mensch-System-Interaktion – Teil 110: Grundsätze der Dialoggestaltung.* – Ersatz für DIN EN ISO 9241-10:1996-07

[8] Abowd, Gregory D. ; Wang, Hung-Ming ; Monk, Andrew F.: A Formal Technique for Automated Dialogue Development. In: *DIS '95: Proceedings of the Conference on Designing Interactive Systems.* New York, NY, USA : ACM Press, 1995, S. 219–226

[9] Abrams, Marc ; Helms, James: User Interface Markup Language (UIML): Working Draft 3.1 / Organization for the Advancement of Structured Information Standards. 2004. – Forschungsbericht

[10] Adam Opel AG: *Opel Cd 70 Navi: Infotainment System.* Rüsselsheim, Germany, 2004. – Art.-Nr. 09 952 84 01/2004

[11] Alur, Rajeev ; Grosu, Radu: Modular refinement of hierarchic reactive machines. In: *ACM Trans. Program. Lang. Syst.* 26 (2004), Nr. 2, S. 339–369

[12] Anderson, John R.: *Kognitive Psychologie.* 2. Heidelberg : Spektrum Akademischer Verlag, 1996

[13] Apple Computer, Inc: *Macintosh Human Interface Guidelines.* Reading : Addison-Wesley, 1992

[14] Apple Computer, Inc: *iPod mini: Benutzerhandbuch.* 2004. – URL: `www.apple.com/de/support/ipod` – D019-0240

[15] Bauer, Andreas ; Leucker, Martin ; Schallhart, Christian: Runtime Verification for LTL and TLTL / TU München. 2007 (TUM-I0724). – Forschungsbericht

[16] Bengler, Klaus u. a.: Usability Engineering bei der Entwicklung von iDrive. In: *Informationstechnik und Technische Informatik* 44 (2002), Nr. 3, S. 145–152

[17] Benz, Sebastian: Combining Test Case Generation for Component and Integration Testing. In: *A-MOST*, 2007, S. 23–33

[18] Berstel, Jean ; Reghizzi, Stefano C. ; Roussel, Gilles ; Pietro, Pierluigi S.: A scalable formal method for design and automatic checking of user interfaces. In: *ACM Trans. Softw. Eng. Methodol.* 14 (2005), Nr. 2, S. 124–167

[19] Biere, Armin ; Cimatti, Alessandro ; Clarke, Edmund M. ; Zhu, Yunshan: Symbolic Model Checking without BDDs. In: *TACAS '99: Proceedings of the 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*. London, UK : Springer-Verlag, 1999, S. 193–207

[20] Bolognesi, Tommaso ; Brinksma, Ed: Introduction to the ISO specification language LOTOS. In: *Comput. Netw. ISDN Syst.* 14 (1987), Nr. 1, S. 25–59

[21] Booch, Grady ; Rumbaugh, James ; Jacobson, Ivar: *Das UML-Benutzerhandbuch*. Addison-Wesley, 1999

[22] Broy, Manfred: The Specification of System Components by State Transition Diagrams. Technische Universität München, 1997 (TUM-I9729). – Forschungsbericht

[23] Broy, Manfred: Innovation in Engineering Software Intensive Systems. In: *Managing Development and Application of Digital Technologies*, Springer, 2006, S. 3–15

[24] Broy, Manfred ; Cengarle, María V. ; Rumpe, Bernhard: Semantics of UML, Towards a System Model for UML, Part 3: The State Machine Model / Technische Universität München. 2007. – Forschungsbericht

[25] Broy, Manfred ; Leuxner, Christian ; Sitou, Wassiou ; Spanfelner, Bernd ; Winter, Sebastian: Formalizing the Notion of Adaptive System Behavior. In: *SAC '09: Proceedings of the 2009 ACM symposium on Applied Computing*, ACM, 2009, S. 1029–1033

[26] Broy, Manfred ; Rumpe, Bernhard: Modulare hierarchische Modellierung als Grundlage der Software- und Systementwicklung. In: *Informatik Spektrum* 30 (2007), Nr. 1, S. 3–8

[27] Broy, Manfred ; Stolen, Ketil: *Specification and Development of Interactive Systems: FOCUS on Streams, Interfaces, and Refinement*. New York : Springer-Verlag, 2001

[28] Bryant, Randal E.: Graph-Based Algorithms for Boolean Function Manipulation. In: *IEEE Trans. Comput.* 35 (1986), Nr. 8, S. 677–691

[29] Bubb, Heiner: Fahrerassistenz primär ein Beitrag zum Komfort oder für die Sicherheit? In: *Der Fahrer im 21. Jahrhundert: Anforderungen, Anwendungen, Aspekte für Mensch-Maschine-Systeme.* Düsseldorf : VDI-Verlag, 2003, S. 25–33

[30] Campos, José C. ; Harrison, Michael D.: Model Checking Interactor Specifications. In: *Automated Software Engg.* 8 (2001), Nr. 3-4, S. 275–310

[31] Card, Stuart K. ; Moran, Thomas P. ; Newell, Allen: *The Psychology of Human-Computer Interaction.* Hillsdale, NJ : Erlbaum, 1983

[32] Caspi, P. ; Pilaud, D. ; Halbwachs, N. ; Plaice, J. A.: LUSTRE: a declarative language for real-time programming. In: *POPL '87*, ACM, 1987, S. 178–188

[33] Cimatti, A. ; Clarke, E. ; Giunchiglia, E. ; Giunchiglia, F. ; Pistore, M. ; Roveri, M. ; Sebastiani, R. ; Tacchella, A.: NuSMV Version 2: An OpenSource Tool for Symbolic Model Checking. In: *Proc. International Conference on Computer-Aided Verification (CAV 2002)* Bd. 2404. Copenhagen, Denmark : Springer, July 2002 (LNCS)

[34] Clarke, Edmund M. ; Emerson, E. A.: Design and synthesis of synchronization skeletons using branching time temporal logic. In: *Logics of Programs* Bd. 131, Springer, 1982 (LNCS)

[35] Commission of the European Communities: Updated Version of the European Statement of Principles on Human Machine Interface (HMI) for In-Vehicle Information and Communication Systems. In: *Commission Recommendation of 22 December 2006 on safe and efficient in-vehicle information and communication systems: update of the European statement of principles on human machine interface.* Brussels, Belgium : European Union, 2006

[36] Conway, Melvin E.: Design of a separable transition-diagram compiler. In: *Commun. ACM* 6 (1963), Nr. 7, S. 396–408

[37] Coutaz, Joëlle: PAC, an Object Oriented Model for Dialogue Design. In: Bullinger, H.-J. (Hrsg.) ; Shackel, B. (Hrsg.): *Human-Computer Interaction – INTERACT'87*, Elsevier Science Publishers B.V., 1987, S. 431–436

[38] d'Ausbourg, Bruno u. a.: Helping the Automated Validation Process of User Interfaces Systems. In: *ICSE '98: Proceedings of the 20th International Conference on Software Engineering.* Washington, DC, USA : IEEE Computer Society, 1998, S. 219–228

[39] Degani, Asaf: *Modeling Human-Machine Systems: On Modes, Error, and Patterns of Interaction*, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA, Ph.D. dissertation, 1996

[40] Dill, David L. ; Drexler, Andreas J. ; Hu, Alan J. ; Yang, C. H.: Protocol Verification as a Hardware Design Aid. In: *ICCD '92*, IEEE Computer Society, 1992, S. 522–525

[41] Dix, Alan J.: *Formal Methods for Interactive Systems.* London : Academic Press, 1991 (Computer and People Series)

[42] Dix, Alan J.: Formal Methods in HCI: Moving Towards an Engineering Approach. In: *NDISD'93 – HCI: Making Software Usable, Wolverhampton*, 1993

[43] Duke, David ; Faconti, Giorgio ; Harrison, Michael ; Paternò, Fabio: Unifying views of interactors. In: *AVI '94: Proceedings of the workshop on Advanced visual interfaces.* New York, NY, USA : ACM, 1994, S. 143–152

[44] Dwyer, Matthew B. ; Avrunin, George S. ; Corbett, James C.: Property specification patterns for finite-state verification. In: *Proceedings of the Second Workshop on Formal Methods in Software Practice*, ACM Press, 1998, S. 7–15

[45] Dwyer, Matthew B. ; Carr, Vicki ; Hines, Laura: Model Checking Graphical User Interfaces Using Abstractions. In: *ESEC '97/FSE-5: Proceedings of the 6th European Conference.* New York, NY, USA : Springer-Verlag New York, Inc., 1997, S. 244–261

[46] Eclipse Foundation: *Eclipse: An Open Development Platform.* – URL: `http://www.eclipse.org/` (zugegriffen am 11.08. 2008)

[47] Edmund M. Clarke, Jr. ; Grumberg, Orna ; Peled, Doron A.: *Model Checking.* The MIT Press, 1999

[48] Garavel, Hubert u. a.: CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes. In: *CAV 2007*, 2007, S. 158–163

[49] Geiser, Georg: *Mensch-Maschine-Kommunikation.* München : Oldenbourg Verlag, 1990

[50] Gram, Christian (Hrsg.) ; Cockton, Gilbert (Hrsg.): *Design principles for interactive software.* London : Chapman & Hall, Ltd., 1997

[51] Green, M.: Report on Dialogue Specification Tools. In: Pfaff, Günther E. (Hrsg.): *Seeheim Workshop on User Interface Management Systems*, Springer-Verlag, 1985, S. 9–20

[52] Grosu, Radu u. a.: State Transition Diagrams / Technische Universität München. 1996 (TUM-I9630). – Forschungsbericht

[53] Grosu, Radu ; Rumpe, Bernhard: Concurrent Timed Port Automata / Technische Universität München. 1995 (TUM-I9533). – Forschungsbericht

[54] Grudin, Jonathan: The Case Against User Interface Consistency. In: *Commun. ACM* 32 (1989), Nr. 10, S. 1164–1173

[55] Hamberger, Werner u. a.: *Audi Multi Media Interface MMI.* 2004. – Präsentation auf der Euroforum Automobile Cockpits/Human Machine Interface

[56] Harel, David: Statecharts: A Visual Formalism for Complex Systems. In: *Science of Computer Programming* 8 (1987), June, Nr. 3, S. 231–274

[57] Harrison, Michael ; Thimbleby, Harold: The Role of Formal Methods in Human-Computer Interaction. In: Harrison, Michael (Hrsg.) ; Thimbleby, Harold (Hrsg.): *Formal Methods in Human-Computer Interaction*. Cambridge : Cambridge University Press, 1990 (Cambridge Series on Human-Computer Interaction)

[58] Henzinger, Thomas A.: Masaccio: A Formal Model for Embedded Components. In: *IFIP TCS*, 2000, S. 549–563

[59] Holzinger, Andreas: Usability Engineering Methods for Software Developers. In: *Commun. ACM* 48 (2005), Nr. 1, S. 71–74

[60] Holzmann, Gerard J.: *The SPIN Model Checker: Primer and Reference Manual*. Boston : Addison-Wesley, 2004

[61] Horrocks, Ian: *Constructing the User Interface with Statecharts*. Boston, USA : Addison-Wesley, 1999

[62] Huber, Franz ; Schätz, Bernhard ; Spies, Katharina: AutoFocus – Ein Werkzeugkonzept zur Beschreibung verteilter Systeme. In: Herzog, Ulrich (Hrsg.): *Formale Beschreibungstechniken für verteilte Systeme*, Universität Erlangen-Nürnberg, 1996

[63] Jeffries, Robin ; Miller, James R. ; Wharton, Cathleen ; Uyeda, Kathy: User interface evaluation in the real world: a comparison of four techniques. In: *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 1991, S. 119–124

[64] Kieras, David: A Guide to GOMS Model Usability Evaluation Using NGOMSL. In: Helander, Martin G. (Hrsg.) ; Landauer, Thomas K. (Hrsg.) ; Prabhu, Prasad V. (Hrsg.): *Handbook of Human-Computer Interaction*. 2. Amsterdam : Elsevier Science B.V., 1997, S. 733–766

[65] Kistler, Günter: *Ein Model-Checking basierter Ansatz zur Prüfung von Nutzungsschnittstellen*, Technische Universität München, Fakultät für Informatik, Diplomarbeit, 2004

[66] Krasner, Glenn E. ; Pope, Stephen T.: A Cookbook for Using the Model-View Controller User Interface Paradigm in Smalltalk-80. In: *J. Object Oriented Program.* 1 (1988), Nr. 3, S. 26–49

[67] Kuhn, Lukas D.: *Entwurf, Definition und praktische Umsetzung eines MMI-Spezifikationsformalismus*, Institut für Informatik, Ludwig-Maximilians-Universität München, Diplomarbeit in Informatik, 2006

[68] Lang, Manfred: Usability Engineering. In: *Informationstechnik und Technische Informatik* 44 (2002), Nr. 3, S. 3–4

[69] Loer, Karsten ; Harrison, Michael: Formal Interactive Systems Analysis and Usability Inspection Methods: Two Incompatible Worlds? In: *Interactive Systems: Design, Specification, and Verifikation, DSV-IS 2000* Bd. 1946, Springer, 2001 (Lecture Notes in Computer Science), S. 169–190

[70] Manna, Zohar ; Pnueli, Amir: *The Temporal Logic of Reactive and Concurrent Systems: Specification.* New York : Springer-Verlag, 1992

[71] Mantei, Marilyn M. ; Teorey, Toby J.: Cost/benefit analysis for incorporating human factors in the software lifecycle. In: *Commun. ACM* 31 (1988), Nr. 4, S. 428–439

[72] Markopoulos, Panos: Interactors: formal architectural models of user interface software. In: Kent, A. (Hrsg.) ; Williams, J.G. (Hrsg.): *Encyclopedia of Microcomputers* Bd. 27. New York : Marcel-Dekker, 2001, S. 203–235

[73] McMillan, K.L.: *Symbolic Model Checking.* Kluwer, 1993

[74] Microsoft Corporation: *Windows Presentation Foundation: XAML.* – URL: `http://msdn.microsoft.com/en-us/library/ms747122.aspx` (zugegriffen am 02.02. 2009)

[75] Moore, Edward F.: Gedanken-experiments on sequential machines. In: Shannon, C.E. (Hrsg.) ; McCarthy, J. (Hrsg.): *Automata studies*, Princeton Univ. Press, 1956, S. 179–210

[76] Mori, Giulio ; Paternò, Fabio ; Santoro, Carmen: CTTE: Support for Developing and Analyzing Task Models for Interactive System Design. In: *IEEE Transactions on Software Engineering* 28 (2002), Nr. 8, S. 797–813

[77] Mozilla Corporation: *XUL: XML User Interface Language.* – URL: `https://developer.mozilla.org/en/XUL` (zugegriffen am 02.02. 2009)

[78] Myers, Brad A.: A new model for handling input. In: *ACM Trans. Inf. Syst.* 8 (1990), Nr. 3, S. 289–320

[79] *Kapitel* 1. In: Myers, Brad A.: *Languages for Developing User Interfaces.* Boston : Jones and Bartlett Publishers, 1992, S. 1–20

[80] Myers, Brad A.: Why are Human-Computer Interfaces Difficult to Design and Implement? Pittsburgh, PA, USA : Carnegie Mellon University, Juli 1993 (CMU-CS-93-183). – Forschungsbericht

[81] Nielsen, Jakob: *Usability Engineering.* Boston : AP Professional, 1993

[82] Nirschl, G. ; Blum, E.J.: MMI-Prüfliste: Verfahren und Werkzeug zur Bewertung von Mensch-Maschine-Systemen im Kraftfahrzeug. In: Bundesanstalt für Strassenwesen (Hrsg.): *Informations- und Assistenzsysteme im Auto benutzergerecht gestalten*, Wirtschaftsverlag NW, 2000, S. 42–49

[83] Norman, Donald A.: Design rules based on analyses of human error. In: *Commun. ACM* 26 (1983), Nr. 4, S. 254–258

[84] Norman, Donald A.: *The design of everyday things.* 3. London : MIT Press, 2000

[85] Norman, Donald A.: Logic versus usage: the case for activity-centered design. In: *interactions* 13 (2006), Nr. 6, S. 45–ff

[86] oAW: *openArchitectureWare: The Leading Platform for Professional Model-Driven Software Development.* – URL: `http://www.openarchitectureware.org/` (zugegriffen am 11.08. 2008)

[87] Olsen, Dan R. Jr.: Propositional production systems for dialog description. In: *CHI '90*, ACM, 1990, S. 57–64

[88] Paternò, Fabio: *Model-Based Design and Evaluation of Interactive Applications.* London : Springer-Verlag, 1999

[89] Paternò, Fabio ; Santoro, Carmen: Integrating Model Checking and HCI Tools to Help Designers Verify User Interface Properties. In: *Interactive Systems: Design, Specification, and Verifikation, DSV-IS 2000* Bd. 1946, Springer, 2001 (Lecture Notes in Computer Science), S. 135–150

[90] Philipps, Jan ; Slotosch, Oscar: The Quest for Correct Systems: Model Checking of Diagrams and Datatypes. In: *APSEC'99: Asian Pacific Software Engineering Conference*, IEEE Computer Society, 1999, S. 449–458

[91] Raskin, Jef: *Das intelligente Interface: Neue Ansätze für die Entwicklung interaktiver Benutzerschnittstellen.* Boston : Addison-Wesley, 2001

[92] Rushby, John M.: Analyzing Cockpit Interfaces Using Formal Methods. In: *Electr. Notes Theor. Comput. Sci.* 43 (2001)

[93] Schätz, Bernhard: Building Components from Functions. In: *Electr. Notes Theor. Comput. Sci.* 160 (2006), S. 321–334

[94] Schätz, Bernhard: Modular Functional Specification of Reactive Components. In: *Proceedings of Formal Aspects of Component Systems FACS'07*, 2007

[95] Shneiderman, Ben: *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* 3. Reading, Massachusetts : Addison-Wesley, 1998

[96] Souchon, Nathalie ; Vanderdonckt, Jean: A review of XML-compliant user interface description languages. In: *Lecture notes in computer science* 2844 (2003), S. 377–391

[97] Strasser, Thomas: *Konzeption, Beschreibung und Realisierung von Multimodal-Widgets zur Erstellung von homogenen HMIs*, Technische Universität München, Diplomarbeit in Informatik, 2006

[98] Syme, Don ; Granicz, Adam ; Cisternino, Antonio: *Expert F#.* Apress, 2007

[99] Thimbleby, Harold: *User Interface Design.* New York : ACM Press, 1990

[100] Thimbleby, Harold: *Press On: Principles of interaction programming.* MIT Press, 2007

[101] Thimbleby, Harold ; Gow, Jeremy: Applying graph theory to interaction design. In: *Proceedings of Engineering Interactive Systems*, Springer, 2008 (LNCS). – to appear

[102] Tidwell, Jenifer: *Designing Interfaces: Patterns for Effective Interaction Design*. O'Reilly, 2005

[103] Wagner, Stefan: A Literature Survey of the Quality Economics of Defect-Detection Techniques. In: *Proc. 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE'06)*, ACM Press, 2006

[104] Wasserman, Anthony I.: Extending State Transition Diagrams for the Specification of Human-Computer Interaction. In: *IEEE Trans. Softw. Eng.* 11 (1985), Nr. 8, S. 699–713

[105] Winter, Sebastian ; Wagner, Stefan ; Deissenboeck, Florian: A Comprehensive Model of Usability. In: *Proceedings of Engineering Interactive Systems*, Springer, 2008 (LNCS)

[106] Wolf, Hagen: *Datenblätter zur ergonomischen Ist-Analyse des Navigationssystems*. 2004. – Enthalten im Abschlussbericht der Phase 1 des TUMMIC-Projekts

A Details zum Beispiel „MP3-Spieler“

Dieser Anhang beschreibt Details zum Beispiel „MP3-Spieler“ aus Abs. 4.4. Zuerst werden die Zustandsübergangsdiagramme beschrieben. Anschließend werden die Modi-Konfigurationen mit Hilfe von Tabellen festgelegt. Zuletzt wird der Übergangsgraph der Modi-Konfigurationen angegeben.

A.1 Zustandsübergangsdiagramme

Im Folgenden werden die Zustandsübergangsdiagramme sowie die Tabellen, welche die Übergänge spezifizieren, für das Beispiel „MP3-Spieler“ angegeben. Für jeden zusammengesetzten Zustand werden seine einfachen Zustände angegeben und für jeden dieser einfachen Zustände werden diejenigen Übergänge beschrieben, die den Zustand aktivieren.

Bei der Spezifikation der Übergänge wird für die Bedienaktionen (Nil, Plus, Minus, Select, Skip, Menu oder Timeout) die folgende Schreibweise verwendet: In den Vorbedingungen wird, beispielsweise für die Bedienaktion Menu, statt „$b = $ Menu“ nur „Menu“ geschrieben und in den Nachbedingungen wird statt „$b' = $ Menu“ ebenfalls nur „Menu“ angegeben, wobei b der Kanal der Bedienaktionen ist.

A.1.1 Orthogonaler Zustand „Player“

Der orthogonale Zustand „Player“ modelliert die Komponente des Geräts, die auf Anfrage Musiktitel abspielt. Abb. A.1 zeigt das Diagramm des orthogonalen Zustands „Player“. Die Übergänge sind in Tab. A.1 angegeben.

