Kontextsensitive Personalisierung automotiver Benutzerschnittstellen

Entwicklung und Anwendung eines regelbasierten Verfahrens zur Erkennung situationsabhängiger Mensch-Maschine-Interaktionen

vorgelegt von
Dipl.-Ing. Sandro Rodriguez Garzon
aus Berlin

an der Fakultät IV - Elektrotechnik und Informatik
der Technischen Universität Berlin
zur Erlangung des akademischen Grades

Doktor der Ingenieurwissenschaften
- Dr.-Ing. -

genehmigte Dissertation

Promotionsausschuss

Vorsitzender:	Prof. Dr.-Ing. Clemens Gühmann
Gutachter:	Prof. Dr. Peter Pepper
Gutachter:	Prof. Dr. Dr.-Ing. Wolfgang Minker

Tag der wissenschaftlichen Aussprache: 29. November 2013

Berlin 2013
D83

Ich halte dafür, daß das einzige Ziel der Wissenschaft darin besteht, die Mühseligkeit der menschlichen Existenz zu erleichtern.

Bertolt Brecht, Leben des Galilei, 1939

Zusammenfassung

Moderne In-Car-Infotainment-Systeme verfügen aufgrund der in den letzten Jahren erzielten technologischen Fortschritte bei der Vernetzung von Fahrzeugen untereinander und mit dem Internet über eine Fülle neuer Funktionalitäten, die bisher ausschließlich Home-Entertainment-Systemen oder mobilen Computern wie beispielsweise Smartphones oder Tablets vorbehalten blieben. Die damit einhergehende Zunahme der Komplexität automotiver Benutzerschnittstellen stellt die Automobilhersteller jedoch vor die Herausforderung, neue und effiziente Bedienkonzepte zu entwickeln, die neben dem markenspezifischen Look-And-Feel auch die Integration der Benutzerschnittstelle bestehender mobiler Endgeräte oder auch die durch nachträgliche Installation von Applikationen ermöglichte individuelle Gestaltung der Benutzerschnittstelle berücksichtigen.

Im Vordergrund der Untersuchungen dieser Forschungsarbeit steht daher insbesondere das Bedienkonzept der Personalisierung, welches sich mit der Anpassungsfähigkeit der Benutzerschnittstellen an die individuellen Bedürfnisse der Benutzer auseinandersetzt. Dazu wird ein neuartiges Verfahren zur Verarbeitung unterschiedlicher Sensordaten vorgestellt, welches es zukünftigen mobilen Systemen wie z.B. In-Car-Infotainment-Systemen ermöglicht, das situationsabhängige Benutzerverhalten implizit zu erfassen, automatisch auf Regelmäßigkeiten hin untersuchen zu lassen und die sich daraus ergebenden kontextsensitiven Bedienmuster als Ausgangsbasis für proaktive Adaptionen der Benutzerschnittstelle zu nutzen. Bei dem Verfahren werden erstmals bestimmte Methoden zur Erkennung sequentieller Muster und bekannte Techniken des nicht-überwachten Gruppierens innerhalb eines Prozesses gemeinsam angewendet, um in der Fülle von benutzer- und umgebungsspezifischen Sensordaten, ohne ein explizites Eingreifen durch den Benutzer, automatisiert ähnliche Sequenzen von Mensch-Maschine-Interaktionen zu erkennen. Unterstützt wird dieses als Dienst konzipierte Verfahren zur kontextsensitiven Personalisierung durch in Vorlagen zu definierende Regeln, die je nach Anwendungsfall bestimmen, welche spezifischen Mensch-Maschine-Interaktionen zu einer kontextsensitiven Personalisierung beitragen können, inwiefern aufgezeichnete Interaktionen zum Zweck der Prüfung auf Relevanz zusammengefasst werden und auf welche Art und Weise Situationen erkannt werden, in denen die ermittelten kontextsensitiven Präferenzen gegebenenfalls eine Modifikation der Benutzerschnittstelle auslösen. Vorgestellt werden - im Sinne eines Proof-of-Concepts - eine dazu passende prototypische automotive Benutzerschnittstelle, die mit Regeln versehenen Vorlagen und zwei unterschiedliche Formen der kontextsensitiven Personalisierung, welche beispielhaft in vier automotiven Bedienszenarien zum Einsatz kommen.

Außerdem wird eine Methode der modellbasierten Simulation der interaktionsbezogenen Sensordaten präsentiert, mit der die über einen längeren Zeitraum stattfindende kontextsensitive Personalisierung zu Testzwecken im Labor simuliert werden kann. Ergänzt wird die Forschungsarbeit um eine detaillierte Beschreibung des Ablaufs und der Ergebnisse einer im Rahmen dieser Forschung erfolgten Benutzerevaluation, die auf Grundlage eines ebenfalls hier entworfenen Echtzeitsimulators für kontextuelle Informationen durchgeführt wurde. Die Benutzerevaluation diente unter anderem der Untersuchung der durch die kontextsensitive Personalisierung hervorgerufenen Ablenkung des Fahrers und der Bewertung der Attraktivität adaptiver Bestandteile einer automotiven Benutzerschnittstelle.

Abstract

The technological progress of the last years regarding the connectivity between cars and between cars and the internet made it possible for modern in-car-infotainment systems to offer a wide range of new features that are formerly known from home entertainment systems or mobile computers like smartphones or tablets. Due to an accompanying increase of the complexity of automotive user interfaces, car manufactures are faced with the challenge to develop new and efficient interaction concepts. Thereby, the brand-specific look-and-feel, the integration of mobile devices as well as the user-specific design - which is imposed by the subsequent and user-triggered installation of applications - need to be considered.

In particular, this study focuses on the interaction concept of personalization that deals with tailoring the user interfaces towards the individual needs of the users. Therefore, a novel method for the processing of various sensor data is introduced. It provides future mobile systems e.g. in-car-infotainment systems with the ability to capture the situation-dependent user behavior implicitly, to search for regular situation-dependent user behavior and to use the discovered contextual behavior patterns for the purpose of proactive user interface adaptations. For the first time, methods for the detection of sequential patterns and well-known techniques of unsupervised clustering are applied within a common process in order to find similar sequences of human-machine interactions within numerous user-specific and environment-specific sensor data and without the need of an explicit involvement of the user. The service-oriented method is supported by clearly defined rules which determine the way specific human-machine interactions affect the contextual personalization, how captured interactions are grouped to verify their relevance and how situations for the modification of the user interfaces get detected based on the identified contextual preferences. The rule templates and an appropriate prototypical automotive user interface are also introduced as a proof-of-concept along with an extension of the user interface with two different types of contextual personalizations which are adopted in 4 sample automotive use cases.

Additionally, a model-based simulation for interaction-specific sensor data is presented. It allows within a lab environment to simulate long term contextual personalizations for the purpose of tests. Finally, this study is completed by a detailed description of the process and the results of an user evaluation. The user evaluation investigates the driver distraction - caused by the contextual personalization - and the overall attractiveness of the adaptive parts of an automotive user interface. It was conducted based on a real-time simulator for contextual sensor data that gets introduced as well.

Danksagung

Ich möchte mich an dieser Stelle herzlich bei Prof. Dr. Peter Pepper von der Technischen Universität Berlin, Prof. Dr. Ralf-Guido Herrtwich von der Daimler AG und bei Prof. Dr. Dr. Wolfgang Minker von der Universität Ulm bedanken, ohne deren Unterstützung diese Forschungsarbeit nicht möglich gewesen wäre. Während Prof. Dr. Ralf-Guido Herrtwich mir am Daimler Center for Automotive Information Technology Innovations den Anstoß zur Untersuchung personalisierbarer automotiver Benutzerschnittstellen gab, schuf Prof. Dr. Peter Pepper am Fachbereich für Übersetzerbau und Programmiersprachen ein Forschungsumfeld, dem es am fachlichen Rat, an der Unterstützung durch fachbezogene Diskussionen, an regelmäßigen Kolloquien, internationalen Forschungsgästen und diversen Möglichkeiten zur Publikation ausgewählter Beiträge dieser Forschungsarbeit niemals mangelte. Prof. Dr. Dr. Wolfgang Minker danke ich darüber hinaus für seine wertvollen Anmerkungen, konstruktiven Verbesserungsvorschläge und die Übernahme des Zweitgutachtens.

Dr. André Berton von der Daimler AG möchte ich hiermit in besonderen Maße für seine Unterstützung des Promotionsvorhabens seitens der Daimler AG danken. Gemeinsam mit seinen Kollegen der Daimler AG Dr. Stefan Schwehr, Paul Heisterkamp und Mark Poguntke inspirierte er mich zu praxisnahen Anwendungen des vorgestellten Verfahrens und stellte außerdem die nötige Infrastruktur für die Benutzerevaluation und für die realitätsnahen Simulationen des Verfahrens zur Verfügung. Christian Lorenz und Philipp Fischer von der Daimler AG danke ich ebenfalls für ihre Unterstützung, da sie mir unter anderem einen Forschungsaufenthalt am Mercedes-Benz Research & Development Center in Palo Alto ermöglichten und mich dabei auf die zündende Idee zu dem vorgestellten Verfahren brachten.

Meinen Kollegen des Fachbereichs für Übersetzerbau und Programmiersprachen an der Technischen Universität Berlin Judith Rohloff, Florian Lorenzen, Christoph Höger und Martin Zuber danke ich darüber hinaus herzlich für die vielen konstruktiven Diskussionen, die im Rahmen zahlreicher kritischer Auseinandersetzungen mit dem entwickeltem Verfahren und dessen Anwendung im automotiven Bereich stets nützliche Ideen und wertvolle Verbesserungsvorschläge hervorbrachten.

Ich möchte mich weiterhin bei Kristof Schütt, Dmytro Hritsevskyy, Martin Görg, Francois Gerard und Waldemar Lichtner bedanken, deren im Rahmen von Diplom- und Bachelorarbeiten erarbeiteten Prototypen entweder die Funktionsweise bestimmter Teilaspekte des vorgestellten Verfahrens in der Praxis bestätigten oder die eine Simulation des Verfahrens im Labor erst möglich machten. Ein besonderer Dank gilt dabei Kristof Schütt und Dmytro Hritsevskyy, die mich außerdem als studentische Mitarbeiter am Daimler Center for Automotive Information Technology Innovations maßgeblich bei der prototypischen

Umsetzung sowohl des Verfahrens als auch der Implementierung einer beispielhaften automotiven Benutzerschnittstelle unterstützten.

Ein großer Dank geht auch an alle Kollegen des Daimler Centers for Automotive Information Technology Innovations, Kollegen des Fachbereichs für Übersetzerbau und Programmiersprachen und alle weiteren Probanden für ihre tatkräftige Unterstützung bei der Durchführung der Benutzerevaluation.

Schließlich und keineswegs zuletzt möchte ich mich herzlich bei meiner gesamten Familie und meiner Freundin Karen Genz bedanken. Während meine Mutter Bettina Rodriguez Garzon akribisch Kapitel für Kapitel Korrektur gelesen hat, erhielt ich von meinem Vater Orlando Rodriguez Garzon wertvolle Anregungen zum Layout, der Gestaltung und dem Inhalt der Forschungsarbeit. Meine Schwester Fabiola Rodriguez Garzon und Karen Genz standen mir bei meinen Forschungen immer tatkräftig zur Seite und motivierten mich auch in kritischen Zeiten die Ziele nicht aus den Augen zu verlieren.

Publikationen

Folgende Beiträge bei wissenschaftlichen Konferenzen, Symposien und Workshops behandeln Ideen, Methoden, Verfahren und Erkenntnisse, die wesentlich zur Erstellung dieser Forschungsarbeit beitrugen. Bestimmte Inhalte dieser Forschungsarbeit sind daher in der einen oder anderen Form in den veröffentlichten Beiträgen enthalten.

- Fischer, Philipp; Cebulla, Michael; Berton, André; Nürnberger, Andreas; Rodriguez Garzon, Sandro: Adaptive Personalization of Multimodal Vehicular Interfaces using a Hybrid Recommendation Approach with Fuzzy Preferences. In: *Proc. of IEEE Intelligent Vehicles Symposium*, S. 209-213, Eindhoven, Niederlande, Juni 2008

- Rodriguez Garzon, Sandro: Torwards Model-based Personalization. *Doctoral Consortium bei der 1st and 17th Int. Conf. on User Modeling, Adaptation and Personalization*, Trento, Italien, Juni 2009

- Rodriguez Garzon, Sandro; Cebulla, Michael: Model-based Personalization within an Adaptable Human-Machine Interface Environment that is Capable of Learning from User Interactions. In: *Proc. of 3rd Int. Conf. on Advances in Computer-Human Interactions*, S. 191-198, Sint Maarten, Niederlande, Februar 2010

- Görg, Martin; Cebulla, Michael; Rodriguez Garzon, Sandro: A Framework for Abstract Representation and Recognition of Gestures in Multi-Touch Applications. In: *Proc. of 3rd Int. Conf. on Advances in Computer-Human Interactions*, S. 143-147, Sint Maarten, Niederlande, Februar 2010

- Rodriguez Garzon, Sandro; Schütt, Kristof: Road-Based Adaptation of In-Car-Infotainment Systems. In: *Proc. of 4th Int. Conf. on Advances in Computer-Human Interactions*, S. 147-152, Gosier, Guadeloupe, Frankreich, Februar 2011

- Rodriguez Garzon, Sandro; Schütt, Kristof: Discover Significant Situations for User Interface Adaptations. In: *Proc. of the 3rd Workshop on Multimodal Interfaces for Automotive Applications*, S. 29-32, Palo Alto, CA, USA, Februar 2011

- Rodriguez Garzon, Sandro; Poguntke, Mark: The Personal Adaptive In-Car HMI: Integration of External Applications for Personalized Use. In: *Proc. of the Int. Workshop on Augmenting User Models with Real World Experiences to Enhance Personalization and Adaptation*, S. IV, Gerona, Spanien, Juli 2011

- Rodriguez Garzon, Sandro; Poguntke, Mark: The Personal Adaptive In-Car HMI: Integration of External Applications for Personalized Use. In: *Advances in User Modeling. Revised Selected Papers.*, S. 35-46, 2012

- Rodriguez Garzon, Sandro; Hritsevskyy, Dmytro: Model-based Generation of Scenario-specific Event Sequences for the Simulation of Recurrent User Behavior within Context-Aware Applications. In: *Proc. of Symposium on Theory of Modeling and Simulation*, S. 176-181, Orlando, FL, USA, März 2012

- Rodriguez Garzon, Sandro: Intelligent In-Car-Infotainment Systems: A Contextual Personalized Approach. In: *Proc. of 8th Int. Conf. on Intelligent Environments*, S. 315-318, Guanajuato, Mexiko, Juni 2012

- Rodriguez Garzon, Sandro: Intelligent In-Car-Infotainment System: A Prototypical Implementation. In *Proc. of 8th Int. Conf. on Intelligent Environments*, S. 371-374, Guanajuato, Mexiko, Juni 2012

- Rodriguez Garzon, Sandro: Situation-Aware Personalization of Automotive User Interfaces. In *Adjunct Proc. of 4th Int. Conf. on Automotive User Interfaces and Interactive Vehicular Applications*, S. 15-16, Portsmouth, NH, USA, Oktober 2012

Inhaltsverzeichnis

Teil I Grundlagen

1 Einführung 3
1.1 Motivation 5
1.2 Problem- und Aufgabenstellungen 8
1.3 Wissenschaftliche Beiträge 10
1.4 Überblick 12

2 Grundlagen 15
2.1 Adaptive Systeme 15
2.1.1 Benutzeradaptives System 17
2.1.2 Benutzer- und kontextadaptives System 18
2.2 Personalisierung 21
2.2.1 Benutzermodellierung 22
2.2.2 Anpassung der Benutzerschnittstelle 29
2.3 Personalisierung im automotiven Bereich 31
2.3.1 Anforderungen adaptiver und automotiver Benutzerschnittstellen . . . 31
2.3.2 Pattern Matching zur Erkennung relevanten Benutzerverhaltens 34
2.3.3 Clustering relevanten Benutzerverhaltens 36
2.3.4 Vier Anwendungsfälle kontextsensitiver Personalisierungen im Fahrzeug 38

3 Verwandte Arbeiten 43
3.1 Kontextsensitive Personalisierung im Umfeld der Ambient Intelligence 44
3.1.1 Verfahren zur Erfassung der Interaktionen und des Kontextes 45
3.1.2 Verfahren zur Ermittlung kontextsensitiver Präferenzen 46
3.1.3 Anwendungen kontextsensitiver Anpassungen 53
3.1.4 Diskussion 54
3.2 Verfahren zur Simulation kontextsensitiven Benutzerverhaltens 56
3.2.1 Ausführbare Simulationen mit simulierter Zeit 56
3.2.2 Interaktive Simulationen in Echtzeit 58
3.2.3 Diskussion 59

Teil II Verfahren und Implementierung

4 Kontextsensitive Personalisierung 63
4.1 Systemdienst zur kontextsensitiven Personalisierung 63
4.2 Verfahren der vorlagenbasierten kontextsensitiven Personalisierung 67
4.2.1 Ermittlung des Kontextes 69
4.2.2 Ermittlung relevanter Interaktionen 71
4.2.3 Gruppierung relevanter Interaktionen 81
4.2.4 Erkennung der Situationen kontextsensitiver Präferenzen 91
4.2.5 Betrachtung der Zeit- und Speicherkomplexitäten 95

5 Prototypische Realisierung 101
5.1 HMI-Prototyp 102
5.1.1 Bedienung 104
5.1.2 Erweiterung der Benutzerschnittstelle um kontextsensitive Adaptivität 106
5.1.3 Technische Umsetzung 110
5.2 Systemdienst der kontextsensitiven Personalisierung 112
5.2.1 Schnittstelle 112
5.2.2 Konfiguration über Vorlagen 114
5.2.3 Technische Umsetzung und Datenbank 119
5.3 Simulation der kontextrelevanten Sensordaten 120
5.3.1 Simulation auf Grundlage der Sensordaten eines Versuchsfahrzeugs . 121
5.3.2 Simulation mit Hilfe eines Fahrsimulators 122

Teil III Erprobung

6 Simulation 127
6.1 Simulation des Benutzer- und Umgebungsverhaltens 128
6.2 Quantitative Simulation anhand eines Szenarios 130
6.2.1 Modellierung der Interaktionen des Benutzers und der Umgebung . . . 131
6.2.2 Generierung und Simulation der Interaktionen 137
6.2.3 Technische Umsetzung 138

7 Evaluation 141
7.1 Fragestellungen und Hypothesen 142
7.2 Benutzerevaluation 144
7.2.1 Fragebögen 144
7.2.2 Aufbau und Durchführung 147
7.2.3 Ergebnisse und Auswertung 150

Teil IV Résumé

8 Fazit 159
8.1 Problemstellungen und Ergebnisse 159
8.2 Voraussetzungen und Einschränkungen 162
8.3 Offene Fragen 164
8.4 Ausblick 166

Literaturverzeichnis 175

Systemdienst 187
A.1 Anwendungsfallabhängige Instanziierung der Situationsnachrichten 187
A.2 Schematische Spezifikation der Situations- und Dienstnachrichten 188
A.3 UML-Sequenzdiagramm der Kommunikation zwischen dem HMI-Prototyp und dem Systemdienst 193
A.4 Schematische Spezifikation der Vorlagen 194

Simulation 203
B.1 Szenarienspezifisches Modell der Rückfahrt 203

Benutzerevaluation 205
C.1 Fragebogen: Einführung 205
C.2 Fragebogen: Laufend 207
C.3 Fragebogen: Abschluss 208

Eidesstattliche Erklärung 213

Teil I
Grundlagen

The most profound technologies are those that disappear.
They weave themselves into the fabric of everyday life
until they are indistinguishable from it.

Mark Weiser, The Computer for the 21st Century, 1991

Kapitel 1
Einführung

In immer mehr Bereichen des gesellschaftlichen Lebens sind Computer, die in früheren Jahren aufgrund ihrer komplexen Bedienbarkeit und hohen Investitionskosten zunächst nur der wissenschaftlichen Gemeinde vorbehalten blieben, nicht mehr wegzudenken. Computergestützte Systeme erleichtern unter anderem das Steuern von Fahrzeugen, vereinfachen beispielsweise die Überwachung des Flugverkehrs und ermöglichen außerdem die Kommunikation über ein weltumspannendes Netzwerk. Auch die im automotiven Bereich eingesetzten Bordcomputer, die zunächst ausschließlich dazu dienten, einen hohen Komfort während der Fahrt, beispielsweise durch den Einsatz einer Klimaautomatik zu gewährleisten, sind aufgrund der neuen Unterhaltungs- und Informationsmöglichkeiten ebenfalls ein fester Bestandteil jedes modernen Fahrzeugs.

Der Erfolg eines neuen, computergestützten Systems lässt sich oftmals auf diverse Eigenschaften des Systems zurückführen. So wird ein hoher Grad an Akzeptanz eines solchen Systems - innerhalb verschiedenster Anwenderschichten - nicht unbedingt nur durch dessen Erweiterung mit neuer Funktionalität, sondern vor allem durch eine Vereinfachung der Bedienung erzielt. Insbesondere am Fall der mobilen Nutzung des Internets ist dabei beispielsweise zu beobachten, dass es der Industrie erst durch die natürlich anmutende Bedienung von Smartphones auf der Basis der Multitouch-Display-Technologie gelang, das mobile Browsen auf Webseiten auch einer breiten Bevölkerung zugänglich zu machen. Einen ähnlichen Erfolg erzielten die Computerfirmen Anfang der neunziger Jahre nach der Einführung der grafischen Benutzeroberfläche.

Der durch diese Erfolge in den Fokus gerückte Wissenschaftszweig der Mensch-Maschine-Interaktion erfährt dabei gerade im letzten Jahrzehnt eine erhöhte Aufmerksamkeit. Da immer komplexere Aufgaben mit Hilfe der Systeme ausgeführt werden können, ist es insbesondere für die Entwickler einer Benutzerschnittstelle immer schwieriger die Balance zwischen einem breiten Funktionsspektrum und einer leichten, intuitiven Bedienbarkeit zu finden. Daher ist es aufgrund des Erfolgs nicht überraschend, dass die *American Dialect Society* das Wort „App“ zum Wort des Jahres 2010 gekürt hat [Zimmer u. a., 2011]. Die App als kompakte, überschaubare und nachladbare Applikation für das Smartphone oder den TV, die meistens nur einem einzigen klar definierten Zweck dient, bedient dabei oftmals genau zwei Grundbedürfnisse der Benutzer: Intuitive und bedarfsgerechte Anpassungsfähigkeit der Bedienung bzw. Funktionalität. Eine App lässt sich oftmals aufgrund der Reduzierung auf das Wesentliche schnell und einfach erlernen. Außerdem ist der Benutzer in der Lage, nur bestimmte Apps auf sein computergestütztes System zu laden, die er in seinem Kontext auch als nützlich erachtet. Dem Benutzer ist es also im Rahmen der Möglichkeiten der Hard- und Software gestattet, seine persönliche Benutzerschnittstelle mitzugestalten.

Während das Konzept der bedarfsgerechten Anpassung der Benutzerschnittstelle in der Welt der Mobiltelefone und Tabletrechner bereits recht früh Einzug gehalten hat, bleibt die statisch wirkende Benutzerschnittstelle in der Automobilbranche nach wie vor vorherrschend. Bordcomputer können nur in einem beschränkten Maß, das oftmals nicht über die Einstellung der Sprache hinaus geht, an die speziellen Bedürfnisse eines Benutzers angepasst werden. Derzeitige Bordcomputer unterstützen weder die Möglichkeit, das Aussehen ihrer Benutzerschnittstelle vom Benutzer ändern zu lassen, noch eigenhändig bzw. automatisch nicht benötigte Funktionalitäten auszublenden oder präferierte Funktionalitäten hervorzuheben. Dabei dienen die zuletzt genannten Varianten der bedarfsgerechten Anpassung in Betriebssystemen für Heim- und Arbeitsplatzrechner seit Jahren als probate Mittel die Bedienung zu *personalisieren*. Das computergestützte System überwacht dabei jede Interaktion des Benutzers mit dem System, erkennt alle häufig stattfindenden ähnlichen Interaktionen und passt die Benutzerschnittstelle gegebenenfalls an die persönlichen Präferenzen an.

Die Gründe für die sehr zögerliche Integration der zuletzt genannten Varianten der bedarfsgerechten Anpassung der Benutzerschnittstelle liegen in den besonderen Anforderungen der automotiven Benutzerschnittstelle. Im Gegensatz zu Mobiltelefonen und Tabletrechnern, deren Bedienung die volle Aufmerksamkeit des Nutzers voraussetzen, dienen Bordcomputer nur der Unterstützung des Fahrers während der Fahrt. Die automotive Benutzerschnittstelle muss dabei gewährleisten, dass die gewünschten Funktionalitäten des Bordcomputers während der Fahrt ohne hohen kognitiven oder physischen Aufwand ausgewählt und ausgeführt werden können. Eine personalisierte Benutzerschnittstelle kann jedoch mit etwaigen automatischen Anpassungen sehr schnell Gefahr laufen, den Benutzer durch eine neue Bedienung zu überfordern und damit die Aufmerksamkeit unbeabsichtigt auf sich zu ziehen.

Ein weiterer Grund für die sehr zögerliche Integration liegt in dem begrenzten Spektrum an Funktionalitäten, die die Bordcomputer bis dato auf Grund weitgehend fehlender Internetkonnektivität zur Verfügung stellen. Für die derzeit integrierten Funktionalitäten stehen genug physikalisch vorhandene Schaltflächen zur Verfügung um auch speziellen Bedürfnissen effizient gerecht zu werden. Beispielsweise ist die grundlegende Funktionalität des Autoradios in vielen Fahrzeugmodellen über Tasten direkt und schnell erreichbar. Es bedarf also abgesehen vom Programmieren der bevorzugten Radiosender und dem Belegen von Funktionstasten zur direkten Wahl eines Radiosenders keiner weiteren individuellen Anpassung. Da das Internet in Zukunft zweifellos fester Bestandteil eines Bordcomputers sein wird, ist es notwendig, die damit einhergehende Fülle an neuen Funktionalitäten eines Internetradios bedarfsgerecht zu präsentieren und effizient nutzbar zu machen. Neben dem Radiosender müssen dann weitere Möglichkeiten bei der Bedienung des Autoradios wie beispielsweise das Einstellen eines Musikgenres oder die Wahl bestimmter Interpreten beachtet werden. Aufgrund der begrenzten Anzahl physikalisch vorhandener Tasten ist es jedoch notwendig, zusätzliche Funktionalitäten innerhalb einer komplexen Menüstruktur zu verwalten. Neue Funktionalitäten können also nicht mehr direkt über physikalisch vorhandene Tasten ausgeführt werden, sondern müssen mühevoll über die Menüstruktur angewählt werden. Die nachträgliche Integration bestehender Entertainment-Geräte, wie beispielsweise MP3-Player, stellt die Automobilhersteller dabei vor ähnliche Herausforderungen [Lee u. a., 2012]. Auch hier gilt es, komplexe Menüstrukturen, die ursprünglich mit unterschiedlichen Voraussetzungen konzipiert wurden, in das bestehende Bedienkonzept zu integrieren, ohne dabei die kognitive und physische Last der Bedienung signifikant zu steigern. In beiden Fällen ließen sich aus Sicht des Bordcomputers die regelmäßig über die Menüstruktur

mühselig ausgewählten Funktionalitäten des Autoradios oder des MP3-Players implizit erkennen und die zur Verfügung stehenden Direktwahltasten mit diesen Funktionalitäten belegen. Die Anzahl der für die Anwahl und Ausführung einer präferierten Funktion notwendigen Bedienschritte würde sich dadurch deutlich reduzieren.

Das Ziel der dieser Arbeit zugrunde liegenden Forschung ist es unter anderem, diverse Möglichkeiten der Personalisierung von Benutzerschnittstellen mobiler Systeme anhand von Beispielen zu ergründen und deren die Bedienung betreffende effizienzsteigernde Wirkung an prototypischen Implementierungen im Labor zu zeigen. Die zentrale wissenschaftliche Fragestellung besteht jedoch darin, auf welche Weise sich der für die Personalisierung notwendige Prozess der Verarbeitung von Sensordaten automatisieren lässt. Der konzeptionelle Schwerpunkt liegt also nicht auf der Entwicklung einer vom Benutzer gänzlich selbst zu konfigurierenden Benutzerschnittstelle, sondern auf der Entwicklung eines Systems, welches regelmäßiges Benutzerverhalten implizit erkennt und diese Informationen nutzt, um effizienzsteigernde Anpassungen der Benutzerschnittstelle möglichst automatisch herbeizuführen. Das Verhalten des Benutzers eines mobilen Systems wird jedoch durch kontextuelle Umstände wie z.B. der aktuellen Position und das aktuelle soziale Umfeld des Benutzers beeinflusst. Ziebart zeigt beispielsweise in [Ziebart u. a., 2008] anhand einer Umfrage, dass sowohl die Position des Fahrzeugs als auch die Verkehrslage einen Einfluss auf die Bedienung automotiver Benutzerschnittstellen ausüben. Für eine geeignete Personalisierung der Benutzerschnittstelle gilt es aus Sicht des Systems also nicht nur die Bedienung der Benutzerschnittstelle zu analysieren, sondern darüber hinaus auch die Situationen automatisiert zu erfassen und im Verarbeitungsprozess zu berücksichtigen.

Da automotive Benutzerschnittstellen nicht nur über ein breites Spektrum an Sensoren zur Erfassung der Situation verfügen, sondern sich aufgrund rechtlicher Bestimmungen möglichst effizient bedienen lassen müssen, eignet sich der automotive Bereich im besonderen Maße als beispielhaftes Anwendungsgebiet für die situationsabhängige bzw. kontextsensitive Personalisierung. So lassen sich z.B. auf Basis der kontextsensitiven Personalisierung in verschiedenen Situationen bedarfsgerecht unterschiedliche Varianten der automotiven Benutzerschnittstelle präsentieren, um wie am oben genannten Beispiel die Anzahl der notwendigen Bedienschritte zur Ausführung einer situationsbedingt bevorzugten Funktionalität zu reduzieren. Es werden zur Veranschaulichung des praktischen Nutzens der kontextsensitiven Personalisierung beispielhaft zwei unterschiedliche Möglichkeiten gegenübergestellt und diskutiert, wie mit Hilfe des erlernten situationsabhängigen Benutzerverhaltens die Bedienung des Bordcomputers sowohl ohne jegliches Zutun als auch durch bedingtes Eingreifen des Benutzers verbessert werden kann. Außerdem wird untersucht, inwiefern sich das dynamische Verhalten einer kontextsensitiv personalisierten Benutzerschnittstelle über einen längeren Zeitraum für Langzeittests oder zur Präsentation in Echtzeit simulieren lässt.

1.1 Motivation

Individuelle Anpassungen der Benutzerschnittstelle können aus Sicht des Benutzers vielerlei Vorzüge bieten. Da eine dynamische Benutzerschnittstelle im Gegensatz zur statischen Variante sich mit der Zeit verändert oder sich verändern lässt, ist es möglich den Wünschen unerfahrener und erfahrener Benutzer gleichermaßen gerecht zu werden. Beispielsweise können nicht benötigte Funktionalitäten ausgeblendet, aufwendig zu bedienende Funktio-

nalitäten vereinfacht oder thematisch unterschiedliche Funktionalitäten kombiniert dargestellt werden. Welche Art der Anpassung im konkreten Fall sinnvoll ist, kann entweder der Benutzer selber entscheiden oder es der Benutzerschnittstelle überlassen, eine geeignete Anpassung vorzuschlagen. Die Individualisierung der Benutzerschnittstelle kann also vom Benutzer eigenhändig - also bewusst - ausgeführt oder vom Bordcomputer automatisch ausgelöst werden.

Die bekannteste und oftmals unter dem Sammelbegriff *Customization* bezeichnete Anpassungsfähigkeit der Benutzerschnittstelle dient dem Benutzer dazu, z.B. das visuelle und klangliche Erscheinungsbild der Benutzerschnittstelle auf die individuellen Vorlieben abzustimmen. In allen modernen Betriebssystemen für Mobiltelefone ist es beispielsweise möglich, eigene Fotos als Hintergrundbilder für den Bildschirm festzulegen oder eigene Audioaufzeichnungen als Klingeltöne zu verwenden. Hierbei ergreift der Benutzer die Initiative und gestaltet die Benutzerschnittstelle nach seinen Wünschen. Dabei wird der Grad an Identifikation mit dem Produkt und die Selbstdarstellung durch das Produkt, die oftmals unter den Begriffen *Joy of Use* [Hatscher, 2001] oder *hedonische Qualität* [Hassenzahl, 2003] zusammengefasst werden, gefördert.

Eine weitere Art, Benutzerschnittstellen durch individuelle Anpassung attraktiver zu gestalten, ist es, das explorative Verhalten von Benutzern zu unterstützen und sogar zu fördern. Oft werden dabei dem Benutzer spezielle Inhalte präsentiert, die auf der Basis individueller Vorlieben zusammengestellt wurden. Diese Methode kann im Gegensatz zur vorherigen Individualisierung auch automatisch durchgeführt werden. Dabei kann der Bordcomputer die Vorlieben des Benutzers anhand der historischen Bedienung ermitteln und im weiteren Verlauf als Informationsfilter verwenden. Aus Sicht der Benutzer scheint bei dieser Art der Individualisierung das Produkt genau zu wissen, welche Informationen oder welche Funktionalitäten bis dato unbekannt sind, aber zukünftig interessant und nützlich sein könnten.

Automatische Anpassungen der Benutzerschnittstelle werden meistens jedoch dadurch motiviert, dass sich die Bedienung durch Modifikationen vereinfachen lässt. Zu den bekanntesten Beispielen zählt hierbei sicherlich die in vielen Betriebssystemen enthaltene Schnellstartleiste für Programme. Aufgrund des limitierten Anzeigebereichs auf dem Bildschirm wird hier versucht nur die am häufigsten benutzten Programme über eine begrenzte Anzahl immer erreichbarer Bedienelemente direkt ausführbar zu machen. Diese Maßnahme führt letztendlich zu einer Reduzierung aller vom Benutzer auszuführenden Interaktionen, die im Durchschnitt für die Anwahl und Ausführung eines Programms notwendig sind. In einem solchen Fall wird also die Effizienz der Bedienung und die damit verbundene pragmatische Qualität [Hassenzahl, 2003] des Produkts gesteigert.

Auch wenn die beiden eingangs erwähnten Varianten der Individualisierung in den letzten Jahren einen hohen Zuspruch bei den Geräteherstellern erfuhren, so ist es speziell im Bereich der automotiven Benutzerschnittstellen aufgrund der besonderen Anforderungen sinnvoll, die Effizienz der Bedienung zu steigern. Im Gegensatz zu Systemen, die sich im vollen Umfang aller physikalisch möglichen Eingabemodalitäten bedienen, ist die Eingabe der meisten Bordcomputer auf die in der Abbildung 1.1 rechts dargestellte, nur mit einer Hand steuerbare zentrale Bedieneinheit (ZBE) als Dreh-Drück-(Schiebe)-Steller mit Tasten oder auf die Sprachbedienung beschränkt. Aber aufgrund der bis dato nur sehr einschränkten Möglichkeiten der Spracherkennung bleibt die ZBE mit ihrer begrenzten Anzahl an Freiheitsgraden die primäre Eingabemodalität. Dies hat zur Folge, dass Dienste, die sich komfortabel und effizient nur mit beiden Händen steuern lassen, im Fahrzeug entweder nur sehr umständlich zu bedienen sind oder einer generellen Neugestaltung der Benutzeroberfläche bedürfen. Als Beispiel sei hier die mühevolle Eingabe einer Internetadresse über

die ZBE erwähnt. Da es mit der ZBE nicht möglich ist, Zeichen direkt einzugeben, ist es notwendig, ein gewünschtes Zeichen auf einer virtuellen Tastatur am Bildschirm (siehe Abbildung 1.1 links) durch die Positionierung eines Cursors zu selektieren und durch das Drücken der ZBE auszuwählen. Somit wird ein Vielfaches der Bedienschritte benötigt, die sich durch eine Eingabe mit einer realen Tastatur ergeben würden. Mit Hilfe einer personalisierten Liste der zuletzt aufgerufenen Internetadressen ließen sich die Bedienschritte in diesem Fall beispielsweise auf das Scrollen in der Liste und das Bestätigen einer Auswahl beschränken.



Abb. 1.1 Links: Nutzung der Bildschirmtastatur zur Eingabe des Straßennamens im Navigationssystem. Rechts: Zentrale Bedieneinheit (ZBE) als Dreh-Drück-(Schiebe)-Steller mit Tasten.

Im automotiven Umfeld führt die Reduzierung der notwendigen Bedienschritte jedoch nicht nur zu einem geringeren Interaktionsaufwand, sondern insbesondere zu einer Reduzierung der durch die Bedienung hervorgerufenen Ablenkung. Die Ergebnisse einer Benutzerevaluation [Tchankue u. a., 2011] zeigen dabei, dass sich durch den Einsatz sich selbst anpassender Benutzerschnittstellen im automotiven Bereich nicht nur die Ablenkung reduzieren, sondern auch das Vertrauen in die Benutzerschnittstelle erhöhen lässt. Wäre der Bordcomputer nun in der Lage, ein präzises Modell der individuellen Interaktionen eines Benutzers zu erstellen, so wäre es sogar möglich, in Einzelfällen ganz auf eine Interaktion zu verzichten. Für eine derart proaktiv unterstützende Benutzerschnittstelle ist es zunächst jedoch notwendig, die Präferenzen bzw. Gewohnheiten des Benutzers in geeigneter Weise zu erfassen. Dieser Prozess kann sowohl manuell durch eine Befragung des Benutzers durch den Bordcomputer, als auch implizit und damit unaufdringlich durch die automatische Analyse der Interaktionen geschehen. Eine Befragung setzt dabei voraus, dass der Benutzer sich seiner Gewohnheiten bzgl. der Bedienung des Bordcomputers bewusst ist. Da diese Annahme nicht in jedem Fall zu gewährleisten ist und da auch nicht jede vorstellbare Interaktion in allen erdenklichen Situationen aufgrund rechtlicher Bestimmungen ausgeführt werden darf, ist es speziell im automotiven Umfeld geeigneter, jede tatsächlich stattfindende Interaktion aufzuzeichnen und auf ihre maßgebliche Regelmäßigkeit hin zu prüfen. Der Bordcomputer könnte dann die regelmäßigen Interaktionen, die der Benutzer ohnehin in einer bestimmten Situation ausführen würde, automatisieren. Das Autoradio könnte sich beispielsweise selbstständig zu bestimmten Zeiten auf den in genau dieser Zeitperiode vom Benutzer bevorzugten Radiosender einstellen. Eine visuelle Ablenkung durch den Bordcomputer wäre in diesem Fall also ausgeschlossen.

1.2 Problem- und Aufgabenstellungen

Der Entwurf sich an den Benutzer anpassender Benutzerschnittstellen erfordert ein koordiniertes Zusammenspiel diverser Technologien und Verfahren aus den Bereichen der Konzeption, Entwicklung, Spezifikation, Implementierung, Test und Evaluation von Benutzerschnittstellen. Spezielle Herausforderungen ergeben sich dabei insbesondere aus der Tatsache, dass sich derartige Benutzerschnittstellen fortlaufend in Zuständen befinden, die sich durch einen hohen Grad an Variabilität auszeichnen. Beispielsweise kann der visuelle Anteil der Benutzerschnittstelle innerhalb eines Zustandes über die Zeit hinweg diverse Ausprägungen annehmen. In bestimmten Zuständen, die zum Zeitpunkt der Auslieferung des Produkts durch ein bestimmtes Verhalten charakterisiert werden, muss die Benutzerschnittstelle jedoch auch nach einer - durch die Personalisierung ausgelösten - Anpassung des zugrunde liegenden Verhaltens eine fehlerfreie Bedienung ermöglichen. Diese mitunter hohe Dynamik, die sich aus der Mannigfaltigkeit des individuellen Nutzungsverhaltens ergibt, muss dabei durch eine hoch dynamische Spezifikation eindeutig definiert, während der Entwicklung sorgfältig differenziert und durch neuartige Testmethoden validiert werden. Ausgehend von diesen Betrachtungen lassen sich diverse Themenbereiche mit unterschiedlichen Herausforderungen und Problemstellungen identifizieren, die im Nachfolgenden jeweils getrennt voneinander diskutiert werden.

Konzeption: Die konzeptionelle Ausgestaltung der Mechanismen zur automatischen Personalisierung in den jeweiligen Anwendungsgebieten beginnt bei der einfachen Vorbelegung von Eingabeparametern auf Formularen bis hin zu einer Modifikation der Dialogstruktur. Sowohl das Anwendungsgebiet mit seinen speziellen Gegebenheiten als auch der spezifische Anwendungsfall bestimmen dabei, in welchem Maß und auf welche Art die Personalisierung eine Benutzerschnittstelle verändert. Diese Forschungsarbeit widmet sich insbesondere dem Anwendungsgebiet der automotiven Benutzerschnittstellen und den damit einhergehenden Einschränkungen bei der Bedienung. Es gilt, automotive Anwendungsfälle der Personalisierung zu entwickeln, konzeptionell zu beschreiben und das Maß und die Art der jeweiligen Personalisierung an den speziellen Anforderungen im automotiven Umfeld zu messen.

Entwicklung: Für eine sich personalisierende Benutzerschnittstelle, die sich auf der Grundlage regelmäßigen Interaktionsverhaltens automatisch verändert, ist es notwendig, einen Prozess zu entwickeln, der jeden Aspekt von der Aufzeichnung der Interaktionen bis hin zur Ausführung der Personalisierung berücksichtigt. Der Prozess wird als eine Abfolge einzelner Verarbeitungsschritte verstanden, deren Ausführung letztendlich zu einer gegebenenfalls automatischen Anpassung der Benutzerschnittstelle führt. Die Verarbeitungsschritte sollen dabei möglichst autonom vom Bordcomputer im Hintergrund ausgeführt werden und kein Eingreifen des Benutzers erfordern. Kern dieser Arbeit ist die Entwicklung des Teilprozesses zur Ermittlung, Verarbeitung und Verwendung regelmäßigen Interaktionsverhaltens zum Zweck der Personalisierung. Dagegen wird die Visualisierung der Personalisierung in der Benutzeroberfläche und die Ausführung im System in dieser Forschungsarbeit als gegeben vorausgesetzt und nicht näher betrachtet. Prototypische Implementierungen der Benutzerschnittstelle und des Systems dienen lediglich zur beispielhaften Visualisierung der Ergebnisse und zur Evaluation.

Spezifikation: Die Entwicklung einer Benutzerschnittstelle, deren Verhalten sich an dem individuellen Verhalten eines noch unbekannten Benutzers orientiert, wird wie bereits oben erwähnt durch eine hoch dynamische Spezifikation der Benutzeroberfläche unterstützt werden müssen. Ohne eine für diesen Zweck entwickelte Spezifikationssprache oder spezielle Spezifikationskonstrukte besteht die Gefahr, dass die Benutzerschnittstelle durch eine Anpassung des Verhaltens in einen inkonsistenten Zustand versetzt werden kann. Dazu zählen Zustände, in denen beispielsweise aufgrund von Fehlern im strukturellen Aufbau der grafischen Benutzeroberfläche keine Bedienung mehr möglich ist. Außerdem ist es im automotiven Bereich notwendig, Anpassungen nur im Rahmen der gesetzlichen Vorschriften für Bordcomputer zu erlauben. Das unerlaubte Ausführen von Funktionalitäten, wie beispielsweise das Abspielen von Videos während der Fahrt, dürfen durch die automatische Personalisierung weder angeboten noch ausgelöst werden. Auch in diesem Fall ist eine eindeutige Spezifikation der potentiellen Anpassungen unerlässlich.

Der Themenbereich der Spezifikation wird dazu in drei Problembereiche unterteilt, die zwar eng miteinander verwoben, aber thematisch voneinander getrennt werden. Neben der Spezifikation der Benutzerschnittstelle und des Systems ist es notwendig zu spezifizieren, welchen Einfluss das Interaktionsverhalten auf die Personalisierung ausübt. Letzteres ist dann auch einer der für diese Arbeit relevanten Problembereiche. Dabei gilt es, den oben beschriebenen Teilprozess zur Ermittlung, Verarbeitung und Verwendung regelmäßigen Interaktionsverhaltes durch eine eindeutige Spezifikation der Personalisierung zu unterstützen. Die Art und Weise, wie das ermittelte Wissen über das individuelle Interaktionsverhalten in der Benutzerschnittstelle verwendet und welche Funktionalität auf Seiten des Systems letztendlich angesprochen wird, ist nicht Bestandteil dieser Spezifikation.

Implementierung: Die Implementierung von Mechanismen der Personalisierung unterliegt speziell im automotiven Bereich vielfältigen Restriktionen. Diese Einschränkungen ergeben sich insbesondere aus der Tatsache, dass es sich bei dem Bordcomputer um ein eingebettetes System handelt, welches unter den Gesichtspunkten des minimalen Energie- und Speicherverbrauchs entwickelt wird. Sowohl der Prozess der Verarbeitung als auch der Prozess der Persistenz großer Datenmengen wird dadurch stark beeinträchtigt. Aufgrund der langen Entwicklungszyklen in der Automobilindustrie von bis zu zehn Jahren ist es außerdem erforderlich, automotive Systeme zu entwicklen, die über Erweiterungsmöglichkeiten verfügen oder sich während der Laufzeit zumindest an neue oder noch unbekannte Situationen anpassen lassen, ohne dafür eine Werkstatt aufsuchen zu müssen. In dieser Arbeit gilt es also, für den Prozess der Personalisierung eine geeignete Wahl an Algorithmen zu verwenden, die sich durch einen effizienten Speicherbedarf kennzeichnen und sich gleichzeitig auch während des Einsatzes flexibel konfigurieren lassen. Darüber hinaus ist es aufgrund des geringen Datenspeichers notwendig, den durch die Bedienung des Bordcomputers generierten Datenstrom geeignet zu filtern und nur die für die Personalisierung erforderlichen Interaktionen im weiteren Verlauf der Verarbeitung zu verwenden. Für die Evaluation des zu entwickelnden Verfahrens gilt es außerdem, einen Prototypen zu implementieren, der den oben beschriebenen Teilprozess zur Ermittlung, Verarbeitung und Verwendung regelmäßigen Benutzerverhaltens mit der dazugehörigen Spezifikation der Personalisierung vollständig unterstützt. Die Benutzerschnittstelle mit den entsprechenden Möglichkeiten der Anpassung und das dahinter liegende System werden hierzu beispielhaft implementiert. Zur Simulation der Umgebung wird außerdem ein Werkzeug benötigt, welches den Prototyp mit den Daten aller kontextuellen Einflussfaktoren versorgt, die in den weiter unten beschriebenen Beispielszenarien als relevant eingestuft werden.

Test: Da Langzeittests für sich anpassende Benutzerschnittstellen in einer realen Umgebung über mehrere Jahre hinweg weder praktikabel noch umsetzbar sind, gilt es, das potentielle Benutzerverhalten in Form einzelner Szenarien zu simulieren und das Verhalten des Systems diesbezüglich zu prüfen. Besondere Problemstellungen ergeben sich dabei insbesondere bei der Simulation des Benutzerverhaltens. Einerseits sollte das zu simulierende Benutzerverhalten möglichst zufällig sein, andererseits aber auch einer gewissen Regelmäßigkeit unterliegen. Wird ein solches Benutzerverhalten unter den beschriebenen Voraussetzungen simuliert, so kann untersucht bzw. getestet werden, ob während des Analyseprozesses die gewünschte Regelmäßigkeit erkannt und korrekt verarbeitet wird. In dieser Forschungsarbeit gilt es also, eine Methode zu entwickeln, die es mit Hilfe eines zu untersuchenden Szenarios ermöglichen soll, regelmäßig stattfindendes Benutzerverhalten zu Testzwecken zu simulieren. Die eigentliche Testmethodik, nämlich der automatisierte Vergleich des simulierten Benutzerverhaltens mit den Ergebnissen des zu testenden Analyseprozesses, würde den Rahmen dieser Arbeit sprengen und wird daher nicht weiter behandelt.

Evaluation: Die im Abschnitt 1.1 aufgeführten Annahmen bzgl. des zusätzlichen Nutzens personalisierter Benutzerschnittstellen müssen außerdem in Form von Hypothesen formuliert und anhand einer szenarienspezifischen Evaluation überprüft werden. Dazu ist es notwendig, einen Prototyp der Benutzerschnittstelle und einen Fahrsimulator als Simulationsumgebung zu entwickeln, mit deren Hilfe die Probanden die kontextsensitiven Personalisierungen im Labor und in Echtzeit erfahren können. Begleitend erfolgt die Anfertigung eines geeigneten Fragebogens, der eine Bewertung der Personalisierungen aus Sicht der Benutzer ermöglicht. Eine solche Evaluation basiert also auf den subjektiven Eindrücken der Probanden, die während und nach der Bedienung des Prototyps und des Fahrsimulators erfasst werden.

1.3 Wissenschaftliche Beiträge

Diese Forschungsarbeit befasst sich im Kern mit der Entwicklung und Anwendung kontextsensitiver Personalisierung im Bereich der automotiven Benutzerschnittstellen. Die wesentlichen wissenschaftlichen Beiträge konzentrieren sich dabei auf das Verfahren zur Erkennung und Verarbeitung regelmäßigen Benutzerverhaltens. Außerdem werden potentielle Anwendungsfälle aus dem automotiven Bereich vorgestellt und auf der Basis des genannten Verfahrens beispielhaft implementiert. Ebenfalls neuartig ist das im Rahmen dieser Forschungsarbeit entwickelte Verfahren zur Simulation des für die Personalisierung notwendigen situationsabhängigen Benutzerverhaltens. Im Folgenden werden nun die wesentlichen wissenschaftlichen Beiträge im Kontext des entsprechenden Themenbereichs zusammenfassend aufgeführt:

- **Anwendungen der kontextsensitiven Personalisierung:** Im Verlauf der Ausarbeitung wurden zwei unterschiedliche Konzepte entwickelt, um automotive Benutzerschnittstellen in Abhängigkeit von der Situation zu personalisieren. Beide Konzepte verfolgen dabei das gemeinsame Ziel, die Bedienung des Bordcomputers weitestgehend durch Automatisierungen bzw. Teilautomatisierungen zu beschleunigen. Während ein Konzept vorsieht, den Benutzer auf oft in bestimmten Situationen ausgeführte Funktionen ausschließlich

hinzuweisen, wird im alternativen Konzept die automatische Ausführung eben dieser Funktionalität propagiert. Im zuerst genannten Konzept werden dabei in bestimmten Situationen die vom Benutzer bevorzugten Funktionalitäten angezeigt und können dann manuell durch den Benutzer ausgeführt werden. Der Benutzer entscheidet also während einer bestimmten Situation, ob die von der Benutzerschnittstelle vorgeschlagene Funktionalität für ihn relevant ist und ausgeführt werden soll. Bevor es im alternativen Konzept zur automatischen Ausführung der bevorzugten Funktionalität kommt, bedarf es einer einmaligen Bestätigung durch den Benutzer. Diese erfolgt auf Grundlage einer Auflistung bzw. Visualisierung aller vom Bordcomputer ermittelten individuellen Bedienmuster mit den dazugehörigen Situationen. Dadurch ist der Benutzer in der Lage zu entscheiden, ob in der entsprechenden Situation eine vom System ermittelte Funktionalität stets automatisch, also ohne eine weitere Bestätigung, ausgeführt werden soll. Auf diese Art und Weise können prinzipiell alle häufig verwendeten Funktionalitäten der automotiven Benutzerschnittstelle kontextsensitiv personalisiert werden. Um den praktischen Nutzen der Konzepte im Fahrzeug zu verdeutlichen, wurden außerdem beispielhaft vier Szenarien ausgearbeitet, die sich für den Einsatz der kontextsensitiven Personalisierung eignen.

- **Kontextsensitive Personalisierung durch einen Dienst:** Das in dieser Forschungsarbeit vorgestellte Verfahren zur kontextsensitiven Personalisierung basiert grundlegend auf der Idee, den Prozess der Analyse des Benutzerverhaltens von dem Prozess der Anwendung der Personalisierung in der Benutzerschnittstelle logisch zu trennen. Im Gegensatz zum Ansatz von Coutand [Coutand, 2008], in dem eine ähnliche Trennung der Ausführung einer individuellen Modifikation der Benutzerschnittstelle von der Verhaltensanalyse vorausgesetzt wird, dient eine solche Trennung in dem hier vorgestellten Verfahren nicht in erster Linie dazu, individuelle Präferenzen aus Sicherheitsgründen zentral zu verwalten oder applikationsübergreifend zu verwenden, sondern den durch relativ hohen Rechenaufwand gekennzeichneten Analyseprozess auch außerhalb eines eingebetteten Systems ausführen zu können. Zukünftige Bordcomputer, die das hier entwickelte Verfahren umsetzen, sind dann in der Lage die Benutzerschnittstellen lokal innerhalb des Fahrzeugs auszuführen, während die Analyse durch einen externen Service bereitgestellt wird. Sowohl die rechenintensive Ermittlung typischer Bedienmuster als auch die Bestimmung eines geeigneten Zeitpunktes für die Anwendung der Personalisierung können dadurch beispielsweise in eine Cloud ausgelagert werden.

- **Interaktion als Sequenz von Ereignissen und Kontexten:** Die Analyse des Benutzerverhaltens findet hier ausschließlich auf Basis von Ereignissen statt, die im Laufe der Bedienung in der Benutzerschnittstelle generiert werden. Der Vergleich historischer Ereignisse mit aktuellen Ereignissen gibt dann Aufschluss darüber, welches Benutzerverhalten in bestimmten Situationen zu erwarten ist. Der wesentliche Unterschied des hier vorgestellten Verfahrens gegenüber den bekannten Lösungen der kontextsensitiven Personalisierung ist die Abstraktion einer Interaktion als Sequenz von Ereignissen. Die während einer Interaktion erzeugten Ereignisse, wie beispielsweise die aufeinander folgenden Betätigungen verschiedenster Schalter, werden in dem hier vorgestellten Ansatz gegebenenfalls in kausale Abhängigkeit zueinander gesetzt. Erst das zeitlich geordnete Auftreten eines oder mehrerer kausal abhängiger Ereignisse kann dann dazu führen, dass sich die Benutzerschnittstelle auf Basis der sich daraus ableitenden regelmäßigen Interaktion personalisiert. In den bisherigen Ansätzen wurden Ereignisse, die aufgrund ihres regelmäßigen Auftretens zu einer kontextsensitiven Personalisierung beitrugen, nur einzeln

und unabhängig voneinander betrachtet. Im Fall der automotiven Benutzerschnittstelle ist es jedoch möglich, dass sich eine zu untersuchende Interaktion dadurch kennzeichnet, dass sie auf zwei oder mehr kausal abhängigen Ereignissen basiert, die möglicherweise in unterschiedlichen Kontexten anzutreffen sind. Eine Interaktion besteht hier also nicht nur aus einer Sequenz von Ereignissen, sondern auch aus einer Sequenz evtl. verschiedener Kontexte.

- **Dynamische Konfiguration der kontextsensitiven Personalisierung:** Da im Umfeld der automotiven Benutzerschnittstellen bestimmte Bedienmuster von Interesse, rechtlich erlaubt oder zum Zeitpunkt der Entwicklung evtl. noch unbekannt sind, war es nötig ein Verfahren zu entwickeln, dass sich durch eine Konfiguration jederzeit parametrisieren lässt. In der Konfiguration wird dabei allgemein beschrieben, welche Klassen von Interaktionen durch den Analyseprozess erkannt werden können und auf welche Art und Weise die ermittelten Bedienmuster zu einer kontextsensitiven Personalisierung beitragen. Diese Zweiteilung basiert auf einer zweistufigen Verarbeitung: Ermittlung typischen Benutzerverhaltens und Erkennung von Situationen, in denen ein typisches Benutzerverhalten zu erwarten ist. Durch diese flexible Konfiguration ist es außerdem möglich, in Situationen zu personalisieren, die nicht in allen Aspekten mit den Situationen übereinstimmen müssen, die sich durch die entsprechenden Bedienmuster ergeben. Dies eignet sich besonders zur Definition von zusätzlichen Nebenbedingungen, die zum Zeitpunkt der Personalisierung erfüllt sein müssen.

- **Simulation situationsabhängigen Benutzerverhaltens:** Im Rahmen dieser Forschungsarbeit wurde außerdem eine neue Methode zur Simulation regelmäßigen und situationsabhängigen Benutzerverhaltens entwickelt. Die zu Testzwecken entworfene Methode umschreibt dabei eine spezielle Vorgehensweise sowohl bei der Modellierung als auch bei der Generierung und Simulation regelmäßigen Benutzerverhaltens. Bei der Anwendung der Methode wird zunächst das Benutzerverhalten, also ein zu untersuchendes Szenario, durch eine qualitative Beschreibung der charakteristischen Sequenz von Ereignissen modelliert. In einem zweiten Schritt werden beliebige Sequenzen von Ereignissen auf der Grundlage des Modells generiert und im letzten Schritt zeitlich geordnet simuliert.

1.4 Überblick

Die Forschungsarbeit beginnt im Kapitel 2 zunächst mit der Definition grundlegender Begrifflichkeiten aus dem Bereich adaptiver Systeme. Nach einer ersten informellen Beschreibung der Eigenschaften kontextsensitiv personalisierbarer Systeme i. Allg. folgt im Abschnitt 2.2 die Einführung des fundamentalen Konzepts der Personalisierung und der damit einhergehenden Benutzermodellierung. Dabei werden nicht nur die unterschiedlichen Formen der Benutzerinformationen, sondern auch die dazu jeweils passenden Methoden der Datenanalyse vorgestellt. Der anschließende Abschnitt 2.3 konzentriert sich dahingegen insbesondere auf den Bereich automotiver Benutzerschnittstellen und den damit verbundenen domänenspezifischen Anforderungen, die eine personalisierbare, automotive Benutzerschnittstelle aus Sicht des Benutzers zu erfüllen hat. Ergänzt wird der Abschnitt 2.3 um eine kurze Einführung der sich für die Benutzermodellierung im Fahrzeug eignenden Methoden der Mustererkennung und des nicht-überwachten Gruppierens. Abschließend werden vier

potentielle Anwendungsfälle der kontextsensitiven Personalisierung im automotiven Bereich vorgestellt, die in den darauf folgenden Kapiteln im Sinne eines laufenden Beispiels wiederholt als Exempel aufgegriffen werden.

Das Kapitel 3 widmet sich unter anderem allen bekannten wissenschaftlichen Forschungsarbeiten, die den Bereich der kontextsensitiven Personalisierung inhaltlich entweder nur tangieren oder der kontextsensitiven Personalisierung direkt zugeordnet werden können. Während im Unterabschnitt 3.1.1 zunächst diverse Methoden der Datenerfassung eingeführt werden, konzentriert sich der Unterabschnitt 3.1.2 dann gänzlich auf den Stand der Technik bzgl. aller - zum Zeitpunkt der Anfertigung der Forschungsarbeit - bekannten Methoden der kontextsensitiven Personalisierung. Unterschieden wird dabei zwischen allgemeinen Methoden, die in diversen Anwendungsgebieten wie z.B. der Heimautomatisierung eingesetzt werden können und speziellen Methoden, die hauptsächlich in mobilen Geräten zum Einsatz kommen. Nach einer kurzen Zusammenfassung aller betreffenden Anwendungsgebiete der kontextsensitiven Personalisierung folgt eine Erörterung und Diskussion der beschriebenen Forschungsansätze und Lösungsvorschläge. Der zweite Teil des Kapitels beschäftigt sich dann im Abschnitt 3.2 mit dem Stand der Technik hinsichtlich der Simulation kontextsensitiven Benutzerverhaltens. Dabei werden zunächst einige Verfahren vorgestellt, die eine reproduzierbare Simulation der Sensordaten sowohl des Benutzers als auch des dazugehörigen Kontextes auf Basis von Modellen ermöglichen. Die im darauf folgenden Unterabschnitt 3.2.2 beschriebenen Forschungsansätze erlauben es dann außerdem noch, den Benutzer und den Kontext in Echtzeit innerhalb virtueller Umgebungen zu simulieren.

Der zweite Teil dieser Forschungsarbeit beginnt im Kapitel 4 mit einer vollständigen Beschreibung des im Rahmen dieser Untersuchung entwickelten Verfahrens der kontextsensitiven Personalisierung. Im Abschnitt 4.1 wird dazu zunächst das grundlegende Konzept einer für die kontextsensitive Personalisierung notwendigen und in einen Systemdienst auszulagernden Datenanalyse vorgestellt. Im anschließenden Abschnitt 4.2 wird dann jedem der vier notwendigen Prozessschritte, die während der Datenanalyse ausgeführt werden, ein eigener Unterabschnitt gewidmet. Nach einer kurzen Erläuterung der in einem ersten Prozessschritt durchzuführenden Vorverarbeitung kontextbezogener Ereignisse folgt in Unterabschnitt 4.2.2 die Einführung einer spezifischen Methode zur Beschreibung temporaler Ereignismuster. Der Unterabschnitt 4.2.3 stellt dann eine Methode vor, mit der erkannte Bedienmuster anhand ihrer Kontexte gruppiert werden können. Der letzte Prozessschritt, der sich der Berechnung des Zeitpunktes zur Ausführung der kontextsensitiven Personalisierung widmet, wird dann im Unterabschnitt 4.2.4 beschrieben. Abgeschlossen wird das Kapitel im Unterabschnitt 4.2.5 mit einer Analyse der Komplexität des vorgestellten Verfahrens.

Der Aufbau und die Struktur der prototypischen Umsetzung des Verfahrens, der dazu passenden automotiven Benutzerschnittstelle und der Komponenten zur Simulation werden im Kapitel 5 thematisiert. Im Abschnitt 5.1 wird dazu die prototypische Implementierung einer kontextsensitiv personalisierbaren, automotiven Benutzerschnittstelle - hier als HMI-Prototyp bezeichnet - vorgestellt. Der Abschnitt enthält neben einer Beschreibung der allgemeinen Bedienung und technischen Umsetzung der Benutzerschnittstelle auch einen Unterabschnitt, der sich speziell mit den hier entwickelten Erweiterungen der Benutzerschnittstelle um kontextsensitiv personalisierbare Bedienelemente beschäftigt. Der darauf folgende Abschnitt 5.2 widmet sich - anhand eines Beispiels - der über die Vorlagen stattfindenden Parametrisierung des prototypischen Systemdienstes. Dabei werden sowohl die Schnittstelle zwischen dem Systemdienst und der Benutzerschnittstelle/Anwendung als auch die technische Umsetzung konkretisiert. Der letzte Abschnitt des Kapitels stellt dann ein im

Rahmen dieser Forschungsarbeit speziell für den HMI-Prototyp entwickeltes Werkzeug zur Simulation des Kontextes vor.

Das Kapitel 6 erörtert zu Beginn diverse Möglichkeiten, das Benutzer- und Umgebungsverhalten zu modellieren und anschließend zu simulieren. Im Abschnitt 6.2 erfolgt dabei die Beschreibung der quantitativen Simulation als mögliches Mittel zur Untersuchung kontextsensitiv personalisierbarer Systeme. Dazu wird das Benutzer- und Umgebungsverhalten aus einem der Anwendungsfälle aus Unterabschnitt 2.3.4 zunächst beispielhaft modelliert. Eine Skizzierung des algorithmischen Ablaufs der Generierung potentiellen Benutzer- und Umgebungsverhaltens auf Basis des erstellten Modells kann dann dem Unterabschnitt 6.2.2 entnommen werden. Eine kurze Einführung zur technischen Umsetzung, die in Form eines prototypischen Werkzeugs vorliegt, erfolgt abschließend im letzten Unterabschnitt.

Die während der Untersuchung der kontextsensitiven Personalisierung automotiver Benutzerschnittstellen durchgeführte Benutzerevaluation ist das Thema des siebten Kapitels. Der Abschnitt 7.1 widmet sich zunächst allen Fragestellungen und Hypothesen, die sich im Rahmen einer im Labor durchgeführten Benutzerevaluation bzgl. kontextsensitiv personalisierbarer automotiver Benutzerschnittstellen beantworten lassen. Daraufhin werden im Unterabschnitt 7.2 die diversen Fragebögen und Untersuchungsmethoden vorgestellt, die bei der Benutzerevaluation zum Einsatz kamen. Der grundlegende Ablauf der Befragung eines einzelnen Probanden, also die Reihenfolge, in der die Fragebögen ausgehändigt wurden, der Prototyp vorgestellt wurde und beispielhaft vom Probanden ausprobiert werden konnte, wird in Unterabschnitt 7.2.2 beschrieben. Die Aufführung, Auswertung und Bewertung aller Ergebnisse der Benutzerevaluation erfolgt dann im Unterabschnitt 7.2.3.

Im vierten und letzten Teil der Forschungsarbeit werden im Abschnitt 8.1 noch einmal alle Problemstellungen beschrieben, die es während der Entwicklung einer kontextsensitiven personalisierten automotiven Benutzerschnittstelle zu lösen galt. Jede Problemstellung wird dabei von einer Zusammenfassung des jeweils dazu vorgestellten Lösungsansatzes begleitet. Etwaige Voraussetzungen, die vor dem Einsatz der kontextsensitiven Personalisierung erfüllt sein müssen und Einschränkungen, die mit dem Einsatz des vorgestellten Verfahrens einhergehen, werden im Abschnitt 8.2 diskutiert. Alle Forschungsfragen, die in der Forschungsarbeit unbeantwortet bleiben, aber dennoch einer gründlichen Untersuchung unterzogen werden müssen, sind Bestandteil des Abschnitts 8.3. Im Abschnitt 8.4 werden abschließend mögliche Verbesserungen oder potentielle Erweiterungen des Prozesses der Datenanalyse und der kontextsensitiv personalisierbaren automotiven Benutzerschnittstelle vorgestellt und die Anwendungsmöglichkeiten des hier beschriebenen Verfahrens skizziert.

Kapitel 2
Grundlagen

In diesem Kapitel werden die wesentlichen Eigenschaften *adaptiver Systeme* und deren Besonderheiten im Anwendungsgebiet der *adaptiven Benutzerschnittstellen* diskutiert. Dazu werden zunächst grundlegende Begrifflichkeiten, die bis dato im Umfeld der Entwicklung adaptiver Systeme häufig nur informell beschrieben wurden, eingeführt und gegebenenfalls definiert. Genauer untersucht werden dabei adaptive Benutzerschnittstellen, die sich sowohl abhängig vom aktuellen Umfeld als auch abhängig vom historischen Gebrauch verändern. Die für die Umsetzung sich personalisierender Benutzerschnittstellen notwendigen Techniken des maschinellen Lernens werden dazu zusammenfassend vorgestellt und deren Anwendung im Umfeld der adaptiven Benutzerschnittstellen erläutert. Außerdem werden die besonderen Anforderungen und Beschränkungen formuliert, die es bei der Anwendung der kontextsensitiven Personalisierung in der Mensch-Maschine-Schnittstelle von Fahrzeugen zu beachten gilt. Abgeschlossen wird das Kapitel mit der Einführung von Szenarien bzw. Anwendungsfällen, die sich speziell für eine mögliche Anwendung der kontextsensitiven Personalisierung in *In-Car-Infotainment-Systemen* eignen.

2.1 Adaptive Systeme

Ein adaptives System umschreibt im alltäglichen Sprachgebrauch ein technisches Objekt, dessen Systemverhalten sich dem Verhalten ein oder mehrerer Subjekte, die von dem technischen Objekt entweder direkt Gebrauch machen und/oder als dessen Kontext fungieren, anpasst. Bei einem Subjekt kann es sich dabei um einen Person, ein Lebewesen i. Allg. oder ein weiteres technisches Objekt handeln. Zwischen den Subjekten und dem technischen Objekt bestehen also Wechselwirkungen, die sich durch den Austausch von Nachrichten, welche von den Subjekten oder dem technischen Objekt bzw. System erzeugt werden, kennzeichnen. Diese bidirektionale Kommunikation erlaubt dann auch die Erfassung des Systemverhaltens eines adaptiven Systems durch eine Gegenüberstellung der dem System zugeführten Eingaben mit den vom System produzierten Ausgaben. Das zu untersuchende System verfügt demnach über *Kommunikationskanäle*, die der Interaktion mit den umgebenden Subjekten dienen. Jede Nachricht wird dabei von einem bestimmten Sender in der jeweiligen Domäne aufgrund eines dort stattgefundenen Vorfalls produziert und dem Empfänger zugeführt. Die Nachrichten repräsentieren also Ereignisse, die entweder zu einem bestimmten Zeitpunkt im System oder den Subjekten stattgefunden haben oder über einen bestimmten Zeitraum hinweg stattfinden. Laut der Definition 2.1 von Etzion [Etzion

u. Niblett, 2010] kann sowohl der Vorfall als auch die daraus folgende maschinenlesbare Entität als Ereignis bezeichnet werden.

Definition 2.1 (Ereignis [Etzion u. Niblett, 2010]). An event is an occurrence within a particular system or domain; it is something that has happened, or is contemplated as having happened in that domain. The word event is also used to mean a programming entity that represents such an occurrence in a computing system.

Diese Forschungsarbeit beschäftigt sich jedoch nicht mit den jeweiligen Ursachen oder Eigenschaften eines Vorfalls, sondern mit dessen Verarbeitung innerhalb des Systems. Daher wird im Nachfolgenden das Objekt, welches aufgrund eines bestimmten Vorfalls produziert wurde und einen Kommunikationskanal als Ein- oder Ausgabe passiert, als Ereignis bezeichnet. Interagieren ein oder mehrere Subjekte mit dem System, so kommt es über einen gewissen Zeitraum zu einem wechselseitigen Austausch mehrerer Ereignisse. Gegeben sei nun die Menge E aller möglichen Ereignisse. Da ein Ereignis von einem Sender zu einem bestimmten Zeitpunkt produziert wird, existiert außerdem eine Funktion $t_{event} : E \to \mathbb{R}$, die jedem Ereignis einen bestimmten Zeitstempel zuweist. Daraus ergibt sich für den wechselseitigen Austausch von Ereignissen der folgend definierte Begriff:

Definition 2.2 (Interaktion). Eine Interaktion ist eine zeitlich geordnete Menge von Ereignissen, die einem System in einem bestimmten Zeitintervall zugeführt und entnommen wird. Die Menge möglicher Interaktionen ist daher gegeben durch:

$$I = \{(e_1, ..., e_n) | e \in E \wedge n > 0 \wedge t_{event}(e_{i-1}) < t_{event}(e_i)\} \text{ (Interaktionen)}$$

Anders als bei der Definition von Dix [Dix u. a., 2004], die eine Interaktion als *„... any communication between a user and computer, ...“* bezeichnet, wird die Interaktion durch die Definition 2.2 allgemeiner beschrieben. Das mit dem System interagierende Subjekt kann nach dieser Definition auch ein Lebenwesen i. Allg. oder ein technisches Objekt sein.

Das Verhalten eines beliebigen Systems ist durch eine Funktion $V : \check{I} \to \hat{I}$ gekennzeichnet, die alle dem System zugeführten Eingaben auf spezifische Ausgaben abbildet. Die beiden Symbole „ˇ“ und „ˆ“ kennzeichnen dabei im Folgenden die Richtung der Kommunikation. Alle Eingaben zum System werden mit „ˇ“ und alle Ausgaben vom System mit „ˆ“ annotiert. Die Menge $\check{I}$ enthält somit beispielsweise nur Ereignisse, die dem System zugeführt werden. Herkömmliche bzw. reaktive Systeme berücksichtigen bei der Ausführung - also der Bestimmung einer Ausgabe - nur die aktuellste Eingabe, die dem System unmittelbar vor der Berechnung von V zugeführt wurde. Das Charakteristikum adaptiver Systeme besteht hingegen darin, dass auch ein wesentlicher Teil der historischen Eingaben das Systemverhalten beeinflussen. Angelehnt an die Umschreibung von *Self-adaptive Software* aus dem Broad Agency Announcement der DARPA aus dem Jahr 1997 [Laddaga, 1997] wird ein adaptives System daher wie folgt definiert:

Definition 2.3 (Adaptives System). Ein offenes System ist adaptiv, wenn es sein Verhalten auf Grundlage historischer Eingaben evaluiert und sich immer dann automatisch modifiziert, wenn sich aus der Evaluation ein effizienteres Verhalten ableiten lässt.

Jedes adaptive System verfügt demnach über ein oder mehrere Eigenschaften, die es laufend zu überprüfen und durch Modifikationen des Systemverhaltens automatisch zu optimieren gilt. Welche Eigenschaften im Einzelnen untersucht werden, hängt vom Einsatzgebiet des Systems ab. Während ein mit einem Menschen interagierendes System z.B. die

Fehlerrate evaluiert, die sich nach einer gewissen Zeit bzgl. der Bedienung des Systems einstellt, kann die Steuerelektronik des Fahrwerks eines Fahrzeugs über Beschleunigungssensoren messen, wie stark das Fahrzeug abhängig von den örtlichen Straßenbedingungen vibriert. Im ersten Beispiel ließe sich automatisch die Schriftgröße anpassen, um proaktiv auf Bedienfehler zu reagieren und damit die Fehlerrate zu reduzieren. Im zweiten Beispiel könnte die Steuerelektronik des Fahrwerks z.B. einen straßenspezifischen Dämpfungsfaktor für die aktive Federung einführen, um das Maß der Vibrationen während einer Fahrt auf unterschiedlichen Straßen möglichst gering zu halten.

2.1.1 Benutzeradaptives System

Adaptive Systeme werden in diversen Anwendungsgebieten wie beispielsweise im Bereich der autonom fahrenden Fahrzeuge dazu verwendet, unter Beachtung des Kontextes eine gewünschte Funktionalität zu gewährleisten. Solche Systeme können in Form von Softwarekomponenten vorliegen, die innerhalb eines eingebetteten Systems und unabhängig vom Fahrer Teile des Fahrzeugs autonom steuern. Andere technische Objekte stehen dabei stellvertretend für das Subjekt, während der Kontext durch eine Reihe von Sensoren bereitgestellt wird, die ihrerseits ebenfalls technische Objekte sind.

Diese Forschungsarbeit untersucht jedoch ausschließlich adaptive Systeme, die durch einen oder mehrere Menschen bedient bzw. verwendet werden können. Das heißt, mindestens ein Subjekt wird durch einen Menschen bzw. Benutzer repräsentiert. Ein solches adaptives System verfügt infolgedessen über eine Benutzerschnittstelle, die der Mensch-Computer-Interaktion dient. Dabei können Benutzer in der Benutzerschnittstelle Ereignisse auslösen, die dann dem adaptiven System zugeführt werden. Auch in umgekehrter Richtung kann ein adaptives System durch entsprechende Ereignisse die Benutzerschnittstelle verändern. Alle Kommunikationskanäle, die der Interaktion zwischen dem Benutzer und dem System dienen, werden zusammenfassend als Benutzerschnittstelle bezeichnet. Sie kann diverse Formen annehmen, die jeweils dem Anwendungsgebiet entsprechend ausfallen. Benutzerschnittstellen können aus einfachen Bildschirmen bestehen und nur eine unidirektionale Kommunikation ermöglichen oder aber aus einer Kombination verschiedenster Modalitäten zusammengesetzt sein, um sowohl bidirektionale als auch ineinander greifende multimodale Kommunikation zu ermöglichen.

Definition 2.4 (Benutzeradaptives System [Jameson, 2008]). A user-adaptive system can be defined as an interactive system that adapts its behavior to individual users on the basis of processes of user model acquisition and application that involve some form of learning, inference, or decision making.

Mit der Definition 2.4 führt Jameson den speziellen Begriff des *benutzeradaptiven Systems* ein. Er umschreibt damit ein spezifisches adaptives System, dass die Eingaben über eine Benutzerschnittstelle entgegen nimmt, die Ausgaben über eine Benutzerschnittstelle nach außen trägt und sich gegebenenfalls von mehreren Benutzern verwenden lässt. Diese Definition konkretisiert außerdem die Art und Weise, mit der die Eingaben der Benutzer innerhalb des Systems verarbeitet werden müssen. Damit beschränkt sich Jameson auf Systeme, die mit Hilfe diverser Lerntechniken Informationen über den Benutzer in einem sogenannten *Benutzermodell* zusammentragen, um darauf aufbauend das Systemverhalten

zu modifizieren[1]. Ein benutzeradaptives System ist also aufgrund der Berücksichtigung der Historie dadurch gekennzeichnet, dass eine bestimmte Eingabe des Benutzers nicht immer zu der gleichen Ausgabe in der Benutzerschnittstelle führen muss.