Die Variable „ct“ ist mit Null belegt, wenn gegenwärtig kein Titel abgespielt wird. Wenn der Player einen Titel abspielt, dann ist die Variable „ct“ mit der Nummer des Titels belegt. Mit Hilfe der lokalen Variablen „ht“ merkt sich der Player den Titel, den er als nächstes spielen wird.

Der Player besitzt die einfachen Zustände „Off“, „Fsh“ und „On“.

Zustand „Off“. Wenn der Zustand „Off“ aktiviert ist, spielt der Player keinen Titel, d. h. die Variable „ct“ ist mit dem Wert Null belegt. Der Zustand „Off“ wird aktiviert, wenn

- die initiale Transition schaltet (Übergang 0).
- keine Aufforderung zur Wiedergabe eines Titels vorliegt (3).

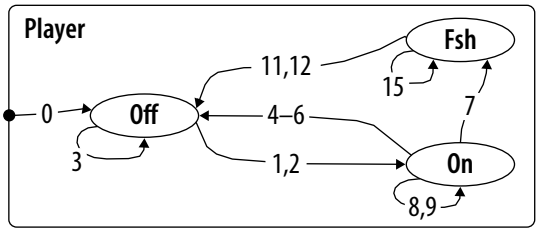


Abbildung A.1: Diagramm für „Player“

Player		
local $ht : \{0,1,2,3\}$		
in-out $ct : \{0,1,2,3\}$		
in nxt, cnt, st : Bool; $at : \{0,1,2,3\}$		
out fd, pl, cp, hn : Bool		
0	–	–
1	$st \wedge \neg cnt \wedge at \neq 0$	$ct' = at \wedge ht' = 0$
2	$\neg st \wedge cnt \wedge at = 0 \wedge ht \neq 0$	$ct' = ht \wedge ht' = 0$
3	$\neg st \wedge \neg cnt \wedge at = 0$	$ct' = 0$
4	$st \wedge at \neq 0$	$ht' = at \wedge ct' = 0 \wedge hn'$
5	$\neg st \wedge nxt \wedge ct < 3$	$ht' = ct + 1 \wedge ct' = 0 \wedge hn'$
6	$\neg st \wedge nxt \wedge ct = 3$	$ht' = 0 \wedge ct' = 0$
7	$\neg st \wedge \neg nxt \wedge cnt$	$ct' = ct \wedge pl' \wedge fd'$
8	$\neg st \wedge \neg nxt \wedge cnt$	$ct' = ct \wedge pl'$
9	$\neg st \wedge \neg nxt \wedge \neg cnt$	$ct' = ct$
11	$\neg st \wedge cnt \wedge \neg nxt \wedge ct < 3$	$ht' = ct + 1 \wedge ct' = 0 \wedge hn' \wedge cp'$
12	$\neg st \wedge cnt \wedge \neg nxt \wedge ct = 3$	$ht' = 0 \wedge ct' = 0 \wedge cp'$
15	$\neg st \wedge \neg cnt \wedge \neg nxt$	$ct' = ct$

Tabelle A.1: Übergangstabelle für „Player“

- der Zustand „Fsh“ die Kontrolle besitzt und die Umgebung den Player durch die Variable „cnt“ bittet fortzufahren (11, 12), wobei der der Player die Fortsetzung durch die Variable „cp“ signalisiert; Die Variable „hn“ gibt zusätzlich an, ob der Player über einen nächsten Titel verfügt, den er spielen kann.

- der Zustand „On“ die Kontrolle besitzt und die Umgebung eine Aufforderung schickt, einen anderen Titel zu spielen (4–6); Die Aufforderung erfolgt durch die Signale „st“ oder „nxt“, wobei mit dem Signal „st“ auch das Argument „at“ verschickt wird.

Zustand „On“. Wenn der Zustand „On“ aktiviert ist, spielt der Player einen Titel. Der Zustand „On“ wird aktiviert, wenn

- der Zustand „Off“ die Kontrolle besitzt und die Umgebung die Aufforderungen „st“ oder „cnt“ verschickt (1, 2); Im Fall, dass das Signal „cnt“ auftritt, wird der Titel gespielt, der in der Variable „ht“ vermerkt ist; im Fall, dass das Signal

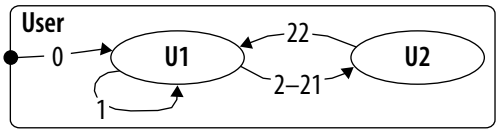


Abbildung A.2: Diagramm für „User“

„st“ auftritt, wird der Titel gespielt, der durch das Argument „at“ festgelegt ist.

- keine Aufforderung zur Fortsetzung vorliegt (9).
- eine Aufforderung zur Fortsetzung, das Signal „cnt“, vorliegt und der Titel noch nicht beendet ist (8), wobei der Player die Fortsetzung durch die Variable „pl“ signalisiert. Die Tatsache, ob ein Titel beendet ist oder nicht, wird nichtdeterministisch modelliert.

Zustand „Fsh“. Wenn der Zustand „Fsh“ aktiviert ist, ist der Titel, der gegenwärtig gespielt wird, beendet. Der Zustand „Fsh“ wird aktiviert, wenn der Zustand „On“ die Kontrolle besitzt, der Titel beendet ist und die Umgebung das Signal zur Fortsetzung verschickt (7). Durch das Signal „fd“ teilt der Player der Umgebung die Beendigung des Titels mit. In der Folge bleibt der Zustand „Fsh“, solange kein Signal auftritt, aktiviert (15).

A.1.2 Orthogonaler Zustand „User“

Abb. A.2 zeigt das Diagramm des Zustands, der den Nutzer repräsentiert. Die Übergänge sind in Tab. A.2 angegeben. Der Nutzer reagiert unmittelbar auf die Rückmeldung des Mediators mit einer Bedienaktion. Nur diejenigen Bedienaktionen, die für eine Modi-Konfiguration definiert sind, werden ausgeführt.

Timeouts werdem als implizite Bedienaktionen modelliert. Somit ist die Reaktion des Mediators auf einen Timeout, als eine Reaktion auf eine Bedienaktion des Nutzers aufzufassen. Diese Art der Modellierung macht deutlich, dass bei Timeouts Annahmen über Nutzerintentionen gemacht werden.

Der Nutzer besitzt die beiden einfachen Zustände „U1“ und „U2“.

Zustand „U1“. Der Zustand „U1“ wird durch die initiale Transition aktiviert (0) und bleibt aktiviert, solange keine Rückmeldung des Mediators vorliegt (1). Der Zustand „U1“ wird zudem aktiviert, wenn der Zustand „U2“ die Kontrolle besitzt (22).

User		
in $r : M$ out $b : B$		
0	–	Nil
1	$r = 0$	Nil
2	$r = 1$	Timeout
3	$r = 2$	Menu ∨ Select ∨ Skip
4	$r = 3$	Select ∨ Timeout
5	$r = 4$	Select ∨ Plus ∨ Skip ∨ Timeout
6	$r = 5$	Select ∨ Plus ∨ Skip ∨ Timeout
7	$r = 6$	Select ∨ Minus ∨ Skip ∨ Timeout
8	$r = 7$	Select ∨ Minus ∨ Skip ∨ Timeout
9	$r = 8$	Menu ∨ Select ∨ Plus ∨ Timeout
10	$r = 9$	Menu ∨ Select ∨ Plus ∨ Skip ∨ Timeout
11	$r = 10$	Menu ∨ Select ∨ Plus ∨ Skip ∨ Timeout
12	$r = 11$	Menu ∨ Select ∨ Plus ∨ Skip ∨ Timeout
13	$r = 12$	Menu ∨ Select ∨ Plus ∨ Minus ∨ Timeout
14	$r = 13$	Menu ∨ Select ∨ Plus ∨ Minus ∨ Skip ∨ Timeout
15	$r = 14$	Menu ∨ Select ∨ Plus ∨ Minus ∨ Skip ∨ Timeout
16	$r = 15$	Menu ∨ Select ∨ Plus ∨ Minus ∨ Skip ∨ Timeout
17	$r = 16$	Menu ∨ Select ∨ Minus ∨ Timeout
18	$r = 17$	Menu ∨ Select ∨ Minus ∨ Skip ∨ Timeout
19	$r = 18$	Menu ∨ Select ∨ Minus ∨ Skip ∨ Timeout
20	$r = 19$	Menu ∨ Skip ∨ Timeout
21	$r = 20$	Menu ∨ Skip ∨ Timeout
22	–	Nil

Tabelle A.2: Übergangstabelle für „User“

Zustand „U2“. Der Zustand „U2“ wird aktiviert, wenn die Umgebung eine Rückmeldung verschickt (2–21). Welche Bedienaktion der Nutzer durchführt, ist abhängig von der übermittelten Modi-Konfiguration. Das Verhalten des Nutzers ist nichtdeterministisch.

Die Modi-Konfiguration Eins stellt einen speziellen Fall dar: der Nutzer kann nichts tun, außer Zeit verstreichen zu lassen (Timeout).

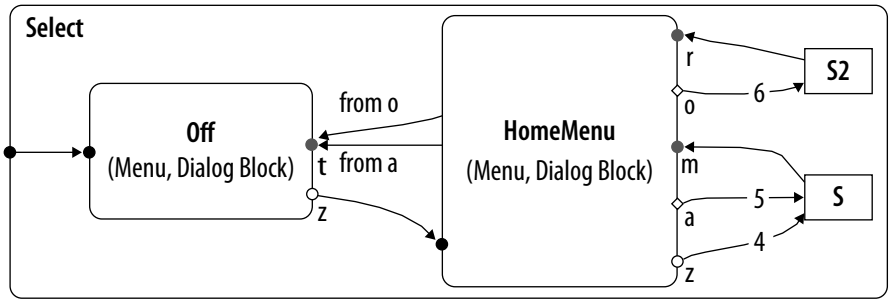


Abbildung A.3: Diagramm für „Select“

Select		
local o, $ho : \{0,1,2,3\}$; h, $hh : \{0,1,2\}$; w : Bool		
in $b : B$; fsh, esh : Bool; $ct : \{0,1,2,3\}$		
out $r : M$; st : Bool; $at : \{0,1,2,3\}$		
4	SELECT $\wedge\ o \neq ct$	$st' \wedge at' = o \wedge$ STORE $\wedge$ resume(Listen)
5	SELECT $\wedge\ o = ct \vee$ TIMEOUT $\wedge\ ct \neq 0 \wedge w$	STORE $\wedge$ resume(Listen)
6	SELECT $\wedge\ h = 2 \vee$ TIMEOUT $\wedge\ ct \neq 0 \wedge w$	STORE $\wedge$ resume(Listen)
STORE $\stackrel{\text{def}}{=}$ $(ho' = o \wedge hh' = h \wedge o' = 0 \wedge h' = 0 \wedge r' = 0)$		

Tabelle A.3: Übergangstabelle für „Select“

A.1.3 Orthogonaler Zustand „Select“

Der orthogonale Zustand „Select“ stellt den Dialogstrang dar, der die Dialoge verwaltet, mit denen der Nutzer einen Musiktitel zur Wiedergabe auswählen kann. Initial besitzt der Dialogstrang „Select“ den Fokus und ist in Abb. A.3 dargestellt. Die Übergänge sind in Tab. A.3 angegeben.

Der Dialogstrang verwaltet die Attribute der Dialoge „HomeMenu“ und „MusicMenu“. Die Variable „h“ zeigt an, ob die Option „Music ›“ oder „Now Playing ›“ gewählt ist. Die Variable „o“ repräsentiert den gewählten Musiktitel und kann mit den Werten Null, Eins, Zwei oder Drei belegt werden.

Wenn der Titel, der abgespielt wird, beendet ist, benötigt der Mediator einen Dialogschritt, um den Dialog „NowPlaying“ zu schließen und einen Nachfolgedialog zu öffnen. Deshalb gibt er die Modi-Konfiguration Eins zurück. Der Nutzer kann dann nur mit der Bedienaktion TIMEOUT reagieren. Die Variable „w“ wird mit dem Wert FALSCH belegt, wenn der Mediator die Modi-Konfiguration Eins zurückgibt.

Der Dialogstrang verfügt über die zwei einfachen Zustände „S“ und „S2“.

Zustand „S“. Wenn der Zustand „S“ aktiviert ist, wurde der Dialog „MusicMenu“ angezeigt. Der Zustand „S“ wird aktiviert, wenn

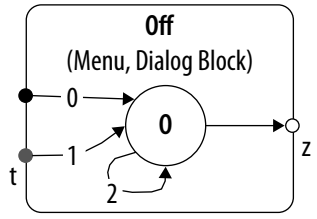


Abbildung A.4: Diagramm für „Off“

Off		
in $b : B$; $ct : \{0,1,2,3\}$		
out $r : M$		
0	Nil	$r' = 2$
1	Timeout $\wedge\, ct = 0$	$r' = 2$
2	Nil	$r' = 0$

Tabelle A.4: Übergangstabelle für „Off“

- die Stelle „z“ über die Kontrolle verfügt, der Nutzer die Bedienaktion Select durchführt und der gewählte Titel nicht dem Titel, der gegenwärtig gespielt wird, entspricht (4); Die Attribute der Dialoge werden gespeichert und der Player wird aufgefordert, den ausgewählten Musiktitel abzuspielen; Der Fokus wird dem Dialogstrang „Listen“ übergeben.
- die Stelle „a“ über die Kontrolle verfügt, der Nutzer die Bedienaktion Select durchführt und der gewählte Titel dem Titel, der gegenwärtig gespielt wird, entspricht (5); Die Attribute werden gespeichert und der Fokus wird dem Dialogstrang „Listen“ übergeben; Die Dialoge „MusicMenu“ und „HomeMenu“ werden dabei abgebrochen. Die gleiche Wirkung erzielt die Bedienaktion Timeout.

Zustand „S2“. Wenn der Zustand „S2“ aktiviert ist, wurde der Dialog „HomeMenu“ angezeigt. Der Zustand „S2“ wird aktiviert, wenn der Nutzer die Option „Now Playing ›“ ausführt (6); Der Dialog „HomeMenu“ wird abgebrochen und der Fokus dem Dialogstang Listen übergeben. Die gleiche Wirkung erzielt die Bedienaktion Timeout.

A.1.4 Zusammengesetzter Zustand „Off“

Der Dialog „Off“ ist aktiviert, wenn das Gerät ausgeschaltet ist. Die Anzeige des Dialogs ist ein deaktiviertes Display. Abb. A.4 zeigt den zusammengesetzten Zustand „Off“. Die Übergänge sind in Tab. A.4 angegeben.

Der Dialog besitzt nur einen einfachen Zustand, das ist der Zustand „O“. Wenn der Zustand „O“ aktiviert ist, ist das Gerät ausgeschaltet. Der Zustand „O“ wird aktiviert, wenn

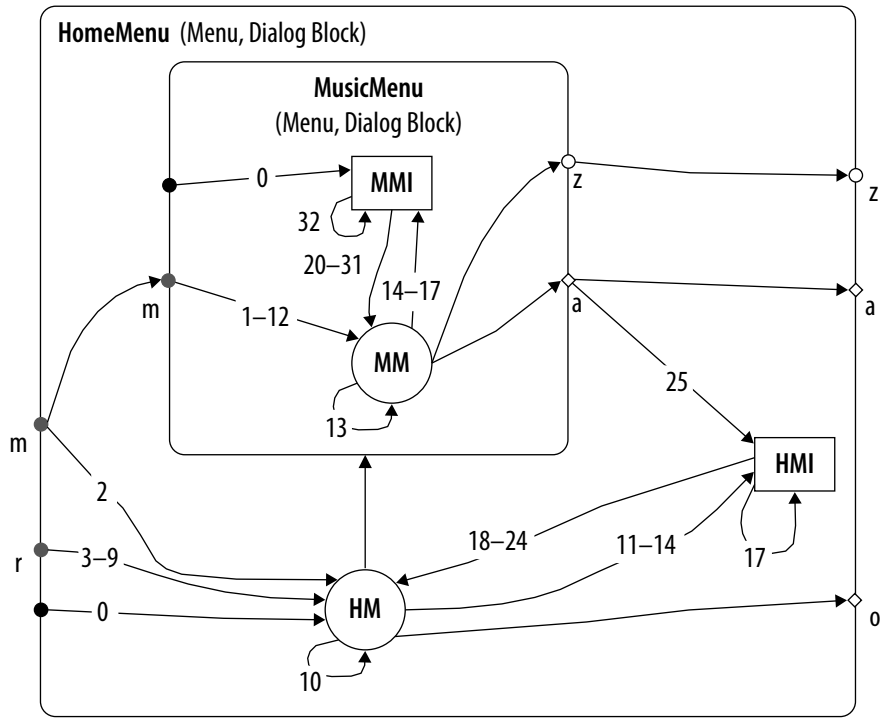


Abbildung A.5: Diagramm für „HomeMenu“ und „MusicMenu“

- die initiale Transition schaltet (0).
- der Nutzer die implizite Bedienaktion TIMEOUT durchführt während das Gerät keinen Titel abspielt (1); Das ist in dem Beispiel „MP3-Spieler“ die einzige Möglichkeit, das Gerät auszuschalten.

Der Zustand bietet der Umgebung die Kontrolle über die Stelle „z“ an.

A.1.5 Zusammengesetzte Zustände „HomeMenu“ und „MusicMenu“

Mit Hilfe der Dialoge „HomeMenu“ und „MusicMenu“ kann der Nutzer einen Titel auswählen oder einen Wechsel zur Anzeige „NowPlaying“ vollziehen. Abb. A.5 zeigt das Diagramm des zusammengesetzten Zustands „HomeMenu“. Die Übergänge sind in Tab. A.5 und in Tab. A.6 angegeben. Zuerst wird der Zustand „HomeMenu“ beschrieben und anschließend der Zustand „MusicMenu“.

Zustand „HomeMenu“

Der zusammengesetzte Zustand „HomeMenu“ besitzt die einfachen Zustände „HM“ und „HMI“.

Zustand „HM“. Wenn der Zustand „HM“ aktiviert ist, wird die Anzeige „HomeMenu“ dargestellt. Der Zustand „HM“ wird aktiviert, wenn

HomeMenu		
in-out o, $ho : \{0,1,2,3\}$; h, $hh : \{0,1,2\}$; w : Bool		
in $b : B$; fsh, esh : Bool; $ct : \{0,1,2,3\}$		
out $r : M$		
0	MENU ∨ SELECT ∨ SKIP	$o' = 1 \wedge h' = 1 \wedge r' = 3$
2	restart ∧ NIL ∧ $ct = 0$	run ∧ $o' = 1 \wedge h' = 1 \wedge r' = 3$
3	restart ∧ NIL ∧ $ct = 0$	run ∧ $o' = 1 \wedge h' = 1 \wedge r' = 3$
4	resume ∧ NIL ∧ $hh = 1 \wedge ct = 0$	run ∧ $o' = ho \wedge h' = hh \wedge r' = 3$
5	resume ∧ NIL ∧ $hh = 1 \wedge ct \neq 0 \wedge ct \neq 3 \wedge fsh \wedge \neg esh$	run ∧ $o' = ho \wedge h' = hh \wedge r' = 4$
6	resume ∧ NIL ∧ $hh = 1 \wedge ct \neq 0 \wedge ct = 3 \wedge fsh \wedge \neg esh$	run ∧ $o' = ho \wedge h' = hh \wedge r' = 5$
7	resume ∧ NIL ∧ $hh = 2 \wedge ct \neq 0 \wedge ct \neq 3 \wedge fsh \wedge \neg esh$	run ∧ $o' = ho \wedge h' = hh \wedge r' = 6$
8	resume ∧ NIL ∧ $hh = 2 \wedge ct \neq 0 \wedge ct = 3 \wedge fsh \wedge \neg esh$	run ∧ $o' = ho \wedge h' = hh \wedge r' = 7$
9	resume ∧ NIL ∧ $ct \neq 0 \wedge fsh \wedge esh$	run ∧ $o' = ho \wedge h' = hh$ $\wedge \neg w' \wedge r' = 1$
10	NIL	$r' = 0$
11	SKIP ∧ $ct \neq 0$	$r' = 0$
12	PLUS ∧ $ct \neq 0 \wedge h = 1$	$h' = 2 \wedge r' = 0$
13	MINUS ∧ $h = 2$	$h' = 1 \wedge r' = 0$
14	TIMEOUT ∧ $ct \neq 0 \wedge \neg w$	$w' \wedge r' = 0$
17	NIL ∧ $\neg fsh \wedge \neg esh$	$r' = 0$
18	NIL ∧ $h = 1 \wedge fsh \wedge \neg esh \wedge ct \neq 0 \wedge ct \neq 3$	$r' = 4$
19	NIL ∧ $h = 1 \wedge fsh \wedge \neg esh \wedge ct \neq 0 \wedge ct = 3$	$r' = 5$
20	NIL ∧ $h = 2 \wedge fsh \wedge \neg esh \wedge ct \neq 0 \wedge ct \neq 3$	$r' = 6$
21	NIL ∧ $h = 2 \wedge fsh \wedge \neg esh \wedge ct \neq 0 \wedge ct = 3$	$r' = 7$
22	NIL ∧ $fsh \wedge esh \wedge ct \neq 0$	$\neg w' \wedge r' = 1$
23	NIL ∧ $fsh \wedge \neg esh \wedge h \neq 2 \wedge ct = 0$	$r' = 3$
24	NIL ∧ $fsh \wedge \neg esh \wedge h = 2 \wedge ct = 0$	$h' = 1 \wedge r' = 3$
25	MENU	$r' = 0$

Tabelle A.5: Übergangstabelle für „HomeMenu“

- die Standard-Stelle über die Kontrolle verfügt (d. h. das Gerät ist ausgeschaltet) und der Nutzer führt eine der Bedienaktionen MENU, SELECT, oder SKIP durch (0); Es wird die Rückmeldung 3 ausgegeben und die Attribute „o“ und „h“ initialisiert.