Heutige automotive Benutzerschnittstellen legen - bis auf wenige Ausnahmen - keine Informationen über den Benutzer an. Sie reagieren auf bestimmte Benutzereingaben daher immer mit den gleichen Ausgaben. Es handelt sich daher lediglich um *reaktive* bzw. *benutzersensitive Systeme*. Eine automotive Telefonapplikation ist hingegen heute schon in der Lage, den Verlauf der getätigten Anrufe in einem Benutzermodell zu verwalten. Die Effizienz der Bedienung wird dadurch gesteigert, dass der Benutzer die Möglichkeit erhält, eine bestimmte Telefonnummer über die Liste der am häufigsten gewählten Telefonnummern auszuwählen, statt sie mühevoll über die Bildschirmtastatur oder ein Zahlenfeld eingeben zu müssen. Trotz des einfachen Benutzermodells und einer trivialen Lerntechnik kann eine solche Telefonapplikation aufgrund dieser Eigenschaften der Gruppe der benutzeradaptiven Systeme zugeordnet werden.

2.1.2 Benutzer- und kontextadaptives System

Typischerweise wird ein adaptives System, welches durch einen Menschen bedient wird, in einem bestimmten Kontext verwendet. Das Systemverhalten der Klimaautomatik im Fahrzeug hängt beispielsweise nicht nur davon ab, auf welche Temperatur das System vom Benutzer eingestellt ist, sondern welche Temperatur aktuell im Fahrzeuginneren gemessen wird. Für die weiteren Ausführungen bedarf es also einer genaueren Untersuchung dieses speziellen Bereichs der realen Welt, der das Systemverhalten ebenfalls beeinflussen kann.

Umgangssprachlich wird unter dem Begriff Kontext zunächst die Summe aller Subjekte verstanden, die sich in unmittelbarer Nähe zum adaptiven System befinden und mit diesem interagieren. Dies beinhaltet alle materiell vorhandenen Objekte wie das Innere eines Fahrzeugs bestehend beispielsweise aus Steuergeräten, dem Fahrer oder sogar mehreren Fahrzeuginsassen. Da jedoch beispielsweise auch die Zeit oder die Außentemperatur des Fahrzeugs das Verhalten des Systems beeinflussen können, wird die Bedeutung des Kontextes um immaterielle Bestandteile erweitert. Der Kontext kann also sowohl aus physikalisch vorhandenen Objekten als auch aus virtuellen Objekten zusammengesetzt sein, die von physikalischen Objekten wie einer Uhr oder einem Thermostat bereitgestellt werden. Brown [Brown, 1996] fasst den Kontext demnach wie folgt zusammen: *„... a combination of elements of the environment that the user's computer knows about.“*. Danach besteht der Kontext also aus allen Elementen der Umgebung, die sich mit dem offenen System ein oder mehrere Kommunikationskanäle teilen. Dey geht in [Dey, 2001] jedoch noch einen Schritt weiter und präzisiert den Kontext wie folgt: *„Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.“*. Damit beschreibt der Kontext in diesem Fall nicht die Elemente selber, sondern repräsentiert durch Informationen den Zustand, in dem sich die Umgebung zu einem gewissen Zeitpunkt oder über einen begrenzten Zeitraum hinweg befindet. Im Gegensatz zu Brown beschränken sich die Informationen des Kontextes bei Dey jedoch nur auf einen bestimmten Ausschnitt aus der Umgebung. Dieser Ausschnitt betrifft

[1] Eine detaillierte Beschreibung der allgemeinen Lernverfahren und eine Definition des Benutzermodells folgt in 2.2.1.

dabei nur die Aspekte der Umgebung, die in irgendeiner Weise für die Kommunikation zwischen Mensch und Applikation relevant sind.

Dies impliziert das Vorhandensein eines offenen Systems, auf dem die Applikation ausgeführt wird, eines Benutzers, der mit der Applikation/dem System interagiert und der Umgebung, die in Form eines weiteren Subjekts dem System Informationen über den Kontext bereitstellt. Auf Basis dieser allgemeingültigen Konstellation, die aus zwei Subjekten und einem System besteht, lassen sich nach Broy [Broy u. a., 2009] vier Arten von Verhaltensweisen klassifizieren, die ein beliebiges System in dieser Konstellation aufweisen kann:

- **Nicht-(kontext)adaptives Verhalten.** Das System zeigt ein Verhalten, welches ausschließlich von den Eingaben des Benutzers und der inneren Struktur des Systems abhängt. Die Umgebung hat keinen Einfluss auf das Verhalten des Systems.

- **Nicht-(kontext)transparentes adaptives Verhalten.** Das System zeigt ein Verhalten, welches von den Eingaben des Benutzers und der Umgebung abhängt. Es besteht jedoch keine Wechselwirkung zwischen dem Benutzer und der Umgebung.

- **Transparentes (kontext)adaptives Verhalten.** Das System zeigt ein Verhalten, welches von den Eingaben des Benutzers und der Umgebung abhängt. Zusätzlich ist es dem Benutzer möglich die Umgebung zu beobachten, aber nicht zu beeinflussen. Aus Sicht des Benutzers wird das Verhalten des Systems nachvollziehbar bzw. transparent.

- **Gesteuertes (kontext)adaptives Verhalten.** Das System zeigt ein Verhalten, welches von den Eingaben des Benutzers und der Umgebung abhängt. Außerdem ist es dem Benutzer neben der Beobachtung der Umgebung möglich, die Umgebung zu beeinflussen.

In dieser Beschreibung nach Broy, in der die Umgebung, die Kommunikation zwischen den beteiligten Parteien und der Einfluss der Subjekte auf das Systemverhalten in den Vordergrund der Definition (kontext)adaptiver Systeme gestellt wird, bleibt jedoch die bei adaptiven Systemen nach Definition 2.3 als wesentlich zu betrachtende Eigenschaft, nämlich der Einfluss der Historie aller Eingaben auf das aktuelle Verhalten, unberücksichtigt. Ein beliebiges System kann demnach bereits ein (kontext)adaptives Verhalten zeigen, wenn es im Sinne eines *kontextsensitiven Systems* auf die Vorfälle aus der Umgebung reagieren kann, ohne die historischen Vorfälle aus der Umgebung oder des Benutzers bei der Reaktion zu berücksichtigen. Die Berechnung einer geeigneten Reaktion erfolgt in diesem Fall auf Grundlage fester Regeln, die sich im Laufe der Benutzung nicht auf die speziellen Bedürfnisse bzw. Umgebungen des Benutzers ausrichten, sondern zum Zeitpunkt der Entwicklung determiniert werden. Es handelt sich daher lediglich um ein reaktives System, das sich abhängig von der Umgebung immer gleich verhält.

Die Gliederung der unterschiedlichen Verhaltensweisen nach Broy erweist sich dennoch als hilfreich, da sich damit aus Sicht des Benutzers verschiedene adaptive und reaktive Systeme charakterisieren lassen. Zeigt ein System das nach Broy beschriebene nicht-(kontext)adaptive Verhalten und wird die Historie der Eingaben bei der Ausführung berücksichtigt, so handelt es sich in dieser Forschungsarbeit folglich um ein benutzeradaptives System. Wird das Systemverhalten zusätzlich von der Umgebung beeinflusst, so ist es für die Bestimmung der spezifischen Ausprägung des offenen Systems zunächst notwendig, die interne Verarbeitung der Eingaben zu konkretisieren. Ein Navigationssystem, welches in

der Lage ist, alle benutzerspezifischen Ziele zu speichern und gleichzeitig z.B. die Route in Abhängigkeit von den Verkehrsmeldungen zu aktualisieren, ist hinsichtlich der Umgebung kontextsensitiv und bzgl. der benutzerspezifischen Ziele benutzeradaptiv. Wird die Funktionalität des Navigationssystems dahingehend erweitert, dass während der Berechnung der Route auf die Erfahrungswerte früherer Fahrten zum gleichen Ziel zurückgegriffen werden kann, so verhält sich dieser Teil des Systems hinsichtlich der Umgebung nicht mehr kontextsensitiv, sondern *kontextadaptiv*. Ein kontextadaptives System wird - angelehnt an die Definition 2.4 - dabei wie folgt definiert:

Definition 2.5 (Kontextadaptives System). Ein offenes System wird als kontextadaptiv bezeichnet, wenn es sein Verhalten auf der Grundlage eines Umgebungsmodells, welches sich aus der Historie aller Eingaben der Umgebung ableiten lässt, automatisch an die individuelle Umgebung anpasst.

Das Umgebungsmodell entspricht in analoger Weise dem Benutzermodell aus Definition 2.4[2]. Die beiden Funktionalitäten, die in dem oben genannten Beispiel des Navigationssystems die Eigenschaften der Benutzer- und Kontextadaptivität aufweisen, stehen jedoch in keinem Verhältnis oder in keiner Relation zueinander. Die beiden Funktionalitäten können also durch zwei voneinander weitestgehend unabhängige Komponenten des Systems realisiert werden, die jeweils benutzeradaptiv und kontextadaptiv sind. Ein beliebiges System kann somit, innerhalb verschiedener Komponenten, unterschiedliche Eigenschaften adaptiver und reaktiver Systeme aufweisen. Im Diagramm 2.1 werden daher noch einmal alle bis dato aufgeführten Systeme aufgezählt und zueinander in Beziehung gesetzt.

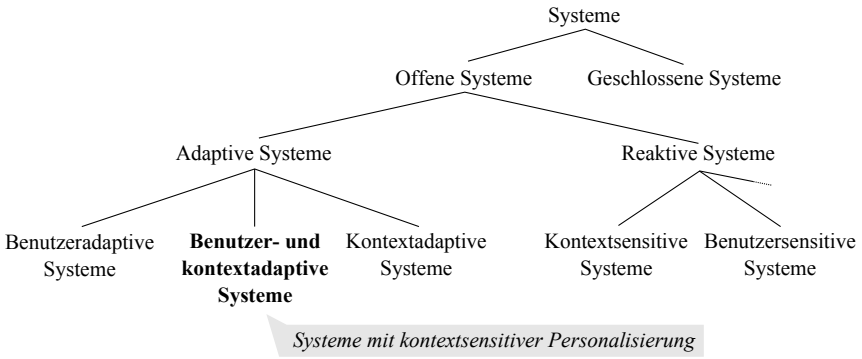


Abb. 2.1 Kategorisierung unterschiedlicher offener bzw. interaktiver Systeme nach der Ausprägung des Einflusses der Eingaben auf die Ausgaben.

Ist ein adaptives System jedoch in der Lage Informationen über den Benutzer und dessen Umgebung zu sammeln und in einem gemeinsamen Modell zu verarbeiten, um die Benutzerschnittstelle situationsabhängig anzupassen, so handelt es sich um ein *benutzer- und kontextadaptives System*. Die Definition ergibt sich dabei wie folgt:

Definition 2.6 (Benutzer- und kontextadaptives System). Ein offenes System wird gleichzeitig als benutzer- und kontextadaptiv bezeichnet, wenn es sein Verhalten auf der Grundlage eines Benutzermodells, welches sich aus der Historie aller Eingaben des Benutzers

[2] Das in Unterabschnitt 2.1 aufgeführte Beispiel eines adaptiven Fahrwerks gehört dabei ebenfalls zu den kontextadaptiven Systemen.

und der Umgebung ableiten lässt, automatisch an die individuellen situationsabhängigen Vorlieben des Benutzers ausrichtet.

Benutzer- und kontextadaptive Systeme unterstützen das Verfahren der *kontextsensitiven Personalisierung*, das in den nachfolgenden Abschnitten genauer untersucht wird. Während der kontextsensitiven Personalisierung werden benutzerspezifische Abhängigkeiten zwischen den Eingaben des Benutzers und den Eingaben aus der Umgebung ermittelt. Werden diese Korrelationen (nachfolgend als *kontextsensitive Präferenzen* bezeichnet) erkannt, so kann das System auf Basis dieser Informationen und der aktuellen Eingaben entscheiden, ob eine individuelle und kontextabhängige Anpassung der Benutzerschnittstelle in Hinblick auf eine Effizienzsteigerung der Bedienung sinnvoll wäre. Damit gehört jedes System, welches das in den nachfolgenden Kapiteln dieser Forschungsarbeit vorgestellte Verfahren zur situationsabhängigen Anpassung automotiver Benutzerschnittstellen unterstützt, zu der Gruppe der benutzer- und kontextadaptiven Systeme.

2.2 Personalisierung

Die Anpassungsfähigkeit der Benutzerschnittstellen i. Allg. kann laut Weld [Weld u. a., 2003], abhängig von der Art der Erfassung aller Benutzerinformationen, grundsätzlich zwei Formen annehmen. Während der Benutzer bei der sogenannten *Customization* alle relevanten Informationen über die Benutzerschnittstelle dem entsprechenden System entweder explizit oder durch eine Demonstration übergibt, werden die Informationen bei der *Adaptation* implizit während der Bedienung gesammelt und verarbeitet.

Die oftmals zur individuellen Gestaltung der Benutzerschnittstelle genutzte Form der Customization ist sehr populär und wird daher von einem Großteil aktueller Applikationen unterstützt. Bekannte Beispiele sind das Ändern der Farbschemata, Schriftgrößen, Hintergrundbilder oder das bedarfsgerechte Anpassen von Werkzeugleisten. Aber auch komplexe Anpassungen, wie beispielsweise das durch eine Demonstration vereinfachte Erstellen eines Makros zur sequentiellen Ausführung einer bestimmten Gruppe von Funktionalitäten, können manuell durchgeführt werden. Die Anpassungen werden also auf expliziten Wunsch des Benutzers ausgeführt. Es wird somit in jedem Fall davon ausgegangen, dass es sich bei den vom System gesammelten Informationen um valide Präferenzen des Benutzers handelt. Die dazugehörige Benutzerschnittstelle wird als *anpassbare Benutzerschnittstelle (engl. adaptable user interface)* bezeichnet.

Im Gegensatz zur Customization findet bei der Adaptation die Erfassung aller Benutzerinformationen nicht direkt über die manuelle Eingabe durch den Benutzer, sondern durch die indirekte Erfassung jeder relevanten Interaktion statt. Das System beobachtet dabei also jede Interaktion mit dem Benutzer und gegebenenfalls mit der Umgebung und entscheidet eigenhändig, also ohne explizites Eingreifen des Benutzers, über etwaige Anpassungen des Systemverhaltens. Liegen die relevanten Information vor, die auf eine mögliche Anpassung schließen lassen, so werden diese Informationen im Fall der *kontextfreien Personalisierung* instantan verwendet, um die Benutzerschnittstelle dynamisch an den Benutzer anzupassen. Während der Anpassung der Benutzerschnittstelle wird dabei angenommen, dass die zugrunde liegenden Informationen die realen Präferenzen des Benutzers hinreichend genau approximieren. Der für die Form der Adaption notwendige Prozess der Datenverarbeitung innerhalb des benutzeradaptiven Systems wird von Coutand in [Coutand, 2008] wie folgt definiert:

Definition 2.7 (Personalisierung[3] [Coutand, 2008]). Personalization is the task of providing adapted capabilities to the users of a system, on the basis of implicitly gathered information.

Wird außerdem noch die Umgebung bei der Erfassung der Präferenzen berücksichtigt, so wird allgemein von der bereits oben erwähnten kontextsensitiven Personalisierung gesprochen. Im Gegensatz zur kontextfreien Personalisierung wird die Benutzerschnittstelle hier jedoch nicht unmittelbar nach der Erfassung der Präferenzen angepasst, sondern immer nur in bestimmten Umgebungen bzw. Kontexten modifiziert. Unterstützt ein System den Prozess der Personalisierung i. Allg., so bezeichnet Langley in [Langley, 1997] die dazugehörige Benutzerschnittstelle als *adaptive Benutzerschnittstelle (engl. adaptive user interface)*. Das für die entsprechende Informationsverarbeitung zuständige Verfahren der Personalisierung erfolgt immer in zwei aufeinander folgenden Schritten: *Benutzermodellierung* und Anwendung der Anpassung [Jameson, 2008]. Während in der Benutzermodellierung die Erfassung und Verwaltung der Benutzer- und Kontextinformationen stattfindet, wird durch die Anwendung der Anpassung die Benutzerschnittstelle modifiziert. Die nachfolgenden Unterabschnitte widmen sich jeweils der Untersuchung eines der beiden notwendigen Schritte der Personalisierung.

2.2.1 Benutzermodellierung

Im Abschnitt 2.1 wurde bereits erwähnt, dass die Kommunikation zwischen dem System und den Subjekten auf dem Austausch von Ereignissen basiert. Jedes Ereignis enthält dabei alle Informationen, die den zugrunde liegenden Vorfall vollständig charakterisieren. In einigen Fällen lässt sich jedoch ein komplexer Vorfall erst durch eine kombinierte Betrachtung mehrerer zunächst unabhängiger Ereignisse eindeutig bestimmen. Komplexe Vorfälle dieser Art werden während der Kommunikation durch das System abstrahiert, zu Ereignissen verarbeitet und gegebenenfalls bei der Abstraktion weiterer komplexer Vorfälle berücksichtigt. Treten in der Eingabe z.B. die Ereignisse für die Vorfälle „Drücken eines Schalters“ und „Loslassen eines Schalters“ in der entsprechenden Reihenfolge auf, so kann das System auf die Betätigung eines Schalters schließen und daraus ein neues Ereignis erzeugen. Sind nun folgende Teilmengen gegeben: Ereignisse, die vom Benutzer $\check{E}_{user}^{level} \subset E$ und die vom System in der Benutzerschnittstelle $\hat{E}_{user}^{level} \subset E$ auf einer beliebigen Abstraktionsebene *level* verursacht werden, dann enthalten folgende Mengen

$$\check{I}_{user}^{level} = \{(e_1,...,e_n) \mid e \in \check{E}_{user}^{level} \wedge (n > 0) \wedge t_{event}(e_{i-1}) < t_{event}(e_i)\}$$
$$\hat{I}_{user}^{level} = \{(e_1,...,e_n) \mid e \in \hat{E}_{user}^{level} \wedge (n > 0) \wedge t_{event}(e_{i-1}) < t_{event}(e_i)\}$$

alle potentiellen Ein- und Ausgaben des Systems bzgl. des Benutzers, die sich durch beliebige Kombination aller möglichen Ereignisse der betreffenden Abstraktionsebene ergeben. Alle Ein- und Ausgaben bzgl. der Umgebung werden in analoger Weise definiert. Während die Ereignisse der Eingabe aus Sicht des Systems im Laufe der Verarbeitung zu Informationen abstrahiert werden, die ihrerseits ebenfalls als Ereignisse dargestellt werden können, findet in der Ausgabe eine Konkretisierung, also eine Abbildung der Informationen auf

[3] Der Begriff Personalisierung wird im weiteren Verlauf der Forschungsarbeit sowohl für den oben beschriebenen Prozess der Adaptation als auch für die aus Sicht des Benutzers erfassbare Änderung des Verhaltens verwendet.

Ereignisse statt. Die auf einer bestimmten Abstraktionsebene auftretenden Ereignisse ergeben sich im Fall der Abstraktion nicht nur aus den Kombinationen der Ereignisse der nächst niedrigeren Abstraktionsebene, sondern aus den Ereignissen aller niedrigeren Abstraktionsebenen. In vielen gängigen kontextadaptiven Systemen findet die Abstraktion der Eingaben aus der Umgebung größtenteils innerhalb einer *kontextabhängigen Middleware* statt, in der alle von Sensoren zur Verfügung gestellten Ereignisse applikationsübergreifend gesammelt und auf zu abstrahierende Eingaben untersucht werden. Die Konkretisierung findet im Gegensatz zur Abstraktion oftmals innerhalb der betreffenden Applikation statt. Wird bei der Konkretisierung beispielsweise der Fahrer vom System vor einer unmittelbaren Gefahr mit Hilfe eines blinkenden Symbols auf dem Tachometer gewarnt, so wird das Ereignis der Warnung innerhalb einer Applikation zur Visualisierung von Störmeldungen auf die beiden sich abwechselnden Ereignisse für das Ein- und Ausblenden des Symbols abgebildet. Die Implementierung der Applikation bestimmt dabei, nach welchen Vorschriften und Regeln bei der Konkretisierung vorgegangen wird und in welcher Weise die applikationsspezifische Benutzerschnittstelle modifiziert wird. Dies gilt analog für die Abstraktion der Ereignisse des Benutzers, die größtenteils in der applikationsspezifischen Benutzerschnittstelle verarbeitet werden. Das in dieser Forschungsarbeit vorgestellte und applikationsunabhängige Verfahren zur Verarbeitung der Sensordaten enthält ebenfalls einen Verarbeitungsschritt, bei dem mehrere nacheinander auftretende Ereignisse in der Eingabe, unabhängig von ihrer Abstraktionsebene, zu höherwertigen Informationen bzw. Ereignissen verarbeitet werden. Das Verfahren unterscheidet dabei allerdings nicht zwischen den Ereignissen aus der Umgebung und den Ereignissen vom Benutzer.

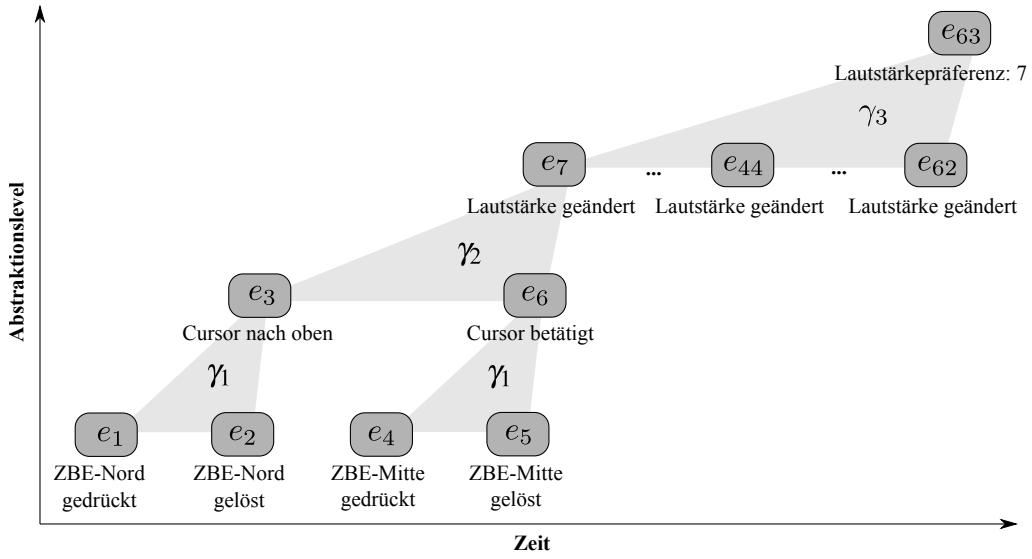


Abb. 2.2 Beispielhafte Menge von Ereignissen in der Eingabe, die bei der Änderung der Lautstärke mit der ZBE durch den Benutzer auf der untersten Ebene verursacht und vom System auf verschiedenen Abstraktionsebenen gefolgert werden können.

Abbildung 2.2 zeigt am Beispiel der Lautstärkereglung mit der ZBE die potentiell zu abstrahierenden Ereignisse auf verschiedenen Abstraktionsebenen. Die Abstraktionschritte γ_1 und γ_2 werden in diesem Fall in der applikationsspezifischen Benutzerschnittstelle und der Applikation ausgeführt, während der Abstraktionsschritt γ_3 durch den Prozess

der Benutzermodellierung implementiert wird. Letzterer Verarbeitungsschritt findet derzeit hauptsächlich ebenfalls innerhalb der Applikationen statt.

Alle Informationen über den Benutzer, die sich aus der Analyse mehrerer aufgezeichneter Sequenzen von Ereignissen aller Abstraktionsebenen ergeben, werden gemeinsam im Benutzermodell gespeichert. Wie oben beschrieben, entscheidet das System auf Basis dieser Informationen, welche Modifikationen in der Benutzerschnittstelle auszuführen sind. Kobsa fasst daher den Begriff des Benutzermodells wie folgt zusammen:

Definition 2.8 (Benutzermodell [Kobsa, 1995]). User models are collections of information and assumptions about individual users (as well as user groups) which are needed in the adaptation process.

Ein Benutzermodell kann neben den Informationen eines Benutzers also auch die Informationen einer Gruppe von Benutzern enthalten. Werden in einem adaptiven System beispielsweise Informationen über mehrere Benutzer aggregiert und findet die Adaption dann auf Basis der aggregierten Informationen statt, so bedient sich der Prozess der Benutzermodellierung der Methode des *Collaborative Filtering*. Diese Methode wird oftmals in Anwendungsgebieten genutzt, deren Systeme sich auch dann personalisieren sollen, wenn nur wenige Präferenzen des neuen Benutzers vorliegen. Beispielsweise ließe sich das Autoradio an einem bestimmten Ort, den der Benutzer noch nie betreten hat, automatisch auf einen speziellen Radiosender einstellen, der vorzugsweise von anderen Fahrern, deren Präferenzen mit den bekannten Präferenzen des Benutzers übereinstimmen, in dieser Ortschaft gehört wird. Wird jedoch ein Benutzer unabhängig von anderen Benutzern modelliert, so müssen in einer Lernphase zunächst relevante Interaktionen gesammelt werden, bevor es zu einer automatischen Anpassung kommt. Auch eine Kombination beider Methoden in einem Benutzermodell ist möglich.

Da der Benutzer auf der Basis eines facettenreichen Spektrums an Erfahrungen, Vorlieben und Emotionen handelt, das mit einem computergestützten System bis heute nur zum Teil oder überhaupt nicht erfassbar ist, ist es nicht möglich, die Vollständigkeit der im Benutzermodell gesammelten Informationen zu gewährleisten. Das Benutzermodell beschränkt sich ausschließlich auf approximative Abbildungen der realen Präferenzen, Ziele oder Absichten. Es werden also nur Verhaltensmuster gespeichert, die mit Hilfe von Sensoren auch erfassbar sind. Es ist aufgrund der beispielsweise im Fahrzeug zur Verfügung stehenden Sensoren möglich, die individuelle Wahl eines Musikgenres im Autoradio zusammen mit der damit typischerweise einhergehenden Tageszeit und dem damit verbundenen Ort im Benutzermodell zu speichern. Mögliche Gründe für einen solchen Schritt des Benutzers bleiben dem System jedoch verborgen. Letztendlich werden im Benutzermodell also alle Informationen zusammengefasst, die der Benutzer dem System durch seine Bedienung implizit zuführt. Diese Informationen können im Benutzermodell durch Mengen von Ereignissen repräsentiert werden oder die im nächsten Unterabschnitt 2.2.1.1 genannten Formen annehmen.

2.2.1.1 Informationen des Benutzers

Alle Informationen eines Benutzers werden im Nachfolgenden zusammenfassend als das *Profil* eines Benutzers bezeichnet. Zu den im Profil enthaltenen Informationen zählen dabei die Interessen, Ziele, Absichten, speziellen Merkmale und Eigenschaften, aber auch kognitive Modelle des Benutzers. Innerhalb des Benutzermodells kann das Profil grundsätzlich

auf zwei verschiedene Arten repräsentiert werden: *stereotyp* oder *individuell*. Wird ein stereotypisches Profil angelegt, wie dies oft beim Collaborative Filtering der Fall ist, so wird jeder Benutzer einer bestimmten Gruppe von Benutzern zugeordnet, deren Informationen denen des Benutzers ähneln. Stereotypische Benutzermodelle enthalten also ausschließlich eine Menge von Stereotypen und eine Funktion, die jeden Benutzer auf einen Stereotyp abbildet. Im Gegensatz zu den stereotypischen Profilen werden beim individuellen Profil alle Informationen separat für jeden Benutzer einzeln verarbeitet und abgelegt. Da sich in der kontextsensitiven Personalisierung die zum Teil sehr komplexen Interaktionen verschiedener Benutzer in den gleichen Kontexten stark unterscheiden können, ist es bei dieser Art der Personalisierung von Vorteil ein individuelles Profil zu erstellen. Daher wird in dem weiter unten vorgestellten Verfahren der Personalisierung automotiver Benutzerschnittstellen jeder Benutzer individuell verwaltet. Welche spezifischen Informationen im Benutzermodell gesammelt werden, hängt dabei vom jeweiligen Anwendungsfall ab. Eine nach persönlichen Vorlieben sortierte Liste von Zielen für das Navigationssystem erfordert beispielsweise andere Informationen, als eine Benutzerschnittstelle, die die Geschwindigkeit der Sprachdialoge an die Fehlerquote der Bedienung anpasst. Im einen Fall ist es nötig, alle bisher angefahrenen Ziele im Benutzermodell zu sammeln, während im anderen Fall alle Abbrüche einer Interaktion über die Zeit hinweg aufsummiert werden müssen.

Blues	**Funk**	**Indie**	**Klassik**	**Punk**
0,75	0,15	0,05	0,85	0,85

Politik	**Wirtschaft**	**Börse**	**Sport**	**Technik**
0,65	0,45	0,75	0,20	0,90

Tabelle 2.1 Beispielhafte Schlüsselwertpaare für das Ressort im Nachrichtenportal und das Musikgenre im Autoradio. Je höher der Wert eines Schlüsselwertpaares, desto eher handelt es sich bei dem betreffenden Schlüssel um eine Präferenz des Benutzers.

In den sogenannten *Recommender Systems* werden große Datenmengen daraufhin untersucht, ob möglicherweise Informationen enthalten sind, die für einen Benutzer relevant sein könnten. Dieser Prozess basiert auf Benutzerinformationen, die mit Hilfe der Techniken des *Data Minings* auf der Grundlage früherer Interaktionen ermittelt und zu individuellen Filtern verarbeitet wurden. Für diese Art von Anwendungen ist es üblich, die Informationen über den Benutzer in Form von *Schlüsselwertpaaren* im Benutzermodell zu speichern. Die Schlüssel beschreiben dabei beispielsweise ein bestimmtes Interessengebiet oder ein allgemeines Merkmal. Die Ausprägung eines Interessengebiets oder Merkmals wird dann durch die entsprechende Gewichtung, also den Wert des Schlüsselwertpaares, vorgegeben. Die Tabelle 2.1 zeigt beispielhaft zwei Listen von Schlüsselwertpaaren für die Merkmale des bevorzugten Nachrichtenressorts und des präferierten Musikgenres.

Werden z.B. mit Hilfe der Techniken des Data Minings kausale Beziehungen zwischen Ereignissen in der Bedienung gesucht, so ist es nötig, ähnliche Sequenzen von Interaktionen im Benutzermodell zusammenfassend zu speichern. Abbildung 2.3 zeigt eine beispielhafte Interaktion, die von einem Benutzer täglich ausgeführt wird, um die Verkehrssituation auf dem Weg ins Büro zu überprüfen. Findet eine solche Interkation wiederholt statt, so handelt es sich dabei um ein *sequentielles Interaktionsmuster*. Ein sequentielles Interaktionsmuster wird allgemein wie folgt definiert:

Definition 2.9 (Sequentielles Interaktionsmuster). Ein sequentielles Interaktionsmuster beschreibt die gemeinsamen Merkmale ähnlicher Ereignissequenzen, die in einem definierten Zeitintervall mit einer bestimmten Häufigkeit in der Eingabe auftreten.

Treten ähnliche Eingabesequenzen an verschiedenen Tagen wiederholt zur gleichen Zeit oder im gleichen Zeitraum auf, so wird die dem sequentiellen Interaktionsmuster zugrunde liegende Interaktion als *tägliche Routine* bezeichnet. Wird eine Eingabesequenz wiederholt an einem bestimmten Ort ausgeführt, so wird analog dazu der Begriff *örtliche Routine* verwendet. Eine *Routine* bezeichnet also auf abstrakter Ebene die wiederholte Ausführung einer oder mehrerer Aktionen des Benutzers, deren Struktur sich durch sequentielle Interaktionsmuster formal erfassen lassen. Im Benutzermodell werden dafür oftmals *prädiktive statistische Modelle* [Zukerman u. Albrecht, 2001] angelegt, die es dem System erlauben, das Verhalten eines Benutzers in einer kompakten Form und unter Beachtung der zufallsbedingten Anteile des Verhaltens abzuspeichern. Auf Basis dieser Modelle können nicht nur Interaktionen einfacher vorhergesagt, sondern auch bestimmte Aufgaben und Intentionen gefolgert bzw. abgeschätzt werden.



Abb. 2.3 Ablauf einer Interaktion zwischen dem Benutzer und dem Navigationssystem.

Markov-Modelle erweisen sich dabei als besonders geeignet für die Modellierung sequentieller Interaktionsmuster. Das Verhalten eines Benutzers wird innerhalb eines solchen Modells durch einen Graphen spezifiziert, der aus einer Menge von Zuständen und Transitionen besteht und einen stochastischen Prozess beschreibt. Während einzelne Interaktionsschritte jeweils durch Zustände modelliert werden, dienen Transitionen dazu, den Übergang von einem Zustand zum nächsten zu modellieren. Ein Zustandswechsel findet dabei nur mit einer bestimmten Wahrscheinlichkeit statt, die sich aus der Analyse beispielhaften Verhaltens ergibt. Der Nachteil dieser Methode liegt in der Tatsache, dass die Wahrscheinlichkeit eines Zustandswechsels nur davon abhängt, welcher Zustand letztmalig eintrat. Jede mögliche Vorhersage über den nächsten Interaktionsschritt ergibt sich also nur anhand des aktuellen Zustands und dessen ausgehende Transitionen, ohne eine Berücksichtigung der zuvor besuchten Zustände. Die Einführung von *Markov-Modellen höherer Ordnung* führte zu einer Lockerung der Markov-Eigenschaft, da Zustandswechsel nun mit bedingten Wahrscheinlichkeiten versehen werden konnten. Die Entscheidung über den nächsten Interaktionsschritt wird hier jedoch durch eine feste Anzahl vorhergehender Interaktionsschritte beeinflusst. Oft werden Markov-Modelle zunächst strukturell modelliert und danach durch konkrete Sequenzen von Ereignissen präzisiert. Die Wahrscheinlichkeitsverteilungen der Transitionen werden also sukzessive an die realen Verteilungen approximiert. Neben den Markov-Modellen höherer Ordnung [Armentano u. Amandi, 2009] werden aber auch Erweiterungen wie *relationale Markov-Modelle* [Anderson u. a., 2002] oder *Hidden Markov-Modelle* [Duong u. a., 2006; Nguyen u. a., 2005] im Benutzermodell verwendet.

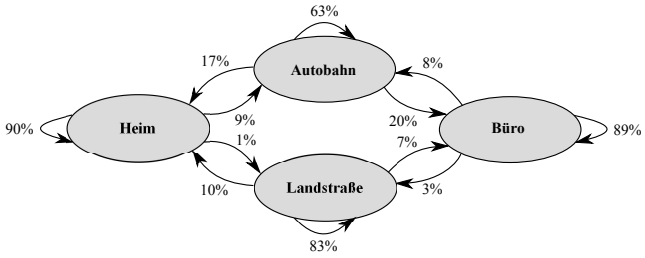


Abb. 2.4 Beispielhafte Visualisierung eines Markov-Modells, welches das Fahrverhalten eines Berufspendlers abbildet.

Ein großer Teil der Forschungsarbeiten zum Thema sequentieller Interaktionsmuster in der Benutzermodellierung stellt jedoch individuelle Formalismen zur Modellierung wiederkehrender Verhaltensmuster vor. Die Mächtigkeit der Formalismen und die Strukturen der jeweiligen Modelle orientieren sich dabei speziell an den dafür vorgesehenen Domänen. Beispielsweise unterscheiden sich sequentielle Interaktionsmuster automotiver Benutzerschnittstellen durch den hochgradig dynamischen Kontext von sequentiellen Interaktionsmustern, die während der Bedienung des statischen Heimcomputers registriert werden. Eine Wiederverwendung der Modelle sequentieller Interaktionsmuster in unterschiedlichen adaptiven Systemen ist im Unterschied zu den auf Schlüsselwertpaaren basierenden Modellen bis dato ohne zusätzliches Wissen nur in den wenigsten Fällen möglich.

2.2.1.2 Extraktion der Informationen durch Maschinelles Lernen

Oftmals lässt sich aus der Struktur der Informationen im Benutzermodell ableiten, welche Techniken des maschinellen Lernens sich für die Benutzermodellierung eignen. Im Nachfolgenden werden die sowohl in benutzeradaptiven Systemen als auch in benutzer- und kontextadaptiven Systemen gängigsten Techniken des maschinellen Lernens zusammenfassend vorgestellt.

- **Künstliche neuronale Netze.** Die Aufgabe eines künstlichen neuronalen Netzes (KNN) besteht darin, Eigenschaften einer bestimmten Datenmenge durch ein Netz von Verarbeitungseinheiten, die den menschlichen Neuronen nachempfunden wurden, zu repräsentieren. Dabei zeichnet sich jede Verarbeitungseinheit durch eine Funktion aus, die eine Menge von gewichteten Eingabeparametern zu einem Ausgabedatum verarbeitet und dieses einer weiteren Verarbeitungseinheit zur Verfügung stellt. Wird einem KNN beispielsweise eine Folge von Ereignissen zugeführt, die bestimmte Funktionsaufrufe in der Benutzerschnittstelle zusammen mit zu diesen Zeitpunkten gültigen Umgebungen kennzeichnen, so werden die Gewichtungen der Eingabeparameter so weit angepasst, dass die spezifischen Interaktionen nur anhand der Umgebungsereignisse berechnet werden können. Ein KNN approximiert also die Rechenvorschrift bzw. Kausalitäten einer beliebigen Datenmenge. Die Rechenvorschrift, auf deren Grundlage eine bestimmte Entscheidung anlässlich z.B. eines speziellen Umgebungsereignisses getroffen wird, lässt sich jedoch nur schwer nachvollziehen. Der Einsatz von KNNs zur Verhaltensanalyse ist daher nur in Anwendungsgebieten sinnvoll, in denen keine für den Benutzer verständliche Begründung für eine kontextsensitive Anpassung der Benutzerschnittstelle benötigt wird.

- **Klassifikation und Clustering.** Während der Klassifikation werden Objekte auf der Grundlage verschiedener Kriterien in bestimmte Gruppen eingeteilt. Die Kriterien werden dazu oft durch *Entscheidungsbäume* repräsentiert, die vor der eigentlichen Klassifikation anhand von Beispieldaten mit den dafür vorgesehenen Gruppen sukzessive aufgebaut und verfeinert werden. Häufig werden Kriterien aber auch durch *Fuzzy Regeln* beschrieben. Die Gruppen können in diesem Fall auch überlappend sein, da Fuzzy Regeln typischerweise unscharf formulierte Bedingungen enthalten, die die Subjektivität menschlicher Entscheidungen tendenziell besser approximieren. Allgemein können spezifische Interaktionen auf Basis der Klassifikation beispielsweise bestimmten Gruppen von Umgebungen zugeordnet werden. Im Gegensatz zum *Clustering* muss bei diesem Verfahren jedoch vorher bekannt sein, welcher Gruppe ein Objekt aus einer Menge von Trainingsdaten zugeordnet wird. Im Clustering sind die Gruppen vorher nicht bekannt und werden auf Basis der Trainingsdaten automatisch extrahiert. Der Vorteil der Klassifikation und des Clusterings besteht in der Nachvollziehbarkeit der resultierenden Regeln, die automatisch aus den Entscheidungsbäumen generiert werden können. Im Unterabschnitt 2.3.3 wird das Clustering unter anderem aufgrund dieser im automotiven Bereich notwendigen Bedingung detaillierter beschrieben und in dem hier vorgestellten Verfahren der kontextsensitiven Personalisierung verwendet.

- **Sequential Pattern Mining und Pattern Matching.** Das Auffinden von häufigen oder durch bestimmte Kriterien als relevant definierten Sequenzen von Objekten innerhalb großer und geordneter Datenmengen ist das Aufgabengebiet des *Sequential Pattern Mining*. Dieses Wissen kann beispielsweise, wie bereits im letzten Unterabschnitt angedeutet, dazu genutzt werden, den Benutzer an bestimmten Stellen im Dialog, in denen es häufig zu einer fehlerhaften Bedienung kommt, proaktiv auf alternative Möglichkeiten, von denen der Benutzer ohnehin regelmäßig nach der Fehlbedienung Gebrauch macht, hinzuweisen. Im Gegensatz zum sogenannten *Pattern Matching*, bei dem bekannte Muster in den Daten gesucht werden, findet ein Algorithmus des Sequential Pattern Mining bisher unbekannte Muster. Das Pattern Matching eignet sich also insbesondere zum Einsatz in Domänen, in denen der Ergebnisbereich durch bekannte Muster eingeschränkt werden kann. Eine mögliche Anwendung des Pattern Matching im automotiven Bereich wird im Unterabschnitt 2.3.2 diskutiert.

- **Reinforcement Learning**. Beim Reinforcement Learning handelt es sich um eine Methode des maschinellen Lernens, die schon bestehende Modelle oder Regeln durch etwaiges Feedback modifiziert bzw. spezialisiert. Die Modelle und Regeln werden dabei durch Bestätigung gefestigt und durch negatives Feedback verändert. Da genauso wie beim Pattern Matching bereits vor dem Beginn des Lernprozesses ein Modell oder eine Regelsammlung vorhanden sein muss, eignet sich das Reinforcement Learning daher eher für die Weiterentwicklung oder Präzisierung von sequentiellen Interaktionsmustern, die durch andere Techniken des maschinellen Lernens ermittelt wurden. Ergab sich durch eine Klassifikation oder ein Clustering beispielsweise der Zusammenhang zwischen einer spezifischen Interaktion und einer Tageszeit, so wäre es durch Reinforcement Learning möglich, diese kontextsensitive Präferenz - durch einen Verzicht auf die Interaktion an Wochenenden - ausschließlich auf Arbeitstage zu präzisieren.

- **Fallbasiertes Lernen.** Beim fallbasierten Lernen werden neue Probleme mit historischen Problemen verglichen, um Lösungen der historischen Problemstellungen auf aktuelle

Problemstellungen zu übertragen. Für eine Ähnlichkeitsanalyse der Probleme ist es jedoch nötig, Vergleichsfunktionen zu entwickeln, die eine Quantifizierung der Ähnlichkeit ermöglichen. Diese sogenannten *Distanzfunktionen* können jedoch nur dann geeignet formuliert werden, wenn die Probleme in ihrer Struktur bereits bekannt sind. Dies erweist sich insbesondere dann als schwierig, wenn Probleme aus vielen einzelnen Faktoren bestehen, deren Relevanz sich von Problemklasse zu Problemklasse unterscheidet. Fallbasiertes Lernen wird in adaptiven Benutzerschnittstellen verwendet, um implizit erfasste Interaktionen des Benutzers mit den historischen Interaktionen zu vergleichen und bei Übereinstimmung die Benutzerschnittstelle, falls nötig, zu adaptieren. Der Vorteil dieser Methode liegt in der kurzen Lernphase, die es bereits ermöglicht, nach nur einem erfassten Fall zu personalisieren. Eine detaillierte Untersuchung dieser Methode und deren Anwendung zur kontextsensitiven Personalisierung erfolgt im Unterabschnitt 3.1.2.

Nur in den seltensten Fällen wird ausschließlich eine Technik bei der Personalisierung angewendet. Häufig werden verschiedene Techniken kombiniert, um z.B. mit Hilfe des Sequential Pattern Mining ein Verhaltensmodell zu generieren und dieses durch das Reinforcement Learning zu präzisieren. Im Abschnitt 2.3 werden die in dem hier vorgestellten Verfahren der kontextsensitiven Personalisierung angewandten Techniken des maschinellen Lernens detaillierter vorgestellt.

2.2.2 Anpassung der Benutzerschnittstelle

Während des zweiten Schrittes der Personalisierung wird auf Grundlage der im Benutzermodell gesammelten Informationen entschieden, welche Modifikationen in der Ausgabe bzw. Benutzerschnittstelle auszuführen sind. Hierbei wird generell zwischen Modifikationen unterschieden, die entweder direkt nach der Erfassung der Informationen angewendet werden und bestehen bleiben, bis neue Informationen eine weitere Modifikation bedingen, oder kontextsensitiven Modifikationen, die nur unter bestimmten Bedingungen gültig sind und daher auch nur unter diesen Bedingungen ausgeführt werden dürfen. Werden beispielsweise empfangene E-Mails in einem Verwaltungsprogramm für Nachrichten nach ihrer Wichtigkeit priorisiert dargestellt, so wird der durch Benutzermodellierung erfasste individuelle Filter unmittelbar im Verwaltungsprogramm angewendet. Der Filter wird jedoch erst dann ausgetauscht, wenn die Benutzermodellierung einen aktualisierten Filter bereitstellt. Damit ist also zu jeder Zeit nur maximal ein individueller Filter vorhanden. Wird der Filter jedoch in Beziehung zum Kontext gesetzt, so kann das System auf Grundlage verschiedener kontextsensitiver Filter entscheiden, dass beispielsweise private Nachrichten höher priorisiert dargestellt werden, wenn der Benutzer das Verwaltungsprogramm zu Hause nutzt, und umgekehrt niedriger, wenn sich der Benutzer am Arbeitsplatz befindet.

Anpassungen der Benutzerschnittstelle werden auch dahingehend differenziert, ob sie generell durch Filter die Informationsbeschaffung beeinflussen oder den Benutzer lediglich bei der Bedienung unterstützen [Jameson, 2008]. Die Anpassungen der bereits oben erwähnten Recommender Systeme werden der ersten Kategorie zugeordnet, da neue Informationen klassifiziert und gefiltert werden müssen. Die in dieser Forschungsarbeit vorgestellten kontextsensitiven Personalisierungen modifizieren die Benutzerschnittstelle jedoch ausschließlich mit der Absicht, die Bedienung zu vereinfachen und nicht den Informations-

fluss zu verändern. Jameson [Jameson, 2008] unterscheidet innerhalb der zweiten Kategorie fünf verschiedene Formen der Anpassung:

- **Automatisierte Routinen.** Einige Funktionalitäten erfordern zu ihrer Ausführung eine komplexe oder aufwendige Interaktion zwischen Benutzer und System. Dazu zählt beispielsweise das manuelle Sortieren von E-Mails in bestimmte Ordner oder die Eingabe einer Zieladresse im Navigationssystem. Werden die Eingaben der Interaktionen mit den dazugehörigen Parametern im Benutzermodell gespeichert, so kann der Prozess der Sortierung automatisch ausgeführt oder bestimmte Felder bei der Eingabe des Ziels vorbelegt werden. Die Routinen werden also automatisch ausgeführt.

- **Änderung der Struktur.** Benutzerschnittstellen können oft auch nur in subtiler Weise angepasst bzw. verändert werden. Die präferierte Funktionalität wird also nicht automatisch ausgeführt, sondern der Aufwand bei der manuellen Ausführung dieser Funktionalität reduziert. Dies kann beispielsweise bedeuten, dass die für eine Funktionalität notwendigen Schritte der Interaktion auf ein Minimum reduziert werden. Dies geschieht durch das Umsortieren von Schaltflächen oder durch das Verändern der Gestalt von Schaltflächen. Das Umgestalten der Benutzerschnittstelle, wie beispielsweise das Verändern der Schriftgröße oder der Größe der Schaltflächen, kann insbesondere in benutzeradaptiven Systemen, die hauptsächlich von Benutzern älterer Generationen genutzt werden, die Effizienz der Bedienung steigern.

- **Proaktive Hilfe.** Die Bedienung lässt sich jedoch auch durch unterstützende Dialoge oder Hinweise vereinfachen. Sucht der Benutzer eine bestimmte Funktionalität, so kann das System mit Hilfe von Hinweisen bzgl. der gewünschten Funktionalität diese leichter zugänglich machen. Für diese Art der proaktiven Hilfe muss das System jedoch in der Lage sein, die Ziele und Absichten des Benutzers auf der Basis bestimmter Interaktionen zuverlässig zu erkennen. Das bekannteste Beispiel einer solchen proaktiven Hilfe stellt der von der Firma Microsoft[4] in dem Textverarbeitungsprogramm Office™97[4] vorgestellte Büroassistent dar. Die Hinweise oder Vorschläge des Büroassistenten beziehen sich dabei immer auf Funktionen, die im Kontext der Bedienung nützlich sein könnten.

- **Vermittelnde Interaktion.** Eine spezielle Technik der Anpassung, die vor allem im automotiven Bereich sinnvoll eingesetzt werden kann, ist das Modifizieren in Abhängigkeit vom kognitiven oder emotionalen Zustand des Benutzers. In dieser Methode wird während der Bedienung ein kognitives oder emotionales Benutzermodell aufgebaut, welches anschließend festlegt, welche Modifikation in der gegenwärtigen Situation angebracht wäre. Ist der Fahrer beispielsweise gestresst oder befindet er sich in einer unüberschaubaren Verkehrssituation, so könnte die Signalisierung des Empfangs einer Nachricht zeitlich verzögert werden. Aber auch das automatische Einstellen der Lautstärke eines Mobiltelefons in Abhängigkeit vom aktuellen Stresslevel des Benutzers, welcher sich gegebenenfalls über ein Sensorarmband ermitteln lässt, stellt eine praktische Anwendung dieser Technik dar.

- **Kontrollierte Dialoge.** Oft werden in modernen adaptiven Systemen auch die Dialoge zwischen dem Benutzer und dem System auf Basis des Benutzermodells verändert. Diese überwiegend bei Sprachdialogen angewandte Technik ändert beispielsweise die

[4] http://www.microsoft.com

Dauer eines Dialogs in Abhängigkeit vom Grad der Erfahrung des Benutzers. Bestimmte für den Dialog hilfreiche Erklärungen des Systems in Bezug auf einzelne Interaktionsschritte können bei einem erfahrenen Benutzer inhaltlich zusammengefasst oder sogar ganz weggelassen werden. Auch die Geschwindigkeit der Sprachausgabe lässt sich in bestimmten Grenzen variieren.

Brusilovsky stellt in [Brusilovsky, 1996] eine hierarchische Kategorisierung verschiedenster Typen von Anpassungen vor, die speziell im Bereich der adaptiven Hypermedien [Brusilovsky, 2001] auftreten. Eine solche Kategorisierung lässt sich jedoch auch auf Benutzerschnittstellen übertragen, die nicht zu den Hypermedien gehören. Innerhalb dieses Forschungsgebiets wird zunächst grundsätzlich zwischen der adaptiven Präsentation von Informationen und der adaptiven Unterstützung der Navigation unterschieden. Anpassungen von multimedialen Inhalten oder Textfragmenten werden nach Brusilovsky der Kategorie der adaptiven Präsentationen zugeordnet, während benutzerspezifische Hilfen bei der Navigation, wie beispielsweise nützliche oder annotierte Links, zur zweiten Kategorie gezählt werden.

Einige Beispiele aus der Praxis lassen sich jedoch nicht eindeutig einer bestimmten Form der Anpassung zuordnen, da sich die verschiedenen Kategorien nicht exakt voneinander abgrenzen lassen. Um das bestmögliche Ergebnis für ein bestimmtes Anwendungsgebiet zu erzielen, werden wie auch bei den Techniken des maschinellen Lernens oftmals verschiedene Formen kombiniert. Die in dieser Arbeit eingeführten Formen der kontextsensitiven Anpassungen automotiver Benutzerschnittstellen werden im Unterabschnitt 2.3.4 vorgestellt und in die von Jameson aufgestellten Kategorien eingeordnet.

2.3 Personalisierung im automotiven Bereich

Dieser Abschnitt widmet sich zunächst der Formulierung spezieller Anforderungen, die eine adaptive Benutzerschnittstelle sowohl i. Allg. als auch im Bereich der automotiven Systeme aus Sicht der Anwender erfüllen muss. Darauf aufbauend lassen sich maschinelle Lerntechniken ableiten, die sich insbesondere für den Einsatz in In-Car-Infotainment Systemen eignen. Abschließend werden zwei Formen der kontextsensitiven Personalisierung automotiver Benutzerschnittstellen vorgestellt und durch vier unterschiedliche Anwendungsszenarien motiviert.

2.3.1 Anforderungen adaptiver und automotiver Benutzerschnittstellen

Jede Anpassung einer adaptiven oder automotiven Benutzerschnittstelle führt zwangsläufig zu einer Umstellung des mentalen Modells, das der Benutzer von der Benutzerschnittstelle besitzt. Übersteigt das Maß bestimmte Grenzen, so wird die Anpassung als störend empfunden. In welchem Maß sich dieser Prozess vollzieht, hängt von der Komplexität, Form, Umsetzung und von weiteren Aspekten der Anpassung ab. Folglich sollten bestimmte Kriterien formuliert werden, die sowohl eine Bewertung dieser adaptiven Eigenschaften einer Benutzerschnittstelle als auch die Vergleichbarkeit verschiedener Umsetzungen ermöglichen. Im Folgenden werden dazu zunächst allgemeine Kriterien formuliert und anschließend durch die speziellen Anforderungen aus dem automotiven Bereich ergänzt. Laut Gajos and Jame-

son beeinflussen folgende allgemeine Kriterien den Grad der Akzeptanz benutzeradaptiver Systeme [Gajos u. Jameson, 2009]:

- **Stabilität.** Die Stabilität ist ein Grad dafür, wie stark eine Benutzerschnittstelle durch eine Adaption verändert wird. Werden beispielsweise Schaltflächen in Abhängigkeit von der Häufigkeit ihrer Nutzung personalisiert, so kann dies durch Weglassen einzelner und Umordnen aller Schaltflächen oder durch einfaches Hervorheben oft benutzter Schaltflächen geschehen. Im ersten Fall ist eine geringe Stabilität des Layouts zu erwarten, während im letzten Fall die Stabilität aufgrund einer nur subtilen Änderung des Layouts höher ausfällt. Benutzer bevorzugen i. Allg. eine hohe Stabilität.

- **Lokalität.** Bei der Lokalität handelt es sich auch um eine das Layout betreffende Metrik der Personalisierung. Sie beschreibt die logische Distanz, die zwischen dem Bereich liegt, an dem die zur Personalisierung notwendigen Informationen gesammelt wurden, und dem Ort, an dem die eigentliche Modifikation der Benutzerschnittstelle ausgeführt wird. Gegeben sei wie im oberen Beispiel die Personalisierung von Schaltflächen in Abhängigkeit von der Häufigkeit der Nutzung. Werden die häufig genutzten Schaltflächen immerwährend und unabhängig von der zur Zeit genutzten Applikation in einem separaten Teil des Bildschirms dargestellt, so ist die logische Distanz zwischen den ursprünglichen Positionen der Schaltfläche und der Position im separaten Teil des Bildschirms gering. Findet die Modifizierung direkt an der Schaltfläche statt, so ist die Lokalität hoch. Die Benutzer bevorzugen i. Allg. ein hohes Maß an Lokalität.

- **Wahlfreiheit.** Außerdem wird unterschieden, ob der Benutzer gezwungen ist, etwaige Adaptionen zu nutzen, oder ob er sich entscheiden kann, welche der vom System vorgeschlagenen Veränderungen in seinem Kontext sinnvoll sind. Die zuvor beschriebenen Schaltflächen im separaten Teil des Bildschirms stellen im Sinne der Wahlfreiheit beispielsweise nur eine alternative Möglichkeit der Bedienung dar. Die ursprünglichen Schaltflächen bleiben davon unangetastet. Werden die Schaltflächen umsortiert, so ist der Benutzer gezwungen, sein mentales Modell des Layouts laufend anzupassen. Es ist insbesondere für unerfahrene Benutzer daher oft geeigneter, die Wahlfreiheit durch alternative Bedienmöglichkeiten zu gewährleisten.

- **Genauigkeit.** Nicht jede automatisch durchgeführte Adaption, welche durch die oben erwähnten Techniken und Methoden ermittelt wurde, führt zu dem vom Benutzer gewünschten Ergebnis. Beispielsweise kann in einigen Situationen eine Adaption entweder unpassend oder sogar unerwünscht sein. Werden bestimmte Schaltflächen ausschließlich in einer bestimmten Situation benutzt, dann ist das visuelle Hervorheben dieser Elemente in einer anderen Situation stets irreführend. Wie jedoch bereits in 2.2.1 beschrieben, kann ein benutzeradaptives System oder ein benutzer- und kontextadaptives System die realen Präferenzen nur approximieren. Die Genauigkeit beschreibt dabei das Präzisionsmaß der Vorhersagen. Wird eine hohe Präzision erzielt, so steigert dies die Effizienz der Benutzung und damit einhergehend die Zufriedenheit des Benutzers.

- **Vorhersehbarkeit.** Können Benutzer nachvollziehen, aus welchen Gründen das System eine bestimmte Adaption ausführt, so ist das Verhalten des benutzeradaptiven Systems aus Sicht des Benutzers vorhersehbar. Die Komplexität der vom benutzeradaptiven System berechneten Bedingungen für eine Adaption beeinflusst dabei maßgeblich den Grad

an Transparenz. Werden Benutzermodelle beispielsweise durch komplexe statistische Modelle abgebildet, so ist es nur begrenzt möglich, die Entscheidungen für eine Adaption vermittelbar zu gestalten. Wird eine Schaltfläche beispielsweise laut dem Benutzermodell erst dann betätigt, wenn zehn oder mehr Umweltfaktoren bestimmte Zustände annehmen, dann kann die Entscheidung zur Adaption, nämlich die automatische Ausführung der der Schaltfläche zugrunde liegenden Funktionalität, zwar im Sinne der Genauigkeit sehr präzise, aber durch das breite Spektrum an Einflüssen auch sehr kompliziert sein.

- **Effizienz.** Auch die durch eine Adaption möglicherweise gesteigerte allgemeine Effizienz der Bedienung spielt eine große Rolle bei der Bewertung adaptiver Benutzerschnittstellen. Meist wird gerade durch die Personalisierung versucht, den Benutzer entweder sanft zu unterstützen oder durch eine Reduzierung notwendiger Bedienschritte die Bedienung zu vereinfachen. Findet jedoch fälschlicherweise eine Adaption statt, die dann auch noch rückgängig gemacht werden muss, so beeinflusst dies die gesamte Effizienz der Bedienung. Eine höhere Effizienz der Bedienung wird vor allem im Bereich mobiler Geräte angestrebt, da die Benutzerschnittstellen meistens nur über kleine Bildschirme verfügen und sich die Bedienung oft in Abhängigkeit von der Umgebung automatisieren lässt.

- **Schnelligkeit.** Die Anwendung der in Unterabschnitt 2.2.1.2 erwähnten Techniken des maschinellen Lernens, um Benutzerschnittstellen zu personalisieren, führt zunächst immer zu einer Lernphase, in der die Benutzer stets ohne Feedback auskommen müssen. Die Adaptionen können also erst dann ausgeführt werden, wenn auf Basis dieser Verfahren das Verhalten des Benutzers im Modell ausreichend genau approximiert wurde. Je schneller sich das adaptive System an die individuellen Aspekte der Bedienung anpasst, desto eher wird die Personalisierung als positiv bzw. effizienzsteigernd empfunden. Ist der Benutzer erst einmal ein Experte der Bedienung, dann verringert sich insbesondere der Nutzen von Adaptionen, die beispielsweise eine geringe Stabilität aufweisen.

Einige der Kriterien müssen jedoch gegeneinander abgewogen werden, da oftmals die Verbesserung beispielsweise der Genauigkeit mit einer Verschlechterung der Vorhersagbarkeit einhergeht. Dabei sollte je nach Anwendungsfall entschieden werden, welchen Kriterien besondere Aufmerksamkeit geschenkt wird. Neben diesen allgemeinen Bewertungskriterien müssen jedoch auch die speziellen Merkmale der Bedienung adaptiver Benutzerschnittstellen im automotiven Bereich berücksichtigt werden:

- **Kognitiver Aufwand.** Während dem kognitiven Aufwand, der sich bei der Bedienung einer modifizierten Benutzerschnittstelle ergibt, bei der Personalisierung von mobilen Telefonen oder Heimcomputern wenig Beachtung geschenkt wird, spielt dieser im automotiven Umfeld eine entscheidende Rolle. Modifizierte Anteile der Benutzerschnittstelle sollten ohne großen kognitiven Aufwand bedienbar sein, damit der Fahrer stets der Fahrzeugsteuerung seine volle Aufmerksamkeit schenken kann. Das Umordnen von Schaltflächen in der automotiven Benutzerschnittstelle würde beispielsweise zwangsläufig den kognitiven Aufwand der Bedienung auf ein kritisches Maß erhöhen, da der Fahrer gezwungen wird, das veränderte Layout zu erkunden.

- **Grad der Ablenkung.** Wird eine automotive Benutzerschnittstelle automatisch modifiziert, so darf der Prozess der Anpassung selber, also die Durchführung der Änderung, den Fahrer niemals ablenken. Als besonders ablenkend gelten dabei beispielsweise aufwen-

dige Animationen der Modifizierung oder stark blinkende Hinweise auf Veränderungen. Das Hervorheben einer Schaltfläche sollte daher z.B. dezent durch eine einmalige Vergrößerung oder Wahl einer anderen Hintergrundfarbe erzielt werden, und nicht durch Animationen oder sich schnell wiederholende Farbänderungen in den Vordergrund treten.

- **Physischer Aufwand.** Während sich der physische Aufwand z.B. bei der Bedienung eines Heimcomputers aus dem Einsatz beider Hände ergibt, beschränkt sich dieser bei der Nutzung eines In-Car-Infotainment-Systems größtenteils nur auf die mit einer Hand zu nutzende ZBE. In einigen Fällen kann das In-Car-Infotainment-System auch mit Hilfe der auf beiden Seiten des Lenkrads angebrachten Tasten bedient werden. Etwaig modifizierte Benutzerschnittstellen sollten den zur Bedienung der ursprünglichen Version notwendigen physischen Aufwand nicht übersteigen.

- **Rechtliche Einschränkungen.** In Abschnitt 1.1 wurde bereits darauf hingewiesen, dass spezielle Ausprägungen der Benutzerschnittstelle in bestimmten Situationen aus rechtlichen Gründen nicht erlaubt sind. Beispielsweise darf ein In-Car-Infotainment-System während der Fahrt keine bewegten Bilder anzeigen. Die Personalisierung ist daher an zusätzliche Bedingungen gebunden, die dem System und der Benutzerschnittstelle bekannt sein müssen.

Um einen geringen kognitiven Aufwand bei der Bedienung kontextsensitiv personalisierter automotiver Benutzerschnittstellen zu gewährleisten, sollte eine Form der Modifikation gewählt werden, die eine hohe Stabilität gewährleistet. Da die Genauigkeit und Effizienz ebenfalls mit dem kognitiven Aufwand korrelieren, sind auch diese Kriterien priorisiert zu berücksichtigen. Werden beispielsweise unpassende Anpassungen ausgeführt, so entsteht durch die oftmals notwendige Korrektur ein zusätzlicher kognitiver Aufwand, der die Effizienz reduziert. Dienen die Personalisierungen der Effizienzsteigerung, beispielsweise durch eine Reduzierung der Bedienschritte, so führt dies umgekehrt zu einem geringeren kognitiven Aufwand. Um jedoch die vom System ermittelten Bedingungen einer kontextsensitiven Personalisierung so einfach und nachvollziehbar wie möglich zu gestalten, sollte die Genauigkeit tendenziell niedriger ausfallen und durch alternative Bedienmöglichkeiten oder zu bestätigende Anpassungen kompensiert werden. Zusammen mit einer Offenlegung der Gründe für eine Personalisierung ließe sich damit automatisch auch eine Steigerung der Vorhersehbarkeit bzw. Transparenz erzielen [Bauer, 2004]. Um das Benutzerverhalten möglichst schnell zu erfassen, sollte das System während der Benutzermodellierung auf Algorithmen des maschinellen Lernens zurückgreifen, die bereits auf der Basis weniger ähnlicher Interaktionen auf kontextsensitive Präferenzen schließen können. Eine zusätzliche individuelle Parametrisierbarkeit der Geschwindigkeit wäre außerdem wünschenswert.

2.3.2 Pattern Matching zur Erkennung relevanten Benutzerverhaltens

Wie bereits in Abschnitt 2.2.1.2 beschrieben, bezeichnet Sequential Pattern Mining den Prozess der Suche nach unbekannten sequentiellen Mustern innerhalb einer geordneten Menge beliebiger Objekte. Werden die Eingaben eines In-Car-Infotainment-Systems auf derartige Weise analysiert, so können typische sequentielle und situationsabhängige Bedienmuster

erkannt und gegebenenfalls automatisiert werden. Dadurch ließe sich beispielsweise erkennen, dass ein Benutzer immer dann die Lautstärke auf eine präferierte Höhe reguliert, wenn er zuvor das Radio auf einen bestimmten Sender eingestellt hat. Die Personalisierung könnte folglich darin bestehen, die Lautstärke nach jedem Radiosenderwechsel automatisch zu regulieren. Diese Art der Analyse wird aufgrund der hohen Komplexität des Sequential Pattern Mining meistens durch zusätzliche Bedingungen unterstützt, die beispielsweise eine maximale Länge der zu entdeckenden Muster vorschreiben. Damit kann nicht nur der Aufwand der Analyse reduziert, sondern auch der Ergebnisraum verkleinert werden. Ob die zur Suche verwendete Technik dem Gebiet des Sequential Pattern Mining oder dem Bereich des Pattern Matching zugeordnet werden kann, hängt dabei von der Komplexität der Bedingungen ab.

Oftmals werden nur sequentielle Muster gesucht, die in einer endlichen Menge von Sequenzen mit einer bestimmten Häufigkeit auftreten [Srikant u. Agrawal, 1996]. Da die Ergebnisse (einer unter dieser, das gesamte Muster betreffenden Bedingung ausgeführten Suche) mit Ausnahme ihrer Häufigkeit weitestgehend unbekannt sind, wird in diesem Fall vom Sequential Pattern Mining gesprochen. Im Bereich mobiler eingebetteter Systeme erweist sich die Anwendung des Sequential Pattern Mining jedoch aufgrund der hohen Diversität der Ergebnisse und der begrenzten Ressourcen als schwierig. Da jede Teilsequenz einer häufigen Sequenz ebenfalls häufig im Sinne der Bedingung ist, ergibt sich ein sehr großer Ergebnisraum. Bzgl. des Beispiels der Lautstärkereglung wären auch alle Sequenzen von Ereignissen häufig, die immer zwischen dem Radiosenderwechsel und der Lautstärkereglung auftreten. Dies können beispielsweise alle Bedienschritte sein, die ein Benutzer im Menü der Benutzerschnittstelle ausführen muss, um ausgehend vom Radio zur Laustärkereglung zu gelangen. Im Bereich der automotiven Anwendungsfälle haben die diversen Teilsequenzen jedoch oftmals keine Relevanz. Die Einführung einer minimalen Länge oder Dauer einer gesuchten häufigen Sequenz würde in diesem Fall bei der Analyse gegebenenfalls zu einem besseren Ergebnis führen. Letztere Bedingung wird dabei speziell bei der Analyse von Daten berücksichtigt, die aus zeitlich geordneten Elementen bestehen [Mannila u. a., 1997]. Das Sequential Pattern Mining wird aufgrund dieser Problematiken vornehmlich in Forschungsbereichen eingesetzt, in denen ein Experte - ausgehend von einem Ergebnisraum - die Selektion relevanter Sequenzen vornehmen kann. Der Einsatz des Sequential Pattern Mining innerhalb automotiver Systeme scheitert also nicht nur an den begrenzten Ressourcen eingebetteter Systeme, sondern vor allem an dem sehr individuellen Prozess der Selektion relevanter Muster.

Der Einsatz regulärer Ausdrücke zur Formulierung zu suchender sequentieller Muster [Garofalakis u. a., 1999] führt dann zur Technik des Pattern Matching. Die Kriterien, die ein relevantes Muster zu erfüllen hat, werden dabei deutlich präziser spezifiziert. Durch die regulären Ausdrucke werden nicht nur allgemeine Eigenschaften, sondern auch strukturelle Eigenheiten der zu suchenden sequentiellen Muster definiert. Im Bereich von CEP-Systemen (Complex Event Processing) werden kontinuierliche Sequenzen von Ereignissen zeitnah daraufhin untersucht, ob Teilsequenzen von Ereignissen einem vordefinierten sequentiellen Muster entsprechen. Die Besonderheit der CEP-Systeme besteht darin, dass erkannte Teilsequenzen wiederum zu Ereignissen verarbeitet und einem neuen Datenstrom als solche hinzugefügt werden können. Sogenannte *Event Pattern Languages* (EPL) dienen den CEP-Systemen zur formalen Beschreibung der zu suchenden sequentiellen Muster. Die EPLs bedienen sich dabei genauso wie die regulären Ausdrücke zahlreicher logischer Operatoren wie der Konjunktion, Disjunktion oder Negation zur Spezifikation komplexer Relationen zwischen Ereignissen bzw. Zeichenketten. Im Gegensatz zu den regulären Aus-

drücken können mit den EPLs jedoch auch zusätzliche Prädikate bzgl. der Eigenschaften von Ereignissen formuliert werden. Damit können mehrere Ereignisse des gleichen Typs auf Basis der durch die Prädikate beschriebenen Eigenschaften unterschieden werden. Außerdem können mit EPLs auch kausale oder temporale Korrelationen zwischen Ereignissen formuliert werden, die zwei Ereignisse in eine Ursache/Wirkung Beziehung setzen oder z.B. das Zeitintervall zwischen zwei Ereignissen auf einen maximalen Wert beschränken.

Die Suche nach sequentiellen Interaktionsmustern kann im Anwendungsgebiet automotiver Benutzerschnittstellen daher mit Hilfe der EPLs auf rechtlich erlaubte und für einen Anwendungsfall relevante Interaktionen beschränkt werden. Durch die Wahl einfacher und leicht vermittelbarer sequentieller Interaktionsmuster ist es außerdem möglich, eine hohe Nachvollziehbarkeit des darauf aufbauenden Systemverhaltens zu gewährleisten. Sogar die vollständige Offenlegung der sequentiellen Interaktionsmuster durch das System und eine anschließende Parametrisierbarkeit durch den Benutzer sind denkbar. Im Abschnitt 4.2 wird daher eine für die Beschreibung automotiver Interaktionen geeignete EPL vorgestellt und deren Anwendung zur Formulierung relevanter Interaktionen und Situationen exemplarisch beschrieben.

2.3.3 Clustering relevanten Benutzerverhaltens

Im automotiven Bereich dienen zahllose Sensoren dazu, ein für die diversen Steuergeräte zu verarbeitendes Abbild der aktuellen Situation zu schaffen. Dabei trägt jeder Sensor einen Teil des digitalen Abbildes zusammen. Es lässt sich also zu jedem Zeitpunkt der aktuelle Stand des gesamten digitalen Abbildes „einfrieren“ und abspeichern. Wurden nun durch das im vorherigen Abschnitt beschriebene Pattern Matching relevante Interaktionen gefunden, so gilt es die Interaktionen auf Basis der zu den entsprechenden Zeitpunkten gültigen Situationen, die sich aus einem oder mehreren digitalen Abbildern der Umgebung zusammensetzen, zu gruppieren. Interaktionen, die in einer ähnlichen Umgebung bzw. Situation stattfanden, können somit zusammengefasst werden. Eine während der Entwicklung durchzuführende Definition bestimmter Gruppen von Situationen ist aufgrund der Vielfalt möglicher Umgebungen nicht sinnvoll. Aufgezeichnete Interaktionen können also nicht, wie bei der Klassifizierung üblich, bekannten Gruppen von Situationen zugeordnet werden, sondern müssen gemeinsam auf unbekannte Gruppen hin untersucht werden. Dafür eignet sich die in Abschnitt 2.2.1.2 vorgestellte Methode des Clusterings.

Beim Clustering handelt es sich um ein Verfahren der Gruppierung, das ausgehend von einer Menge von Objekten, die als Merkmale z.B. die konkreten Positionen des Fahrzeugs als Geo-Koordinaten enthalten können, Gruppen von ähnlichen Objekten berechnet. Da der Gruppierungsprozess dazu über kein Vorwissen bzgl. der Gruppenzugehörigkeit einzelner Objekte verfügen muss, wird das Clustering dem *unüberwachten Lernen* zugeordnet. Während des Clusterings werden folglich sowohl alle Gruppen als auch die Gruppenzugehörigkeiten einzelner Objekte ermittelt. Die durch das Clustering resultierende Aufteilung der Menge von Objekten kann diverse Formen annehmen, die von einer disjunkten Aufteilung über eine überlappende (Fuzzy-Mengen) bis hin zu einer probabilistischen Gruppenzuordnung der Objekte reichen kann. Auch eine hierarchische Aufteilung der Klassen in verschiedene Granularitätsstufen, die sich in Form von Dendrogrammen visualisieren lassen, ist möglich.

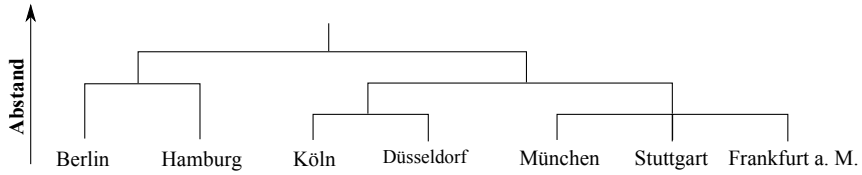


Abb. 2.5 Visualisierung einer hierarchischen Gruppierung deutscher Großstädte mit Hilfe eines Dendrogramms.

Aus der gewünschten Form der Aufteilung lässt sich ableiten, welcher Clustering-Algorithmus sich in besonderer Weise für das betreffende Anwendungsgebiet eignet. Wird beispielsweise eine hierarchische Aufteilung gesucht, so bietet es sich an, *agglomerativ* oder *teilend* zu gruppieren. Dabei wird im ersten Fall zunächst damit begonnen, jedem Objekt eine eigene Gruppe zuzuordnen. In jeder weiteren Stufe werden benachbarte Gruppen fusioniert, bis eine gewünschte Zahl an Gruppen erreicht ist. Das teilende Gruppieren verläuft dabei im umgekehrter Richtung und beginnt zunächst nur mit einer Gruppe, die alle Objekte enthält, und endet in einer Gruppierung mit der gewünschten Anzahl an Gruppen. Abbildung 2.5 zeigt das Ergebnis der Anwendung eines hierarchischen Verfahrens zum distanzbasierten Gruppieren von Städten. Wird jedoch eine disjunkte und nicht-hierarchische Aufteilung benötigt, und ist die Anzahl der Gruppen bekannt, so eignen sich *partitionierende Verfahren*. Diese ermitteln die optimale Aufteilung einer Menge von Objekten bzgl. eines zu definierenden Gütekriteriums. Partitionierende Verfahren generieren zunächst per Zufall eine Aufteilung mit der zuvor definierten Anzahl von Gruppen. Jeder weitere Schritt besteht aus der Prüfung, ob sich das Gütekriterium durch eine Änderung der Gruppenzugehörigkeit einzelner Objekte verbessern lässt.

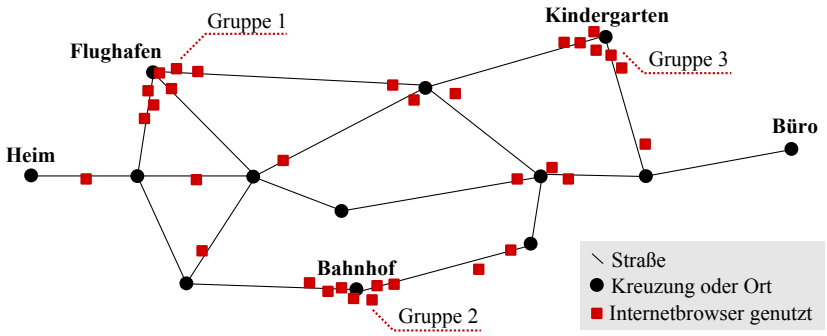


Abb. 2.6 Beispielhafte Nutzung des Internetbrowsers eines In-Car-Infotainment-Systems und die durch ein dichtebasiertes Verfahren ermittelten Gruppen häufiger Nutzung.

Im Fall der kontextsensitiven Personalisierung werden zwar disjunkte Aufteilungen gesucht, jedoch können keine Annahmen über die zu erwartende Anzahl der Gruppen getroffen werden. Da jeder Benutzer unterschiedlich intensiv und in einer unterschiedlichen Anzahl an Umgebungen mit einem System interagiert, ist eine Vorgabe der Anzahl der Gruppen auch nicht wünschenswert. Um dennoch eine disjunkte Aufteilung zu ermöglichen, wird in dem in dieser Forschungsarbeit vorgestellten Verfahren zur kontextsensitiven Personalisierung ein *dichtebasiertes Verfahren* angewandt. Beim dichtebasierten Verfahren werden (ausgehend von einer minimalen Dichte) Gruppen ermittelt, die mindestens eine minima-

le Anzahl von Objekten innerhalb bestimmter Radien enthalten. Die Anzahl der Gruppen variiert dabei in Abhängigkeit von der minimalen Dichte. In der Abbildung 2.6 wird eine dichtebasierte, disjunkte Aufteilung visualisiert, die aus drei unterschiedlichen Gruppen besteht.

Die Ähnlichkeit bzw. Distanz zweier Objekte $o_x, o_y \in O$ ergibt sich beim Clustering entweder anhand einer *Ähnlichkeitsmatrix*, oder durch ein Distanzmaß bzw. eine Distanzfunktion. Bei der Ähnlichkeitsmatrix handelt es sich um eine symmetrische NxN-Matrix, welche die Ähnlichkeitswerte aller Objektpaare enthält. Diese eignet sich insbesondere in Anwendungsgebieten, in denen die Distanzen einzelner Objekte nicht berechnet werden können, sondern stets aus einer Tabelle abgelesen werden müssen. Eine Distanzfunktion $d : OxO \rightarrow \mathbb{R}$ berechnet dagegen den Abstand zweier beliebiger Objekte. Für jede beliebige Distanzfunktion gilt dabei, dass $d(o_x, o_x) = 0$, $d(o_x, o_y) \geq 0$ und $d(o_x, o_y) = d(o_y, o_x)$. Wird durch die Menge von Objekten außerdem ein metrischer Raum aufgespannt, auf dem ein *metrisches Distanzmaß* definiert ist, dann gilt zusätzlich die Dreiecksungleichung $d(o_x, o_y) + d(o_y, o_z) \geq d(o_x, o_z)$. Zu den gebräuchlichsten metrischen Distanzfunktionen zählt der für den n-dimensionalen euklidischen Raum $\mathbb{R}^{\mathbb{N}^*}$ definierte *Euklidische Abstand* mit

$$d(o_x, o_y) = \sqrt{(o_{x_1} - o_{y_1})^2 + ... + (o_{x_n} - o_{y_n})^2}$$

und die für die Summennorm definierte *Manhattan-Distanz* mit

$$d(o_x, o_y) = \sum_{i=1}^{n} |o_{x_i} - o_{y_i}|.$$

Da das in dieser Forschungsarbeit verwendete Clustering-Verfahren nicht nur auf dem Vergleich numerischer Merkmale der Objekte basiert, werden weitere Distanzfunktionen in den nachfolgenden Kapiteln an geeigneter Stelle vorgestellt.

2.3.4 Vier Anwendungsfälle kontextsensitiver Personalisierungen im Fahrzeug

Das kontextsensitive Personalisieren von Benutzerschnittstellen wird insbesondere im Bereich mobiler Anwendungen eingesetzt, da die meist kleinen Bildschirme und wenigen Möglichkeiten der Interaktion dazu führen, dass die Geräte sehr unhandlich zu bedienen sind. Es ist also gerade in diesem Anwendungsgebiet von Vorteil, das Funktionsspektrum an die jeweilige Situation anzupassen. Auch automotive Benutzerschnittstellen können im Hinblick auf die Bedienbarkeit und Attraktivität von einer kontextsensitiven Personalisierung profitieren. Situationsabhängige Adaptionen können vor allem die Bedienung der neuen Applikationen und Services, die oftmals aus verwandten Anwendungsbereichen übernommen und für die Nutzung im Fahrzeug angepasst werden, verbessern. Um den Einsatz der kontextsensitiven Personalisierung im automotiven Bereich zu motivieren, wird nun ein beispielhaftes Benutzerverhalten beschrieben, das sich für vier unterschiedliche Anwendungsfälle kontextsensitiver Personalisierungen eignet:

„Montags bis freitags fährt Frau Müller jeden Morgen mit ihrem Auto zu ihrer Arbeitsstelle. Da sie auf dem Weg üblicherweise noch ihre Tochter zum Bahnhof bringt, wird die Fahrt am Bahnhof kurz unterbrochen. Nach dem Verlassen des Bahnhofsgeländes entschei-

det sich Frau Müller abhängig von der zu erwartenden Verkehrslage, ob sie die Fahrt entweder auf der Landstraße oder aber auf der Autobahn fortsetzt. Unabhängig davon, für welche Option sie sich entscheidet, passiert sie nach der Autobahn oder der Landstraße den selben Tunnel. Unmittelbar bei der Einfahrt in den Tunnel bricht der Empfang ihres präferierten Radiosenders JazzFM ab. Auch nach dem Verlassen des Tunnels ist der Empfang von JazzFM weiterhin nicht möglich. Daher entscheidet sich Frau Müller stets, ihren ebenfalls präferierten Radiosender KlassikFM aufzurufen. Kurz darauf erreicht sie ihre Arbeitsstelle.

Montags bis freitags tritt Frau Müller ihre Rückfahrt üblicherweise zwischen 17:00 Uhr und 18:30 Uhr an. Beim Passieren eines benachbarten Tunnels bricht der Empfang des auf der Hinfahrt eingestellten Radiosenders KlassikFM ab. Nach dem Verlassen des Tunnels entscheidet sich Frau Müller stets, ihre Einstellung im Radio rückgängig zu machen und wechselt daraufhin zurück auf den Radiosender JazzFM. Unmittelbar darauf entscheidet sich Frau Müller, abermals abhängig von der Verkehrslage, ob sie die Fahrt entweder auf der Landstraße oder der Autobahn fortsetzt. Kurz vor der Ankunft am Wohnort fährt sie stets noch einmal zum Bahnhof, um ihre Tochter abzuholen. Während sie am Bahnhof auf ihre Tochter wartet, öffnet sie im Internetbrowser des In-Car-Infotainment-Systems regelmäßig die Webseite der Bahn, um die aktuellen Ankunftszeiten der Züge zu erfragen. Nachdem ihre Tochter angekommen ist, wird die Fahrt bis zum Wohnort ohne Unterbrechung fortgesetzt.

Freitags ist es gelegentlich möglich, dass Frau Müller ihre Arbeitsstelle bereits zwischen 14:00 Uhr und 16:30 verlässt. An einem solchen Tag sieht Frau Müller immer von einem Stopp am Bahnhof ab, da ihre Tochter erst sehr viel später mit den Zug ankommt. Daher ruft sie über das In-Car-Infotainment-System unmittelbar beim Beginn der Rückfahrt ihren Ehegatten an und bittet ihn, an diesem Tag das Abholen der Tochter zu übernehmen. Ihre Rückfahrt verläuft danach bis auf das Aussetzen des Stopps am Bahnhof in gleicher Weise wie an allen anderen Arbeitstagen.“

Das Benutzerverhalten von Frau Müller wird aus Übersichtsgründen noch einmal in der Abbildung 2.7 visualisiert. Alle Interaktionen mit dem Fahrzeug, wie z.B. das Starten des Autos oder der Wechsel des Radiosenders, werden in der Abbildung mit Hilfe von roten und gestrichelten Linien hervorgehoben. Da die Steuerung des Fahrzeugs in den folgenden Anwendungsfällen keine Rolle spielt, werden diese Interaktionen hier nicht weiter aufgeführt.

Bevor jedoch im einzelnen auf die Anwendungsmöglichkeiten der kontextsensitiven Personalisierung bzgl. des Benutzerverhaltens von Frau Müller eingegangen wird, werden zunächst zwei allgemeine Formen der kontextsensitiven Personalisierung automotiver Benutzerschnittstellen vorgestellt. Eine Möglichkeit kontextsensitiv zu personalisieren besteht darin, einen separaten Bildschirm in der Mittelkonsole einzuführen, auf dem ausschließlich kontextsensitiv personalisierte Inhalte präsentiert werden. Die Inhalte auf diesem Bildschirm können beispielsweise aus *personalisierten Schaltflächen* (nachfolgend als *kontextsensitiv personalisierte Schnellstartbuttons* bezeichnet) bestehen, deren Funktionen sich abhängig vom Ort, der Zeit oder anderer Umweltfaktoren ändern. Werden nun in einer bestimmten Situation wiederholt spezifische Funktionen des In-Car-Infotainment-Systems aufgerufen, so können diese Funktionen beim Eintreten einer solchen Situation als kontextsensitiv personalisierte Schnellstartbuttons angeboten werden. Somit können Benutzer ihre präferierten Funktionen ohne den Umweg über eine Navigation durch die evtl. komplexe Menüstruktur direkt durch Betätigung dieser Schnellstartbuttons ausführen. Die durch diese Art der Personalisierung ausgeführten Anpassungen der Benutzerschnittstelle gehören zu der in Unterabschnitt 2.2.2 vorgestellten Form der proaktiven Hilfe. Der Benutzer wird da-

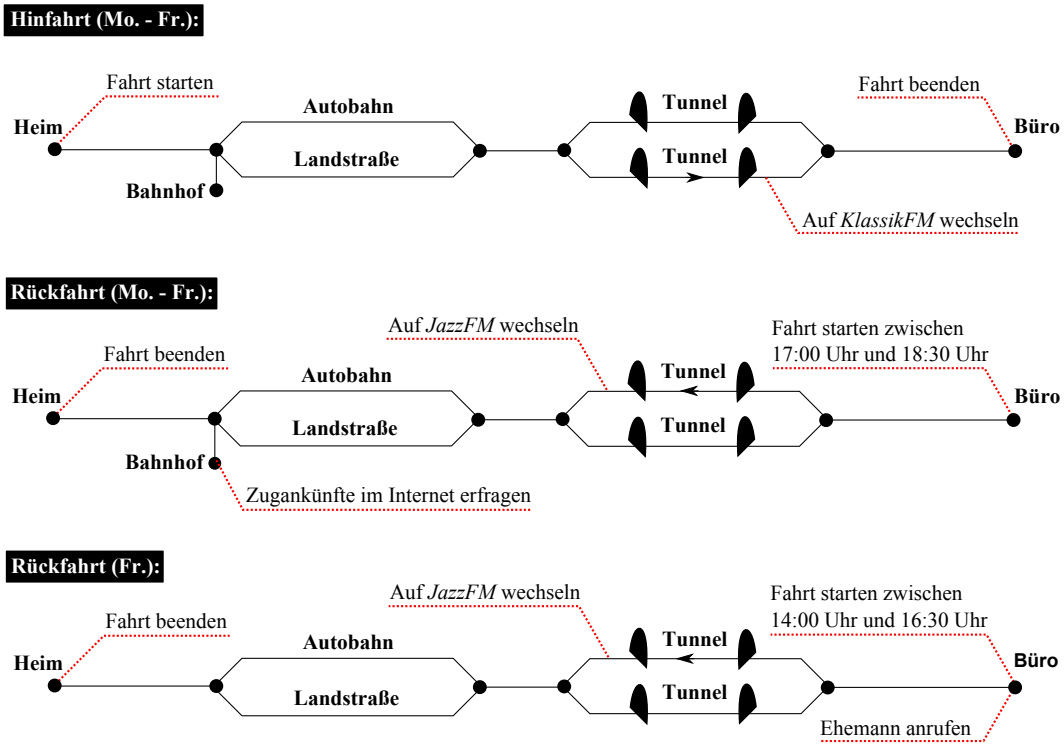


Abb. 2.7 Beispielhaftes Benutzerverhalten von Frau Müller an Arbeitstagen.

mit lediglich auf Funktionen hingewiesen, die er im Bedarfsfall jedoch manuell auszuführen hat.

Die zweite Form der kontextsensitiven Personalisierung besteht darin, die besagten Funktionen zu automatisieren (nachfolgend als *kontextsensitiv personalisierte Automatisierung* bezeichnet). Anpassungen dieser Art werden den im Unterabschnitt 2.2.2 beschriebenen automatisierten Routinen zugeordnet. Während der kontextsensitiv personalisierten Automatisierung werden häufig genutzte Funktionen beim Eintritt einer entsprechenden Situation automatisch vom In-Car-Infotainment-System ausgeführt. Ein Nachteil der kontextsensitiv personalisierten Automatisierung besteht darin, dass die Automatisierungen auch in Situationen ausgeführt werden, die laut System zwar auf eine Automatisierung schließen lassen, jedoch nicht in jedem Fall vom Benutzer in gleicher Weise interpretiert werden. Diese Automatisierungen müssen dann vom Benutzer manuell rückgängig gemacht werden. Um dieser Gefahr entgegen zu wirken, kann die Automatisierung kurz vor ihrer Ausführung akustisch oder visuell angekündigt werden. Der Benutzer kann dann die Automatisierung über die Spracheingabe oder durch ein einfaches Betätigen einer Schaltfläche - wenn nötig - abbrechen. Auf Basis dieser beiden Formen der kontextsensitiven Personalisierung lassen sich nun folgende Anwendungsfälle aus dem Benutzerverhalten von Frau Müller ableiten:

- **Anwendungsfall A:** Das kontextsensitiv personalisierbare In-Car-Infotainment-System kann erkennen, dass die Frau immer dann den Radiosender *KlassikFM* einstellt, wenn sie sich am Tunnelausgang auf dem Weg zur Arbeitsstelle befindet. Gleiches gilt für die auf der Rückfahrt stattfindenden Radiosenderwechsel auf *JazzFM* am Ausgang des benachbarten Tunnels. Damit ist das In-Car-Infotainment-System in der Lage, beim durchfahren dieser Orte die entsprechenden Radiosenderwechsel automatisch auszuführen (kon-

textsensitiv personalisierte Automatisierung). Frau Müller wird in diesen Fällen also vollständig von der Bedienung des In-Car-Infotainment-Systems befreit.

- **Anwendungsfall B:** Die Kausalität, dass Frau Müller an Arbeitstagen immer zur Arbeitsstelle fährt, wenn sie von ihrem Wohnort losfährt, kann vom In-Car-Infotainment-System ebenfalls erkannt werden. Die automotive Benutzerschnittstelle kann auf Basis dieser Informationen beim Beginn der Fahrt am Wohnort einen kontextsensitiv personalisierten Schnellstartbutton anzeigen, der bei Ausführung alle Verkehrsinformationen bzgl. der Strecke zur Arbeitsstelle anzeigt. Damit kann Frau Müller nach der direkten Betätigung des Schnellstartbuttons und der angezeigten Verkehrsinformationen besser einschätzen, ob sie über die Landstraße oder über die Autobahn zu ihrer Arbeitsstelle gelangt. Das In-Car-Infotainment-System geht in diesem Fall also einen Schritt weiter, und folgert aus der kontextsensitiven Präferenz die Anwendung einer Interaktion, die von Frau Müller bisher nie in dieser Situation durchgeführt wurde. Hierbei handelt es sich also um eine Personalisierung zur Förderung des explorativen Benutzerverhaltens und zur Optimierung der Fahrzeit.

- **Anwendungsfall C:** Das In-Car-Infotainment-System ist in der Lage, die Kausalität zwischen dem Beginn der Rückfahrt von der Arbeitsstelle und dem am Bahnhof stattfindenden Aufruf der Webseite der Bahn zu erkennen. Dies ermöglicht es dem In-Car-Infotainment-System, immer dann automatisch den Internetbrowser zusammen mit der Webseite der Bahn zu öffnen, wenn Frau Müller mit dem Fahrzeug auf das Bahngelände gelangt und zuvor von der Arbeitsstelle losgefahren ist. Dabei stellt das System durch die Kausalität sicher, dass der Internetbrowser nicht auch auf der Hinfahrt beim kurzen Halt am Bahnhof automatisch ausgeführt wird. Durch diese kontextsensitiv personalisierte Automatisierung erspart sich Frau Müller somit das Öffnen des Internetbrowsers und die mühevolle Eingabe der URL über die Bildschirmtastatur.

- **Anwendungsfall D:** Das In-Car-Infotainment-System kann erkennen, dass Frau Müller immer dann ihren Ehemann anruft, wenn sie freitags zwischen 14:00 Uhr und 16:30 Uhr von ihrer Arbeitsstelle losfährt. Diese orts- und zeitabhängige Präferenz kann dazu verwendet werden, einen kontextsensitiv personalisierten Schnellstartbutton am Beginn der Rückfahrt anzuzeigen, der es Frau Müller durch eine direkte Betätigung ermöglicht, den Anruf an ihren Ehemann zu initiieren. Frau Müller wird dadurch nicht nur an den notwendigen Anruf erinnert, sondern auch noch durch das System von der Eingabe der Telefonnummer befreit.

In allen Anwendungsfällen werden die kontextsensitiven Präferenzen automatisiert erkannt und die zugehörigen Adaptionen kontextsensitiv ausgeführt. Werden auf Basis dieser erkannten kontextsensitiven Präferenzen jedoch fälschlicherweise Adaptionen ausgeführt, die vom Benutzer in jedem Fall unerwünscht sind, so kann dies wie oben beschrieben jederzeit abgebrochen oder rückgängig gemacht werden. Das Abbrechen oder das Zurücksetzen kann vom System zusätzlich als negatives Feedback interpretiert werden und somit den Lernprozess positiv beeinflussen. Um dennoch den dadurch induzierten Aufwand zu umgehen, ist es möglich, jede erkannte kontextsensitive Präferenz dem Benutzer in der Benutzerschnittstelle zunächst zu präsentieren. Der Benutzer kann dann entscheiden, ob die dazugehörigen Adaptionen automatisiert ausgeführt oder die speziellen kontextsensitiven

Präferenzen in Zukunft vom Prozess der Personalisierung ignoriert werden sollen. Damit liegt es in der Hand des Benutzers, die kontextsensitive Personalisierung nach Belieben zu steuern. Eine solche Methode des Feedbacks kann in beiden Formen der kontextsensitiven Personalisierung zur Anwendung kommen. Es wäre beispielsweise im Rahmen der kontextsensitiven Personalisierung von Schnellstartbuttons möglich, die Anzeige von Schnellstartbuttons auf die im Sinne des Benutzers relevanten Funktionalitäten zu beschränken. Eine beispielhafte Implementierung der Feedback-Möglichkeit wird im Abschnitt 5.1.2 vorgestellt.

Da die Schnellstartbuttons immer in einem separaten Bildschirm angezeigt werden, ist der Grad an Stabilität zunächst hoch. Aufgrund der sich möglicherweise laufend ändernden Reihenfolge, mit der die Schnellstartbuttons auf dem separaten Bildschirm angezeigt werden, ist die Stabilität letztendlich jedoch gering. Der Benutzer muss schließlich jeden Schnellstartbutton zunächst visuell aufnehmen, bevor er dessen Funktionalität erkennt. Die kontextsensitiv personalisierte Automatisierung weist ebenfalls eine sehr geringe Stabilität auf, da die Ausführung einer oder mehrerer gebündelter Interaktionen theoretisch in jedem Moment stattfinden kann. Aufgrund der großen logischen Distanz zwischen den betroffenen Applikationen und den im separaten Bildschirm angezeigten Schnellstartbuttons, ist die Lokalität dieser Adaption als gering einzuschätzen. Dies trifft ebenfalls auf die kontextsensitiv personalisierte Automatisierung zu, deren Ausführung, wie bereits bei der Stabilität beschrieben, jederzeit und unabhängig vom jeweiligen Zustand des Menüs im Hauptbildschirm stattfinden kann.

Durch die visuelle Trennung des statischen und dynamischen Anteils der Benutzerschnittstelle ist es im Fall der kontextsensitiv personalisierten Schnellstartbuttons möglich, die Schnellstartbuttons als eine alternative Möglichkeit der Bedienung anzubieten, die vom Benutzer entweder genutzt oder ignoriert werden kann. Demgegenüber führt die kontextsensitiv personalisierte Automatisierung zwangsläufig zu einer Ausführung der bevorzugten Funktionalität. Die Vorhersehbarkeit hängt in beiden Formen davon ab, wie komplex sich die zur kontextsensitiven Präferenz gehörende Situation darstellt. Da die Schnellstartbuttons für die bevorzugten Radiosenderwechsel nur in Abhängigkeit vom Ort erscheinen, ist die Vorhersehbarkeit in diesem Fall gegeben. Der Entscheidungsprozess für einen automatischen Aufruf des Internetbrowsers erweist sich jedoch als deutlich komplexer, da in diesem Fall verschiedene Interaktionen und deren Kontexte in kausaler Beziehung zueinander stehen. Die Genauigkeit und die Schnelligkeit hängen maßgeblich von dem Verfahren ab, das innerhalb des adaptiven Systems zur Erkennung der kontextsensitiven Präferenzen genutzt wird. Der relative kognitive und physische Aufwand und der Grad der Ablenkung werden im Kapitel 7 durch die Gegenüberstellung beider Varianten im Rahmen einer Benutzerevaluation ermittelt. Aufgrund des in dieser Forschungsarbeit vorgestellten regelbasierten Verfahrens wird außerdem sichergestellt, dass keine widerrechtlichen Personalisierungen ausgeführt werden können.

Kapitel 3
Verwandte Arbeiten

Mark Weiser propagierte bereits 1991 seine Vision [Weiser, 1991] einer Welt, in der neue Technologien so weit Einzug in den Alltag halten, dass sie aus Sicht der Benutzer nicht mehr als solche wahrgenommen werden. Weiser führte für diese Art der Entwicklung den Begriff *Ubiquitous Computing* [Weiser, 1991] ein, der den Zustand der nahtlosen Integration aller Art von Technologien im Umfeld des Menschen beschreibt. Die Lebensweise des Menschen sollte sich demnach nicht an den neuen Technologien ausrichten, sondern von deren neuen Möglichkeiten profitieren. Dabei sollte möglichst jedes technische Objekt, das sich in der Umgebung des Menschen befindet, Teil eines gemeinsamen intelligenten Netzes sein, das dem Menschen während seines Daseins unterstützend beiseite steht.

Davon inspiriert wurde 2001 von der ISTAG der Begriff *Ambient Intelligence* eingeführt [ISTAG, 1999]. Ambient Intelligence umschreibt ein interdiziplinäres Forschungsgebiet, das sich der Methoden und Verfahren der künstlichen Intelligenz, des *Pervasive Computing* [Hansmann u. a., 2001], der Mensch-Computer-Interaktion, der *Sensornetze* [Dargie u. Poellabauer, 2010] und der *Multiagentensysteme* [Woolridge, 2009] bedient, um das von Weiser lancierte Paradigma des Ubiquitous Computing umzusetzen [Nakashima u. a., 2009]. Die dafür notwendige physikalische Infrastruktur wird unter den Begriffen *Intelligent Environment* [Coen, 1998] oder *Smart Environment* [Nakashima u. a., 2009] zusammengefasst. Auch wenn das Forschungsgebiet der Ambient Intelligence inhaltlich damit nicht eindeutig vom Ubiquitous Computing abgegrenzt werden kann, so konzentriert sich ein nicht unerheblicher Anteil der Forschung auf das Entwickeln geeigneter Mensch-Maschine-Schnittstellen innerhalb der intelligenten Umgebungen. Augusto beschreibt dazu ein unter den Gesichtspunkten der Ambient Intelligence entwickeltes System als „*A digital environment that proactively, but sensibly, supports people in their daily lives.*“[Augusto u. McCullagh, 2007]. Eine solche Definition wird dann auch durch die von Aarts [Aarts, 2004] für die Ambient Intelligence aufgelisteten Kernpunkte konkretisiert. Aarts nennt dabei die Adaptivität, das Personalisieren und das Antizipieren neben der Sensitivität bzgl. des Kontextes und der Einbettung in vernetzen Geräten als die Grundpfeiler zur Gestaltung geeigneter Mensch-Maschine-Schnittstellen in intelligenten Umgebungen. Da das in dieser Forschungsarbeit vorgestellte Verfahren viele der von Aarts beschriebenen Aspekte einer intelligenten Umgebung implementiert, werden im Nachfolgenden verwandte Arbeiten aus dem Gebiet der Ambient Intelligence vorgestellt und der Stand der Forschung insbesondere hinsichtlich der kontextsensitiven Personalisierung erörtert. Abschließend wird außerdem auf das Gebiet der Simulation situationsabhängigen Benutzerverhaltens eingegangen und die in diesem Forschungszweig angewandten Techniken eingeführt.

3.1 Kontextsensitive Personalisierung im Umfeld der Ambient Intelligence

Das Konzept eines autonomen Assistenten, der den Benutzer während der Bedienung eines Systems, wie von Aarts propagiert, beobachtet und proaktiv unterstützt, ohne unmittelbar vor der Ausführung einer eigenverantwortlichen Aktion von dem Benutzer dazu explizit aufgefordert zu werden, wird erstmalig von Maes in [Maes, 1994] anhand des *persönlichen Assistenten* beschrieben. Der persönliche Assistent ist aufgrund konzeptuellen Wissens oder eines individuellen Benutzermodells in der Lage, wiederkehrende Aktionen zu automatisieren, komplexe Interaktionen mit dem System zu erleichtern und sogar zu antizipieren. Maes differenziert dabei grundlegend drei Klassen von persönlichen Assistenten: *benutzerprogrammierbare*, *wissensbasierte* und *lernende Assistenten*. Benutzerprogrammierbare Assistenten erfordern die explizite Eingabe von Präferenzen durch den Benutzer selbst. Der benutzerprogrammierbare Assistent dient also nur zur Ausführung individuell konfigurierbarer Aktionen, deren sich der Benutzer ohnehin bewusst ist. Dagegen versucht ein wissensbasierter Assistent die Pläne des Benutzers anhand impliziter Eingaben und allgemeingültiger Konzepte zu erkennen und entsprechend bestimmter Regeln zu handeln. Oftmals besitzen wissensbasierte Assistenten jedoch eine geringere Flexibiltiät als benutzerprogrammierbare Assistenten, da nicht jedes potentielle Benutzerverhalten zum Zeitpunkt der Entwicklung vorhergesagt und daher als Konzept zur Laufzeit vorliegen kann. Die Akkumulation von Informationen über einen Benutzer, wie sie der Prozess der Personalisierung vorschreibt, ist aufgrund des regelbasierten Ansatzes nicht vorgesehen. Lernende Assistenten versuchen hingegen, Informationen über den Benutzer implizit zu sammeln und mit Hilfe minimalen konzeptuellen Vorwissens und maschineller Lerntechniken auf regelmäßiges Benutzerverhalten hin zu untersuchen. Ein lernender Assistent benötigt zur Ausführung proaktiver Anpassungen also weder umfangreiches Vorwissen, noch eine vom Benutzer zu initiierende Konfiguration. Damit erfüllen lernende Assistenten alle Voraussetzungen für den im Abschnitt 2.2 beschriebenen Prozess der Personalisierung und die für die Umsetzung der Ambient Intelligence notwendigen Merkmale einer vorausschauenden Benutzerschnittstelle.

Den für die Realisierung der Ambient Intelligence notwendigen Anpassungsprozess innerhalb des Netzes aus lernenden Assistenten und der Infrastruktur unterteilt Aztiria [Aztiria u. a., 2010a] in drei aufeinander folgende Teilprozesse: *Sensing*, *Reasoning* und *Acting*. Mit Hilfe von Sensoren werden zunächst sowohl die Vorfälle in der Umgebung als auch die durch den Benutzer ausgelösten Vorfälle in maschinenlesbare Ereignisse umgewandelt und zu abstrakteren Modellen des Kontextes und der Interaktionen verarbeitet. Während im Teilprozess des Sensing also lediglich der aktuelle Kontext oder die aktuelle Interaktion ermittelt wird, werden im Reasoning alle Kontexte und Interaktionen gesammelt und auf kontextsensitive Präferenzen hin untersucht. Außerdem wird während des Reasoning entschieden, ob eine sich aus den bereits zuvor ermittelten kontextsensitiven Präferenzen ergebene Anpassung zum derzeitigen Zeitpunkt angebracht wäre. Abschließend beschreibt der Teilprozess des Acting die eigentliche Modifikation der jeweiligen Benutzerschnittstelle, also die Anwendung auf Applikationsebene. Die vorliegende Forschungsarbeit knüpft an diese Gliederung an und widmet sich in den nachfolgenden Unterabschnitten jeweils einem Teilprozess. Da das in dieser Arbeit vorgestellte Verfahren zur automatischen Extraktion kontextsensitiver Präferenzen größtenteils dem Reasoning zugeordnet werden kann, werden verwandte Arbeiten diesbezüglich ausführlicher behandelt.

3.1.1 Verfahren zur Erfassung der Interaktionen und des Kontextes

I. Allg. findet die Abtastung des Kontextes und der Interaktionen über Sensoren statt. Dabei erfolgt der Prozess entweder zentral innerhalb eines Sensors oder auch verteilt über ein Netz von Sensoren. Die Sensoren können im lernenden Assistenten integriert sein und/oder durch die umgebende Infrastruktur bereitgestellt werden. Indulska unterscheidet dabei drei Arten von Sensoren: *physikalische*, *virtuelle* und *logische* Sensoren [Indulska u. Sutton, 2003].

Zu den von physikalischen Sensoren typischerweise erfassten Daten zählen unter anderem die Position (beispielsweise mit Hilfe der RFID- oder GPS-Technologien), die relative Ausrichtung von Geräten durch Beschleunigungssensoren oder Magnetometer und die Intensität des Umgebungslichts durch Lichtsensoren. Im automotiven Bereich kommen zusätzlich noch Drucksensoren in den Sitzen, die Temperatur- und Feuchtigkeitsfühler im Innen- und Außenbereich, Müdigkeitssensoren etc. hinzu. Die Auswertung der durch physikalische Sensoren erfassten Daten beginnt dabei oftmals bereits innerhalb der Sensoren. Beispielsweise können Temperatursensoren Werte bereitstellen, die den Mittelwert der Temperatur über einen spezifischen Zeitraum kennzeichnen. Da kontinuierliche Daten in jedem Fall diskretisiert werden, findet in jedem physikalischen Sensor ein minimales Maß an Abstraktion statt. Eine Auflistung diverser physikalischer Sensoren, deren Daten zur sinnvollen Kontextualisierung eines Systems beitragen können, kann dem Artikel von Schmidt [Schmidt u. Laerhoven, 2001] entnommen werden.

Werden Sensoren in die Benutzerschnittstelle integriert, so handelt es sich um virtuelle Sensoren. Diese dienen dem Abtasten von Ereignissen, die während der Interaktion zwischen dem Benutzer und dem adaptiven System auftreten. Hilbert beschreibt in [Hilbert u. Redmiles, 2000] sechs Abstraktionsebenen, die sich auf diese Weise in der Benutzerschnittstelle ergeben können. Hilberts Kategorisierung beginnt dabei mit den physikalischen Ereignissen, wie z.B. das Drücken einer Schaltfläche, und endet auf der höchsten Abstraktionsebene mit Ereignissen, die z.B. das Beenden einer spezifischen Tätigkeit des Benutzers beschreiben. In diversen Arbeiten werden insbesondere die Ereignisse aus den unteren Abstraktionsebenen abgetastet, um beispielsweise Vorhersagen über die Nutzung der Kommandozeile [Davison u. Hirsh, 1998; Korvemaker u. Greiner, 2000] anzufertigen oder allgemein das Benutzerverhalten zu antizipieren [Künzer u. a., 2004; Nazemi u. a., 2010]. Erfolgreich eingesetzt wird die Analyse dieser Ereignisse im Forschungsgebiet des *Web Mining*. Da hier die Ereignisse serverseitig sehr fein granular abgetastet werden können, ist es beispielsweise möglich, Verlaufsprofile von Benutzern auf Basis der von ihnen aufgerufenen Webseiten zu erstellen. Die interaktionsbezogenen Daten können hierbei auch clientseitig aufgezeichnet und dann unmittelbar dem Server, auf dem das Web Mining letztendlich durchgeführt wird, übermittelt werden. Diese werden dann oft zur automatischen Modifikation der Struktur von Webseiten [Anderson u. a., 2002; Perkowitz u. Etzioni, 1999] verwendet. Eine Übersicht bekannter Methoden des Web Mining kann der Arbeit von Kosala [Kosala u. Blockeel, 2000] entnommen werden.

Logische Sensoren liefern hingegen Informationen, die sich aus den Daten zahlreicher virtueller und physikalischer Sensoren zusammensetzen oder sich durch weiteres explizites Wissen folgern lassen. Damit gehören die Sensoren, die in der Hilbertschen Kategorisierung Ereignisse der höheren Abstraktionsebenen verarbeiten, ebenfalls zu den logischen Sensoren. Ein logischer Sensor kann beispielsweise auf der Basis der Eingaben aus der Benutzerschnittstelle und zusätzlichen Wissens ermitteln, welcher Tätigkeit der Benutzer nachgeht. Aber auch im Bereich der physikalischen Sensoren kann eine Kombination diverser Sensoren von Vorteil sein. Beispielsweise verbessert die zusätzliche Berücksichtigung

der Signalstärke des drahtlosen Netzwerks oder der eindeutigen Identifikationsnummer der nächsten Basisstation im Mobilfunk die Zuverlässigkeit und Genauigkeit der durch die GPS-Technologie ermittelten Position.

Zur gemeinsamen Verarbeitung der Daten aller Sensoren kommen üblicherweise sogenannte *Context-Aware-Frameworks* zum Einsatz. Diese oftmals als Middleware-Infrastruktur vorgesehenen Systeme bilden dabei die zentrale Schnittstelle zwischen den kontextsensitiven Anwendungen und den Sensoren. Es wird durch die Kapselung der für die Verarbeitung der Rohdaten zuständigen Algorithmen innerhalb der Context-Aware-Frameworks außerdem erreicht, dass Sensoren durch neue Technologien ausgetauscht oder ergänzt werden können, ohne dass es einer Änderung der Implementierung auf Seiten der Anwendungen bedarf. Eine Übersicht der bekanntesten Context-Aware-Frameworks zur Verarbeitung der von Sensoren gelieferten Daten hin zu abstrakteren Modellen des Kontextes kann der Studie von Baldauf [Baldauf u. a., 2007] entnommen werden. Prinzipiell eigenen sich die meisten der dort vorgestellten Methoden dazu, Ereignisse aus der automotiven Benutzerschnittstelle und den im Fahrzeug zur Verfügung stehenden physikalischen Sensoren für den Teilprozess des Reasonings vorzubereiten.

3.1.2 Verfahren zur Ermittlung kontextsensitiver Präferenzen

Wie bereits einleitend beschrieben, werden während des Reasonings persönliche Präferenzen ermittelt, die sich durch die Analyse der durch den Teilprozess des Sensings zur Verfügung gestellten Daten ergeben. Im speziellen Fall der kontextsensitiven Personalisierung wird jedoch neben den Interaktionen auch der zugehörige Kontext bei der Analyse berücksichtigt. Lernende Assistenten, die eine kontextsensitive Personalisierung unterstützen, müssen demnach in der Lage sein, die sich aus der Häufigkeit bestimmter Interaktionen ergebenden persönlichen Präferenzen auch unter den Gesichtspunkten der während der Interaktionen gültigen Kontexte zu untersuchen. Bekannte Arbeiten auf diesem Gebiet lassen sich anhand der Flexibilität der vorgestellten Ansätze gegenüber den zu untersuchenden Daten in allgemeine und spezielle Methoden unterteilen. Während spezielle Methoden eingesetzt werden, um spezifische Daten wie z.B. GPS-Positionen zu verarbeiten, können allgemeinere Methoden auch zur Untersuchung heterogen strukturierter Daten, wie beispielsweise eine Kombination aus der Temperatur in reellen Zahlen und der Fahrzeuginsassen als Objekte, eingesetzt werden. Die nachfolgende Erörterung bekannter Methoden und Verfahren konzentriert sich dabei insbesondere auf Techniken der kontextsensitiven Personalisierung, die möglichst ohne Überwachung durch den Benutzer oder eines Experten, also ohne einen zusätzlichen Eingriff während der Laufzeit, ausgeführt werden können.

Allgemeine Methoden

Eine besonders in den letzten Jahren häufig verwendete Methode ist das Ermitteln von *Association Rules*. Eine Association Rule beschreibt die gegenseitige Abhängigkeit zweier Typen von Objekten innerhalb einer Menge von geordneten Tupeln von Objekten. In dieser Menge kann jeder Objekttyp mehrfach auftreten. Besteht die Menge beispielsweise aus mit Zeitstempeln versehenen Aufzeichnungen von Interaktionen (Tupeln) eines Benutzers, die jeweils aus mehreren Einstellungen des Systems (Objekten) bestehen und über einen

längeren Zeitraum gesammelt wurden, dann wird der Zusammenhang, dass z.B. eine Einstellung A oft zusammen mit einer Einstellung B auftritt, als Association Rule bezeichnet. Die Kennwerte einer Association Rule geben dabei jeweils an, wie häufig dieser Zusammenhang in der Gesamtmenge aller Tupel erkannt wurde (*Unterstützung*) und wie hoch die Wahrscheinlichkeit ist, dass eine Einstellung A zusammen mit einer Einstellung B auftritt (*Konfidenz*).

Schiaffino stellt in [Schiaffino u. Amandi, 2006] einen *persönlichen Agenten* vor, der den Benutzer bei der Bedienung seines elektronischen Kalenders unterstützt. Der persönliche Agent ist in der Lage, typisches Benutzerverhalten aus den individuellen Interaktionen abzuleiten und die Benutzerschnittstelle auf Basis dieser Regelmäßigkeiten automatisch zu verändern. Beispielsweise kann der Benutzer während des Eintragens eines Termins für eine Sitzung benachrichtigt werden, dass ein geladener Gast üblicherweise in der für den Termin vorgesehenen Zeit nicht erreichbar ist. Der vorgestellte Ansatz untersucht dabei insbesondere die individuellen Präferenzen bzgl. der Art der Benachrichtigung, die vom Benutzer bevorzugt wird. In der Methode werden dazu Sequenzen von Ereignissen analysiert, die während der Benutzung aufgezeichnet wurden und deren Ereignisse jeweils einen Kontext, eine bestimmte Aktion (z.B. Warnung), die genutzte Modalität, die zu diesen Zeitpunkt ausgeführte Aufgabe, usw. enthalten. Mit Hilfe des in [Agrawal u. Srikant, 1994] vorgestellten Apriori-Algorithmus werden die Sequenzen von Ereignissen zunächst auf das Vorhandensein von Association Rules hin untersucht. Die Ergebnisse werden dann durch einen zusätzlichen Filter auf die für den Anwendungsfall relevanten Association Rules reduziert. Ein Filter könnte dabei beispielsweise allgemein die Bedingung spezifizieren, dass nur Association Rules Beachtung finden, in denen ein Feedback auf eine Benachrichtigung folgt. Die auf diese Weise gefilterten Regeln bilden dann das typische Profil eines Benutzers in Bezug auf die situativen Vorlieben beim Benachrichtigen ab. Dem persönlichen Agenten wird also darüber Auskunft erteilt, in welcher Situation der Benutzer welcher Art von Benachrichtigung den Vorzug gibt.

Tanca [Tanca u. a., 2011] beschreibt ein Verfahren zur kontextsensitiven Personalisierung von Datenbankabfragen. Als Anwendungsbeispiel dient ein Empfehlungsdienst für Hotels, der auf Basis der aktuellen Position, den persönlichen Präferenzen und dem sozialen Kontext verschiedene Empfehlungen ausspricht. Das von Tanca vorgestellte System greift dabei ebenfalls auf Association Rules zurück, die durch die Analyse der impliziten Eingaben des Benutzers ermittelt werden. Eine gefundene Regelmäßigkeit wird dabei durch ein 3-Tupel dargestellt, welches die Association Rule und die beiden Kennwerte Unterstützung und Konfidenz enthält. Die Association Rule bildet einen konkreten Kontext auf eine bevorzugte Parametrisierung des Systems ab. Beispielsweise könnte eine solche Regel beschreiben, dass nur dann Spa-Hotels empfohlen werden, wenn sich der Benutzer in Anwesenheit von Freunden befindet. Andernfalls bevorzugt der Benutzer Business-Hotels. Die Unterstützung und die Konfidenz geben jeweils an, wie oft der Benutzer in Anwesenheit von Freunden einer Spa-Hotel Empfehlung folgte und der bedingten Wahrscheinlichkeit, dass der Benutzer dieser Empfehlung den Vorzug gibt, wenn er sich in Begleitung von Freunden befindet. Tanca führt außerdem noch an, dass das System auch um eine Personalisierung für neue Benutzer erweitert werden könnte, für die noch keine Aktivitäten vorliegen. Dabei würde die Ähnlichkeit der Präferenzen zwischen zwei Benutzern Aufschluss darüber geben, wie der bisher unbekannte Benutzer im gleichen Kontext agieren könnte. Die mitunter hohe Zahl an Association Rules, die sich in beiden Ansätzen von Schiaffino und Tanca ergeben können, erweist sich zwar hinsichtlich einer umfassenden Ermittelung aller Korrelationen als hilfreich, führt aber in der Praxis zur Anwendung zusätzlich zu definierender Filter. Da-

her werden Association Rules vornehmlich dann ermittelt, wenn ein Experte zur Bewertung der Relevanz gefundener Regeln hinzugezogen werden kann.

Alternativ dazu bedient sich die Methode des fallbasierten Schließens des Vergleichs neuer und historischer Probleme, um bekannte Lösungsstrategien auch auf bisher unbekannte, aber ähnliche Probleme anzuwenden. Das konkrete Problem wird zusammen mit der angewandten Lösungsstrategie üblicherweise als ein *Fall (engl. case)* [Aamodt u. Plaza, 1994] bezeichnet. Wird die Methode des fallbasierten Schließens zur kontextsensitiven Personalisierung verwendet, so umschreibt das Problem üblicherweise einen konkreten Kontext und die Lösungsstrategie eine zu diesem Kontext passende Konfiguration der Benutzeroberfläche. Tritt ein neuer und noch unbekannter Kontext auf, so wird dieser mit den Kontexten historischer Fälle verglichen, um auf eine geeignete Konfiguration zu schließen. Ziel ist es also, aus einer Menge von Fällen, die gemeinsam den Erfahrungsschatz ausmachen, das ähnlichste Problem zu extrahieren und die bekannte Lösungsstrategie anzuwenden. Im Gegensatz zur oben genannten Regel-Induktion wird beim fallbasierten Schließen jedoch implizit generalisiert. Jeder neue Fall wird einzeln mit jedem konkreten Fall aus dem Erfahrungsschatz verglichen. Ist der neue Fall ähnlich oder sogar gleich, dann kann die Lösungsstrategie des alten Problems übernommen und der neue Fall gegebenenfalls dem Erfahrungsschatz hinzugefügt werden. Anders als bei der Regel-Induktion, in der mehrere ähnliche Fälle zunächst zusammengefasst und dann mit jedem neuen Fall verglichen werden, findet die Generalisierung beim fallbasierten Schließen erst zum Zeitpunkt des Vergleichs konkreter Fälle statt.

Mikalsen präsentiert in [Mikalsen u. Kofod-Petersen, 2005] ein System, das typisches Benutzerverhalten auf Basis des fallbasierten Schließens ermittelt und an mobile Geräte zum Zweck der Personalisierung weiterleitet. Das im Rahmen des *AmbieSense* Projekts entstandene System unterteilt den Prozess der Benutzermodellierung in zwei Teilprozesse, die aufgrund ihrer unterschiedlichen Anforderungen bzgl. des Speicherbedarfs zum einen innerhalb des mobilen Geräts und zum anderen in der das Gerät umgebenden Infrastruktur ausgeführt werden. Auf den mobilen Geräten werden über eine kontextsensitive Middleware die aktuellen Kontexte ermittelt und an ebenfalls auf den mobilen Geräten installierte Komponenten bzw. *Agenten* weitergeleitet. Jeder Agent deckt jeweils einen speziellen Aufgabenbereich des Prozesses der Personalisierung ab. Beispielsweise ermittelt eine Komponente die Ziele des Benutzers, während eine andere die zu personalisierenden Applikationen implementiert. Die für das fallbasierte Schließen zuständige Komponente wird laufend über den aktuellen Kontext informiert. Dieser wird dann mit den historischen Kontexten verglichen und die zu dem aktuellen Kontext passenden Ziele des Benutzers, an die Komponente für die Steuerung der Applikationen übermittelt. Daraufhin wird die am besten geeignete Applikation, dem Ziel des Benutzers nach, entsprechend adaptiert. Die Methode des fallbasierten Schließens bedingt jedoch das Vorhandensein einer umfangreichen Datenbank, in der jeder Fall bzw. jede Erfahrung, bestehend aus dem Kontext (Problembeschreibung) und der Parametrisierung des Systems (Lösungsstrategie), einzeln abgespeichert wird. Im AmbieSense Projekt werden daher alle Erfahrungen auf Seiten der Infrastruktur zunächst zwischengespeichert, dann generalisiert und später mit den mobilen Geräten synchronisiert. Im Gegensatz zur reinen Form des fallbasierten Schließens werden in der Infrastruktur bei dieser Variante stereotypische Fälle erzeugt. Damit ist es nicht nötig, die Vergleichsfunktion auf alle Fälle einzeln anzuwenden, sondern auf den Vergleich mit den Stereotypen zu beschränken. Verwendet wurde ein auf dieser Methode basierendes System im Umfeld eines Flughafens. Die vorgestellte Applikation kann anhand der Zeit, der Dauer seit dem letzten Bezahlen mit Kreditkarte, dem Ort innerhalb des Flughafens und dem Geschmack des Be-

nutzers ermitteln, ob eine proaktive Benachrichtigung des Benutzers, in Anbetracht der in der näheren Umgebung liegenden Gastronomien, angebracht wäre.

Ein intelligentes Haus, in dem sich beispielsweise die Klimaanlage je nach Tageszeit an die besonderen Bedürfnisse der Bewohner anpasst oder das Fernsehgerät automatisch das von den Bewohnern bevorzugte Programm wählt, wird von Ma in [Ma u. a., 2005] vorgestellt. Das System implementiert ebenfalls das fallbasierte Schließen, um auf individuelle Einstellungen der Haushaltsgeräte zu schließen. Ma widmet sich insbesondere dem Problem, eine geeignete Distanzfunktion zu definieren, die den Vergleich zweier Kontexte auf unterschiedlichen Hierarchieebenen ausführt. Ohne eine solche Distanzfunktion ist es nicht möglich, die Ähnlichkeit zweier Problemstellungen zu quantifizieren. Die Merkmale zweier Kontexte (Problemstellungen) werden dabei zunächst paarweise miteinander verglichen. Jedes Ergebnis fließt dann mit einer bestimmten Gewichtung in den paarweisen Vergleich mehrerer Gruppen von Merkmalen ein. Die gesamten Kontexte werden dann auf Basis dieser lokal begrenzten Gegenüberstellungen global miteinander verglichen. Da jedoch die Relevanz eines Merkmals innerhalb der vorgestellten Distanzfunktion fest definiert ist, werden individuelle Unterschiede bei den Vergleichen nicht berücksichtigt. Ma stellt außerdem auf konzeptioneller Ebene eine allgemeine Architektur eines kontextsensitiven Systems vor, das auf Basis des fallbasierten Schließens der kontextsensitiven Personalisierung dient.

In [Sadeh u. a., 2005] stellt Sadeh ein personalisiertes Filtersystem für Nachrichten vor, welches abhängig vom Kontext bestimmt, welche Nachrichten dem Benutzer direkt präsentiert werden. Der Benutzer kann seine Präferenzen bzgl. der Filterung entweder explizit dem Filtersystem mitteilen oder es dem System überlassen, auf Basis historischer Interaktionen auf individuelle Filter zu schließen. Beispiele der im Kontext zu berücksichtigenden Aspekte der Situation sind die Erreichbarkeit oder die Position des Benutzers. Das Filtersystem setzt für die Personalisierung die im *MyCampus* Projekt entwickelte Semantic-Web-Infrastruktur ein, in der sowohl der Kontext ermittelt, als auch das Benutzermodell erstellt und verwaltet wird. Auch hier wird das fallbasierte Schließen eingesetzt, damit das Filtersystem auf jede eintreffende Nachricht individuell reagieren kann. Sadeh verwendet für die Gegenüberstellung der Kontexte einen speziellen *Nearest-Neighborhood*-Algorithmus, der neben dem Kontext auch einige Bestandteile der Nachrichten miteinander vergleicht. Eine im Rahmen des MyCampus Projekts durchgeführte Evaluation ergab, dass die Genauigkeit der Personalisierung mit Hilfe des fallbasierten Schließens bei über 80 % lag, während die explizite Eingabe der Präferenzen nur eine Genauigkeit von knapp über 50 % zur Folge hatte.

Coutand beschreibt in [Coutand, 2008] ein System zur kontextsensitiven Personalisierung, das die Vorgänge der Erkennung und Verwaltung persönlicher Präferenzen logisch von der Anwendung der Personalisierung trennt. Die Überwachung des Kontextes und die Benutzermodellierung werden dabei in einer separaten Systemkomponente ausgeführt, die von jeder Applikation für die Personalisierung genutzt werden kann. Hat ein Entwickler den Wunsch, die Benutzerschnittstelle einer bestimmten Applikation kontextsensitiv zu personalisieren, so kann er die Applikation mit Hilfe von Vorlagen für eine Personalisierung bei der Systemkomponente anmelden. Die Vorlagen enthalten dabei spezifische Aspekte der Applikation und die dazugehörige Parametrisierung der Personalisierung. Auf Basis dieser Vorlagen führt die Systemkomponente dann die Suche nach kontextsensitiven Präferenzen aus und benachrichtigt die jeweilige Applikation, wenn eine Adaption angewendet werden soll. Diese Trennung dient zum einen der Zentralisierung der Benutzermodellierung zur Verbesserung des Datenschutzes und zum anderen der Wiederverwendung von kontextsensitiven Präferenzen über die Grenzen einer Applikationen hinaus. Die vorgestellte System-

komponente implementiert dabei ebenfalls die Methode des fallbasierten Schließens. In der dazugehörigen Distanzfunktion zum Vergleich zweier Fälle werden im Gegensatz zu den oben beschriebenen Ansätzen auch spezifische Vorlieben des Benutzers berücksichtigt. So ist es möglich, dass sich zwei konkrete Kontexte aus Sicht des Benutzers gleichen, während ein anderer Benutzer beide Kontexte unterscheidet. Während der Ausführung der Distanzfunktion werden alle historischen Fälle zunächst nach der zu personalisierenden Applikation klassifiziert und anhand der Kontexte agglomerativ in Gruppen eingeordnet. Danach werden alle Gruppen automatisch benannt und letztendlich mit dem neuen Fall verglichen. Dieser Prozess wird dabei immer dann ausgeführt, wenn eine für die kontextsensitive Personalisierung angemeldete Applikation ein zu untersuchendes Interaktionsverhalten an die Systemkomponente meldet. Da also während der Ausführung der Distanzfunktion alle bekannten Fälle aufwendig gruppiert werden müssen, steht und fällt die Performance bzw. Reaktionsgeschwindigkeit des Systems mit der Aufwandsklasse des Gruppierungsalgorithmus.

Flanagan stellt zur automatischen Erkennung individueller Zustände des Kontexts in [Flanagan, 2005] eine Modifikation des in [Flanagan, 2003] beschriebenen SCM-Algorithmus (Symbol String Clustering Map) vor. Diese wird von Vetek [Vetek u. a., 2009] dazu verwendet, das Menü eines mobilen Telefons kontextsensitiv zu personalisieren. Ähnlich wie in der Distanzfunktion von Coutand [Coutand, 2008], besteht auch hier der Grundgedanke des Algorithmus darin, eine Menge konkreter Kontexte, die direkt durch Abtastung der Sensoren ermittelt wurden, unmittelbar miteinander zu vergleichen und zu gruppieren, um daraus abstraktere Kontexte, sogenannte *contextual states*, zu erstellen. Diese Gruppen konkreter Kontexte, können dann laut Flanagan mit den Interaktionen des Benutzers direkt korreliert werden. Da dieser Prozess automatisch, also ohne ein Feedback durch den Benutzer auskommt, wird er wie alle Clusteringalgorithmen dem unüberwachten Gruppieren zugeordnet. Der Algorithmus arbeitet jedoch ausschließlich auf sogenannten symbolischen Strings, die den Kontext als n-Tupel von Symbolen beschreiben. Wird jedoch ein Aspekt der Umgebung betrachtet, dessen Wertebereich beispielsweise dem der reellen Zahlen entspricht, so ist es notwendig, diesen Wertebereich vorher zu diskretisieren und jedes Intervall durch ein eindeutiges Symbol zu ersetzen. In der *SmartActions* Applikation von Vetek besteht der Kontext daher aus einem symbolischen String, in dem die Position des mobilen Telefons durch eine eindeutige Zellennummer repräsentiert wird. Moderne mobile Geräte nutzen jedoch das GPS Signal zur Positionsbestimmung, dessen Wertebereich theoretisch dem der reellen Zahlen entspricht.

Cheng [Cheng u. a., 2008] bedient sich ebenfalls eines Algorithmus des unüberwachten Gruppierens. Cheng nutzt in diesem Fall eine Variante des *Co-Clusterings*, um eine Menge einzelner Eingaben im Mobiltelefon (z.B. Anrufe) zusammen mit dem zu diesem Zeitpunkt gültigen Kontext zu gruppieren. Jede resultierende Gruppe ist dann durch einen Schwerpunkt (engl. Centroid) gekennzeichnet, der die gesamte Gruppe repräsentativ vertritt. Finden neue Eingaben statt, so wird der zu diesen Eingaben gehörende Kontext mit dem Kontext jedes Schwerpunktes verglichen. Die Ergebnisse dieser Vergleiche werden dem Grad der Ähnlichkeit nach sortiert und deren Eingaben in der Benutzerschnittstelle präsentiert. Je nach Kontext erscheinen also verschiedene Empfehlungen für oft verwendete Funktionen, die ohne Umwege über das Menü direkt ausgeführt werden können. Sowohl im Ansatz von Flanagan als auch in dem von Cheng muss vor der Ausführung die Zahl der Gruppen bekannt sein, damit die Algorithmen eine gute Performance liefern. Der OPC-Algorithmus (One-Pass Clustering) [Jeong u. a., 2009] von Jeong greift die Idee des Co-Clusterings auf und erweitert diesen um Techniken zur automatischen Schätzung der Anzahl initialer

Gruppen und zugehöriger Centroids. In [Cheng u. a., 2009] vergleicht Cheng den OPC-Algorithmus unter anderem mit dem zuvor beschriebenen Co-Clustering Ansatz und ermitteln anhand von Beispieldaten aus Mobiltelefonen, dass der OPC-Algorithmus insgesamt eine bessere Performance liefert. Einzelne Eingaben einer Interaktion werden jedoch sowohl von Cheng als auch Jeong unabhängig voneinander betrachtet. Die evtl. kausalen Beziehungen zwischen den Gruppen werden daher leider nicht berücksichtigt.

Inspiriert durch ihren ersten Versuch zur Entwicklung eines kontextsensitiven Mobiltelefons [Siewiorek u. a., 2003], stellt Krause in [Krause u. a., 2006] ein System vor, das mit Hilfe diverser tragbarer Sensoren (z.B. *SenSay* Armband und Mikrofon) kontextsensitive Präferenzen ermittelt. Auf Basis der Präferenzen kann das Mobiltelefon in Abhängigkeit von der Position, der Aktivität, dem Kalender und physikalisch messbaren Werten wie der Hauttemperatur oder dem Hautwiderstand z.B. in den Lautlos-Modus wechseln oder Anrufe vorschlagen. Der in die zwei Stufen „Klassifizieren des Kontextes“ und „Erlernen der Präferenzen“ gegliederte Prozess bedarf genauso wie die oben beschriebenen Ansätze keinerlei Überwachung durch den Benutzer. Während der Klassifizierung des Kontextes wird zunächst eine Self-Organizing-Map mit den Sensordaten trainiert und deren Teile anschließend durch einen zu den partitionierenden Verfahren zählenden *K-Means-Clustering*-Algorithmus gruppiert. Ein Markov-Modell erster Ordnung dient dann dazu, etwaige Übergänge zwischen zwei Gruppen von Kontexten vorherzusagen. In einer zweiten Stufe wird ein bayessches Netz mit den Kontexten und den Präferenzen des Benutzers trainiert und die gemeinsame Verteilung aller Variablen ermittelt.

Aztiria stellt in [Aztiria u. a., 2009b, 2010b] ein allgemeines Verfahren zur automatischen Erkennung regelmäßig stattfindender Sequenzen von Interaktionen vor. Diese sequentiellen Muster kontextabhängiger Interaktionen können dann in den von Aztiria vorgestellten Szenarien innerhalb einer intelligenten Umgebung dazu verwendet werden, abhängig vom Kontext automatisch die Beleuchtung, das Radio oder den Heißwasserbereiter an- bzw. abzuschalten. Das Verfahren beruht dabei auf dem in [Aztiria u. a., 2009a] beschriebenen A_{PUBS}-Algorithmus, der temporale Verhaltensmuster aus Sequenzen von Ereignissen extrahiert. In einem ersten Schritt findet dazu eine Abstraktion der Interaktionen mit Hilfe vordefinierter Aktionsvorlagen statt. Auf Basis dieser vom Anwendungsgebiet abhängigen Aktionsvorlagen werden spezielle Sequenzen erkannt und zu komplexen Ereignissen verarbeitet. Beispielsweise kann aus den Ereignissen „Licht an in Küche“ und „Tasse wird Schrank entnommen“ das komplexe Ereignis „Kaffee wird zubereitet “ automatisiert konstruiert werden. Für die Suche nach regelmäßig stattfindenden Sequenzen von Aktivitäten, die sowohl aus einfachen wie auch komplexen Ereignissen bestehen können, verwendet der A_{PUBS}-Algorithmus den in [Agrawal u. Srikant, 1995] beschriebenen erweiterten Apriori-Algorithmus. Mit Hilfe von Algorithmen zur Erkennung von Arbeitsabläufen [van der Aalst u. a., 2004] werden dann die Topologien aller häufigen Sequenzen von Ereignissen generiert. Eine Topologie bündelt dabei mehrere typische Sequenzen von Ereignissen, die sich zwar in der Reihenfolge unterscheiden, aber letztendlich der gleichen Aktivität zugeordnet werden können. Der letzte Schritt besteht darin, durch Klassifikation der verschiedenen Muster die dazugehörigen speziellen Bedingungen bzgl. der Zeit und des Kontextes zu bestimmen. Werden sequentielle Muster durch das System ermittelt, so kann der Benutzer diese mit Hilfe einer sprachgesteuerten Benutzerschnittstelle „einsehen“ und gegebenenfalls bestätigen oder bearbeiten.

Spezielle Methoden

Im automotiven Bereich konzentriert sich die Anwendung kontextsensitiver Personalisierung auf Funktionalitäten wie die Vorhersage des nächsten Fahrmanövers, der potentiellen Route oder des Fahrziels. Implementiert ein lernender Assistent beispielsweise die Vorhersage des nächsten Fahrmanövers, so könnte dieser zum richtigen Zeitpunkt automatisch den Blinker setzen. Ähnliche Szenarien ergeben sich für mobile Geräte i. Allg., da typische Bewegungsmuster dazu genutzt werden könnten, um z.B. Applikationen für den öffentlichen Nahverkehr automatisiert zu konfigurieren. Bei dieser Art der kontextsensitiven Personalisierung handelt es sich jedoch um einen Spezialfall, da die zu untersuchenden Interaktionen ausschließlich aus einer Reihe von Positionen bestehen. Die im Zuge der kontextsensitiven Personalisierung zu ermittelnden Korrelationen zwischen häufigen Interaktionen und bestimmten Kontexten sind daher schon implizit vorhanden. Die Aufgaben der lernenden Assistenten während des Reasoning bestehen also darin, auf Basis dieser Sequenzen von Positionen zum einen, die für den Benutzer signifikanten Orte zu identifizieren und zum anderen, ein geeignetes probabilistisches Benutzermodell des Fahrverhaltens zu erstellen. Oftmals werden die Methoden zur Ermittlung der signifikanten Orte der Bildung des probabilistischen Benutzermodells vorangestellt, da sich eine Bearbeitung der Ortsdaten als einfacher erweist, wenn bereits mehrere nah beieinander liegende Positionen zusammengefasst werden.

Die Extraktion individueller Orte oder Regionen aus einer Sequenz von z.B. GPS-Positionen findet daher in den meisten Fällen durch Akkumulation und Aggregation ähnlicher Positionen statt. Die Ansätze aus diesem Forschungsbereich bedienen sich häufig diverser Methoden des zeitbasierten oder dichtebasierten Gruppierens. Kang nutzt in [Kang u. a., 2004] das zeitbasierte Gruppieren, um eine Menge von Positionen auch anhand ihrer Zeitstempel signifikanten Regionen zuzuordnen. Dabei bestimmt die Aufenthaltsdauer an einem Ort maßgeblich die Entscheidung, ob die zu dieser Zeit gespeicherten Positionen zu einer Region zusammengefasst werden. Kang führt an, dass eine solche Methode z.B. dazu genutzt werden könnte, die Lautstärke des Telefons abhängig von der Region zu personalisieren. In [Zhou u. a., 2007] beschreibt Zhou den DJ-Cluster-Discovery-Algorithmus, der eine Menge nah beieinander liegender Positionen erst dann zu einer signifikanten Region zusammenfasst, wenn genug Positionen innerhalb eines bestimmten Radius aufgezeichnet werden. Ashbrook [Ashbrook u. Starner, 2002] stellt eine Methode vor, in der zunächst alle Positionen aus den GPS-Sequenzen gelöscht werden, an denen der Benutzer in Bewegung war. Damit nehmt Ashbrook an, dass ein Ort für einen Benutzer erst dann interessant ist, wenn er sich länger dort aufhält. Die auf diese Weise gefilterten Sequenzen werden dann mit Hilfe des K-Means-Clusterings gruppiert. Dabei werden iterativ fiktive Positionen interpoliert, die den Mittelwert zwischen nah beieinander liegenden Positionen darstellen. Abschließend werden alle echten GPS-Positionen, die sich innerhalb des Radius einer interpolierten Position befinden, zu einer Gruppe bzw. Region zusammengefasst. Die signifikanten Orte bzw. Regionen werden dann als Zustände eines Markov-Modells interpretiert, das nach einer Lernphase die Vorhersage der als nächstes zu betretenden Region erleichtert. Zwei Personen, die sich bekannt sind und sich als nächstes Ziel die gleiche Region teilen, können laut Ashbrook auf Basis dieser Methode über deren potentielles Zusammentreffen frühzeitig informiert werden.

In [Marmasse u. Schmandt, 2002] verwendet Marmasse die drei verschiedenen Techniken Bayessche Klassifizierung, Vergleich von Histogrammen und Hidden-Markov-Modellierung, um die individuellen Routen ausschließlich auf Basis der Positionen und der

zugehörigen Zeitstempel vorherzusagen. Ebenfalls Teil des von Marmasse beschriebenen ortsabhängigen Benutzermodells sind dabei die signifikanten Orte und Regionen. Im Gegensatz zu den zuvor beschriebenen Ansätzen, signalisiert in diesem Fall das aufgrund des Betretens eines Gebäudes abbrechende GPS-Signal einen möglicherweise signifikanten Ort. Auf Anfrage des Systems bestätigt dann der Benutzer die Relevanz eines Ortes, indem er diesem einen Namen gibt. Sowohl die Erinnerung an ortsabhängige Aufgaben als auch die Bereitstellung ortsabhängiger Internetinhalte werden von Marmasse als potentielle Szenarien genannt.

Torkkola beschreibt in [Torkkola u. a., 2007] ein System, welches den Fahrer mittels proaktiver Benachrichtigungen bzgl. potentieller Verzögerungen auf der vom System vorhergesagten Route informiert. Torkkola interpretiert dabei eine individuelle Route als eine Sequenz von GPS-Positionen, deren Start- und Endpunkte einem signifikanten Ort entsprechen. Die signifikanten Orte werden dabei mit Hilfe der Summen von gewichteten Normalverteilungen approximiert. Anschließend vergleicht die Methode alle GPS-Positionen paarweise und erhält dadurch Aufschluss darüber, welche Routen sich ähneln. Letztendlich wird die Häufigkeit aller Routen zwischen zwei festen signifikanten Orten bestimmt und zur Vorhersage der Route genutzt.

Krumm nutzt in [Krumm, 2008] ähnlich wie Ashbrook ebenfalls Markov-Modelle zur Vorhersage. Während Ashbrook jedem Zustand des Markov-Modells einen Aufenthalt an einem bestimmten signifikanten Ort zuweist, wird im Modell von Krumm jedes Straßensegment zwischen zwei Kreuzungen durch einen Zustand repräsentiert. Um außerdem die Genauigkeit der Vorhersage des nächsten Fahrmanövers zu verbessern, setzt Krumm auch Markov-Modelle höherer Ordnung ein. Dadurch ist es möglich, die Vorhersage nicht ausschließlich vom aktuellen Zustand abhängig zu machen, sondern auch vorherige Fahrmanöver zu berücksichtigen. Krumm zählt diverse Anwendungen auf, die er in antizipierte Fahrerwarnungen, Fahrerinformationen und automatisches Fahrzeugverhalten gliedert. Darunter befinden sich Szenarien wie eine automatische Scheinwerferausrichtung oder die Verdunkelung der Frontscheibe. In [Doshi u. Trivedi, 2011] werden zahllose weitere Arbeiten vorgestellt, die auf Techniken wie Bayesschen Netzen, Neuronalen Netzen oder *Support-Vector-Maschinen* basieren.

3.1.3 Anwendungen kontextsensitiver Anpassungen

Einige der oben beschriebenen Methoden wurden insbesondere zum Einsatz in mobilen Telefonen bzw. Smartphones konzipiert [Flanagan, 2005; Cheng u. a., 2008; Krause u. a., 2006; Siewiorek u. a., 2003; Coutand, 2008]. Die Anpassungen konzentrieren sich in diesen Fällen entweder auf kontextsensitive Empfehlungen zur Bedienung (z.B. SMS an bestimmte Personen) oder automatische Konfigurationen von Einstellungen (z.B. bevorzugte Lautstärkereglung). Böhmer stellt in [Böhmer u. Bauer, 2010] ebenfalls ein Szenario kontextsensitiver Personalisierung für Smartphones vor. Darin beschreibt er eine Benutzerschnittstelle, deren Icons sich in Abhängigkeit von der kontextsensitiven Nutzung der zugehörigen Applikationen restrukturieren. Motiviert wird diese Anwendung durch die Annahme, dass bestimmte Applikationen in einem Kontext häufiger genutzt werden als andere und daher möglichst weit oben im Menü erscheinen sollen. Bestätigt wurde diese Annahme anhand einer durch Befragung ausgeführten Evaluation in verschiedenen Umfeldern wie z.B. einem Flughafen oder einem Einkaufszentrum. Dabei ließ sich mit Hilfe eines De-

monstrators zeigen, dass die Teilnehmer der Studie ein kontextabhängiges Verhalten bei der Anordnung der Applikationen auf dem Smartphone zeigten. Ein weiteres sehr beliebtes Anwendungsgebiet stellt das intelligente Gebäude dar [Ma u. a., 2005; Aztiria u. a., 2009b, 2010b; Das u. a., 2002]. Das nennt in [Das u. a., 2002] dabei ähnliche Anpassungen wie sie Ma [Ma u. a., 2005] beschreibt. Das fügt außerdem noch Szenarien wie das automatische Anschalten von Fernsehgeräten und Einstellen der bevorzugten Sendungen, die automatische Konfiguration der Gartenanlage und das An- bzw. Abschalten der Dusche hinzu.

Im Gegensatz zu Smartphones und Tablets findet nach dem aktuellen Stand der Technik die kontextsensitive Personalisierung bisher keine Anwendung in automotiven Benutzerschnittstellen, wenngleich bis dato diverse Anwendungsszenarien [Krumm, 2008] im Bereich sicherheitsrelevanter Funktionen vorgestellt wurden. Die Forschung konzentriert sich im automotiven Bereich insbesondere auf kontextsensitive Benutzerschnittstellen, die den Informationsfluss zwischen dem In-Car-Infotainment-System und dem Benutzer in Abhängigkeit von der kognitiven Beanspruchung des Fahrers regulieren. Z.B. wird in [Piechulla u. a., 2003] versucht, ankommende Anrufe im Fall eines hohen Verkehrsaufkommens und kritischer Fahreigenschaften automatisch an den Anrufbeantworter weiterzuleiten. Lavie verfolgt in [Lavie u. Meyer, 2010] einen ähnliches Verfahren, bei dem sie Anrufe abhängig von der kognitiven Beanspruchung pausieren, wiederaufnehmen oder abbrechen lässt. Da außerdem während des Verfahrens ein Benutzermodell generiert wird, um automatisiert geeignete Text-Antwortnachrichten zu erstellen, handelt es sich hierbei um eine Benutzerschnittstelle eines benutzer- und kontextadaptiven Systems. Das Verfahren von Iqbal [Iqbal u. a., 2011] geht noch einen Schritt weiter und informiert den Fahrer und den Anrufer im Fall einer vom System initiierten Unterbrechung des Telefonats über die Gründe des Pausierens. Darüber hinaus werden im automotiven Bereich, wie schon im letzten Abschnitt beschrieben, Systeme untersucht, die eine Vorhersage der Route ermöglichen. Die benutzer- und kontextadaptive Benutzerschnittstelle des *Adaptive Route Advisors* [Rogers u. a., 2000] erstellt beispielsweise lokalisierte Vorschläge für Routen, die vom Benutzer bevorzugt befahren werden.

3.1.4 Diskussion

In vielen der oben beschriebenen Ansätze werden Korrelationen zwischen einer bestimmten Interaktion, die meist durch ein einziges Ereignis repräsentiert wird, und den konkreten Kontexten ermittelt [Schiaffino u. Amandi, 2006; Tanca u. a., 2011; Mikalsen u. Kofod-Petersen, 2005; Ma u. a., 2005; Sadeh u. a., 2005; Coutand, 2008; Cheng u. a., 2008; Jeong u. a., 2009]. Da in diesen Fällen der Kontext nur die spezielle und zu einem bestimmten Zeitpunkt erfasste Ausprägung aller für den Anwendungsfall relevanter Faktoren der Umwelt widerspiegelt, findet eine potentielle kontextsensitive Personalisierung nur beim Auftreten genau dieser Kontexte statt. Coutand begründet diesen Umstand mit „*... in contextual personalization, the different contexts in which the user might find himself cannot be considered as different states of the same system. Rather, the contexts are independent of and unrelated to one another. In short, the selection of a specific user preference is not influenced by the previous user preference.*“[Coutand, 2008]. Das in der vorliegenden Forschungsarbeit vorgestellte Verfahren gründet jedoch auf der konträren Annahme, dass das Interaktionsverhalten sehr wohl davon abhängen kann, in welchen Kontexten sich der Benutzer zuvor befand oder welche Konfiguration der Benutzer vorher wählte. Eine bestimmte kontextsen-

sitive Personalisierung kann demnach nicht nur aufgrund des Eintretens eines konkreten Kontextes ausgeführt werden, sondern auch durch eine kausale Kette verschiedener Kontexte bedingt sein. Als Beispiel seien hier die in Abschnitt 2.3.4 beschriebenen Anwendungsfälle B und C genannt. In einer Analyse der Anforderungen für eine auf dem fallbasierten Schließen basierenden zukünftigen kontextsensitiven Benutzermodellierung [Zimmermann, 2003], beschreibt Zimmermann die damit einhergehenden Probleme durch „ ... *problem of finding an appropiate representation of time-extended cases and episodes as well as to a definition of a similarity measure for episodes, time and time periods*“. Zimmermann beschreibt neben der Herausforderung, eine geeignete Darstellung einzelner oder Ketten temporaler Fälle (Episoden) zu entwickeln, auch gleichzeitig das Kernproblem der Definition einer geeigneten Distanzfunktion für Episoden. Da zur Vorhersage von Routen oder Fahrmanövern die Verwaltung temporär geordneter Ketten notwendig ist, wurden in diesem Forschungsbereich recht frühzeitig passende Gruppierungmechanismen für Episoden entwickelt. Die vorgestellten Methoden bedienen sich wie oben beschrieben dabei oftmals der Markov-Modelle [Ashbrook u. Starner, 2002; Marmasse u. Schmandt, 2002; Krumm, 2008] oder eigener Darstellungen [Torkkola u. a., 2007], um Episoden in geeigneter Weise zu speichern und zu bearbeiten. Aber erst durch die Verwendung von Markov-Modellen höherer Ordnung gelingt es, Routen ganzheitlich, und nicht nur von Straßenkreuzung zu Straßenkreuzung, zu modellieren. Eine Distanzfunktion zur Ermittlung der Ähnlichkeit zweier Routen wird im Fall der Markov-Modellierung jedoch nicht benötigt, da das Modell durch seinen probabilistischen Charakter bereits mehrere ähnliche Routen in sich vereint. In [Torkkola u. a., 2007] führt Torkkola aufgrund einer eigenen Darstellungsform eine Distanzfunktion ein, die jeweils paarweise signifikante Orte zweier Routen vergleicht, um damit die Ähnlichkeit zu quantifizieren. Diese speziellen Lösungen lassen sich nicht in jedem Fall auf die Analyse von Interaktionen übertragen, da laut Aztiria gilt „*It seems logical that the user most of the times carries out that set of actions in a specific order. Although that order can sometimes be random, ...*“[Aztiria u. a., 2009b]. Zwei häufige Sequenzen von Ereignissen in der Eingabe, die sich nur anhand der Reihenfolge von Ereignissen unterscheiden, lassen sich also nicht zwangsläufig auf verschiedene Interaktionen abbilden. Aztiria nennt als anschauliches Beispiel die Zubereitung eines Kaffees, bei der die Reihenfolge der einzelnen Arbeitsschritte „Kaffee in die Tasse geben“ und „Wasser in die Tasse geben“ auf einer höheren Abstraktionsebene irrelevant ist. In dem Verfahren von Aztiria kommen daher wie oben beschrieben Workflow-Algorithmen zur Anwendung, die allein anhand der Sequenzen von Ereignissen ein Prozessmodell z.B. in Form eines *Petri-Netzes* ermitteln. Aufgrund der hohen Komplexität des von Aztiria vorgestellten A_{PUBS}-Algorithmus, der sich durch das Sequential Pattern Mining, den Workflow-Algorithmen, der Klassifizierung und der schwer zu kontrollierenden Diversität der Ergebnisse ergibt, ist dieses Verfahren für den Einsatz innerhalb eines In-Car-Infotainment-Systems weniger geeignet.

In allen untersuchten Verfahren zur kontextsensitiven Personalisierung wird nicht zwischen den Situationen unterschieden, die zu einer kontextsensitiven Präferenz führen, und denen, in denen personalisiert wird. Im Anwendungsfall B unterscheiden sich diese Situation jedoch voneinander. Hier führt die kausale Beziehung zweier Orte beim „Auto starten“ und beim „Auto abschalten“ zu einer Präferenz, auf Basis derer eine Interaktion („Verkehrsmeldungen anzeigen“) mit nur einem der beiden Orte assoziiert wird. Aufgrund dieser Differenzierung wird in dem hier vorgestellten Verfahren erstmals zwischen den Verhaltensmustern unterschieden, die während der Phase des Reasoning durch ihre signifikante Regelmäßigkeit auffallen und denen, die zur Anwendung der Anpassungen führen.

3.2 Verfahren zur Simulation kontextsensitiven Benutzerverhaltens

Bevor ein System mit kontextsensitiver Personalisierung in den produktiven Einsatz geht, ist es unabdingbar, dessen Funktionsweise durch Tests zu evaluieren. Da das Verhalten eines solchen Systems aber in besonderer Weise durch eine Vielzahl komplexer Umweltfaktoren beeinflusst wird, sind Tests in einer realen Umgebung sowohl mit einem hohen Kosten- und Zeitaufwand verbunden, als auch, im Fall der Überprüfung einer Fehlfunktion, schwer reproduzierbar. Ein geeignetes Mittel, das Testen kontextsensitiver Systeme zu erleichtern, liegt darin, den Kontext und das damit einhergehende Benutzerverhalten zu simulieren. Dabei ist es auf Basis z.B. einer modellbasierten Simulation aller Sensordaten möglich, die Entwicklung des Systemverhaltens über einen langen simulierten Zeitraum und in sehr unterschiedlichen simulierten Umgebungen zu evaluieren. Die dazu notwendigen Werkzeuge werden von Bylund in zwei unterschiedliche Kategorien eingeordnet [Bylund u. Espinoza, 2002]. Während eine Art von Werkzeugen dazu dient, Sensordaten entweder auf Basis von Modellen der Umwelt zu produzieren oder auf Grundlage aufgezeichneter realer Sensordaten zu simulieren, ist es beim sogenannten interaktiven Test aus Sicht der Entwickler möglich, sich in einer virtuellen Umgebung in Echtzeit zu bewegen und die Anpassungen des Systems direkt zu erfahren. Letztere Variante dient aufgrund der Simulation einer semirealistischen Umgebung oftmals auch zur anschaulichen Demonstration kontextsensitiver Benutzerschnittstellen. Im Nachfolgenden werden zunächst Verfahren der ersten Kategorie und darauffolgend auch interaktive Ansätze vorgestellt und erörtert.