- die Stelle „m“ über die Kontrolle verfügt, der Fokus übergeben wird und das Gerät keinen Titel wiedergibt (2);

- die Stelle „r“ über die Kontrolle verfügt und der Fokus übergeben wird (3-9); Entsprechen der Situation, wird die jeweilige Modi-Konfiguration zurückgegeben; Der Übergang 3 setzt die Werte der Dialogattribute, wie aufgefordert („restart“), zurück; Das Signal „esh“ zeigt an, dass der Titel beendet ist; In Folge dessen wird die Variable „w“ mit FALSCH belegt (9).

- der Zustand „HMI“ die Kontrolle besitzt und der Dialogstrang „Listen“ seine Aktionen beendet hat (signalisiert durch „fsh“) (18–24); Je nach Situation, wird die entsprechende Modi-Konfiguration zurückgegeben.

Zustand „HMI“. Wenn der Zustand „HMI“ aktiviert ist, hat der Dialogstrang „Listen“ seinen Aktionen nicht beendet. Der Zustand „HM“ wird aktiviert, wenn

- der Zustand „HM“ die Kontrolle besitzt, der Nutzer die Bedienaktion SKIP durchführt und das Gerät einen Titel abspielt (11).
- der Zustand „HM“ die Kontrolle besitzt und der Nutzer die Option „Now Playing ›“ auswählt (12).
- der Zustand „HM“ die Kontrolle besitzt und der Nutzer die Option „Music ›“ auswählt (13).
- der Zustand „HM“ die Kontrolle besitzt, die Variable „w“ mit FALSCH belegt und der Nutzer die Bedienaktion TIMEOUT durchführt (14).
- der Dialogstrang „Listen“ nicht fertig ist (17).
- die Stelle „a“ über die Kontrolle verfügt und der Nutzer die Bedienaktion MENU durchführt (25); Der Dialog „MusicMenu“ wird dabei abgebrochen.

Zustand „MusicMenu“

Diese Menü bietet dem Nutzer eine Liste von drei Titel an, die er auswählen und ausführen kann. Wenn das Gerät einen Titel wiedergibt, kann der Nutzer mit Hilfe der Bedienaktion SKIP die Wiedergabe des nächsten Titels fordern. Der Dialog besitzt die einfachen Zustände „MM“ und „MMI“.

Zustand „MM“. Wenn der Zustand „MM“ aktiviert ist, wird die Anzeige „MusicMenu“ dargestellt. Der Zustand „MM“ wird aktiviert, wenn

- die Stelle „m“ über die Kontrolle verfügt und der Fokus übergeben wird (1–12); Entsprechend der Situation wird eine Modi-Konfiguration zurückgegeben. Wenn der Titel, der wiedergegeben wird, beendet ist („esh“), wird die die Variable „w“ mit FALSCH belegt (12).
- der Zustand „MMI“ die Kontrolle besitzt und der Dialogstrang „Listen“ fertig ist (20–31).

Zustand „MMI“. Wenn der Zustand „MMI“ aktiviert ist, ist der Dialogstrang „Listen“ mit seinen Aktionen noch nicht fertig. Das heißt, es wird im Zustand „MMI“ solange gewartet, bis der Dialogstrang „Listen“ fertig ist. Der Zustand „MMI“ wird aktiviert, wenn

- die Standard-Stelle über die Kontrolle verfügt, d. h. im Dialog „HomeMenu“ ist die Option „Music ›“ ausgewählt, und der Nutzer führt die Bedienaktion SELECT durch (0).
- der Nutzer einen neue Option auswählt (14, 15).
- der Nutzer die Bedienaktion SKIP durchführt (16).

- der Nutzer die Bedienaktion TIMEOUT durchführt, wobei die Variable „w“ mit FALSCH belegt ist (17).
- der Dialogstrang „Listen“ nicht fertig ist (32).

MusicMenu		
in-out o, $ho : \{0,1,2,3\}$; $h : \{0,1,2\}$; w : Bool		
in $hh : \{0,1,2\}$; $b : B$; fsh, esh : Bool; $ct : \{0,1,2,3\}$		
out $r : M$		
0	$\textsc{Select} \wedge h = 1$	$r' = 0$
1	$\textsc{Res} \wedge ho = 1 \wedge ct = 0$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 8$
2	$\textsc{Res} \wedge ho = 1 \wedge ct \neq 0 \wedge ct \neq 3 \wedge ct \neq ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 9$
3	$\textsc{Res} \wedge ho = 1 \wedge ct \neq 0 \wedge ct = 3 \wedge ct \neq ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 10$
4	$\textsc{Res} \wedge ho = 1 \wedge ct \neq 0 \wedge ct \neq 3 \wedge ct = ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 11$
5	$\textsc{Res} \wedge ho \neq 1 \wedge ho \neq 3 \wedge ct = 0$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 12$
6	$\textsc{Res} \wedge ho \neq 1 \wedge ho \neq 3 \wedge ct \neq 0 \wedge ct \neq 3 \wedge ct \neq ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 13$
7	$\textsc{Res} \wedge ho \neq 1 \wedge ho \neq 3 \wedge ct \neq 0 \wedge ct = 3 \wedge ct \neq ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 14$
8	$\textsc{Res} \wedge ho \neq 1 \wedge ho \neq 3 \wedge ct \neq 0 \wedge ct \neq 3 \wedge ct = ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 15$
9	$\textsc{Res} \wedge ho = 3 \wedge ct = 0$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 16$
10	$\textsc{Res} \wedge ho = 3 \wedge ct \neq 0 \wedge ct \neq 3 \wedge ct \neq ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 17$
11	$\textsc{Res} \wedge ho = 3 \wedge ct \neq 0 \wedge ct = 3 \wedge ct = ho$	$\mathsf{run} \wedge o' = ho \wedge h' = hh \wedge r' = 18$
12	$\mathsf{resume} \wedge \textsc{Nil} \wedge fsh \wedge esh \wedge ct \neq 0$	$\mathsf{run} \wedge o' = ho \wedge h' = hh$ $\wedge \neg w' \wedge r' = 1$
13	$\textsc{Nil}$	$r' = 0$
14	$\textsc{Plus} \wedge o < 3$	$o' = o + 1 \wedge r' = 0$
15	$\textsc{Minus} \wedge o > 1$	$o' = o - 1 \wedge r' = 0$
16	$\textsc{Skip} \wedge ct \neq 0$	$r' = 0$
17	$\textsc{Timeout} \wedge \neg w$	$w' \wedge r' = 0$
20	$\textsc{Fed} \wedge o = 1 \wedge ct = 0$	$r' = 8$
21	$\textsc{Fed} \wedge o = 1 \wedge ct \neq 0 \wedge ct \neq o \wedge ct \neq 3$	$r' = 9$
22	$\textsc{Fed} \wedge o = 1 \wedge ct \neq 0 \wedge ct \neq o \wedge ct = 3$	$r' = 10$
23	$\textsc{Fed} \wedge o = 1 \wedge ct \neq 0 \wedge ct = o \wedge ct \neq 3$	$r' = 11$
24	$\textsc{Fed} \wedge o \neq 1 \wedge o \neq 3 \wedge ct = 0$	$r' = 12$
25	$\textsc{Fed} \wedge o \neq 1 \wedge o \neq 3 \wedge ct \neq 0 \wedge ct \neq o \wedge ct \neq 3$	$r' = 13$
26	$\textsc{Fed} \wedge o \neq 1 \wedge o \neq 3 \wedge ct \neq 0 \wedge ct \neq o \wedge ct = 3$	$r' = 14$
27	$\textsc{Fed} \wedge o \neq 1 \wedge o \neq 3 \wedge ct \neq 0 \wedge ct = o \wedge ct \neq 3$	$r' = 15$
28	$\textsc{Fed} \wedge o = 3 \wedge ct = 0$	$r' = 16$
29	$\textsc{Fed} \wedge o = 3 \wedge ct \neq 0 \wedge ct \neq o \wedge ct \neq 3$	$r' = 17$
30	$\textsc{Fed} \wedge o = 3 \wedge ct \neq 0 \wedge ct = o \wedge ct = 3$	$r' = 18$
31	$\textsc{Nil} \wedge fsh \wedge esh \wedge ct \neq 0$	$\neg w' \wedge r' = 1$
32	$\textsc{Nil} \wedge \neg fsh \wedge \neg esh$	$r' = 0$

$$\textsc{Res} \stackrel{\text{def}}{=} (\mathsf{resume} \wedge \textsc{Nil} \wedge fsh \wedge \neg esh)$$
$$\textsc{Fed} \stackrel{\text{def}}{=} (\textsc{Nil} \wedge fsh \wedge \neg esh)$$

Tabelle A.6: Übergangstabelle für „MusicMenu“

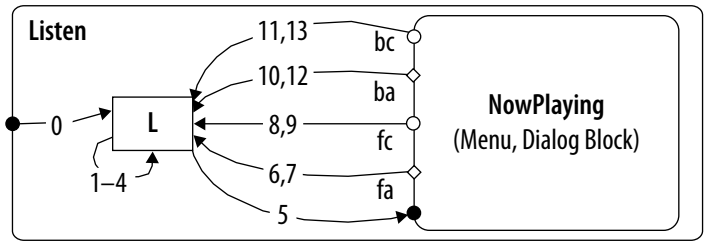


Abbildung A.6: Diagramm für „Listen“

Listen		
local bac : Bool in $b : B$; $ct : \{0,1,2,3\}$; hn, cp, fd, pl : Bool out fsh, esh, nxt, cnt : Bool; $r : M$		
0	–	–
1	halt $\wedge$ Nil $\wedge$ $ct = 0$	–
2	halt $\wedge$ (Menu $\vee$ Minus $\vee$ Plus $\vee$ Skip $\vee$ Select $\vee$ Timeout)	fsh'
3	resume $\wedge$ Nil	run $\wedge$ $r' = 0$
4	run $\wedge$ Nil $\wedge$ $ct = 0$	$r' = 0$
5	–	–
6	Nil $\wedge$ $ct = 0 \wedge hn \wedge \neg cp$	$r' = 0 \wedge cnt'$
7	Nil $\wedge$ $ct = 0 \wedge \neg hn \wedge \neg cp$	restart(Select) $\wedge$ $r' = 0 \wedge bac'$
8	Nil $\wedge$ $ct = 0 \wedge hn \wedge cp$	$r' = 0 \wedge cnt'$
9	Nil $\wedge$ $ct = 0 \wedge \neg hn \wedge cp$	restart(Select) $\wedge$ $r' = 0 \wedge bac'$
10	halt $\wedge$ Nil $\wedge$ $ct = 0 \wedge hn \wedge \neg cp$	cnt'
11	halt $\wedge$ Nil $\wedge$ $ct = 0 \wedge hn \wedge cp$	cnt'
12	halt $\wedge$ Nil $\wedge$ $ct = 0 \wedge \neg hn \wedge \neg cp$	fsh'
13	halt $\wedge$ Nil $\wedge$ $ct = 0 \wedge \neg hn \wedge cp$	fsh'

Tabelle A.7: Übergangstabelle für „Listen“

A.1.6 Orthogonaler Zustand „Listen“

Der Zustand „Listen“ stellt einen Dialogstrang dar. Er verwaltet den Dialog „NowPlaying“, der es dem Nutzer ermöglicht, die Wiedergabe der Musiktitel zu steuern. Abb. A.6 zeigt den Dialogstrang „Listen“. Die Übergänge sind in Tab. A.7 angegeben.

Der orthogonale Zustand „Listen“ besitzt den einfachen Zustand „L“. Wenn der Zustand „L“ aktiviert ist, gibt das Gerät keinen Titel wieder und der Dialog „NowPlaying“ ist geschlossen. Der Zustand „L“ wird aktiviert, wenn

- die initiale Transition schaltet; Der Dialogstrang „Listen“ besitzt jedoch initial nicht den Fokus.
- der Nutzer keine Bedienaktion durchführt, die eine Aufforderung zur Wiedergabe darstellt (2).

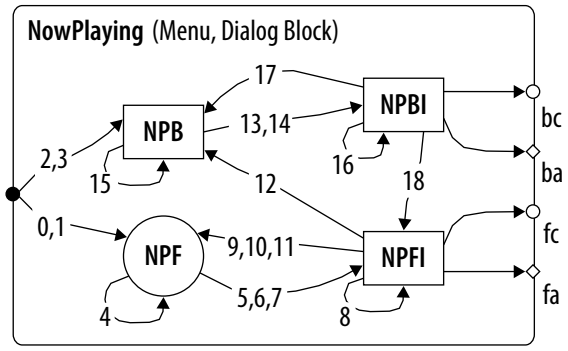


Abbildung A.7: Diagramm für „NowPlaying“

- der Fokus von der Umgebung überreicht wird (3).
- die Stelle „fa“ über die Kontrolle verfügt und kein Titel vom Gerät wiedergegeben wird (6, 7); Wenn es einen nächsten Titel gibt („hn“) wird der Player aufgefordert fortzufahren („cnt“); Wenn es jedoch keinen nächsten Titel gibt, wird der Fokus dem Dialogstrang „Select“ übergeben, wobei „Select“ aufgefordert wird, die Attribute zurückzusetzen („restart“); Der Dialog „NowPlaying“ wird dabei abgebrochen.
- die Stelle „fc“ über die Kontrolle verfügt, kein Titel vom Gerät wiedergegeben wird und der Player anhand der Variable „cp“ signalisiert, dass der Titel nicht abgebrochen wurde (8, 9); Die Variable „hn“ gibt an, ob der Player einen weiteren Titel abspielen kann; Der Dialog „NowPlaying“ wird dabei geschlossen.
- die Stelle „ba“ über die Kontrolle verfügt, der Dialogstrang den Fokus nicht besitzt („halt“) und kein Titel wiedergegeben wird (10, 12); Wenn es keinen weiteren Titel zu spielen gibt, teilt der Dialogstrang „Listen“ dem Dialogstrang „Select“ mit, dass er fertig ist („fsh“); Andernfalls wird der Player aufgefordert fortzusetzen („cnt“); Der Dialog „NowPlaying“ wird dabei abgebrochen.
- die Stelle „bc“ über die Kontrolle verfügt, der Dialogstrang den Fokus nicht besitzt („halt“) und kein Titel wiedergegeben wird (11, 13); Diese Übergänge verhalten sich analog zu den Übergängen 10 und 12, mit dem Unterschied, dass der Dialog „NowPlaying“ geschlossen wird, da der Titel nicht abgebrochen wurde („cp“).

A.1.7 Zusammengesetzter Zustand „NowPlaying“

Abb. A.7 zeigt das Diagramm des zusammengesetzten Zustands „NowPlaying“. Die Übergänge sind in Tab. A.8 angegeben.

Die Variable „bac“ wird mit dem Wert Falsch belegt, wenn der Nutzer die Bedienaktion Menu durchgeführt hat und in Folge dessen der Fokus zurück an den Dialogstrang „Select“ gegeben werden muss. Der Zustand „NowPlaying“ stellt einen Dialog dar und besitzt die einfachen Zustände „NPB“, „NPF“, „NPBI“ und „NPFI“.

NowPlaying		
in-out bac : Bool in $b : B$; $ct : \{0,1,2,3\}$; fd, pl : Bool out fsh, esh, nxt, cnt : Bool; $r : M$		
0	run $\wedge$ Nil $\wedge\, ct \neq 0 \wedge ct \neq 3 \wedge bac$	$r' = 19$
1	run $\wedge$ Nil $\wedge\, ct \neq 0 \wedge ct = 3 \wedge bac$	$r' = 20$
2	run $\wedge$ Nil $\wedge\, ct \neq 0 \wedge \neg bac$	$fsh' \wedge$ resume(Select) $\wedge\, r' = 0 \wedge bac'$
3	halt $\wedge$ Nil $\wedge\, ct \neq 0$	fsh'
4	Nil	$r' = 0$
5	Timeout	$r' = 0 \wedge cnt'$
6	Skip	$r' = 0 \wedge nxt'$
7	Menu	$r' = 0 \wedge cnt' \wedge \neg bac'$
8	Nil $\wedge\, ct \neq 0 \wedge \neg pl$	$r' = 0$
9	Nil $\wedge\, ct \neq 0 \wedge pl \wedge \neg fd \wedge ct \neq 3 \wedge bac$	$r' = 19$
10	Nil $\wedge\, ct \neq 0 \wedge pl \wedge \neg fd \wedge ct = 3 \wedge bac$	$r' = 20$
11	Nil $\wedge\, ct \neq 0 \wedge pl \wedge fd \wedge bac$	$r' = 1$
12	Nil $\wedge\, ct \neq 0 \wedge pl \wedge \neg bac$	$fsh' \wedge esh' = fd$ $\wedge$ resume(Select) $\wedge\, r' = 0 \wedge bac'$
13	halt $\wedge$ (Menu $\vee$ Minus $\vee$ Plus $\vee$ Select $\vee$ Timeout)	cnt'
14	halt $\wedge$ Skip	nxt'
15	halt $\wedge$ Nil $\wedge\, ct \neq 0$	–
16	halt $\wedge$ Nil $\wedge\, ct \neq 0 \wedge \neg pl$	–
17	halt $\wedge$ Nil $\wedge\, ct \neq 0 \wedge pl$	$fsh' \wedge esh' = fd$
18	resume $\wedge$ Nil	run $\wedge\, r' = 0$

Tabelle A.8: Übergangstabelle für „NowPlaying“

Zustand „NPB“. Wenn der Zustand „NPB“ aktiviert ist, besitzt der Dialogstrang „Listen“ nicht den Fokus, d. h. der Dialog „NowPlaying“ befindet sich im Zustand „bereit“. Der Zustand „NPB“ wird aktiviert, wenn

- die Standard-Stelle über die Kontrolle verfügt, der Dialogstrang „Listen“ nicht den Fokus besitzt oder die Variable „bac“ mit Falsch belegt ist und das Gerät einen Titel wiedergibt (2, 3); Im Fall, dass „bac“ mit Falsch belegt ist, wird der Fokus dem Dialogstrang „Select“ übergeben.
- der Zustand „NPFI“ die Kontrolle besitzt, die Variable „bac“ mit dem Wert Falsch belegt ist und der Player seine Fortsetzung signalisiert („pl“) (12); Der Fokus wird dem Dialogstrang „Listen“ übergeben.
- der Zustand „NPBI“ die Kontrolle besitzt, der Dialogstrang „Listen“ nicht den Fokus besitzt und der Player seine Fortsetzung signalisiert („pl“) (17).

Zustand „NPF“. Wenn der Zustand „NPF“ aktiviert ist, wird die Anzeige „Now-Playing“ dargestellt“. Der Dialog „NowPlaying“ ist im Zustand „fokussiert“. Der Zustand „NPF“ wird aktiviert, wenn

- die Standard-Stelle über die Kontrolle verfügt, der Dialogstrang „Listen“ den Fokus besitzt und das Gerät einen Titel wiedergibt (0, 1).
- der Zustand „NPFI“ die Kontrolle besitzt und der Player signalisiert, dass er seine Fortsetzung abgeschlossen hat („pl“) (9, 10, 11).

Zustand „NPBI“. Wenn der Zustand „NPBI“ aktiviert ist, ist hat Player seine Fortsetzung nicht abgeschlossen („¬pl“) und der Dialog ist im Zustand „bereit“. Der Zustand „NPB“ wird aktiviert, wenn

- der Player seine Fortsetzung nicht abgeschlossen hat („¬pl“) (16).
- der Nutzer eine Bedienaktion (außer Skip) durchführt (13); Der Player wird aufgefordert fortzusetzen („cnt“).
- der Nutzer die Bedienaktion Skip durchführt (14); Der Player wird aufgefordert, den nächsten Titel zu spielen („nxt“).

Zustand „NPFI“. Wenn der Zustand „NPBI“ aktiviert ist, hat der Player seine Fortsetzung nicht abgeschlossen („¬pl“) und der Dialog ist im Zustand „fokussiert“. Der Zustand „NPFI“ wird aktiviert, wenn

- der Player seine Fortsetzung nicht abgeschlossen hat („¬pl“) (8).
- der Nutzer die Bedienaktion Timeout durchführt (5); Der Player wird aufgefordert fortzusetzen („cnt“).
- der Nutzer die Bedienaktion Skip durchführt (6); Der Player wird aufgefordert, den nächsten Titel zu spielen („nxt“).
- der Nutzer die Bedienaktion Menu durchführt (7); Der Player wird aufgefordert fortzusetzen und die Variable „bac“ wird mit dem Wert Falsch belegt.

A.2 Tabellen der Modi-Konfigurationen

Im Folgenden sind die Tabellen der Modi-Konfigurationen angegeben. Für die Modi-Konfigurationen 1–7 s. Tab. A.9; für 8–14 s. Tab. A.10 und für 15–20 s. Tab. A.11.