3.2.1 Ausführbare Simulationen mit simulierter Zeit

Sanmugalingam widmet sich in [Sanmugalingam u. Coulouris, 2002] insbesondere der Simulation von Positionen mehrerer Menschen in einer virtuellen Welt. Der vorgestellte Ansatz basiert auf der Idee, die Bewegungen von Menschen durch ein prozessorientiertes, hybrides System zu modellieren. Die Menschen können sich demnach in einem kontinuierlichen Raum bewegen und zu diskreten Zeitpunkten beispielsweise die Geschwindigkeit wechseln. Damit bekannte Lösungen aus der Regelungstheorie, wie die Modellierung eines Tempomaten, auch auf die Simulation menschlicher Bewegungen angewendet werden können, modelliert Sanmugalingam Menschen als autonome Fahrzeuge. Das Verfahren wurde in erster Linie dazu entwickelt, das Verhalten ortsbasierter Systeme wie dem *QoS-DREAM* Middleware Projekt zu untersuchen. Moral nutzt in [Morla u. Davies, 2004] ebenfalls einen eigens implementierten Positionssimulator für die Generierung von Positionsdaten innerhalb einer Test- und Simulationsumgebung für ortsbasierte Systeme. Die modulare Struktur dieser Test- und Simulationsumgebung erlaubt es darüber hinaus, unterschiedliche Kontextsimulatoren über Webservice-Schnittstellen in die Umgebung zu integrieren. Als Anwendungsbeispiel nennt Moral das Testen einer Jacke, die der Kontrolle aller lebenswichtigen Funktionen des Menschen dient. Dabei gilt es unter anderem, die Reaktion einer solchen Jacke in Bezug auf den zu simulierenden Kontext zu untersuchen.

Das Verfahren von Huebscher [Huebscher u. McCann, 2004] konzentriert sich insbesondere auf die Simulation der von Infrarot-Näherungssensoren, Drucksensoren (z.B. in Stühlen oder Betten) und RFID-Systemen gelieferten Daten. Huebscher kombiniert dazu zwei unterschiedliche Modelle, die zum einen bestimmte Aufgaben eines Menschen (z.B. „zu Bett gehen “ oder „frühstücken “) beschreiben und zum anderen die Ausführung die-

ser Aufgaben bestimmten Regionen innerhalb einer Wohnung zuordnen. Wird laut dem Aufgabenmodell eine bestimmte Aufgabe ausgeführt, so werden alle Sensoren, die sich im unmittelbaren Umkreis der für die Aufgabe vorgesehenen Region befinden, entsprechend simuliert. Wie im Verfahren von Sanmugalingam, werden also auch hier vornehmlich Positionsdaten simuliert. Angewendet wurde die Methode im Zusammenspiel mit dem *TOSSIM* Werkzeug, dass sowohl die stochastische als auch gesteuerte Simulation eines Sensornetzes ermöglicht.

Battestini beschreibt in [Battestini u. Flanagan, 2005] eine Methode zur Modellierung und Simulation allgemein strukturierter Kontextinformationen auf der Basis von Mixture-Modellen. Das zur Modellierung statistischer Verteilungen geeignete Mixture-Modell besteht hierbei aus einer Überlagerung K uni-modaler Wahrscheinlichkeitsverteilungen für alle Kontexte. Somit beschreibt das Mixture-Modell einen Wahrscheinlichkeitsraum mit K Gruppen, bei der jede Gruppe einen separaten Kontext bzw. Contextual State [Greenberg, 2001] repräsentiert. Die Ausführung eines Mixture-Modells ergibt dann Kontexte wie beispielsweise ein 3-Tupel aus dem Ort, der Zeit und der Aktivität. Da sich ein zu untersuchendes Objekt jedoch mit der Zeit in verschiedenen Kontexten befindet, ist es nicht sinnvoll, die Auftrittsswahrscheinlichkeiten für bestimmte Kontexte, wie im klassischen Mixture-Modell vorgegeben, zu fixieren. Battestini führt daher zeitabhängige Mixture-Wahrscheinlichkeiten ein, die der Generierung eines bestimmten Kontextes pro Zeiteinheit eine höhere Wahrscheinlichkeit zuordnen, als der Generierung der restlichen potentiellen Kontexte.

Bei SimuContext [Broens u. van Halteren, 2006] handelt es sich um ein von Broens entwickeltes Framework zur Simulation beliebiger Kontexte. Über eine Konfiguration kann der Entwickler festlegen, welche Quelle zur Generierung des Kontextes zur Laufzeit vom Framework herangezogen wird. Für die in der Quelle produzierten Werte des Kontextes können entweder vordefinierte Funktionen, wie beispielsweise die zufällige oder periodische Generierung, hinzugezogen oder eigene Methoden implementiert werden. Sollen z.B. temporal abhängige Kontexte simuliert werden, so ist es notwendig, die dafür vorgesehene Logik in einer eigenen Quelle zu implementieren. Im Gegensatz zu den vorherigen Verfahren werden die Kontexte also nicht modelliert und damit in einer bestimmten Situation gefolgert, sondern individuell implementiert. Das Framework bietet zur Reproduktion spezieller Ketten von Kontexten außerdem die Möglichkeit, aufgezeichnete Sensordaten als Quelle zu verwenden.

Martin stellt in [Martin u. Nurmi, 2006] einen Simulator für Applikationen vor, welche innerhalb einer Ubiquitous Computing Umgebung getestet werden. Der Simulator besteht grundsätzlich aus drei unterschiedlichen Modellen: Agenten-, Welt- und Kontextmodell. Für jedes zu simulierende Objekt, dessen kontextsensitives Verhalten sich der Untersuchung unterzieht, wird ein eigenes Agentenmodell angelegt. Im Agentenmodell wird dabei auf der Basis eines Zustandsmodells beschrieben, welchen Einfluss der unmittelbare Kontext und weitere relevante Agenten auf das Verhalten des Agenten haben. Ein bestimmter Kontext kann beispielsweise dazu führen, dass der Gemütszustand im Agentenmodell von traurig auf glücklich umgestellt wird. Das Weltmodell dient der Beschreibung der räumlichen Umgebung der Agenten. Aufgebaut wird das Weltmodell mit Hilfe von zweidimensionalen Karten, die zum einen Gebäude oder undurchdringliche Wände markieren und zum anderen interessante Orte spezifizieren. Das Kontextmodell ergänzt dann das Weltmodell um Regionen, in denen der Kontext bestimmte Ausprägungen annimmt. Umgesetzt wird dies ebenfalls durch die Überlagerung von Karten, auf denen die betroffenen Regionen durch geometrische Objekte repräsentiert werden.

Bruneau widmet sich in [Bruneau u. a., 2009] ebenfalls der Entwicklung einer Test- und Simulationsumgebung für Applikationen, die im Sinne des Pervasive Computing sowohl simuliert als auch getestet werden sollen. Eine Beschreibung der Umgebung auf der Basis der Spezifikationssprache *DiaSpec* dient dabei zur Parametrisierung der Test- und Simulationsumgebung. Das Framework generiert mit Hilfe dieser Beschreibung eine Emulationsumgebung für zu testende Applikationen und ein passendes Framework zur Programmierung von Simulationen. Eine von Bruneau entwickelte grafische Benutzeroberfläche wird außerdem zur Konfiguration von Szenarien, zur Beobachtung laufender Simulationen und zum Debuggen genutzt.

Kakousis stellt in [Kakousis u. a., 2010] ein Simulationswerkzeug vor, das auf Basis der Beschreibung eines Szenarios in der Lage ist, konkrete Kontexte zu generieren. Als Anwendungsbeispiel dient die *SatMotion*-Applikation, die mit Hilfe des Simulationswerkzeugs evaluiert werden soll. Die auf einem PDA installierbare SatMotion-Applikation wird von Antenneninstallateuren genutzt, um beispielsweise den nächsten Montageort zu erfragen. Ein passendes Szenario besteht darin, die Applikation in verschiedenen Umgebungen z.B. mit WLAN- oder 3G-Empfang zu testen. Zur Beschreibung eines Szenarios wird nacheinander jedes Ereignis (z.B. Aufruf einer Applikation oder Kontextänderung) mit konkreten Parametern spezifiziert.

3.2.2 Interaktive Simulationen in Echtzeit

Der erste erfolgreiche Versuch, einen Kontext in Echtzeit innerhalb einer virtuellen Umgebung zu generieren, erfolgte durch Bylund [Bylund u. Espinoza, 2002]. Bylund nutzt die Game-Engine eines First-Person-Shooter-Computerspiels, um dem Entwickler eine realitätsnahe Bewegung innerhalb eines räumlich begrenzten Gebietes zu ermöglichen und gleichzeitig in Echtzeit die kontextsensitiven Anpassungen zu erfahren. Der quelloffene Teil der Game-Engine wurde dahingehend verändert, den Zugriff auf Informationen über den Spieler in der virtuellen Umgebung wie die Position, Höhe, Geschwindigkeit, Orientierung, und die Aktivität (z.B. Laufen oder Schwimmen) auch von außerhalb der Anwendung zu ermöglichen. Zur Verarbeitung der diversen Informationen hin zu einem auf einer höheren Abstraktionsebene verwertbaren Kontext verwendeten Bylund das *Context Toolkit* von Salber [Salber u. a., 1999]. Das zum Zweck der Implementierung und Kapselung mehrerer Sensoren vom Context Toolkit zur Verfügung gestellte Konzept der „widgets“, dient dabei als Schnittstelle zwischen dem Computerspiel und dem Context Toolkit. Erfolgreich angewendet wurde dieser Prototyp im Zusammenspiel mit einer früheren Variante von *GeoNotes* [Persson u. a., 2002], das das Hinterlassen von ortsabhängigen Notizen erlaubt. Bylund zeigt damit erstmals den praktischen Nutzen virtueller Umgebungen als Test- und Demonstrationsumgebung für kontextsensitive Personalisierungen.

Barton präsentiert in [Barton u. Vijayaraghavan, 2003] den *UbiWise* Prototypen, der zum Zweck des Testens kontextsensitiver Applikationen die Simulation einer virtuellen Umgebung implementiert. *UbiWise* basiert dabei auf der gleichen Game-Engine, die auch Bylund [Bylund u. Espinoza, 2002] nutzt. Im Gegensatz zu Bylund werden hier jedoch nicht nur die Informationen des virtuellen Spielers verarbeitet, sondern auch Gegenstände in der virtuellen Umgebung simuliert. Dabei konzentriert sich Barton vor allem auf die technische Umsetzung möglicher Interaktionen zwischen dem virtuellen Spieler und der ihn umgebenden virtuellen Objekte. Dazu entwickelt Barton zwei unterschiedliche Sichten auf die

Simulation. In der einen Sicht wird die virtuelle Umgebung dreidimensional dargestellt, während eine zweite Sicht alle relevanten Gegenstände, die auch in der dreidimensionalen Sicht zu sehen sind, zweidimensional auf den Desktop abträgt. Die Trennung zwischen der virtuellen Umgebung und der zweidimensionalen Sicht auf die Gegenstände ergibt sich aus dem pragmatischen Grund, dass sich in der zweidimesionalen Sicht ein Gegenstand wie gewohnt mit der Computermaus bedienen lässt und sich die dreidimensionale Sicht zur realitätsnahen Visualisierung eignet.

Bei *3DSim* [Nazari Shirehjini u. Klar, 2005] handelt es sich um ein Rapid Prototyping Werkzeug für Anwendungen der Ambient Intelligence. Ähnlich wie in [Bylund u. Espinoza, 2002] und [Barton u. Vijayaraghavan, 2003] beschrieben, wird es dem Entwickler auch hier ermöglicht, Applikationen innerhalb einer virtuellen Umgebung zu validieren und zu testen. Dabei stehen insbesondere die adaptiven Benutzerschnittstellen, die Systeme zur Koordination der Ausgabe multimedialer Inhalte, die Einbindungen heterogener Systeme, die Komponenten zur Erkennung von Situationen und die strategischen Komponenten zur Planung im Blickpunkt der Untersuchungen. Zu diesen Zwecken implementiert 3DSim das Environmental Monitoring System (EMS), welches Daten sowohl von virtuellen als auch realen Sensoren entgegennimmt, fusioniert und zu auf höherer Abstraktionsebene nutzbaren Kontexten verarbeitet. In der Standardkonfiguration erlaubt 3DSim dabei die Verarbeitung der Position virtueller Objekte bzw. Avatare, der Lichtverhältnisse und der Zustände virtueller Objekte (z.B. an/aus). Befindet sich 3DSim im Visualisierungsmodus, so können die von realen Sensoren zur Verfügung gestellten Daten mit Hilfe der virtuellen Umgebung visualisiert und damit auf ihre Zuverlässigkeit hin geprüft werden. Im Gegensatz dazu werden im Simulationsmodus, getriggert durch den Benutzer, virtuelle Sensoren mit Hilfe einer zweidimensionalen Benutzeroberfläche und dreidimensionaler Gesten simuliert. Eine genauere Beschreibung der Simulation virtueller Sensoren ist in [Nazari Shirehjini u. Klar, 2005] jedoch nicht gegeben.

McGlinn stellt in [McGlinn u. a., 2010] das Werkzeug *SimCon* vor, dass insbesondere die Evaluation von Applikationen unterstützt, die für die Steuerung intelligenter Gebäude konzipiert sind. Auch hier findet die Evaluation mit Hilfe einer virtuellen Umgebung statt, die genauso wie in [Bylund u. Espinoza, 2002] und in [Barton u. Vijayaraghavan, 2003] durch eine Game-Engine simuliert wird. Im Gegensatz zu den vorherigen Methoden, widmet sich das Verfahren von McGlinn insbesondere der Modellierung der virtuellen Sensoren. Als Modellierungsprache dient in diesem Fall *sensorML*, dass zur systemübergreifenden Beschreibung diverser Eigenschaften von Sensoren entwickelt wurde. Innerhalb des *SimConfig* Werkzeugs werden die mit Hilfe des Industry Foundation Classes (IFC) Standards modellierten Gebäude mit den Modellen der Sensoren kombiniert. Zur Laufzeit werden die von der virtuellen Umgebung erzeugten Positionsdaten dann dem *SimCon* Generator gemeldet, der auf Basis der Gebäude- und Sensormodelle ein „contum“ erzeugt, dass einem Kontext entspricht. Beispielhaft angewendet wurde *SimCon* zur Visualisierung und Simulation von Temperaturdaten in intelligenten Gebäuden.

3.2.3 Diskussion

Während sich die Forschungsansätze aus 3.2.2 also vornehmlich dazu eignen, Tests im kleineren Maßstab in Echtzeit durchzuführen, dienen die Forschungansätze aus 3.2.1 zur Durchführung von Tests im größeren Maßstab sowohl hinsichtlich der zu untersuchenden

Dauer als auch der Struktur der zu simulierenden Kontexte. Letztere Forschungsansätze beschreiben Testumgebungen für kontextsensitive Applikationen [Morla u. Davies, 2004; Broens u. van Halteren, 2006; Bruneau u. a., 2009], die automatische und modellbasierte Generierung bestimmter Kontextinformationen [Huebscher u. McCann, 2004; Battestini u. Flanagan, 2005; Martin u. Nurmi, 2006; Kakousis u. a., 2010] oder die Simulation der Positionen großer Mengen von zu untersuchenden Objekten [Sanmugalingam u. Coulouris, 2002]. Die Testumgebungen stellen zwar die applikationsunabhängigen Infrastrukturen zum Testen von kontextsensitiven Applikationen zur Verfügung, nicht jedoch die Logik der Generierung der Kontexte. Diese muss für eine spezielle Anwendung im Fall der Positionsdaten eigens implementiert [Morla u. Davies, 2004], durch externe Quellen bereitgestellt [Morla u. Davies, 2004; Broens u. van Halteren, 2006; Bruneau u. a., 2009] oder explizit in Form von Ereignislisten formuliert werden [Kakousis u. a., 2010]. Da sich der Kontext im automotiven Bereich aufgrund laufender Positionswechsel im Millisekundenbereich ständig ändert, ist eine explizite Spezifikation aller Änderungen und Ereignisse - wie in [Kakousis u. a., 2010] beschrieben - nicht praktikabel. Auch eine szenarienspezifische Implementierung ist nicht wünschenswert, da viele unterschiedliche Szenarien mit verschiedenen Parametrisierungen untersucht werden müssen.

Bei der modellbasierten Generierung der Kontexte werden die zu simulierenden Kontexte oftmals ausschließlich in Beziehung zur Position gesetzt [Sanmugalingam u. Coulouris, 2002; Huebscher u. McCann, 2004; Martin u. Nurmi, 2006]. Die Werte bestimmter zu simulierender Sensoren ergeben sich dabei entweder aus einer eindeutigen Abbildung von Benutzeraktionen (wie z.B. „auf einem Stuhl sitzen “ für den Sitzsensor) auf Regionen in einer Wohnung [Huebscher u. McCann, 2004] oder der Grad an Transparenz geometrischer Objekte auf zweidimensionalen Karten [Martin u. Nurmi, 2006]. In beiden Fällen können Korrelationen zwischen der Zeit und einer beliebigen Interaktion nur bedingt modelliert werden. Martin führt daher ein Zustandsmodell ein, welches die Bewegung eines Objekts innerhalb der zu simulierenden Welt spezifiziert. Trotzdem bleibt die Position der zentrale Angelpunkt des zu simulierenden Kontextes. Korrelationen zwischen beliebigen Aspekten der Umgebung können somit nicht simuliert werden. Battestini geht dagegen noch einen Schritt weiter und ermöglicht durch kombinierte Wahrscheinlichkeitsverteilungen die Modellierung beliebiger Kontexte und damit beliebiger Korrelationen. Die Möglichkeit einer Einflussnahme auf die Wahrscheinlichkeit bestimmter Kontexte und damit Modellierung temporaler Wechsel zwischen Kontexten wird zwar informell vorgeschlagen, aber nicht konkretisiert.

Die in 3.2.2 vorgestellten Lösungen zur Simulation der Umgebungen in Echtzeit setzten voraus, dass die Umgebung und die Sensoren vollständig modelliert werden müssen und sich das zu untersuchende Objekt wie ein zu Fuß gehender Mensch innerhalb der virtuellen Welt fortbewegt. Werden automotive Applikationen simuliert, die in allen mit Straßen versehenen Gebieten der Welt anwendbar sind, so wäre eine Modellierung der dazu notwendigen Umgebung zum jetzigen Zeitpunkt nur mit einem Aufwand zu bewerkstelligen, der in keinem Verhältnis zum Nutzen der Simulationsumgebung steht. Außerdem müsste die Game-Engine dahingehend erweitert werden, dass die Bewegungsphysik in der virtuellen Welt ordnungsgemäß durch eine Fahrphysik ersetzt wird. Wie trotz des Aufwands dennoch kostengünstig eine realitätsnahe Simulation der Position des Fahrzeugs, der Ausrichtung und der Höhe über den Meeresspiegel im Labor durch die Bereitstellung einer virtuellen Umgebung erzielt werden kann, wird in Unterabschnitt 5.3.2 beschrieben.

Teil II
Verfahren und Implementierung

Omnia, quæ a nobis vere dignoſci poſſunt,
dependent a clara Methodo,
qua diſtinguimus ſimilia a diſſimilibus.

Caroli Linnæi, Genera Plantarum, 1737

All that truly can be known by us
depends on a clear method
by which we distinguish the similar from the dissimilar.

Übersetzung durch Staffan Müller-Wille und Karen Reeds, 2007

Kapitel 4
Kontextsensitive Personalisierung

In diesem Kapitel wird ein neues Verfahren der kontextsensitiven Personalisierung vorgestellt. Dazu wird einleitend auf konzeptioneller Ebene ein Systemdienst beschrieben, welcher den für die kontextsensitive Personalisierung notwendigen Prozess der Verarbeitung kontextspezifischer Benutzerdaten - unabhängig von der zu personalisierenden Anwendung - kapselt. Anschließend werden nacheinander alle Verarbeitungsschritte zur Ermittlung kontextsensitiver Präferenzen und zur Wiedererkennung der Situationen kontextsensitiver Präferenzen im Einzelnen anhand mehrerer Beispiele beschrieben und abschließend bzgl. der Zeit- und Speicherkomplexitäten diskutiert.

4.1 Systemdienst zur kontextsensitiven Personalisierung

Moderne In-Car-Infotainment-Systeme können unter anderem navigieren, TV-Sendungen empfangen und Filme abspielen. Damit unterscheiden sich moderne Bordcomputer aus Sicht der Benutzer nur noch unwesentlich von Heimcomputern. Im Unterschied zu den sowohl auf Hard- als auch Softwareebene erweiterbaren Computern im Consumerbereich, deren breites Funktionsspektrum einen Einsatz in vielen unterschiedlichen Anwendungsgebieten ermöglicht, bleibt das eingebettete System des Bordcomputers aufgrund des sicherheitskritischen Einsatzgebiets auf die in diesem speziellen Anwendungsgebiet notwendigen Funktionalitäten beschränkt. Da es jedoch in Zukunft, auch nach dem Abschluss der Entwicklung eines solchen Systems, möglich sein wird, die Funktionalität des In-Car-Infotainment-Systems während des Betriebs zu erweitern und sich die Hardware kaum noch von der eines Heimcomputers unterscheidet, ähnelt der Aufbau der Betriebssysteme zunehmend denen der Heim- oder Mobilcomputer. Der Aufbau lässt sich daher vereinfacht in die folgenden drei Ebenen aufteilen: *Benutzerschnittstellen-*, *Anwendungs-* und *Systemebene* (siehe Abbildung 4.1).

Auf der Systemebene werden alle Funktionen implementiert, die zum Betrieb der Hardware notwendig sind. Dazu zählen beispielsweise die Verwaltung des Arbeitsspeichers ebenso wie die Ansteuerung diverser Ein- und Ausgabegeräte. Außerdem dient die Systemebene den Applikationen auf der Anwendungsebene als Ausführungsumgebung. Werden auf der Anwendungsebene spezielle Ressourcen wie z.B. die aktuelle Position oder die Außentemperatur des Fahrzeugs benötigt, um die Applikationen des Navigationssystems und der Wettervorhersage auszuführen, so werden diese Ressourcen auf der Systemebene generiert und der Anwendungsebene zur Verfügung gestellt. Somit handelt es sich bei der

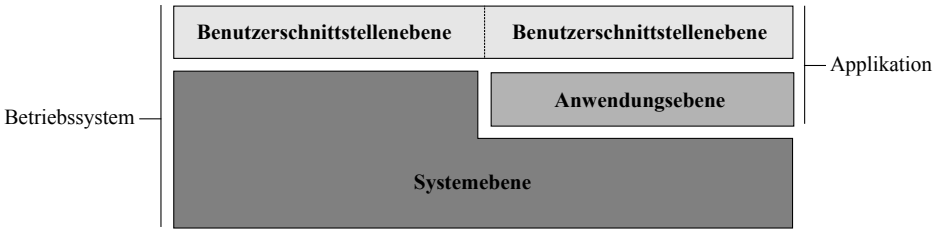


Abb. 4.1 Ebenendarstellung des Betriebssystems eines In-Car-Infotainment-Systems.

Systemebene um die Schnittstelle zwischen den Applikationen und der zur Verfügung stehenden Hardware und den Sensoren. Die Systemebene und die Anwendungsebene verfügen zwar jeweils über eigene Benutzerschnittstellenebenen, nutzen jedoch alle Modalitäten der Ein- und Ausgabe gemeinsam. Abbildung 4.2 zeigt dazu beispielhaft die auf dem Bildschirm in der Mittelkonsole angezeigte grafische Benutzerschnittstelle des Systems und der Applikation des Navigationssystems.



Abb. 4.2 Eine gemeinsame Nutzung einer Ausgabemodalität am Beispiel der Ausführung des Navigationssystems. Im oberen Teil werden die Status- und Menüleiste angezeigt, während der untere Teil aus der Kartendarstellung und dem Applikationsmenü besteht.

Während sich die grafischen Ausgaben der zur Systemebene gehörenden Benutzerschnittstellenebene während der Laufzeit nur in geringem Maß ändern und lediglich als Rahmen für die grafischen Ausgaben der Applikationen dienen, verhält sich die Benutzerschnittstelle der Anwendungsebene von Applikation zu Applikation unterschiedlich. Ergeben sich nun spezielle Bedienmuster bzgl. einer bestimmten Applikation, so führt dies aufgrund des individuellen Charakters der dazugehörigen Benutzerschnittstelle ausschließlich zu Anpassungen innerhalb der Applikation und deren Benutzerschnittstelle. In heutigen Systemen findet der Prozess der Personalisierung daher vornehmlich innerhalb der Anwendungsebene statt. Ermittelt eine Applikation etwaige benutzerspezifische Präferenzen, so können diese mangels einer auf Systemebene angesiedelten zentralen Verwaltung der Präferenzen und eines applikationsunabhängigen Benutzermodells daher in keiner anderen Applikation wiederverwendet werden.

Automotive Benutzerschnittstellen verfügen jedoch abhängig von der Marke und der Modellreihe vornehmlich über ein einheitliches *Look&Feel* der Bedienung. Der grafische Aufbau sowie die Dialogstruktur der Benutzerschnittstelle verschiedener Applikationen ähneln sich derart, dass Personalisierungen nicht nur innerhalb einer Anwendung, sondern auch ap-

plikationsübergreifend von Nutzen sein könnten. Wählt der Benutzer beispielsweise innerhalb der Radioapplikation immer ein von der Tageszeit und dem Gemütszustand abhängiges Musikgenre, so könnte diese kontextsensitive Präferenz mit Hilfe einer zentralen Verwaltung an eine Applikation zur Verwaltung der Musiksammlung weitergereicht werden. Diese Applikation könnte dann zeitabhängig sortierte Listen der Musiksammlung erstellen. Eine systemweit einheitliche Benutzermodellierung ermöglicht es sogar, die Benutzerschnittstellen der Systemebene um kontextsensitive Bestandteile zu erweitern. Als Beispiel seien hier die im Unterabschnitt 2.3.4 vorgestellten kontextsensitiv personalisierten Schnellstartbuttons und die kontextsensitiv personalisierte Automatisierung genannt. Coutand nennt in [Coutand, 2008] den zentralisierten Schutz der personenbezogenen Daten als weiteren Vorteil einer applikationsübergreifenden Benutzermodellierung. Sicherheitskritische Benutzerinformationen wie z.B. die Fahrzeugposition oder die Anzahl der Fahrzeuginsassen werden zwar während der Benutzermodellierung auf Systemebene ausgewertet, aber nicht zur Ausführung der Personalisierung auf Anwendungsebene benötigt. Eine unsachgemäße Weitergabe dieser kontextuellen Informationen an Applikationen von Fremdanbietern wird damit ausgeschlossen. Da sich der mit einem hohen Ressourcenverbrauch verbundene Prozess der Benutzermodellierung damit auch vereinfacht auf vertrauenswürdige externe Rechnerkapazitäten auslagern lässt, ist eine kontextsensitive Personalisierung auch innerhalb des Bereichs eingebetteter Systeme möglich.

Die systemweite Verarbeitung von Benutzerinformationen findet aufgrund dieser Eigenschaften in dem hier vorgestellten Verfahren innerhalb eines *Systemdienstes* auf der Systemebene statt. Dieser untersucht die von logischen, virtuellen und physikalischen Sensoren erzeugten *Eingabeereignisse* gemeinsam auf das Vorhandensein möglicher kontextsensitiver Präferenzen und benachrichtigt in der geeigneten Situation die dem Systemdienst zuvor gemeldeten *Aktuatoren* durch entsprechende *Ausgabeereignisse* über die anzuwendenden Anpassungen. Aktuatoren können sich dabei genauso wie Sensoren auf allen Ebenen im Betriebssystem befinden. Ein Aktuator in der Systemebene könnte beispielsweise die interne Priorisierung auszuführender Funktionen mit Hilfe der kontextsensitiven Präferenzen verbessern während ein Aktuator auf Anwendungsebene die oben als Beispiel genannte Anfertigung einer zeitabhängig sortierten Liste durchführen könnte.

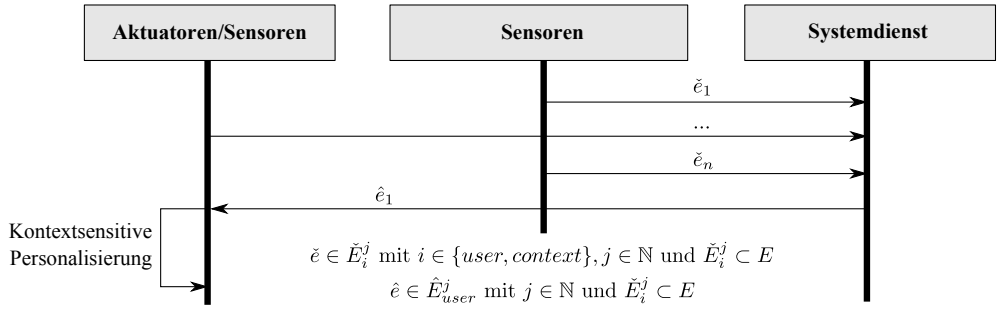


Abb. 4.3 Asynchrone Kommunikation des Systemdienstes zur kontextsensitiven Personalisierung.

Die Identifikation des Benutzers, z.B. über eine Analyse der individuellen Sitzstellungen [Riener u. Ferscha, 2008], des charakteristischen Fahrverhaltens [Miyajima u. a., 2007] oder eine Stimmenerkennung [Bernsen u. Dybkjaer, 2005], wird bei diesem Verfahren als gegeben vorausgesetzt. Jedes Eingabeereignis enthält daher implizit einen eindeutigen Identifikator, der in nachfolgenden Betrachtungen jedoch nicht weiter behandelt wird.

Die Schnittstelle des Systemdienstes verfügt also über einen Eingang für Eingabeereignisse und einen Ausgang für Ausgabeereignisse. Sei nun durch F_{in} die Menge aller Arten von Vorfällen gegeben, die sowohl innerhalb als auch außerhalb des Systemdienstes zu einem Ereignis führen können, so wird die Menge aller Ereignisse innerhalb des Systemdienstes wie folgt definiert:

$$E = \{(\lambda, (p_1, ..., p_n)) | \lambda \in F_{in} \wedge n \in \mathbb{N} \wedge p_x \in \{(q_{name}, q_{value}) | q_{name} \in F_{na} \wedge q_{value} \in F_{va}\}\}$$

Jedes Ereignis verfügt demnach über eine eindeutige Kennzeichnung des Typs λ und eine beliebig lange Liste von *Ereignisparametern* $(p_1, ..., p_n)$. Die Ereignisparameter enthalten zusätzliche Informationen über den zu repräsentierenden Vorfall und bestehen im Einzelnen wiederum aus einer eindeutigen Kennzeichnung der Art des Ereignisparameters q_{name} (im Nachfolgenden als *Ereignisparametername* bezeichnet) und dem zugehörigen *Ereignisparameterwert* q_{value}. Die Wertebereiche der Ereignisparameter ergeben sich aus der Menge aller möglichen Ereignisparameternamen F_{na} und der Menge aller Ausprägungen F_{va}. Als Beispiel sei ein Eingabeereignis eines GPS-Sensors

$$\check{e}_3 = (Position, ((Längengrad, 13.32180), (Breitengrad, 52.51260)))$$

gegeben, das als Parameter den Längen- und Breitengrad der Fahrzeugposition nach dem geodätischen Referenzsystem WGS84 enthält. Eine vom Systemdienst an einen Aktuator gerichtete Benachrichtigung bzw. Ausgabeereignis

$$\hat{e}_{134} = (Browser, (URL, www.google.de))$$

könnte beispielsweise in diesem Fall den Browser dazu veranlassen, die entsprechende URL aufzurufen. Das Ereignis enthält jedoch keine Informationen über die dazugehörige Situation. Das Ausgabeereignis muss dem Aktuator also genau zu dem Zeitpunkt zugeführt werden, an dem die Personalisierung stattfinden soll.

Am Eingang werden die Ereignisse aus Sicht des Systemdienstes hinsichtlich ihrer Quellen und ihrer zugehörigen Abstraktionsebene nicht unterschieden. Dies erhöht, wie in den nachfolgenden Abschnitten verdeutlicht wird, die Flexibilität des Systemdienstes bzgl. der Festlegung des Kontextes. Zu jedem Zeitpunkt t liegt dem Systemdienst somit eine bestimmte Eingabesequenz $\check{s} \in \check{S}$ mit

$$\check{S} = \bigcup_{n \in \mathbb{N}} \check{S}_n$$
$$\check{S}_n = \{(\check{e}_1, ..., \check{e}_n) | t_{event}(\check{e}_{i-1}) < t_{event}(\check{e}_i) \wedge \check{e} \in \check{E}_{user} \cup \check{E}_{context}\}$$

vor, die Ereignisse verschiedener Abstraktionsebenen und Quellen enthält. Die Aufgabe des Systemdienstes besteht darin, eine Sequenz von Eingabeereignissen $\check{s}$ auf eine Sequenz von Ausgabeereignissen $\hat{s} \in \hat{S}$ mit

$$\hat{S} = \bigcup_{n \in \mathbb{N}} \hat{S}_n$$
$$\hat{S}_n = \{(\hat{e}_1, ..., \hat{e}_n) | t_{event}(\hat{e}_{i-1}) < t_{event}(\hat{e}_i) \wedge \hat{e} \in \hat{E}_{user}\}$$

abzubilden. Auf Basis welcher konkreter Eingabeereignisse es im Systemdienst zu der Generierung einer Benachrichtigung kommt, die letztendlich zu einer kontextsensitiven Personalisierung führt, hängt im Einzelnen von der internen Struktur des Systemdienstes und

dessen Parametrisierung ab. Im nächsten Abschnitt werden daher die Interna des Systemdienstes vorgestellt und eine beispielhafte Parametrisierung anhand der Anwendungsfälle A bis D aus Unterabschnitt 2.3.4 beschrieben.

4.2 Verfahren der vorlagenbasierten kontextsensitiven Personalisierung

Innerhalb des Systemdienstes werden die Eingabeereignisse auf Basis eines prozessorientierten Verfahrens auf kontextsensitive Präferenzen hin untersucht und die dazugehörigen Ausgabeereignisse in Abhängigkeit von der aktuellen Situation an die Aktuatoren zum Zweck der Personalisierung weitergereicht. Das Verfahren gliedert sich dabei in die vier Prozesse *Ermittlung des Kontextes*, *Ermittlung relevanter Interaktionen*, *Gruppierung der Interaktionen* und *Erkennung der Situationen kontextsensitiver Präferenzen*. Das Verfahren beginnt zunächst mit der Ermittlung des aktuellen Kontextes, welcher sich aus den Eingabeereignissen ableiten lässt. Die beiden nächsten Schritte widmen sich der Suche nach bestimmten Interaktionen, die im Sinne der Parametrisierung des Systemdienstes als relevant eingestuft werden, um diese anschließend anhand der zugehörigen Kontexte zu gruppieren. Hier werden also die kontextsensitiven Präferenzen ermittelt, welche sich direkt aus den Gruppen der gesuchten Interaktionen herleiten lassen. Der letzte Prozess besteht im Wesentlichen daraus, die zu den kontextsensitiven Präferenzen passenden Situationen zu erkennen, um die kontextsensitive Personalisierung im richtigen Moment auszuführen zu können. Das Diagramm in Abbildung 4.4 zeigt eine Übersicht des prozessorientierten Verfahrens.

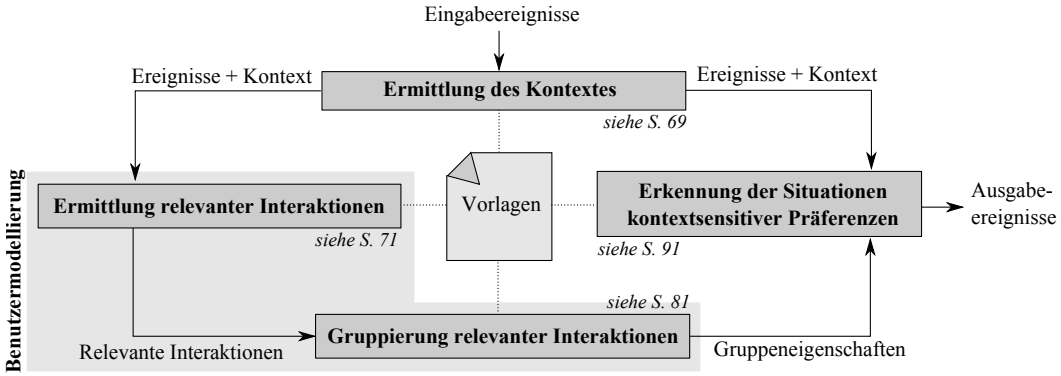


Abb. 4.4 Übersicht des Verfahrens der kontextsensitiven Personalisierung.

Aufgrund der im Unterabschnitt 2.3.1 genannten Anforderungen ist es insbesondere im Bereich automotiver Benutzerschnittstellen notwendig, das Verfahren über eine Parametrisierung explizit zu steuern. Während in einer nicht gesteuerten Variante der kontextsensitiven Personalisierung Bedienmuster zur Laufzeit ermittelt und wiedererkannt werden, die zum Zeitpunkt der Entwicklung unbekannt sind und möglicherweise gegen rechtliche Bestimmungen verstoßen, beschränkt sich die gesteuerte kontextsensitive Personalisierung auf das Wiedererkennen bekannter Bedienmuster. Für eine gesteuerte kontextsensitive Personalisierung bedarf es also im Gegensatz zur nicht gesteuerten Variante einer expliziten Beschreibung des zu erwartenden Benutzerverhaltens. Es wird in diesem Fall also angenommen, dass die Bedienmuster i. Allg. zum Zeitpunkt der Entwicklung bekannt sind.

Der Grad an Flexibilität der zu definierenden Bedienmuster bestimmt dabei, wie sehr die individuellen Vorlieben einzelner Benutzer während der Benutzermodellierung approximiert werden können. Beschreibt ein Bedienmuster beispielsweise den Zusammenhang, dass ein Benutzer innerhalb der Ortschaft *Berlin* bevorzugt *KlassikFM* hört, so ist ein solches Bedienmuster sehr stark auf die jeweilige Person und deren Umgebung zugeschnitten. Diese sogenannten speziellen Bedienmuster sind nicht allgemein anwendbar und können während der Entwicklung auch nicht vorhergesehen werden. Sie müssen daher zur Laufzeit entweder vom Systemdienst ermittelt oder vom Benutzer manuell eingestellt werden. Beschreibt ein Bedienmuster jedoch lediglich die Korrelation, dass beispielsweise Radiosenderwechsel oftmals in Abhängigkeit von der Position des Fahrzeugs ausgeführt werden, so kann ein Systemdienst anhand dieser generischen Bedienmuster zur Laufzeit beliebig viele Varianten spezieller Bedienmuster automatisch herleiten.

In dem hier vorgestellten Verfahren werden generische Bedienmuster daher in separaten *Vorlagen* definiert und dem Systemdienst als Parametrisierung zur Verfügung gestellt. Die Vorlagen enthalten somit - für jedes Bedienmuster separat - alle Parametrisierungen der oben genannten Prozesse. Am Beispiel des generischen Bedienmusters eines ortsabhängigen Radiosenderwechsels ergeben sich in der jeweiligen Vorlage somit auf informeller Ebene folgende vier Parametrisierungen:

- **Ermittlung des Kontextes:** Interpretiere die Position des Fahrzeugs als Kontext. Die Position besteht aus dem Längen- und dem Breitengrad im WGS84 Format.

- **Ermittlung relevanter Interaktionen:** Ermittle relevante Radiosenderwechsel. Ein Radiosenderwechsel ist immer dann relevant, wenn unmittelbar danach - innerhalb von 10 Sekunden - kein weiterer Radiosenderwechsel ausgeführt wurde.

- **Gruppierung relevanter Interaktionen:** Gruppiere alle relevanten Radiosenderwechsel anhand der Namen der Radiosender und der zum Zeitpunkt der Radiosenderwechsel gültigen Positionen des Fahrzeugs. Alle relevanten Radiosenderwechsel einer Gruppe werden mit dem gleichen Radiosendernamen assoziiert. Die Position, an der ein gruppierter Radiosenderwechsel ausgeführt wurde, darf sich nie weiter als 500 m von der Position befinden, an der ein weiterer Radiosenderwechsel aus der gleichen Gruppe erfolgte.

- **Erkennung der Situationen kontextsensitiver Präferenzen:** Erkenne den Zeitpunkt des Betretens einer relevanten Region. Eine relevante Region ergibt sich durch das Aufspannen der Positionen aller sich in einer Gruppe befindlichen aufgezeichneten Interaktionen der relevanten Radiosenderwechsel.

Zur Ausführung der kontextsensitiven Personalisierungen aller im Unterabschnitt 2.3.4 vorgestellten Anwendungsfälle bedarf es also jeweils der Definition einer Vorlage bestehend aus diesen vier Bestandteilen einer Parametrisierung. Abhängig von der technischen Umsetzung können die Vorlagen dem Systemdienst sowohl vor als auch während der Laufzeit übergeben werden. Somit kann ein In-Car-Infotainment-System, welches dieses Verfahren unterstützt, auch noch nach dem Abschluss der Entwicklung mit Hilfe neuer Vorlagen an die aktuellen Gegebenheiten angepasst werden. Wird eine Applikation nachträglich installiert, so kann auch der dadurch induzierte neue Anwendungsfall über eine weitere Vorlage abgebildet werden. Weitere technische Details zur Definition der Vorlagen und deren Verarbeitung innerhalb der prototypischen Implementierung werden im Kapitel 5 erläutert.

Nachfolgend werden nacheinander alle vier Prozesse zusammen mit beispielhaften Parametrisierungen vorgestellt.

4.2.1 Ermittlung des Kontextes

Moderne Fahrzeuge verfügen über eine Vielzahl physikalischer, virtueller und logischer Sensoren, die z.B. das Parken durch Ultraschallsensoren erleichtern oder das Einschalten der Scheibenwischer durch Regensensoren automatisieren. Während einige der Sensoren ausschließlich dazu dienen, einzelne Steuergeräte mit den für die Funktionalität notwendigen Daten zu versorgen, generieren andere Sensoren Daten, deren Ausprägungen einen Einfluss auf die Bedienung des In-Car-Infotainment-Systems ausüben. Dabei erfassen die Sensoren im letzteren Fall entweder interaktionsbedingte oder benutzerunabhängige Daten. Die Ausprägung der Außentemperatur oder Verkehrssituation verhält sich z.B. unabhängig vom Fahrzeug und den Fahrzeuginsassen, während die Position des Fahrzeugs, wenn auch indirekt, sich aus der laufenden Interaktion mit der Benutzerschnittstelle ergibt.

Werden kontextsensitive Präferenzen auf Basis der interaktionsbedingten und benutzerunabhängigen Daten ermittelt und zur Personalisierung verwendet, so zeigt ein solches adaptives System die in Unterabschnitt 2.1.2 von Broy aufgezählten adaptiven Verhaltensweisen. Eine ortsabhängige individuelle Einstellung führt demnach zu einem gesteuerten (kontext)adaptiven Verhalten, da die Fahrzeugposition, wie oben beschrieben, vom Fahrer beeinflusst wird. Ein transparentes (kontext)adaptives Verhalten ist beispielsweise bei einer individuellen Einstellung der Klimaautomatik zu erwarten, während sich ein nicht-transparentes (kontext)adaptives Verhalten durch eine kontextsensitive Präferenz ergibt, deren Kontext sich nicht vom Benutzer beobachten lässt. Wenngleich jede dieser adaptiven Verhaltensweisen durch das hier vorgestellte Verfahren unterstützt wird, so beschränkt sich der zu ermittelnde Kontext im Fall eines automotiven Einsatzes aufgrund des in Unterabschnitt 2.3.1 eingeführten Kriteriums der Vorhersagbarkeit nur auf die vom Benutzer wahrnehmbaren Aspekte der Umgebung. Ein nicht-transparentes kontextsensitives Verhalten wird somit ausgeschlossen. Um dies zu gewährleisten, werden alle Aspekte der Umgebung, die für einen Anwendungsfall als relevant eingestuft werden, in den Vorlagen definiert.

Zur Analyse des aktuellen Kontextes steht dem Systemdienst neben der Parametrisierung in den Vorlagen ausschließlich die Sequenz von Eingabeereignissen $\check{s}$ zur Verfügung. Die Eingabeereignisse können neben den üblicherweise als Kontext interpretierten Positions- oder Zeitdaten auch Informationen enthalten, die sich unmittelbar aus der Interaktion mit der Benutzerschnittstelle ergeben und durch virtuelle oder logische Sensoren erfasst wurden. Somit kann der zu ermittelnde Kontext in dem hier vorgestellten Verfahren flexibel definiert werden und z.B. auch die aktuell eingestellte Sprache oder Lautstärke der Benutzerschnittstelle oder die Anzahl der Fahrzeuginsassen umfassen. Die wesentliche Aufgabe des Prozesses besteht darin, anhand der in den Vorlagen beschriebenen Parametrisierungen und der Sequenz von Eingabeereignissen $\check{s}$, die digitalen Abbilder der Zustände aller Aspekte der Umgebung zu berechnen. Ein solches digitales Abbild kann, wie im Unterabschnitt 2.3.3 kurz vorgestellt, durch das zu einem bestimmten Zeitpunkt durchgeführte Einfrieren der Daten aller Sensoren erfolgen. Lei führt dazu in [Lei u. a., 2002] den Begriff des *transienten Kontextes* ein, der *„...reflects the environment at a single point in time“*. Angelehnt an Lei's Definition eines transienten Kontextes und die in Unterabschnitt 2.1.2 eingeführte

Definition des Kontextes von Dey [Dey, 2001], wird ein solches digitales Abbild der Umgebung in dem hier vorgestellten Verfahren wie folgt definiert:

Definition 4.1 (Kontext). Alle zu einem bestimmten Zeitpunkt erfassten Daten, die zur Charakterisierung der Situation eines zu untersuchenden Objekts beitragen.

Eine Situation wird in diesem Sinne als ein Zustand verstanden, der sich aus den aktuellen und historischen Ausprägungen der Werte aller Sensoren des zu untersuchenden Objekts herleiten lässt. Während es sich beim Kontext also nur um das digitale Abbild der Umgebung zu einem bestimmten Zeitpunkt handelt, wird bei der Situation auch die Historie berücksichtigt. Befindet sich Frau Müller in Anwendungsfall C beispielsweise - auf ihre Tochter wartend - am Bahnhof, so wird die aktuelle Position des Fahrzeugs als Kontext interpretiert. Die Situation Frau Müllers kennzeichnet sich jedoch dadurch, dass sich Frau Müller aktuell am Bahnhof befindet und dass sie zuvor vom Parkplatz ihres Büros losgefahren ist. Situationen lassen sich also nur durch die Analyse des aktuellen Kontextes und der zeitlich geordneten historischen Kontexte vollständig erfassen.

Innerhalb des Systemdienstes wird der zu einem bestimmten Zeitpunkt gültige Kontext eines zu untersuchenden Objekts durch einen Kontextvektor $\mathbf{c}$ repräsentiert, der sich wiederum aus Kontextattributen zusammensetzt. Jedes Kontextattribut repräsentiert dabei einen bestimmten Teilaspekt der Umgebung, welcher im Sinne der Parametrisierung relevant ist. Ein Kontextattribut besteht dazu wiederum aus jeweils einem Kontextattributnamen a_{name}, der den Aspekt der Umgebung aus der Menge aller Aspekte F_{env} eindeutig identifiziert, und einem Kontextattributwert a_{value}, der eine beliebige Ausprägung des Aspektes der Umgebung repräsentiert. Beispielsweise könnte ein Kontextattribut für den Längengrad der Position des Fahrzeugs, ähnlich wie bei einem Ereignisparameter, die Ausprägung $(Längengrad, 13.32180)$ annehmen.

$$\mathbf{c} = \begin{pmatrix} (a_{name_1}, a_{value_1}) \\ (a_{name_2}, a_{value_2}) \\ \cdots \\ (a_{name_n}, a_{value_n}) \end{pmatrix} \; mit\ n \in \mathbb{N}^*,\ a_{name_x} \in F_{env},\ a_{value_x} \in F_{sta} \quad \text{(Kontextvektor)}$$

Eine bestimmte Ausprägung des Kontextvektors repräsentiert den aktuellen Kontext des Benutzers immer genau so lange, bis ein kontextrelevantes Ereignis eintrifft, welches in der Folge zu einer Aktualisierung des Kontextvektors führt. Gegeben sei die Funktion $f_{type} : E \rightarrow F_{in}$, die jedes Ereignis auf den eindeutigen Typ des Vorfalls abbildet, und die Funktion $f_{edit_context} : C \times E \rightarrow C$, die eine neue Ausprägung des Kontextvektors anhand der aktuellen Ausprägung und eines Eingabereignisses berechnet. Der gesamte Verarbeitungsschritt eines Ereignisses lässt sich damit durch die folgende Funktion beschreiben:

$$f_{update_context}(\mathbf{c}, \check{e}) = \begin{cases} \mathbf{c}' = f_{edit_context}(\mathbf{c}, \check{e})) & \text{wenn } f_{type}(\check{e}) \in F_{rel} \subseteq F_{in} \\ \mathbf{c}' = \mathbf{c} & \text{wenn } f_{type}(\check{e}) \notin F_{rel} \subseteq F_{in} \end{cases}$$

Außerdem gilt für den neu berechneten Kontextvektor $\mathbf{c}'$ folgendes: $t_{context}(\mathbf{c}) < t_{context}(\mathbf{c}') = t_{event}(\check{e})$. In den Vorlagen werden sowohl die Menge F_{rel}, die alle für den Kontext relevanten Ereignistypen enthält, als auch die Funktion $f_{edit_context}$ vorgegeben. In $f_{edit_context}$ wird dazu definiert, auf welche Art und Weise sich die neuen Kontextattributwerte aus den Ereignisparameterwerten berechnen. Dabei kann es sich z.B. um eine einfache Zuweisung oder die komplexe Berechnung eines Kontextattributwertes auf der Basis mehrerer Ereignispara-

meterwerte handeln. In Abbildung 4.5 wird eine beispielhafte Entwicklung der Ausprägung des Kontextvektors auf einer Zeitleiste abgetraten.

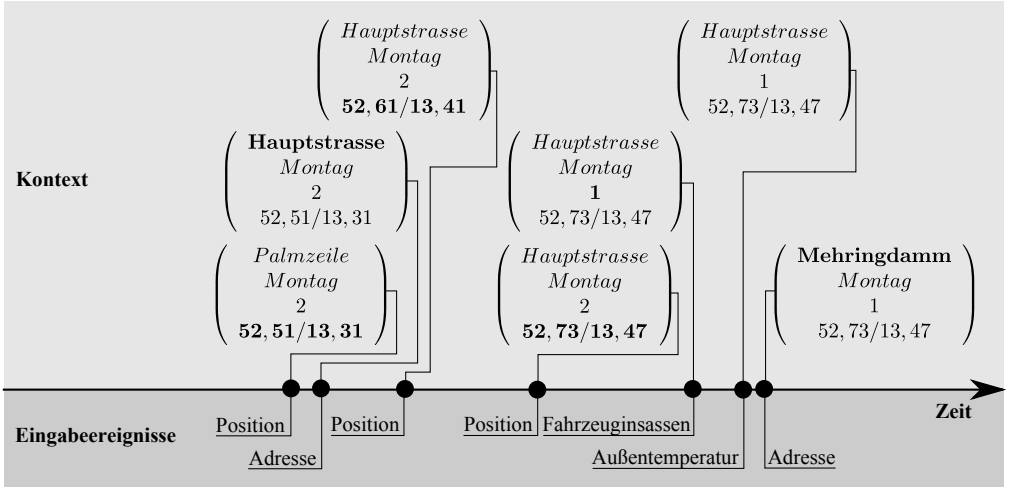


Abb. 4.5 Eine beispielhafte Auflistung aller in einem kurzen Zeitraum möglichen Ausprägungen eines Kontextvektors, der aus den Kontextattributen Straßenname, Wochentag, Fahrzeuginsassen und Position besteht. Wurden Kontextattributwerte aktualisiert, so werden diese hervorgehoben.

Da die Ausprägungen des Kontextvektors in den nachfolgenden Prozessen immer gemeinsam mit den Eingabeereignissen betrachtet werden, gilt es die verschiedenen Kontexte der Sequenz von Eingabeereignissen beizufügen. Der gesamte Verarbeitungsschritt dieses Prozesses lässt sich somit allgemein durch folgende Funktion $f_{context} : \check{S} \rightarrow \mathring{S}$ beschreiben:

$$f_{context}(\check{s} = (\check{e}_1, \check{e}_2, ..., \check{e}_n)) = ((\check{e}_1, \mathbf{c}'_1), (\check{e}_2, \mathbf{c}'_2), ..., (\check{e}_n, \mathbf{c}'_n)) = \mathring{s}$$
$$\text{mit } t_{event}(\check{e}_x) = t_{context}(\mathbf{c}'_x)$$

Jedem Eingabeereignis $\check{e}_x$ aus der Sequenz von Eingabeereignissen $\check{s}$ wird somit zur weiteren Verarbeitung der zu dem Zeitpunkt des Vorfalls gültige Kontext hinzugefügt. In der Tabelle 4.1 werden die in den Vorlagen definierten kontextrelevanten Ereignistypen F_{rel} und die Funktion zum Aktualisieren des Kontextvektors $f_{edit_context}$ für die in den Anwendungsfällen A bis D zu berücksichtigenden Aspekte der Umgebung beispielhaft aufgelistet. In der prototypischen Implementierung wurde neben der einfachen Zuweisung von Werten auch ein sogenannter *Map-Matching*-Algorithmus in $f_{edit_context}$ umgesetzt. Damit können die Längen- und Breitengrade der Position mit Hilfe des entsprechenden Kartenmaterials auf einen eindeutigen Identifikator der Straße abgebildet werden. In den folgenden Beispielen wird jedoch vorausgesetzt, dass dieser Identifikator - hier der Straßenname - bereits durch den entsprechenden Sensor bereitgestellt wird.

4.2.2 Ermittlung relevanter Interaktionen

In diesem Prozess werden die für die Verarbeitung eines Anwendungsfalls notwendigen relevanten Interaktionen auf Basis der in den Vorlagen zu definierenden sequentiellen Interaktionsmuster aus der Sequenz von Ereignis-Kontext-Paaren $\mathring{s}$ extrahiert. Gilt es beispiels-

F_{rel}	$f_{edit\ context}$		
Typ des Ereignisses	**Ereignisparameter**	**Operation**	**Kontextattribut**
Position	Längengrad	=	Längengrad
	Breitengrad	=	Breitengrad
	Straßenname	=	Straße
*	Zeitstempel	=	Zeit

Tabelle 4.1 In den Vorlagen zu definierende Auflistung aller kontextrelevanten Eingabeereignisse und der Aktualisierungsfunktion des Kontextvektors bzgl. der Anwendungsfälle A bis D. Das Sternsymbol * steht stellvertretend für alle Ereignistypen. Die Operation der Zuweisung eines Ereignisparameterwertes an einen Kontextattributwert wird durch das Symbol = gekennzeichnet.

weise das Anrufverhalten des Benutzers zu untersuchen, so müssen in diesem Prozess alle Anrufe im Datenstrom aus 2-Tupeln zuverlässig erkannt und dem nächsten Prozess gemeldet werden. Wird jeder Anruf als ein separater Vorfall gewertet und durch ein eigenes Ereignis repräsentiert, das in den Parametern beispielsweise sowohl den Zeitpunkt des Anrufs als auch die gewählte Telefonnummer enthält, so ist jedes eintreffende 2-Tupel auf den entsprechenden Typ des Vorfalls hin zu untersuchen.

In einigen Fällen ist es jedoch möglich, dass der Benutzer unbeabsichtigt die falsche Telefonnummer wählt und kurz darauf der Anruf wieder abbricht. In diesen Fällen werden Interaktionen erkannt, die zwar wegen des gesuchten Vorfalls dem Suchkriterium entsprechen, aber aufgrund ihres fehlerbedingten Zustandekommens nicht einer relevanten Interaktion gleichzusetzen sind. Solche Interaktionen sollten folglich, auch wenn sie wiederholt auftreten, unter keinen Umständen zu einer kontextsensitiven Personalisierung führen. Daher gilt es neben dem Typ eines Ereignisses auch dessen Korrelationen mit anderen Ereignissen zu untersuchen. Würde das Abbrechen eines Anrufs ebenfalls zu einem Ereignis führen, so ließe sich die Relevanz eines Anrufs durch die Beobachtung der unmittelbar auf einen Anruf folgenden Ereignisse präziser bestimmen. Findet das Abbruchereignis außerdem nur bis zu einer maximalen Zeitdauer nach dem Anruf statt, so könnte bei der Analyse zusätzlich zur kausalen Beziehung beider Ereignisse auch die temporale Korrelation berücksichtigt werden. Das in den Vorlagen zu definierende sequentielle Interaktionsmuster dient dabei in allen genannten Fällen zur kompakten Beschreibung der gemeinsamen Eigenschaften aller Sequenzen von 2-Tupeln, die es bzgl. des Anwendungsfalls zu erkennen gilt.

Gegeben sei die Teilsequenzbeziehung $a \subseteq b$ zweier Sequenzen von Ereignis-Kontext-Paaren $a = (a_1, ..., a_n)$ und $b = (b_1, ..., b_m)$ mit $n \leq m$ unter der Bedingung, dass die natürlichen Zahlen $1 \leq i_1 < i_2, ..., i_n \leq m$ existieren, so dass $a_1 = b_{i_1}, a_2 = b_{i_2}, ..., a_n = b_{i_n}$ gilt. Außerdem sind alle Sequenzen von Ereignis-Kontext-Paaren $\mathring{s}_1, \mathring{s}_2, ..., \mathring{s}_k$ der letzten k Fahrten und ein Algorithmus $p_{pattern}$ gegeben, der - angewandt auf eine beliebige Sequenz $\mathring{s}_x \in \mathring{S}$ - alle einem sequentiellen Interaktionsmuster $p \in P$ entsprechenden Teilsequenzen bzw. relevanten Interaktionen $i \subseteq \mathring{s}_x$ erkennt. Die Aufgabe dieses Prozesses besteht darin, den Algorithmus $p_{pattern}$ jeweils einzeln mit allen Sequenzen $\mathring{s}_1$ bis $\mathring{s}_k$ und den individuellen sequentiellen Interaktionsmustern[1] der betreffenden Anwendungsfälle auszuführen. Die Sequenzen $\mathring{s}_1$ bis $\mathring{s}_k$ werden also bzgl. der Interaktionsmuster aller Anwendungsfälle eva-

[1] Im Nachfolgenden wird der Einfachheit halber stets der allgemeinere Begriff „Interaktionsmuster“ statt der spezifischen Variante „sequentielles Interaktionsmuster“ verwendet.

luiert. Während der Ausführung von $p_{pattern}$ wird jedes 2-Tupel einer beliebigen Sequenz $\mathring{s}$ - beginnend mit dem Ältesten und abschließend mit dem Aktuellsten - einmal bzgl. der Möglichkeit untersucht, ob es ein Bestandteil einer Teilsequenz von $\mathring{s}$ ist, die einem Interaktionsmuster P entspricht. Ist zu einem bestimmten Zeitpunkt während der Verarbeitung bereits eine solche Teilsequenz bekannt, so wird diese unmittelbar an den nächsten Prozess weitergegeben. Eine relevante Interaktion besteht also aus einer Teilsequenz von 2-Tupeln, die die Eigenschaften eines Interaktionsmusters erfüllt.

4.2.2.1 EPL zur Beschreibung eines Interaktionsmusters

Zur Formulierung eines Interaktionsmusters wird auf die in Unterabschnitt 2.3.2 einleitend beschriebenen EPLs und deren Techniken zurückgegriffen. Im Nachfolgenden werden daher anhand der Anwendungsfälle A bis D die in diesem Verfahren genutzten Mechanismen bekannter EPLs im Einzelnen vorgestellt und die daraus abzuleitende Syntax zusammen mit der zugehörigen Semantik erläutert. Dabei werden nur die Operatoren eingeführt, die sich zur Formalisierung der typischen Eigenschaften automotiver Mensch-Maschine-Interaktionen eignen.

Temporale Operatoren

Werden Interaktionsmuster für die Anwendungsfälle A bis D formuliert, so müssen mindestens alle Ereignistypen aufgezählt werden, die in den relevanten Interaktionen zu erwarten sind. Besteht jede zu erkennende relevante Interaktion eines Anwendungsfalls beispielsweise nur aus einem 2-Tupel, welches ein Ereignis eines bekannten Typs aufweist, so lässt sich diese Eigenschaft aller relevanten Interaktionen in allgemeiner Form durch ein *einfaches Interaktionsmuster* beschreiben, das sich ausschließlich aus dem Namen des Ereignistyps zusammensetzt. Das einfache Interaktionsmuster

RadiosenderWechseln

beschreibt daher beispielsweise die Gemeinsamkeiten aller relevanten Radiosenderwechsel aus dem Anwendungsfall A. Folglich besteht jede dieser relevanten Interaktionen aus einem 2-Tupel mit einem Ereignis vom Typ „RadiosenderWechseln“. Während des Pattern Matchings, also der Ausführung des Algorithmus $p_{pattern}$ mit einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$, wird mit diesem Interaktionsmuster das erste 2-Tupel in der Sequenz $\mathring{s}_x$ identifiziert, dass ein Ereignis vom Typ „RadiosenderWechseln“ enthält.

Im Anwendungsfall B werden jedoch relevante Interaktionen gesucht, die sich aus zwei unterschiedlichen 2-Tupeln zusammensetzen. Sie bestehen aus einem 2-Tupel, dass ein Ereignis vom Typ „FahrtStarten“ enthält, und einem 2-Tupel, dass ein dem ersten Ereignis zeitlich nachfolgendes Ereignis vom Typ „FahrtBeenden“ enthält. Derartige Aussagen können üblicherweise mit Hilfe der Linearen Temporal-Logik formuliert werden. So lässt sich beispielsweise mit der Formel $A \wedge \Diamond B$ analog beschreiben, dass zunächst die Aussage A und irgendwann später auch die Aussage B gelten muss. In dieser Forschungsarbeit wird das zeitliche in Beziehung setzen zweier Ereignisse jedoch vereinfacht über den Einsatz des temporalen $\rightarrow$-Operators ermöglicht. Ein einfaches Interaktionsmuster wird durch die Anwendung eines oder mehrerer beliebiger Operatoren wie z.B. des $\rightarrow$-Operators zu einem *komplexen Interaktionsmuster*. Alle aus zwei kausal abhängigen 2-Tupeln bestehenden re-

levanten Interaktionen des Anwendungsfalls B lassen sich somit in allgemeiner Form durch das folgende komplexe Interaktionsmuster beschreiben:

$$\mathrm{FahrtStarten} \rightarrow \mathrm{FahrtBeenden}$$

Wird während des Pattern Matchings ein Ereignis vom Typ „FahrtStarten“ erkannt, so konzentriert sich die Suche von $p_{pattern}$ danach ausschließlich auf Ereignisse vom Typ „FahrtBeenden“. Der binäre $\rightarrow$-Operator kann jedoch nicht nur dazu verwendet werden, zwei einfache Interaktionsmuster in temporale Beziehung zueinander zu setzten, sondern auch zur temporalen Verknüpfung zweier komplexer Interaktionsmuster dienen. Werden beispielsweise während des Startvorgangs im Fahrzeug nacheinander die Ereignisse vom Typ „MotorStarten“ und vom Typ „HandbremseLösen“ erzeugt, dann kann das obige komplexe Interaktionsmuster durch die spezifischere Variante

$$(\mathrm{MotorStarten} \rightarrow \mathrm{HandbremseLösen}) \rightarrow \mathrm{FahrtBeenden}$$

ersetzt werden. In diesem Fall werden beim Pattern Matching die absoluten Zeitabstände zwischen den einzelnen Ereignissen nicht untersucht. Das Ereignis vom Typ „FahrtBeenden“ kann in einer relevanten Interaktion also unmittelbar oder erst nach mehreren Stunden auf das Ereignis vom Typ „HandbremseLösen“ folgen.

Im Anwendungsfall D kann ein Anruf - wie oben beschrieben - auch unbeabsichtigt ausgelöst und kurz darauf wieder beendet werden. Zwei Ereignis-Kontext-Paare mit den Ereignissen vom Typ „AnrufStarten“ und vom Typ „AnrufBeenden“ bilden gemeinsam daher erst dann eine relevante Interaktion, wenn der Anruf mindestens 20 Sekunden dauerte. Zur Beschreibung einer solchen Eigenschaft ist es notwendig, die Auswertung eines beliebigen einfachen oder komplexen Interaktionsmusters um eine minimale Dauer Δy zu verschieben. Dies wird durch den $()_{\Delta y}$-Operator ermöglicht. Die zusätzliche Bedingung im komplexen Muster

$$\mathrm{AnrufStarten} \rightarrow (\mathrm{AnrufBeenden})_{20sek}$$

gibt dabei folglich an, dass zwischen den beiden betreffenden Ereignissen mindestens 20 Sekunden und maximal unendlich viele Sekunden vergehen müssen, damit diese eine relevante Interaktion formen. Sollte der gültige Zeitrahmen eines Interaktionsmusters jedoch begrenzt werden, so ist es mit Hilfe des $()^{\Delta y}$-Operators möglich, den Gültigkeitsbereich eines beliebigen einfachen oder komplexen Interaktionsmusters auf eine Dauer Δy einzuschränken. Als Beispiel sei eine relevante Interaktion aus dem Anwendungsfall C gegeben. Hier könnte eine Aktivierung des Vollbildmodus im Internetbrowser unmittelbar nach dem Webseitenaufruf auf die Relevanz des Webseitenaufrufs hinweisen. Damit ließe sich mit Hilfe des $()^{\Delta y}$-Operators das folgende komplexe Interaktionsmuster formulieren:

$$\mathrm{WebseiteAufrufen} \rightarrow (\mathrm{VollbildAktivieren})^{20sek}$$

Der gültige Zeitrahmen beginnt in diesem Fall unmittelbar nach dem Eintreten des ersten Ereignisses vom Typ „WebseiteAufrufen“ und endet entweder mit einer Erkennung einer relevanten Interaktion nach dem Eintreten eines Ereignisse vom Typ „VollbildAktivieren“ - innerhalb von 20 Sekunden danach - oder nach Ablauf der 20 Sekunden. Beim Pattern Matching wird also nur dann eine relevante Interaktion erkannt, wenn der Vollbildmodus innerhalb von 20 Sekunden nach dem Webseitenaufruf aktiviert wurde.

Logische Operatoren

Werden die Webseitenaufrufe im Anwendungsfall C jedoch erst dann als relevant eingestuft, wenn sie zur Anzeige von Webseiten mit ortsabhängigen Informationen führen, so gilt es außerdem zu überprüfen, ob der Internetbrowser aufgrund einer auszuführenden Lokalisierung der Inhalte auf die Position des Fahrzeugs zugreift. Da in diesem Anwendungsfall ohnehin davon ausgegangen wird, dass sich das Verhalten bei der Benutzung des Internetbrowsers ortsabhängig ändert, würde eine solche zusätzliche Prüfung ein weiteres Indiz für die Relevanz einer Interaktion liefern. Um das zusätzliche Kriterium in das bestehende komplexe Interaktionsmuster zu integrieren, bedarf es des von Luckham in [Luckham u. Vera, 1995] vorgestellten &-Operators. Zwei komplexe Interaktionsmuster *a* und *b*, die über einen logischen Operator der Konjunktion miteinander verknüpft sind, beschreiben in allgemeiner Form alle Interaktionen, auf die zuerst die Aussage *a* und dann die Aussage *b* (oder umgekehrt) zutrifft. Da bei der relevanten Interaktion des Webseitenaufrufs vorher nicht immer bekannt ist, welches der Ereignisse „VollbildAktivieren“ und „PositionAbfragen“ zuerst stattfindet, ergibt sich die Erweiterung des obigen Interaktionsmusters wie folgt:

$$\mathrm{WebseiteAufrufen} \rightarrow (\mathrm{VollbildAktivieren} \;\&\; \mathrm{PositionAbfragen})^{20sek}$$

Während der Eingabe einer URL kommt es jedoch häufig zu Fehlern, die vom Benutzer erst nach dem Ladevorgang bemerkt werden. In einem solchen Fall wird die Eingabe vom Benutzer wiederholt und der Ladevorgang erneut ausgeführt. Nach einem gescheiterten Versuch, der für die Personalisierung als irrelevant eingestuft werden kann, folgt also unmittelbar ein Webseitenaufruf, der potentiell relevant sein könnte. Ein Webseitenaufruf ist folglich erst dann relevant, wenn unmittelbar danach kein weiterer Webseitenaufruf erfolgte. Für eine solche Aussage bedarf es der Möglichkeit, ein Ereignis oder ein komplexes Interaktionsmuster zu negieren. Im Gegensatz zum Anwendungsgebiet der regulären Ausdrücke ist die zu untersuchende Menge an Daten beim CEP unendlich. Wird also z.B. ein spezielles Ereignis negiert, so würde das Pattern Matching erst dann eine relevante Interaktion erkennen, wenn der Datenstrom endet. Daher wird der Gültigkeitsbereich einer Negation in EPLs üblicherweise zeitlich begrenzt und/oder an zusätzliche Ereignisse gebunden. In der hier vorgestellten EPL kann eine Aussage negiert werden, wenn die zu negierende Aussage im Nutzungskontext über einen Start- und Endzeitpunkt des Gültigkeitsbereiches verfügt. Diese bereits von Zhu in SEL [Zhu u. Sethi, 2001] verwendete Methode, Negationen flexibel in EPLs einzusetzen, führt unter Anwendung des ¬-Operators im komplexen Interaktionsmuster

$$\mathrm{WebseiteAufrufen} \rightarrow (\neg(\mathrm{WebseiteAufrufen}))^{20sek}$$

dazu, dass ein Webseitenaufruf erst dann als relevant eingestuft wird, wenn innerhalb von 20 Sekunden kein weiterer Webseitenaufruf stattfindet. Während der Startzeitpunkt des Gültigkeitsbereiches der Negation durch den Zeitpunkt des ersten Ereignisses gegeben ist, legt die zeitliche Bedingung am temporalen Operator das Ende dieses Gültigkeitbereiches fest.

In einigen Fällen lässt sich eine relevante Interaktion auch durch zwei unterschiedliche Interaktionsmuster formulieren. Um diese Interaktionsmuster dennoch in einem gemeinsamen Interaktionsmuster zu vereinen, wird mit dem |-Operator die Disjunktion eingeführt. Die Disjunktion zweier einfacher oder komplexer Interaktionsmuster *a* und *b* führt laut Luckham in der RAPIDE-EPL [Luckham u. Vera, 1995] während des Pattern Matching

dazu, dass alle Interaktionen als relevant eingestuft werden, die entweder durch das Interaktionsmuster *a* oder durch das Interaktionsmuster *b* beschrieben werden können. Gilt es im Anwendungsfall D zur Ermittlung der kontextsensitiven Präferenz nicht nur das Anrufverhalten, sondern auch die SMS-Kommunikation zu untersuchen, so könnte das dazugehörige komplexe Interaktionsmuster beispielsweise mit Hilfe des |-Operators wie folgt formuliert werden:

AnrufStarten | SMSSenden

Somit wird ein kontextsensitiv personalisierter Schnellstartbutton zur direkten Ausführung eines Anrufs immer in Situationen angezeigt, in denen entweder wiederholt telefoniert oder Kurznachrichten verschickt wurden.

Parameterfilter

Bisher wurden bei der Formulierung der Interaktionsmuster nur die temporalen und logischen Beziehungen zwischen den Ereignissen berücksichtigt. Die in den Ereignisparametern enthaltenen Eigenschaften der Vorfälle sollten jedoch ebenfalls bestimmte Bedingungen erfüllen. Sind für einen Anwendungsfall beispielsweise nur Radiosenderwechsel relevant, die von einem bestimmten Benutzer X ausgeführt werden, so muss das einfache Interaktionsmuster für den Radiosenderwechsel um einen *Parameterfilter* erweitert werden. Dieser Parameterfilter gibt dabei an, über welchen zulässigen Wertebereich sich die Ereignisparameterwerte innerhalb einer relevanten Interaktion erstrecken dürfen. Das um einen Parameterfilter erweiterte einfache Interaktionsmuster des Radiosenderwechsels

RadiosenderWechseln[Benutzer=X]

beschreibt demnach nur Ereignisse, die vom Typ „RadiosenderWechseln“ sind und deren Ereignisparameter „Benutzer“ den Wert X enthält. Ein Parameterfilter kann dabei auch mehrere Bedingungen enthalten, die über die logischen Operatoren && (Konjunktion), || (Disjunktion) und ! (Negation) miteinander verknüpft werden. Sollte beispielsweise ein bestimmter Radiosender Y auf Wunsch des Benutzers von der kontextsensitiven Personalisierung ausgeschlossen werden, dann ergibt sich das einfache Interaktionsmuster wie folgt:

RadiosenderWechseln[(Benutzer=X) && !(Radiosender=Y)]

Während der Ereignisparameterwert des Ereignisparameters mit Namen „Benutzer“ der Konstante X entsprechen muss, darf der Ereignisparameterwert des Ereignisparameters mit Namen „Radiosender“ nicht den Wert Y annehmen. Neben der Operation des direkten Vergleichs, können auch die Vergleichsoperatoren $<$, $>$, $\leq$ und $\geq$ zur Definition einer Bedingung genutzt werden. Auch das mehrfache Aufführen des gleichen Parameters in verschiedenen Bedingungen ist möglich. Eine Definition der Beziehungen zwischen den Werten der Parameter verschiedener Ereignisse innerhalb eines Interaktionsmusters wird im Gegensatz zu der EPL für die CEP-Engine von Esper [EsperTech Inc., 2010] hier nicht unterstützt. Der Grund für den Ausschluss dieser Funktionalität liegt in der problematischen Anwendung des |-Operators. Werden zwei Parameter in Beziehung zueinander gesetzt, die sich auf zwei verschiedene Ereignisse verteilen, die über den |-Operator miteinander verknüpft sind, so kann diese Beziehung während des Pattern Matching nie aufgelöst werden, da entweder das Eintreten des einen oder des anderen Ereignisses zur Erkennung einer relevanten Interaktion führt.

Subausdruck-Operator

Wird ein Pattern Matching auf Basis der oben vorgestellten einfachen und komplexen Interaktionsmuster durchgeführt, so wird in einer Sequenz $\mathring{s}_x$ jeweils nur die älteste relevante Interaktion gefunden, die den genannten Interaktionsmustern entspricht. Alle nachfolgenden relevanten Interaktionen in $\mathring{s}_x$ werden jedoch ignoriert. Da der Algorithmus $p_{pattern}$ auf jede Sequenz $\mathring{s}_1$ bis $\mathring{s}_k$ einzeln angewandt wird, können somit maximal k relevante Interaktionen gefunden werden. Während einer Fahrt können jedoch z.B. mehrere relevante Radiosenderwechsel oder Webseitenaufrufe ausgeführt werden. Daher gilt es, den im einfachen Interaktionsmuster des Anwendungsfalls A beschriebenen Ereignistyp in $\mathring{s}_x$ beliebig oft zu finden. Diese Anforderung wird durch den unären #-Operator formuliert. Dieser in der EPL von Esper als Every-Operator [EsperTech Inc., 2010] bezeichnete Subausdruck-Operator veranlasst den Pattern Matching Algorithmus, nach jeder Erkennung eines dem #-Operator übergebenen Subausdrucks mit einer neuen Suche zu beginnen. Mit dem komplexen Interaktionsmuster

$$\#(\mathrm{Radiosenderwechsel})$$

werden somit alle relevanten Radiosenderwechsel in einer beliebigen Sequenz $\mathring{s}_x$ erkannt. Innerhalb einer Sequenz $\mathring{s}_x$, die alle bei einer Fahrt generierten 2-Tupel enthält, treten die Ereignisse „FahrtStarten“ und „FahrtBeenden“ jeweils nur maximal einmal auf. Eine Anwendung des #-Operator innerhalb des Interaktionsmusters aus Anwendungsfall B ist daher nicht notwendig. Im Anwendungsfall C können jedoch genauso wie im Anwendungsfall A beliebig viele relevante Interaktionen pro Fahrt auftreten. Die analoge Anwendung des Subausdruck-Operators auf das gesamte komplexe Interaktionsmuster zur Erkennung relevanter Webseitenaufrufe

$$\#(\mathrm{WebseiteAufrufen} \rightarrow (\neg(\mathrm{WebseiteAufrufen}))^{20sek})$$

führt jedoch nicht zu dem gewünschten Ergebnis. Werden beispielsweise zwei Webseitenaufrufe innerhalb von 20 Sekunden nacheinander getätigt, so führt der zweite Webseitenaufruf zur negativen Auswertung der Untersuchung des ersten Webseitenaufrufs. Der zweite Webseitenaufruf ist aufgrund der Negation jedoch ein Teil der zuvor untersuchten Teilsequenz. Daher wird die Suche nach einem weiteren relevanten Webseitenaufruf erst nach dem zweiten Webseitenaufruf gestartet. Der zweite Webseitenaufruf wird also nicht weiter untersucht, obwohl er relevant sein könnte. Um dennoch jeden relevanten Webseitenaufruf zu erkennen, gilt es, jedes Ereignis vom Typ Webseitenaufruf gesondert zu untersuchen. Wird der Subausdruck-Operator, wie im komplexen Interaktionsmuster

$$\#(\mathrm{WebseiteAufrufen}) \rightarrow (\neg(\mathrm{WebseiteAufrufen}))^{20sek}$$

gezeigt, nur auf das Ereignis vom Typ „WebseitenAufrufen“ angewendet, so wird jedes Ereignis vom Typ „WebseitenAufrufen“ dahingehend untersucht, ob es innerhalb von 20 Sekunden nicht von einem Ereignis gleichen Typs gefolgt wird. Generell gilt dabei, dass ab dem Moment, an dem ein Ereignis vom Typ „WebseitenAufrufen“ auftritt, der Pattern Matching Algorithmus den ersten Teil des Interaktionsmusters positiv auswertet und anfängt, nach einem innerhalb von 20 Sekunden folgenden Ereignis gleichen Typs zu suchen. Parallel dazu startet der Pattern Matching Algorihtmus jedoch die Suche nach einem Ereignis vom Typ „WebseitenAufrufen“ von Neuem. Somit finden ab dem Zeitpunkt des ersten Ereignisses vom Typ „WebseitenAufrufen“ zwei unabhängige Untersuchungen statt, die jeweils zu zwei unabhängigen Ergebnissen führen können. Im Laufe der Erkennung

eines solchen Interaktionsmusters können jedoch maximal zwei Untersuchungen gleichzeitig stattfinden. Die generelle Suche nach Ereignissen vom Typ „WebseitenAufrufen“, die durch die Anwendung des #-Operator initiiert wird, findet also immer statt. Die dazu parallel auszuführenden Untersuchungen nach folgenden Ereignissen gleichen Typs können sich jedoch zeitlich nie überschneiden, da sich diese Untersuchungen beenden, sobald entweder kein oder ein Ereignis gleichen Typs innerhalb von 20 Sekunden auftritt.

Der Subausdruck-Operator veranlasst den Pattern Matching Algorithmus folglich nach jeder positiven Auswertung eines übergebenen Subausdrucks, mit einer neuen unabhängigen Untersuchung des Subausdrucks zu beginnen. Gleichzeitig wird - aufgrund der positiven Auswertung des Subausdrucks - die Untersuchung des den Subausdruck umgebenden Interaktionsmusters fortgesetzt. In Abbildung 4.6 werden zur Verdeutlichung der Funktionsweise des Subausdruck-Operators die Auswertungen von $p_{pattern}$ einer Beispielsequenz bzgl. fünf unterschiedlicher Interaktionsmuster aufgezeigt. Der untere Teil der Abbildung zeigt beispielhaft die Reihenfolge der sequentiellen Auswertung des fünften Interaktionsmusters. Alle Äste werden dabei parallel abgearbeitet.

Beispielsequenz: $a_1, a_2, c_3, b_4, a_5, c_6, b_7, b_8$	
Interaktionsmuster	**Relevante Interaktionen**
$\#(a \rightarrow b)$	$(a_1, b_4), (a_5, b_7)$
$\#a \rightarrow b$	$(a_1, b_4), (a_2, b_4), (a_5, b_7)$
$a \rightarrow \#b$	$(a_1, b_4), (a_1, b_7), (a_1, b_8)$
$\#a \rightarrow \#b$	$(a_1, b_4), (a_2, b_4), (a_1, b_7), (a_2, b_7), (a_5, b_7), (a_1, b_8), (a_2, b_8), (a_5, b_8)$
$\#_1(\#_2 a \rightarrow b)$	$(a_1, b_4), (a_2, b_4), (a_5, b_7), (a_5, b_7), (a_5, b_7)$

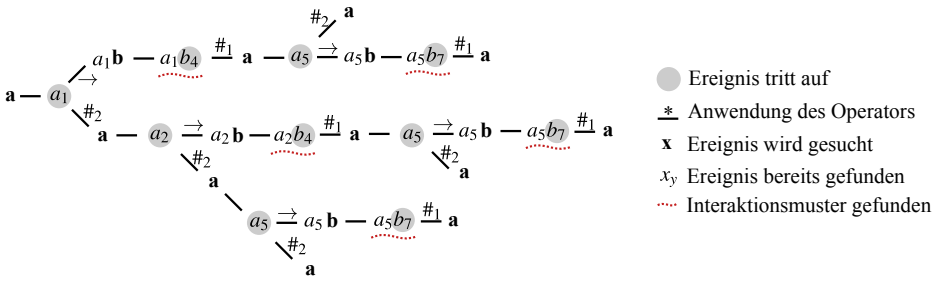


Abb. 4.6 Oben: Beispielhafte Auflistung der Ergebnisse fünf unterschiedlicher Evaluationen der gleichen Beispielsequenz. Unten: Reihenfolge der Anwendungen aller Operatoren des fünften Interaktionsmusters.

Syntax und Semantik

Die in den obigen Beispielen verwendete Syntax zur Formulierung der diversen einfachen und komplexen Interaktionsmuster wird im folgenden zusammenfassend in BNF-Form dargestellt. Die nicht-terminalen Symbole P, F, C und Q stehen stellvertretend für das komplexe Interaktionsmuster P, den Parameterfilter F, die Bedingung eines Parameterfilters C und den Vergleichsoperator Q. Die terminalen Symbole R, N, P und W stehen dabei stellvertretend für das einfache Interaktionsmuster R, den Zeitrahmen N, den Ereignisparameternamen PA und den Ereignisparameterwert W.

$$
\begin{aligned}
P = {} & R \mid R\,'['\,F\,']' \mid P\,' \rightarrow {}' P \mid '('\,P\,')' \mid '('\,P\,')'_N \\
& \mid '('\,P\,')'^N \mid P\,'\&'\,P \mid '\neg' P \mid P\,'|'\,P \mid \#(P) \\
F = {} & C \mid F\,'\&\&'\,F \mid F\,'||'\,F \mid '!' F \mid '('\,F\,')' \\
C = {} & PA\ Q\ W \\
Q = {} & '=' \mid '<' \mid '>' \mid '\leq' \mid '\geq'
\end{aligned}
$$

Innerhalb eines Parameterfilters hat der !-Operator Vorrang gegenüber dem &&-Operator und dieser wiederum Vorrang gegenüber dem ||-Operator. Die Vorrangstellungen der Operatoren auf der Ereignisebene ergeben sich mit absteigender Priorität wie folgt:

$$
\begin{aligned}
1 = {} & \text{\#-Operator und } \neg\text{-Operator} \\
2 = {} & ()^{\Delta y}\text{-Operator und } ()_{\Delta y}\text{-Operator} \\
3 = {} & \&\text{-Operator} \\
4 = {} & |\text{-Operator} \\
5 = {} & \rightarrow \text{-Operator}
\end{aligned}
$$

Werden mehrere Operatoren der gleichen Priorität innerhalb eines Ausdrucks ohne Klammerung verwendet, so findet die Auswertung von links nach rechts statt. Die hier vorgestellte Syntax wird im weiteren Verlauf dieser Forschungsarbeit aufgrund ihrer Einfach- und Kompaktheit zur Formulierung aller weiteren beispielhaften Interaktionsmuster verwendet. Innerhalb der prototypischen Implementierung kommt jedoch eine auf der erweiterten Auszeichnungssprache XML basierende Syntax zur Beschreibung der Interaktionsmuster zum Einsatz. Die oben beschriebene Syntax lässt sich jedoch eindeutig auf die in der Implementierung verwendete Syntax abbilden.

In dem hier vorgestellten Verfahren wird aus Gründen der praxisnahen Anwendung angenommen, dass eine bestimmte Menge von Ereignis-Kontext-Paaren nur dann eine gemeinsame relevante Interaktion formen kann, wenn die Zeitdifferenzen zwischen den Vorfällen aller an einer relevanten Interaktion beteiligten 2-Tupel jeweils einen maximalen Zeitrahmen nicht überschreiten. Eine relevante Interaktion kann sich beispielsweise nicht über mehrere Wochen hinziehen. Jedes Interaktionsmuster erhält daher obligatorisch die Bedingung, dass eine dem Interaktionsmuster entsprechende Sequenz immer nur eine maximale Dauer aufweisen kann. Da der Gültigkeitsbereich eines Interaktionsmusters in diesem Verfahren ohnehin durch die Dauer einer Fahrt beschränkt wird, entspricht der maximal zulässige Zeitrahmen in diesem Fall der Dauer der längsten Fahrt. Ab wann eine neue Fahrt begonnen wird oder alternativ nach z.B. einem Tankstopp fortgesetzt wird, muss dabei gesondert festgestellt werden. Nachfolgend wird jedoch vorausgesetzt, dass dem Verfahren die Start- und Endzeitpunkte aller Fahrten zur Verfügung gestellt werden. Somit ergibt sich der maximale Gültigkeitsbereich eines Operators durch das Zeitfenster t_{start} bis t_{end}.

Sei außerdem durch $t_{detect}(p)$ der Zeitpunkt gegeben, an dem ein Interaktionsmuster $p \in P$ erkannt wird. Dieser Zeitpunkt bezieht sich dabei auf die künstliche Zeit, die sich aus den Zeitstempeln jedes Ereignisses berechnet, und nicht aus der absoluten Zeit, die sich aus dem zentralen Taktgeber des Systems ableitet. Da die Verarbeitung eines Ereignisses mit einem Zeitaufwand verbunden ist, besteht immer eine Differenz zwischen der absoluten und der künstlichen Zeit. Näheres dazu wird im Unterabschnitt 4.2.5 erläutert. Unter diesen Voraussetzungen ergibt sich die Semantik für die hier beschriebene EPL durch einen rekursiven Aufbau wie folgt:

- Einfaches Interaktionsmuster *em*: Ein einfaches Interaktionsmuster *em* gilt bei der Evaluation einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn bei der seriellen Verarbeitung das erste 2-Tupel (e_i, c_i) angetroffen wird, das ein Ereignis vom Typ *em* enthält. Für alle anderen 2-Tupel (e_j, c_j) in $\mathring{s}_x$ mit $f_{type}(e_j) = f_{type}(e_i)$ gilt folglich $t_{event}(e_j) > t_{event}(e_i)$. Der Zeitpunkt $t_{detect}(p = em)$ entspricht $t_{event}(e_i)$.

- $\#(P)$: Ein komplexes Interaktionsmuster $\#(P)$ gilt bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ jedes Mal als erkannt, wenn das Interaktionsmuster P erkannt wird.

- $\neg(P)$: Ein komplexes Interaktionsmuster $\neg(P)$ gilt bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn innerhalb des durch t_{start} und t_{end} aufgespannten Zeitraums keine Teilsequenz $\mathring{s}_i \subseteq \mathring{s}_x$ dem Interaktionsmuster P entspricht. Der Zeitpunkt $t_{detect}(\neg(P))$ entspricht t_{end}.

- $(P)^{\Delta y}$ und $(P)_{\Delta y}$: Ein komplexes Interaktionsmuster $(P)^{\Delta y}$ gilt bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn innerhalb des durch t_{start} bis $t_{start} + \Delta y$ aufgespannten Zeitraums eine Teilsequenz $\mathring{s}_i \subseteq \mathring{s}_x$ dem Interaktionsmuster P entspricht. Analog gilt das komplexe Interaktionsmuster $(P)_{\Delta y}$ bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn innerhalb des durch $t_{start} + \Delta y$ bis t_{end} aufgespannten Zeitraums eine Teilsequenz $\mathring{s}_i \subseteq \mathring{s}_x$ dem Interaktionsmuster P entspricht. $t_{detect}((P)^{\Delta y})$ entspricht $t_{detect}(P)$ mit $t_{start} \leq t_{detect}(P) \leq t_{start} + \Delta y$, während $t_{detect}((P)_{\Delta y})$ dem Zeitpunkt $t_{detect}(P)$ mit $t_{start} + \Delta y \leq t_{detect}(P) \leq t_{end}$ entspricht.

- P_1 & P_2: Ein komplexes Interaktionsmuster P_1 & P_2 gilt bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn beide Interaktionsmuster P_1 und P_2 erkannt wurden. Der Zeitpunkt $t_{detect}(P_1 \,\&\, P_2)$ entspricht $max(t_{detect}(P_1), t_{detect}(P_2))$.

- $P_1 \mid P_2$: Ein komplexes Interaktionsmuster $P_1 \mid P_2$ gilt bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn entweder Interaktionsmuster P_1 oder das Interaktionsmuster P_2 erkannt wurde. Der Zeitpunkt $t_{detect}(P_1 \mid P_2)$ entspricht $min(t_{detect}(P_1), t_{detect}(P_2))$.

- $P_1 \rightarrow P_2$: Ein komplexes Interaktionsmuster $P_1 \rightarrow P_2$ gilt bei der seriellen Verarbeitung einer beliebigen Sequenz $\mathring{s}_x \in \mathring{S}$ als erkannt, wenn das Interaktionsmuster P_2 innerhalb des Zeitraumes nach dem Zeitpunkt $t_{detect}(P_1)$ erkannt wird. Der Zeitpunkt $t_{detect}(P_1 \rightarrow P_2)$ entspricht $t_{detect}(P_2)$ mit $t_{detect}(P_1) < t_{detect}(P_2) \leq t_{end}$.

Die Interaktionsmuster für die Anwendungsfälle A bis D werden zusammenfassend in der Tabelle 4.2 dargestellt. Die auf diesen Interaktionsmustern basierende beispielhafte Parametrisierung des Prozesses zur Ermittlung relevanter Interaktionen wird im weiteren Verlauf dieser Forschungsarbeit als gegeben vorausgesetzt.

Anwendungsfall	Interaktionsmuster
A	$\#(\mathrm{RadiosenderWechseln}) \rightarrow (\neg(\mathrm{RadiosenderWechseln}))^{10sek}$
B	$\mathrm{FahrtStarten} \rightarrow (\mathrm{FahrtBeenden})_{15min}$
C	$\mathrm{FahrtStarten} \rightarrow (\#(\mathrm{WebseiteAufrufen}) \rightarrow (\neg\ \mathrm{WebseiteAufrufen})^{10sek})$
D	$\#(\mathrm{AnrufStarten}) \rightarrow \mathrm{AnrufBeenden}[\mathrm{Dauer}{>}10\mathrm{Sek}]$

Tabelle 4.2 Die in den Vorlagen mit Hilfe der vorgestellten EPL zu definierenden sequentiellen Interaktionsmuster für die Anwendungsfälle A bis D.

4.2.2.2 Pattern Matching auf Basis der EPL

Zur praktischen Anwendung einer EPL bedarf es eines Pattern Matching Algorithmus, der eine möglichst effiziente Auswertung des kontinuierlichen Stroms an Ereignissen ermöglicht. Da dieser Prozess innerhalb von CEP-Systemen allein durch den Zufluss von Daten gesteuert wird, basieren viele der Implementierungen auf Daten-getriggerten Verfahren mit endlichen Automaten [Gehani u. a., 1992; Agrawal u. a., 2008], Petri-Netzen [Gatziu u. Dittrich, 1994] und Ereignis-Bäumen [Chakravarthy u. a., 1994] oder Varianten des RETE Algorithmus [Berstel, 2002; Walzer u. a., 2007]. Dabei werden beispielsweise durch Interaktionsmuster definierte endliche Automaten für jede zu untersuchende Teilsequenz während der Erkennung instanziiert und Ereignis für Ereignis abgearbeitet. Eine Untersuchung dieser Verfahren und die Entwicklung eines für die vorgestellte EPL geeigneten Pattern Matching Algorithmus ist nicht Bestandteil dieser Forschungsarbeit. Da die oben beschriebene EPL nur eine Teilmenge aller bekannten Operatoren für EPLs unterstützt, kann sie auf bekannte EPLs abgebildet und durch deren Pattern Matching Algorithmen effizient implementiert werden. Aufgrund der direkten oder indirekten Unterstützung aller der oben vorgestellten Operatoren, wird innerhalb der prototypischen Implementierung daher die oben definierte EPL auf die EPL von Esper [EsperTech Inc., 2010] abgebildet und deren Pattern Matching Engine verwendet. Welche Bedingungen es beim Einsatz bestehender Pattern Matching Algorithmen dennoch zu beachten gilt, wird in Unterabschnitt 4.2.5 beschrieben.

4.2.3 Gruppierung relevanter Interaktionen

Während im zuvor beschriebenen Prozess relevante Interaktionen erkannt werden, gilt es in dem hier vorgestellten Prozess, die relevanten Interaktionen anhand der Kontexte zu gruppieren. Diese Gruppierung findet nicht unmittelbar nach der Ermittlung einer relevanten Interaktion statt, sondern wird erst ab einer bestimmten Anzahl neu erkannter relevanter Interaktionen durchgeführt. Alternativ ließe sich bei selten auftretenden relevanten Interaktionen auch eine Periode definieren, nach der die Gruppierung ausgeführt wird. Auch eine Gruppierung nach dem Ende einer Fahrt ist möglich, wenn die dafür notwendigen Ressourcen auch in einem abgeschalteten Fahrzeug zur Verfügung stehen.

Während der Ausführung der Gruppierung werden die neuen relevanten Interaktionen, die seit dem letzten Gruppieren neu ermittelt wurden, gemeinsam mit den älteren relevanten Interaktionen, die zuvor bereits mindestens einmal eine Gruppierung durchliefen, bestimmten Gruppen zugeordnet. Ein künstlicher Verfall der Gültigkeit relevanter Interaktio-

nen gewährleistet dabei, dass nur relevante Interaktionen bei der Gruppierung berücksichtigt werden, deren Vorfälle nicht zu weit in der Vergangenheit liegen. Inwiefern der künstliche Verfall und die Ergebnisse einer älteren Gruppierung den Prozess der Gruppierung beeinflussen, wird im Unterabschnitt 4.2.3.2 ausführlich behandelt.

Eine Gruppe relevanter Interaktionen zeichnet sich unter anderem dadurch aus, dass sich die in ihr befindlichen relevanten Interaktionen bzgl. bestimmter Kriterien ähneln und laut dem künstlichen Verfall gültig sind. Aufgrund der Tatsache, dass erst dann eine kontextsensitive Präferenz erkannt werden soll, wenn genug relevante Interaktionen auf eine solche Präferenz hinweisen, überschreitet oder gleicht die Anzahl der Mitglieder jeder Gruppe außerdem immer einen in den Vorlagen definierten Mindestwert $\varsigma \in \mathbb{N}^*$. Eine Gruppe lässt sich nicht nur durch eine Auflistung aller in ihr enthaltenen relevanten Interaktionen charakterisieren, sondern auch durch ihre typischen Eigenschaften. Die typischen Eigenschaften einer Gruppe ergeben sich aus einer Zusammenlegung der Ausprägungen bestimmter Ereignisparameter oder Kontextattribute aller relevanten Interaktionen einer Gruppe. Fand im Anwendungsfall D beispielsweise der früheste relevante Anruf einer Gruppe um 14:04 Uhr und der späteste relevante Anruf der Gruppe um 16:18 Uhr statt, so kann die Ausprägung der typischen Eigenschaft „Zeit“ durch ein Zeitintervall [14:04,16:18] beschrieben werden. Alle typischen Eigenschaften einer Gruppe werden innerhalb des *Gruppenvektors*

$$\mathbf{g} = \begin{pmatrix} (g_{name_1}, g_{value_1}) \\ \dots \\ (g_{name_n}, g_{value_n}) \end{pmatrix} \; mit \; n \in \mathbb{N}^*, \; g_{name_x} \in F_{property}, \; g_{value_x} \in F_{g_{name_x}}$$

zusammengefasst. Jede Gruppe wird folglich auf einen individuellen Gruppenvektor abgebildet. Welche Eigenschaften g_{name} im Einzelnen Bestandteil eines Gruppenvektors sind, kann sich von Anwendungsfall zu Anwendungsfall unterscheiden und ergibt sich aus den im Unterabschnitt 4.2.3.1 beschriebenen *Vergleichseigenschaften* relevanter Interaktionen. Der in diesem Prozess zu berechnende Wert g_{value} einer typischen Eigenschaft kann aus einem *Wahrheitswert*, einer Liste von *Nominalwerten*, einer Liste von *zusammengesetzten Werten* oder aus einem *Intervall* bestehen. Die nominale Eigenschaft „Straße“ einer Gruppe aus dem Anwendungsfall D besteht beispielsweise aus einer Liste der Namen aller Straßen, in denen Anrufe getätigt wurden, während die Intervalleigenschaft „Zeit“ die Tageszeit beschreibt, zu der üblicherweise telefoniert wird. Eine Liste von Positionen bzw. zusammengesetzter Werte, jeweils bestehend aus zwei Längen- und zwei Breitengraden, aus Anwendungsfall C könnte hingegen einen geometrischen Raum aufspannen, innerhalb dessen regelmäßig ein Webseitenaufruf erfolgt.

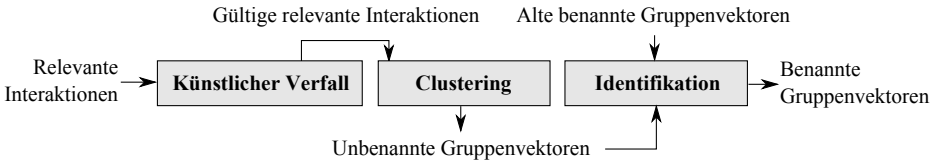


Abb. 4.7 Übersicht aller Verarbeitungsschritte innerhalb des Prozesses zur Gruppierung relevanter Interaktionen.

Das Ziel dieses Prozesses besteht darin, die Gruppen und insbesondere die Ausprägungen der typischen Eigenschaften vollständig und effizient automatisch zu ermitteln. Der Prozess der Gruppierung durchläuft dabei nacheinander verschiedene Verarbeitungsschritte:

Künstlicher Verfall, *Clustering* und *Identifikation* (siehe Abbildung 4.7). In einem ersten Schritt werden alle bisher gespeicherten relevanten Interaktionen daraufhin untersucht, ob deren Vorfälle zu weit in der Vergangenheit liegen. Da relevante Interaktionen aus mehreren Ereignissen bestehen können, werden für diese Untersuchung immer nur die ältesten Zeitstempel einer relevanten Interaktion betrachtet. Ist eine relevante Interaktion zu alt, so wird sie aus dem Speicher bzw. aus der Historie entfernt. Alternativ ließe sich der künstliche Verfall auch durch eine maximal zulässige Anzahl gesammelter relevanter Interaktionen implementieren. Umfasst die Historie bereits die maximale Anzahl zulässiger relevanter Interaktionen, so führt die Ermittlung einer neuen in diesem Fall zur Entfernung der ältesten relevanten Interaktion. Diese Art des künstlichen Verfalls hat dabei den Vorteil, dass der zulässige Speicher auch dann nicht überschritten wird, wenn ungewöhnlich viele relevante Interaktionen innerhalb eines kurzen Zeitraums erkannt werden. Eine Kombination beider Verfahren ist auch möglich. In den nächsten Unterabschnitten werden nacheinander die beiden auf den künstlichen Verfall folgenden Verarbeitungsschritte vorgestellt.

4.2.3.1 Clustering gültiger relevanter Interaktionen

Je nach Szenario können in den Vorlagen unterschiedlich viele Interaktionsmuster für den gleichen Anwendungsfall definiert werden. Der Radiosenderwechsel lässt sich beispielsweise durch zwei Interaktionsmuster definieren, die einen Radiosenderwechsel auf Basis der Betätigung der ZBE oder der Direktwahltasten beschreiben. Relevante Interaktionen unterscheiden sich also nicht nur bzgl. der Ausprägungen der Ereignisparameter und Kontextattribute, sondern möglicherweise auch in ihrem strukturellen Aufbau. Da der Gruppierungsprozess jedoch voraussetzt, dass relevante Interaktionen untereinander vergleichbar sind, müssen Vergleichseigenschaften einer relevanten Interaktion definiert werden. Die unterschiedlichen relevanten Radiosenderwechsel müssen beispielsweise jeweils mindestens und in irgendeiner Form den Namen des neu eingestellten Radiosenders beschreiben. Eine solche Vergleichseigenschaft dient letztlich der Differenzierung unterschiedlicher kontextsensitiver Präferenzen, die sich aus den Gruppen relevanter Interaktionen ableiten lassen. Alle für einen Anwendungsfall relevanten Vergleichseigenschaften werden im *Eigenschaftsvektor*

$$\mathbf{r} = \begin{pmatrix} (r_{name_1}, r_{value_1}) \\ \dots \\ (r_{name_n}, r_{value_n}) \end{pmatrix} \; mit \; n \in \mathbb{N}^*, \; r_{name_x} \in F_{property}, \; r_{value_x} \in F_{r_{name_x}} \tag{4.1}$$

zusammengefasst. Dieser setzt sich aus einer Reihe von 2-Tupeln zusammen, die jeweils aus dem Bezeichner einer Vergleichseigenschaft r_{name} und dem dazugehörigen Wert r_{value} bestehen. Die Ausprägung r_{value} einer beliebigen Vergleichseigenschaft kann unterschiedliche Formen annehmen. Es kann sich dabei um einen Wahrheitswert mit den Ausprägungen „Richtig“ oder „Falsch“, einen Nominalwert wie z.B. einen Straßennamen, eine Zahl wie z.B. die Außentemperatur oder einen zusammengesetzten Wert (z.B. Position bestehend aus Längen- und Breitengrad) handeln. Jede relevante Interaktion lässt sich auf genau einen Eigenschaftsvektor abbilden, der die relevante Interaktion bzgl. des Anwendungsfalls vollständig charakterisiert. Die Anzahl und die Typen der Vergleichseigenschaften können sich von Anwendungsfall zu Anwendungsfall unterscheiden und werden in den Vorlagen vorgegeben. Die Eigenschaftsvektoren ermöglichen es nun, die Ähnlichkeit zweier relevanter Interaktionen durch eine Gegenüberstellung der Ausprägungen der Vergleichseigen-

schaften mit Hilfe einer Distanzfunktion zu quantifizieren.

Innerhalb eines Anwendungsfalls lassen sich *exklusive Vergleichseigenschaften* identifizieren, deren Ausprägungen so individuell sind, dass sich zwei relevante Interaktionen nur dann ähneln können, wenn sie über die gleichen Ausprägungen dieser Vergleichseigenschaften verfügen. Nicht jeder Anwendungsfall muss dabei über exklusive Vergleichseigenschaften verfügen. Die relevanten Interaktionen aus Anwendungsfall B besitzen beispielsweise keine Vergleichseigenschaft, deren unterschiedliche Ausprägungen automatisch den Schluss zuließen, dass diese zu unterschiedlichen kontextsensitiven Präferenzen führen. In der Tabelle 4.3 werden daher nur die exklusiven Vergleichseigenschaften der Anwendungsfälle A, C und D aufgelistet. Der Ereignistyp in der dritten Spalte identifiziert dabei genau das Ereignis einer relevanten Interaktion, dessen in Spalte vier beschriebener Ereignisparameter zu den exklusiven Vergleichseigenschaften gezählt wird. Abhängig vom Anwendungsfall lassen sich hier auch alternativ Kontextattribute als exklusive Vergleichseigenschaften spezifizieren. Da eine relevante Interaktion aus mehreren Ereignissen des gleichen Typs bestehen kann, wird in der prototypischen Implementierung ein eindeutiger Bezeichner für jedes zu erkennende Ereignis im Interaktionsmuster eingeführt. In diesen Beispielen ist jedoch eine eindeutige Identifizierung über den Ereignistyp möglich. Daher wird der Bezeichner in den beiden nachfolgenden Tabellen 4.3 und 4.4 der Übersichtlichkeit halber nicht aufgeführt.

Anwendungsfall	Vergleichseigenschaft	Ereignistyp des betreffenden Ereignis-Kontext-Paares	Ereignisparameter
A	Radiosender	RadiosenderWechseln	Radiosendername
C	Webseite	WebseiteAufrufen	URL
D	Telefonnr.	AnrufBeenden	Telefonnr.

Tabelle 4.3 Die in den Vorlagen zu definierenden exklusiven Vergleichseigenschaften der Anwendungsfälle A, C und D.

Neben der Definition exklusiver Vergleichseigenschaften gilt es weitere *nicht-exklusive Vergleichseigenschaften* der relevanten Interaktionen in den Vorlagen zu definieren, deren Ausprägungen sich innerhalb einer Gruppe ähnlicher relevanter Interaktionen nicht gleichen müssen. Die nicht-exklusiven Vergleichseigenschaften beziehen sich dabei in den meisten Fällen auf Kontextattribute, deren Werte aus einem kontinuierlichen Wertebereich stammen. Das Kontextattribut „Zeit“ aus Anwendungsfall D ist ein Beispiel einer solchen nicht-exklusiven Vergleichseigenschaft.

Der Eigenschaftsvektor besteht also nicht nur aus exklusiven, sondern auch aus nicht-exklusiven Vergleichseigenschaften. Der Eigenschaftsvektor aus Anwendungsfall A besteht beispielsweise zum einen aus der exklusiven Vergleichseigenschaft des Radiosendernamens und zum anderen aus der nicht-exklusiven Vergleichseigenschaft „Position“. Ähnliches gilt für den Anwendungsfall D, bei dem die exklusive Vergleichseigenschaft „Telefonnr.“ und die beiden nicht-exklusiven Vergleichseigenschaften „Straße“ und „Zeit“ Bestandteile des Eigenschaftsvektors sind.

Während sich die Vergleichseigenschaften im Anwendungsfall A und D nur auf die Ereignisparameter und Kontextattribute eines bestimmten 2-Tupels einer relevanten Interaktion beziehen, liegen die Vergleichseigenschaften in den Anwendungsfällen B und C über mehrere 2-Tupel einer relevanten Interaktion verteilt. Im Anwendungsfall B werden nämlich nicht nur die Positionen des Fahrzeugs zum Beginn der Fahrt, sondern auch die Po-

Anwendungsfall	Vergleichseigenschaft	Ereignistyp des betreffenden Ereignis-Kontext-Paares	Kontextattribut
A	Position	RadiosenderWechseln	Position
B	Startposition	FahrtStarten	Position
	Endposition	FahrtBeenden	
C	Startposition	FahrtStarten	Position
	Zwischenposition	WebseiteAufrufen	
D	Straße	AnrufStarten	Straße
	Zeit		Zeit

Tabelle 4.4 Die in den Vorlagen zu definierenden nicht-exklusiven Vergleichseigenschaften der Anwendungsfälle A bis D.

sitionen am Ende der Fahrt untersucht. Diese Werte ergeben sich jedoch aus zwei verschiedenen Kontextvektoren, die sich jeweils in dem 2-Tupel mit dem Ereignistyp „FahrtStarten“ und in dem 2-Tupel mit dem Ereignistyp „FahrtBeenden“ befinden. Folglich enthält der Eigenschaftsvektor die beiden nicht-exklusiven Vergleichseigenschaften „Startposition“ und „Endposition“. Die Ähnlichkeit zweier Eigenschaftsvektoren dieses Anwendungsfalls kann also davon abhängen, ob sich deren Start- und Endpositionen ähneln. Ähnliches gilt für den Anwendungsfall C, bei dem die Position des Fahrzeugs zum Beginn der Fahrt, die Position zum Zeitpunkt des Webseitenaufrufs und die exklusive Vergleichseigenschaft „Webseite “ im Eigenschaftsvektor enthalten sind. Aus Sicht des nachfolgend vorgestellten Clusterings ist die Herkunft der Vergleichseigenschaften jedoch irrelevant. In Tabelle 4.4 werden abschließend noch einmal alle nicht-exklusiven Vergleichseigenschaften zusammenfasst.

Distanz- und Ähnlichkeitsfunktionen im automotiven Bereich

Die Ähnlichkeit der Werte zweier Vergleichseigenschaften unterschiedlicher relevanter Interaktionen lässt sich, wie oben beschrieben, über eine Distanzfunktion ermitteln. Da sich die Wertebereiche der Ereignisparameter und Kontextattribute, auf die sich die Vergleichseigenschaften beziehen, jedoch stark unterscheiden können, gilt es, für jede nicht-exklusive Vergleichseigenschaft eine individuelle Distanzfunktion einzuführen. Für die Berechnung der Ähnlichkeit zweier Ausprägungen der nicht-exklusiven Vergleichseigenschaft „Zeit“ aus Anwendungsfall D eignet sich beispielsweise folgende Funktion:

$$f_{distance}(t_1, t_2) = |t_1 - t_2|$$

Die Ähnlichkeit zweier Zeitstempel unterschiedlicher relevanter Interaktionen ergibt sich demnach aus der Größe der Zeitdifferenz beider Zeitstempel. Um jedoch während der Gruppierung zwischen ähnlichen und nicht-ähnlichen Ausprägungen zu unterscheiden, wird zusätzlich eine maximale Distanz ε zwischen zwei Ausprägungen definiert. Zwei Ausprägungen o_1 und o_2 einer nicht-exklusiven Vergleichseigenschaft sind demnach per Definition ähnlich, wenn die Distanz zwischen ihnen das maximale Maß nicht übersteigt:

$$f_{simlar}(o_1, o_2, \varepsilon) = \begin{cases} true & \text{wenn } f_{distance}(o_1, o_2) \leq \varepsilon \\ false & \text{wenn } f_{distance}(o_1, o_2) > \varepsilon \end{cases}$$

Da nur einzelne Werte der Vergleichseigenschaften auf Ähnlichkeit hin überprüft werden, wird diese Abbildung hier als *lokale Ähnlichkeitsfunktion* bezeichnet. Die Zeitstempel zweier relevanter Interaktionen können sich somit z.B. ähneln, wenn sie nicht länger als 10 Minuten auseinander liegen.
Für die Berechnung der Distanz zweier Positionen aus den Anwendungsfällen A bis C eignen sich hingegen zwei unterschiedliche Distanzfunktionen. Eine Variante besteht darin, die direkte Entfernung beider Positionen ohne die Berücksichtigung von Hindernissen mit Hilfe der aus der Navigation bekannten *Haversine Formel* zu berechnen.

$$f_{distance}(p_1,p_2) = 2 * Erdradius * \sin^{-1}(\sqrt{\sin^2(a) + \cos(lat(p_1)) * \cos(lat(p_2)) * \sin^2(b)})$$

mit

$$a = \frac{lat(p_2) - lat(p_1)}{2} \text{ und } b = \frac{long(p_2) - long(p_1))}{2}$$

Der Längen- und Breitengrad einer Position p wird dabei mit $lat(p)$ und $long(p)$ angegeben. Zwei Positionen können somit ähnlich sein, wenn sie laut der Haversine Formel beispielsweise nicht weiter als 100 m voneinander entfernt sind.

Werden im Fahrzeug zwei relevante Interaktionen ermittelt, deren Positionen sich laut der Haversine Formel nur um wenige Meter unterscheiden, so kann es sich dennoch, trotz einer spezifizierten maximalen Distanz ε von 100 Meter, um zwei relevante Interaktionen handeln, die bzgl. der echten Entfernung nicht ähnlich sind. Da sich ein Fahrzeug im Gegensatz zu einem tragbaren Computer, wie einem Smartphone, nur innerhalb des Straßennetzes bewegen kann, lässt sich die Distanz zweier Positionen im automotiven Bereich präziser bestimmen, wenn nicht die direkte Entfernung, sondern die Streckenlänge des kürzesten Weges im Straßennetz als Distanz verwendet wird. Eine solche Distanz wird auch allgemein als *Netzwerk-Distanz* bezeichnet.

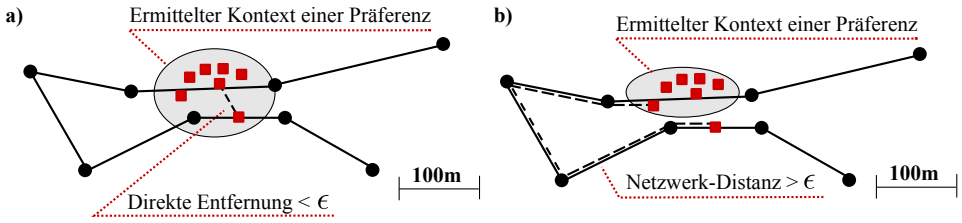


Abb. 4.8 Gegenüberstellung der Kontexte zweier Präferenzen, die auf Basis der selben relevanten Interaktionen und unter Zuhilfenahme unterschiedlicher Distanzfunktionen ermittelt wurden.

In Abbildung 4.8 wird veranschaulicht, inwiefern die Wahl einer Distanzfunktion das Ergebnis der Gruppierung und schließlich der gesamten kontextsensitiven Personalisierung beeinflusst. Dazu wurden mehrere Positionen relevanter Interaktionen auf einer Karte abgebildet. In a) sind alle Positionen aufgrund der direkten Entfernung ähnlich, während in b) eine Position aufgrund der Streckenlänge des kürzesten Weges keine Ähnlichkeit mit den restlichen Positionen aufweist. Eine temporäre Ungenauigkeit des GPS-Sensors könnte beispielsweise zu einer solchen ungewöhnlichen Positionsbestimmung geführt haben. In a) besteht der Kontext der daraus ermittelten kontextsensitiven Präferenz unter Berücksichtigung der direkten Entfernung folglich aus einer Region, die sich über zwei bis drei Straßenab-

schnitte erstreckt, obwohl in zwei der drei betreffenden Straßenabschnitte möglicherweise nie oder nur sehr selten relevante Interaktionen stattfinden. Die Ausführung der Personalisierung könnte also in einer Situation erfolgen, die der Benutzer nicht mit dieser Personalisierung assoziiert.

Eine Berechnung der Netzwerk-Distanz setzt jedoch voraus, dass das Straßennetz dem Bordcomputer als Modell z.B. in Form eines Graphen vorliegt. Moderne In-Car-Infotainment-Systeme erfüllen diese Bedingung, da sie größtenteils serienmäßig über ein Navigationssystem verfügen, dessen Kartenmaterial sich zur Berechnung der Netzwerk-Distanz eignet. Während die Berechnung der Distanz auf Grundlage der Haversine Formel mit einem konstanten Zeitaufwand verbunden ist, liegen die Algorithmen zur Suche nach dem kürzesten Weg zwischen zwei Punkten in einem Graphen, z.B. mit dem Dijkstra Algorithmus, in der Komplexitätsklasse $\mathcal{O}(n^2)$. Die Anzahl der Kanten n des Graphenmodells bestimmt dabei maßgeblich den Zeitaufwand. Hier gilt es also, abhängig von den zur Verfügung stehenden Ressourcen, beide Berechnungsmethoden gegeneinander abzuwägen.

Für die eindeutige Feststellung der Ähnlichkeit zweier Eigenschaftsvektoren ist es nun erforderlich, eine *globale Ähnlichkeitsfunktion* $f_{SIMILAR} : r \times r \rightarrow \{true, false\}$ einzuführen, die auf Basis aller Distanzfunktionen und Mindestabstände definiert, in welchem Fall sich zwei Eigenschaftsvektoren ähneln. Seien mit r_{1_1} und r_{2_1} die beiden ersten Vergleichseigenschaftswerte der Eigenschaftsvektoren r_1 und r_2 bezeichnet, so ergibt sich die globale Ähnlichkeitsfunktion allgemein wie folgt:

$$f_{SIMILAR}(r_1, r_2) = \begin{cases} true & \text{wenn } \forall r_{1_x}, r_{2_x}(f_{similar}(r_{1_x}, r_{2_x}) \wedge x \in \mathbb{N}^*) \\ false & \text{wenn } \exists r_{1_y}, r_{2_y}((\neg f_{similar}(r_{1_y}, r_{2_y})) \wedge y \in \mathbb{N}^*) \end{cases}$$

Demnach sind zwei Eigenschaftsvektoren per Definition ähnlich, wenn sich die Elemente aller Dimensionen paarweise laut der lokalen Ähnlichkeitsfunktionen ähneln. Anders als bei einer globalen Ähnlichkeitsfunktion, die auf einer mit einem Mindestmaß versehenen Manhattan-Distanz (siehe Unterabschnitt 2.3.3) basiert, werden die Distanzen hier pro Dimension und nicht über alle Dimensionen hinweg berechnet. Die einzelnen Distanzen müssen also nicht wie bei der Manhattan-Distanz in der Summe ein Mindestmaß einhalten, sondern sich pro Dimension an ein spezielles Mindestmaß halten. Damit entfällt das beim Clustering oftmals als Skaleninvarianz bezeichnete Problem, nachdem sich nur dann eine sinnvolle Summe der Distanzen berechnen lässt, wenn die Skalen der einzelnen Dimensionen normiert werden. Alle Distanzfunktionen und die dazugehörigen Mindestabstände ε werden für jede nicht-exklusive Vergleichseigenschaft separat in den Vorlagen spezifiziert.

Dichtebasiertes Clustering mit GDBSCAN

Die Eigenschaftsvektoren aller relevanten Interaktionen lassen sich auf Grundlage der in den Vorlagen definierten Distanzfunktionen und Mindestabstände aller nicht-exklusiven Vergleichseigenschaften miteinander vergleichen und gruppieren. Das hier gewählte Gruppierungsverfahren basiert auf dem von Ester u.a. in [Ester u. a., 1996] vorgestellten dichtebasierten Clustering-Algorithmus DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Die zu untersuchenden Elemente, die im Fall des hier vorgestellten Verfahrens den Eigenschaftsvektoren entsprechen, werden beim DBSCAN immer dann zu bestimmten Gruppen zusammengefasst, wenn sich mindestens j Elemente ähneln. Dies entspricht der oben genannten Bedingung, wonach relevante Interaktionen nur dann zu einer

kontextsensitiven Präferenz führen, wenn die Mitgliederzahl bzw. Anzahl an Eigenschaftsvektoren einer Gruppe einen bestimmten Wert ς übersteigt. Während sich eine Gruppe in einer Region mit hoher Dichte an Elementen befindet, liegen alle anderen Elemente, die keiner Gruppe zugeordnet werden können, in einem Gebiet mit geringer Dichte. Die Struktur der Ergebnisse von DBSCAN kann beliebige Formen annehmen und somit auf vielfältige Weise individuelle Präferenzen abbilden.

In DBSCAN werden drei Arten von Elementen identifiziert: *Kern-*, *Grenz-* und *Rauschelemente*. Ein Kernelement zeichnet sich dadurch aus, dass sich in der direkten *Nachbarschaft*, die durch einen vordefinierten Radius aufgespannt wird, eine Mindestanzahl j von Elementen befinden. Damit ist ein Kernelement ein zentraler Bestandteil einer Gruppe. Sei mit R die Menge aller Eigenschaftsvektoren gegeben, dann setzt sich die Nachbarschaft $N()$ eines Eigenschaftsvektors r_x wie folgt zusammen:

$$N(r_x) = \{r : R | f_{SIMILAR}(r_x, r) \wedge r_x \neq r\} \tag{4.2}$$

Eine solche Definition der Nachbarschaft, die nicht auf einem Vergleich mit einem global gültigen maximalen Radius, sondern auf einem beliebigen Prädikat $f_{SIMILAR}$ basiert, kommt in dem erweiterten Algorithmus GDBSCAN (Generalized DBSCAN)[Sander u. a., 1998] zum Einsatz. Ein Element r ist somit ein Kernelement, wenn gilt: $|N(r)| \geq j$. Mit dem Begriff Grenzelement wird ein Element bezeichnet, welches sich am Rand einer Gruppe befindet. Somit gilt für ein beliebiges Grenzelement r_x folgendes: $0 < |N(r_x)| < j$ und $\exists r_y \in N(r_x) | N(r_y) \geq j$. Alle anderen Elemente, die weder Kern- noch Grenzelemente sind werden als Rauschelemente bezeichnet. In [Ester u. a., 1996] werden außerdem drei Arten von Beziehungen definiert, die zwischen zwei beliebigen Elementen einer Gruppe bestehen können:

- Ein Element r_x ist vom einem Element r_y *direkt dichte-erreichbar*($dde(r_x, r_y)$), wenn gilt: $r_x \in N(r_y)$ und $|N(r_y)| \geq j$

- Ein Element r_x ist vom einem Element r_y *dichte-erreichbar*($de(r_x, r_y)$), wenn eine Kette von Elementen $r_1, ..., r_n$ mit $r_1 = r_y$ und $r_n = r_x$ existiert, so das gilt: r_{i+1} ist direkt dichte-erreichbar von r_i

- Ein Element r_x ist mit einem Element r_y *dichte-verbunden*($dv(r_x, r_y)$), wenn ein Element r_z existiert, von dem beide Elemente r_x und r_y dichte-erreichbar sind.

Für eine Gruppe bzw. ein Cluster C gilt somit: 1. $\forall r_x, r_y \in R | r_x \in C \wedge de(r_y, r_x) \rightarrow r_y \in C$ und 2. $\forall r_x, r_y \in C | dv(r_x, r_y)$. Auf Grundlage dieser Definitionen ergibt sich der GDBSCAN-Algorithmus zur Ermittlung aller Cluster wie im Pseudocode 4.1 beschrieben. Der GDBSCAN-Algorithmus durchläuft demnach alle zu untersuchenden Elemente und erstellt bzw. expandiert ein Cluster, sobald es sich bei einem Element um ein Kernelement handelt. Das Ergebnis der Anwendung von GDBSCAN zur Gruppierung von Eigenschaftsvektoren besteht zum einen aus verschiedenen Mengen, die jeweils die Eigenschaftsvektoren einer Gruppe umfassen, und zum anderen aus der Menge aller Eigenschaftsvektoren, deren relevante Interaktionen nur selten auftraten. Um eine Gruppe von Eigenschaftsvektoren auf einen eindeutigen Gruppenvektor abzubilden, der die typischen Eigenschaften der Gruppe repräsentiert, bedarf es zunächst einer Zusammenlegung der Werte nicht-exklusiver Vergleichseigenschaften. Die Zusammenlegung kann durch eine Aufzählung aller Aus-

```
Besuche jedes e aus R
  Wenn Status(e) = 'Unbekannt', dann
    ne := berechneNachbarschaft(e)
    Wenn (|ne| < j), dann  //Ist es kein Kernelement?
      Status(e) := 'Rauschen'
    sonst
      cid := erzeugeNeueClusterID
      Status(e) := cid
      Setzte den Status aller Elemente von ne auf cid
      Solange |ne| > 0 gilt, besuche jedes s aus ne
        ns := berechneNachbarschaft(s)
        Wenn (|ns|>= j), dann
          Besuche jedes t aus ns
            Wenn Status(t) = 'Unbekannt|Rauschen', dann
              Wenn Status(t) = 'Unbekannt', dann
                Füge t der Menge ne hinzu
              Status(t) := cid
        lösche s aus der Menge ne
```

Quelltext 4.1 Pseudocode des GDBSCAN-Algorithmus.

prägungen einer bestimmten nicht-exklusiven Vergleichseigenschaft (Liste von Nominalwerten) oder z.B. durch die Berechnung eines Intervalls erfolgen. Während sich die Zeit im Anwendungsfall D zu einem Intervall zusammenfassen lässt, werden die Straßen in einer Liste von Nominalwerten aufgezählt. Letztendlich besteht ein Gruppenvektor also aus den zusammengefassten nicht-exklusiven Vergleichseigenschaften und den individuellen Ausprägungen der exklusiven Vergleichseigenschaften. In Abbildung 4.9 werden vier beispielhafte Gruppenvektoren der Anwendungsfälle A bis D aufgezeigt. Die Positionen in a), b) und c) spannen rechteckige Regionen[2] auf, die im Nordwesten durch die jeweils erste Position und im Südosten durch die jeweils zweite Position gegeben sind.

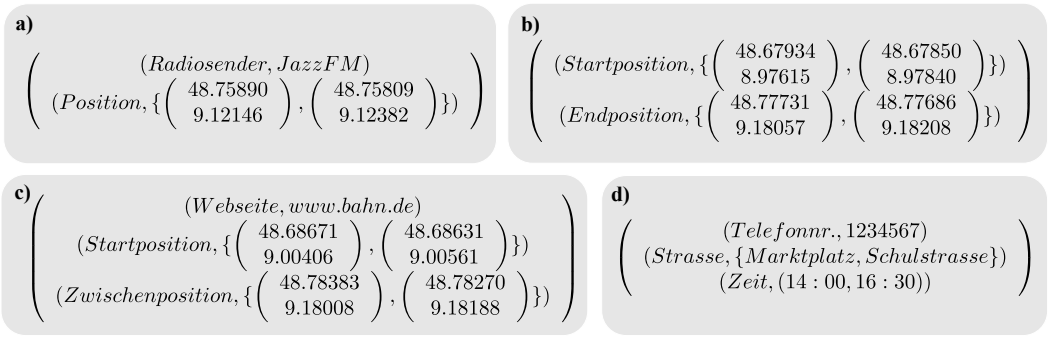


Abb. 4.9 Beispielhafte Gruppenvektoren unter a) für Anwendungsfall A, unter b) für Anwendungsfall B, unter c) für Anwendungsfall C und unter d) für Anwendungsfall D.

[2] Es wird angenommen, dass das Verhältnis zwischen der Größe der Region, in dem sich ein Fahrzeug üblicherweise aufhält, und der Fläche der Erde so klein ist, dass die unterschiedlichen Distanzen zwischen zwei Längen- und Breitengraden auf unterschiedlichen Breiten- und Längengraden zu vernachlässigen sind.

4.2.3.2 Identifikation neuer und veränderter Gruppen

Der letzte Verarbeitungsschritt in diesem Prozess besteht in der Gegenüberstellung der aktuellen Gruppeneigenschaften in Form der ermittelten Gruppenvektoren mit den Gruppeneigenschaften, die sich aus der vorherigen Gruppierung ergaben. Da durch ein Clustering eine neue und von dem letzten Clustering unabhängig ermittelte Menge von Gruppenvektoren vorliegt, gilt es bei dieser Untersuchung die zeitlichen Veränderungen einzelner Gruppen zu identifizieren. Dazu müssen die neuen und bisher unbenannten Gruppenvektoren mit den alten benannten Gruppenvektoren in Beziehung zueinander gesetzt werden. Hierbei muss untersucht werden, ob sich eine neue unbenannte Gruppe auf eine ältere benannte Gruppe zurückführen lässt, oder ob es sich um eine gänzlich neue kontextsensitive Präferenz handelt.

Motiviert wird dieser Schritt durch die Möglichkeit, einzelne Gruppen aus Sicht der Benutzer identifizierbar zu machen. Die bereits in Unterabschnitt 2.3.4 vorgestellte Feedback-Möglichkeit erlaubt es dem Benutzer beispielsweise, bestimmte und vom System vorgeschlagene kontextsensitive Präferenzen zu bestätigen bzw. zu aktivieren. Diese führen also auf expliziten Wunsch den Benutzers in der geeigneten Situation immer zu einer kontextsensitiven Personalisierung. Verschiebt sich beispielsweise die bevorzugte Region einer solchen Präferenz in Richtung eines benachbarten Gebiets, so muss das System in der Lage sein, diese Änderung auch zu erkennen. Andernfalls würde das System nach jeder Gruppierung neue Empfehlungen kontextsensitiver Präferenzen aussprechen, deren Ausprägungen sich vielleicht nur in subtiler Weise von denen der alten kontextsensitiven Präferenzen unterscheiden. Bei den kontextsensitiv personalisierten Schnellstartbuttons ohne Feedback ist die Untersuchung der Evolution der Gruppen jedoch z.B. nicht notwendig, da die dortigen Präferenzen in jedem Fall zu einer Personalisierung der Schnellstartbuttons führen. Die Empfehlungen des Systemdienstes, eine bestimmte Funktionalität über einen Schnellstartbutton anzubieten, werden hier also vom System direkt akzeptiert und umgesetzt. In diesem Fall werden die neuen und unbenannten Gruppenvektoren direkt an den nächsten Prozess des Verfahrens weitergegeben.

Die Ausprägungen der typischen Eigenschaften einer Gruppe, im Nachfolgenden durch deren Wertebereiche gegeben, können sich über die Zeit hinweg auf fünf mögliche Arten verändern. Zu den häufigsten Veränderungen zählen die *Vergrößerung* und *Verkleinerung* der Wertebereiche. Im ersten Fall kommen neue Eigenschaftsvektoren zu einer Gruppe hinzu, deren Ausprägungen der nicht-exklusiven Vergleichseigenschaften den Wertebereich ein oder mehrerer typischer Eigenschaften erweitern, während im zweiten Fall, aufgrund des künstlichen Verfalls bestimmter Eigenschaftsvektoren, die Wertebereiche wieder verkleinert werden. Bei der *Verschiebung* des Wertebereichs wird ein Teilbereich verkleinert und ein anderer Teilbereich gleichzeitig vergrößert. Bei allen drei Formen der Veränderung wird angenommen, dass sich die Wertebereiche der Eigenschaften der neuen und unbenannten Gruppe mit denen der Eigenschaften einer alten benannten Gruppe überschneiden. Eine neue Gruppe lässt sich somit durch die Überschneidung der Wertebereiche einzelner Eigenschaften auf eine alte benannte Gruppe zurückführen. Die drei Formen der Veränderung liegen jedoch nur dann eindeutig vor, wenn sich die Wertebereiche der alten Gruppe mit keinem Wertebereich einer weiteren neuen Gruppe überschneiden. Sollte dies dennoch der Fall sein, so entscheiden die Größen der verschiedenen Schnittmengen jeweils darüber, ob es sich um eine der drei oben genannten Formen der Veränderung oder um eine *Verschmelzung* oder *Aufspaltung* handelt.

Eine *Verschmelzung* zweier alter Gruppen ist dann gegeben, wenn sich die Wertebereiche der Eigenschaften beider Gruppen mit den Wertebereichen der Eigenschaften einer neuen Gruppe schneiden. Befinden sich die Wertebereiche der Eigenschaften beider alter Gruppen dabei vollständig innerhalb des Wertebereichs der Eigenschaften der neuen Gruppe, dann kann wahlweise der Identifikator der einen oder der anderen alten Gruppe von der neuen Gruppe übernommen werden. Ist dies nicht gegeben, so muss außerdem überprüft werden, ob analog zu den vorherigen Formen der Veränderung, Schnittmengen mit weiteren neuen Gruppen existieren. Kann dies ausgeschlossen werden, so wird wie bei der vollständigen Inklusion vorgegangen. Andernfalls wird analog ein Größenvergleich der betreffenden Schnittmengen durchgeführt (siehe a) in Abbildung 4.10).

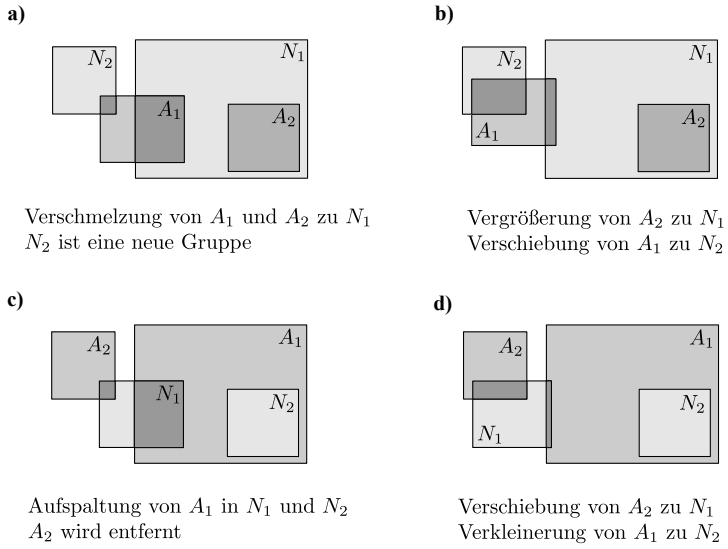


Abb. 4.10 Veränderungen der Gruppeneigenschaft „Position“ anhand der Gegenüberstellung der Kontexte der neuen Gruppen (N_1 und N_2) und alten Gruppen (A_1 und A_2).

Das *Aufspalten* der Wertebereiche einer alten Gruppe in Wertebereiche zweier neuer Gruppen wird immer dann ausgeführt, wenn die beiden neuen Gruppen nur Schnittmengen mit dem Wertebereich einer alten Gruppe besitzen. In diesen Fällen wird ein Identifikator übernommen und ein weiterer generiert. Sollte eine Schnittmenge mit einer weiteren alten Gruppe vorliegen, dann wird ähnlich wie in den vorherigen Fällen auf Basis der Größe der Schnittmenge über die Form der Veränderung entschieden.

In diesem Verarbeitungsschritt werden also letztendlich alle Schnittmengen der neuen und alten Gruppen berechnet und ggf. deren Größen verglichen, damit neue Gruppen identifiziert und die Veränderungen der alten Gruppen erkannt werden können. Das Ergebnis besteht aus einer Menge benannter Gruppenvektoren, auf deren Grundlage im nachfolgenden Prozess die passenden Situationen zum Personalisieren erkannt werden.

4.2.4 Erkennung der Situationen kontextsensitiver Präferenzen

Die ermittelten Gruppenvektoren enthalten implizite Informationen über die Situationen, in denen die zugehörigen Präferenzen bevorzugt werden. Daher lassen sich aus den Grup-

penvektoren situative Bedingungen ableiten, die eine Situation erfüllen muss, damit es zu einer Personalisierung kommt. Der letzte Schritt der kontextsensitiven Personalisierung besteht also darin, die aktuelle Situation des Systems laufend daraufhin zu untersuchen, ob sie die situativen Bedingungen einer kontextsensitiven Präferenz erfüllt. Ist dies der Fall, so wird beispielsweise das Autoradio im Anwendungsfall A vom Systemdienst automatisiert dazu aufgefordert, den entsprechenden Radiosender einzustellen. Im Gegensatz zu den im Unterabschnitt 3.1.2 vorgestellten und vorzugsweise auf Markov-Modellen basierenden Verfahren wird hier also erst dann personalisiert, wenn - ähnlich wie bei der Anwendung des fallbasierten Schließens - die geeignete Situation bereits eingetreten ist. Es wird also nicht versucht, die nächste zu erwartende Situation abzuschätzen und bereits zuvor die sich daraus abzuleitende kontextsensitive Einstellung automatisiert zu konfigurieren, sondern erst dann die Benutzerschnittstelle anzupassen, wenn die Situation alle situativen Bedingungen einer kontextsensitiven Präferenz erfüllt. Dadurch kommt es in diesem Verfahren zu keinen Personalisierungen, die auf einer falschen Prognose bzgl. zukünftig zu erwartender Situationen basieren. Zur Überprüfung der aktuellen Situation bzgl. der situativen Bedingungen einer kontextsensitiven Präferenz müssen jedoch folgende Voraussetzungen erfüllt sein:

- Da das System ohne zusätzliche Informationen nicht in der Lage ist, die situativen Bedingungen einer kontextsensitiven Präferenz eigenhändig von den zur Konfiguration notwendigen Einstellungen zu unterscheiden, ist es für eine automatische Herleitung der situativen Bedingungen unabdingbar, jede Eigenschaft einer Gruppe in den Vorlagen entweder der Konfiguration oder den situativen Bedingungen zuzuordnen. Beispielsweise muss für den Anwendungsfall A in den Vorlagen eindeutig deklariert sein, dass die Gruppeneigenschaft „Position“ eine situative Bedingung vorgibt und dass es sich beim „Radiosender“ um eine für die Konfiguration relevante Eigenschaft handelt.

- Etwaige temporale Beziehungen zwischen den situativen Bedingungen bleiben dem System ohne eine manuelle Definition in den Vorlagen ebenfalls verborgen. Im Anwendungsfall C muss beispielsweise überprüft werden, ob sich die aktuelle Situation auf zwei zeitlich nacheinander aufgetretene Kontexte zurückführen lässt, die jeweils eine der zwei situativen Bedingungen „Startposition“ und „Zwischenposition“ erfüllen. Während der erste Kontext eine Position enthalten muss, die sich innerhalb der durch die Startposition aufgespannten Region befindet, muss sich die Position im zweiten Kontext innerhalb der Region befinden, die durch die Zwischenposition aufgespannt wird. Die URL dient lediglich zur Konfiguration der Personalisierung.

- Die sich aus den kontextsensitiven Präferenzen ergebenden situativen Bedingungen erfüllen nicht immer alle Eigenschaften, derer es zur Ausführung einer Personalisierung im automotiven Bereich bedarf. Im Anwendungsfall A würde der Systemdienst im Fall, dass die Situation alle situativen Bedingungen erfüllt, das Autoradio auch dann zu einem Wechsel des Radiosenders auffordern, wenn der Radiosender bereits eingestellt ist. Eine Vermeidung solcher Schwachstellen lässt sich nur durch den Einsatz zusätzlicher und manuell zu definierender Nebenbedingungen bewerkstelligen. Die aktuelle Situation sollte also nicht nur die situativen Bedingungen, sondern auch die Nebenbedingungen erfüllen. Damit ließen sich auch rechtliche Bestimmungen im automotiven Bereich, wie z.B. das nicht gestattete Abspielen von Videos während der Fahrt, an geeigneter Stelle in den Prozess der Personalisierung einarbeiten.

Innerhalb des Systemdienstes lässt sich die aktuelle Situation aus der aktuellen Sequenz von Ereignis-Kontext-Paaren herleiten. Liegt nun eine kontextsensitive Präferenz eines Anwendungsfalls vor, so gilt es, zur Laufzeit die aktuelle Situation bzgl. der oben genannten Voraussetzungen zu untersuchen. Es wird also eine Sequenz von Ereignis-Kontext-Paaren gesucht, die alle Bedingungen der kontextsensitiven Präferenz und der zusätzlichen Nebenbedingungen erfüllt. Analog zu den Interaktionsmustern aus Unterabschnitt 4.2.2, eignet sich hierzu die Formulierung eines sequentiellen Musters auf der Basis der zuvor eingeführten EPL. Damit lassen sich alle Bedingungen einer gesuchten Situation innerhalb eines sogenannten *Situationsmusters* in den Vorlagen kompakt formulieren und mit Hilfe des Pattern Matching in der Sequenz von Ereignis-Kontext-Paaren erkennen. Es wird also laufend überprüft, ob die aktuelle Situation dem vorgegebenen Situationsmuster entspricht.

Da die spezifischen Bedingungen der kontextsensitiven Präferenzen jedoch erst zur Laufzeit ermittelt werden, stehen Platzhalter in den Situationsmustern stellvertretend für die erst zur Laufzeit ermittelten situativen Bedingungen. Bei den in den Vorlagen zu definierenden sequentiellen Mustern handelt es sich also um generische Situationsmuster, deren Konkretisierung erst zur Laufzeit stattfindet. Für den Anwendungsfall A lässt sich beispielsweise das folgende generische Situationsmuster formulieren:

```
#(Position[$Position])
```

Das Dollarzeichen dient hierbei zur Einführung eines Platzhalters, der sich in diesem Fall auf die Gruppeneigenschaft mit Namen „Position“ bezieht. Somit wird jedes Ereignis vom Typ „Position“ darauf hin untersucht, ob es die Parameterfilter erfüllt, die sich aus der Gruppeneigenschaft mit Namen „Position“ ergeben. Durch den Platzhalter wird also implizit festgelegt, welche Gruppeneigenschaften den situativen Bedingungen zugeordnet werden. Liegt eine kontextsensitive Präferenz wie in Abbildung 4.9 a) vor, so wird zur Laufzeit das oben beschriebene generische Situationsmuster folgendermaßen konkretisiert:

```
#(Position[Breitengrad≤"48.75890" && Längengrad≥"9.12146"
      && Breitengrad≥"48.75809" && Längengrad≤"9.12382"])
```

Damit ist der Systemdienst in der Lage, das Autoradio immer dann zu einem Radiosenderwechsel aufzufordern, wenn ein Ereignis vom Typ „Position“ vorliegt, dessen Position - bestehend aus Längen- und Breitengrad - sich innerhalb der gesuchten Region befindet. Die Form der Konkretisierung hängt dabei vom Typ der Ausprägung der Gruppeneigenschaft ab. Da es sich im Anwendungsfall A um eine Liste zusammengesetzter Werte handelt, muss die Funktionsvorschrift für die Konkretisierung vordefiniert sein. In Anwendungsfall D bestehen die Werte der Gruppeneigenschaften „Straße“ und „Zeit“ jedoch beispielsweise aus einer Liste von Nominalwerten und einem Intervall. Ein generisches Situationsmuster könnte wie folgt aussehen:

```
#(Context[$Strasse && $Zeit])
```

Anders als bei den bisher aufgeführten Interaktions- oder Situationsmustern wird hier nicht ein Ereignis, sondern der aktuelle Kontextvektor untersucht. Dieser kann jedoch auf die gleiche Art und Weise wie bei der Untersuchung eines Ereignisses über die bekannten Operatoren der EPL angesprochen werden. In Kombination mit der kontextsensitiven Präferenz aus Abbildung 4.9 d) ergibt sich das folgende konkrete Situationsmuster:

```
#(Context[(Strasse="Marktplatz" || Strasse="Schulstrasse")
                    && (Zeit>="14:00" && Zeit<="16:00")])
```

Die Konkretisierung einer situativen Bedingung, deren Gruppeneigenschaft eine Liste von Nominalwerten enthält, findet durch eine disjunktive Verknüpfung der Überprüfungen aller Nominalwerte statt. Besteht der Wert einer Gruppeneigenschaft aus einem Intervall, so ergibt sich die situative Bedingung erwartungsgemäß als eine Überprüfung auf Einhaltung der zugelassenen Maximal- und Minimalwerte. Während die Konkretisierung der Bedingungen bei Wahrheitswerten, Listen von Nominalwerten und Intervallen ohne zusätzliche Anweisungen automatisiert ausgeführt werden kann, ist eine automatische Konkretisierung der Bedingungen im Fall von Listen zusammengesetzter Werte nur über eine separate Definition der Funktionsvorschriften möglich.

Mit Hilfe der temporalen Operatoren ist es außerdem möglich die temporalen Beziehungen zwischen den verschiedenen situativen Bedingungen einer kontextsensitiven Präferenz ordnungsgemäß in einem generischen Situationsmuster zu berücksichtigen. Wie bereits oben erwähnt, muss sich eine für die Personalisierung in Anwendungsfall C geeignete Situation aus zwei zeitlich nacheinander auftretenden Kontexten herleiten können. Für jedes dieser Kontexte müssen dabei unterschiedliche situative Bedingungen gelten. Ein generisches Situationsmuster für diesen Anwendungsfall könnte z.B. wie folgt aufgebaut sein:

```
#( Position[$Startposition] → Position[$Zwischenposition])
```

Im Anwendungsfall C liegt also immer eine für die Personalisierung relevante Situation vor, wenn das Fahrzeug zunächst auf dem Parkplatz des Büros gestartet wird und sich im Laufe der Fahrt dem Bahnhof nähert. Dadurch wird ausgeschlossen, dass sich der Webbrowser schon auf der Hinfahrt während des kurzen Halts am Bahnhof fälschlicherweise automatisch öffnet. In diesem Verfahren kann die Ausführung einer kontextsensitiven Personalisierung also nicht nur von einem zeit-fixierten Zustand, sondern auch von den historischen Zuständen der Sensoren abhängen.

Zusätzliche Nebenbedingungen lassen sich z.b. innerhalb der Parameterlisten durch separate Bedingungen in das generische Situationsmuster integrieren. Das um eine zusätzliche Nebenbedingung erweiterte generische Situationsmuster aus Anwendungsfall A ergibt sich demnach wie folgt:

```
#(Context[$Position && !Radiosender])
```

Das Autoradio darf vom Systemdienst demnach nur dann dazu aufgefordert werden, auf den präferierten Radiosender zu wechseln, wenn sich das Fahrzeug innerhalb der ermittelten Region befindet und der aktuell eingestellte Radiosender nicht dem Radiosender entspricht, der sich aus der kontextsensitiven Präferenz ergibt.

Während bei der kontextsensitiv personalisierten Automatisierung die Personalisierung stets unmittelbar am Beginn des Zeitraumes stattfindet, ab dem das Situationsmuster gilt, besteht die Personalisierung bei den kontextsensitiv personalisierten Schnellstartbuttons aus zwei unterschiedlichen Anpassungen, die jeweils am Beginn und am Ende diesen Zeitraumes ausgeführt werden. Zu Beginn wird der entsprechende Schnellstartbutton angezeigt und am Ende wieder aus der Liste der Schnellstartbuttons entfernt. Damit wird gewährleistet, dass der Schnellstartbutton nur innerhalb der entsprechenden Situation angezeigt wird. Zur planmäßigen Ausführung beider Formen der kontextsensitiven Personalisierung automotiver Benutzerschnittstellen ist es daher notwendig, dass der Systemdienst sowohl den Beginn als auch das Ende einer sich aus der kontextsensitiven Präferenz ableitenden Situation meldet. Das oben beschriebene generische Situationsmuster aus Anwendungsfall D führt jedoch beispielsweise dazu, dass der Systemdienst bei jedem Kontextvektor eines neuen Ereignis-Kontext-Paares, der die beiden situativen Bedingungen erfüllt, die Situation erneut erkennt

und jeweils eine Meldung an die Liste der Schnellstartbuttons weiterleitet. Nur die Aufteilung in zwei unterschiedliche generische Situationsmuster ermöglicht es in diesem Fall, die beiden unterschiedlichen Zeitpunkte der gesuchten Situation zu differenzieren. Der Beginn der gesuchten Situation wird dabei mit Hilfe des generischen Situationsmusters

```
#(Context[!($Strasse) || !($Zeit)]) →
        (Context[$Strasse && $Zeit]
              & ¬ Context[!($Strasse) || !($Zeit)])
```

erkannt, während sich das Ende der gesuchten Situation durch das generische Situationsmuster

```
#(Context[$Strasse && $Zeit]) →
        (Context[!($Strasse) || !($Zeit)]
              & ¬ Context[$Strasse && $Zeit])
```

beschreiben lässt. Das erste generische Situationsmuster führt demnach erst dann zu einer Erkennung bzw. Meldung, wenn sich das Fahrzeug in der gesuchten Situation befindet und sich zum Zeitpunkt des Vorfalls des vorherigen Ereignisses nicht in der Situation befand. Die Situation ist also neu eingetreten. Der umgekehrte Fall wird im zweiten generischen Situationsmuster formuliert. In den Vorlagen können also nicht nur mehrere Interaktionsmuster, sondern auch mehrere Situationsmuster für den gleichen Anwendungsfall definiert werden. In Tabelle 4.5 werden alle generischen Situationsmuster der Anwendungsfälle A bis D zusammenfassend aufgeführt.

Anwendungsfall	Zeitpunkt	Situationsmuster
A	Beginn	`#(Position[!$Position]) → (Position[$Position] & ¬ Position[!$Position])`
B	Beginn	`Position[$Startposition]`
	Ende	`Position[$Startposition] → Position[!$Startposition]`
C	Beginn	`#(Position[$Startposition]) → (Position[$Zwischenposition] & ¬Position[$Startposition])`
D	Beginn	`#(Context[!($Strasse) \|\| !($Zeit)]) → (Context[$Strasse && $Zeit] & ¬ Context[!($Strasse) || !($Zeit)])`
	Ende	`#(Context[$Strasse && $Zeit]) → (Context[!($Strasse) || !($Zeit)] & ¬ Context[$Strasse && $Zeit])`

Tabelle 4.5 Die in den Vorlagen zu definierenden generischen Situationsmuster der Anwendungsfälle A bis D.

4.2.5 Betrachtung der Zeit- und Speicherkomplexitäten

Jeder Verarbeitungsschritt des Verfahrens kennzeichnet sich durch eine individuelle Zeit- und Speicherkomplexität, die es nun im Einzelnen zu untersuchen gilt. Der gesamte Zeit-

raum zwischen dem Beginn der Analyse des Benutzerverhaltens und der Ausführung einer Personalisierung wird dazu in bestimmte Zeitabschnitte unterteilt, für die im Bereich eingebetteter Systeme üblicherweise jeweils individuelle Beschränkungen bzgl. der maximalen Verarbeitungsdauer und des maximalen Speicherbedarfs gelten. In Abbildung 4.11 werden diese Zeitabschnitte am Beispiel der kontextsensitiven Präferenz und Personalisierung des ortsabhängigen Radiosenderwechsels auf einer Zeitleiste abgetragen. In diesem Beispiel wird die kontextsensitive Personalisierung der Benutzerschnittstelle ausgeführt, wenn sich mindestens zwei relevante Interaktionen bzgl. ihrer Eigenschaften ähneln. Die für diesen Anwendungsfall irrelevanten Ereignisse werden in der Abbildung nicht dargestellt. Außerdem bleiben die Zeit- und Speicherkomplexitäten, die sich innerhalb der Sensoren während der Abtastung oder bei der Ausführung der Anpassung ergeben, in den nachfolgenden Untersuchungen unberücksichtigt.

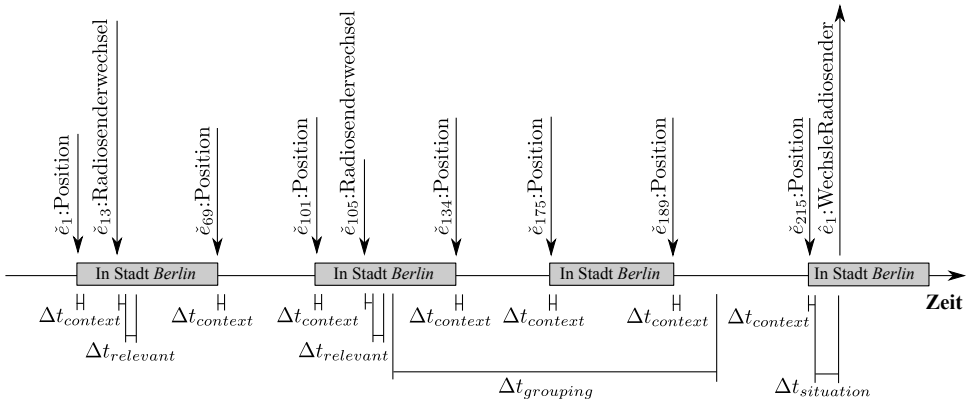


Abb. 4.11 Die Verarbeitungsdauern der einzelnen Verarbeitungsschritte des Verfahrens der kontextsensitiven Personalisierung.

Ermittlung des Kontextes

Da während des Verfahrens eine unendliche Sequenz von Ereignissen verarbeitet wird, beziehen sich nachfolgende Betrachtungen immer auf ein zu untersuchendes festes Zeitintervall τ. Die maximale Anzahl an Ereignissen innerhalb eines beliebig positionierten Zeitintervalls τ wird mit κ_τ angegeben. Die Verarbeitungsdauer eines Ereignisses zum Zweck der Ermittlung des Kontextes $\Delta t_{context}$ ergibt sich aus der Zeitkomplexität der Funktion $f_{context}$, die zum einen aus der Überprüfung des Ereignistyps auf Relevanz mit einem Aufwand von $T_{types}(\kappa_{types})$ (κ_{types} ist die Anzahl der in den Vorlagen definierten relevanten Ereignistypen) und zum anderen aus der Zeitkomplexität der Funktion $f_{edit_context}$ besteht. Wird nun im Worst-Case-Fall angenommen, dass sich die Anzahl der Ereignisparameter κ_{param} des größtmöglichen Ereignisses und die Anzahl der Kontextattribute des Kontextvektors gleichen, und wird jedem Kontextattribut der Wert eines Ereignisparameters mit der konstanten Komplexität c_{alloc} zugewiesen, so ergibt sich der gesamte Zeitbedarf der Verarbeitung jedes Ereignisses zu $T_{f_{context}} = T_{types}(\kappa_{types}) + \kappa_{param} * c_{alloc}$. Da jedes Ereignis der Teilsequenz in τ genau einmal untersucht wird, liegt dieser Prozess in der Komplexitätsklasse $\mathcal{O}(\kappa_\tau * (\kappa_{types} + \kappa_{param}))$.