Modi-Konf.	Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
1	Timeout		●	○		●	○		○	●	
2	Skip	●	●	○	●	●	○	●	○	○	●
2	Select	●	●	○	●	●	○	●	○	●	
2	Menu	●	●	○	●	●	○	●	○	○	●
3	Timeout	●	○	●	●	●	○	●	○	○	●
3	Select	●	○	○	●	○	○	●	○	●	
4	Plus	○	○	○	○	○	○	○	○	○	●
4	Skip	●	○	●	●	●	○	○	●	●	
4	Timeout	○	○	●	○	●	○	●	○	●	
4	Select	●	○	○	●	○	○	●	○	●	
5	Plus	○	○	○	○	○	○	○	○	○	●
5	Skip	○	○	●	○	●	○	○	●	○	●
5	Timeout	○	○	●	○	●	○	●	○	●	
5	Select	●	○	○	●	○	○	●	○	●	
6	Minus	○	○	○	○	○	○	○	○	○	●
6	Skip	●	○	●	●	●	○	○	●	●	
6	Timeout	○	○	●	○	●	○	●	○	●	
6	Select	○	○	●	○	●	○	●	○	○	●
7	Minus	○	○	○	○	○	○	○	○	○	●
7	Skip	○	○	●	○	●	○	○	●	○	●
7	Timeout	○	○	●	○	●	○	●	○	●	
7	Select	○	○	●	○	●	○	●	○	○	●

Tabelle A.9: Modi-Konfigurationen 1–7

Modi-Konf.	Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
8	Menu	○	○	●	○	●	○	●	○	○	●
8	Plus	○	○	○	○	○	○	○	○	●	●
8	Timeout	●	○	●	●	●	○	●	○	○	●
8	Select	●	●	○	●	●	○	●	●	○	●
9	Menu	○	○	●	○	●	○	●	○	○	●
9	Plus	○	○	○	○	○	○	○	○	●	●
9	Skip	●	○	●	●	●	○	○	●	●	
9	Timeout	○	○	●	○	●	○	●	○	●	
9	Select	●	●	●	●	●	○	●	●	○	●
10	Menu	○	○	●	○	●	○	●	○	○	●
10	Plus	○	○	○	○	○	○	○	○	●	●
10	Skip	○	○	●	○	●	○	○	●	○	●
10	Timeout	○	○	●	○	●	○	●	○	●	
10	Select	●	●	●	●	●	○	●	●	○	●
11	Menu	○	○	●	○	●	○	●	○	○	●
11	Plus	○	○	○	○	○	○	○	○	●	●
11	Skip	●	○	●	●	●	○	○	●	●	●
11	Timeout	○	○	●	○	●	○	●	○	●	
11	Select	○	○	●	○	●	○	●	○	○	●
12	Menu	○	○	●	○	●	○	●	○	○	●
12	Plus	○	○	○	○	○	○	○	○	○	●
12	Minus	○	○	○	○	○	○	○	○	○	●
12	Timeout	●	○	●	●	●	○	●	○	○	●
12	Select	●	●	○	●	●	○	●	●	○	●
13	Menu	○	○	●	○	●	○	●	○	○	●
13	Plus	○	○	○	○	○	○	○	○	○	●
13	Minus	○	○	○	○	○	○	○	○	○	●
13	Skip	●	○	●	●	●	○	○	●	●	●
13	Timeout	○	○	●	○	●	○	●	○	●	
13	Select	●	●	●	●	●	○	●	●	○	●
14	Menu	○	○	●	○	●	○	●	○	○	●
14	Plus	○	○	○	○	○	○	○	○	○	●
14	Minus	○	○	○	○	○	○	○	○	○	●
14	Skip	○	○	●	○	●	○	○	●	○	●
14	Timeout	○	○	●	○	●	○	●	○	●	
14	Select	●	●	●	●	●	○	●	●	○	●

Tabelle A.10: Modi-Konfigurationen 8–14

Modi-Konf.	Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
15	Menu	○	○	●	○	●	○	●	○	○	●
15	Plus	○	○	○	○	○	○	○	○	○	●
15	Minus	○	○	○	○	○	○	○	○	○	●
15	Skip	●	○	●	●	●	○	○	●	●	
15	Timeout	○	○	●	○	●	○	●	○	●	
15	Select	○	○	●	○	●	○	●	○	○	●
16	Menu	○	○	●	○	●	○	●	○	○	●
16	Minus	○	○	○	○	○	○	○	○	●	●
16	Timeout	●	○	●	●	●	○	●	○	○	●
16	Select	●	●	○	●	●	○	●	●	○	●
17	Menu	○	○	●	○	●	○	●	○	○	●
17	Minus	○	○	○	○	○	○	○	○	●	●
17	Skip	●	○	●	●	●	○	○	●	●	
17	Timeout	○	○	●	○	●	○	●	○	●	
17	Select	●	●	●	●	●	○	●	●	○	●
18	Menu	○	○	●	○	●	○	●	○	○	●
18	Minus	○	○	○	○	○	○	○	○	●	●
18	Skip	○	○	●	○	●	○	○	●	○	●
18	Timeout	○	○	●	○	●	○	●	○	●	
18	Select	○	○	●	○	●	○	●	○	○	●
19	Menu	●	○	○	●	○	○	●	○		
19	Skip	●	○	●	●	●	○	○	●	●	
19	Timeout	○	○	○	○	○	○	○	○	●	
20	Menu	●	○	○	●	○	○	●	○		
20	Skip	●	○	●	●	●	○	●	●	○	●
20	Timeout	○	○	○	○	○	○	○	○	●	

Tabelle A.11: Modi-Konfigurationen 15–20

A.3 Übergänge zwischen den Modi-Konfigurationen

Abb. A.8 zeigt die Übergänge zwischen den Modi-Konfigurationen. Tab. A.12 gibt zusätzlich die Bedienaktionen an, die die Übergänge bewirken.

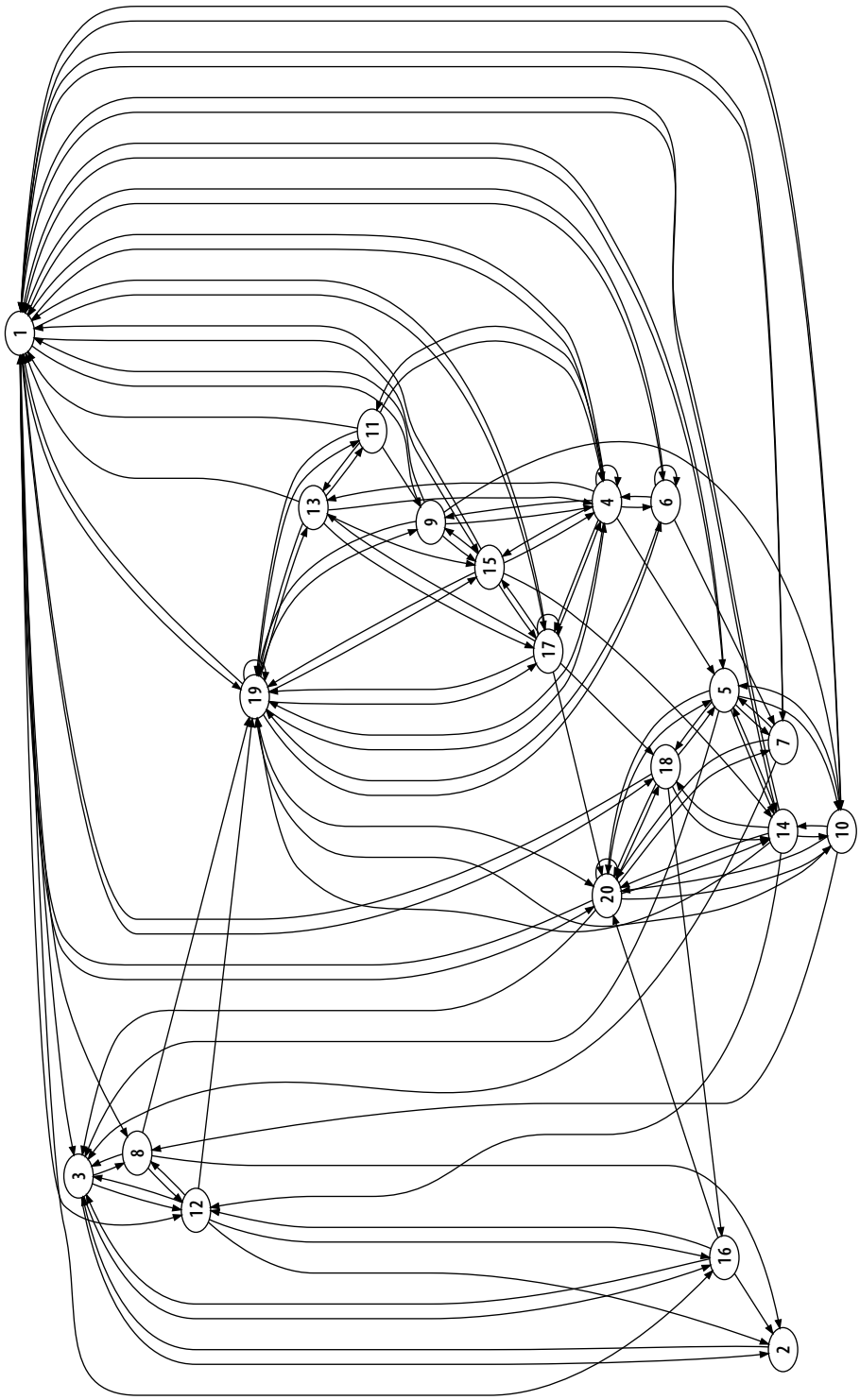


Abbildung A.8: Übergänge zwischen den Modi-Konfigurationen

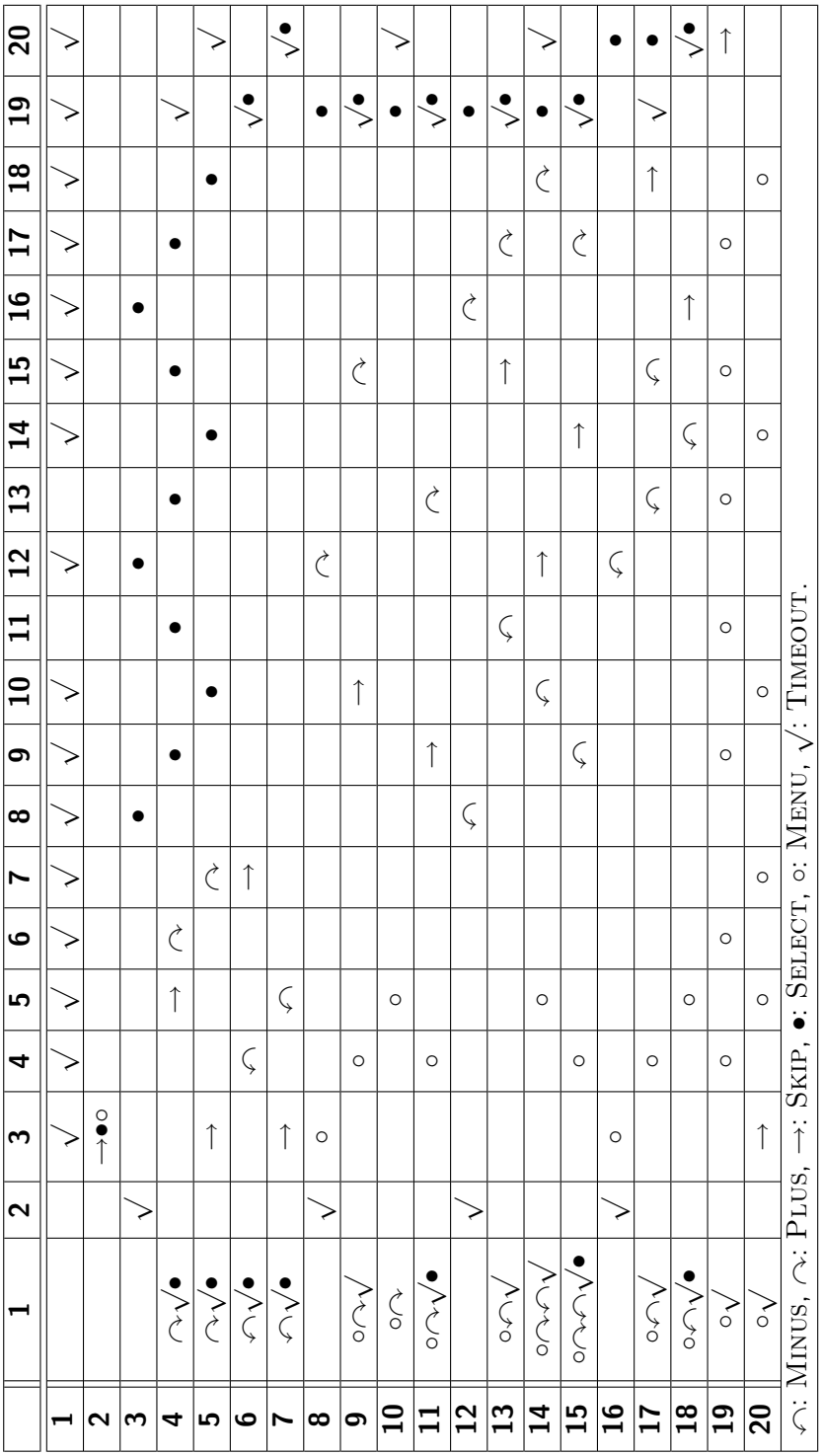


Tabelle A.12: Mit Bedienaktionen markierte Übergänge

B Details zur Fallstudie „E60“

Dieser Anhang beschreibt Details zur Fallstudie „E60“ aus Kapitel 5. Es werden die Zustandsübergangsdiagramme sowie Tabellen, die die Übergänge spezifizieren, für das System angegeben. Im Anschluss daran werden die Modi-Konfigurationen mit Hilfe von Tabellen beschrieben. Zuletzt wird der Übergangsgraph der Modi-Konfigurationen angegeben.

B.1 Zustandsübergangsdiagramme

Im Folgenden wird das Verhalten des Systems spezifiziert.

Bei der Spezifikation der Übergänge wird für die Bedienaktionen (Nil, Up, Down, Left, Right, LongRight, Press, Menu oder Timeout) die folgende Schreibweise verwendet: In den Vorbedingungen wird, beispielsweise für die Bedienaktion Up, statt „$b = \text{Up}$“ nur „Up“ geschrieben und in den Nachbedingungen wird statt „$b' = \text{Up}$“ ebenfalls nur „Up“ angegeben, wobei b der Kanal der Bedienaktionen ist.

Beim Verschicken der Rückmeldung, prüft der Mediator, ob eine Applikationsmeldung, signalisiert durch die Variable msg, vorliegt. Wenn eine Meldung vorliegt, wird der gegenwärtige Dialog im nächsten Schritt unterbrochen, d. h. es wird die Modi-Konfiguration 1 zurückgegeben und die Variable, die die Unterbrechung signalisiert, gesetzt. Sei k eine Modi-Konfiguration und f eine Variable, dann gilt

$$\text{Ret}(k,\, i) \quad \stackrel{\text{def}}{=} \quad (\neg msg \ \wedge \ r' = k \ \wedge \ \neg i') \vee (msg \ \wedge \ r' = 1 \ \wedge \ i'),$$

wobei r der Kanal der Rückmeldungen ist.

B.1.1 Orthogonaler Zustand „User“

Abb. B.1 zeigt das Zustandsübergangsdiagramm für die Komponente „User“. Die Übergänge sind in der Tabelle B.1 festgelegt. Der orthogonale Zustand „User“ beschreibt das Verhalten des Nutzers in Reaktion auf eine Rückmeldung des Mediators. Die möglichen Bedienaktionen werden nichtdeterministisch gewählt. Der Übergang 16 zeigt die Verwendung der impliziten Bedienaktion Timeout. Dadurch kann beibehalten werden, dass der Mediator nicht selbständig agiert, sondern auf auf Bedienaktionen oder Applikationsnachrichten reagiert. Durch die Rückmeldung teilt der Mediator dem Nutzer mit, welche Modi-Konfiguration vorliegt. Der Nutzer führt dann eine derjenigen Bedienaktionen durch, die für diese Konfiguration definiert sind.

User		
in $r : M$ out $b : B$		
0	–	Nil
1	$r = 0$	Nil
2	$r = 1$	Timeout
3	$r = 2$	Press ∨ Up ∨ Right ∨ Menu ∨ LongRight
4	$r = 3$	Press ∨ Up ∨ Down ∨ Right ∨ Menu ∨ LongRight
5	$r = 4$	Press ∨ Down ∨ Menu ∨ Right ∨ LongRight
6	$r = 5$	Press ∨ Up ∨ Down ∨ Right ∨ Menu ∨ LongRight
7	$r = 6$	Press ∨ Up ∨ Down ∨ Right ∨ Menu ∨ LongRight
8	$r = 7$	Press ∨ Up ∨ Right ∨ Menu ∨ LongRight
9	$r = 8$	Press ∨ Up ∨ Down ∨ Menu ∨ Right ∨ LongRight
10	$r = 9$	Press ∨ Down ∨ Menu ∨ Right ∨ LongRight
11	$r = 10$	Press ∨ Down ∨ Menu ∨ Right ∨ LongRight
12	$r = 11$	Press ∨ Up ∨ Down ∨ Right ∨ Menu ∨ LongRight
13	$r = 12$	Up ∨ Press ∨ Right ∨ Menu ∨ LongRight
14	$r = 13$	Press ∨ Up ∨ Right ∨ Menu ∨ LongRight
15	$r = 14$	Press ∨ Up ∨ Right ∨ Menu ∨ LongRight
16	$r = 15$	Press ∨ Up ∨ Down ∨ Right ∨ Left ∨ Timeout ∨ Menu ∨ LongRight
17	$r = 16$	Press ∨ Right ∨ Menu ∨ LongRight
18	$r = 17$	Press ∨ Up ∨ Down ∨ Right ∨ Left ∨ Menu ∨ LongRight
19	$r = 18$	Press ∨ Up ∨ Down ∨ Right ∨ Left ∨ Menu ∨ LongRight
20	$r = 19$	Left ∨ Menu ∨ LongRight
21	$r = 20$	Right ∨ LongRight
22	$r = 21$	Press ∨ Down
23	$r = 22$	Up
24	$r = 23$	Press ∨ Up ∨ Down ∨ Menu ∨ Right ∨ LongRight
25	–	Nil

Tabelle B.1: Übergangstabelle für „User“

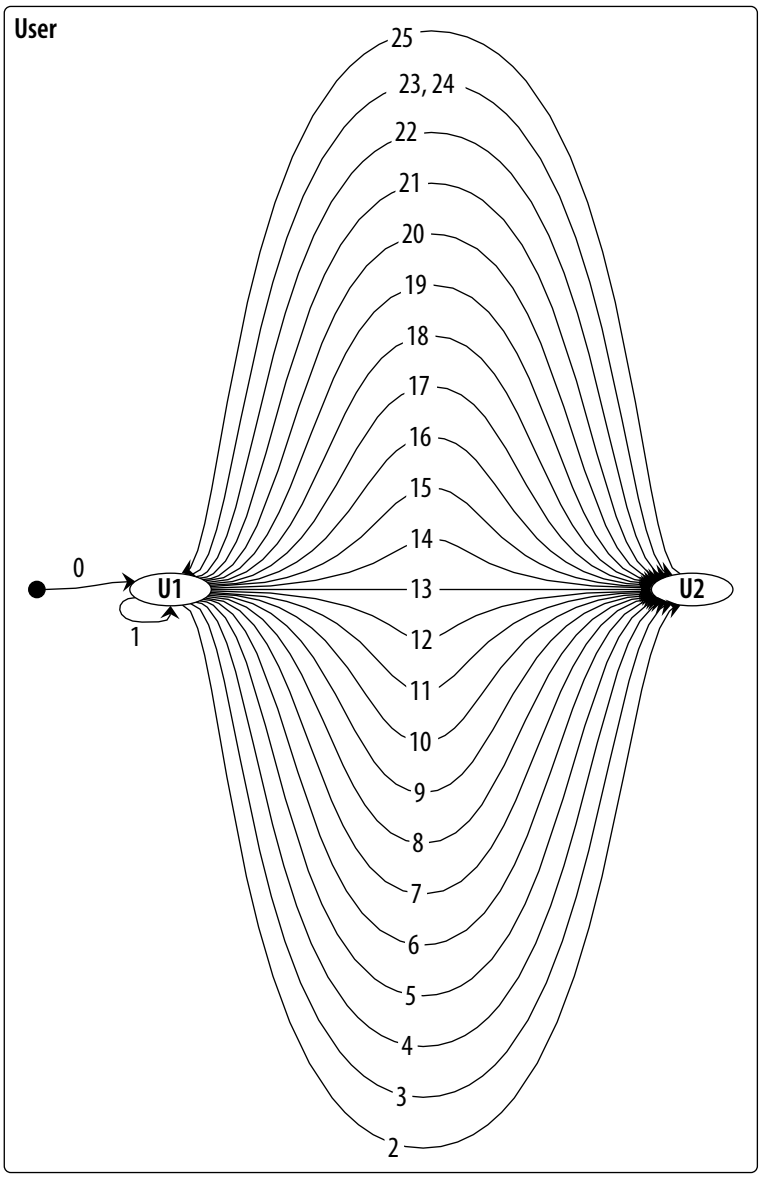


Abbildung B.1: Diagramm für „User“

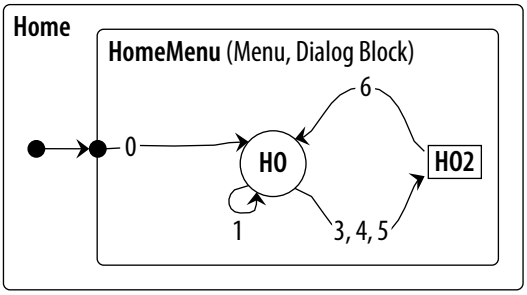


Abbildung B.2: Diagramm für „Home“

B.1.2 Orthogonaler Zustand „Home“

Abb. B.2 zeigt das Diagramm der Komponente „Home“. Die orthogonalen Zustände des Mediators laufen quasiparallel ab. Der orthogonale Zustand „Home“ ist in diesem Zusammenspiel der Initiale und besitzt somit zu Beginn den Fokus. Er stellt zudem einen Dialogstrang dar.