Um eine akkumulative Verzögerung bei der Ermittlung des Kontextes über das Zeitintervall hinaus zu vermeiden, muss die maximale Anzahl der Ereignisse pro Zeitein-

heit κ_τ durch $\tau / T_{f_{context}}$ nach oben beschränkt werden. Treten alle Ereignisse gehäuft zu Beginn von τ auf, so ergibt sich die maximale Verarbeitungsdauer im Worst-Case, unter der Voraussetzung, dass Ereignisse nicht gleichzeitig verarbeitet werden können, zu $max(\Delta t_{context}) = \kappa_\tau * T_{f_{context}}$. Bevor es zur Verarbeitung der Ereignisse kommt, werden diese zunächst in einem Zwischenspeicher abgelegt. Der maximale Speicherbedarf ergibt sich folglich aus der maximalen Anzahl von Ereignissen κ_τ multipliziert mit dem Speicherbedarf des größtmöglichen Ereignisses. Bei diesem Verfahren ist die Wahl einer kurzen maximalen Verarbeitungsdauer kontextrelevanter Ereignisse zwingend notwendig, um in den nachfolgenden Prozessen eine zeitnahe Erkennung der Interaktionen und Situationen zu gewährleisten. Bei Nichtbeachtung einer schnellen Verarbeitung des Kontextes könnte es in unserem Beispiel dazu kommen, dass das Radio erst dann zum Wechseln des Radiosenders aufgefordert wird, wenn das Fahrzeug die Stadt *Berlin* bereits wieder verlassen hat.

Erkennung relevanter Interaktionen und Situationen

Während der Ermittlung relevanter Interaktionen wird, genauso wie bei der Ermittlung des Kontextes, jedes Ereignis in τ genau einmal untersucht. Die zeitliche Komplexität jeder einzelnen Untersuchung hängt in diesem Fall von diversen Faktoren, wie z.B. der Anzahl zu suchender Interaktionsmuster, Anzahl der Parameterfilter und der logischen und temporalen Struktur der Interaktionsmuster ab. Da eine genaue Untersuchung insbesondere der Zeitkomplexität des Pattern Matching bzgl. einzelner Klassen von Interaktionsmustern und deren Operatoren den Umfang dieser Forschungsarbeit sprengen würde, sei der interessierte Leser an dieser Stelle auf die detaillierte Analyse der Zeitkomplexität datenstrombasierter Datenverarbeitung von Mak [Mak, 2008] verwiesen. Welche maximale Verarbeitungsdauer $max(\Delta t_{relevant})$ im Einzelfall, ohne die Betrachtung der Interna, zulässig ist, hängt davon ab, ob dem Benutzer ein Feedback unmittelbar nach der Ausführung einer relevanten Interaktion angeboten werden soll. Bei einem solchen Feedback könnte es sich beispielsweise um ein Icon in der grafischen Benutzerschnittstelle handeln, welches immer dann kurzzeitig aufleuchtet, wenn eine relevante Interaktion erkannt wurde. Die maximale Differenz zwischen der künstlichen Zeit eines Ereignisses und der echten Zeit $\Delta t_{relevant}$ muss in diesem Fall also möglichst gering gehalten werden. Ist kein Feedback vorgesehen, so muss die maximale Verarbeitungsdauer keine Echtzeitbedingungen erfüllen, da relevante Interaktionen auch noch nach einer Fahrt berechnet werden können.

Gleichgültig, welcher algorithmische Ansatz beim Pattern Matching verfolgt wird, ist es in jedem Fall nötig, die für den Anwendungsfall relevanten Ereignisse während der Dauer der Untersuchung einer möglicherweise relevanten Teilsequenz in einem Zwischenspeicher abzulegen. Im Anwendungsfall B ist es während des Pattern Matchings beispielsweise nötig, das erste Ereignis vom Typ „FahrtStarten“ mindestens so lange im Zwischenspeicher zu halten, bis entweder ein Ereignis vom Typ „FahrtBeenden“erkannt wird oder bis das Ende des maximalen Zeitrahmens erreicht ist. Alle Ereignisse dazwischen werden aufgrund ihrer Irrelevanz bzgl. dieses Anwendungsfalls wieder verworfen und benötigen daher keinen zusätzlichen Speicherplatz. Wird also ein beliebiges Interaktionsmuster beim Pattern Matching verwendet, das über keinen Subausdruck-Operator verfügt, so ergibt sich die maximale Größe des notwendigen Zwischenspeichers $S_{interact}$ aus der maximalen Anzahl an Ereignissen κ_τ multipliziert mit der maximalen Größe eines Ereignisses s_{event}. Dabei wird angenommen, dass eine relevante Teilsequenz maximal aus so vielen Ereignissen bestehen kann, wie im erlaubten Zeitrahmen überhaupt auftreten können. Damit liegt der Speicher-

verbrauch das Pattern Matching unter den bis dato beschriebenen Vorbedingungen in einer konstanten Komplexitätsklasse.

Werden n Interaktionsmuster ohne Subausdruck-Operator mit einer maximalen Länge der zu suchenden relevanten Interaktionen κ_τ gleichzeitig untersucht, so liegt der Speicherverbrauch $S_{interact} = s_{event} * n * \kappa_\tau$ in der Komplexitätsklasse $\mathcal{O}(n)$. Die Vervielfachung durch n ergibt sich aus der Tatsache, dass sich im Worst-Case alle Interaktionsmuster verschiedener Anwendungsfälle bzgl. der Ereignistypen gleichen und eine Länge κ_τ aufweisen. Beim Pattern Matching muss ein Ereignis in einem solchen Fall zwar nicht mehrfach für jede separate Untersuchung im Zwischenspeicher vorliegen, aber mindestens mit einem Vermerk versehen sein, der die Zugehörigkeit des Ereignisses zu einer bestimmten Untersuchung kennzeichnet. Die Anzahl der Vermerke für jedes im Zwischenspeicher befindliche Ereignis entspricht dabei der Anzahl der Untersuchungen, in denen das Ereignis ein Teil der zu untersuchenden Teilsequenz ist. Letztendlich ändert die Einführung von Vermerken zur Optimierung des Ressourcenverbrauchs daher nichts an der Komplexitätsklasse. Der Einfachheit halber wird der Speicherverbrauch des Vermerks daher im Nachfolgenden dem Speicherverbrauch eines Ereignisses gleichgesetzt.

Der Subausdruck-Operator führt beim Pattern Matching außerdem noch dazu, dass für ein Interaktionsmuster mehrere Untersuchungen gleichzeitig stattfinden können, da eine Sequenz mehrere sich überschneidende relevante Interaktionen aufweisen kann. Wird ein dem Subausdruck-Operator übergebener Subausdruck gefunden, so wird zum einen damit begonnen, den weiteren Teil des Interaktionsmusters zu überprüfen, und zum anderen die Suche nach dem Subausdruck neu zu starten. Der Einsatz eines Subausdruck-Operators in einem Interaktionsmuster führt demnach im Worst-Case, während der Evaluation aller Ereignisse innerhalb von τ, zu einer κ_τ-Vervielfachung der Untersuchungen. In diesem Fall erfüllt jedes Ereignis in τ die Bedingungen des Subausdrucks und führt demnach zu einer separaten Untersuchung. Mit jedem weiteren Subausdruck innerhalb eines Interaktionsmusters vervielfacht sich die Anzahl an Untersuchungen und folglich der Erkennungen. Die Veränderung der Anzahl gleichzeitiger Untersuchungen aufgrund der Hinzunahme von Subausdruck-Operatoren wird in Abbildung 4.12 beispielhaft aufgezeigt.

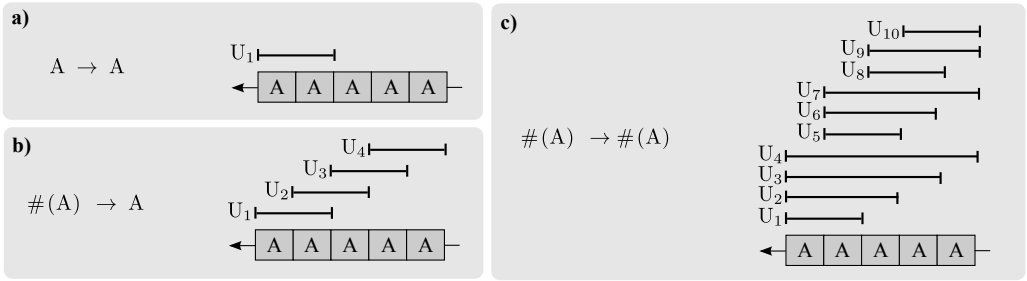


Abb. 4.12 Die Anzahl gleichzeitiger Untersuchungen bei verschiedenen Evaluationen der gleichen Beispielsequenz.

Für eine praktische Anwendung des Pattern Matching im automotiven Bereich ist es daher aufgrund der in eingebetteten Systemen begrenzten Ressourcen ratsam, einen geeigneten Zeitrahmen und damit eine maximale Länge einer zu suchenden relevanten Teilsequenz einzuführen und die Anzahl der Subausdruck-Operatoren innerhalb eines Interaktionsmusters zu begrenzen. Ersterer Bedingung kann natürlicherweise Folge geleistet werden, da sich im automotiven Bereich, wie oben beschrieben, eine sinnvoll zur Personalisierung eignende

Interaktion nicht über Wochen hinziehen kann und auf eine Fahrt beschränkt ist. Bei der zweiten Bedingung bedarf es einer genaueren Untersuchung der maximalen Häufigkeit von Ereignissen, die innerhalb eines Subausdruck-Operators definiert werden und dem Kontext, in dem der Subausdruck-Operator verwendet wird. Während die Subausdrücke der Interaktionsmuster aus den Anwendungsfällen A und C in der Praxis nur maximal einmal pro Sekunde gültig sind und deren neu gestartete Untersuchungen nach einem weiteren Auftreten sofort wieder abgebrochen werden, würde ein Interaktionsmuster

$$(\#(\mathrm{PositionUpdaten})) \rightarrow (\mathrm{FahrtBeenden})$$

in einer Sekunde genauso so viele parallele Untersuchungen während des Pattern Matching starten, wie Aktualisierungen der Positionsdaten vom GPS-Sensor pro Sekunde vorliegen. Da moderne GPS-Sensoren mit einer Frequenz von über 20/s neue Daten bereitstellen können, würden während einer Fahrt von nur 10 Minuten bis zu 12000 gleichzeitige Untersuchungen stattfinden. Damit wären die Ressourcen eingebetteter Systeme in wenigen Minuten aufgebraucht. Zusätzlich zu den oben genannten Bedingungen muss beim praktischen Einsatz im Fahrzeug außerdem noch die gesamte Anzahl zu untersuchender sequentieller Muster limitiert werden. Dies betrifft vor allem die automatisierte Instanziierung von Situationsmustern. Während bei der Erkennung relevanter Interaktionen ohnehin auf eine endliche Menge von Interaktionsmustern zurückgegriffen wird, ist die Menge der Situationsmuster theoretisch unbeschränkt. So lassen sich z.B. für den Anwendungsfall A unendliche viele Regionen erkennen, in denen bestimmte Radiosender bevorzugt werden. Erst mit der Einführung einer maximalen Anzahl zulässiger kontextsensitiver Präferenzen pro Anwendungsfall - wie weiter unten beschrieben durch die Einführung eines künstlichen Verfalls - kann die Anzahl instanziierter Situationsmuster nach oben beschränkt werden.

Die Verarbeitungsdauer des letzten Prozesses $\Delta t_{situation}$ (Erkennung der Situationen kontextsensitiver Präferenzen) ist genauso wie die Verarbeitung des Kontextes als zeitkritisch zu betrachten, da die Ausführung einer Personalisierung möglichst zeitnah nach dem Eintritt einer Situation erfolgen sollte. Da während dieses Prozesses ebenfalls ein Pattern Matching mit Hilfe der gleichen EPL und auf Grundlage der gleichen Sequenz durchgeführt wird, ist die Zeit- und Speicherkomplexität dieser Verarbeitung prinzipiell der des Prozesses zur Ermittlung relevanter Interaktionen gleichzusetzen. Hinzu kommt jedoch die Tatsache, dass die Anzahl und Komplexität der konkretisierten Situationsmuster sich erst zur Laufzeit berechnen. Dabei hängt die Anzahl der konkretisierten Situationsmuster von der Anzahl der kontextsensitiven Präferenzen ab. Diese Zahl lässt sich jedoch nur durch den künstlichen Verfall relevanter Interaktionen nach oben beschränken. Die Implementierung eines künstlichen Verfalls wird somit zu einer notwendigen Bedingung für dieses Verfahren. Dies führt auch dazu, dass die Parameterfilter der konkretisierten Situationsmuster eine bestimmte Komplexität nicht überschreiten können. Die Anzahl der Straßennamen, die es z.B. während der Situationserkennung im Anwendungsfall D für jede kontextsensitive Präferenz zu überprüfen gilt, ist somit ebenfalls durch die Anzahl möglicher relevanter Interaktionen nach oben beschränkt.

Gruppierung relevanter Interaktionen

Bei der Verarbeitungsdauer $\Delta t_{grouping}$ des Prozesses zur Ermittlung der Gruppenvektoren handelt es sich um keinen echtzeitkritischen Wert, da kontextsensitive Präferenzen nicht direkt nach der Ausführung relevanter Interaktionen vorliegen müssen. Dieser Verarbei-

tungsschritt kann z.B. auch während der Zeit ausgeführt werden, in der das Fahrzeug nicht in Betrieb ist. Trotzdem sollte die zeitliche Komplexität dieses Prozesses möglichst gering gehalten werden, damit das Kriterium der Schnelligkeit der Personalisierung (siehe Unterabschnitt 2.3.1) erfüllt werden kann. Eine kontextsensitive Präferenz darf z.B. nicht erst mehrere Wochen nach der Ausführung der zugehörigen relevanten Interaktionen dem Benutzer präsentiert werden. Die geeignete maximale Verarbeitungsdauer sollte also anhand weiterführender Langzeitevaluationen mit potentiellen Benutzern ermittelt werden.

Die Zusammensetzung der Verarbeitungsdauer dieses Prozesses hängt von der gewünschten Form der kontextsensitiven Personalisierung ab. Wird keine Identifizierung der Gruppen durchgeführt, so ergibt sich die Verarbeitungsdauer lediglich aus den zeitlichen Aufwänden des künstlichen Verfalls und des Clusterings. Ist eine Identifizierung notwendig, so kommt außerdem noch der Aufwand für die Berechnung und Gegenüberstellung der Schnittmengen aller Gruppenvektoren hinzu. Der lineare Aufwand des künstlichen Verfalls lässt sich in beiden Fällen vernachlässigen, da dieser auch in den Algorithmus des Clusterings integriert werden kann, in dem ohnehin jedes Ereignis mindestens einmal untersucht werden muss. Die Laufzeitkomplexität von GDBSCAN hängt maßgeblich vom Aufwand der Berechnung der Nachbarschaft ab, da für jedes zu untersuchende Element genau einmal die Nachbarschaft berechnet wird. Liegen alle Elemente bereits in einer geeigneten Form sortiert vor, so ergibt sich die Komplexitätsklasse von GDBSCAN bei einer Menge von n relevanten Interaktionen zu $\mathcal{O}(n * log\, n)$. Andernfalls muss jede relevante Interaktion einzeln mit jeder anderen relevanten Interaktion auf eine mögliche Ähnlichkeit hin untersucht werden. Dies führt erwartungsgemäß zu einer Komplexitätsklasse von $\mathcal{O}(n^2)$. Mögliche Optimierungen werden in [Sander u. a., 1998] vorgestellt. In beiden Fällen wurde angenommen, dass die Berechnungen aller lokalen Ähnlichkeitsfunktionen jeweils mit einem konstanten Aufwand verbunden sind. Hierbei wird auch deutlich, dass bei der Verwendung einer Netzwerk-Distanzfunktion, die über einen quadratischen Aufwand bzgl. der Anzahl an Straßensegmenten verfügt, ein kritischer Gesamtaufwand entstehen würde. Der Aufwand bzgl. der Identifizierung neuer und veränderter Gruppen ergibt sich letztendlich aus der Berechnung aller Schnittmengen zwischen neuen und alten Gruppen, der Identifizierung der größten Schnittmenge jeder neuen Gruppe und der Wahl einer geeigneten Form der Veränderung. Auch hier ist die Anzahl der Gruppen durch die maximale Anzahl relevanter Interaktionen begrenzt.

Kapitel 5
Prototypische Realisierung

Im Rahmen der Untersuchungen entstand eine prototypische Implementierung der Benutzerschnittstelle eines In-Car-Infotainment-Systems, dessen Benutzeroberfläche sich unter Laborbedingungen kontextsensitiv personalisiert. Als Vorlage diente die Benutzerschnittstelle eines In-Car-Infotainment-Systems der Daimler AG[1]. Da diese Benutzerschnittstelle in der Originalversion jedoch über keine Möglichkeit der kontextsensitiven Personalisierung verfügt, wurde der Prototyp außerdem um die Funktionalität der kontextsensitiven Benutzermodellierung ergänzt und die grafische Ausgabe hinsichtlich der beiden im Unterabschnitt 2.3.4 genannten Formen der kontextsensitiven Personalisierung erweitert. Die Implementierung der Benutzerschnittstelle bedient sich der Funktionalität eines prototypisch implementierten Systemdienstes, der das Verfahren der kontextsensitiven Personalisierung aus Kapitel 4 vollständig umsetzt.

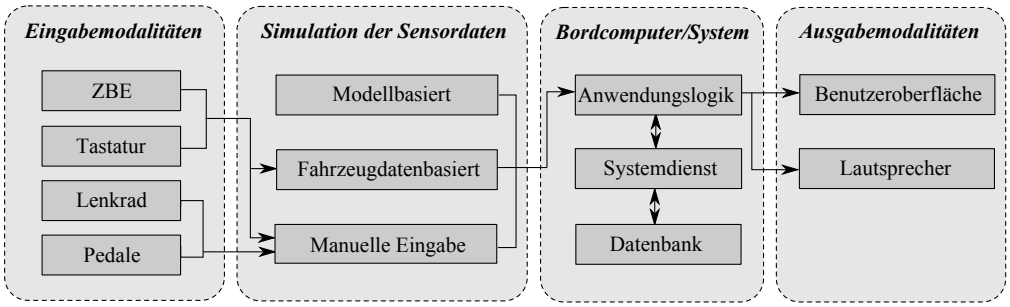


Abb. 5.1 Schematische Übersicht des Prototyps der kontextsensitiv personalisierbaren automotiven Benutzerschnittstelle.

Gesteuert wird der Prototyp, abhängig von der Art der Simulation, mit der zentralen Bedieneinheit, einem Lenkrad, zwei Pedalen und der Computer-Tastatur. Die Simulation der notwendigen Sensordaten kann dabei wahlweise auf drei unterschiedliche Arten erfolgen. Die Sensordaten können in Echtzeit manuell über ein separates Werkzeug dem Prototyp des In-Car-Infotainment-Systems übermittelt, aus den in einem Versuchsfahrzeug aufgezeichneten echten Sensordaten extrahiert oder mit Hilfe einer modellbasierten Szenarienbeschreibung generiert werden. In Abbildung 5.1 werden alle Komponenten in einer schematischen Übersicht des Aufbaus gezeigt.

[1] http://www.daimler.com

Dieses Kapitel widmet sich einer detaillierten Beschreibung der um adaptive Bestandteile erweiterten grafischen Benutzeroberfläche, des Systemdienstes inkl. der notwendigen Vorlagen und der verschiedenen Werkzeuge zur Simulation der interaktions- und kontextbezogenen Sensordaten. Da sich Kapitel 6 ausführlich mit der modellbasierten Generierung der Sensordaten beschäftigt, werden hier ausschließlich die Implementierungen der manuellen Eingabe der Sensordaten und die Simulationen auf Grundlage der aufgezeichneten Sensordaten vorgestellt.

5.1 HMI-Prototyp

Die Implementierung der Anwendungslogik und die grafische Benutzeroberfläche des prototypischen In-Car-Infotainment-Systems bilden zusammen den sogenannten *HMI-Prototyp*. Der HMI-Prototyp wurde im Rahmen dieser Forschungsarbeit implementiert und verfügt neben einigen gängigen Funktionen eines Bordcomputers wie z.B. modifizierbare Spracheinstellungen und flexible Helligkeitsstufen des Bildschirms auch über ein Navigationssystem, einen Browser, eine Radioapplikation und eine simulierte Telefonapplikation. Eine detaillierte Beschreibung der technischen Umsetzung folgt in Unterabschnitt 5.1.3. Da in der Anwendungslogik von den Eingaben einzelner Eingabemodalitäten abstrahiert wird, wurden die speziellen Anbindungen verschiedener Eingabemodalitäten in externe Komponenten ausgelagert und diese über eine gemeinsame Schnittstelle mit dem HMI-Prototyp verbunden. Auch der Systemdienst zur kontextsensitiven Personalisierung und die zugehörige Datenbank sind als externe Komponenten implementiert und gehören daher ebenfalls nicht zum HMI-Prototyp.



Abb. 5.2 Benutzeroberfläche des HMI-Prototyps während der manuellen Eingabe des Fahrziels im Navigationssystem.

Die Benutzeroberfläche des HMI-Prototyps orientiert sich an der Benutzeroberfläche des In-Car-Infotainment-Systems COMAND APS NTG4[1]. Sie setzt sich aus einfachen Grafikelementen, die applikationsübergreifend Verwendung finden, und aus spezifischen Grafikelementen zusammen, die nur jeweils innerhalb der Ansichten einer Applikation zum Einsatz kommen. Der systemspezifische Teil der Benutzeroberfläche besteht aus der Status- und Applikationsleiste. Die am oberen Rand des Bildschirms positionierte Statusleiste zeigt z.B. die Uhrzeit oder die simulierte Empfangsqualität der Mobilfunkverbindung an. Direkt darunter befindet sich die Applikationsleiste, in der die zur Zeit aktive Applikation hervorgehoben wird (siehe Abbildung 5.2) oder eine Applikation ausgewählt und gestartet werden kann. Während die Applikationsleiste auf einem einfachen und parametrisierbaren Grafik-

element einer horizontalen Menüleiste basiert, stellt die Statusleiste ein spezifisches Grafikelement dar, welches ausschließlich in der Benutzerschnittstelle der Systemebene zum Einsatz kommt. Im restlichen Teil des Bildschirms, dem applikationsspezifischen Teil der Benutzeroberfläche, werden die Grafikelemente jeweils einer Applikation angezeigt. In Abbildung 5.2 wird als Beispiel die Ansicht während der manuellen Zieleingabe beim aktivierten Navigationssystem dargestellt. Hier besteht der applikationsspezifische Teil der Benutzeroberfläche aus einer Visualisierung der Zieladresse und aus zwei spezifischen Menüleisten.

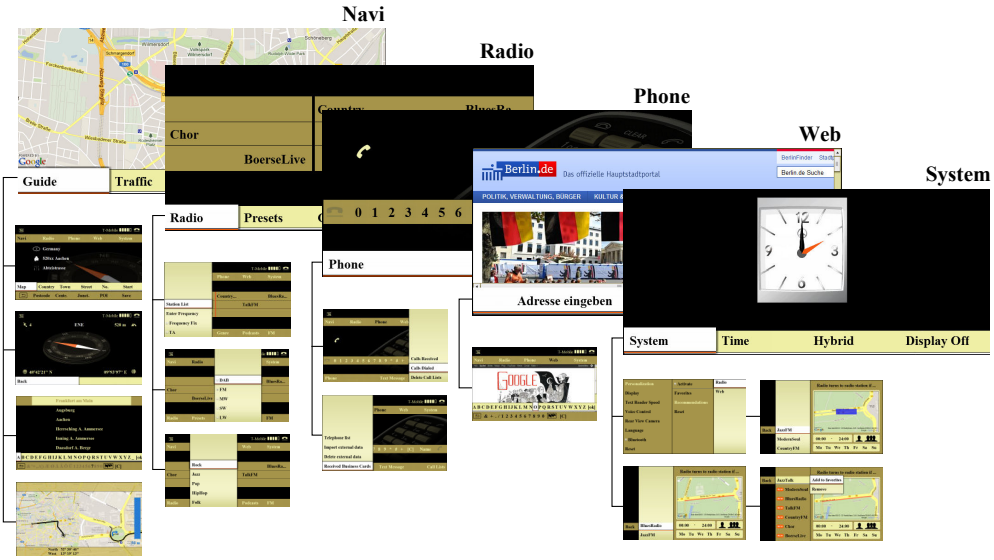


Abb. 5.3 Einige Ansichten und Menüs der Benutzeroberfläche des HMI-Prototyps.

Jede Applikation besteht wiederum aus einer Vielzahl unterschiedlicher Ansichten, die jeweils alle - zur Ausführung einer spezifischen Funktionalität notwendigen - Grafikelemente enthalten. Die Art, die strukturelle Anordnung und das Aussehen der Grafikelemente innerhalb der verschiedenen Ansichten werden dabei durch die Spezifikation des COMAND APS NTG4 vorgegeben. Von der Spezifikation abweichende Ansichten, die für eine kontextsensitive Personalisierung ausgelegt sind, werden im Unterabschnitt 5.1.2.1 und 5.1.2.2 vorgestellt. Auf eine vollständige Beschreibung aller Ansichten und Funktionalitäten des HMI-Prototyps wird hier jedoch verzichtet. Der interessierte Leser sei hierzu auf die Spezifikation des COMMAND APS NTG4 verwiesen. Alle fünf Hauptansichten und einige spezifische Ansichten und Menüs der Applikationen werden in Abbildung 5.3 dargestellt. Alle systemweiten Einstellmöglichkeiten werden hier unter der Applikation „System“ zusammengefasst. In einigen Ansichten der Applikationen werden die Status- und Applikationsleiste aus Platzgründen ausgeblendet. Die Benutzeroberfläche des Navigationssystems befindet sich beispielsweise immer im Vollbildmodus, wenn die Zielposition mit Hilfe eines Fadenkreuzes auf einer Landkarte angegeben wird.

5.1.1 Bedienung

Die Bedienung des HMI-Prototyps erfolgt wahlweise über einen Dreh-Drück-(Schiebe)-Steller und zusätzlichen Tasten (zusammen die ZBE) oder über eine Computertastatur. Sowohl die ZBE als auch die Tastatur können in der Eingabe des HMI-Prototyps zunächst 20 Eingabeereignisse unterschiedlichen Typs hervorrufen. Jede der in Abbildung 5.4 durch einen Doppelpfeil gekennzeichneten Drück-, Dreh- und Schiebebewegungen des Dreh-Drück-(Schiebe)-Stellers wird auf jeweils zwei unterschiedliche Ereignistypen abgebildet, die zum einen den Vorgang der Betätigung und zum anderen den Vorgang des Loslassens beschreiben. Die beiden Drehrichtungen werden jeweils nur auf einen Ereignistyp abgebildet. Der Benutzer ist mit Hilfe dieser 20 Freiheitsgrade in der Lage die Position eines Cursors auf der Benutzeroberfläche zu steuern. Der Cursor kann sich dabei entweder in der Applikationsleiste oder auf einem Grafikelement innerhalb einer applikationsspezifischen Ansicht befinden. Die Auswahl bzw. Ausführung einer Funktionalität, die durch den Cursor markiert wird, findet dann jeweils durch das Drücken und Loslassen des Dreh-Drück-(Schiebe)-Stellers oder durch die Betätigung der Enter-Taste auf der Tastatur statt.

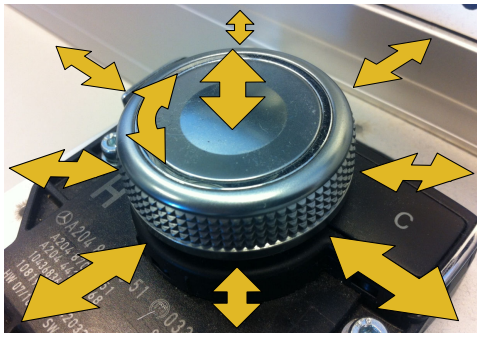


Abb. 5.4 Alle verwendbaren Freiheitsgrade des Dreh-Drück-(Schiebe)-Stellers der ZBE.

Das Verhalten des Cursors innerhalb der Zieleingabe im Navigationssystem wird in Abbildung 5.5 beispielhaft durch ein UML-Zustandsdiagramm visualisiert. Jede Position des Cursors wird durch einen individuellen Zustand modelliert. Wegen der Übersichtlichkeit wurde die Modellierung des Cursorverhaltens in der unteren spezifischen Menüleiste ausgeblendet. Befindet sich der Cursor beispielsweise auf dem Subelement „Navi“ und möchte der Benutzer die Zielführung starten, so braucht man 6 Eingabeereignisse (1xTyp „zurück“, 4x „rechts“, 1x „bestätigen“), um den Cursor richtig zu positionieren und die Funktionalität über das Drücken des Dreh-Drück-(Schiebe)-Stellers auszuführen. Dies führt insgesamt zu 12 Benutzeraktionen an der ZBE, da nach jeder Betätigung des Dreh-Drück-(Schiebe)-Stellers dieser durch Loslassen wieder in seine ursprüngliche Position gebracht werden muss. Die Anzahl der Bedienschritte, die zur Ausführung einer bestimmten Funktionalität des In-Car-Infotainment-Systems notwendig sind, hängt - unter der Voraussetzung, dass das In-Car-Infotainment-System ausschließlich über den Dreh-Drück-(Schiebe)-Steller bedient wird - von der grundlegenden Struktur der Benutzeroberfläche und der aktuellen Position des Cursors ab. In den nachfolgenden Betrachtungen, in denen die Bedienschritte zusammenfassend für die Anwendungsfälle A bis D identifiziert werden, wird immer beispielhaft davon ausgegangen, dass sich der Cursor vor der Bedienung in der Applikationsleiste auf dem Subelement „Navi“ befindet.

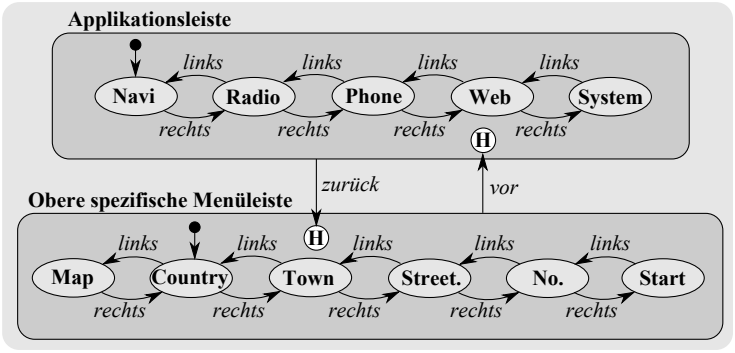


Abb. 5.5 UML-Zustandsdiagramm des Verhaltens bzw. Position des Cursors bzgl. der Applikations- und der oberen spezifischen Menüleiste bei der manuellen Zieleingabe im Navigationssystem.

Soll im Anwendungsfall A der präferierte Radiosender manuell eingestellt werden, so sind mindestens drei Benutzeraktionen notwendig. Der Cursor wird durch eine Drehbewegung des Dreh-Drück-(Schiebe)-Stellers zunächst auf das Subelement „Radio“ verschoben. Die Ausführung der Radioapplikation findet dann durch das Drücken und Loslassen des Dreh-Drück-(Schiebe)-Stellers statt. Daraufhin befindet sich der Cursor automatisch innerhalb der Radioansicht auf der Nadel des Frequenzschiebers (siehe Abbildung 5.6). Wird im günstigsten Fall davon ausgegangen, dass sich die Frequenz des präferierten Radiosenders direkt neben der aktuell eingestellten Frequenz befindet, so ist zur Ausführung des Radiosenderwechsels abschließend nur eine Drehbewegung des Dreh-Drück-(Schiebe)-Stellers in eine der beiden Richtungen erforderlich.

Im Anwendungsfall B entsteht zunächst kein Bedienaufwand, da der Benutzer die zu personalisierende Funktionalität nicht ausführt. Hier wird in erster Linie also nicht versucht, den Bedienaufwand zu reduzieren, sondern das explorative Verhalten des Benutzers zu fördern. Stimmt das vom System antizipierte Fahrziel jedoch mit dem gewünschten Fahrziel überein, und ist der Benutzer an den Verkehrsmeldungen interessiert, so kann er durch die Abkürzung über den kontextsensitiv personalisierten Schnellstartbutton den Bedienaufwand von mindestens 23 Benutzeraktionen auf zwei reduzieren. Die hohe Anzahl an Bedienschritten ergibt sich aus der Tatsache, dass für die Anzeige der Verkehrsmeldungen zunächst das Fahrziel eingegeben werden muss.

Der Aufruf der Webseite im Anwendungsfall C erweist sich - unter der Bedingung, dass die URL nicht in den Favoriten gespeichert ist - als problematischer, da dabei die URL der Webseite manuell über die Bildschirmtastatur eingegeben werden muss. Insgesamt ergeben sich somit - unter der Voraussetzung, dass die URL „bahn.de“ aufgerufen wird - 74 Benutzeraktionen bei der Steuerung über die ZBE. Sieben der Benutzeraktionen entfallen auf die Navigation durch die Menüstruktur, während sich die Benutzeraktionen bei der Eingabe der URL auf eine Anzahl von 67 summieren. Selbst wenn die URL bereits in einer Favoritenliste enthalten ist, sind mindestens 10 Benutzeraktionen auszuführen, um den gewünschten Eintrag in einer solchen Liste auszuwählen und anzuzeigen.

Für einen manuell getätigten Anruf an eine in der Liste gespeicherte Telefonnummer (Anwendungsfall D) sind ebenfalls mindestens 10 Benutzeraktionen auszuführen. Diese Zahl ergibt sich aufgrund der Navigation durch die Menüstruktur und der Anwahl einer Telefonnummer aus der Telefonnummernliste. Auch hier wird vom günstigsten Fall ausgegangen, bei dem sich der Cursor nach dem Aufruf der Telefonnummernliste direkt auf

dem präferierten Eintrag befindet. Somit entfällt das aufwendige Scrollen durch die Telefonnummernliste. Im nachfolgendem Unterabschnitt wird beschrieben, wie die genannten Bedienaufwände durch kontextsensitive Personalisierungen reduziert werden.

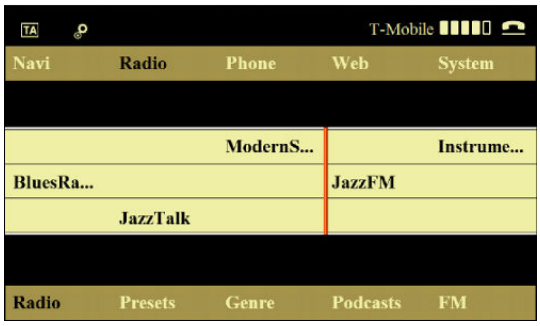


Abb. 5.6 Ansicht der Radioapplikation kurz nach der durch einen Radiosenderwechsel hervorgerufenen Erkennung einer relevanten Interaktion (signalisiert durch das Zahnräder-Icon in der Statusleiste).

5.1.2 Erweiterung der Benutzerschnittstelle um kontextsensitive Adaptivität

Während der Bedienung des HMI-Prototyps und der parallel laufenden impliziten Datenerfassung durch den Systemdienst werden bestimmte, manuell ausgeführte Bedienschritte bzw. Interaktionen, die sich entweder für eine kontextsensitive Personalisierung eignen oder eine kontextsensitive Personalisierung zur Folge haben, gesondert verarbeitet. Die manuelle Ausführung einer der oben beschriebenen Einstellungen bzgl. der Anwendungsfälle A bis D führt innerhalb des Systemdienstes - nach dem Verfahren aus Kapitel 4 - zu der Erkennung einer relevanten Interaktion. Um eine hohe Nachvollziehbarkeit der Benutzerschnittstelle zu gewährleisten, wird unmittelbar nach der Ermittlung einer solchen relevanten Interaktion - für die Dauer von einer Sekunde - ein Zahnräder-Icon in der Statusleiste eingeblendet (siehe Abbildung 5.6). Das System setzt den Benutzer dabei über die Tatsache in Kenntnis, dass aufgrund der unmittelbar zuvor ausgeführten Interaktion in Zukunft möglicherweise eine kontextsensitive Personalisierung dieser Interaktion durchgeführt wird. Diese Erweiterung kann unabhängig vom Anwendungsfall oder der Form der kontextsensitiven Personalisierung in der Benutzeroberfläche umgesetzt werden. Voraussetzung für eine solche Funktionalität ist jedoch die Limitierung der Verarbeitungsdauern $\Delta t_{context}$ und $\Delta t_{relevant}$ (siehe Unterabschnitt 4.2.5). Die Summe beider Verarbeitungsdauern darf dabei einen kritischen Wert nicht überschreiten. Überschreitet die Summe dieser Verarbeitungsdauern dennoch einen idealerweise durch Studien zu ermittelnden Grenzwert, so kann das Zähnräder-Icon möglicherweise zu spät angezeigt werden. Der Benutzer kann währenddessen bereits andere Interaktionen ausgeführt haben, die dann fälschlicherweise mit der kontextsensitiven Personalisierung assoziiert werden. Der Benutzer sollte also zeitnah über die Relevanz seiner Bedienung informiert werden.

Im Nachfolgenden werden die prototypischen Implementierungen der beiden im Kapitel 2 beschriebenen kontextsensitiven Personalisierungen einer automotiven Benutzerschnittstelle zusammen mit der Möglichkeit des benutzerspezifischen Feedbacks vorgestellt.

5.1.2.1 Kontextsensitiv personalisierte Schnellstartbuttons

Für die im Unterabschnitt 2.3.4 vorgestellten kontextsensitiv personalisierbaren Schnellstartbuttons wird im HMI-Prototyp eine eigene Benutzeroberfläche eingeführt. Im realen Fahrzeug sollte diese idealerweise auf einem separaten Bildschirm, der möglichst in der Mittelkonsole oberhalb des Hauptbildschirms platziert ist, angezeigt werden. Diese Benutzeroberfläche besteht fortlaufend aus vier grafischen Elementen. Jedes Element steht dabei entweder stellvertretend für eine bestimmte, vom System in der aktuellen Situation empfohlene Konfiguration oder für einen Platzhalter ohne Konfiguration (durch ein Fragezeichen gekennzeichnet). Eine beispielhafte Belegung der kontextsensitiv personalisierten Schnellstartbuttons wird in Abbildung 5.7 dargestellt. In diesem Beispiel werden lediglich drei Konfigurationen angezeigt, da während der kontextsensitiven Benutzermodellierung nur drei kontextsensitive Präferenzen mit der aktuellen Situation assoziiert wurden.



Abb. 5.7 Beispielhafte Belegung der kontextsensitiv personalisierten Schnellstartbuttons.

Ein Element, welches mit einer Konfiguration belegt ist, besteht aus einem Icon und einem Text unterhalb des Icons. Durch das Icon wird die betreffende Applikation und/oder Funktionalität gekennzeichnet, während der Text auf die benutzerspezifische Parametrisierung der Konfiguration hinweist. Der Sendemast zusammen mit dem Untertitel „JazzTalk“ visualisiert beispielsweise die Konfiguration der Radioapplikation mit dem Radiosender „JazzTalk“. Analog dazu kennzeichnet das Icon mit der Kompassnadel die Konfiguration der Zielführung zur „B2 Ebertstrasse“. Das dritte Element mit dem Stausymbol und dem Ziel „B2 Ebertstrasse“ ist dagegen ein Beispiel der in Anwendungsfall B vorgestellten kontextsensitiven Personalisierung. Hier wird die Anzeige der Verkehrsmeldungen, die sich auf die kürzeste Strecke zum vom System antizipierten Ziel beziehen, in der aktuellen Situation als Konfiguration empfohlen.

In jeder Situation werden maximal vier Konfigurationen angezeigt, auch wenn im Benutzermodell mehr als vier kontextsensitive Präferenzen mit der aktuellen Situation assoziiert werden. Somit wird während der Fahrt sichergestellt, dass der Benutzer bei der Begutachtung der sich dynamisch ändernden Schnellstartbuttons keine Überforderung erfährt. Verfügt das Benutzermodell in der aktuellen Situation dennoch über eine weitere Empfehlung, so wird diese zunächst zurückgehalten. Ändert sich die Situation dahingehend, dass eine der bisher angezeigten Konfigurationen die Gültigkeit verliert, dann wird diese durch die zurückgehaltene Empfehlung ersetzt.

Bei den kontextsensitiv personalisierten Schnellstartbuttons handelt es sich zunächst jedoch nur um eine Visualisierung der kontextsensitiven Präferenzen bzw. antizipierten Konfigurationen. Es bedarf letztendlich der Initiative des Benutzers, um eine der empfohlenen

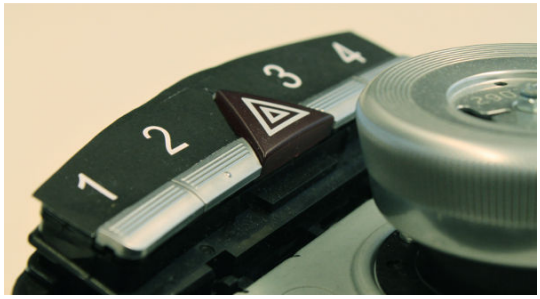


Abb. 5.8 Zusätzliche und nummerierte Multifunktionstasten an der zentralen Bedieneinheit.

und angezeigten Konfigurationen durchzuführen. Das zur gewünschten Konfiguration passende grafische Element wird dazu nicht über den Cursor angewählt und durch Betätigung des Dreh-Drück-(Schiebe)-Stellers ausgeführt, sondern über eine von vier separaten Tasten an der ZBE - direkt durch einen einfachen Knopfdruck - ausgelöst. Jede der vier zusätzlichen Tasten an der ZBE hat also eine von der Situation abhängige Funktionalität, die durch den dazu passenden Schnellstartbutton visualisiert wird. Die Nutzung der mit variabler Funktionalität versehenen physikalischen Tasten reduziert den Bedienaufwand um ein Vielfaches, da die mühevolle Navigation durch die Menüstruktur entfällt. Der Dreh-Drück-(Schiebe)-Steller und die nummerierten, multifunktionalen Tasten der prototypischen Benutzerschnittstelle werden in Abbildung 5.8 dargestellt.

5.1.2.2 Kontextsensitiv personalisierte Automatisierung

Die kontextsensitiv personalisierte Automatisierung für die Anwendungsfälle A und C findet im HMI-Prototyp nur dann statt, wenn entsprechende kontextsensitive Präferenzen vorliegen und diese durch den Benutzer validiert wurden. Automatisch erkannte Präferenzen werden also nicht unmittelbar in Personalisierungen umgesetzt, sondern müssen vom Benutzer zunächst begutachtet und als praktisch sinnvoll eingestuft werden. Die Konfiguration und die dazugehörige Situation einer kontextsensitiven Präferenz müssen also gemeinsam in einer Feedbackansicht visualisiert werden. Da es sich bei den kontextsensitiven Präferenzen um persönliche Informationen handelt, die nicht an Applikationen von Drittherstellern weitergegeben werden sollten, und da kontextsensitive Automatisierungen applikationsübergreifend stattfinden können, wurde die Applikation des Systems um die Feedbackansicht erweitert.

Zur Feedbackansicht gelangt der Benutzer über das applikationsspezifische Menü des Systems (siehe Abbildung 5.9 links). Befindet sich der Cursor auf dem Subelement „System“ und bestätigt der Benutzer die Auswahl, so öffnet sich ein vertikales Untermenü mit einem Eintrag „Personalization“ (siehe Abbildung 5.9 rechts). Über die Auswahl und Bestätigung des Eintrags „Personalization“ gelangt der Benutzer dann in das vertikale Untermenü mit den Einträgen „Activate“, „Favorites“, „Recommendations“ und „Reset“. Der Benutzer ist mit Hilfe der Option „Activate“ in der Lage, die kontextsensitiv personalisierte Automatisierung global an- oder abzuschalten. Eine Deaktivierung dieser Option führt folglich dazu, dass es in der Benutzeroberfläche niemals zu Automatisierungen kommt. Die Anwahl und Bestätigung des Eintrags „Reset“ führt zu einer Löschung aller bisher durch die Benutzermodellierung erkannten kontextsensitiven Präferenzen. Im Fall der Abbildung

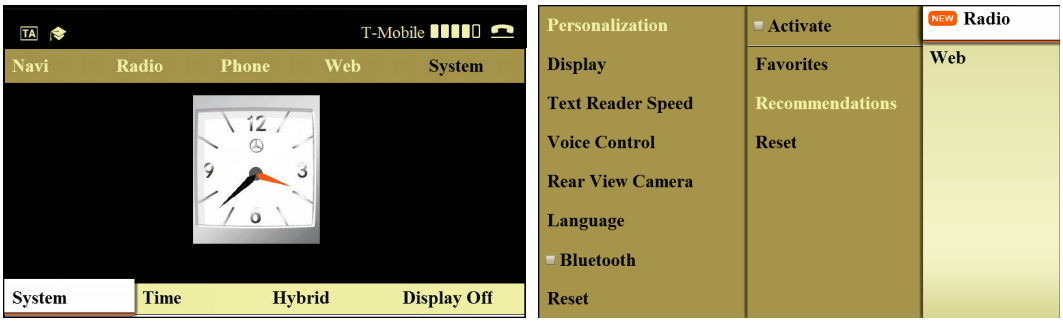


Abb. 5.9 Links: Hauptansicht der Applikation des Systems; Rechts: Untermenü vom System, Untermenü der Personalisierung und Untermenü der Empfehlungen des Systems.

5.9 weist das Untermenü des Eintrags „Recommendations“ zwei Einträge auf: „Radio“ und „Web“ . Dies bedeutet, dass das System jeweils mindestens eine kontextsensitive Präferenz bzgl. der Radioeinstellungen und bzgl. der Browsereinstellungen ermittelt hat. Das NEW-Icon beschreibt dabei die Tatsache, dass dem System mindestens eine neue und bisher nicht begutachtete kontextsensitive Präferenz bzgl. der Radioeinstellungen vorliegt. Dies wird zusätzlich durch ein Doktorhut-Icon links in der Statusleiste signalisiert, das immer dann angezeigt wird, wenn mindestens eine neue kontextsensitive Präferenz, die sich zur Automatisierung eignet, vorliegt (siehe Abbildung 5.9 links). Dadurch wird der Benutzer von der Aufgabe entbunden, in regelmäßigen Abständen die Feedbackansicht aufzusuchen, um diese auf das Vorhandensein von neuen Empfehlungen hin zu überprüfen. Um eine kontextsensitive Präferenz zu begutachten, wählt der Benutzer den entsprechenden Eintrag „Radio“ aus und gelangt somit in die Feedbackansicht für die Radioeinstellungen (siehe Abbildung 5.10).

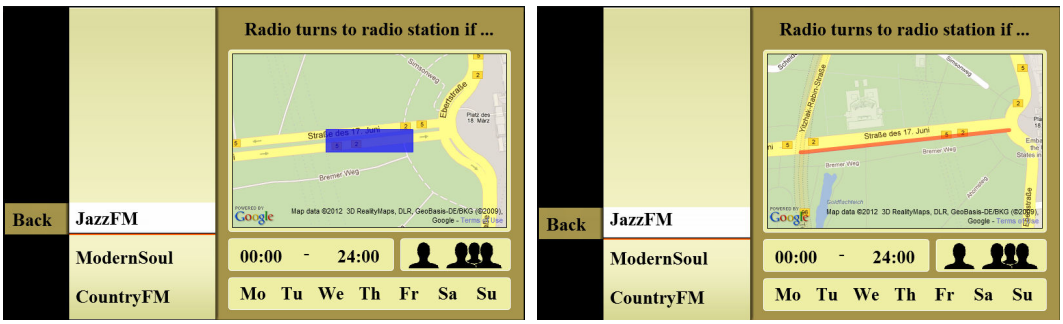


Abb. 5.10 Links: Feedbackansicht der Radioeinstellungen mit regionenabhängigen Präferenzen; Rechts: Feedbackansicht der Radioeinstellungen mit straßensegment-abhängigen Präferenzen.

Die Feedbackansichten in Abbildung 5.10 links und rechts bestehen auf der linken Seite aus einer Auflistung aller durch die Radiosendernamen identifizierten kontextsensitiven Präferenzen. Der Benutzer ist nun in der Lage, jeden Eintrag einzeln anzuwählen und die zugehörige Situation im rechten Teil des Bildschirms einzusehen. Visualisiert wird die Situation durch diverse Grafikelemente, die jeweils den Ort, die Tageszeit, den Wochentag und die Anzahl der Fahrzeuginsassen einer kontextsensitiven Präferenz anzeigen.

Wie im Unterabschnitt 2.3.4 vorgestellt, werden in den Anwendungsfällen A und C nur ortsabhängige Routinen untersucht. Abhängig von den in den Vorlagen gewählten Distanzfunktionen beschreiben die Situationen entweder rechteckig geformte Regionen oder bestimmte Straßensegmente. Beide Möglichkeiten werden jeweils durch ein blaues Rechteck (Region) oder eine rote Linie (Straßenabschnitt) in der Abbildung 5.10 dargestellt. In beiden Fällen kann die personalisierte Automatisierung zu jeder Tageszeit 00:00 Uhr bis 24:00 Uhr, in An- oder Abwesenheit weiterer Fahrzeuginsassen und an jedem Wochentag eintreten.

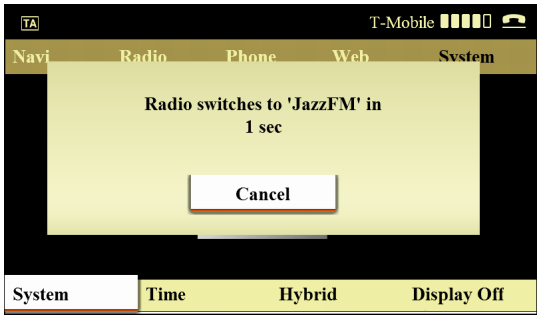


Abb. 5.11 Signalisierung einer kurz bevorstehenden kontextsensitiven Automatisierung.

Entspricht eine empfohlene kontextsensitive Präferenz den Wünschen des Benutzers, so kann er/sie die Präferenz über ein Untermenü zu seinen Favoriten hinzufügen. Über den Eintrag „Favorites “ im Untermenü „Personalization“ (siehe Abbildung 5.9 rechts) ist der Benutzer dann in der Lage, alle favorisierten kontextsensitiven Präferenzen, ähnlich wie in der Feedbackansicht, einzeln zu begutachten und außerdem zu aktivieren. Ist eine solche kontextsensitive Präferenz aktiviert, so wird die entsprechende Konfiguration immer dann automatisch umgesetzt, wenn die entsprechende Situation vorliegt. Der Benutzer wird in einer solchen Situation durch eine Dialogbox auf die anstehende kontextsensitiv personalisierte Automatisierung aufmerksam gemacht (siehe Abbildung 5.11). Die Signalisierung ließe sich alternativ auch durch das Abspielen eines Tons oder einer personalisierten Sprachansage implementieren. Die Dialogbox enthält Informationen zur Automatisierung, einen Timer und eine Abbruchtaste. Es wird dem Benutzer also ermöglicht, die anstehende Automatisierung innerhalb der Restlaufzeit des Timers durch Betätigung des Dreh-Drück-(Schiebe)-Stellers abzubrechen. In Ausnahmefällen kann somit eine Automatisierung verhindert werden, ohne zuvor die kontextsensitive Präferenz in den Favoriten manuell deaktivieren zu müssen. Bei einer kontextsensitiv personalisierten Automatisierung entsteht also bis auf die optionale Kontrolle der Inhalte der Dialogbox kein weiterer Bedienaufwand.

5.1.3 Technische Umsetzung

Der HMI-Prototyp wurde als Adobe® Flash®-Applikation[2] umgesetzt. Die Implementierung basiert auf dem Open-Source Framework Flex®[3], welches speziell für die Entwicklung von Flash®-Applikationen für Betriebssysteme konzipiert wurde. Die Anwendungs-

[2] http://www.adobe.com/
[3] http://www.adobe.com/de/products/flex.html

logik des HMI-Prototyps wurde folglich mit der Programmiersprache ActionScript® entwickelt, während die Auszeichnungssprache Adobe® MXML zur Gestaltung der grafischen Benutzerschnittstelle diente. Die Laufzeitumgebung Adobe® AIR® ermöglicht die Ausführung des HMI-Prototyps als Desktop-Anwendung innerhalb jedes von Adobe® AIR® unterstützen Betriebssystems, wie z.B. Microsoft® Windows® oder Mac OS®.

Das integrierte Navigationssystem verfügt über bekannte Funktionalitäten wie eine Kartendarstellung, eine Zieleingabe, die Berechnung einer geeigneten Route, eine Turn-by-Turn-Navigation oder die Anzeige aktueller Verkehrsmeldungen. Die sowohl bei der Turn-by-Turn-Navigation als auch bei der Eingabe des Fahrziels (über eine Karte) verwendeten Geodaten werden durch die Google Maps™ API[4] in Form von Bildkacheln zur Verfügung gestellt. Die in der manuellen Zieleingabe angezeigten Städte, Postleitzahlen, Straßennamen, Hausnummern und Point-of-Interests werden hingegen durch den Street Director der Firma Elektrobit Group[5] bereitgestellt. Die Berechnung der Route auf der Basis der aktuellen Position als GPS-Koordinate und der durch die Geocodierung der Google Maps™ API berechneten Zielkoordinate findet über die Microsoft® Bing® Maps API[6] statt. Sowohl die Funktionalität der Geocodierung, als auch die Berechnung der Route werden von den jeweiligen APIs durch einen RESTful Web Service implementiert. Die Bereitstellung der TMC-Verkehrsmeldungen Deutschlands über einen RSS-Feed als XML-Datei erfolgt auf einer Webseite der CloudMade Deutschland GmbH[7]. Die XML-Datei wird vom HMI-Prototyp entsprechend eingelesen und in geeigneter Form im Menü des Navigationssystems visualisiert.

Die prototypische Radioapplikation greift zur Simulation der Radiosender entweder auf vorkonfigurierte Internetradio-Streams zu oder spielt im Offline-Modus integrierte Aufzeichnungen bestimmter Radiosender ab. Die Web-Applikation nutzt den im Open-Source-Framework Flex® zur Verfügung gestellten Browser, um HTML Webseiten darzustellen. Im Gegensatz zu den zuvor beschriebenen Applikationen handelt es sich bei der Telefonapplikation nur um eine Simulation der notwendigen Benutzeroberfläche. Die Anbindung eines realen Mobiltelefons an den HMI-Prototyp ist nicht möglich.

Das Open-Source-Framework Flex® bietet außerdem die Möglichkeit, Tastenbefehle der Computertastatur direkt an den HMI-Prototypen weiterzuleiten. Somit entfällt die Entwicklung eines Treibers für diese Eingabemodalität. Da die ZBE nur über einen CAN-Bus an einen handelsüblichen Computer angebunden werden kann, kommt hier ein CAN-USB-Adapter der Firma PEAK-System Technik GmbH[8] zum Einsatz. Damit lassen sich die Nachrichten auf dem CAN-Bus über den USB-Port des Computers auslesen. Ein im Rahmen dieser Forschungsarbeit entwickelter Treiber nimmt die Befehle der ZBE über den USB-Port entgegen und stellt sie dem HMI-Prototypen in geeigneter Form über eine TCP/IP-Verbindung zur Verfügung. Bei der im Rahmen dieser Forschungsarbeit verwendeten ZBE handelt es sich um ein Modell der Mercedes-Benz S-Klasse der Baureihe 221.

[4] https://developers.google.com/maps/
[5] http://www.elektrobit.com/
[6] http://www.microsoft.com/maps/developers/web.aspx
[7] http://www.freiefahrt.info/
[8] http://www.peak-system.com

5.2 Systemdienst der kontextsensitiven Personalisierung

Die Datenverarbeitung zum Zweck der kontextsensitiven Personalisierung findet, wie im Kapitel 4 beschrieben, innerhalb des Systemdienstes statt. Dieser liegt in der prototypischen Implementierung in Form einer eigenständigen Java-Applikation vor. Um die im letzten Abschnitt vorgestellten kontextsensitiven Personalisierungen des HMI-Prototyps zu ermöglichen, ist es erforderlich, den Systemdienst mit den zur kontextsensitiven Personalisierung notwendigen Daten zu versorgen und auf die möglichen Empfehlungen des Systemdienstes bzgl. kontextsensitiver Funktionalität in entsprechender Weise zu reagieren. Im Nachfolgenden wird zunächst die Schnittstelle des Systemdienstes vorgestellt. Anschließend widmet sich dieser Abschnitt einer ausführlichen Beschreibung der für die Parametrisierung des Systemdienstes notwendigen Vorlagen.

5.2.1 Schnittstelle

Die Kommunikation zwischen dem HMI-Prototyp und dem Systemdienst basiert konzeptionell auf dem Client-Server-Modell. Der Systemdienst agiert in der Rolle des Servers während der HMI-Prototyp sich als Client beim Systemdienst registriert. Somit können sich beliebig viele Komponenten bzw. Applikationen beim Systemdienst für die kontextsensitive Personalisierung anmelden. Jede Nachricht - zwischen dem HMI-Prototyp und dem Systemdienst - besteht aus einem durch ein Schema spezifiziertes XML-Dokument. Dabei wird grundlegend zwischen zwei Formen von Nachrichten unterschieden: *Situationsnachrichten* und *Dienstnachrichten*. Die Situationsnachrichten entsprechen den in Kapitel 4 vorgestellten Eingabeereignissen. Der Quelltext 5.1 zeigt den Ausschnitt einer Situationsnachricht, die vom HMI-Prototyp aufgrund einer Änderung der Fahrzeugposition generiert und an den Systemdienst übermittelt wurde. Die möglichen Ereignistypen und deren Eigenschaften werden gemeinsam in einer separaten Datei, die im Nachfolgenden als Ereignistypbeschreibung bezeichnet wird, spezifiziert. Eine beispielhafte Ereignistypbeschreibung kann im Anhang A.1 eingesehen werden. Neue Ereignistypen, die aufgrund der Installation einer neuen kontextsensitiv personalisierbaren Applikation auftreten können, müssen dem Systemdienst über diese Datei bekannt gemacht werden.

```
<LoggedEvent user="Mueller" type="Location">
  <Property name="latitude" value="52.512877"/>
  <Property name="longitude" value="13.320236"/>
  <Property name="direction" value="172"/>
</LoggedEvent>
```

Quelltext 5.1 Beispiel einer Situationsnachricht bzgl. der Position des Fahrzeugs

Anfragen, Aktualisierungen der kontextsensitiven Präferenzen und die vom Systemdienst ausgehende Signalisierung bestimmter Situationen werden über die Dienstnachrichten abgewickelt. Die Dienstnachrichten können also im Gegensatz zu den Situationsnachrichten in beide Richtungen verschickt werden. Die Kommunikation über die Dienstnachrichten ist in diesem Fall speziell auf die Anforderungen des HMI-Prototyps ausgelegt worden. Die Struktur der Benutzeroberfläche des HMI-Prototyps gibt hierbei vor, in welchem Zustand welche Art von Anfragen an den Systemdienst gestellt werden und welche Antworten vom

Systemdienst erwartet werden. Befindet sich die Benutzeroberfläche beispielsweise in der Feedbackansicht, so wird nach dem Scrollvorgang in der Liste eine neue Anfrage bzgl. der neu markierten Präferenz an den Systemdienst gestellt, um die Ansicht entsprechend aktualisieren zu können. Der Systemdienst antwortet dann mit einer vollständigen Beschreibung der angefragten Präferenz. Der Quelltext 5.2 zeigt eine solche Anfrage und die dazu passende Antwort des Systemdienstes. Die Antwort enthält sowohl die zu tätigende Einstellung in der Radioapplikation als auch die Situation (in Form zweier eindeutiger Identifikatoren der betreffenden Straßensegmente). Die allgemeine Struktur aller Situations- und Dienstnachrichten des Systemdienstes werden im Anhang A.2 durch ein entsprechendes XML-Schema spezifiziert. Ein UML-Sequenzdiagramm ergänzt das XML-Schema im Anhang A.3 um eine Beschreibung der einzelnen Kommunikationsabläufe.

```
<UseCaseQuery id="27" parameter="true" situation="true"/>
-----------------------------------------------------------------
<UseCaseResult>
  <UseCase id="27" group="Radio">
    <Properties>
      <Property name="nameOfRadioStation" type="String">JazzFM</
         Property>
    </Properties>
    <Situation>
      <Factor name="location">
        <Fragment ...>
          <FragmentNominal>7892345</FragmentNominal>
        </Fragment>
        <Fragment ...>
          <FragmentNominal>2344525</FragmentNominal>
        </Fragment>
      </Factor>
    </Situation>
  </UseCase>
</UseCaseResult>
```

Quelltext 5.2 Oben: Anfrage bzgl. einer speziellen kontextsensitiven Präferenz mit der Identifikationsnummer 27, Unten: Passende Antwort mit den Einstellungen und der Situation der kontextsensitiven Präferenz

Das Datenmodell wird laut der Anhänge A.2 und A.3 strikt vom Systemdienst verwaltet. Im Datenmodell werden nicht nur die Konfigurationen samt Situationen gespeichert, sondern auch die aktuelle Relevanz einer kontextsensitiven Präferenz vermerkt. Aktiviert der Benutzer - zwecks einer gewünschten kontextsensitiven Automatisierung - eine kontextsensitive Präferenz, so ist dieser Schritt dem Systemdienst über eine Dienstnachricht zu melden. Der Systemdienst ist somit in der Lage, den Prozess der Erkennung relevanter Situationen bedarfsgerecht zu steuern und nur dann Personalisierungen zu signalisieren, wenn diese zuvor explizit durch den Benutzer bestätigt wurden. Damit wird der Ressourcenverbrauch während der Ermittlung relevanter Situationen gesenkt, da die Suche nach unerwünschten Situationsmustern deaktiviert werden kann.

5.2.2 Konfiguration über Vorlagen

Die Konfiguration der Datenverarbeitung im Systemdienst findet über die Vorlagen statt, die in einer gemeinsamen XML-Datei verwaltet werden. Diese Datei enthält Informationen zur Erstellung des Kontextvektors, sequentielle Interaktionsmuster zur Ermittlung relevanter Interaktionen/Situationen und die Konfigurationen bzgl. der Gruppierung relevanter Interaktionen. Im prototypischen Systemdienst wird die XML-Datei während des Startvorgangs geladen, ausgewertet und die spezifischen Bestandteile an die entsprechenden Komponenten des Systemdienstes weitergereicht. Eine Aktualisierung der Vorlagendatei während der Laufzeit wäre zwar möglich, wurde im vorliegenden Prototyp jedoch nicht implementiert. Das vollständige XML-Schema der Vorlagendatei ist dem Anhang A.4 zu entnehmen. Die drei wichtigsten Bestandteile der Vorlagendatei, nämlich die Konfiguration des Kontextes, die Konfiguration der Benutzermodellierung und die Konfiguration der Situationen, werden in den nachfolgenden Unterabschnitten vorgestellt.

5.2.2.1 Konfiguration des Kontextes

Der im Systemdienst als relevant zu betrachtende Kontext wird in der sogenannten *Kontextsignatur* festgelegt. Diese wird global definiert und gilt daher für jeden in den Vorlagen spezifizierten Anwendungsfall. Die Kontextsignatur beschreibt implizit die Menge F_{rel} der für den Kontextvektor relevanten Ereignistypen und die Ausprägung der Abbildung $f_{edit_context}$ für jeden Ereignistyp der Menge F_{rel} (siehe Unterabschnitt 4.2.1).

```
<ContextSignature>
  <Attribute name="time" type="Time" >
    <Input event="Event">
      <EventAssignment source="ts" target="timestamp"/>
    </Input>
  </Attribute>
  <Attribute name="position" type="Position" >
    <Input event="Location">
      <EventAssignment source="latitude" target="lat" />
      <EventAssignment source="longitude" target="long" />
      <EventAssignment source="direction" target="dir" />
    </Input>
  </Attribute>
  ...
</ContextSignature>
```

Quelltext 5.3 Kontextsignatur der Kontextattribute für die Zeit und die Position des Fahrzeugs.

Beim Quelltext 5.3 handelt es sich um einen beispielhaften Ausschnitt der Kontextsignatur einer Vorlagendatei für die Anwendungsfälle A bis D. Hier werden die Kontextattribute (mit dem Tag `Attribute`) mit den Namen `time` und `position` für die Zeit bzw. Position des Fahrzeugs eingeführt. Der Wert des Kontextattributs der Zeit ist immer dann von einer möglichen Änderung betroffen, wenn ein Ereignis jedweden Typs auftritt. Dieser Sachverhalt entspricht dem Symbol „*“ in Tabelle 4.1 und wird durch die Angabe von `Event` im Tag `Input` spezifiziert. Der Wert des Kontextattributs `position` ändert sich dagegen nur, wenn ein Ereignis mit dem Typ `Location` auftritt.

Jedes Kontextattribut ist von einem bestimmten *Kontexttyp*, der im Attribut `type` angegeben wird. Der Kontexttyp entspricht intern einer Java-Klasse, die dem Systemdienst gemeinsam mit den Vorlagen und den Ereignistypbeschreibungen zur Verfügung stehen muss. Jedes Kontextattribut wird folglich durch ein Java-Objekt repräsentiert. Über den Tag `EventAssignment` wird dem Systemdienst nun im Sinne von $f_{edit_context}$ vorgeschrieben, welche Ereignisparameter die Datengrundlage des zum Kontextattribut gehörenden Java-Objekts bilden. Hier werden also bestimmte Ereignisparameter auf die entsprechenden Eigenschaften des Java-Objekts abgebildet. Der Wert des Ereignisparameters `latitude` eines Ereignisses vom Typ `Location` wird beispielsweise der Eigenschaft `lat` des Java-Objekts vom Typ `Position` zugewiesen. Der Grund für die Einführung von Kontexttypen liegt in der Möglichkeit, individuelle Distanzfunktionen durch entsprechende Methoden in den Java-Klassen zu spezifizieren. Ein Kontexttyp wird also an eine oder mehrere spezifische Distanzfunktionen gekoppelt. Die Java-Klasse mit dem Namen `Position` kann z.B. die Implementierungen für die beiden im Unterabschnitt 4.2.3.1 vorgestellten Distanzfunktionen aufweisen. In einem späteren Schritt ist der Systemdienst damit in der Lage, Werte des besagten Kontextattributs zu gruppieren. Die Festlegung auf eine bestimmte Distanzfunktion findet dann an entsprechender Stelle in der Konfiguration der Benutzermodellierung statt.

5.2.2.2 Konfiguration der Benutzermodellierung

Die Konfiguration der Benutzermodellierung, also die Spezifikation der relevanten Interaktionsmuster und die Parametrisierung der Gruppierung, wird unter dem Tag `Discovery` zusammengefasst. Für jeden Anwendungsfall wird nun ein eindeutig bezeichneter `Case` angelegt, der das sequentielle Interaktionsmuster, die exklusiven und nicht-exklusiven Vergleichseigenschaften und die Parametrisierung der Gruppierung enthält. Die Parametrisierung des Anwendungsfalls A mit dem dafür eingeführten Bezeichner `RadioChange` wird gekürzt im Quelltext 5.4 dargestellt. Innerhalb des Tags `Action` werden die sequentiellen Interaktionsmuster spezifiziert, während unter dem Tag `Context` die nicht-exklusiven Vergleichseigenschaften und die Einstellungen bzgl. der Gruppierung definiert werden. Die exklusiven Vergleichseigenschaften werden schließlich innerhalb des Tags `Properties` aufgezählt.

```
<Discovery>
    <Case id="RadioChange">
      <Action>...</Action>
      <Properties>...</Properties>
      <Context>...</Context>
    </Case>
    ...
</Discovery>
```

Quelltext 5.4 Ausschnitt aus der Konfiguration für die Benutzermodellierung bzgl. des Anwendungsfalls A

Der Quelltext 5.5 enthält nun ein sequentielles Interaktionsmuster, welches in allgemeiner Weise jede relevante Interaktion aus dem Anwendungsfall A beschreibt (siehe Tabelle 4.2). Dieses sequentielle Interaktionsmuster wird beim Start des Systemdienstes eingelesen und in die EPL von Esper [EsperTech Inc., 2010] konvertiert. Der Systemdienst nutzt dann

die Pattern Matching Engine von Esper, um in der Sequenz von Ereignissen nach relevanten Interaktionen zu suchen. Ein Interaktionsmuster wird hierbei durch das Tag `Pattern` eingeführt. Das Tag `FollowedBy` innerhalb des Interaktionsmusters aus dem Quelltext 5.5 entspricht dem temporalen $\rightarrow$-Operator. Dieser verknüpft den Subausdruck des #-Operators - hier gegeben durch das Tag `Every` - mit dem Subausdruck einer Anwendung des logischen &-Operators (Tag `And`). Die Reihenfolge der beiden Subausdrücke legt dabei die zeitliche Ordnung fest. Das heißt, erst muss ein Ereignis vom Typ „RadioStationChanged“ gefunden werden, bevor der durch eine Konjunktion verknüpfte Subausdruck ausgewertet wird. Da die EPL von Esper keinen $()^{\Delta y}$-Operator aufweist, wurde die in eine zeitliche Bedingung gerahmte Negation eines Ereignisses vom Typ „RadiosenderWechseln“ in zwei durch eine Konjunktion verknüpfte Interaktionsmuster aufgespalten. Alternativ kommt hierbei der in der EPL von Esper vorgestellte spezielle Ausdruck `TimerInterval` zum Einsatz. Dieser Operator verzögert die Auswertung eines Ausdrucks um ein definiertes Zeitintervall. Wird nun der Ausdruck `TimerInterval` mit 10 Sekunden parametrisiert und mit der Negation des Ereignisses eines Radiosenderwechsels konjugiert, so kann die Anwendung des $()^{\Delta y}$-Operators auf eine Negation des Ereignisses vom Typ „RadiosenderWechseln“ nachgebildet werden. Der hier vorgestellte $()^{\Delta y}$-Operator lässt sich also eindeutig auf eine Kombination mehrerer in der EPL von Esper zur Verfügung stehenden Operatoren abbilden. Weitere anwendbare Operatoren können dem XML-Schema der Vorlagen in Anhang A.4 entnommen werden.

```
<Action>
  <Pattern>
    <FollowedBy>
      <Every>
        <Event id="a" type="RadioStationChanged"/>
      </Every>
      <And>
        <TimerInterval>
          <Constant type="Double">10</Constant>
        </TimerInterval>
        <Not>
          <Event type="RadioStationChanged"/>
        </Not>
      </And>
    </FollowedBy>
  </Pattern>
</Action>
```

Quelltext 5.5 Sequentielles Interaktionsmuster für einen relevanten Radiosenderwechsel.

In dieser Umsetzung des Verfahrens zur kontextsensitiven Personalisierung setzt sich eine relevante Interaktion nur aus den Ereignissen zusammen, die im sequentiellen Interaktionsmuster mit eindeutigen Bezeichnern versehen sind. Wird also eine Sequenz von Ereignissen gefunden, die den Eigenschaften des sequentiellen Interaktionsmusters entspricht, so sind in den nachfolgenden Prozessen nur die Ereignisse der Sequenz identifizierbar, die im Interaktionsmuster mit einem Attribut `id` versehen wurden. Im Fall des sequentiellen Interaktionsmusters aus dem Quelltext 5.5 ist nur das zuerst auftretende Ereignis mit einem Bezeichner versehen worden. Eine relevante Interaktion besteht in diesem Fall folglich nur aus einem 2-Tupel mit der internen Bezeichnung `a`. Enthalten ist der Kon-

textvektor zum Zeitpunkt des Radiosenderwechsels und das zugehörige Ereignis vom Typ `RadioStationChanged`. Das zweite und sich durch keinen Bezeichner kennzeichnende Ereignis vom Typ `RadioStationChanged` ist irrelevant, da es zum einen nie vorkommen und zum anderen in der nachfolgenden Verarbeitung keine Rolle spielen darf.

Für die Konfiguration der Gruppierung aller relevanten Interaktionen ist es nun erforderlich, alle exklusiven Vergleichseigenschaften aufzuzählen. Laut der Tabelle 4.3 handelt es sich beim Radiosendernamen um die einzige exklusive Vergleichseigenschaft des Anwendungsfalls A. Wie im Quelltext 5.6 dargestellt, wird folglich nur der Ereignisparameter `stationName` aufgezählt. Das dazugehörige Ereignis einer relevanten Interaktion wird dabei über den 2-Tupel-Bezeichner `a` identifiziert. Bestehen die relevanten Interaktionen in einem bestimmten Anwendungsfall aus mehr als einem 2-Tupel, so können über die eindeutigen Bezeichner der Ereignisse verschiedene exklusive Vergleichseigenschaften identifiziert bzw. definiert werden. Die Bezeichnung der Vergleichseigenschaft über das Attribut `id` des Tags `Property` entspricht letztendlich der Bezeichnung, der auf dieser Vergleichseigenschaft basierenden Gruppeneigenschaft.

```
<Properties>
  <Property id="nameOfRadioStation" event="a" eventProperty="
     stationName"/>
</Properties>
```

Quelltext 5.6 Auflistung der exklusiven Vergleichseigenschaften für den Anwendungsfall A.

Die nicht-exklusiven Vergleichseigenschaften werden bestimmten Gültigkeitsbereichen zugeordnet. Alle in einem Gültigkeitsbereich (hier als `Scope` bezeichnet) aufgezählten nicht-exklusiven Vergleichseigenschaften (hier als `Attribute` bezeichnet), die während der Gruppierung berücksichtigt werden sollen, beziehen sich immer nur auf ein gemeinsames 2-Tupel der relevanten Interaktion. Dieses wird über das Attribut `event` des Tags `Scope` identifiziert. Laut der Tabelle 4.4 existiert für den Anwendungsfall A nur eine nicht-exklusive Vergleichseigenschaft. Infolgedessen wird im Quelltext 5.7 die nicht-exklusive Vergleichseigenschaft mit Namen `location` eingeführt. Sie bezieht sich auf das Kontextattribut `position` des Kontextvektors aus dem 2-Tupel mit der internen Bezeichnung `a`. Die Position des Fahrzeugs zum Zeitpunkt des Radiosenderwechsels ist folglich die einzige nicht-exklusive Vergleichseigenschaft. Jeder Gültigkeitsbereich wird außerdem ebenfalls mit einem eindeutigen Bezeichner versehen. Die aus der Gruppierung aller nicht-exklusiven Vergleichseigenschaften eines Gültigkeitsbereiches resultierenden Gruppeneigenschaften (situativen Bedingungen) werden somit unter einer gemeinsamen Bezeichnung identifizierbar. In diesem Beispiel werden die betreffenden situativen Bedingungen der nicht-exklusiven Vergleichseigenschaften, die zum Zeitpunkt der Personalisierung gelten müssen, unter der Bezeichnung `radioChangeScope` zusammengefasst. Weitere Details zur Verwendung dieses Bezeichners innerhalb der Situationsmuster folgen im nächsten Unterabschnitt 5.2.2.3.

Die für die nicht-exklusive Vergleichseigenschaft geeignete Distanzfunktion wird dann durch das Tag `Similarity` eingeführt. Dieses enthält als Attribute sowohl den Bezeichner der Distanzfunktion (Attribut `distanceFunction`) als auch die für die lokale Ähnlichkeitsfunktion notwendige maximale Distanz (Attribut `epsilonDistance`). Der Bezeichner der Distanzfunktion entspricht in diesem Beispiel dem Namen einer Methode aus der Java-Klasse mit Namen `Position`. Wie bereits oben beschrieben, kann jede dieser Java-Klassen über eine beliebige Anzahl an Distanzfunktionen verfügen. Im Fall des Quelltextes 5.7 wird

```
<Context keepingInterval="30" minCases="3">
  <Scope event="a" id="radioChangeScope">
    <Attribute id="location" feature="position" result="INTERVAL">
      <Similarity distanceFunction="LatLngDistance"
         epsilonDistance="500"/>
    </Attribute>
  </Scope>
</Context>
```

Quelltext 5.7 Auflistung der nicht-exklusiven Vergleichseigenschaften und deren Konfiguration für den Anwendungsfall A.

eine Methode ausgewählt, die die direkte Distanz zwischen zwei GPS-Positionen betrachtet. Zwei GPS-Positionen werden demnach als ähnlich eingestuft, wenn die Distanz zwischen ihnen eine Länge von 500 Metern nicht überschreitet.