HomeMenu		
in-out $int1$: Bool in b : B; msg : Bool out r : M		
0	–	$r' = 20$
1	Nil	$r' = 0$
3	$\neg int1 \wedge$ Right	restart(Navigation) $\wedge\ r' = 0$
4	$\neg int1 \wedge$ LongRight	resume(Navigation) $\wedge\ r' = 0$
5	$int1 \wedge \neg$Nil	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int1'$
6	resume $\wedge$ Nil	run $\wedge$ Ret(20, $int1$)

Tabelle B.2: Übergangstabelle für „HomeMenu“

Orthogonaler Zustand „HomeMenu“. Abb. B.2 zeigt das Diagramm des Zustands „HomeMenu“. Die Übergänge sind in Tabelle B.2 beschrieben. Der zusammengesetzte Zustand „HomeMenu“ stellt einen Dialogblock und das Menü „HomeMenu“ dar. Die Komponente gibt die Rückmeldung 20 aus (0) und wartet auf eine Bedienaktion des Nutzers (1). Reagiert der Nutzer mit der Aktion Right, wird der Fokus an die Komponente „Navigation“ übergeben. Dabei erhält die Komponente „Navigation“ die Auforderung, einen Neustart durchzuführen (3). „Neustart“ bedeutet in diesem Fall, dass alle geöffneten Dialog geschlossen werden und ein neuer Dialog geöffnet wird. Reagiert der Nutzer mit der Aktion LongRight, wird der Fokus ebenfalls an die Komponente „Navigation“ übergeben. Allerdings führt diese ihren Ablauf, in diesem Fall, ohne Neustart durch (4). Signalisiert die Applikation eine Meldung, dann wird der Fokus an die Komponente „CheckControlMessage“ übergeben (5). Diese Komponente stellt keinen Dialogstrang dar. Somit wird der Dialog „CheckControlMessage“ als Unterdialog des Dialogs „HomeMenu“ aktiviert. Befindet sich die Komponente

„Home“ nicht im Besitz des Fokus, wird, wenn diese zur Fortsetzung aufgefordert wird, die Rückmeldung 20 ausgegeben (6).

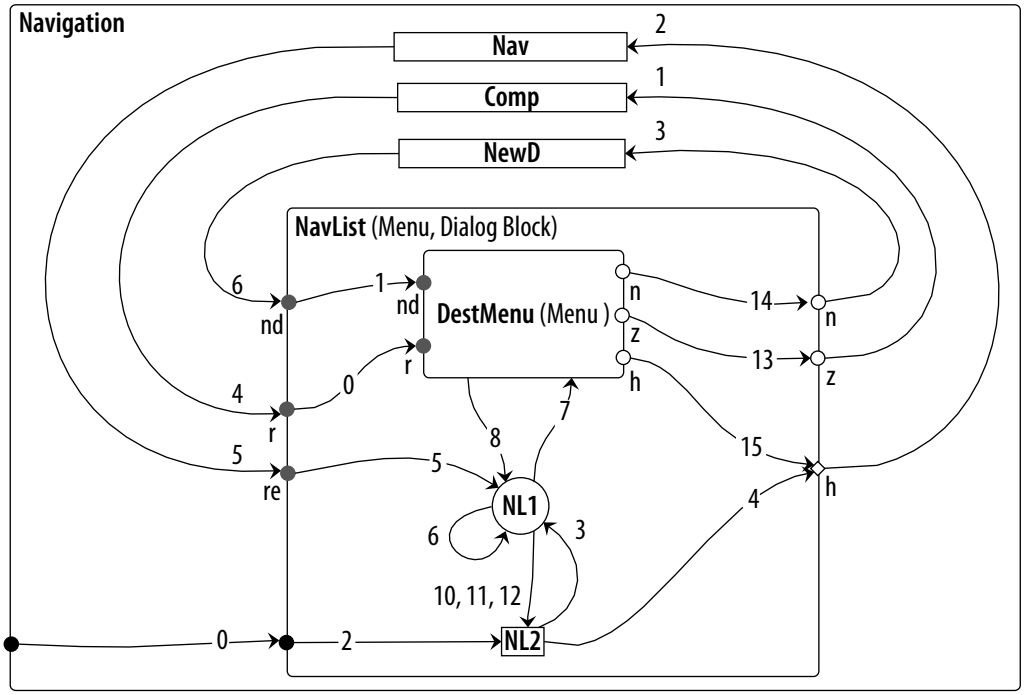


Abbildung B.3: Diagramm für „Navigation“ und „NavList“

Navigation		
local $int3$: Bool in $b : B$; msg, lo, le, fl, st : Bool out $r : M$; $setd$, al, dl, sg, ad, rl, dg : Bool		
0	–	–
1–3	–	$r' = 0$
4–6	Nil	–

Tabelle B.3: Übergangstabelle für „Navigation“

B.1.3 Orthogonaler Zustand „Navigation“

Abb. B.3 zeigt das Zustandsübergangsdiagramm der Komponente „Navigation“. Die Übergänge sind in Tabelle B.3 beschrieben. Der orthogonale Zustand „Navigation“ des Mediators stellt einen Dialogstrang dar. Zu Beginn besitzt dieser nicht den Fokus. Die Eintrittstransition aktiviert den Unterzustand „NavList“ (0).

Die restlichen Übergänge beschreiben das Verhalten des Dialogstrangs bei einem Schließen bzw. einem Abbruch des Dialogs „NavList“. Es sind zwei Fälle zu unterscheiden: „NavList“ wird abgebrochen (2) und im folgenden Schritt wird ein neuer Dialog geöffnet (5); „NavList“ wird geschlossen (1, 3) und im folgenden Schritt wird ein neuer Dialog geöffnet (4, 6). Über die Austrittsstellen „z“ und „n“ wird unterschieden, ob die Zielführung aktiviert worden ist, oder ein Ziel zur Zielliste hinzugefügt worden ist.

NavList		
in-out $int3$: Bool in b : B; msg, lo, le, fl, st : Bool out r : M; $setd$, al, dl, sg, ad, rl, dg : Bool		
0–2	–	–
3	resume $\wedge$ NIL	run $\wedge$ RET(10, $int3$)
4	restart $\wedge$ NIL	run
5	–	RET(10, $int3$)
6	NIL	$r' = 0$
7	$\neg int3 \wedge$ (PRESS $\vee$ DOWN)	–
8	$\neg int3 \wedge$ UP	RET(10, $int3$)
10	$\neg int3 \wedge$ RIGHT	resume(AssistanceWindow) $\wedge\ r' = 0$
11	$\neg int3 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0$
12	$int3 \wedge \neg$NIL	resume(CheckControlMsg.) $\wedge\ r' = 0 \wedge \neg int3'$
13–15	–	–

Tabelle B.4: Übergangstabelle für „NavList“

Zusammengesetzter Zustand „NavList“. Die Übergänge des zusammengesetzten Zustands „NavList“ sind in Tabelle B.4 angegeben. Der Zustand repräsentiert einen Dialogblock und das Menü „NavList“. Das heißt, die Eintrittsaktion öffnet eine Reihe von Dialogen und aktiviert das Menü „NavList“. Konzeptuell steht die Eintrittsaktion für das Öffnen der Dialoge „NavList“, „MapList“ und „DestList“; das Menü „NavList“ wird aktiviert, besitzt aber nicht den Fokus (2). Wird der Dialogstrang fortgesetzt, d. h. das Menü „NavList“ bekommt den Fokus, wird die Rückmeldung 10 ausgegeben (4). Führt der Nutzer als Reaktion die Bedienaktion PRESS oder DOWN durch, wird eine Aktion des Unterzustands „DestMenu“ ausgeführt (7). Führt der Nutzer die Bedienaktion RIGHT durch, wird der Fokus der Komponente „AssistanceWindow“ übergeben (10). Dies bedeutet konzeptuell, dass das Menü „NavList“ in den Zustand „bereit“ wechselt. Die Komfortfunktion, ausgelöst durch die Bedienaktion LONGRIGHT, lässt den Zustand des Dialogstrangs unverändert und gibt den Fokus an den orthogonalen Zustand „Home“ (11). Denselben Effekt erzielt die Bedienaktion MENU. Die Tatsache, dass auf die Aktion MENU zunächst ein Wechsel des Dialogstrangs erfolgt und der Dialogstrang bei der Fortsetzung neu gestartet wird, wird in Kapitel 5 diskutiert.

Zusammengesetzter Zustand „DestMenu“. Abb. B.4 zeigt das Diagramm des zusammengesetzten Zustands „DestMenu“. In Tabelle B.5 sind die Übergänge angegeben. Der Zustand repräsentiert das Menü „DestMenu“. Dieses Menü besteht aus den beiden Optionsgruppen „DestList“ und „MapList“. Im Normalfall wird die Optionsgruppe „DestList“ aktiviert (0). Wenn ein Folgedialog geöffnet werden soll, werden drei Fälle unterschieden:

1. Ist die Zielführung geändert worden (Stelle „r“), und die Zielführung ist nicht aktiv ist, wird ebenfalls die Optionsgruppe „DestList“ aktiviert (1);
2. ist jedoch die Zielführung aktiv, wird die Optionsgrp. „MapList“ aktiviert (4).

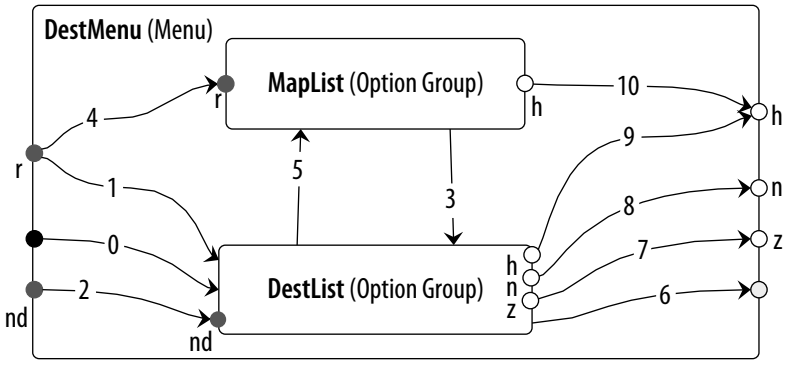


Abbildung B.4: Diagramm für „DestMenu“

DestMenu		
in-out $int3$: Bool in b : B; msg : Bool out r : M; $setd$, al, dl, sg, ad, rl, dg : Bool		
0	–	–
1	in(GOff)	–
2	–	–
3	$\neg int3 \wedge$ Up	–
4	¬in(GOff)	–
5	$\neg int3 \wedge$ Down	–
6–10	–	–

Tabelle B.5: Übergangstabelle für „DestMenu“

3. Wenn ein Dialog als Folge des Hinzufügens eines Ziels geöffnet werden soll (Stelle „nd“), wird die die Optionsgruppe „DestList“ aktiviert.

Die Übergänge 3 und 5 erlauben das Springen zwischen den Optionsgruppen.

Zusammengesetzter Zustand „DestList“. Abb. B.5 zeigt das Diagramm des zusammengesetzten Zustands „DestList“. Die Übergänge sind in Tabelle B.6 spezifiziert. Der Zustand „DestList“ repräsentiert eine Optionsgruppe und gibt bei seiner Aktivierung die Rückmeldung 11 aus (1). Die Optionsgruppe bietet dem Nutzer eine Menge von Zielen an. Welches Ziel ausgewählt ist, wird nicht direkt modelliert. Führt der Nutzer die Bedienaktion Press durch, wird nichtdeterministisch eine der im Folgenden beschriebenen Aktionen ausgeführt:

- Das Menü „SelectInputMenu“ wird geöffnet und aktiviert (9). Das stellt die Aktivierung der Option „neues Ziel“ dar.

- Wenn die Liste der Ziele nicht leer ist, wird das Menü „PullMenu“ für ein Ziel, zu dem nicht geführt wird, geöffnet und aktiviert (10). Das stellt die Aktivierung einer Option, die für ein Ziel steht und nicht markiert ist, dar.

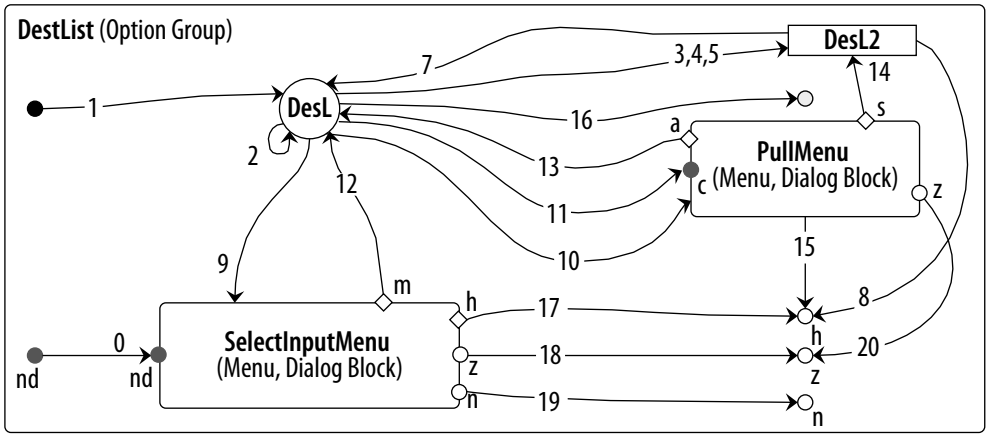


Abbildung B.5: Diagramm für „DestList“

- Wenn die Liste der Ziel nicht leer ist und die Zielführung aktiv ist, wird das Menü „PullMenu“ für das Ziel, zu dem geführt wird, geöffnet und aktiviert (11). Das stellt die Aktivierung einer markierten Option dar.

Bietet der Unterzustand „SelectInputMenu“ über die Stelle „m“ seine Deaktivierung an, kann der Nutzer durch die Bedienaktion Press das Menü „DestList“ aktivieren (12).

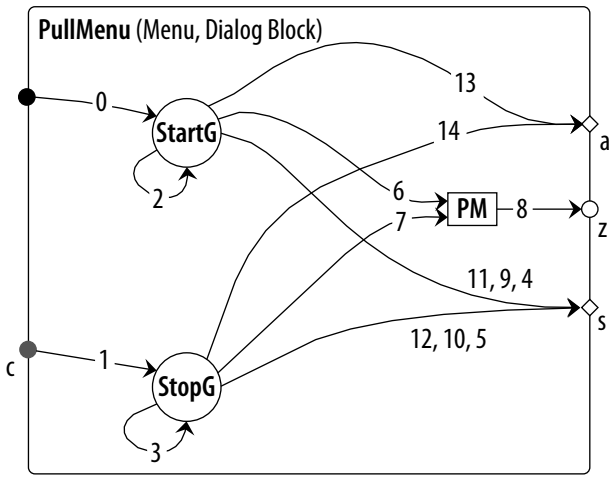


Abbildung B.6: Diagramm für „PullMenu“

Zusammengesetzter Zustand „PullMenu“. Abb. B.6 zeigt das Diagramm des zusammengesetzten Zustands „PullMenu“. Tabelle B.7 beschreibt die Übergänge. Der Zustand repräsentiert einen Dialogblock und das Menü „PullMenu“. Der Zustand kann über zwei Stellen aktiviert werden und gibt in beiden Fällen die Rückmeldung 15 aus (0, 1). Das Menü unterscheidet, ob zu demjenigen Ziel, für das das Menü geöffnet worden ist, gegenwärtig geführt wird oder nicht. Der Unterschied ist bei der Bedienaktion Press des Nutzers sichtbar: Wird zu dem Ziel geführt, wird das Signal „dg“ ausgegeben, d. h. die Zielführung wird abgebrochen; wird zu dem Ziel nicht

DestList		
in-out $int3$: Bool in b : B; msg, lo, le, fl, st : Bool out r : M; $setd$, al, dl, sg, ad, rl, dg : Bool		
0	–	–
1	–	Ret(11, $int3$)
2	Nil	$r' = 0$
3	$int3 \wedge \neg$Nil	resume(CheckControlMsg.) $\wedge\, r' = 0 \wedge \neg int3'$
4	$\neg int3 \wedge$ (Menu $\vee$ LongRight)	resume(Home) $\wedge\, r' = 0$
5	$\neg int3 \wedge$ Right	resume(AssistanceWindow) $\wedge\, r' = 0$
7	resume $\wedge$ Nil	run $\wedge$ Ret(11, $int3$)
8	restart $\wedge$ Nil	run
9	$\neg int3 \wedge$ Press	–
10	$\neg int3 \wedge$ Press $\wedge \neg$in(EmptyDL)	–
11	$\neg int3 \wedge$ Press $\wedge \neg$in(EmptyDL) $\wedge \neg$in(GOff)	–
12	$\neg int3 \wedge$ Press	Ret(11, $int3$)
13	–	Ret(11, $int3$)
14	–	$r' = 0$
15–20	–	–

Tabelle B.6: Übergangstabelle für „DestList“

geführt, wird das Signal „sg“ ausgegeben, d. h. die Zielführung wird gestartet. Das Menü „PullMenu“ ist ein *flüchtiges Menü*. Das bedeutet, es kann weder im Zustand „deaktiviert“ noch im Zustand „bereit“ sein. Deswegen wird bei einer Unterbrechung (der Fokus wird entzogen (9–12) bzw. das Menü wird deaktiviert (13, 14)) der Dialog abgebrochen. Dies kann über die zwei Stellen „a“ oder „s“ erfolgen: Über die Stelle „a“, wenn das Menü deaktiviert und das übergeordnete Menü reaktiviert wird; über die Stelle „s“, wenn dem Menü der Fokus genommen wird. Nachdem der Dialog entweder die Zielführung abgebrochen oder gestartet hat, wird der Dialog über die Stelle „z“ geschlossen (8).

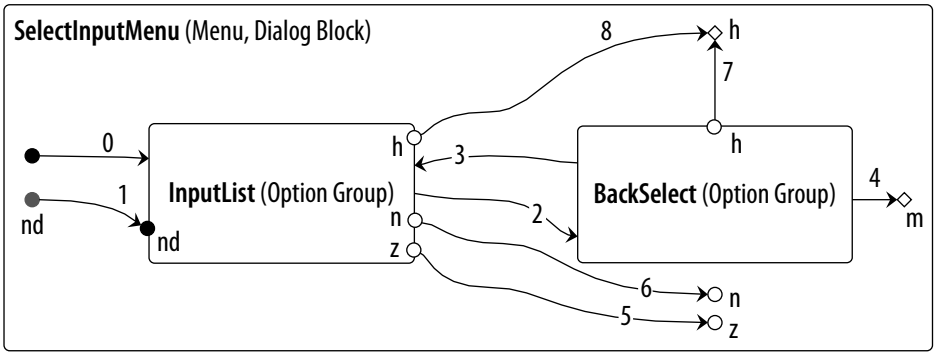


Abbildung B.7: Diagramm für „SelectInputMenu“

PullMenu		
in-out $int3$: Bool in b : B; msg : Bool out r : M; sg, dg : Bool		
0,1	–	RET(15, $int3$)
2,3	NIL	$r' = 0$
4,5	$int3 \wedge \neg$NIL	resume(CheckControlMsg.) $\wedge\ r' = 0 \wedge \neg int3'$
6	$\neg int3 \wedge$ PRESS	$r' = 0 \wedge sg'$
7	$\neg int3 \wedge$ PRESS	$r' = 0 \wedge dg'$
8	–	–
9,10	$\neg int3 \wedge$ RIGHT	resume(AssistanceWindow) $\wedge\ r' = 0$
11,12	$\neg int3 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0$
13,14	$\neg int3 \wedge$ (LEFT $\vee$ UP $\vee$ DOWN $\vee$ TIMO.)	–

Tabelle B.7: Übergangstabelle für „PullMenu“

SelectInputMenu		
in-out $int3$: Bool in b : B; msg, lo, le, fl, st : Bool out r : M; $setd$, al, dl, ad, rl, sg : Bool		
0,1	–	–
2	$\neg int3 \wedge$ UP	–
3	$\neg int3 \wedge$ DOWN	–
4–8	–	–

Tabelle B.8: Übergangstabelle für „SelectInputMenu“

Zusammengesetzter Zustand „SelectInputMenu“. Das Diagramm des zusammengesetzten Zustands „SelectInputMenu“ ist in Abb. B.7 dargestellt. Tabelle B.8 beschreibt die Übergänge. Der Zustand repräsentiert einen Dialogblock sowie das Menü „SelectInputMenu“. Dieses Menü besteht aus den Optionsgruppen „InputList“ und „BackSelect“. Wenn die Umgebung „SelectInputMenu“ aktiviert, wird auch „InputList“ aktiviert (0, 1). Die Übergänge 2 und 3 ermöglichen das Springen zwischen den Optionsgruppen. Wenn die Unteroptionsgruppe „BackSelect“ die Kontrolle über die Standard-Stelle zurückgibt, werden die Dialoge des Blocks abgebrochen und der Zustand „SelectInputMenu“ deaktiviert (4). Wenn die Unteroptionsgruppe „InputList“ die Kontrolle über die Stellen „n“ oder „z“ zurückgibt, werden die Dialoge des Blocks geschlossen und der Zustand „SelectInputMenu“ deaktiviert (5, 6).

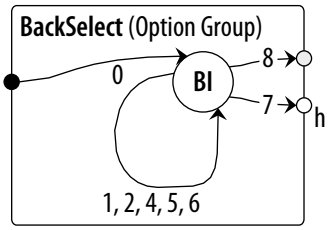


Abbildung B.8: Diagramm für „BackSelect“

BackSelect		
in-out $int3$: Bool in b : B; msg : Bool out r : M		
0	–	RET(9, $int3$)
1	$int3 \wedge \neg$NIL	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
2	NIL	$r' = 0$
4	$\neg int3 \wedge$ RIGHT	resume(AssistanceWindow) $\wedge\ r' = 0$
5	$\neg int3 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0$
6	resume $\wedge$ NIL	run $\wedge$ RET(9, $int3$)
7	restart $\wedge$ NIL	run
8	–	–

Tabelle B.9: Übergangstabelle für „BackSelect“

Zusammengesetzter Zustand „BackSelect“. Abb. B.8 zeigt das Diagramm des zusammengesetzten Zustands „BackSelect“. Die Übergänge sind in Tabelle B.9 beschrieben. Der Zustand „BackSelect“ stellt eine Optionsgruppe dar und gibt bei seiner Aktivierung die Rückmeldung 9 aus (0). Der Übergang 8 gibt die Kontrolle an den übergeordneten Zustand zurück. Die restlichen Übergänge stellen das übliche Verhalten bei den Bedienaktionen des Nutzers dar.

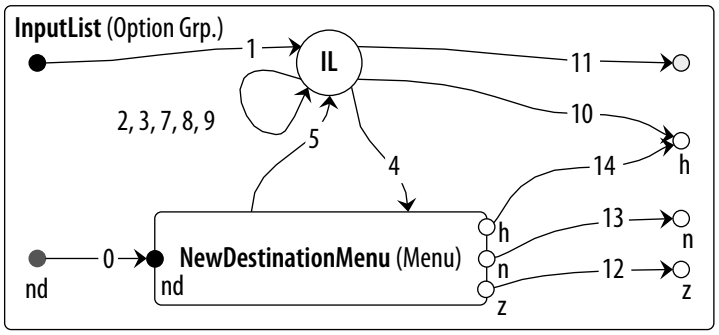


Abbildung B.9: Diagramm für „InputList“

Zusammengesetzter Zustand „InputList“. Das Diagram des Zustands „InputList“ wird in Abb. B.9 dargestellt. Tabelle B.10 beschreibt die Übergänge. Der Zustand „InputList“ stellt eine Optionsgruppe dar und gibt bei seiner Aktivierung die Rückmeldung 8 aus (1). Durch die Bedienaktionen PRESS oder DOWN kann der Nutzer das Untermenü „NewStreetMenu“ aktivieren (4). Ist das Untermenü „NewStreetMenu“ aktiviert und der Nutzer führt die Bedienaktion UP durch, wird es deaktivert, der Zustand „InputList“ reaktiviert und die Rückmeldung „8“ ausgegeben (5). Der Übergang 11 gibt die Kontrolle an den übergeordneten Zustand.

Zusammengesetzter Zustand „NewDestinationMenu“. Das Diagramm des zusammengesetzten Zustands „NewDestinationMenu“ ist in Abb. B.10 dargestellt. Die

InputList		
in-out $int3$: Bool in b : B; msg, lo, le, fl, st : Bool out r : M; $setd$, al, dl, ad, rl, sg : Bool		
0	–	–
1	–	Ret(8, $int3$)
2	Nil	$r' = 0$
3	$int3 \wedge \neg$Nil	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
4	$\neg int3 \wedge$ (Press $\vee$ Down)	–
5	$\neg int3 \wedge$ Up	Ret(8, $int3$)
7	$\neg int3 \wedge$ Right	resume(AssistanceWindow) $\wedge\ r' = 0$
8	$\neg int3 \wedge$ (Menu $\vee$ LongRight)	resume(Home) $\wedge\ r' = 0$
9	resume $\wedge$ Nil	run $\wedge$ Ret(8, $int3$)
10	restart $\wedge$ Nil	run
11–14	–	–

Tabelle B.10: Übergangstabelle für „InputList“

Übergänge sind in Tabelle B.11 angegeben. Der Zustand „NewDestinationMenu“ stellt ein Menü dar. Der Nutzer kann drei verschiedene Optionen mit diesem Menü anstoßen:

- Er kann das aktuelle Ziel ändern, indem er eine neue Straße festlegt;
- Er kann das gegenwärtig gewählte Ziel zur Zielliste hinzufügen („Zur Zielliste hinzufügen“);
- Er kann die Zielführung für das gegenwärtig gewählte Ziel starten („Zielführung starten“).

Diese Optionen sind durch die drei Prompts „NStr“, „AddD“ und „StG“ modelliert. Die Rückmeldungen dieser Prompts sind unterschiedlich, da die Wirkung der Bedienaktion Press jeweils unterschiedlich ist. Die Optionen können nur dann aktiviert werden, wenn die jeweiligen Vorbedingungen erfüllt sind. Die Option „StG“ erfordert, dass ein gegenwärtig gewähltes Ziel vorhanden ist (1, 14, 15). Wird die Option „StG“ aktiviert, wird das Signal „sg“ ausgegeben (37) und wenn das Ziel noch nicht in der Zielliste ist, auch das Signal „ad“ (38); im nächsten Schritt wird der Dialog über die Stelle „z“ geschlossen (39). Die Option „AddD“ erfordert, dass das gegenwärtig gewählte Ziel nicht bereits in der Zielliste vorhanden ist (2, 16, 17). Wird die Option „AddD“ aktiviert, dann wird das Signal „ad“ ausgegeben (40) und im nächsten Schritt der Dialog über die Stelle „n“ geschlossen (41). Die Option „NStr“ kann immer aktiviert werden (0, 12, 13). Wird die Option „NStr“ aktiviert, dann wird das Untermenü „NewStreetMenu“ aktiviert (36). Die Übergänge 33, 34, 35 geben die Kontrolle an den übergeordneten Zustand ab.

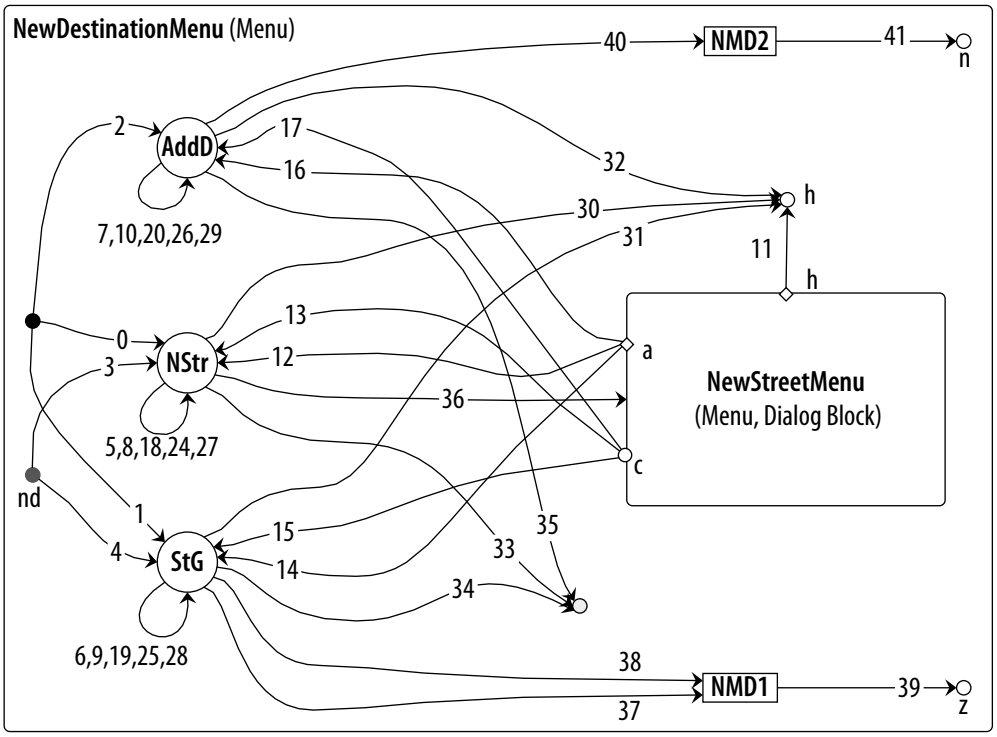


Abbildung B.10: Diagramm für „NewDestinationMenu“

NewDestinationMenu		
in-out $int3$: Bool in $b : B$; msg, lo, le, fl, st : Bool out $r : M$; $setd$, al, dl, ad, rl, sg : Bool		
0	–	$\textsc{Ret}(2, int3)$
1	$\neg$in(NoDest)	$\textsc{Ret}(14, int3)$
2	in(DestNotInList)	$\textsc{Ret}(13, int3)$
3	–	$\textsc{Ret}(2, int3)$
4	$\neg$in(NoDest)	$\textsc{Ret}(14, int3)$
5–7	$\textsc{Nil}$	$r' = 0$
8	$int3 \wedge \neg\textsc{Nil}$	resume(CheckControlMsg.) $\wedge\ r' = 0 \wedge \neg int3'$
9	$int3 \wedge \neg\textsc{Nil}$	resume(CheckControlMsg.) $\wedge\ r' = 0 \wedge \neg int3'$
10	$int3 \wedge \neg\textsc{Nil}$	resume(CheckControlMsg.) $\wedge\ r' = 0 \wedge \neg int3'$
11	–	–
12,13	–	$\textsc{Ret}(2, int3)$
14,15	$\neg$in(NoDest)	$\textsc{Ret}(14, int3)$
16,17	in(DestNotInList)	$\textsc{Ret}(13, int3)$
18–20	$\neg int3 \wedge \textsc{Right}$	resume(AssistanceWindow) $\wedge\ r' = 0$
24–26	$\neg int3 \wedge (\textsc{Menu} \vee \textsc{LongRight})$	resume(Home) $\wedge\ r' = 0$
27	resume $\wedge\ \textsc{Nil}$	run $\wedge\ \textsc{Ret}(2, int3)$
28	resume $\wedge\ \textsc{Nil}$	run $\wedge\ \textsc{Ret}(14, int3)$
29	resume $\wedge\ \textsc{Nil}$	run $\wedge\ \textsc{Ret}(13, int3)$
30–32	restart $\wedge\ \textsc{Nil}$	run
33–35	–	–
36	$\neg int3 \wedge \textsc{Press}$	–
37	$\neg int3 \wedge \textsc{Press} \wedge \neg$in(DestNotInList)	$r' = 0 \wedge sg'$
38	$\neg int3 \wedge \textsc{Press} \wedge$ in(DestNotInList)	$r' = 0 \wedge sg' \wedge ad'$
39	–	–
40	$\neg int3 \wedge \textsc{Press}$	$r' = 0 \wedge ad'$
41	–	–

Tabelle B.11: Übergangstabelle für „NewDestinationMenu“

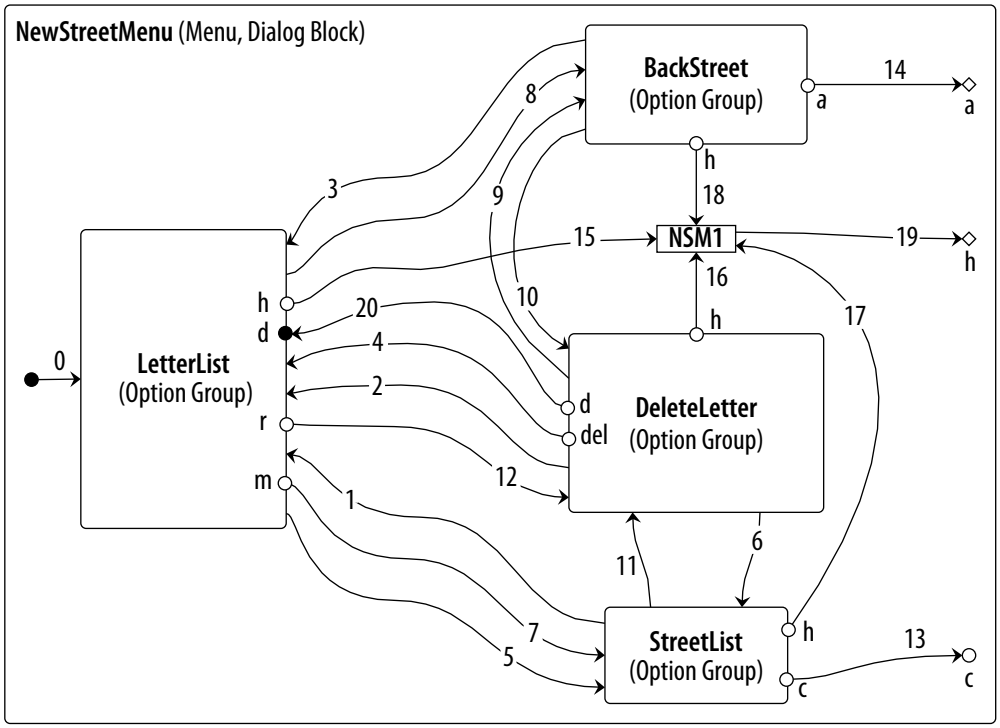


Abbildung B.11: Diagramm für „NewStreetMenu“

Zusammengesetzter Zustand „NewStreetMenu“. Abb. B.11 zeigt das Diagramm des Zustands „NewStreetMenu“. Die Übergänge sind in Tabelle B.12 beschrieben. Der zusammengesetzte Zustand „NewStreetMenu“ repräsentiert einen Dialogblock und das Menü „NewStreetMenu“. Dieses Menü besteht aus den Optionsgruppen „LetterList“, „DeleteLetter“, „StreetList“ und „BackStreet“. Bei der Aktivierung des Zustands „NewStreetMenu“, wird die Kontrolle an den Unterdialog „LetterList“ übergeben (0). Die Übergänge 1 bis 12 beschreiben die möglichen Wechsel zwischen den Optionsgruppen. Wenn der Unterzustand „BackStreet“ die Kontrolle über die Standard-Stelle abgibt, werden die Dialoge abgebrochen und die Kontrolle über die Stelle „a“ an den übergeordneten Zustand gegeben (14). Die Übergänge 15 bis 18 beschreiben das Verhalten, wenn in einer Optionsgruppe die Bedienaktion Menu ausgeführt worden ist: Die bereits spezifizierte Straße wird zurückgesetzt und im nächsten Schritt werden die Dialoge abgebrochen. Wenn der Unterzustand „StreetList“ die Kontrolle über die Stelle „c“ abgibt, bedeutet das, dass der Nutzer die aktuelle Straße übernehmen möchte. Die Dialoge werden dann geschlossen und die Kontrolle über die Stelle „c“ an den übergeordneten Zustand abgegeben.

Zusammengesetzter Zustand „LetterList“. Das Diagramm des Zustands „LetterList“ ist durch die Abb. B.12 gegeben. Der zusammengesetzte Zustand „LetterList“ repräsentiert eine Optionsgruppe und sendet dem Nutzer drei verschiedene Modi-Konfigurationen als Rückmeldung: 3, 5 und 23. Die Übergänge sind in Tabelle B.13

NewStreetMenu		
in-out $int3$: Bool		
in $b : B$; msg, lo, le, fl, st : Bool		
out $r : M$; $setd$, al, dl, rl : Bool		
0	–	–
1	$\neg int3 \wedge$ Up	–
2	$\neg int3 \wedge$ Left	–
3	$\neg int3 \wedge$ Down	–
4	–	–
5,6	$\neg int3 \wedge$ Down	–
7	–	–
8,9	$\neg int3 \wedge$ Up	–
10	$\neg int3 \wedge$ Down	–
11	$\neg int3 \wedge$ Up	–
12–14	–	–
15–18	–	$r' = 0 \wedge rl'$
19,20	–	–

Tabelle B.12: Übergangstabelle für „NewStreetMenu“

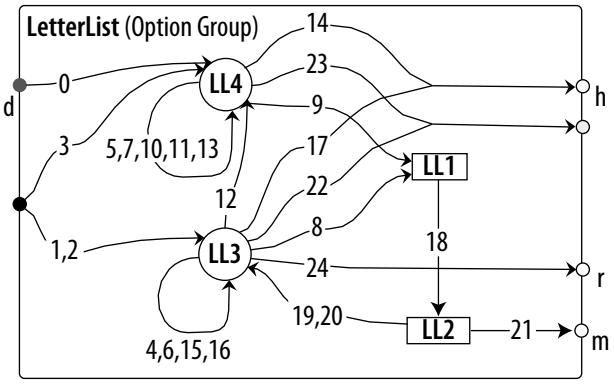


Abbildung B.12: Diagramm für „LetterList“

beschrieben. Der Zustand „LetterList“ kann über unterschiedliche Stellen aktiviert werden.

Wenn eine andere Optionsgruppe aktiv ist und ein Wechsel zum Dialogstrang „Home“ auftritt, dann wird der Zustand „LetterList“ aktiviert und der Fokus an den Dialogstrang „Home“ gegeben (0). Wenn der Zustand „Letter“ die Signale „le“ oder „lo“ sendet, wird die Rückmeldung 3 ausgegeben (1), d. h. die Zeichenkette ist nicht leer und das Hinzufügen eines Zeichens ergibt eine Straße, die verlängerbar ist. Wenn der Zustand „Letter“ das Signal „fl“ sendet, wird die Rückmeldung 5 ausgegeben (2), d. h. die Zeichenkette ist nicht leer und das Hinzufügen eines Zeichens ergibt eine Straße, die nicht mehr verlängerbar ist. Wenn die Zeichenkette keines der Signale „le“, „fl“, „st“ oder „lo“ sendet (3), wird die Rückmeldung 23 ausgegeben, d. h. die Zeichenkette ist leer.

Wenn der Nutzer die Bedienaktion Press durchführt, wird das Signal „al“ ausgege-

LetterList		
in-out $int3$: Bool in b : B; msg, lo, le, fl, st : Bool out r : M; al, rl : Bool		
0	NIL	resume(Home) $\wedge\ r' = 0$
1	$le \vee lo$	RET(3, $int3$)
2	fl	RET(5, $int3$)
3	$\neg le \wedge \neg fl \wedge \neg st \wedge \neg lo$	RET(23, $int3$)
4,5	NIL	$r' = 0$
6,7	$int3 \wedge \neg$NIL	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
8,9	$\neg int3 \wedge$ PRESS	$r' = 0 \wedge al'$
10	$\neg int3 \wedge$ RIGHT	resume(AssistanceWindow) $\wedge\ r' = 0$
11	$\neg int3 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0$
12	$\neg int3 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0 \wedge rl'$
13	resume $\wedge$ NIL	run $\wedge$ RET(23, $int3$)
14	restart $\wedge$ NIL	run
15	resume $\wedge$ NIL $\wedge$ ($le \vee lo$)	run $\wedge$ RET(3, $int3$)
16	resume $\wedge$ NIL $\wedge\ fl$	run $\wedge$ RET(5, $int3$)
17	restart $\wedge$ NIL	run
18	–	$r' = 0$
19	$le \vee lo$	RET(3, $int3$)
20	fl	RET(5, $int3$)
21	st	–
22,23	–	–
24	$\neg int3 \wedge$ RIGHT	–

Tabelle B.13: Übergangstabelle für „LetterList“

ben (8, 9) und ein Takt gewartet (18). Dann wird abhängig vom Zustand der Zeichenkette, die entsprechende Rückmeldung ausgegeben (19, 20). Wenn keine Verlängerung der Straße mehr möglich ist („st“), dann wird die Kontrolle an den übergeordneten Zustands über die Stelle „m“ abgegeben (21).

Wenn der Nutzer den Dialogstrang wechselt, wird die gegenwärtige Zeichenkette durch das Signal „rl“ zurückgesetzt (12). Wenn die Rückmeldung 23 ausgegeben wird und der Nutzer reagiert mit der Bedienaktion RIGHT, wird ins Menü „AssiMenu“ gesprungen (10). Wenn die Rückmeldung 5 oder 3 ausgegeben wird und der Nutzer führt die Bedienaktion RIGHT aus, wird in die Optionsgruppe „DeleteLetter“ gesprungen (24).