Das Attribut `minCases` im Tag `Context` gibt die minimale Anzahl an relevanten Interaktionen an, derer es zur Bildung einer Gruppe bedarf. Der künstliche Verfall wird im Attribut `keepingInterval` angegeben. In diesem Fall werden nur die 30 aktuellsten relevanten Interaktionen bei der Gruppierung berücksichtigt.

5.2.2.3 Konfiguration der Situationen

Die Spezifikation aller Situationen, in denen kontextsensitive Personalisierungen ausgeführt werden sollen, findet unter dem Tag `Detection` statt. Dabei wird die Spezifikation jeder einzelnen Situation in das Tag `Situation` eingerahmt. Die darin enthaltenen generischen Situationsmuster beziehen sich immer auf einen bestimmten Anwendungsfall, der über das Attribut `caseGroup` identifiziert wird. In diesem Beispiel werden die generischen Situationsmuster ausschließlich durch die Ergebnisse der Benutzermodellierung bzgl. des Anwendungsfalls mit dem Bezeichner `RadioChange` (siehe Quelltext 5.4) spezialisiert. Über das Attribut `feedback` kann dem Benutzer außerdem das Recht eingeräumt werden, Empfehlungen des Systems in einer Feedbackansicht einzusehen und zu validieren. Tritt eine mit einem Bezeichner versehene Situation ein, so wird der HMI-Prototyp unverzüglich über diesen Vorfall unterrichtet.

Verschiedene Varianten einer Situation, wie z.B. der Beginn und das Ende einer Situation, werden über unterschiedliche Trigger spezifiziert. Im Anwendungsfall A eignet sich laut der Tabelle 4.5 nur ein Trigger zur Personalisierung. Es sollte ausschließlich zum Beginn der Situation eine Empfehlung an den HMI-Prototyp übermittelt werden. Der Quelltext 5.8 weist somit nur einen Trigger auf. Dieser enthält folglich auch das entsprechende generische Situationsmuster. Bei der Spezifikation des generischen Situationsmusters können - ebenso wie bei der Spezifikation der Interaktionsmuster - alle oben vorgestellten logischen oder temporalen Operatoren zum Einsatz kommen. Zwei Ausnahmen stellen dabei die speziellen Ausdrücke `ContextEntry` und `ContextExit` dar. Diese können nur bei der Spezifikation der generischen Situationsmuster verwendet werden und dienen als Platzhalter für die situativen Bedingungen, die erst zur Laufzeit ermittelt werden. Der Ausdruck `ContextEntry` mit dem Gültigkeitsbereich `radioChangeScope` spezifiziert den Vorfall, bei dem alle situativen Bedingungen, die sich aus dem Gültigkeitsbereich `radioChangeScope` ergeben, erfüllt werden. In diesem Fall tritt der Vorfall folglich immer dann ein, wenn das Fahrzeug ein bestimmtes Straßensegment betritt. Der Ausdruck kürzt damit die in Tabelle 4.5 be-

```
<Detection>
  <Situation id="Radio" feedback="false" caseGroup="RadioChange">
    <Trigger>
      <Every>
        <ContextEntry scope="radioChangeScope">
          <NotEqual>
            <Context attribute="radioStationName"/>
            <CaseGroupProperty property="nameOfRadioStation"/>
          </NotEqual>
        </ContextEntry>
      </Every>
    </Trigger>
  </Situation>
  ...
</Detection>
```

Quelltext 5.8 Spezifikation des generischen Situationsmusters aus Anwendungsfall A.

schriebenen generischen Situationsmuster für den Beginn einer Situation ab. Der Ausdruck `ContextExit` spezifiziert in analoger Weise das Ende einer Situation. Die Anwendung des #-Operators - in Form des Tags `Every` - auf den Ausdruck `ContextEntry` veranlasst den Systemdienst dazu, bei jedem Eintreten dieses Vorfalls den HMI-Prototyp zu benachrichtigen. Somit kann ein mehrmaliges Betreten des entsprechenden Straßensegments während einer Fahrt auch zu einer wiederholten Benachrichtigung und damit Personalisierung führen.

Die Ausdrücke `ContextEntry` und `ContextExit` können optional auch über manuell zu spezifizierende Bedingungen verfügen. Dies ermöglicht die Definition zusätzlicher Nebenbedingungen, die neben den impliziten situativen Bedingungen gelten müssen. Die Bedingungen beziehen sich hier, im Gegensatz zu den Parameterfiltern, jedoch nicht auf Ereignisparameter, sondern auf Kontextattribute. Die Konfiguration im Quelltext 5.8 weist den Systemdienst beispielsweise darauf hin, dass der eingestellte Radiosender zum Zeitpunkt des Betretens einer Situation nicht dem Radiosender der Gruppeneigenschaft `nameOfRadioStation` entsprechen darf. Es findet somit keine Benachrichtigung des HMI-Prototyps statt, wenn der empfohlene Radiosender bereits konfiguriert ist.

5.2.3 Technische Umsetzung und Datenbank

Der in Java implementierte Systemdienst bedient sich des Open Source Apache MINA Frameworks[9] (Multipurpose Infrastructure for Network Applications) zur Umsetzung der Client-Server-Infrastruktur. Das Pattern Matching wird, wie bereits oben beschrieben, mit Hilfe der von EsperTech Inc.[10] als Java-Bibliothek zur Verfügung gestellten Complex Event Processing Engine ausgeführt. Die Complex Event Processing Engine erlaubt es außerdem, durch eine Kombination diverser sequentieller Muster und zusätzlicher datenflusssteuernder und erzeugender EPL-Konstrukte (siehe [EsperTech Inc., 2010]), den Kontextvektor während des Pattern Matchings als EPL-Variable laufend neu zu erstellen und der Sequenz

[9] http://mina.apache.org/
[10] http://esper.codehaus.org/

von Ereignissen hinzuzufügen. Der Prozess der Ermittlung des Kontextes findet somit im Gegensatz zur Gruppierung innerhalb der Complex Event Processing Engine statt.

Relevante Interaktionen und kontextsensitive Präferenzen werden in der objekt-relationalen Datenbank PostgreSQL der PostgreSQL Global Development Group[11] persistiert. Die in diesem Prototyp verwendete PostGIS Erweiterung[12] der PostgreSQL-Datenbank macht es außerdem möglich, Geodaten als standardisierte geometrische Objekte in der Datenbank abzulegen und die für GIS-Systeme optimierten Zugriffs- und Berechnungsmethoden zu nutzen. Die zum Einsatz kommenden Geodaten aller Straßen Deutschlands, die über das OpenStreetMap-Projekt[13] frei bezogen werden können, dienen hier als beispielhafte Netzwerktopologie. Die Geodaten werden benötigt, um den Map-Matching-Algorithmus und die Berechnung der Netzwerk-Distanz zu implementieren. Während die Berechnung der Netzwerk-Distanz durch die freie Routing-Bibliothek pgRouting[14] umgesetzt wird, die eine Netzwerktopologie als PostGIS Datenbank voraussetzt, handelt es sich bei dem im Rahmen dieser Arbeit implementierten Map-Matching-Algorithmus um eine vereinfachte Variante des von Hummel in [Hummel, 2006] vorgestellten Algorithmus.

5.3 Simulation der kontextrelevanten Sensordaten

Der HMI-Prototyp stellt dem Systemdienst zunächst nur die Eingabeereignisse zur Verfügung, die sich aus der Nutzung des HMI-Prototyps direkt ableiten lassen und die den Prozess der kontextsensitiven Personalisierung in irgendeiner Weise beeinflussen können. Dabei kann es sich entweder um physikalische Ereignisse wie z.B. die Betätigung der ZBE oder um Ereignisse der Anwendungsebene wie z.B. dem Radiosenderwechsel handeln. Findet jedoch ein Vorfall statt, der dem Kontext des HMI-Prototyps zugeordnet wird, so werden die dazugehörigen Ereignisse üblicherweise durch externe Komponenten erzeugt und dem In-Car-Infotainment-System durch diese übermittelt. Die externe Komponente des GPS-Sensors erzeugt z.B. in regelmäßigen Abständen ein Ereignis, welches die Position und die Geschwindigkeit des Fahrzeugs enthält. Da der HMI-Prototyp unter Laborbedingungen genutzt wird und sich die Position des ausführenden Computers folglich nicht ändert, ist es notwendig, die Position und umfassender, jede Art von kontextrelevanten Sensordaten, geeignet zu simulieren. Das eigens dafür entwickelte Werkzeug mit der Bezeichnung *ContextSimulator* erlaubt es nun, kontextrelevante Sensordaten wie z.B. die Zeit oder den aktuellen Stand der Tankfüllung über eine Benutzeroberfläche entweder zum Beginn der Laufzeit benutzerspezifisch zu konfigurieren oder während der Laufzeit zu ändern. Die Position und Ausrichtung des Fahrzeugs wird innerhalb des Werkzeugs hingegen entweder auf der Grundlage aufgezeichneter Positionsdaten eines Versuchsfahrzeugs oder durch einen ebenfalls im Rahmen der Forschungsarbeit entwickelten Fahrsimulator simuliert. Die letzten beiden Möglichkeiten bestimmen auch den Modus, in dem sich der ContextSimulator während der Simulation befinden kann. Die beiden nachfolgenden Unterabschnitte widmen sich nun jeweils der Vorstellung eines der beiden Simulationsmodi des ContextSimulators.

[11] http://www.postgresql.org/
[12] http://postgis.refractions.net/
[13] http://www.openstreetmap.de/
[14] http://pgrouting.org/

5.3.1 Simulation auf Grundlage der Sensordaten eines Versuchsfahrzeugs

Die Kommunikation zwischen dem HMI-Prototyp und der Java-Applikation des Context-Simulators basiert auf dem Client-Server Modell. Der ContextSimulator agiert dabei in der Rolle des Servers, an dem sich der HMI-Prototyp für den Empfang kontextrelevanter Ereignisse anmelden kann. Nach dem Start des ContextSimulators wird die Benutzeroberfläche sichtbar, deren Registerkarte zur Einstellung der Simulationsparameter und Visualisierung des aktuellen Kontextes in Abbildung 5.12 dargestellt wird. Ein Ausschnitt aus einer dazu passenden Beispieldatei mit den aufgezeichneten Sensordaten eines Versuchsfahrzeugs wird im Quelltext 5.9 aufgezeigt. Das Höhenprofil der aufgezeichneten Strecke wird im unteren rechten Teil der Ansicht visualisiert während die Startposition auf einer Karte im oberen rechten Teil der Ansicht abgebildet wird. Das passende Kartenmaterial wird dazu vom OpenStreetMap-Projekt-Server[13] bezogen. Einstellen lassen sich die Simulationsparameter der Zeit und der Tankfüllung über die Schaltflächen im unteren linken Teil der Ansicht. Eine Zusammenfassung des aktuellen Kontextes kann dann dem oberen linken Teil der Ansicht entnommen werden.

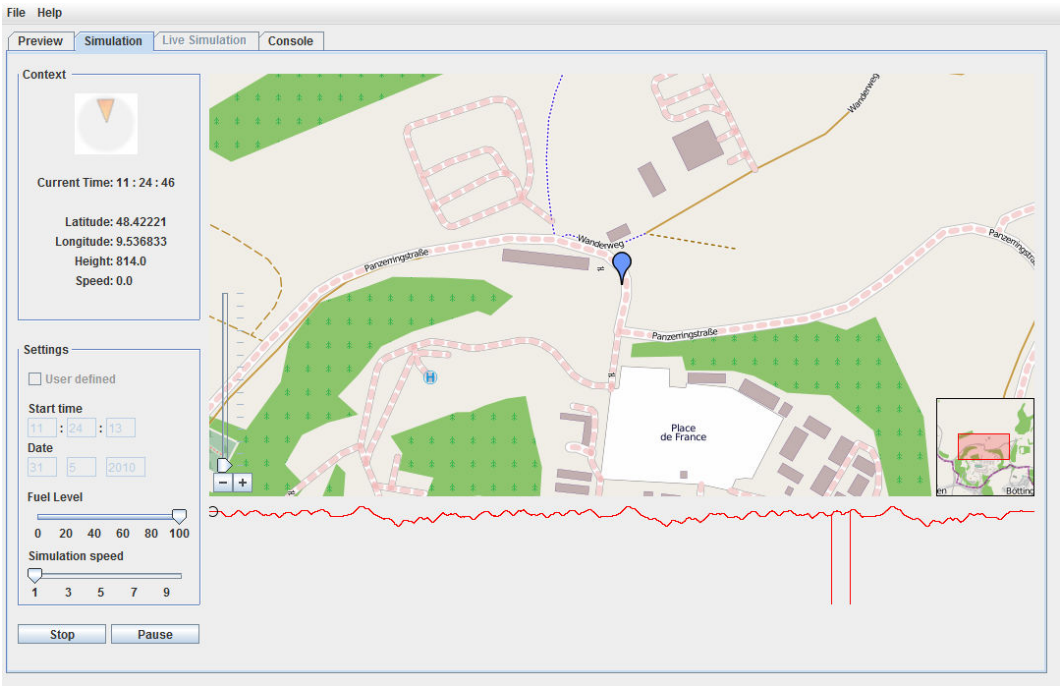


Abb. 5.12 Benutzeroberfläche des ContextSimulators in der Ansicht zur Konfiguration der Simulation auf Basis der aufgezeichneten Positionsdaten eines Versuchsfahrzeugs. Quelle des Kartenmaterials: OpenStreetMap-Projekt (http://www.openstreetmap.de/)

Wird die Simulation gestartet, so werden nacheinander Ereignisse vom Typ „Location“ mit den aktuellen, kontextrelevanten Sensordaten erzeugt und an den HMI-Prototyp übermittelt. Der Startzeitpunkt der Simulationszeit richtet sich entweder nach dem ersten Zeitstempel der Aufzeichnungen oder nach dem vom Benutzer zu definierenden Startzeitpunkt. Die Dauer zwischen den Ereignissen ergibt sich aus der Differenz zwischen den Zeitstempeln direkt aufeinander folgender Aufzeichnungen einzelner Wegpunkte. Der HMI-

Prototyp kann parallel zur Simulation der Sensordaten bedient werden, um die gewünschte kontextsensitive Personalisierung in Echtzeit zu erfahren. Ein einstellbarer Multiplikator ermöglicht es außerdem, die Geschwindigkeit der Simulation um einen bestimmten Faktor zu beschleunigen. Bei einer beschleunigten Simulation ist eine Bedienung des HMI-Prototyps zwar nach wie vor möglich, jedoch aufgrund der Abweichung zwischen der Dauer der manuellen Bedienung und der simulierten Zeit nicht sinnvoll. Diese Funktionalität eignet sich vor allem zum Test und zur Präsentation von kontextsensitiven Personalisierungen, die sich wie im Anwendungsfall B beschrieben, auch ohne eine vorher stattfindende Interaktion mit der Benutzeroberfläche des HMI-Prototyps ergeben können. Der aktuelle Stand der Tankfüllung kann während der Simulation über einen Schieberegler konfiguriert werden. Das dazugehörige Ereignis vom Typ „Fuel“ wird dabei jedoch immer nur dann erzeugt und an den HMI-Prototyp übermittelt, wenn eine Änderung des Tankstandes vorliegt.

```
...
<waypoint>
  <position>
    <lat>48.421967</lat>
    <lon>9.53675</lon>
  </position>
  <time>1275297572130</time>
  <direction>188.6</direction>
  <height>808.0</height>
  <speed>0.0</speed>
</waypoint>
<waypoint>
  ...
</waypoint>
...
```

Quelltext 5.9 Beispielhafter Ausschnitt aus einer Datei mit den aufgezeichneten Sensordaten eines Versuchsfahrzeugs.

5.3.2 Simulation mit Hilfe eines Fahrsimulators

Ein weiterer Simulationsmodus des ContextSimulators erlaubt es, die Sensordaten eines GPS-Sensors in Echtzeit mit Hilfe eines Fahrsimulators zu erzeugen. Dieser wird aus dem ContextSimulator heraus gestartet und erzeugt in regelmäßigen Abständen Ereignisse vom Typ „Location“, sowohl mit der GPS-Position und Ausrichtung, als auch mit der Höhe über dem Meeresspiegel und der Geschwindigkeit des simulierten Fahrzeugs. Die entsprechende Ansicht des ContextSimulators zur Konfiguration und Ausführung des Fahrsimulators wird in Abbildung 5.13 dargestellt. Über das Fadenkreuz auf der Karte kann dem zu simulierenden Fahrzeug eine beliebige Startposition innerhalb der Weltkarte zugewiesen werden. Der Startzeitpunkt der Simulationszeit lässt sich genauso wie im oben beschriebenen Simulationsmodus gegebenenfalls manuell konfigurieren. Wird von dieser Möglichkeit nicht Gebrauch gemacht, so stimmt die Simulationszeit mit der aktuellen Uhrzeit überein. Über den Reiter „Console“ kann dann zur Laufzeit ebenfalls die Tankfüllung konfiguriert werden.

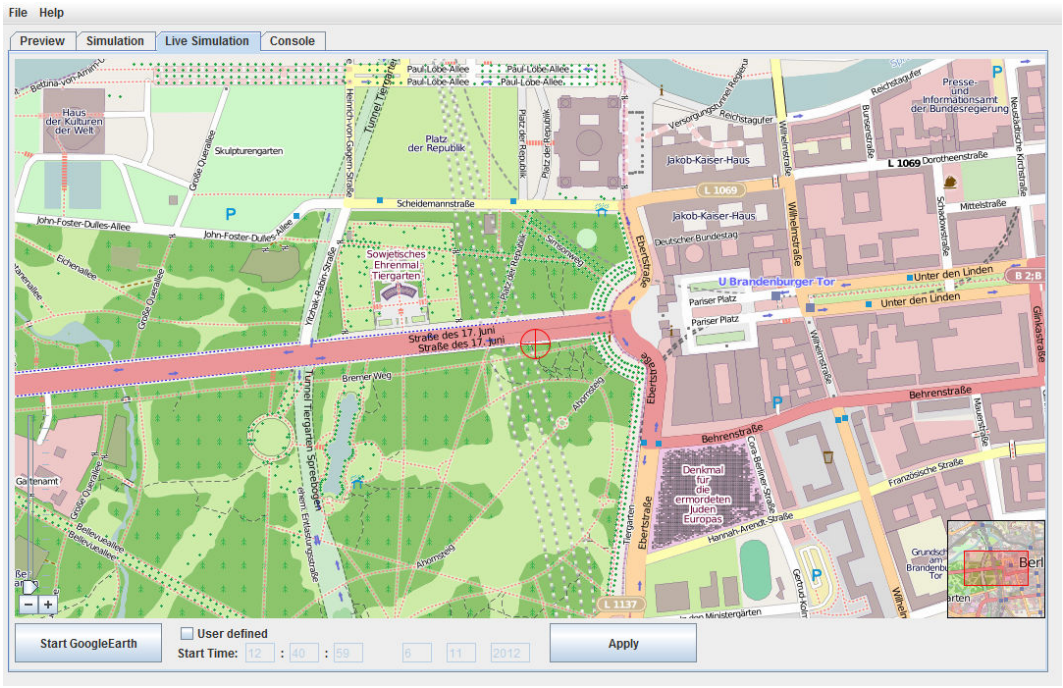


Abb. 5.13 Die Benutzeroberfläche des ContextSimulators mit der Ansicht zur Konfiguration und Ausführung des Fahrsimulators. Quelle des Kartenmaterials: OpenStreetMap-Projekt (http://www.openstreetmap.de/)

Der über diese Ansicht gestartete Fahrsimulator basiert auf einer ausführbaren HTML-Seite in Form einer HTML-Applikation, in der ein entsprechendes grafisches Kartenobjekt, welches durch die Google Maps™ API für Javascript[15] zur Verfügung gestellt wird, integriert ist. Das Kartenobjekt kann über die entsprechende Schnittstelle mit der Position, Höhe über dem Boden, Himmelsrichtung und Neigung parametrisiert werden, um die Karte entsprechend zu kippen, zu verschieben und zu drehen. Dreidimensionale Objekte wie z.B. Häuser und Bäume werden durch den Google Maps™-Server als verortete 3D-Modelle bereitgestellt, durch das Kartenobjekt gerendert und bei Änderung der Sicht aktualisiert. Ergänzt wird das Kartenobjekt um einen mit Adobe® Flash® implementierten Tachometer, der die Geschwindigkeit des Fahrzeugs anzeigt. Abbildung 5.14 zeigt die Fahreransicht der HTML-Applikation.

Die auf Javascript basierende Implementierung der Fahrphysik des Fahrsimulators ist ebenfalls Bestandteil der HTML-Applikation. Gesteuert wird diese Implementierung über ein Lenkrad und zwei Pedale, deren Signale über eine Javascript-Joystick-API[16] empfangen und somit innerhalb der HTML-Seite verarbeitet werden können. Zum Einsatz kommt hier das Modena 360 Lenkrad- und Pedale-Set für Computerspiele der Firma Thrustmaster[17]. Die Position, Geschwindigkeit und Fahrtrichtung des Fahrzeugs wird laufend anhand der Lenkrad- und Pedalpositionen neu berechnet und sowohl an das Kartenobjekt zum Zweck der Aktualisierung der Sicht, als auch an ein auf Adobe® Flash® basierendes Kommuni-

[15] https://developers.google.com/maps/
[16] http://code.google.com/p/javascript-joystick/
[17] http://www.thrustmaster.com

kationsobjekt, welches ebenfalls ein Bestandteil der zugrunde liegenden HTML-Seite ist, übermittelt. Letzteres Objekt registriert sich als Client - genauso wie der HMI-Prototyp - beim ContextSimulator und beliefert diesen in regelmäßigen Abständen mit der aktuellen Position des Fahrzeugs. Die Aufgabe des ContextSimulators besteht nun darin, die Informationen des Fahrsimulators in entsprechende Ereignisse zu verpacken und an den HMI-Prototyp weiterzuleiten.



Abb. 5.14 Visualisierung der Umgebung in der Fahreransicht des Fahrsimulators. Quelle der 3D-Ansicht der Welt: Google Inc. (http://www.google.com/)

Der Fahrsimulator ermöglicht es folglich, die Umgebung des Fahrzeugs unter Laborbedingungen möglichst realitätsnah zu visualisieren und die Steuerung eines Fahrzeugs in einer virtuellen Welt zu simulieren. Gleichzeitig wird der parallel ausgeführte HMI-Prototyp mit den passenden Positionsdaten beliefert. Im Gegensatz zu den im Unterabschnitt 3.2.2 vorgestellten interaktiven Simulationswerkzeugen ist in diesem Fall die Modellierung der virtuellen Welt bereits vorgegeben. Dadurch entfällt der aufwendige Prozess der Erstellung realitätsgetreuer Nachbildungen von Gebäuden und Straßen. Die virtuelle Welt des hier implementierten Fahrsimulators wurde lediglich um einige animierte Lichtsignalanlagen erweitert, um während der Benutzerevaluation im Kapitel 7 ein möglichst echtes Fahrgefühl zu gewährleisten. Durch die Simulation real existierender Ortschaften und Landschaften ist es infolgedessen möglich, die Position des Fahrzeugs durch eine GPS-Position nach dem geodätischen Referenzsystem WGS84 zu modellieren. Ferner kann die Funktionsweise der ortsabhängigen Personalisierung mit jedem beliebigen Ort der Welt simuliert werden, sofern dem Systemdienst die Geodaten, im Fall der optionalen Nutzung der Netzwerk-Distanz, zusätzlich als Netzwerktopologie zur Verfügung stehen.

Teil III
Erprobung

Let us ... suppose the mind to be, as we say, white paper, void of all characters, without any ideas. How comes it to be furnished? Whence comes it by that vast store which the busy and boundless fancy of man has painted on it with an almost endless variety? Whence has it all the materials of reason and knowledge? To this I answer, in one word, from experience.

John Locke, An Essay Concerning Human Understanding, 1690

Kapitel 6
Simulation

Das Untersuchen und Testen von Benutzerschnittstellen, deren Verhalten sich über einen längeren Zeitraum kontinuierlich verändert, stellt die Entwickler vor große Herausforderungen. Die zu testenden Bedienszenarien müssen eindeutig spezifiziert, die entsprechend parametrisierte Benutzerschnittstelle durch beispielhaftes, szenarienabhängiges Benutzer- und Umgebungsverhalten simuliert und das beobachtete Systemverhalten den Erwartungen möglichst automatisiert gegenübergestellt werden. Die für die Simulation notwendigen Eingabeereignisse müssen dabei entweder mit großem Aufwand in realen Fahrzeugen im Rahmen einer breit angelegten Studie aufgezeichnet oder über komplexe Modelle, wie im Unterabschnitt 3.2.1 beschrieben, generiert werden. Während dem Forschungsbereich der Heimautomatisierung bereits frei zugängliche Aufzeichnungen von Sensordaten z.B. aus dem MavHome-Projekt [Cook u. a., 2003] zur Verfügung stehen, liegen dem Forschungsbereich der automotiven Systeme, aufgrund der erst in den letzten Jahren aufkommenden Entwicklung adaptiver, automotiver Benutzerschnittstellen, noch keine vergleichbaren Aufzeichnungen vor. Auch die insbesondere in automotiven Projekten über einen längeren Zeitraum erfassten Positionen der Fahrzeuge, wie beispielsweise bei der Microsoft Multiperson Location Survey [Krumm u. Horvitz, 2005], werden zwar in [Krumm u. Horvitz, 2006] und [Krumm, 2008] als Datengrundlage zur Berechnung des Fahrziels oder des nächsten Fahrmanövers verwendet, aber der Forschungsgemeinde vorerst nicht zur Verfügung gestellt. Ein direkter Vergleich unterschiedlicher Implementierungen der kontextsensitiven Personalisierung ist somit nahezu unmöglich.

Die vorzugsweise automatisierte Generierung der Sensordaten erweist sich ebenfalls als schwierig, da die dazu notwendigen Modelle nicht nur den temporalen Charakter routinierter Verhaltensweisen abbilden müssen, sondern auch die Varianz bzw. Unsicherheit der Sensordaten zu berücksichtigen haben. Wird beispielsweise das täglich wiederholte Ausführen einer bestimmten Interaktion praxisnah simuliert, so sollten sich die aus den Modellen zu generierenden und zu simulierenden Interaktionen idealerweise ähneln und nicht bei jeder Simulation gleichen. Solche Modelle müssen also in der Lage sein, sowohl das routinierte Benutzerverhalten als auch dessen Flexibilität formal zu erfassen. Das alternativ dazu durchzuführende explizite Formulieren ähnlicher Sequenzen von Eingabeereignissen, wie dies im Simulationswerkzeug von Kakousis [Kakousis u. a., 2010] umgesetzt wird, eignet sich aufgrund der hohen Quantität der automotiven Sensordaten nicht für den Einsatz im automotiven Bereich.

Infolge dieser Herausforderungen wurde im Rahmen dieser Forschungsarbeit ein Verfahren zur quantitativen Simulation automotiver Sensordaten entwickelt und im Zuge der Untersuchungen des Systemdienstes angewendet. Das Verfahren umfasst sowohl die Modellie-

rung eines zu testenden Szenarios als auch die darauf folgende automatische Generierung unterschiedlicher, szenarienspezifischer Ausprägungen der Sensordaten. Dieses Verfahren ermöglicht es, in einer frühen Phase der Entwicklung das Langzeitverhalten der kontextsensitiven Personalisierung zu untersuchen, ohne zuvor umfangreiche Studien (zum Zweck der Datenerhebung) durchführen zu müssen. In den nachfolgenden Abschnitten wird das Verfahren detailliert und anhand der beispielhaften Modellierung und Simulation des Benutzerverhaltens aus Unterabschnitt 2.3.4 vorgestellt.

6.1 Simulation des Benutzer- und Umgebungsverhaltens

Das Verhalten der im Systemdienst durchgeführten Datenverarbeitung wird maßgeblich von den zugeführten Eingabeereignissen beeinflusst. Damit handelt es sich beim Systemdienst um ein diskretes, ereignisgesteuertes System (DES). Durch eine ereignisgesteuerte Simulation eines solchen Systems im Labor ist es möglich, dessen Funktionalitäten bzw. dessen Langzeitverhalten bereits vor dem Einsatz in der realen Umgebung auf Fehler bzw. Schwächen hin zu untersuchen. Für eine quantitative, ereignisgesteuerte Simulation wird jedoch vorausgesetzt, dass das potentielle Interaktionsverhalten bereits in Form mehrerer unterschiedlicher Sequenzen von Eingabeereignissen vorliegt. Stehen ausreichend viele solcher szenarienspezifischer Sequenzen zur Verfügung, so können die Interaktionen simuliert und der Zustand des DES anschließend mit dem gegebenenfalls formalen Modell des Szenarios verglichen werden.

Im Fall der kontextsensitiven Personalisierung sind die Eingabeereignisse sowohl von zufälliger als auch wiederkehrender bzw. ähnlicher Natur. Eine ordnungsgemäße Untersuchung des Systemdienstes ist aber nur dann möglich, wenn bereits vor der Simulation bekannt ist, welches wiederkehrende Verhalten des Benutzers in den vorliegenden Sequenzen von Eingabeereignissen enthalten ist und in welcher Weise der Systemdienst auf diese Ereignisse zu reagieren hat. Vor der Untersuchung müssen also sowohl die möglicherweise zum großen Teil per Zufall generierten Sequenzen von Ereignissen als auch die Parametrisierung des Systemdienstes aufeinander abgestimmt werden. Werden die Interaktionen mit dem Systemdienst auf Grundlage dieser generierten Sequenzen von Eingabeereignissen nacheinander automatisiert simuliert, dann können z.B. alle Veränderungen, die sich im Laufe der Zeit bei den zu erwartenden kontextsensitiven Präferenzen einstellen, schrittweise untersucht werden.

Das Benutzer- und Umgebungsverhalten ist jedoch von Natur aus vielseitig, durch Zufälle geprägt und nur sehr schwer vorhersehbar. Außerdem sind beide Verhaltensweisen und deren gegenseitige Abhängigkeiten oftmals derart komplex, dass sie weder erfassbar noch durch ein gemeinsames Modell beschreibbar sind. Wird dennoch ein Modell des Benutzers und dessen Umgebung erstellt, so handelt es sich dabei lediglich um eine approximative Abbildung des realen Benutzer- und Umgebungsverhaltens. Bestimmte Verhaltensweisen werden dabei vereinfacht spezifiziert oder gänzlich ausgeschlossen[1]. Für eine modellbasierte Beschreibung dieser Verhaltensweisen kommen üblicherweise mathematische Modelle zum Einsatz, in denen die zu untersuchenden Objekte durch stochastische Prozesse [Ashbrook u. Starner, 2002; Krumm, 2008] und deren gegenseitige Abhängigkeiten durch logische Verknüpfungen repräsentiert werden. Der Benutzer und dessen Umgebung werden

[1] Aus diesen Gründen sollte vor dem produktiven Einsatz in jedem Fall eine umfangreiche Studie, bei der die Sensordaten in der realen Anwendung erfasst werden, durchgeführt werden.

dabei oftmals vereinfacht, genauso wie der Systemdienst, als ein DES betrachtet, das sich in bestimmten Zuständen befinden, die Zustände durch Ereignisse wechseln und Ereignisse nach außen abgeben kann.

Eine Möglichkeit, unter den oben beschriebenen Voraussetzungen das Verhalten des Benutzers und dessen Umgebung umfassend zu simulieren, besteht beispielsweise darin, einen gemeinsamen Ereignisgraph [Schruben, 1983] zu erstellen, der in grafischer und kompakter Form qualitativ die Logik der Ereignisgenerierung des Benutzers und der Umgebung in einem zu untersuchenden Szenario modelliert. Ein Ereignisgraph besteht aus Knoten, die jeweils stellvertretend für den Vorgang der Generierung eines Ereignisses stehen, und gerichteten Kanten zwischen den Knoten, die die temporalen Beziehungen zwischen zwei Ereignissen kennzeichnen. Die Zeitangaben und/oder Bedingungen an den Kanten spezifizieren dabei die Dauer zwischen der Generierung zweier Ereignisse und unter welchen Bedingungen ein Ereignis auf ein bestimmtes Ereignis folgt. Wird ein solcher szenarienspezifischer Ereignisgraph mehrmalig ausgeführt, so können unterschiedliche Sequenzen von Ereignissen des gleichen Szenarios generiert werden, da die Zeitverzögerungen zwischen den Ereignissen nicht fest angegeben, sondern auch durch Wahrscheinlickeitsverteilungen spezifiziert werden können.

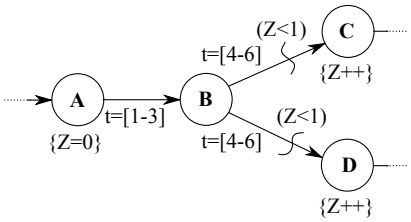


Abb. 6.1 Ein beispielhafter mit einer Verzweigung versehender Ereignisgraph, dessen Simulation unter den aufgeführten Bedingungen zu einer von zwei möglichen Ereignissequenzen (**ABC** oder **ABD**) führt.

Ein Ereignisgraph eignet sich jedoch nicht in jedem Fall zur Spezifikation szenarienspezifischen Benutzer- und Umgebungsverhaltens. Kann beispielsweise in einem Szenario genau eine Verhaltensvariante - von zwei sich gegenseitig ausschließenden Varianten - mit einer bestimmten Wahrscheinlichkeit und nur einmal pro Ablauf bzw. Sequenz auftreten, so kann dies nur über zwei von einem Knoten ausgehende gerichtete Kanten mit zwei verschiedenen, aber aufeinander abgestimmten Wahrscheinlichkeitsverteilungen für die Zeitverzögerungen modelliert werden. Dies entspricht in etwa dem Sachverhalt, dass der Weg von Frau Müller im Anwendungsfall aus Unterabschnitt 2.3.4 entweder über die Landstraße oder über die Autobahn führt. In Abbildung 6.1 wird der Ausschnitt eines entsprechenden Ereignisgraphen gezeigt, welcher eine solche Möglichkeit der Verzweigung enthält. Die Ereignisse **C** und **D** treten demnach vier bis sechs Minuten nach einem Ereignis **B** auf. Da nur eine Variante auftreten darf, ist außerdem eine globale Zustandsvariable Z einzuführen, deren Wert sich nach der Generierung einer Variante ändert und ein Auftreten der alternativen Variante mit Hilfe entsprechender Bedingungen an den zwei gerichteten Kanten ausschließt. Es muss außerdem gewährleistet werden, dass sich die per Zufall ermittelten Dauern **BC** und **BD** beider Varianten nicht gleichen, da der Ereignisgraph in einem solchen Fall die ungültigen Teilsequenzen **BCD** oder **BDC** generiert. Diese sehr umständliche Art und Weise der Beschreibung solcher Sachverhalte kompliziert die Anwendung von Ereignisgraphen insbesondere zur Generierung ähnlicher Varianten des zu untersuchenden,

regelmäßigen Benutzer- und Umgebungsverhaltens. Schließlich tritt eine bestimmte Interaktion oftmals entweder einmal oder überhaupt nicht pro Fahrt auf.

Eine weitere Möglichkeit, das Verhalten des Benutzers und der Umgebung zu simulieren, besteht darin, die Logik der Ereignisgenerierung durch eine diskrete Markov-Kette zu modellieren und diese anschließend auszuführen. Hierbei kann jeder Zustand mit der Generierung eines Ereignisses verknüpft werden. Zwei unterschiedliche Varianten des gleichen Szenarios können ebenfalls über Verzweigungen und den dazugehörigen Wahrscheinlichkeiten modelliert werden. Allerdings können die Zeitintervalle zwischen zwei Zuständen, die über eine Transition miteinander verbunden sind, nicht explizit für jedes Zustandspaar einzeln angegeben werden. Die Intervalle zwischen den Zuständen bzw. Ereignissen wären aufgrund der bei diskreten Markov-Ketten eingeführten, global gültigen Schrittlänge zwischen zwei Zuständen immer gleich. Zeitkontinuierliche Markov-Ketten bieten dagegen die Möglichkeit, die Zeit zwischen zwei Zuständen über die Angabe einer Übergangsrate zu beeinflussen. Die möglichen Aufenthaltsdauern in einem Zustand folgen dabei jedoch immer einer Exponentialverteilung mit dem Parameter der frei wählbaren Übergangsrate. Die Spezifikation eines festen Zeitintervalls ist in diesem Fall nicht möglich.

Die in dieser Forschungsarbeit vorgestellte Methode zur Modellierung des Benutzer- und Umgebungsverhaltens basiert auf einer Modifikation und Erweiterung des *Discrete Event System Specification* (DEVS)-Formalismus, der von Zeigler in [Zeigler, 1976] in erster Linie dazu entwickelt wurde, ein DES zu modellieren und zu analysieren. Ein DEVS-Modell besteht unter anderem aus Zuständen, Transitionen, Eingabe- und Ausgabeereignissen. Mit diesem Formalismus kann das Verhalten eines DES in ähnlicher Weise modelliert werden, wie dies mit dem grafischen StateChart-Formalismus für reaktive Systeme von Harel [Harel u. a., 1987] der Fall ist. Im Gegensatz zu einem StateChart-Modell, in das ein DES-Modell laut Schulz [Schulz u. a., 2000] umwandelbar ist, können in einem DEVS-Modell die Aufenthaltsdauern für jeden Zustand individuell angegeben werden. Im StateChart-Modell gelingt dies erst durch die Hinzunahme einer „timeout“-Funktion. Werden nun die Ausgabeereignisse eines DEVS-Modells, die ausschließlich bei Zustandsübergängen erzeugt werden, als Eingabeereignisse des Systemdienstes interpretiert, so kann über ein DEVS-Modell die Logik der Generierung der Eingabeereignisse eines bestimmten Szenarios spezifiziert werden. Aus welchen Einzelschritten sich der gesamte Prozess der Modellierung und Simulation zusammensetzt und auf welche Art und Weise der DEVS Formalismus modifiziert und erweitert wurde, ist dem nächsten Abschnitt zu entnehmen.

6.2 Quantitative Simulation anhand eines Szenarios

Das hier vorgestellte Verfahren der quantitativen, ereignisgesteuerten Simulation von Benutzer- und Umgebungsereignissen setzt sich aus drei aufeinanderfolgenden Phasen zusammen: *Modellierung*, *Generierung* und *Simulation*. Während der Modellierung wird zunächst ein zu untersuchendes Szenario definiert und anschließend durch ein auf dem modifizierten und erweiterten DEVS-Formalismus basierenden Modell qualitativ beschrieben. In der Phase der Generierung werden auf Grundlage dieses Modells beliebig viele Sequenzen von Ereignissen automatisiert generiert. Jede generierte Sequenz von Ereignissen steht stellvertretend für eine Variante der in diesem Szenario möglichen Interaktionen - sowohl des Benutzers als auch der Umgebung - mit dem System. In einem letzten Schritt werden die Sequenzen von Eingabeereignissen in der zeitlich geordneten Reihenfolge dem zu testenden

Systemdienst zugeführt. Während und nach der Simulation können die kontextsensitiven Präferenzen, die sich laut der Szenarienbeschreibung im Systemdienst einstellen müssen, überprüft werden.

In den nachfolgenden Unterabschnitten werden anhand eines Szenarios der modifizierte und erweiterte DEVS-Formalismus, der algorithmische Ablauf der Generierung und die Simulation im Detail vorgestellt. Die am Montag bis Freitag zwischen 17:00 Uhr und 18:30 Uhr beginnende Rückfahrt von Frau Müller aus Unterabschnitt 2.3.4 dient hierbei als Beispielszenario. Das Ziel der beispielhaften Simulation besteht in diesem Fall darin, den Systemdienst auf die korrekte Erkennung der kontextsensitiven Präferenzen des Radiosenderwechsels nach der Tunnelausfahrt und des Aufrufs der Webseite am Bahnhof hin zu untersuchen. Daher wird in diesem Beispiel nur das Benutzer- und Umgebungsverhalten simuliert, das in der Folge zu den gewünschten kontextsensitiven Präferenzen führt. Die alternative Simulation des Benutzer- und Umgebungsverhaltens, welches die Erkennung von Situationen auslöst, die in der Konsequenz zu Personalisierungen führen, wird hier aufgrund der analogen Vorgehensweise nicht beschrieben.

6.2.1 Modellierung der Interaktionen des Benutzers und der Umgebung

Bei einem DEVS-Modell handelt es sich zunächst einmal entweder um ein atomares oder ein gekoppeltes Modell. Atomare Modelle verfügen im Gegensatz zu den gekoppelten Modellen über keinen hierarchischen Aufbau und können somit keine ineinander geschachtelten Zustände aufweisen. Aufgrund der einfacheren und kompakteren Zusammensetzung basiert der hier vorgestellte modifizierte und erweiterte DEVS-Formalismus ausschließlich auf dem Formalismus atomarer DEVS-Modelle. Ein atomares DEVS-Modell lässt sich grundsätzlich durch ein 7-Tupel beschreiben:

$$\mathbf{M} = < \mathbf{X}, \mathbf{Y}, \mathbf{S}, \delta_{\mathbf{ext}}, \delta_{\mathbf{int}}, \lambda, \mathbf{ta} >$$

$\mathbf{X}$ ist die Menge aller Eingabeereignisse
$\mathbf{Y}$ ist die Menge aller Ausgabeereignisse
$\mathbf{S}$ ist die Menge aller Zustände
$\delta_{\mathbf{ext}} : \mathbf{Q} \times \mathbf{X} \rightarrow \mathbf{S}$, Funktion der externen Transitionen
mit $\mathbf{Q} = \{(s,e) | s \in \mathbf{S}\}$und $0 \leq e \leq \mathbf{ta}(s)$
und e als verstrichene Zeit seit dem letzten Zustandswechsel
$\delta_{\mathbf{int}} : \mathbf{S} \rightarrow \mathbf{S}$, Funktion der internen Transitionen
$\lambda : \mathbf{S} \rightarrow \mathbf{Y}$, Funktion der Ausgabeereignisse:
$\mathbf{ta} : \mathbf{S} \rightarrow \mathbb{R}_0^+$, Zeitfortschrittsfunktion:

Das durch ein DEVS-Modell beschriebene DES befindet sich demnach zu jedem Zeitpunkt immer nur genau in einem Zustand. Der Wechsel von einem Zustand zum nächsten Zustand wird entweder durch ein vor dem Ablauf der in einem Zustand verstrichenen Zeit $e \leq \mathbf{ta}(s)$ eintreffendes Eingabeereignis im Sinne einer externen Transition oder durch den Ablauf der in einem Zustand verstrichenen Zeit $e = \mathbf{ta}(s)$ im Sinne einer internen Transition ausgelöst. Die Aufenthaltsdauer in einem Zustand kann die Werte 0 (sofortiger Zu-

standsübergang), eine beliebige positive Zahl oder auch ∞ annehmen. Im letzten Fall kann ein Zustandsübergang ausschließlich durch ein Eingabeereignis ausgelöst werden. Ausgabeereignisse werden nur bei der Ausführung interner Transitionen abgegeben.

Während der Modellierung des oben genannten Szenarios muss in einem ersten Schritt der Ablauf der Rückfahrt vereinfacht durch eine endliche Zahl von Zuständen beschrieben werden. Eine zeitliche Gliederung der Rückfahrt lässt sich beispielsweise durch die Zustände „vor Tunnel“, „Tunnel“, „nach Tunnel, aber vor Autobahn“, „Autobahn“, „Bahnhof“ und „nach Bahnhof“ erzielen. Gilt es außerdem noch, die Zeit direkt vor und nach der Rückfahrt zu berücksichtigen, so kommen außerdem noch die Zustände „vor Rückfahrt“ und „nach Rückfahrt“ hinzu[2]. Werden die Zustände in der richtigen Reihenfolge nacheinander durch Transitionen miteinander verbunden, dann wird dadurch die temporale Abfolge der spezifischen Zustände definiert. Ist die Menge aller Eingabeereignisse $\mathbf{X}$ leer und handelt es sich bei jeder Aufenthaltsdauer $\mathbf{ta}(s)$ immer um eine endliche reelle Zahl, so ergibt sich die Dauer der Rückfahrt aus der Summe der Aufenthaltsdauern jedes zu durchlaufenden Zustandes.

Eine beispielhafte - auf dem DEVS-Formalismus basierende - grafische Umsetzung des Szenarios ist der Abbildung 6.2 zu entnehmen. Die Zustände werden durch abgerundete Rechtecke repräsentiert und mit einem Namen und der maximalen Aufenthaltsdauer $\mathbf{ta}(s)$ versehen. Den Aufenthaltsdauern liegt in diesem Fall die Einheit Minuten zugrunde. Die in der Regel in den Metadaten enthaltenen Ein- und Ausstiegspunkte aus dem Modell werden durch schwarze Kreise (ohne und mit Umrandung) angegeben. Die an den Transitionen (gerichteten Kanten) mit einem schwarzen Quadrat gekennzeichneten Ereignisse entsprechen den Ausgabeereignissen. Der Übersichtlichkeit halber bleiben die Inhalte der Ausgabeereignisse in Form des Radiosenders *JazzFM* und der Webseitenadresse *www.bahn.de* verborgen. Die Zustände „nach Tunnel, aber vor Autobahn“ und „Bahnhof“ wurden dabei jeweils in zwei separate Zustände aufgeteilt, um den Radiosenderwechsel und den Webseitenaufruf in den passenden Kontext zu integrieren.

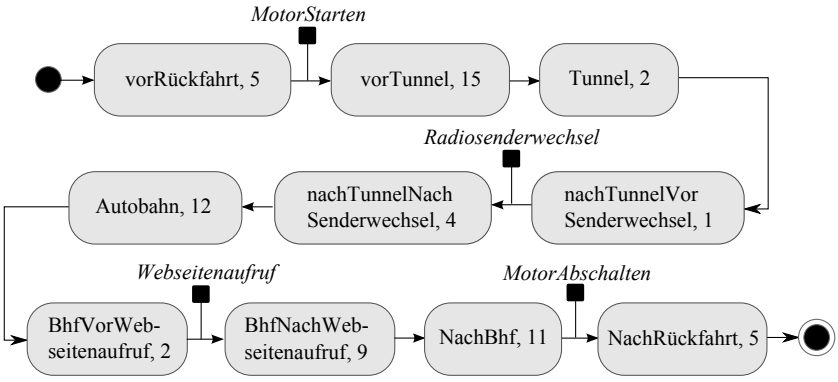


Abb. 6.2 Mögliches DEVS-Modell des zeitlichen Ablaufs der Rückfahrt bzgl. des Szenarios aus Unterabschnitt 2.3.4.

Ein solches DEVS-Modell kann nun mit Hilfe einer Ausführungsumgebung beliebig oft simuliert werden. Die zu erwartende Zustandsabfolge und Reihenfolge der Ausgabeereignisse ist bei jeder Ausführung identisch. Wird jedes Ausgabeereignis, das aufgrund der

[2] Welchem Zweck die beiden Zustände „vor Rückfahrt“ und „nach Rückfahrt“ bei der Simulation dienen, wird im nächsten Unterabschnitt 6.2.2 beschrieben.

Aufenthaltsdauern auch über einen relativen Zeitstempel verfügt, als ein Eingabeereignis des Systemdienstes interpretiert, dann „produziert“ das DEVS-Modell bei jeder Ausführung immer die gleiche szenarienspezifische Sequenz von Eingabeereignissen für den Systemdienst. Das DEVS-Modell bildet bis dato folglich nur die Logik der Ereignisgenerierung für die Vorgänge des Benutzers ab. Eine Ausnahme stellt die Zeit dar, die sich aus den Aufenthaltsdauern der Zustände ergibt. Die Umgebung, beispielsweise in Form des Ortes, ergibt sich nach wie vor nur informell aus dem Namen der Zustände.

Flexible Abläufe

Da sich der Benutzer in einem bestimmten Kontext jedoch nicht immer auf die gleiche Art und Weise verhält, gilt es, bei der Modellierung diese unterschiedlichen Verhaltensweisen zu berücksichtigen. Es kann beispielsweise vorkommen, dass Frau Müller den Radiosenderwechsel oder den Webseitenaufruf nicht bei jeder Rückfahrt durchführt. Außerdem wählt Frau Müller für die Rückfahrt - abhängig von der Verkehrssituation - eine von zwei alternativen Routen, die entweder über die Autobahn oder die Landstraße führt. Die Entscheidungen trifft Frau Müller dabei entweder zufällig oder auf der Basis von Informationen, die dem Modell nicht zur Verfügung stehen. Eine Möglichkeit, diese Variabilität im Modell zu gewährleisten, besteht sicherlich darin, für jedes potentielle Verhalten ein separates deterministisches DEVS-Modell einzuführen. Die Entscheidung darüber, welche Variante simuliert werden soll, muss dann vor der Generierung entweder für jede zu simulierende Rückfahrt manuell getroffen oder durch einen Zufallsgenerator bestimmt werden. Die Multiplikation der Möglichkeiten führt in dem hier diskutierten Szenario bereits zu acht unterschiedlichen DEVS-Modellen. Je mehr Varianten des Verhaltens zulässig sind, desto umfangreicher gestaltet sich die Modellierung. Kommt es beispielsweise vor, dass Frau Müller auf den Rückfahrt den Bahnhof passiert, ohne am Bahnhof zu halten, so steigt die Anzahl der notwendigen DEVS-Modelle bereits auf 12. Dieser Lösungsansatz eignet sich infolgedessen nicht zur Modellierung von Szenarien, die aus einer großen Anzahl von Varianten bestehen.

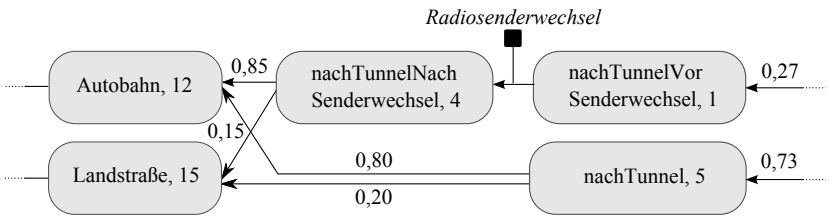


Abb. 6.3 Szenarienspezifisches STDEVS-Modell.

In [Kofman u. Castro, 2006] erweitert Kofman den DEVS-Formalismus um die Möglichkeit, nicht-deterministisches Verhalten eines DES zu modellieren. Bei einem auf dem sogenannten STDEVS -Formalismus (stochastische DEVS) basierenden Modell werden die internen und externen Transitionen nicht als eindeutige Abbildungen, sondern durch Wahrscheinlichkeitsfunktionen beschrieben. Dabei gilt es, genauso wie bei den Markov-Modellen, die Wahrscheinlichkeiten zu definieren, mit denen das DES von einem Zustand $s \in \mathbf{S}$ in einen Zustand $s' \in \mathbf{S}$ übergeht. Damit lassen sich die oben beschriebenen Varianten des Benutzer- und Umgebungsverhaltens in einem STDEVS-Modell vereinen. Abbildung 6.3 enthält dazu

einen Ausschnitt des grafischen, szenarienspezifischen Modells aus Abbildung 6.2, welches mit Hilfe des STDEVS-Formalismus um variables Verhalten erweitert wurde. Die Transitionen werden dabei mit den entsprechenden Wahrscheinlichkeiten $p \in [0,1]$ versehen. Verfügt ein Zustand über mehr als eine ausgehende Transition, dann müssen alle ausgehenden Transitionen mit den entsprechenden Wahrscheinlichkeiten versehen werden. Für alle ausgehenden Transitionen gilt dabei, dass die Summe der Wahrscheinlichkeiten immer eins betragen muss. Verfügt ein Zustand jedoch nur über eine ausgehende Transition, die in jedem Fall nach dem Ablauf der maximalen Aufenthaltsdauer ausgeführt wird, dann kann die Wahrscheinlichkeit der Übersichtlichkeit halber auch weggelassen werden.

Flexible Zeitangaben

In der Praxis können sich von Fahrt zu Fahrt nicht nur die Abläufe des Benutzerverhaltens, sondern auch die Zeitpunkte, an denen bestimmte Interaktionen des Benutzers oder der Umgebung mit dem System stattfinden, ändern. Bisher wurde mit Hilfe eines konstanten Wertes für jeden Zustand festgelegt, wie lange sich das System in einem bestimmten Zustand aufhält. Die Zeitintervalle zwischen den verschiedenen, durch die Ausgabeereignisse gekennzeichneten Interaktionen des Benutzers sind somit, abgesehen von dem oben eingeführten variablen Verhalten, immer gleich. Zwischen den Ereignissen mit den Typen „MotorStarten“ und „Radiosenderwechsel“ vergehen beispielsweise immer 18 Minuten. Wird bei jeder Ausführung des Modells die absolute Anfangszeit explizit - in Form der Metadaten des Modells - angegeben und variiert, so können zwar die absoluten Zeitstempel der Ausgabeereignisse berechnet werden und unterschiedlich sein, aber nicht deren Differenzen. Die Dauer zwischen dem Start des Motors und der Einfahrt in den Tunnel kann in der Praxis jedoch beispielsweise Werte zwischen 10 und 20 Minuten annehmen.

Um die Variabilität der Aufenthaltsdauer im STDEVS-Formalismus zu berücksichtigen, wird die Definition der Zeitfortschrittsfunktion modifiziert. Sie bildet einen beliebigen Zustand folglich nicht auf eine reelle Zahl, sondern auf ein 2-Tupel aus reellen Zahlen ab. Die erste reelle Zahl steht dabei für die minimale Aufenthaltsdauer, während die zweite reelle Zahl die maximale Aufenthaltsdauer angibt. Die konkreten Aufenthaltsdauern werden dann erst zum Zeitpunkt der Ausführung des Modells mit Hilfe eines Zufallsgenerators berechnet. Damit steht die Dauer der simulierten Zeit, beispielsweise die Länge einer Rückfahrt, erst nach der Ausführung des Modells fest. Die relativen Zeitstempel der einzelnen Ausgabeereignisse ergeben sich somit aus der Summe der Aufenthaltsdauern aller zuvor besuchten Zustände.

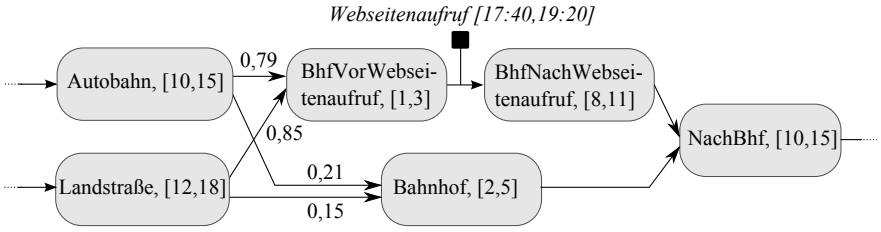


Abb. 6.4 Szenarienspezifisches Teilmodell mit variablen Zeitangaben.

Bei einigen der zu untersuchenden kontextsensitiven Präferenzen handelt es sich um tägliche bzw. zeitliche Routinen. Die dafür zu simulierenden Interaktionen des Benutzers müssen folglich zu bestimmen Tageszeiten oder innerhalb bestimmter Zeitintervalle auftreten. Wird beispielsweise bei dem Webseitenaufruf von Frau Müller auch die Tageszeit berücksichtigt, dann gilt es, im Modell die Zustände, in denen sich Frau Müller am Bahnhof aufhält, auf eine bestimmte Tageszeit einzugrenzen. Nur so kann gewährleistet werden, dass die Generierung des Ausgabeereignisses mit dem Typ „Webseitenaufruf“ innerhalb des gewünschten Kontextes erfolgt. Über eine geeignete Wahl der Aufenthaltsdauern aller zuvor zu durchlaufenden Zustände lässt sich eine solche Bedingung in das Modell integrieren. Bei einer hohen Anzahl von Zuständen kann dies während der Modellierung jedoch schnell zu Fehlern führen, da alle minimalen und maximalen Aufenthaltsdauern aufeinander abgestimmt werden müssen.

Aus diesen Gründen wird es in der hier vorgestellten Methode ermöglicht, Transitionen mit zeitlichen Bedingungen zu versehen. Diese Bedingungen spezifizieren dabei, wann die Ausführung der zugehörigen Transition zulässig ist. Eine STDEVS-konforme Integration solcher Bedingungen setzt jedoch voraus, dass die zeitlichen Bedingungen während der Ausführung des Modells durch Eingabeereignisse $x \in \mathbf{X}$ simuliert werden, die von einem Zeitgeber bereitgestellt werden müssen. Im Gegensatz zum STDEVS-Formalismus ist es hier außerdem möglich, eine Transition sowohl mit einem Eingabeereignis, als auch mit einem Ausgabeereignis zu versehen. Der Zeitstempel des Ausgabeereignisses entspricht in diesem Fall dem Zeitstempel des Eingabeereignisses. In Abbildung 6.4 wird ein Ausschnitt des szenarienspezifischen Modells gezeigt, das um variable Zeitangaben erweitert wurde. Die zeitliche Bedingung an der Transition vom Zustand „BhfVorWebseitenaufruf“ zum Zustand „BhfNachWebseitenaufruf“ gibt durch die Angabe der minimalen und maximalen Tageszeit an, in welchem Zeitrahmen das Ausgabeereignis mit Typ „Webseitenaufruf“ auftreten darf. Optional kann das Zeitintervall der Aufenthaltsdauer auch weggelassen werden. In diesem Fall berechnet sich die konkrete Aufenthaltsdauer auf Basis der Aufenthaltsdauern benachbarter Zustände und den Metadaten des Modells (z.B. Start- und Endzeit). Die Modellierung von Zyklen im Modell ist nur dann erlaubt, wenn die Zustände und Transitionen innerhalb des Zyklus über keine konkreten Zeitangaben verfügen.

Periodische Ereignisse

Das bisherige Modell des Szenarios berücksichtigt - mit Hilfe der Spezifikation entsprechender Ausgabeereignisse - ausschließlich Eingabeereignisse des Systemdienstes, die einmalig und vom Benutzer direkt ausgelöst werden. Die zu spezifizierenden Ausgabeereignisse können dabei jedoch jeden beliebigen Typ annehmen. Damit ist es auch möglich, einmalig auftretende Ereignisse der Umgebung als Ausgabeereignisse zu spezifizieren. Ein solches Ereignis könnte beispielsweise eine Staumeldung für die Autobahn sein, die kurz vor dem Erreichen der Autobahn eintrifft und Frau Müller dazu veranlasst, die Fahrt auf der Landstraße fortzusetzen. Viele der Sensoren, die die Vorgänge der Umgebung zu Ereignissen verarbeiten, geben die abgetasteten Daten jedoch periodisch an den Systemdienst weiter. Dies betrifft insbesondere die Aspekte der Umgebung, die sich laufend (also in kurzen Zeitabständen) ändern. Dazu gehören beispielsweise die Geschwindigkeit oder die Position des Fahrzeugs. Im Unterabschnitt 4.2.5 wurde bereits darauf hingewiesen, dass moderne GPS-Sensoren im Zusammenspiel mit weiteren Sensoren im Betrieb mehr als 20 GPS-Positionen pro Sekunde berechnen können. Da nur maximal ein Ausgabeereignis an jeder

Transition annotiert werden darf, ergäbe dies bei einer durchschnittlichen Rückfahrt von 30 Minuten bis zu 36000 Transitionen, die es im Modell zu berücksichtigen gilt. Ein Modell dieser Größenordnung kommt jedoch in der Praxis, aufgrund des bei der Erstellung zu erwartenden Aufwands und der Komplexität der Modifikationen, nicht zum Einsatz.

Um dennoch die Modellierung periodischer Ereignisse zu ermöglichen, können die Zustände um eine Beschreibung der in diesem Zustand periodisch auftretenden Ereignisse erweitert werden. Dabei gilt es, den Ereignistyp, den Inhalt der Ereignisse und die Dauer zwischen den Ereignissen (Zwischenankunftszeiten) innerhalb der Zustände anzugeben. Der Inhalt der Ereignisse setzt sich in den meisten Fällen aus Werten zusammen, die nur innerhalb des entsprechenden Zustands auftreten können. Innerhalb des Zustandes „vor-Tunnel“ können z.B. nur periodische Ereignisse vom Typ „Position“ auftreten, deren GPS-Koordinaten Positionen beschreiben, die der Region oder den Straßen vor dem Tunnel zugeordnet werden können. Die Zwischenankunftszeiten der Ereignisse können wahlweise als festes Zeitintervall oder über die Angabe einer minimalen und maximalen Zwischenankunftszeit angegeben werden. Die Generierung der periodischen Ereignisse mit den entsprechenden Zeitstempeln und Inhalten findet dann während der Ausführung des Modells statt.

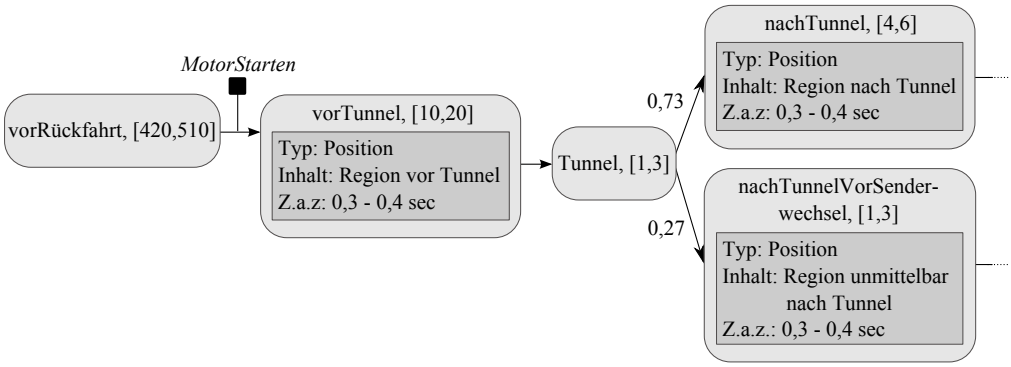


Abb. 6.5 Szenarienspezifisches Teilmodell mit variablen Zeitangaben und periodischen Ereignissen.

Der um Zeitangaben modifizierte STDEVS-Formalismus dient also in erster Linie der Spezifikation des zeitlichen Rahmens einmalig und periodisch auftretender Ereignisse. Die Einführung der zustandsabhängigen Spezifikation periodischer Ereignisse führt jedoch zu keiner Modifikation, sondern zu einer Erweiterung des zugrunde liegenden Formalismus. Der Graph aus Abbildung 6.5 zeigt einen Ausschnitt des um periodische Ereignisse erweiterten Modells. Der potentielle Inhalt der Ereignisse wird dabei nur auf informeller Ebene angegeben. Der Grund liegt darin, dass die Spezifikation des Inhalts periodischer Ereignisse im Gegensatz zur Spezifikation der Zwischenankunftszeiten diverse Formen annehmen kann. Im Fall der periodischen Ereignisse mit dem Typ „Position“ kann beispielsweise eine geordnete Liste aller abzufahrenden Straßenabschnitte oder eine über zwei GPS-Positionen definierte Region vorliegen. Welche Möglichkeiten zur Wahl stehen, wird durch die Implementierung vorgegeben. Das vollständige Modell des Szenarios aus Unterabschnitt 2.3.4 kann dem Anhang B.1 entnommen werden.

6.2.2 Generierung und Simulation der Interaktionen

Vor der Ausführung des szenarienspezifischen Modells gilt es zunächst, das Modell auf syntaktische Fehler und das Vorhandensein von sich widersprechenden Bedingungen zu prüfen. Letztere Problematik betrifft insbesondere Zeitangaben, die im Kontext anderer Zeitangaben und der Metadaten des Modells möglicherweise nicht erfüllbar sind. Wurde eine aus einem Zustand ausgehende Transition beispielsweise mit einem konkreten Zeitpunkt versehen, der zeitlich vor einer Zeitangabe liegt, die an einer dem Zustand eingehenden Transition annotiert wurde, so führt dies zu einem zeit-inkonsistenten Modell. Eine solche Überprüfung kann bereits während der Modellierung, mit entsprechender Unterstützung durch das Modellierungswerkzeug, automatisch durchgeführt werden.

Der erste Verarbeitungsschritt bei der Generierung einer szenarienspezifischen Sequenz von Ereignissen besteht darin, den spezifischen Ablauf der potentiellen Interaktion festzulegen. Dazu wird das Modell vom Einstiegs- bis zum Ausstiegspunkt einmalig traversiert. Verfügt ein Zustand während der Traversierung über mehr als eine ausgehende Transition, so wird entsprechend der Wahrscheinlichkeit - per Zufallsgenerator - eine der ausgehenden Transitionen ausgewählt und der nachfolgende Zustand bestimmt. Das Ergebnis der Traversierung besteht in der Folge aus einem beispielhaften Ablauf der Interaktionen, der nur einen bestimmten Teilgraph des Modells betrifft. Beim diesem Teilgraph handelt es sich wiederum um ein mit flexiblen Zeitangaben versehenes DEVS-Modell.

Im nächsten Schritt gilt es, den festgelegten Ablauf bzw. den Teilgraph mit einem konkreten zeitlichen Rahmen zu versehen. Dazu muss für jeden Zustand die Aufenthaltsdauer, und für jede Transition der Ausführungszeitpunkt bestimmt werden. Dieser sogenannte Scheduling-Prozess vollzieht sich auf der Grundlage sowohl der in den Zuständen und an den Transitionen angegebenen Zeitintervallen, als auch auf den sich in den Metadaten befindenden Start- und Endzeitpunkten des Modells. Ausgehend vom oben beschriebenen Szenario kann der Startzeitpunkt beispielsweise 10:00 Uhr und der Endzeitpunkt 23:59 Uhr betragen. Der erste Zustand „vorRückfahrt“ stellt sich somit immer um 10:00 Uhr ein, während der Zustand „nachRückfahrt“ immer um 23:59 endet. Durch die flexible Angabe der Aufenthaltsdauer von 420 Minuten (7 Stunden) bis 510 Minuten (8,5 Stunden) im Zustand „vorRückfahrt“ kann infolgedessen gewährleistet werden, dass das Ereignis „Motor-Starten“ immer nur im Kontext zwischen 17:00 Uhr und 18:30 auftreten kann. In welchem Moment der Motor abgestellt wird, ergibt sich aufgrund des im Zustand „nachRückfahrt“ fehlenden Zeitintervalls aus den Aufenthaltsdauern und Zeitpunkten aller zuvor durchlaufenen Zustände und Transitionen. Die Auswahl der konkreten Aufenthaltsdauern und Ausführungszeitpunkte der Transition erfolgt während des Scheduling-Prozesses mit Hilfe eines Zufallsgenerators und der im Modell anzugebenden Wahrscheinlichkeitsverteilungen. Ist der zeitliche Rahmen des Ablaufs durch das Scheduling eindeutig bestimmt, so besteht das Ergebnis aus einem Modell, welches bis auf den gleichzeitigen Einsatz von Ein- und Ausgabeereignissen an einer Transition auf dem ursprünglichen DEVS-Formalismus beruht.

Der letzte Schritt der Generierung besteht in der Ausführung des Modells und der damit verbundenen Erstellung einer zeitlich geordneten Liste von Eingabeereignissen für den Systemdienst. Jedes Ausgabeereignis führt dabei, wie oben beschrieben, zur Erstellung eines entsprechenden Eingabeereignisses für den Systemdienst. Innerhalb der Zustände werden dann die Sequenzen von periodischen Ereignissen in Form von zeitlich geordneten Teilsequenzen erstellt und der bestehenden Liste von Eingabeereignissen angefügt. Die Berechnung der dabei notwendigen Zwischenankuftszeiten erfolgt ebenfalls auf Grundlage eines

Zufallsgenerators und der den Ereignissen im Modell zugewiesenen Wahrscheinlichkeitsverteilung der Zwischenankunftszeiten.

Die abschließende Simulation widmet sich der in der zeitlich geordneten Reihenfolge stattfindenden Übermittlung aller generierten Eingabeereignisse an den Systemdienst. Die Dauer zwischen den Übermittlungen zweier Ereignisse kann dabei entweder immer einem konstanten Wert (*periodische Übermittlung*) entsprechen oder sich jeweils aus der Differenz zweier aufeinander folgender Zeitstempel berechnen (*zeitkonforme Übermittlung*). Die erste Variante eignet sich insbesondere für Tests, die die funktionalen Anforderungen über einen längeren, simulierten Zeitraum untersuchen, während sich die zweite Variante zur Untersuchung nicht-funktionaler Anforderungen, wie z.B. der maximal in einer Sekunde zu verarbeitenden Ereignisse, eignet.

Bei der periodischen Übermittlung wird in den Metadaten definiert, in welcher Periode die Ereignisse übermittelt werden. Der Systemdienst bezieht in diesem Fall die simulierte Zeit nicht von einem eigenen Zeitgeber, sondern aus den Zeitstempeln der eintreffenden Ereignisse (externer Zeitgeber). Dies führt jedoch dazu, dass neben den oben generierten Ereignissen auch spezielle periodische Zeitereignisse (ohne Parameter, aber mit Zeitstempeln versehen) an den Systemdienst übermittelt werden müssen. Der Grund dafür liegt in der im Systemdienst stattfindenden Auswertung komplexer Muster, die über temporale Operatoren verfügen können. Das Interaktionsmuster des Anwendungsfalls A in Tabelle 4.2 wird beispielsweise aufgrund des externen Zeitgebers im Systemdienst erst dann erkannt, wenn ein nach 10 Sekunden eintretendes Ereignis den Systemdienst auf das Ablaufen der Zeitspanne aufmerksam macht. Tritt das erste Ereignis nach dem Radiosenderwechsel jedoch erst nach z.B. 50 Sekunden simulierter Zeit auf, dann findet die Erkennung, bezogen auf die simulierte Zeit, zum falschen Zeitpunkt statt. Die kleinste Zeiteinheit, die innerhalb der komplexen Muster im Systemdienst beschrieben werden kann, bestimmt in dieser Variante die Periode, mit der Zeitereignisse generiert und zum Zweck der Zeitgebung an den Systemdienst übermittelt werden müssen. Bei der zeitkonformen Übermittlung entfallen die Zeitereignisse, da die Zeitgeber der Simulationsumgebung und des Systemdienstes übereinstimmen oder synchronisiert sind.

6.2.3 Technische Umsetzung

Der in [Hritsevskyy, 2012] von Hritsevskyy entworfene Prototyp zur Generierung und Simulation des Benutzer- und Umgebungsverhaltens setzt das oben beschriebene Verfahren beispielhaft innerhalb einer Java-Anwendung um. Die Anwendung liest ein auf XML-Basis erstelltes Modell ein, generiert die potentiellen Interaktionen entsprechend dem vorgestellten Verfahren, verbindet sich über eine TCP/IP Verbindung mit dem Systemdienst und führt anschließend die periodische Übermittlung der Ereignisse aus. Eine zeitkonforme Übermittlung ist nicht möglich. Die durch die Simulation hervorgerufenen kontextsensitiven Präferenzen des Systemdienstes können nach oder während der Simulation über eine zusätzliche Ansicht direkt eingesehen werden. Somit können die Auswirkungen einzelner Interaktionen auf den Prozess der Ermittlung kontextsensitiver Präferenzen im Einzelnen untersucht werden.

Die Auswahl des Modells erfolgt über die Hauptansicht der Anwendung, die in Abbildung 6.6 gezeigt wird. Optional kann auch ein zusätzliches *Noise-Modell* geladen werden, in welchem die Generierung von Ereignissen spezifiziert wird, die zunächst in keiner be-

kannten Relation zum zu simulierenden Szenario stehen, aber jederzeit auftreten können. Die oben beschriebene Staumeldung ist ein Beispiel eines solchen Umgebungsereignisses. Die Metadaten können in der Hauptansicht sowohl eingesehen als auch modifiziert werden. Dabei kann der Startzeitpunkt, die Anzahl der pro Sekunde zu übermittelnden Ereignisse, die zu simulierenden Wochentage und die Anzahl der zu generierenden und simulierenden Ereignissequenzen angegeben werden. Der Startzeitpunkt wird nach jedem Durchlauf, bestehend aus der Generierung und Übermittlung einer Ereignissequenz, automatisch um 24 Stunden erhöht. Liegt der dadurch ermittelte Startzeitpunkt an einem Wochentag, den es laut Metadaten auszuschließen gilt, dann wird der Startzeitpunkt so lange um jeweils 24 Stunden erhöht, bis ein zulässiger Wochentag gefunden ist. In einer zusätzlichen Konsolenansicht können außerdem die speziellen Ereignisse des Systemdienstes, wie beispielsweise die Erkennung einer relevanten Interaktion oder Situation, eingesehen werden.

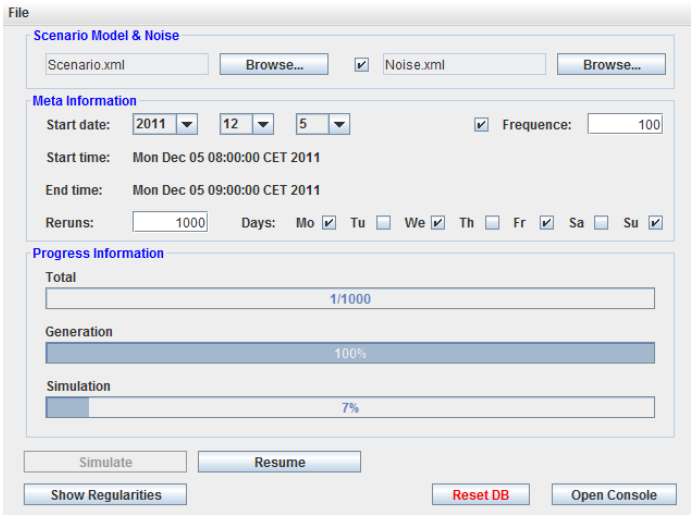


Abb. 6.6 Hauptansicht der Anwendung zur Generierung und Simulation szenarienspezifischen Benutzer- und Umgebungsverhaltens.

Kapitel 7
Evaluation

Der Erfolg eines benutzer- und kontextadaptiven Systems im automotiven Bereich hängt von diversen Faktoren, wie z.B. der technischen Umsetzung, dem Bedarf auf Seiten des Kunden oder dem Kosten-Leistungsverhältnis sowohl aus Sicht der Automobilhersteller als auch aus Sicht der Fahrzeugkäufer ab. Um vor einer möglichen Serienentwicklung dennoch einen Eindruck davon zu erhalten, wie hoch der Nutzen oder die Praktikabilität eines solchen Systems in diesem speziellen Anwendungsgebiet sein kann, werden die Prototypen üblicherweise durch diverse Kunden- und Entwicklergruppen evaluiert. Während bei der Benutzerevaluation die Untersuchungen zur Akzeptanz, Einfachheit und praktischen Nutzbarkeit einer Benutzeroberfläche im Vordergrund stehen, werden bei der Untersuchung durch die Entwicklergruppen die technischen Voraussetzungen evaluiert. Letztere Form der Evaluation würde im Fall des in dieser Forschungsarbeit vorgestellten Verfahrens z.B. die Flexibilität und Mächtigkeit der Konfiguration über die Vorlagen zum Gegenstand der Betrachtungen machen. Eine Benutzerevaluation würde hingegen die kontextsensitiv personalisierten Schnellstartbuttons, die kontextsensitiv personalisierte Automatisierung und die Feedbackansicht hinsichtlich ihrer intuitiven Bedienbarkeit oder Nachvollziehbarkeit prüfen.