Zusammengesetzter Zustand „DeleteLetter“. Das Diagramm des Zustands „DeleteLetter“ ist in Abb. B.13 dargestellt. Die Übergänge sind in Tabelle B.14 beschrieben. Der zusammengesetzte Zustand „DeleteLetter“ stellt eine Optionsgruppe dar und gibt drei verschiedenen Modi-Konfigurationen zurück: Wenn der Zustand „Letter“ das Signal „lo“ sendet, wird die Rückmeldung 18 ausgegeben (0). Wenn der Zustand „Letter“ die Signale „le“ oder „fl“ sednet, wird die Rückmeldung 17 ausgegeben (1). Wenn der Zustand „Letter“ das Signal „st“ sendet, wird die Rückmeldung

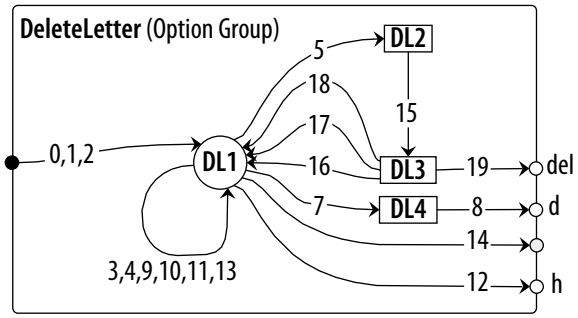


Abbildung B.13: Diagramm für „DeleteLetter“

6 ausgegeben (2).

Reagiert der Nutzer mit der Bedienaktion Press, dann wird das Signal „dl“ an die Applikation geschickt (5) und ein Takt pausiert (15). Darauf folgend wird abhängig von dem Zustand der Zeichenkette die entsprechenden Rückmeldungen ausgegeben (16, 17, 18). Ist die Zeichenkette leer, wird die Kontrolle an den übergeordneten Zustand über die Stelle „del“ gegeben. Das bedeutet, wenn die Zeichenkette leer ist, wird automatisch zur Optionsgruppe „LetterList“ gesprungen.

Wenn der Nutzer die Bedienaktion Menu durchführt, dann wird die Zeichenkette zurückgesetzt (7) und im folgenden Schritt die Kontrolle an den übergeordneten Zustand über die Stelle „d“ gegeben (8). Mit Hilfe des Übergangs 14, kann der übergeordnete Zustand dem Zustand „DeleteLetter“ die Kontrolle entziehen.

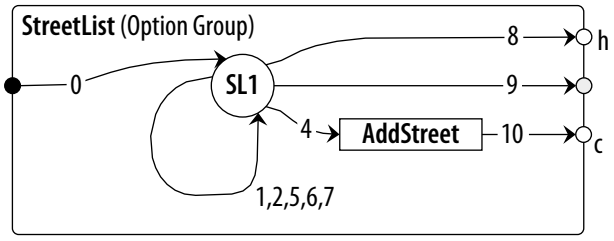


Abbildung B.14: Diagramm für „StreetList“

Zusammengesetzter Zustand „StreetList“. Das Diagramm des Zustands „StreetList“ ist in Abb. B.14 dargestellt. Die Übergänge sind in Tabelle B.15 beschrieben. Wenn der Nutzer den Zustand „StreetList“ aktiviert, wird die Rückmeldung 7 ausgegeben. Mit dem Übergang 9 kann der übergeordnete Zustand die Kontrolle übernehmen. Führt der Nutzer die Bedienaktion Press aus, wird die ausgewählte Straße als Ziel festgelegt („setd“) und die Zeichenkette zurückgesetzt („rl“) (4). Im nächsten Takt verlässt die Kontrolle über die Stelle „c“ den Zustand (10).

Zusammengesetzter Zustand „BackStreet“. Das Diagramm des Zustands „BackStreet“ ist in Abb. B.15 dargestellt. Die Übergänge sind in Tabelle B.16 beschrieben.

DeleteLetter		
in-out $int3$: Bool in b : B; msg, lo, le, fl, st : Bool out r : M; dl, rl : Bool		
0	lo	Ret(18, $int3$)
1	$le \vee fl$	Ret(17, $int3$)
2	st	Ret(6, $int3$)
3	$int3 \wedge \neg$Nil	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
4	$\neg int3 \wedge$ Right	resume(AssistanceWindow) $\wedge\ r' = 0$
5	$\neg int3 \wedge$ Press	$r' = 0 \wedge dl'$
7	$\neg int3 \wedge$ (Menu $\vee$ LongRight)	$r' = 0 \wedge rl'$
8	–	–
9	resume $\wedge$ Nil $\wedge\ lo$	run $\wedge$ Ret(18, $int3$)
10	resume $\wedge$ Nil $\wedge$ $(le \vee fl)$	run $\wedge$ Ret(17, $int3$)
11	resume $\wedge$ Nil $\wedge\ st$	run $\wedge\ r' = 6$
12	restart $\wedge$ Nil	run
13	Nil	$r' = 0$
14	–	–
15	Nil	$r' = 0$
16	$le \vee fl$	Ret(17, $int3$)
17	lo	Ret(18, $int3$)
18	st	Ret(6, $int3$)
19	$\neg le \wedge \neg fl \wedge \neg st \wedge \neg lo$	–

Tabelle B.14: Übergangstabelle für „DeleteLetter“

Der zusammengesetzte Zustand „BackStreet“ stellt eine Optionsgruppe dar. Diese bietet dem Nutzer die Möglichkeit, den Dialog „NewStreetMenu“ abzubrechen. Wenn der Nutzer diesen Zustand aktiviert, wird die Rückmeldung 4 ausgegeben (0). Führt der Nutzer die Bedienaktion Press aus, wird die Zeichenkette anhand des Signals „rl“ zurückgesetzt (3). Im nächsten Takt wird der Zustand über die Stelle „a“ verlassen (9). Der Übergang 10 ermöglicht es dem übergeordneten Zustand, dem Zustand „BackStreet“, die Kontrolle zu entziehen.

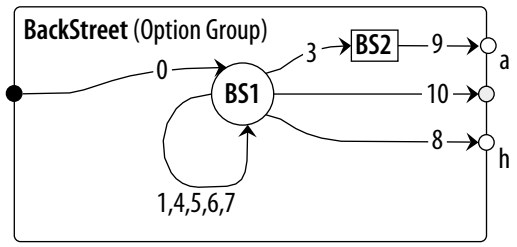


Abbildung B.15: Diagramm für „BackStreet“

StreetList		
in-out $int3$: Bool in b : B; msg : Bool out r : M; $setd$, rl : Bool		
0	–	$\textsc{Ret}(7, int3)$
1	$\textsc{Nil}$	$r' = 0$
2	$int3 \wedge \neg\textsc{Nil}$	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
4	$\neg int3 \wedge \textsc{Press}$	$r' = 0 \wedge setd' \wedge rl'$
5	$\neg int3 \wedge \textsc{Right}$	resume(AssistanceWindow) $\wedge\ r' = 0$
6	$\neg int3 \wedge (\textsc{Menu} \vee \textsc{LongRight})$	resume(Home) $\wedge\ r' = 0 \wedge rl'$
7	resume $\wedge\ \textsc{Nil}$	run $\wedge\ \textsc{Ret}(7, int3)$
8	restart $\wedge\ \textsc{Nil}$	run
9,10	–	–

Tabelle B.15: Übergangstabelle für „StreetList“

BackStreet		
in-out $int3$: Bool in b : B; msg : Bool out r : M; rl : Bool		
0	–	$\textsc{Ret}(4, int3)$
1	$\textsc{Nil}$	$r' = 0$
3	$\neg int3 \wedge \textsc{Press}$	$r' = 0 \wedge rl'$
4	$\neg int3 \wedge \textsc{Right}$	resume(AssistanceWindow) $\wedge\ r' = 0$
5	$\neg int3 \wedge (\textsc{Menu} \vee \textsc{LongRight})$	resume(Home) $\wedge\ r' = 0 \wedge rl'$
6	resume $\wedge\ \textsc{Nil}$	run $\wedge\ \textsc{Ret}(4, int3)$
7	$int3 \wedge \neg\textsc{Nil}$	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
8	restart $\wedge\ \textsc{Nil}$	run
9,10	–	–

Tabelle B.16: Übergangstabelle für „BackStreet“

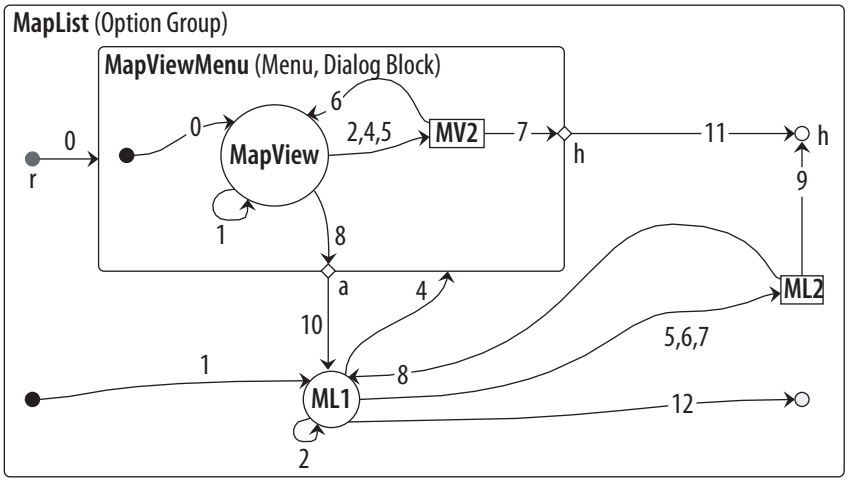


Abbildung B.16: Diagramm für „MapList“

MapList		
in-out $int3$: Bool in b : B; msg : Bool out r : M		
0	–	–
1	–	Ret(12, $int3$)
2	Nil	$r' = 0$
4	$\neg int3 \wedge$ Press	–
5	$int3 \wedge \neg$Nil	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
6	$\neg int3 \wedge$ Right	resume(AssistanceWindow) $\wedge\ r' = 0$
7	$\neg int3 \wedge$ (Menu $\vee$ LongRight)	resume(Home) $\wedge\ r' = 0$
8	resume $\wedge$ Nil	run $\wedge$ Ret(12, $int3$)
9	restart $\wedge$ Nil	run
10	–	Ret(12, $int3$)
11,12	–	–

Tabelle B.17: Übergangstabelle für „MapList“

Zusammengesetzter Zustand „MapList“. Das Diagramm des Zustands „MapList“ ist in Abb. B.16 dargestellt. Die Übergänge sind in Tabelle B.17 beschrieben. Der zusammengesetzte Zustand „MapList“ stellt eine Optionsgruppe dar. Wenn der Nutzer diesen Zustand aktiviert, wird die Rückmeldung 12 ausgegeben (1). Wenn die Umgebung den Zustand über die Stelle „r“ betritt, wird der Unterzustand „MapViewMenu“ aktiviert. Wenn der Nutzer die Bedienaktion Press ausführt, wird der Unterzustand „MapViewMenu“ aktiviert (4). Die restlichen Übergänge stellen das übliche Verhalten dar.

Zusammengesetzter Zustand „MapViewMenu“. Das Diagramm des Zustands „MapViewMenu“ ist in Abb. B.16 dargestellt. Die Übergänge sind in Tabelle B.17 beschrieben. Der zusammengesetzte Zustand „MapViewMenu“ stellt einen Dialogblock

MapViewMenu		
in-out $int3$: Bool in b : B; msg : Bool out r : M		
0	–	RET(16, $int3$)
1	NIL	$r' = 0$
2	$int3 \wedge \neg$NIL	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int3'$
4	$\neg int3 \wedge$ RIGHT	resume(AssistanceWindow) $\wedge\ r' = 0$
5	$\neg int3 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0$
6	resume $\wedge$ NIL	run $\wedge$ RET(16, $int3$)
7	restart $\wedge$ NIL	run
8	$\neg int3 \wedge$ PRESS	–

Tabelle B.18: Übergangstabelle für „MapViewMenu“

sowie ein Menü dar. Wenn dieser Zustand über den Übergang 0 aktiviert wird, gibt er die Rückmeldung 16 aus. Der Dialogblock wird abgebrochen, wenn der Nutzer die Bedienaktion PRESS durchführt (8).

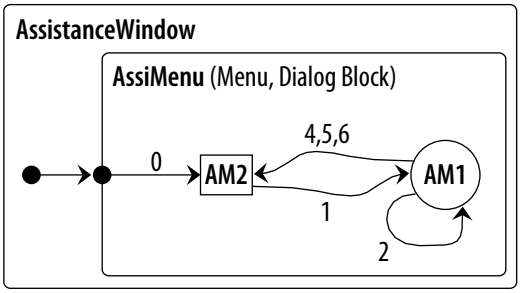


Abbildung B.17: Diagramm für „AssistanceWindow“

B.1.4 Orthogonaler Zustand „AssistanceWindow“

Das Diagramm des Dialogstrangs „AssistanceWindow“ ist in Abb. B.17 dargestellt. Der Nutzer kann in der Regel durch die Bedienaktion RIGHT zu dem Dialogstrang „AssistanceWindow“ wechseln. Aktiviert die Umgebung den Zustand „AssistanceWindow“, wird unmittelbar der Unterzustand „AssiMenu“ aktiviert.

AssiMenu		
in-out $int2$: Bool in b : B; msg : Bool out r : M		
0	–	–
1	resume $\wedge$ NIL	run $\wedge$ RET(19, $int2$)
2	NIL	$r' = 0$
4	$\neg int2 \wedge$ LEFT	resume(Navigation) $\wedge\ r' = 0$
5	$\neg int2 \wedge$ (MENU $\vee$ LONGRIGHT)	resume(Home) $\wedge\ r' = 0$
6	$int2 \wedge \neg$NIL	resume(CheckControlMessage) $\wedge\ r' = 0 \wedge \neg int2'$

Tabelle B.19: Übergangstabelle für „AssiMenu“

Zusammengesetzter Zustand „AssiMenu“. Das Diagramm des Zustands „AssiMenu“ ist in Abb. B.17 dargestellt. Die Übergänge sind in Tabelle B.19 beschrieben. Der zusammengesetzte Zustand „AssiMenu“ stellt einen Dialogblock sowie ein Menü dar. Zu Beginn besitzt das Menü nicht den Fokus. Deshalb wird auch keine Rückmeldung beim initialen Übergang gesendet (0). Bekommt dieses Menü von der Umgebung den Fokus, gibt es die Rückmeldung 19 aus (1). Durch die Bedienakion „Left“ kann der Nutzer den Fokus dem Dialogstrang „Navigation“ zurückgeben (4).

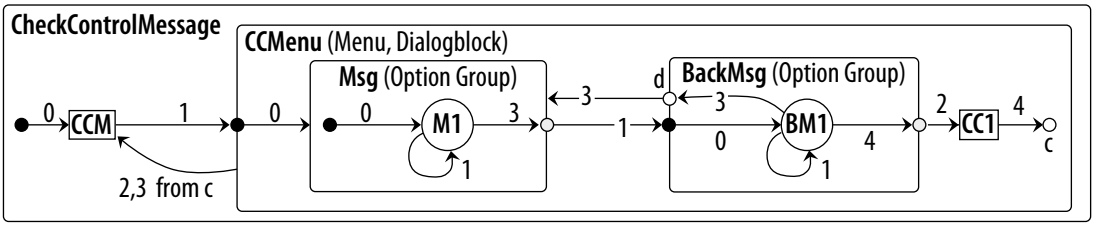


Abbildung B.18: Diagramm für „CheckControlMessage“

CheckControlMessage		
in $b : B$ out $r : M$; ae : Bool		
0	–	–
1	resume $\wedge$ Nil	run
2	from(Navigation) $\vee$ from(AssistanceWindow)	resume(Navigation) $\wedge\, r' = 0$
3	from(Home)	resume(Home) $\wedge\, r' = 0$

Tabelle B.20: Übergangstabelle für „CheckControlMessage“

B.1.5 Orthogonaler Zustand „CheckControlMessage“

Das Diagramm des Zustands „CheckControlMessage“ ist in Abb. B.18 dargestellt. Die Übergänge sind in Tabelle B.20 beschrieben. Der orthogonale Zustand „CheckControlMessage“ stellt einen orthogonalen Zustand dar. Bekommt der Zustand „CheckControlMessage“ die Kontrolle von der Umgebung, aktiviert er den Unterzustand „CCM“. Wenn die Umgebung dem Zustand „CheckControlMessage“ den Fokus übergibt, wird der Unterzustand „CCMenu“ aktiviert (1). Gibt der Unterzustand die Kontrolle ab, wird abhängig davon welcher orthogonale Zustand „CheckControlMessage“ fortgesetzt hat, die Kontrolle abgegeben (2, 3).

CCMenu		
in $b : B$ out $r : M$; ae : Bool		
0,1	–	–
2	–	$r' = 0 \wedge ae'$
3	–	–
4	Nil	–

Tabelle B.21: Übergangstabelle für „CCMenu“

Zusammengesetzter Zustand „CCMenu“. Die Übergänge des Zustands „CCMenu“ sind in Tabelle B.21 beschrieben. Der zusammengesetzte Zustand „CCMenu“ stellt einen Dialogblock sowie ein Menü dar. Der Eintrittsübergang öffnet die beiden Optionsgruppen „Msg“ und „Back-Msg“ und aktiviert den Unterzustand „Msg“ (0). Wird die Kontrolle vom Unterzustand abgegeben, dann bedeutet das, dass der Nutzer

die Meldung bestätigt hat. Darauf folgend wird das Signal „ae“ ausgegeben (2) und im Folgetakt der Dialogblock geschlossen (4).

Msg		
in $b : B$		
out $r : M$		
0	–	$r' = 22$
1	Nil	$r' = 0$
3	Up	–

Tabelle B.22: Übergangstabelle für „Msg“

Zusammengesetzter Zustand „Msg“. Die Übergänge des Zustands „Msg“ sind in Tabelle B.22 beschrieben. Der zusammengesetzte Zustand „Msg“ repräsentiert eine Optionsgruppe. Wird der Zustand aktiviert, dann gibt er die Rückmeldung 22 aus (0). Über den Übergang 3 wird die Kontrolle bei der Bedienaktion Up abgegeben.

BackMsg		
in $b : B$		
out $r : M$		
0	–	$r' = 21$
1	Nil	$r' = 0$
3	Down	–
4	Press	–

Tabelle B.23: Übergangstabelle für „BackMsg“

Zusammengesetzter Zustand „BackMsg“. Die Übergänge des Zustands „BackMsg“ sind in Tabelle B.23 beschrieben. Der zusammengesetzte Zustand „BackMsg“ stellt eine Optionsgruppe dar. Bei seiner Aktivierung wird die Rückmeldung 21 ausgegeben. Führt der Nutzer die Bedienaktion Down durch, wird die Kontrolle über die Stelle „d“ abgegeben. Führt der Nutzer die Bedienaktion Press durch, wird die Kontrolle über die Standard-Stelle abgegeben.

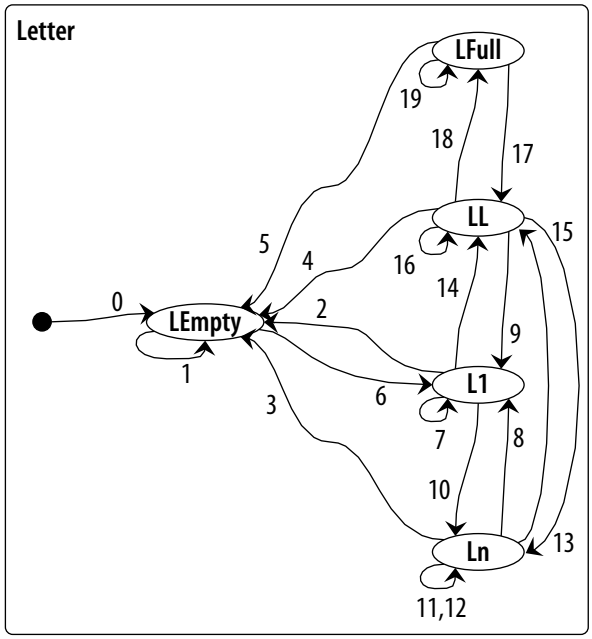


Abbildung B.19: Diagramm für „Letter“

B.1.6 Komponenten der Applikation

Orthogonaler Zustand „Letter“. Das Diagramm des Zustands „Letter“ ist in Abb. B.19 dargestellt. Die Übergänge sind in Tabelle B.24 beschrieben. Dieser zusammengesetzte Zustand ist ein orthogonaler Zustand und ist der Applikation zugeordnet. Er verwaltet die Zeichenkette zur Auswahl der Straße in dem Dialog „NewStreetMenu“. Es werden abstrakte Kontrollzustände verwendet, die das tatsächliche Verhalten abstrahieren. Die Art und Weise der gewählten Modellierung hat zur Folge, dass die Länge der Straßennamen mindestens drei beträgt. Die Komponente kann die folgenden einfachen Zustände einnehmen:

- „LEmpty“: Die Zeichenkette ist leer. Dieser Zustand wird initial aktiviert (0). Wird ein Zeichen hinzugefügt („al“), nimmt „Letter“ den Unterzustand „L1“ ein (6).

- „L1“ (signalisiert durch „lo“): Die Zeichenkette hat genau ein Zeichen. Wenn ein Zeichen gelöscht wird, ist im nächsten Zustand die Zeichenkette leer (2). Wird ein weiteres Zeichen hinzugefügt, dann ist der Zustand „LL“ (14) oder „Ln“ (10). Welches der beiden Zustände gewählt wird, wird nichtdeterministisch entschieden.

- „Ln“ (signalisiert durch „le“): Die Zeichenkette ist nicht leer und das Hinzufügen eines Zeichens ergibt eine Straße, die man verlängern kann. Somit kann der Zustand nach einem Hinzufügen entweder „Ln“ (12) oder „LL“ (15) sein. Wird ein Zeichen gelöscht, dann ist der Folgezustand entweder „Ln“ (12) oder „L1“ (8).

Letter		
in al, dl, rl : Bool out lo, le, fl, st : Bool		
0	–	–
1	$\neg al$	–
2	$dl \vee rl$	–
3–5	rl	–
6	al	lo'
7	$\neg dl \wedge \neg al \wedge \neg rl$	lo'
8,9	dl	lo'
10	al	le'
11	$\neg dl \wedge \neg al \wedge \neg rl$	le'
12	$dl \vee al$	le'
13	dl	le'
14,15	al	fl'
16	$\neg dl \wedge \neg al \wedge \neg rl$	fl'
17	dl	fl'
18	al	st'
19	$\neg dl \wedge \neg rl$	st'

Tabelle B.24: Übergangstabelle für „Letter“

- „LL“ (signalisiert durch „fl“): Die Länge der Zeichenkette ist größer als eins und wenn ein weiteres Zeichen hinzugefügt wird, dann kann die Straße nicht mehr verlängert werden (18). Wenn ein Zeichen gelöscht wird, dann ist danach der Zustand entweder „Ln“ (13) oder „L1“ (9). Die Auswahl ist nichtdeterministisch.

- „LFull“ (signalisiert durch „st“): Die Zeichenkette ist nicht mehr verlängerbar, das heißt es gibt keine Straße mehr, die durch ein weiteres Zeichen beschrieben werden kann. Wenn ein Zeichen gelöscht wird, dann wird der Zustand „LL“ eingenommen (17).

In jedem Zustand kann das Signal „rl“ ein Zurücksetzen auslösen. In der Folge ist der Zustand dann „LEmpty“ (2–5).

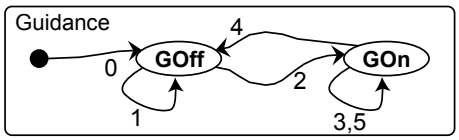


Abbildung B.20: Diagramm für „Guidance“

Orthogonaler Zustand „Guidance“. Das Diagramm des Zustands „Guidance“ ist in Abb. B.20 dargestellt. Die Übergänge sind in Tabelle B.25 beschrieben. Der orthogonale, zusammengesetzte Zustand „Guidance“ stellt die Kontrollzustände der Zielführung dar. Die Zielführung ist entweder deaktiviert („GOff“) oder aktiviert („GOn“).

Guidance		
in sg, dg : Bool		
0	–	–
1	$\neg sg \wedge \neg dg$	–
2	$sg \wedge \neg dg \wedge \neg$in(NoDest)	–
3	$\neg dg \wedge \neg sg$	–
4	$dg \wedge \neg sg$	–
5	$sg \wedge \neg dg \wedge \neg$in(NoDest)	–

Tabelle B.25: Übergangstabelle für „Guidance“

Der initiale Zustand ist „GOff“ (0). Durch das Signal „sg“ kann die Zielführung aktiviert werden. Voraussetzung ist dabei, dass ein Ziel vorliegt (2). Wenn die Zielführung aktiv ist, kann das Ziel gewechselt werden (5) oder die Zielführung deaktiviert werden (4).

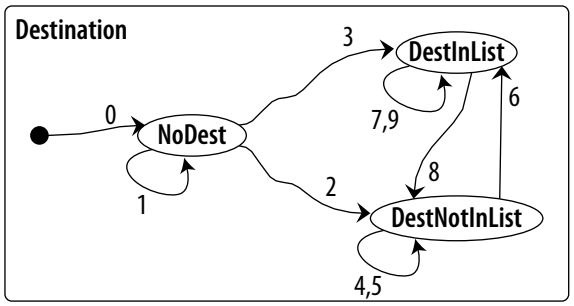


Abbildung B.21: Diagramm für „Destination“

Destination		
in ad, $setd$: Bool		
0	–	–
1	$\neg setd \wedge \neg ad$	–
2	$setd$	–
3	$setd \wedge \neg$in(EmptyDL)	–
4	$\neg setd \wedge \neg ad$	–
5	$setd$	–
6	ad	–
7	$\neg setd \wedge \neg ad$	–
8	$setd$	–
9	$setd \wedge \neg$in(EmptyDL)	–

Tabelle B.26: Übergangstabelle für „Destination“

Orthogonaler Zustand „Destination“. Das Diagramm des Zustands „Destination“ ist in Abb. B.21 dargestellt. Die Übergänge sind in Tabelle B.26 beschrieben. Der orthogonale Zustand „Destination“ stellt das gegenwärtig ausgewählte Ziel dar. Dieses

Ziel ist ein Parameter für eine Reihe an Aktionen. Es gibt die folgenden Kontrollzustände: „NoDest“, d. h. es ist kein Ziel ausgewählt; „DestInList“, d. h. es ist ein Ziel ausgewählt und dieses Ziel ist in der Zielliste enthalten; „DestNotInList“, d. h. es ist ein Ziel ausgewählt und dieses Ziel ist nicht in der Zielliste enthalten. Der initiale Zustand ist „NoDest“. Das Signal „setd“ zeigt, dass ein Ziel ausgewählt worden ist. Wenn die Zielliste nicht leer ist und das Signal „setd“ auftritt, dann kann danach das Ziel in beiden Zuständen sein. Das Verhalten ist an dieser Stelle nichtdeterministisch (2 oder 3). Durch das Signal „ad“ kann das gegenwärtige Ziel explizit zur Liste hinzugefügt werden (6). Wenn die Komponente in dem Kontrollzustand „DestInList“ ist und das Ziel wird gewechselt, dann wird erneut entschieden welcher Kontrollzustand eingenommen wird.

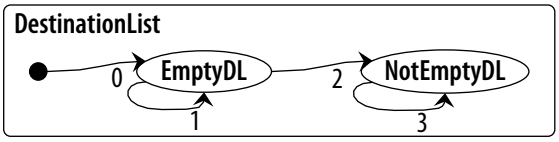


Abbildung B.22: Diagramm für „DestinationList“

DestinationList		
in ad : Bool		
0	–	–
1	$\neg ad$	–
2	ad	–
3	–	–

Tabelle B.27: Übergangstabelle für „DestinationList“

Orthogonaler Zustand „DestinationList“. Das Diagramm des Zustands „DestinationList“ ist in Abb. B.22 dargestellt. Die Übergänge sind in Tabelle B.27 beschrieben. Der orthogonale Zustand „DestList“ modelliert die Zielliste. Die Kontrollzustände sind „EmptyDL“ oder „NotEmptyDL“. „EmptyDL“ bedeutet, dass die Zielliste leer ist; „NotEmptyDL“ bedeutet, dass die Zielliste mindestens ein Ziel enthält. Der initiale Zustand ist „EmptyDL“. Durch das Signal „ad“ wird ein Ziel der Liste hinzugefügt (2).

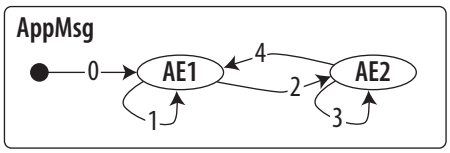


Abbildung B.23: Diagramm für „AppMsg“

Orthogonaler Zustand „AppMsg“. Das Diagramm des Zustands „AppMsg“ ist in Abb. B.23 dargestellt. Die Übergänge sind in Tabelle B.28 beschrieben. Der orthogonale Zustand „AppMsg“ verwaltet die Meldungen der Applikation. Es gibt die

AppMsg		
in ae : Bool out msg : Bool		
0,1	–	–
2	–	msg'
3	$\neg ae$	msg'
4	ae	–

Tabelle B.28: Übergangstabelle für „AppMsg“

Kontrollzustände „AE1“ und „AE2“. Der initiale Zustand ist „AE1“ und bedeutet, dass keine Meldung vorliegt (0). Wenn eine Meldung vorliegt, wird in den Zustand „AE2“ gewechselt und das Signal „msg“ verschickt (2). Die Unterscheidung, ob eine Meldung vorliegt oder nicht, wird nichtdeterministisch modelliert. Solange der Zustand „AE2“ aktiviert ist, wird das Signal „msg“ verschickt. Der Kontrollzustand „AE1“ wird aktiviert, wenn das Signal „ae“ gesetzt wird.

B.2 Tabellen der Modi-Konfigurationen

Im Folgenden sind die Tabellen der Modi-Konfigurationen angegeben.

Für die Modi-Konfigurationen 1–6 s. Tabelle B.29; für 7–14 s. Tabelle B.30 und für 15–23 s. Tabelle B.31.

Modi-Konf.	Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
1	Timeout		○				○	●	○	○	●
2	Press	●	○	○	●	○	○	●	○	●	
2	Up	○	○	○	○	●	○	●	○	●	
2	Right	○	○	○	○	○	○	●	○	○	●
2	LongRight	○	○	○	○	○	○	●	○	●	●
2	Menu	○	○	○	○	○	○	●	○	○	●
3	Up	○	○	○	○	○	●	●	○	○	●
3	Down	○	○	○	○	○	●	●	○	○	●
3	Right	○	○	○	○	○	●	●	○	●	
3	Press	○	○	○	○	○	○	○	●	●	
3	LongRight	○	○	○	○	○	○	●	●	●	
3	Menu	○	○	○	○	○	○	●	●	○	●
4	Down	○	○	○	○	○	●	●	○	●	●
4	Right	○	○	○	○	○	○	●	○	○	●
4	Press	○	○	●	○	●	○	●	●	●	
4	LongRight	○	○	○	○	○	○	●	●	●	
4	Menu	○	○	○	○	○	○	●	●	○	●
5	Up	○	○	○	○	○	●	●	○	○	●
5	Down	○	○	○	○	○	●	●	○	○	●
5	Right	○	○	○	○	○	●	●	○	●	
5	Press	○	○	○	○	○	●	●	●	●	
5	LongRight	○	○	○	○	○	○	●	●	●	
5	Menu	○	○	○	○	○	○	●	●	○	●
6	Up	○	○	○	○	○	●	●	○	○	●
6	Down	○	○	○	○	○	●	●	○	○	●
6	Right	○	○	○	○	○	○	●	○	○	●
6	Press	○	○	○	○	○	○	○	●	●	●
6	LongRight	○	○	○	○	○	●	●	●	●	
6	Menu	○	○	○	○	○	●	●	●	○	●

Tabelle B.29: Modi-Konfigurationen 1–6

Modi-Konf.	Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
7	Up	○	○	○	○	○	●	●	○	●	●
7	Press	○	●	○	○	●	○	●	●	●	
7	Right	○	○	○	○	○	○	●	○	○	●
7	LongRight	○	○	○	○	○	○	●	●	●	
7	Menu	○	○	○	○	○	○	●	●	○	●
8	Up	○	○	○	○	○	●	●	○	○	●
8	Down	○	○	○	●	○	○	●	○	○	●
8	Press	○	○	○	●	○	○	●	○	●	
8	Right	○	○	○	○	○	○	●	○	○	●
8	LongRight	○	○	○	○	○	○	●	○	●	●
8	Menu	○	○	○	○	○	○	●	○	○	●
9	Down	○	○	○	○	○	●	●	○	●	
9	Press	○	○	●	○	●	○	●	○	●	
9	Right	○	○	○	○	○	○	●	○	○	●
9	LongRight	○	○	○	○	○	○	●	○	●	●
9	Menu	○	○	○	○	○	○	●	○	○	●
10	Down	○	○	○	●	○	○	●	○	●	
10	Press	○	○	○	●	○	○	●	○	●	
10	Right	○	○	○	○	○	○	●	○	○	●
10	LongRight	○	○	○	○	○	○	●	○	●	●
10	Menu	○	○	○	○	○	○	●	○	○	●
11	Up	○	○	○	○	●	○	●	○	○	●
11	Down	○	○	○	○	○	●	●	○	○	●
11	Press	●	○	○	●	○	○	●	○	●	
11	Right	○	○	○	○	○	○	●	○	○	●
11	LongRight	○	○	○	○	○	○	●	○	●	●
11	Menu	○	○	○	○	○	○	●	○	○	●
12	Up	○	○	○	○	○	●	●	○	●	
12	Press	●	○	○	●	○	○	●	○	●	
12	Right	○	○	○	○	○	○	●	○	○	●
12	LongRight	○	○	○	○	○	○	●	○	●	●
12	Menu	○	○	○	○	○	○	●	○	○	●
13	Up	○	○	○	○	●	○	●	○	●	
13	Press	●	●	○	●	●	○	○	●	●	
13	Right	○	○	○	○	○	○	●	○	○	●
13	LongRight	○	○	○	○	○	○	●	○	●	●
13	Menu	○	○	○	○	○	○	●	○	○	●
14	Up	○	○	○	○	●	○	●	○	●	
14	Press	●	●	○	●	●	○	●	●	●	
14	Right	○	○	○	○	○	○	●	○	○	●
14	LongRight	○	○	○	○	○	○	●	○	●	●
14	Menu	○	○	○	○	○	○	●	○	○	●

Tabelle B.30: Modi-Konfigurationen 7–14

Modi-Konf.	Bedienaktion	Öffnen	Schließen	Abbruch	Vorwärts	Rückwärts	Tabulator	Sprung	Ansteuerung	Wiederholbar	Stabilität
15	Up	○	○	●	○	●	○	●	○	●	
15	Down	○	○	●	○	●	○	●	○	●	
15	Left	○	○	●	○	●	○	●	○	○	●
15	Right	○	○	●	○	●	○	●	○	○	●
15	Timeout	○	○	●	○	●	○	●	○		
15	Press	●	●	○	●	●	○	●	●	●	
15	LongRight	○	○	●	○	●	○	●	○	●	
15	Menu	○	○	●	○	●	○	●	○	○	●
16	Press	○	○	●	○	●	○	●	○	●	
16	Right	○	○	○	○	○	○	●	○	○	●
16	LongRight	○	○	○	○	○	○	●	○	●	●
16	Menu	○	○	○	○	○	○	●	○	○	●
17	Left	○	○	○	○	○	●	●	○	○	●
17	Up	○	○	○	○	○	●	●	○	○	●
17	Down	○	○	○	○	○	●	●	○	○	●
17	Right	○	○	○	○	○	○	●	○	○	●
17	Press	○	○	○	○	○	○	○	●	●	
17	LongRight	○	○	○	○	○	●	●	●	●	
17	Menu	○	○	○	○	○	●	●	●	○	●
18	Left	○	○	○	○	○	●	●	○	○	●
18	Up	○	○	○	○	○	●	●	○	○	●
18	Down	○	○	○	○	○	●	●	○	○	●
18	Right	○	○	○	○	○	○	●	○	○	●
18	Press	○	○	○	○	○	●	●	●	●	
18	LongRight	○	○	○	○	○	●	●	●	●	
18	Menu	○	○	○	○	○	●	●	●	○	●
19	Left	○	○	○	○	○	○	●	○		
19	LongRight	○	○	○	○	○	○	●	○	●	●
19	Menu	○	○	○	○	○	○	●	○	○	●
20	Right	●	○	●	●	●	○	●		●	
20	LongRight	○	○	○	○	○	○	●	○	●	
21	Press	○	●	○	○	●	○	●	○		
21	Down	○	○	○	○	○	●	●	○	○	●
22	Up	○	○	○	○	○	●	●	○	○	●
23	Up	○	○	○	○	○	●	●	○	○	●
23	Down	○	○	○	○	○	●	●	○	○	●
23	Press	○	○	○	○	○	○	○	●	●	●
23	Right	○	○	○	○	○	○	●	○	○	●
23	LongRight	○	○	○	○	○	○	●	○	●	●
23	Menu	○	○	○	○	○	○	●	○	○	●

Tabelle B.31: Modi-Konfigurationen 15–23

B.3 Übergänge zwischen den Modi-Konfigurationen

Abb. B.24 zeigt die Übergänge zwischen den Modi-Konfigurationen. Tab. B.32 gibt zusätzlich die Bedienaktionen an, die die Übergänge bewirken.

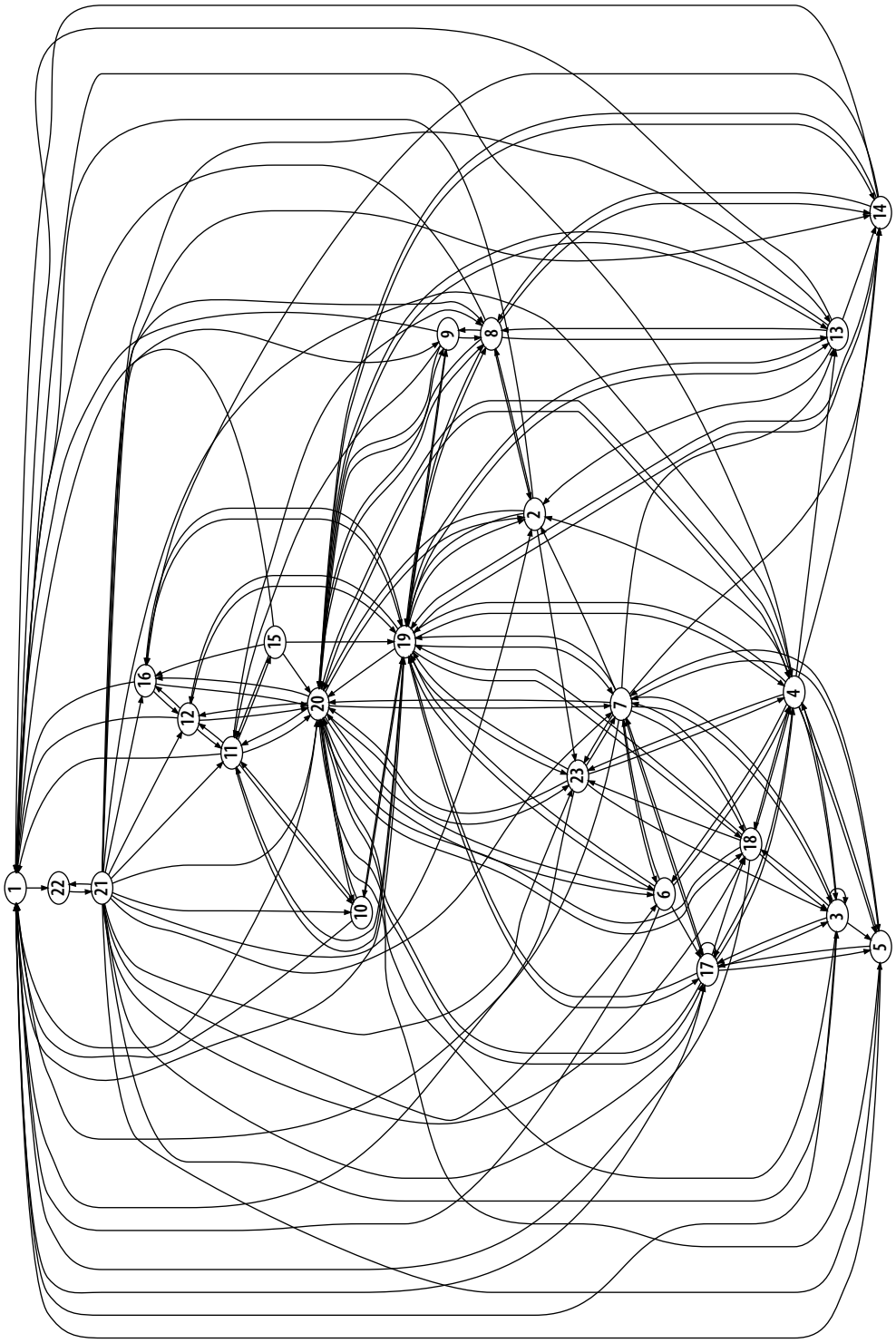


Abbildung B.24: Übergangsgraph der Modi-Konfigurationen

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
1																						√	
2	⋆			←				←											↑	⇑			•
3	⋆				•		→																
4	⋆	•	→		→	→								•					↑	⇑			→
5	⋆			←			•↓																
6	⋆			←			→												↑	⇑			
7	⋆	•	←			←								•	●		←		↑	⇑			←
8	⋆	•↓							←					•↓					↑	⇑			
9	⋆							→			•								↑	⇑			
10	⋆										•↓√								↑	⇑			
11	⋆							•		←		→							↑	⇑			
12	⋆										←					•			↑	⇑			
13	⋆	•						←						•					↑	⇑			
14	⋆							←								•			↑	⇑			
15	⋆										•↑↓←√					•			↑	⇑			
16	⋆											•							↑	⇑			
17	⋆		↓	←	↓		→											•	↑	⇑			
18	⋆		↓	←			→												↑	⇑			•
19	⋆	↓		↓		↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓					↓
20	⋆	⇑		⇑		⇑	⇑	⇑	⇑	↑⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑	⇑					⇑
21		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•		•			•
22																					↑		
23	⋆		•	↑			↓												↑	⇑			

⋆: alle def. Bedienaktionen, ↑: Up, ↓: Down, ←: Left, →: Right, ⇒: LongRight, •: Press, ○: Menu, √: Timeout.

Tabelle B.32: Mit Bedienaktionen markierte Übergänge