Die Durchführung einer Benutzerevaluation im automotiven Bereich erfordert es in der Regel, eine bestimmte Anzahl von Fahrzeugen mit den zu überprüfenden Prototypen zu bestücken und Probanden aus dem potentiellen Käuferkreis anzuwerben. Bei der Untersuchung adaptiver Benutzerschnittstellen ergibt sich außerdem noch die Schwierigkeit, dass eine umfassende Beurteilung der z.B. kontextsensitiven Personalisierung nur dann möglich ist, wenn die Benutzerevaluation im Sinne einer Langzeitevaluation über einen längeren Zeitraum durchgeführt wird. Die Praktikabilität der Benutzerschnittstelle lässt sich beispielsweise erst nach der Lernphase des Systems, deren Dauer von der individuellen Nutzungsintensität jedes einzelnen Probanden abhängt, eindeutig bestimmen. Laut Gajos und Jameson in [Gajos u. Jameson, 2009] können neue Adaptionen z.B. auch einen gegenteiligen Effekt auf den Nutzen einer adaptiven Benutzerschnittstelle ausüben. Vor allem, wenn der Benutzer bereits über ein stabiles mentales Modell der Benutzerschnittstelle verfügt. Eine umfassende Evaluation sollte also mindestens so lange durchgeführt werden, bis der Benutzer eine bestimmte Expertise beim Umgang mit der Benutzerschnittstelle erlangt hat. Überdies ist eine generelle Aussage über den praktischen Nutzen oder die Attraktivität der kontextsensitiven Personalisierung i. Allg. nicht möglich, da eine Feldstudie in jedem Fall mit dem Test einer konkreten Realisierung einhergeht. Die Beurteilung jedes Probanden bezieht sich im Fall der grafischen Benutzeroberfläche und der Bedienung über die ZBE daher immer auf das bestimmte Look&Feel der konkreten Realisierung.

Im Rahmen dieser Forschungsarbeit wurde aufgrund der begrenzten Ressourcen sowohl bzgl. der prototypischen Fahrzeuge, als auch hinsichtlich der zur Verfügung stehenden Probanden eine Benutzerevaluation durchgeführt, die sich insbesondere auf die Auswertung der subjektiven Eindrücke der Probanden, welche direkt nach der Bedienung des Prototyps erfasst wurden, konzentriert. Der Zeitrahmen der Benutzerevaluation wurde dabei auf maximal eine Stunde pro Proband begrenzt. Die Bedienung der kontextsensitiven Benutzerschnittstelle durch die Probanden erfolgte nicht im Fahrzeug, sondern unter Laborbedingungen am HMI-Prototyp. Die unter diesen Voraussetzungen nur mit einer quantitativen Evaluation zu beantwortenden Forschungsfragen werden im nächsten Abschnitt ausführlich vorgestellt. Eine Beschreibung zur Durchführung dieser Studie, die Vorstellung der Ergebnisse und eine abschließende Auswertung folgen im Abschnitt 7.2.

7.1 Fragestellungen und Hypothesen

Die tatsächlichen Gewohnheiten, die sich während der tagtäglichen Fahrt einstellen, lassen sich im Labor nicht erfassen, da weder das echte Fahrgefühl noch der entsprechende Kontext, wie z.B. die sozialen Kontakte oder der mögliche Zeitdruck, hinreichend genau simuliert werden. Es besteht im Rahmen einer Laborstudie also nur die Möglichkeit, über problembezogene Fragebögen die subjektiven Einschätzungen bzgl. der Nutzung einer konkreten Realisierung zu erfassen und über deskriptive Verfahren auszuwerten. Die dabei zu untersuchenden Hypothesen und Fragestellungen müssen infolgedessen vor der Durchführung der Studie eindeutig formuliert werden. Aufgrund der Gegebenheiten beziehen sich die nun folgenden Hypothesen ausschließlich auf die subjektiven Eindrücke jedes einzelnen Probanden und dessen Erfahrung mit der hier vorgestellten prototypischen automotiven Benutzerschnittstelle.

Dieser Forschungsarbeit wurde allgemein die Hypothese zugrunde gelegt, dass potentielle Benutzer einer kontextsensitiv personalisierten, automotiven Benutzerschnittstelle offen gegenüberstehen und diese im Fall eines geringeren Bedienaufwandes der statischen Benutzerschnittstelle vorziehen. Ein Überprüfung dieser allgemeinen Hypothese erweist sich aufgrund der oben beschriebenen Voraussetzungen als schwierig und lässt sich demnach nur mit Hilfe einer quantitativen Langzeitevaluation durchführen. Im Rahmen der hier durchgeführten Benutzerevaluation wurde diese Hypothese daher auf die Annahme reduziert, dass potentielle Benutzer über den Wunsch verfügen, automotive Benutzerschnittstellen manuell anzupassen oder eine automotive Benutzerschnittstelle zu nutzen, die sich den Gewohnheiten des Benutzers automatisch anpasst. Diese Bedürfnisse können über entsprechende allgemeine Fragestellungen vor der Nutzung des Prototyps - und damit möglichst unvoreingenommen - erfragt werden. Über abschließende Fragen, die dem Probanden erst nach der Nutzung des Prototyps gestellt werden, können auch die Veränderungen dieser Bedürfnisse erfasst werden.

Motiviert wurde die kontextsensitive Personalisierung automotiver Benutzerschnittstellen in erster Linie durch die Annahme, dass der Bedienaufwand durch den geeigneten Einsatz kontextsensitiv personalisierter Elemente in der Benutzeroberfläche reduziert werden kann. In absoluten Zahlen der notwendigen Bedienschritte wurde diese Annahme bereits in Abschnitt 5.1 bzgl. der beiden Formen der kontextsensitiven Personalisierung validiert. Ob dieses Ergebnis durch den messbaren, realen Bedienaufwand während der Fahrt bestätigt werden kann, ist mit Hilfe eines Vergleichs der Arbeitsbelastungen zu prüfen, die bei der

Bedienung der kontextsensitiv personalisierten Bedienelemente und der statischen Benutzerschnittstelle entstehen. Die dazu zu validierende Hypothese (in diesem Fall als Alternativhypothese bezeichnet) nimmt an, dass die Arbeitsbelastung bei der Bedienung der beiden vorgestellten Formen der kontextsensitiven Personalisierung i. Allg. geringer ausfällt, als bei der Bedienung der statischen Benutzerschnittstelle. Die gegenteilige Nullhypothese geht dabei von keinem Unterschied der zu vergleichenden Arbeitsbelastungen aus[1].

Eine weitere Fragestellung betrifft das Maß, mit dem eine konkrete Realisierung die speziellen Anforderungen an ein adaptives System aus Sicht der Benutzers erfüllt. Zu den Anforderungen gehören z.B. eine hohe Vorhersagbarkeit, eine ansprechende Darstellung oder eine hohe Effizienz der Bedienung. Inwieweit eine konkrete Realisierung diesen Anforderungen gerecht wird, kann direkt nach der Nutzung des Prototyps über entsprechende Bewertungsskalen erfasst werden. Da zwei unterschiedliche Formen der kontextsensitiven Personalisierung vorgestellt wurden, gilt es, jede Form einzeln bzgl. der speziellen Anforderungen zu bewerten. Überdies wird in diesem Zusammenhang die Hypothese (Nullhypothese) aufgestellt, dass die beiden konkreten Realisierungen der kontextsensitiven Personalisierung die speziellen Anforderungen im gleichen Maße erfüllen.

Sollte die Arbeitsbelastung durch den Einsatz kontextsensitiv personalisierter Elemente in der Benutzerschnittstelle reduziert werden können, so stellt sich dennoch die Frage, ob der Benutzer den Nutzen in gleicher Weise wahrnimmt. Hier müssen also nicht die Arbeitsbelastungen verglichen, sondern die subjektive Qualitätswahrnehmung ermittelt werden. Die insbesondere als pragmatische Qualität [Hassenzahl, 2003] eines Produkts empfundene Qualitätswahrnehmung entspricht dabei dem subjektiven, konkreten Nutzen aus Sicht des Benutzers. Die pragmatischen Eigenschaften umfassen z.B. die erfahrene Unterstützung durch das Produkt oder die gefühlte Kontrolle über das Produkt. Beide Formen der kontextsensitiven Personalisierung sollten daher hinsichtlich ihrer pragmatischen Qualität untersucht und verglichen werden. Dies ist im Fall einer im Labor durchgeführten Benutzerevaluation jedoch nur sinnvoll, wenn der Proband während der Bedienung bestimmte Aufgabenstellungen zu erfüllen hat.

Die ebenfalls einer subjektiven Qualitätswahrnehmung entsprechende hedonische Qualität [Hassenzahl, 2003] einer konkreten Realisierung beschreibt hingegen die Emotionen, die ein Produkt beim Benutzer auslöst. Bei einer hohen hedonischen Qualität wird auch davon gesprochen, dass das Produkt eine hohe Bindung erzeugt, der Benutzer vom Produkt überrascht wird, sich der Benutzer mit dem Produkt identifiziert oder das Produkt sich zu repräsentativen Zwecken eignet. Da es sich bei der kontextsensitiven Personalisierung um ein Verfahren handelt, welches das Produkt an die Vorlieben des Benutzers ausrichtet, wird im Rahmen dieser Forschungsarbeit erwartet, dass in beiden Fällen eine hohe hedonische Qualität vorliegt. Diese hängt, mehr noch als die pragmatische Qualität, von der konkreten Umsetzung und Darstellung der Benutzeroberfläche ab. Zwar dienen die beiden konkreten Realisierungen der kontextsensitiven Personalisierung nur als anschauliche Beispiele und sind somit nicht zentraler Gegenstand dieser Forschungen, sie können aber dennoch einen Eindruck davon vermitteln, welche hedonische Qualität zu erwarten ist.

Die pragmatischen und hedonischen Qualitäten als auch die gesamte Attraktivität der beiden konkreten Realisierungen werden hier im Sinne von Nullhypothesen als äquivalent angenommen. Ein ungleiches Maß dieser Eigenschaften wird dahingegen in den Alternativhypothesen propagiert.

[1] Im nächsten Abschnitt werden die Nullhypothesen mit Hilfe von Zwei-Stichproben-T-Tests auf ihren Wahrheitsgrad untersucht. Liegt der Wahrheitsgrad unter einem vorher festgelegten Wert, so wird die Nullhypothese verworfen und die Alternativhypothese bestätigt.

7.2 Benutzerevaluation

Zur Beantwortung und Überprüfung der oben genannten Fragestellungen und Hypothesen ist es folglich notwendig, eine Benutzerevaluation durchzuführen, die es den Probanden ermöglicht, die zu untersuchende Benutzerschnittstelle in Echtzeit kennenzulernen und anschließend gezielt zu bewerten. Zur Beurteilung der oben genannten Eigenschaften kommt in der hier durchgeführten Benutzerevaluation ausschließlich das Erhebungsinstrument eines Fragebogens zum Einsatz. Jeder Fragebogen widmet sich jeweils der Erfassung bestimmter Aspekte des Prototyps und besteht aus einer thematisch zusammenhängenden Gruppe geschlossener Fragen. In den nachfolgenden Unterabschnitten werden die erstellten Fragebögen der Benutzerevaluation vorgestellt, der Ablauf der Benutzerevaluation beschrieben und die Ergebnisse ausgewertet.

7.2.1 Fragebögen

Zunächst einmal galt es, sich mit Hilfe allgemeiner Fragen einen Überblick darüber zu verschaffen, welchen Hintergrund jeder einzelne Proband vorzuweisen hat. Dazu gehören Fragen zur Altersgruppe, dem Geschlecht, dem höchsten Ausbildungsabschluss und der derzeitigen Tätigkeit. Hinzu kommen außerdem noch Fragen bzgl. der individuellen Mobilität und der Erfahrung mit automotiven Benutzerschnittstellen. Es wurde zusätzlich gefragt, ob es sich bei einem Probanden um einen Besitzer eines Fahrzeugs handelt, wie oft dieser ein Fahrzeug nutzt und welche Funktionen des In-Car-Entertainment-Systems regelmäßig verwendet werden. Alle Fragen sind mit vorformulierten Antworten versehen worden. Abhängig von der Fragestellung war eine Einfach- oder Mehrfachauswahl der Antworten möglich.

Um die Einstellung des Probanden gegenüber adaptiven automotiven Benutzerschnittstellen zu erfassen, wurde eine in [Hauser, 2004] von Hauser speziell zur Evaluation von adaptiven Benutzerschnittstellen entwickelte Fragestellung übernommen und an die spezifischen Gegebenheiten im automotiven Bereich angepasst. Diese Fragestellung wird in Abbildung 7.1 - als beispielhafter Auszug aus dem Fragebogen zur Einführung - dargestellt. Da eine automatische Anpassung der Benutzerschnittstelle im Sinne des hier vorgestellten Verfahrens eine manuelle Konfiguration der Benutzerschnittstelle nicht ausschließt, ist in diesem Fall eine Mehrfachauswahl möglich. Der gesamte Fragebogen mit dem Titel „Einführung“ ist dem Anhang C.1 zu entnehmen.

Die Untersuchung der jeweiligen Arbeitsbelastungen fand nicht über die Erfassung physikalisch messbarer Werte wie z.B. den Pupillenbewegungen oder dem Hautwiderstand, sondern durch die Auswertung eines Fragebogens statt, der die subjektive Arbeitsbelastung anhand sechs unterschiedlicher Kriterien misst. Dieser - am Ames Research Center von Hart in [Hart u. Stavenland, 1988] vorgestellte NASA-TLX (Task Load Index) Fragebogen - erfasst die Arbeitsbelastung anhand der subjektiv wahrgenommenen geistigen und physikalischen Anforderung, dem gefühlten Zeitdruck, dem eingeschätzten Erfolg, dem empfundenen Aufwand und dem verspürten Grad an Frustration. Es handelt sich hierbei um ein Verfahren, welches in unterschiedlichen Anwendungsgebieten zum Einsatz kommen kann. Jedes zu untersuchende Kriterium wird individuell über eine Bewertungsskala erfasst, die es dem Probanden ermöglicht, das Maß der betreffenden Eigenschaft auf 20 Niveaustufen abzubilden. In Abbildung 7.2 wird beispielhaft die Bewertungsskala bzgl. der physikali-

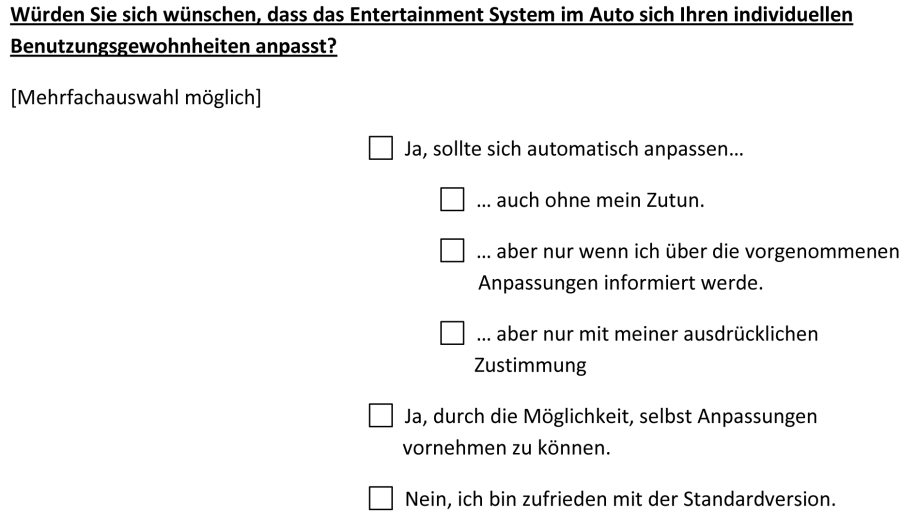


Abb. 7.1 Frage bzgl. der allgemeinen Einstellung gegenüber dem Einsatz adaptiver Benutzerschnittstellen im automotiven Bereich.

schen Anforderung dargestellt. Werden alle Kriterien unmittelbar nach der Ausführung einer Aufgabe über den Fragebogen erfasst, so kann über den gewichteten Durchschnitt aller Bewertungen ein Score der Arbeitsbelastung hinsichtlich dieser Aufgabe errechnet werden. Dadurch lassen sich die Arbeitsbelastungen unterschiedlicher Aufgaben miteinander vergleichen. Der in dieser Benutzerevaluation in der deutschen Übersetzung verwendete NASA-TLX Fragebogen befindet sich im Anhang C.2.

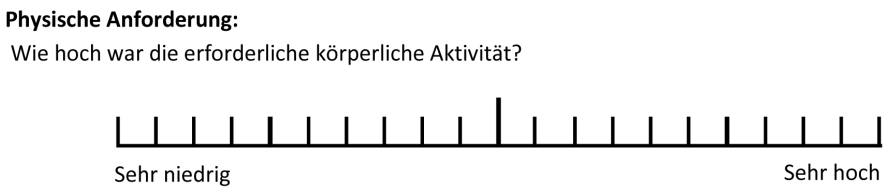


Abb. 7.2 Bewertungsskala des NASA-TLX Fragebogens bzgl. der physikalischen Anforderung, der sich der Benutzer bei der Bewältigung einer Aufgabe ausgesetzt sieht.

Zur Bewertung der Anforderungen galt es, im Gegensatz zur Untersuchung der Arbeitsbelastung, einen gesonderten Fragebogen anzufertigen, der sich speziell den Anforderungen adaptiver Benutzerschnittstellen widmet. Hierzu wurde (ebenso wie im Fall der Erfassung der allgemeinen Einstellung gegenüber adaptiven Benutzerschnittstellen im automotiven Bereich) der Fragebogen bzgl. der Anforderungen adaptiver Systeme aus [Hauser, 2004] übernommen und um die Frage zur Vorhersagbarkeit gekürzt. Letztere Frage wird in einem separaten Fragebogen aufgeführt. Die Anforderungen umfassen den Grad an Unterstützung durch das System, die Eignung der Darstellung, die Einhaltung der Privatsphäre, die Zuverlässigkeit, die benutzerspezifische Ausrichtung und die Beherrschbarkeit des Systems. Jeder Anforderung wurde dabei eine eigene 5-stufige Bewertungsskala gewidmet. Die Bewertungsskala zur Privatsphäre wird beispielhaft in Abbildung 7.3 aufgeführt.

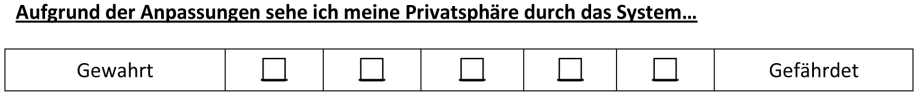


Abb. 7.3 Frage und Bewertungsskala zur Beurteilung des gefühlten Eingriffs in die Privatsphäre.

Die subjektiv wahrgenommene Effizienz der Bedienung wurde aufgrund der zentralen Bedeutung als Hauptmotiv der kontextsensitiven Personalisierung über zwei separat aufgeführte Fragen festgestellt. Da es galt, beide Formen der kontextsensitiven Personalisierung direkt hinsichtlich ihres Bedienaufwands zu vergleichen, wurden die Fragen zur Effizienz im Fragebogen nebeneinander aufgeführt. Hier kam ebenfalls eine Bewertungsskala mit fünf Stufen zum Einsatz. In Abbildung 7.4 werden beide Fragen und die zugehörigen Bewertungsskalen dargestellt.

Mit der personalisierten Schnellstartleiste kann ich die Aufgaben effizient erledigen	Völlig Falsch	☐	☐	☐	☐	☐	Völlig richtig
Mit den personalisierten Automatisierungen kann ich die Aufgaben effizient erledigen	Völlig Falsch	☐	☐	☐	☐	☐	Völlig richtig

Abb. 7.4 Fragen und Bewertungsskala zur Effizienz der Bedienung beider Formen der kontextsensitiven Personalisierung.

Die Messung der Qualitätsmerkmale Gebrauchstauglichkeit und subjektiv wahrgenommene Stimulation und Identifikation fand über den AttrakDiff2-Fragebogen statt. Dieser Fragebogen wurde von Hassenzahl [Hassenzahl u. a., 2003] speziell dazu entwickelt, die pragmatischen und hedonischen Qualitäten einer Benutzerschnittstelle zu beurteilen. Der AttrakDiff2-Fragebogen setzt sich aus einer Liste von 28 Wortpaaren zusammen, die jeweils aus zwei gegensätzlichen Beschreibungen des Zustandes einer gemeinsamen Eigenschaft bestehen. Dem Proband wird über eine 7-stufige Bewertungsskala ermöglicht, die sich durch die Wortpaare ergebende Eigenschaft individuell zu beurteilen. Der Fragebogen gliedert sich in vier Gruppen von Wortpaaren, die sich jeweils auf die speziellen Eigenschaften der pragmatischen Qualität, der hedonischen Qualitäten bzgl. der Stimulation und der Identifikation und der Attraktivität i. Allg. beziehen. Drei Wortpaare, die jeweils eine Eigenschaft der pragmatischen Qualität (einfach, kompliziert), der hedonischen Qualität der Identifikation (fachmännisch, laienhaft) und der Attraktivität (hässlich, schön) beschreiben, werden beispielhaft zusammen mit den entsprechenden Bewertungsskalen in der Abbildung 7.5 aufgelistet. Der gesamte Fragebogen mit dem Titel „Abschluss“, der die Fragen bzgl. der Anforderungen, der Effizienz, der pragmatischen Qualität, der hedonischen Qualitäten, der Attraktivität und der allgemeinen Einstellung gegenüber adaptiven automotiven Benutzerschnittstellen umfasst, kann im Anhang C.3 eingesehen werden.

einfach	☐	☐	☐	☐	☐	☐	☐	kompliziert
fachmännisch	☐	☐	☐	☐	☐	☐	☐	laienhaft
hässlich	☐	☐	☐	☐	☐	☐	☐	schön

Abb. 7.5 Drei beispielhafte Wortpaare bzgl. der pragmatischen Qualität (einfach, kompliziert), der hedonischen Qualität der Identifikation (fachmännisch, laienhaft) und der Attraktivität (hässlich, schön).

7.2.2 Aufbau und Durchführung

Die Benutzerevaluation wurde am *Daimler Center for Automotive Information Technology Innovations* an der Technischen Universität Berlin mit insgesamt 31 Probanden durchgeführt. Die Teilnahme an der Studie erfolgte auf freiwilliger Basis und ohne eine Entlohnung. Die Probanden wurden dazu nicht öffentlich beworben, sondern gezielt auf die Benutzerevaluation aufmerksam gemacht. Aufgrund dieser Umstände handelt es sich hierbei um keine repräsentativ angelegte Benutzerevaluation. Unter den Probanden befanden sich insgesamt neun Frauen und 22 Männer. 23 Probanden wiesen ein abgeschlossenes Hochschulstudium, zwei eine abgeschlossene Ausbildung und sechs einen Schulabschluss auf. 12 der Probanden verfolgten zum Zeitpunkt der Benutzerevaluation ein Studium, während 18 Probanden einer Anstellung nachgingen. 19 der Probanden wurden der Altersgruppe 20 bis 29 Jahre, zehn der Altersgruppe 30 bis 39 Jahre und zwei der Altersgruppe ab 50 Jahren zugeordnet.



Abb. 7.6 Prüfstand aus Sicht des Probanden. Oben links: Ansicht des Fahrsimulators; Oben rechts: Benutzeroberflächen des HMI-Prototyps; Unten links: Lenkrad; Unten rechts: Zentrale Bedieneinheit mit Armlehne. Quelle der 3D-Ansicht der Welt: Google Inc. (http://www.google.com/)

Die Durchführung der Befragungen erfolgte nicht an einem Termin mit allen Probanden gemeinsam, sondern über mehrere Tage hinweg, individuell, mit einem Probanden pro Termin. Die Dauer eines Termins betrug dabei 40 bis 60 Minuten. Der Proband befand sich während des gesamten Zeitraums an dem mit der prototypischen Benutzerschnittstelle bestückten Prüfstand, wenngleich der Prototyp nicht in jeder Phase des Durchgangs zum Einsatz kam. Der Prüfstand bestand aus zwei Bildschirmen, auf denen jeweils die Umgebung des Fahrsimulators und die Benutzeroberfläche des HMI-Prototyps visualisiert wur-

den, aus dem Lenkrad und den Pedalen zur Steuerung des Fahrsimulators und einer zentralen Bedieneinheit inklusive der dazu passenden Armlehne. Die akustische Untermalung mit den entsprechenden Fahrgeräuschen oder Signalen des simulierten Autoradios fand über die Lautsprecher eines der Bildschirme statt. In Abbildung 7.6 wird der Prüfstand während der Benutzung des Prototyps aus Sicht des Probanden dargestellt.

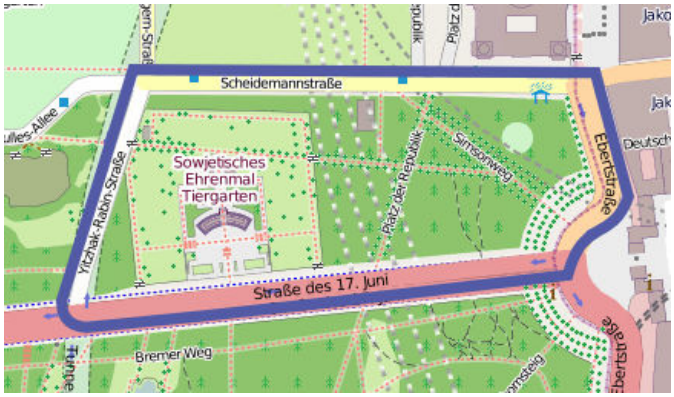


Abb. 7.7 1,4 Kilometer lange Teststrecke durch das Zentrum von Berlin. Quelle des Kartenmaterials: OpenStreetMap-Projekt (http://www.openstreetmap.de/)

Ein Termin, bzw. Durchgang, setzte sich aus drei aufeinander folgenden Phasen zusammen: Einführung, Ausführung und Abschluss. Zu Beginn der Einführungsphase, nachdem der Versuchsleiter den Probanden über die Formalitäten der Evaluation und den anonymisierten Charakter der Befragung informierte, wurde der Fragebogen mit dem Titel „Einführung“ aus Anhang C.1 an den Probanden ausgehändigt. Der Proband beantwortete die Frage zur allgemeinen Einstellung gegenüber adaptiven automotiven Benutzerschnittstellen daher ohne jegliche Kenntnisse bzgl. einer konkreten Realisierung. Nach der Beantwortung aller Fragen wurde der Proband vom Versuchsleiter in die Bedienung des Fahrsimulators eingewiesen. Dazu fuhr der Proband eine vorgegebene Teststrecke ab, die mit insgesamt vier Lichtsignalanlagen (für jede Kreuzung eine) bestückt war. Die Teststrecke wird in Abbildung 7.7 dargestellt. Andere Verkehrsteilnehmer wurden nicht simuliert. Die Fahrt erfolgte ohne eine zeitliche Beschränkung. Als nächstes machte sich der Proband mit der spezifischen Benutzeroberfläche und Bedienung des HMI-Prototyps vertraut. Neben der Einführung in das generelle Bedienkonzept zur Ausführung einer Applikation und dem Wechsel zwischen Applikationen wurde insbesondere die Bedienung der Radioapplikation, der kontextsensitiv personalisierten Schnellstartbuttons, der kontextsensitiv personalisierten Automatisierung und der Feedbackansicht trainiert. Der Versuchsleiter führte die Bedienung zunächst jeweils am Prototypen vor und überließ es dann dem Probanden, eigenhändig in der Menüstruktur zu navigieren und Einstellungen vorzunehmen. Die gesamte Dauer der Einführungsphase betrug 10 bis 20 Minuten.

In der darauf folgenden Ausführungsphase ging es für den Probanden darum, auf der Teststrecke (während der simulierten Fahrt) diverse Radiosenderwechsel auf drei unterschiedliche Arten durchzuführen und jeden dieser Wechsel bzgl. der subjektiv wahrgenommenen Arbeitsbelastung zu bewerten. Dabei sollte der Proband nicht eigenhändig entscheiden, wann ein Radiosenderwechsel stattzufinden hat, sondern wurde vom Versuchsleiter immer an geeigneter Stelle auf der Teststrecke dazu aufgerufen, diesen auszuführen.

Das System konnte somit während der Ausführungsphase örtliche Routinen in Form ortsabhängiger Radiosenderwechsel erkennen, die es den Probanden an den entsprechenden Stellen ermöglichte, die kontextsensitiv personalisierten Schnellstartbuttons auszuführen und die kontextsensitiv personalisierte Automatisierung zu erleben. Die örtlichen Routinen und die dazu passenden kontextsensitiven Personalisierungen wurden also nicht simuliert, sondern während der Ausführung von dem HMI-Prototyp erkannt, und in entsprechende kontextsensitive Personalisierungen umgesetzt. Dies erhöhte die Flexibilität für den Probanden, da zum einen die ortsabhängigen Radiosenderwechsel auch genau an der Stelle erkannt wurden, an denen der Proband diese auch tatsächlich ausführte[2], und zum anderen es dem Probanden überlassen wurde, einen bevorzugten Radiosender zu wählen.

Aus Sicht des Probanden begann die Ausführungsphase also zunächst mit einer kurzen Einweisung in den Ablauf dieser Phase der Benutzerevaluation. Die Probanden wurden darüber informiert, dass sie die Teststrecke dreimal hintereinander zu befahren haben und an den vom Versuchsleiter vorgegebenen Positionen jeweils einen manuellen Radiosenderwechsel, einen Radiosenderwechsel über die kontextsensitiv personalisierten Schnellstartbuttons auszuführen, einen automatischen Radiosenderwechsel zuzulassen und abzubrechen haben. Vor dem Beginn jeder Fahrt wurde der Proband aufgefordert, das Radio auf einen beliebigen Radiosender X einzustellen und anschließend das Navigationssystem zu öffnen. Diese Interaktionen dienten dazu, die Benutzerschnittstelle vor jeder Fahrt in einen definierten Zustand zu versetzen, der einen direkten Vergleich der bei allen drei Fahrten auftretenden Arbeitsbelastungen ermöglichte.

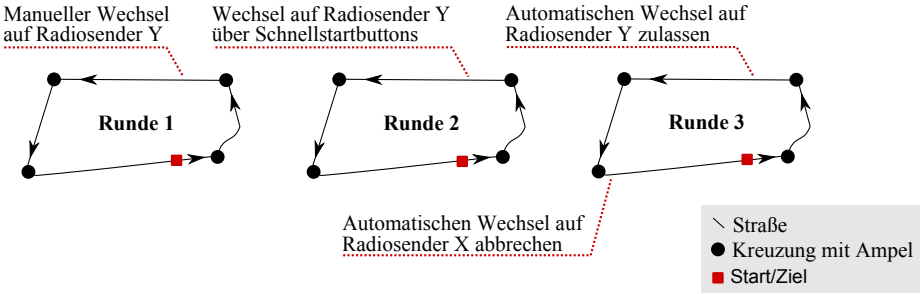


Abb. 7.8 Schematischer Ablauf aller drei Testfahrten während der Ausführungsphase. Die manuellen Wechsel auf den Radiosender X am Start jeder Fahrt und die Aktivierung der Automatisierungen am Start vor der letzten Fahrt werden nicht abgebildet, da diese Interaktionen bei der Erfassung der Arbeitsbelastung nicht berücksichtigt wurden.

Die erste Fahrt startete auf der *Straße des 17. Juni* vor dem *Brandenburger Tor* und verlief dann auf der *Ebertstraße* in Richtung Norden. Nachdem der Proband zweimal links abbog (und folglich zwei Lichtsignalanlagen passierte) und sich auf der *Scheidemannstraße* in Richtung Westen auf der Höhe des *Simsonwegs* befand, bat der Versuchsleiter den Probanden, einen manuellen Radiosenderwechsel auf einen beliebigen Radiosender Y auszuführen. Danach bog der Proband ebenfalls zweimal links ab und gelangte nach der *Yitzhak-Rabin-Straße* zurück auf die *Straße des 17. Juni* in Richtung *Brandenburger Tor*.

[2] Örtlich nah beieinander ausgeführte Radiosenderwechsel führten zu einer örtlichen Routine, deren Ort durch ein Straßensegment repräsentiert wurde. Ein Straßensegment erstreckte sich dabei von einer Straßenkreuzung zur nächsten Straßenkreuzung. Die dazugehörige kontextsensitive Personalisierung erfolgte daher immer beim Betreten des betreffenden Straßensegments.

Kurz vor dem Erreichen des *Brandenburger Tors* wurde die Fahrt vom Versuchsleiter für beendet erklärt. Die vom Probanden subjektiv wahrgenommene Arbeitsbelastung wurde dann unmittelbar danach über den Fragebogen mit dem Titel „Laufend“ aus Anhang C.2 erhoben. Die zweite Fahrt verlief bis auf die Art des Radiosenderwechsels identisch. In diesem Fall wurde der Radiosenderwechsel nicht manuell, sondern über den während der Fahrt auf der *Scheidemannstraße* erscheinenden kontextsensitiv personalisierten Schnellstartbutton ausgeführt. Ein neuer und unausgefüllter Fragebogen mit dem Titel „Laufend“ diente auch hier zur Erfassung der Arbeitsbelastung. Vor der dritten und letzten Fahrt galt es, die vom System empfohlenen kontextsensitiv personalisierten Automatisierungen über die Feedbackansicht einzusehen und beide (Radiosenderwechsel Y auf der *Scheidemannstraße* und Radiosenderwechsel X auf der *Straße des 17. Juni*) zu aktivieren. Der Proband wurde dabei angewiesen, den ersten, bei der kommenden Fahrt auf der *Scheidemannstraße* zu erwartenden automatischen Radiosenderwechsel zuzulassen und den nach dem letzten Abbiegen auf die *Straße des 17. Juni* zu erwartenden automatischen Radiosenderwechsel über die ZBE abzubrechen. Nachdem der Proband die dritte Fahrt erfolgreich absolviert hatte, kam auch hier ein neuer und unausgefüllter Fragebogen mit dem Titel „Laufend“ zur Erfassung der Arbeitsbelastung zum Einsatz. In Abbildung 7.8 wird der Ablauf aller Fahrten schematisch dargestellt. Die Dauer der Ausführungsphase betrug insgesamt 20 bis 25 Minuten.

Die Abschlussphase begann zunächst damit, dass jeder Proband alle Kriterien des NASA-TLX-Fragebogens mit einer persönlichen Gewichtung zu versehen hatte. Dem Probanden wurden dazu nacheinander 15 Karten ausgehändigt, auf denen jeweils zwei unterschiedliche Kriterien verzeichnet waren. Der Proband sollte nun aus dem direkten Vergleich heraus auf jeder Karte das Kriterium markieren, welches seiner Meinung nach einen größeren Einfluss auf die gesamte Arbeitsbelastung ausübte. Dies diente während der Auswertung dazu, die individuellen Differenzen bzgl. der Definition der Arbeitsbelastung zu berücksichtigen. Abgeschlossen wurde der Durchgang mit der Aushändigung des Fragebogens mit dem Titel „Abschluss“ aus Anhang C.3. Dieser enthielt ein weiteres Mal die Frage zur allgemeinen Einstellung gegenüber adaptiven Benutzerschnittstellen im automotiven Bereich, um einen Vorher-Nachher-Vergleich dieser Präferenz zu ermöglichen. Die Dauer der Abschlussphase betrug insgesamt 10 bis 15 Minuten.

7.2.3 Ergebnisse und Auswertung

Unter den Probanden befanden sich insgesamt 12 Autobesitzer und 19 Autonutzer. Somit wurde sichergestellt, dass sich die Ergebnisse dieser Benutzerevaluation nur auf Personen beziehen, die Erfahrungen im Umgang mit automotiven Benutzerschnittstellen aufweisen. Von den 31 Probanden nutzen acht ein Auto wöchentlich, während sieben es mehrmals monatlich und 16 es eher selten nutzen. Das Radio wird von 25 Probanden regelmäßig verwendet und ist damit die am häufigsten genutzte Funktion eines In-Car-Infotainment-Systems. Darauf folgt das Abspielen von digitalen Medien wie z.B. CDs oder MP3-Dateien mit 20 Probanden und das Konfigurieren typischer Komfortfunktionen wie z.B. der Klimaanlage oder der Sitzheizung mit 17 Probanden. 14 Probanden bedienen regelmäßig ein Navigationssystem, während fünf Probanden regelmäßig das Telefon verwenden oder nur ein Proband im Fahrzeug das Internet nutzt. Oftmals merkten die Probanden bzgl. der zuletzt genannten Funktion an, sollte ein Internetzugang durch das In-Car-Infotainment-System

zur Verfügung gestellt werden, diese Funktionalität einer regelmäßigen Nutzung ausgesetzt wäre. Die Ergebnisse dieser Fragestellung müssen also immer unter dem Gesichtspunkt ausgewertet werden, dass zum Zeitpunkt der Evaluation nicht jeder Proband im Besitz eines In-Car-Infotainment-Systems war, welches über alle aufgezählten Funktionalitäten verfügte.

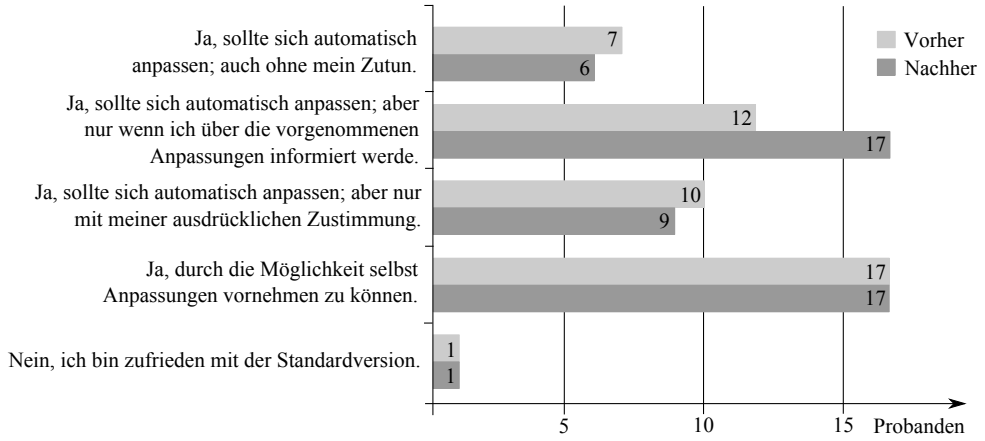


Abb. 7.9 Ergebnisse der Frage: *Würden Sie sich wünschen, dass das Entertainment-System im Auto sich ihren individuellen Benutzungsgewohnheiten anpasst?* Es war eine Mehrfachauswahl möglich.

Hinsichtlich der allgemeinen Einstellung bzgl. adaptiver Benutzerschnittstellen im automotiven Bereich ergeben sich die in Abbildung 7.9 als Diagramm visualisierten Ergebnisse. Das Ergebnis bzgl. der Antwort „Ja, sollte sich automatisch anpassen...“ wurde hier nicht mit aufgezählt, da die Wahl einer der drei Optionen bereits eine Bejahung dieser Frage impliziert. Im Diagramm werden die Ergebnisse aus dem Fragebogen mit dem Titel „Einführung“ denen aus dem Fragebogen mit dem Titel „Abschluss“ direkt gegenübergestellt. Mehr als die Hälfte der Probanden wünscht sich demnach die Möglichkeit, die automotive Benutzerschnittstelle im Sinne einer Customization an die individuellen Bedürfnisse anzupassen. Da die manuelle Anpassung der Benutzerschnittstelle nicht Gegenstand der Untersuchungen war, unterscheiden sich die Ergebnisse aus der Fragestellung vor und nach der Benutzung des Prototyps nicht. 12 Probanden wünschten sich vor der Benutzung des Prototyps, dass sich die Benutzerschnittstelle automatisch anpasst, wenn sie über die Anpassungen informiert werden. Nach der Benutzung des Prototyps erhöhte sich diese Anzahl deutlich, da die Probanden entweder durch die konkrete Realisierung zu den Risiken vollautomatischer Anpassungen sensibilisiert wurden und/oder die beispielhafte Feedbackansicht einen hohen Zuspruch erfuhr.

Insgesamt sprachen sich 21 Probanden vor und 25 Probanden nach der Benutzung des Prototyps für eine automotive Benutzerschnittstelle aus, die sich entweder mit oder ohne eine Zustimmung des Benutzers automatisch anpasst. Diese beiden Werte deuten darauf hin, dass personalisierte Bedienelemente im Kreis der Probanden mehrheitlich erwünscht sind. Die Hypothese aus Abschnitt 7.1, wonach sich Benutzer eine automotive Benutzerschnittstelle wünschen, die sich manuell anpassen lässt und/oder sich automatisch anpasst, wird in Bezug auf diese Gruppe von Probanden bestätigt, da sich 30 der 31 Probanden für eine oder beide Funktionalitäten aussprachen.

Da die Probanden nur solche Interaktionen bei der Beurteilung der Arbeitsbelastung bewerten sollten, die direkt zu einem Radiosenderwechsel oder zu einem Abbruch eines au-

tomatisierten Radiosenderwechsels führten, wird die Arbeitsbelastung, die durch die Nutzung der Feedbackansicht entstand, nicht berücksichtigt. Die für jede Aufgabe bzw. Fahrt berechnete Arbeitsbelastung kann Werte zwischen 0 und 100 annehmen. Eine Möglichkeit der direkten Interpretation eines einzelnen Wertes ist nicht vorgesehen. Also kann bei einem Ergebnis von z.B. 10 weder von einer niedrigen Arbeitsbelastung, noch bei einem Ergebnis von z.B. 95 von einer hohen Arbeitsbelastung gesprochen werden. Es ist lediglich die Aussage zulässig, dass eine Arbeitsbelastung von z.B. 10 niedriger ausfällt, als eine Arbeitsbelastung von z.B. 80. Die errechneten Arbeitsbelastungen dienen also ausschließlich zu Vergleichszwecken. Die Ergebnisse der Arbeitsbelastungen bzgl. aller drei Formen des Radiosenderwechsels werden in Abbildung 7.10 in Form eines Diagramms dargestellt.

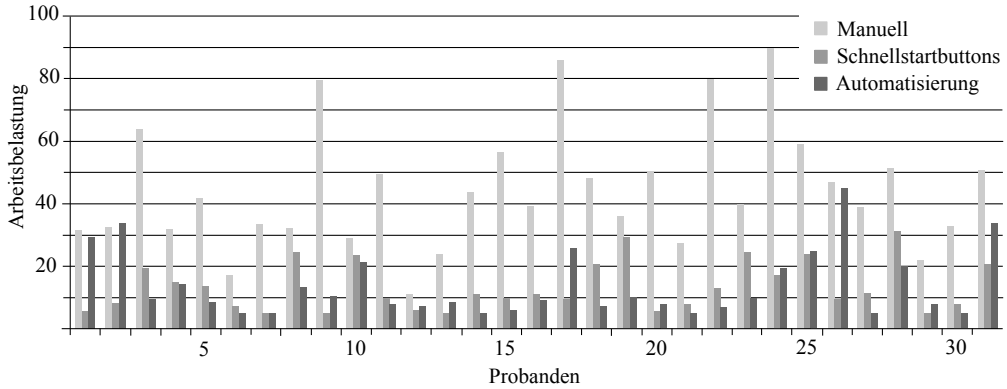


Abb. 7.10 Arbeitsbelastungen aller Probanden während des manuellen Radiosenderwechsels, des Radiosenderwechsels über die Schnellstartbuttons und des automatisierten Radiosenderwechsels.

Die Arbeitsbelastung während eines manuellen Radiosenderwechsels liegt im Mittel $\bar{x}$ bei ca. 44,49. Die Varianz σ^2 beträgt dabei ca. 385,77. Der Mittelwert liegt damit über der mittleren Arbeitsbelastung bei der Nutzung der Schnellstartbuttons $\bar{y}$ von ca. 13,48 ($\sigma^2 \approx 62,30$). Bei der Automatisierung fällt die mittlere Arbeitsbelastung $\bar{z}$ mit einem Wert von ca. 13,81 ($\sigma^2 \approx 110,73$) ähnlich niedrig aus. Die statistische Signifikanz der Differenzen $\Delta_1 = \bar{x} - \bar{y}$ und $\Delta_2 = \bar{x} - \bar{z}$ lässt sich durch entsprechende Hypothesentests für unabhängige Stichproben zeigen. Die Irrtumswahrscheinlichkeit, bzw. das Signifikanzniveau liegt hierbei und in allen nachfolgenden Betrachtungen bei $\alpha = 0.05$. Die Prüfgrößen der beiden zugehörigen F-Tests $F_{xy} \approx 6,19$ und $F_{xz} \approx 3,48$ zeigen dabei zunächst, dass sich die jeweils zu vergleichenden Varianzen der Stichproben wesentlich unterscheiden. Die Überprüfung der Unterschiedlichkeit der Varianzen zweier Stichproben ist in allen folgenden Betrachtungen notwendig, um über die anzuwendende Variante des Student'schen T-Tests zu entscheiden. Wird nun von den beiden Nullhypothesen $H_{xy} : \mu_x = \mu_y$ und $H_{xz} : \mu_x = \mu_z$ ausgegangen, die die Gleichheit der zu erwartenden Mittelwerte aller Stichproben postulieren, so können über zwei einseitige Student'sche T-Tests für den gleichen Stichprobenumfang und laut F-Tests unterschiedlichen Varianzen die angenommenen Nullhypothesen mit den zugehörigen p-Werten $p_{xy} \approx 2,69187 * 10^{-10}$ und $p_{xz} \approx 4,68002 * 10^{-10}$ aufgrund der niedrigen Wahrscheinlichkeit von $p_{xy} < 0.05$ und $p_{xz} < 0.05$ verworfen werden. Die Arbeitsbelastung bei der manuellen Ausführung des Radiosenderwechsels fällt somit signifikant höher aus, als bei einem Radiosenderwechsel über die hier vorgestellten kontextsensitiv personalisierten Bedienelemente. Bei einem Vergleich der Mittelwerte $\bar{y}$ und $\bar{z}$ mit einer laut F-Test ($F_{yz} \approx 1,77$) gleichen Varianz zeigt sich mit Hilfe eines zweisei-

tigen Student'schen T-Tests für den gleichen Stichprobenumfang und bei gleicher Varianz mit $p_{yz} \approx 0.882572$ kein signifikanter Unterschied der Arbeitsbelastungen. Die Nutzung beider kontextsensitiv personalisierten Bedienelemente führt demnach mit einer Irrtumswahrscheinlichkeit von $\alpha = 0.05$ zu einer gleichen Arbeitsbelastung.

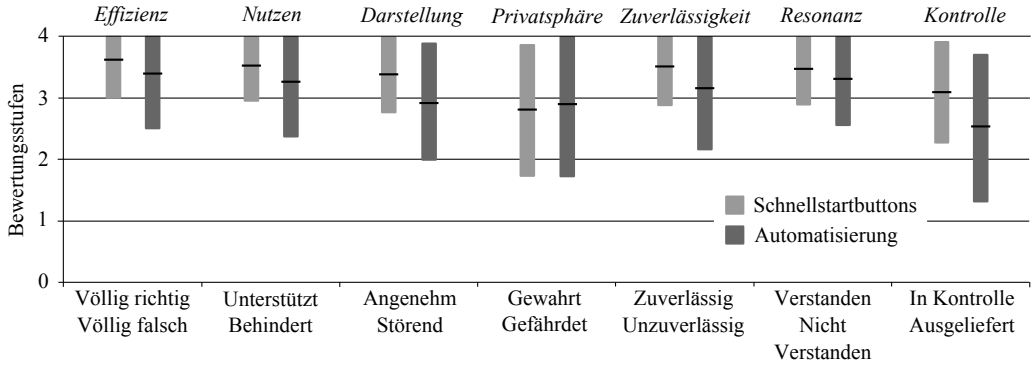


Abb. 7.11 Bewertungen bzgl. der Effizienz und den Anforderungen in Form von Mittelwerten und Konfidenzintervallen.

Die Messung der subjektiv wahrgenommenen Effizienz und der Beurteilungen hinsichtlich der Anforderungen erfolgte, wie in Unterabschnitt 7.2.1 beschrieben, über 5-stufige Bewertungsskalen, deren Extreme mit jeweils gegensätzlichen Antworten versehen wurden. Das in Abbildung 7.11 dargestellte Diagramm enthält die Auswertung der Bewertungen in Form von Mittelwerten und Konfidenzintervallen (bei $\alpha = 0.05$). Die fünf Bewertungsstufen werden in dem Diagramm durch die Zahlen Null bis Vier repräsentiert. Das untere Wort eines an der X-Achse befindlichen Wortpaares entspricht jeweils der niedrigeren Bewertung, während das obere Wort der höchsten Stufe zugeordnet wird. Eine Abbildung der Ergebnisse zu den im Anhang C.3 aufgeführten Fragen ist über den Vergleich der Wortpaare möglich.

Die Mittelwerte der Effizienz von ca. 3,61 ($\sigma^2 \approx 0,37$) für die Schnellstartbuttons und ca. 3,38 ($\sigma^2 \approx 0,77$) für die Automatisierung liegen im Bereich der positiven Wahrnehmung. Auch die damit zusammenhängenden Mittelwerte für den Nutzen von ca. 3,51 ($\sigma^2 \approx 0,32$) für die Schnellstartbuttons und ca. 3,25 ($\sigma^2 \approx 0,79$) für die Automatisierung zeigen, dass ein positiver Nutzen aus der Bedienung der kontextsensitiv personalisierten Bedienelemente gezogen werden kann. Über entsprechende zweiseitige T-Tests für unterschiedliche Varianzen, wie sie bereits oben Verwendung fanden, lassen sich mit den p-Werten von ca. 0,24 ($F \approx 0,48$) für die Effizienz und ca. 0,10 ($F \approx 0,40$) für den Nutzen keine signifikanten Unterschiede bei den Mittelwerten erkennen. Dies stellt sich jedoch bei den Bewertungen bzgl. der Darstellung anders dar. Hier kann durch einen zweiseitigen T-Test für unterschiedliche Varianzen und den daraus errechneten p-Wert von ca. 0,03 ($F \approx 0,40$) belegt werden, dass sich die Mittelwerte von ca. 3,38 ($\sigma^2 \approx 0,37$) für die Schnellstartbuttons und ca. 2,93 ($\sigma^2 \approx 0,92$) für die Automatisierung signifikant unterscheiden. Während die Darstellung der Schnellstartbuttons in einem separaten Bereich der Benutzerschnittstelle in den Augen der Probanden positiv wahrgenommen wurde, bewerteten die Probanden das Konzept einer Dialogbox zur Signalisierung einer anstehenden Automatisierung weniger positiv bis neutral. Die Probanden schlugen im Laufe der Untersuchungen oftmals vor, eine akustische Signalisierung möglicherweise zu bevorzugen.

Die Antworten bzgl. der Privatsphäre mit den Mittelwerten von ca. 2,8 ($\sigma^2 \approx 1,16$) für die Schnellstartbuttons und ca. 2,90 ($\sigma^2 \approx 1,42$) für die Automatisierung weisen darauf hin, dass sich das Speichern der individuellen Präferenzen als ein besonderer Kritikpunkt herausstellt. Die laut zweiseitigem T-Test für gleiche Varianzen signifikant nicht unterscheidbaren Mittelwerte (p-Wert von ca. 0,73 bei $F \approx 0,81$) unterstreichen dabei das mit der Verwendung von adaptiven Benutzerschnittstellen i. Allg. einhergehende Misstrauen gegenüber der maschinellen Erfassung persönlicher Daten. Da die Feedbackansicht, die jede im System gespeicherte Präferenz explizit macht, bei diesen Bewertungen nicht berücksichtigt wurde, ist anzunehmen, dass sich der Mittelwert bzgl. dieses Kriteriums für die Automatisierung unter Berücksichtigung der Feedbackansicht tendenziell aufgrund der Transparenz nach oben verschiebt.

Die Zuverlässigkeit und die Resonanz durch das System werden ähnlich positiv wie die Effizienz und der Nutzen wahrgenommen. Die Antworten zur Resonanz mit den Mittelwerten von ca. 3,41 ($\sigma^2 \approx 0,38$) für die Schnellstartbuttons und ca. 3,29 ($\sigma^2 \approx 0,54$) für die Automatisierung und den Antworten zur Zuverlässigkeit mit den Mittelwerten von ca. 3,51 ($\sigma^2 \approx 0,39$) für die Schnellstartbuttons und ca. 3,16 ($\sigma^2 \approx 1,00$) für die Automatisierung unterscheiden sich jeweils nicht signifikant (p-Werte von ca. 0,10 ($F \approx$0,38) für die Zuverlässigkeit und ca. 0,45 ($F \approx 0,70$) für die Resonanz). Die sich hingegen signifikant unterscheidenden Mittelwerte (p-Wert von ca. 0,03 und F-Wert von ca. 0,47) bzgl. der Beherrschbarkeit von ca. 3,09 ($\sigma^2 \approx 0,69$) für die Schnellstartbuttons und ca. 2,51 ($\sigma^2 \approx 1,45$) für die Automatisierung zeigen jedoch, dass die auf der Dialogbox basierende Signalisierung mit der Option des Abbrechens nicht immer den gewünschten Freiraum bei der Bedienung gewährleistet. Die Probanden erörterten diesbezüglich oftmals den potentiellen Umstand einer unübersichtlichen Verkehrssituation, in der die Benutzung der ZBE zum Abbrechen der Automatisierung als sicherheitskritisch eingestuft werden kann.

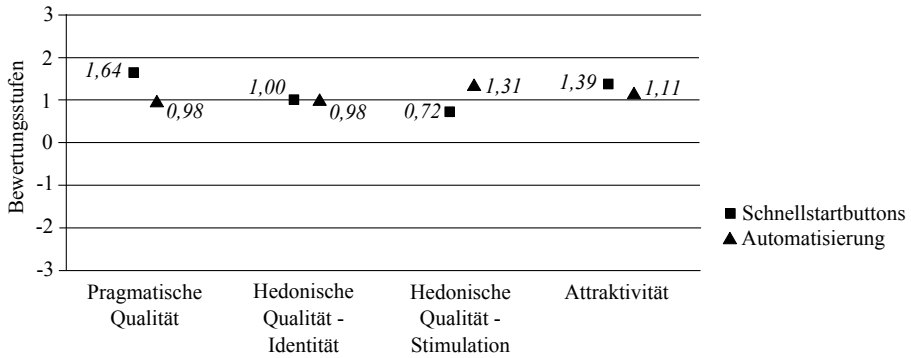


Abb. 7.12 Mittelwerte der pragmatischen und hedonischen Qualitäten (Identifikation und Stimulation) und der Attraktivität.

Die sieben Bewertungsstufen des AttrakDiff2-Fragebogens wurden analog zu den fünf Bewertungsstufen der Effizienz und den Anforderungen auf die ganzen Zahlen -3 bis 3 abgebildet. Ein hoher Wert steht hierbei stellvertretend für ein hohes Maß der jeweiligen Eigenschaft, während umgekehrt ein geringer Wert einen Mangel der jeweiligen Eigenschaft aufdeckt. Die Mittelwerte aller durch den AttrakDiff2-Fragebogen erfassten, subjektiv wahrgenommenen Qualitäten werden im Diagramm in Abbildung 7.12 aufgezeigt.

Die pragmatische Qualität der Schnellstartbuttons ($\sigma^2 \approx 2,38$) befindet sich im überdurchschnittlichen Bereich, während die Realisierung der Automatisierung ($\sigma^2 \approx 2,90$) als neu-

tral eingestuft wird. Ein zweiseitiger T-Test für Stichproben mit gleichen Varianzen widerlegt dabei mit einem p-Wert von ca. $3,35 * 10^{-5}$ ($F \approx 0,81$) die Annahme, dass sich die beiden Mittelwerte nicht signifikant unterscheiden. Die Schnellstartbuttons werden demnach von den Probanden im Gegensatz zu der Automatisierung als sehr praktisch eingestuft und in Hinsicht auf die Zweckdienlichkeit bevorzugt. Die beiden Formen der kontextsensitiven Personalisierung weisen mit einem p-Wert von ca. 0,84 ($F \approx 0,98$) jedoch keinen signifikanten Unterschied bei der hedonischen Qualität bzgl. der Identifikation auf. In beiden Fällen besteht zwar die Möglichkeit, sich mit den beiden kontextsensitiven Personalisierungen zu identifizieren, aber letztendlich entspricht die Benutzerschnittstelle den gewohnten Standards. Bei der hedonischen Qualität der Stimulation hingegen zeigt die Automatisierung ($\sigma^2 \approx 1,55$) überdurchschnittliche Werte. Bestätigt wurde diese Erkenntnis bereits während der Benutzerevaluation, da das automatische Erscheinen der Signalisierung und der daraufhin stattfindende Wechsel des Radiosenders oftmals eine positive Verblüffung bei den Probanden auslöste. Das in den Augen der Probanden erwartungsgemäße Verhalten der Schnellstartbuttons führte jedoch dazu, dass sich die Probanden nur geringfügig ($\sigma^2 \approx 1,52$) von dieser konkreten Realisierung fesseln ließen. Die Ungleichheit der Eindrücke wird auch durch den mit einem p-Wert von ca. $4,4 * 10^{-7}$ ($F \approx 0,97$) nachweislich signifikanten Unterschied der Mittelwerte deutlich. Insgesamt wurden die beiden konkreten Realisierungen jedoch als überdurchschnittlich attraktiv empfunden (σ^2 von ca. 1,03 für die Schnellstartbuttons und ca. 1,45 für die Automatisierung). Die Attraktivität der Schnellstartbuttons ist jedoch signifikant (mit einem p-Wert von ca. 0,01 ($F \approx 0,70$)) höher als bei der Automatisierung. In den abschließenden Bemerkungen zur Studie wurde dieser Unterschied oftmals ebenfalls informell bekundet, indem sich die Probanden mehrheitlich für die Schnellstartbuttons aussprachen.

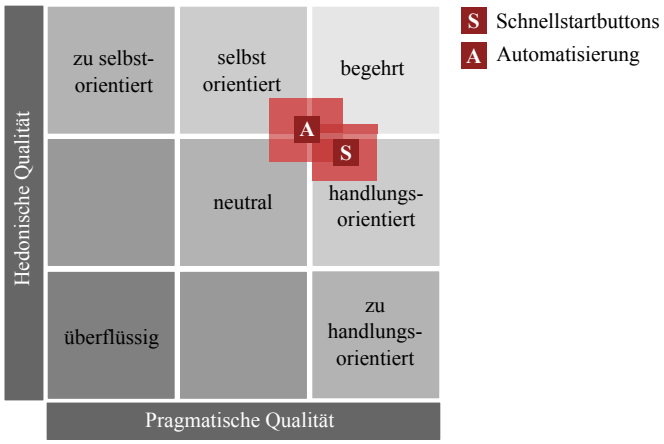


Abb. 7.13 Portfoliodarstellung mit den Mittelwerten und den Konfidenzintervallen ($\alpha = 0,05$) der pragmatischen und hedonischen Qualität.

Werden die Ergebnisse der hedonischen Qualitätswahrnehmung zusammengefasst und gemeinsam mit der pragmatischen Qualität auf die von Burmester in [Burmester u. a., 2007] vorgestellte Portfoliodarstellung aus Abbildung 7.13 abgetragen, so zeigt sich, dass die Ergebnisse insbesondere der Automatisierung nicht genau einem Bereich zugeordnet werden können. Eine abschließende Bewertung der Ergebnisse bzgl. der Automatisierung ist daher nicht möglich. Laut der Interpretationshilfe aus [Burmester u. a., 2007] kann es sich bei die-

ser konkreten Realisierung um ein durchschnittliches Produkt, welches zwar fasziniert und stimuliert, aber eine ineffiziente Bedienung vorschreibt, oder aber um eine optimale Lösung handeln, die den Nutzer unterstützt und gleichzeitig anregt. Es kann sich jedoch genauso wie die Schnellstartbuttons auch um ein Produkt handeln, welches zwar den Ansprüchen der Aufgabenerfüllung genügt, aber weder neugierig macht, noch zur Identifikation einlädt. Die Ergebnisse bzgl. der Schnellstartbuttons weisen jedoch einen Konfidenzbereich auf, der sich größtenteils mit dem zuletzt genannten Bereich überschneidet. Die kontextsensitiv personalisierten Schnellstartbuttons sind somit aller Wahrscheinlichkeit nach, auf Kosten der hedonischen Qualität, zu handlungsorientiert. Eine Reduzierung der Streuung und die damit einhergehende verbesserte Einordnung der beiden Formen der kontextsensitiven Personalisierung kann möglicherweise über einen höheren Stichprobenumfang erzielt werden.

Teil IV
Résumé

Research is to see what everybody else has seen,
and to think what nobody else has thought."

Albert von Szent-Györgyi, 1893-1986

Kapitel 8
Fazit

Dieses Kapitel widmet sich der abschließenden Zusammenfassung und Diskussion aller in dieser Forschungsarbeit entwickelten Lösungsansätze zur Umsetzung und Anwendung kontextsensitiv personalisierter Bedienelemente im Bereich automotiver Benutzerschnittstellen. Dazu werden einleitend die einzelnen Problemstellungen rekapituliert und den vorgestellten Verfahren bzw. Methoden gegenübergestellt. Die notwendigen Voraussetzungen bzw. Annahmen, die vor oder während des Einsatzes der kontextsensitiven Personalisierung im Fahrzeug zwangsläufig erfüllt sein müssen und die Einschränkungen, die sich insbesondere aus dem strukturellen Aufbau des Systemdienstes ergeben, werden danach im Einzelnen dargelegt und diskutiert. Daraufhin werden alle noch offenen Fragen bzw. ungelöste Problematiken, die im Rahmen dieser Forschungsarbeit nicht behandelt wurden, aber deren Beantwortung bzw. Untersuchung - für eine erfolgreiche Anwendung der kontextsensitiven Personalisierung im automotiven Bereich - unumgänglich ist, im Detail erörtert. Der letzte Abschnitt befasst sich dann mit weiterführenden Ideen, die sich entweder auf einzelne Verbesserungen der vorgestellten Verfahren oder auf mögliche Erweiterungen z.B. der Benutzerschnittstelle beziehen.

8.1 Problemstellungen und Ergebnisse

Das grundlegende Ziel der Untersuchungen bestand in dem Entwurf und der Implementierung eines Verfahrens zur kontextsensitiven Personalisierung von Mensch-Maschine-Interaktionen. Im Mittelpunkt stand dabei die beispielhafte Umsetzung und Anwendung der kontextsensitiven Personalisierung im Bereich automotiver Benutzerschnittstellen. Es galt, die insbesondere im automotiven Bereich zahlreich vorhandenen Sensordaten implizit auf gegebenenfalls vorhandene statistisch relevante Korrelationen zwischen den bekannten Bedienmustern bei In-Car-Infotainment-Systemen und den individuellen Nutzungskontexten zu untersuchen. Die automotive Benutzerschnittstelle sollte sich dann auf Grundlage der dabei ermittelten kontextsensitiven Präferenzen automatisch an die Bedürfnisse des Benutzers in der jeweiligen Situation anpassen. Abschließend sollte die potentielle Anwendbarkeit der kontextsensitiven Personalisierung im automotiven Bereich anhand einer Benutzerevaluation, die es an einer zu implementierenden prototypischen automotiven Benutzerschnittstelle durchzuführen galt, unter Beweis gestellt werden.

Als Herausforderung stellte sich dabei vor allem die automatische Analyse der Sensordaten dar, die möglichst ohne einen Eingriff durch den Benutzer und im Hintergrund durch-

geführt werden sollte. Die Mannigfaltigkeit des potentiellen Benutzer- und Umgebungsverhaltens führt während der Datenverarbeitung zu einer Vielzahl von individuellen und unterschiedlichen Bedienmustern, für die es jeweils - ebenfalls automatisiert - eine passende Modifikation der Benutzerschnittstelle zu suchen gilt. Irrelevante Bedienmuster können dabei Modifikationen auslösen, die die Effizienz der Bedienung reduzieren und/oder den Benutzer verwirren. Da die automotive Benutzerschnittstelle jedoch in einem sicherheitskritischen Umfeld genutzt wird, für das besondere Restriktionen und Bestimmungen gelten, wurde in diesem Fall ein regelbasiertes Verfahren der Datenanalyse entwickelt. Somit wird sichergestellt, dass nur praktisch relevante Bedienmuster erkannt und entsprechend erlaubte Modifikationen durchgeführt werden können.

Das Verfahren umfasst unter anderem die automatisierte Erkennung sowohl bekannter Bedienmuster, als auch deren Korrelationen mit den benutzerabhängigen Ausprägungen des Kontextes. Dazu galt es, eine Möglichkeit zu finden, bekannte Bedienmuster formal für den Analyseprozess zu erfassen. Die bekannten Bedienmuster werden dem Verfahren daher in Form von Interaktionsmustern bereitgestellt, deren Definitionen in Form von EPL-Anweisungen in den Vorlagen zu finden sind. Der Prozess zur Ermittlung der Bedienmuster kann dann mit jedem beliebigen Pattern-Matching-Algorithmus, der die oben beschriebenen EPL-Techniken implementiert, umgesetzt werden. Dabei blieb zunächst offen, in welcher Weise die etwaigen Korrelationen zwischen den erkannten Bedienmustern und den während der Erkennung gültigen Kontexten ermittelt werden sollten. Bestimmte Aspekte des Kontextes, die üblicherweise im Verdacht stehen, spezielle Bedienmuster zu begünstigen, auszulösen oder mit diesen zu korrelieren, werden dem Verfahren daher ebenfalls über die Vorlagen zur Verfügung gestellt. Erkannte Bedienmuster können somit anhand der Ausprägungen des zu berücksichtigen Kontextes über den Einsatz eines dichtebasierten, nicht-überwachten Clustering-Algorithmus automatisiert gruppiert und zu kontextsensitiven Präferenzen verarbeitet werden. Da die Situation, in der eine kontextsensitive Personalisierung der Benutzerschnittstelle ausgeführt werden soll, nicht immer im vollem Umfang mit der Situation übereinstimmt, die in der dazugehörigen kontextsensitiven Präferenz ermittelt wurde, galt es außerdem, parametrisierbare Situationsmuster einzuführen. Bei dem Situationsmuster handelt es sich um eine in den Vorlagen zu definierende EPL-Anweisung, mit der sich eine zu erkennende Situation in gleicher Weise wie ein Interaktionsmuster durch ein sequentielles Ereignismuster beschreiben lässt. Ein Situationsmuster wird, im Gegensatz zum Interaktionsmuster, jedoch erst während der Laufzeit durch eine dazugehörige kontextsensitive Präferenz konkretisiert. Die Ermittlung der Situationen, deren Erkennung folglich jeweils zu einer Modifikation der Benutzerschnittstelle führt, findet somit ebenfalls über Pattern-Matching-Algorithmen statt.

Das Verfahren erlaubt die Definition beliebiger Interaktions- und Situationsmuster, die sich mit Hilfe der oben beschriebenen EPL-Mechanismen in den Vorlagen formulieren lassen. Die freie Wahl der zu berücksichtigen Aspekte des Kontextes erweist sich ebenfalls als praktisch, da für jeden Anwendungsfall die passende Untermenge dieser Aspekte definiert werden kann. Zusammen mit den variabel zu definierenden Distanzfunktionen und den zugehörigen Entfernungen für die Gruppierung können die Vorlagen überdies an die unterschiedlichen Gegebenheiten verschiedener Absatzmärkte angepasst werden. Auch eine mögliche Einsicht des Benutzers in die Vorlagen und der damit verbundenen optionalen und manuellen Anpassung der Vorlagen ist denkbar, da die Vorlagen bei jedem Start des Fahrzeugs erneut eingelesen werden. Durch die Auslagerung der Datenanalyse in einen Systemdienst wird außerdem sichergestellt, dass die kontextsensitiven Präferenzen vor der unerlaubten Einsicht durch Anwendungen von Dritt-Anbietern geschützt sind, sich aber trotz-

dem gegebenenfalls applikationsübergreifend nutzen lassen. Darüber hinaus ermöglicht die logische Trennung des Prozesses der Benutzermodellierung von der Umsetzung der Benutzeroberfläche die Verlegung der ressourcenintensiven Datenanalyse und Persistenz von dem eingebetteten System im Fahrzeug hin zu der das Fahrzeug umgebenden Infrastruktur. Voraussetzung hierfür ist jedoch eine möglichst unterbrechungs- und verzögerungsfreie Breitbandanbindung des Fahrzeugs an die Infrastruktur. Das Verfahren wurde zwar im Rahmen der Personalisierung automotiver Benutzeroberflächen entwickelt, es kann aber auch in diversen Bereichen, wie z.B. der Heimautomatisierung oder benutzeradaptiven Systemen i. Allg., zum Einsatz kommen.

Um die Validität des Verfahrens nachzuweisen, wurde sowohl der Systemdienst als auch die dazu passende automotive Benutzerschnittstelle prototypisch implementiert. Die Umsetzung des Systemdienstes umfasst dabei alle Funktionalitäten, die in Kapitel 4 vorgestellt und in Kapitel 5 konkretisiert wurden. Da sich die bisherigen Forschungsansätze zur kontextsensitiven Personalisierung fast ausschließlich auf eine Anwendung in Mobiltelefonen oder auf den Forschungsbereich der Heimautomatisierung konzentrieren, und die kontextsensitiven Präferenzen im Bereich der automotiven Benutzerschnittstellen bisher lediglich bei der Berechnung der Route im Navigationssystem berücksichtigt werden, handelt es sich bei der prototypischen Benutzerschnittstelle um die erste Umsetzung einer kontextsensitiven Personalisierung im automotiven Bereich. Sie umfasst die automatische Ausführung und Empfehlung von Funktionen, die der Benutzer in den entsprechenden Situationen bevorzugt ausführt. Außerdem können die implizit ermittelten kontextsensitiven Präferenzen in einer angemessenen Form eingesehen, aktiviert und deaktiviert werden.

Überprüft wurde die Praktikabilität dieser Umsetzung durch eine mit 31 Probanden durchgeführte Benutzerevaluation, aus deren Ergebnissen sich insbesondere ein signifikant geringerer Bedienaufwand bei der Benutzung und eine neutrale bis hohe hedonische und pragmatische Qualität der beiden vorgestellten Formen der kontextsensitiven Personalisierung nachweisen ließ. Bei der Benutzerevaluation wurde außerdem der Frage nachgegangen, ob eine sich an die Benutzungsgewohnheiten anpassende automotive Benutzerschnittstelle aus Sicht der potentiellen Nutzer überhaupt erwünscht ist. 25 von 31 Probanden sprachen sich nach der Nutzung des Prototyps für eine automatische Anpassung der automotiven Benutzerschnittstelle aus, wenn einzelne Bedingungen, beispielsweise bzgl. der automatischen Personalisierung nur mit ausdrücklicher Zustimmung durch den Benutzer, erfüllt werden. Weitere Ergebnisse der Benutzerevaluation weisen jedoch darauf hin, dass die Wahrung der Privatsphäre durch das System in beiden Formen der kontextsensitiven Personalisierung nicht immer zufriedenstellend eingehalten und das Gefühl der vollständigen Kontrolle über das System, insbesondere bei der automatischen Ausführung von Funktionen, nicht immer gegeben ist. Aufgrund des Umfangs und der Zusammensetzung des Probandenfeldes handelt es sich bei der Benutzerevaluation allerdings um keine repräsentative Befragung bzw. Evaluation. Eine Bestätigung der Ergebnisse kann demnach nur im Rahmen einer breit angelegten Studie erfolgen.

Während der Entwicklung des oben vorgestellten Verfahrens der kontextsensitiven Personalisierung offenbarte sich außerdem die Herausforderung, die Funktionalitäten kontextsensitiv personalisierter Systeme in einem akzeptablen Zeitrahmen bzw. vertretbaren Arbeitsaufwand zu untersuchen bzw. zu testen. Da der Systemdienst anfänglich nur in Echtzeit mit den Sensordaten versorgt werden konnte, ließen sich die zahlreichen Interaktionen, die es zur Erkennung einer kontextsensitiven Präferenz zu simulieren galt, nur manuell durchführen. Aus diesem Grund wurde im Rahmen dieser Forschungsarbeit ein Verfahren zur Modellierung, Generierung und Simulation szenarienspezifischen Benutzer- und Umge-

bungsverhaltens entwickelt, welches in einer frühen Phase der Entwicklung die quantitative Simulation ähnlicher Interaktionen ermöglicht. Das manuelle Ausführen der Interaktionen in Echtzeit wird dabei ersetzt durch die automatische Ausführung generierter Interaktionen in simulierter Zeit. In dem Verfahren wird der Zeitrahmen für die Generierung aller szenarienspezifischer Sensordaten zunächst auf der Basis eines modifizierten und erweiterten DEVS-Formalismus modelliert. Eine Simulationsumgebung generiert dann auf Grundlage des Modells eine beliebige Anzahl ähnlicher szenarienspezifischer Sequenzen von Ereignissen und übermittelt diese anschließend dem zu untersuchenden System. Die Parameter einzelner Ereignisse werden dabei dem Modell entnommen, während die Parameter periodischer Ereignisse aufgrund der Vielfalt an Möglichkeiten durch die jeweilige Implementierung bestimmt werden. Das Verfahren macht es möglich, den Systemdienst auch über einen längeren simulierten Zeitraum auf das korrekte Erkennen von kontextsensitiven Präferenzen und Situationen hin zu untersuchen. Das mögliche Anwendungsgebiet des Verfahrens und des dazu implementierten Prototyps umfasst alle kontextsensitiv personalisierbaren Systeme, die über eine ereignisgesteuerte Datenverarbeitung verfügen.

8.2 Voraussetzungen und Einschränkungen

Eine wichtige Voraussetzung für den Einsatz des vorgestellten Verfahrens zur kontextsensitiven Personalisierung im automotiven Bereich besteht in der Identifizierbarkeit des Fahrers bzw. Benutzers. Dem System muss zwecks der Abbildung kontextsensitiver Interaktionen auf benutzerspezifische Präferenzen und der Auslösung kontextsensitiver Personalisierungen der Benutzerschnittstelle zu jeder Zeit bekannt sein, welcher Benutzer derzeitig mit dem System interagiert. Zwar können die kontextsensitiven Präferenzen auch ohne die Identifizierung der Benutzer ermittelt und die entsprechenden Personalisierungen ausgeführt werden, jedoch führen diese Art von Präferenzen dazu, dass sich das Fahrzeug nicht individuell an die Bedürfnisse jedes einzelnen Benutzers, sondern einheitlich an die gemeinsamen Bedürfnisse aller Benutzer anpasst. Dies könnte im Einzelfall zu widersprüchlichen Präferenzen führen und in der Konsequenz zu kontraproduktiven Modifikationen der Benutzerschnittstelle. Der Fahrer bzw. Benutzer sollte also vor dem Beginn einer Fahrt entweder automatisch oder manuell, z.B. über eine Gesichtserkennung oder die Eingabe eines Benutzernamens, einer eindeutigen Benutzerkennung zugeordnet werden können.

Des Weiteren setzt das Verfahren voraus, dass bestimmte, allgemeine Bedienmuster bereits vor dem Einsatz des Verfahrens bekannt sind. Ein allgemeines Bedienmuster besteht sowohl aus einer Beschreibung der generellen Struktur einer spezifischen Benutzerinteraktion, z.B. eines Radiosenderwechsels, als auch aus den Aspekten des Kontextes, z.B. Position des Fahrzeugs und der Zeit, mit denen die Benutzerinteraktionen innerhalb des Nutzerkreises üblicherweise korrelieren. Diese Informationen lassen sich im Vorfeld entweder durch Befragungen der Nutzer oder durch eine Beobachtung des Nutzerverhaltens gewinnen. In [Ziebart u. a., 2008] zeigten Ziebart u.a. beispielsweise durch eine Befragung von 21 Probanden, die regelmäßig mit dem Fahrzeug unterwegs sind, dass die Wahl einer geeigneten Route bei 15 Probanden von der Tageszeit und dem Wochentag beeinflusst wird.

Das präzise Ermitteln der bei einem Bedienmuster zu untersuchenden Aspekte des Kontextes ist auch daher von hoher Bedeutung für das Verfahren, da eine kontextsensitive Präferenz erst dann ermittelt werden kann, wenn eine Korrelation des Bedienmusters mit allen spezifizierten Aspekten vorliegt. Tritt ein Bedienmuster also häufig im Zusammen-

hang mit einer spezifischen Ausprägung eines Aspekts des Kontextes auf und wird in den Vorlagen dem Bedienmuster ein weiterer Aspekt zugeordnet mit dem das Bedienmuster jedoch nicht korreliert, so kann keine kontextsensitive Präferenz erkannt werden. Die statische Spezifikation der zu berücksichtigenden Aspekte des Kontextes kann außerdem zu dem sogenannten „curse of dimensionality"-Problem führen, welches die grundlegende Schwierigkeit beschreibt, statistisch relevante Korrelationen in hochdimensionalen Daten zu erkennen. Übersteigt die Anzahl der zu berücksichtigenden Aspekte demnach einen kritischen Wert, so sinkt die Wahrscheinlichkeit, dass eine kontextsensitive Präferenz erkannt werden kann, deutlich unter einen für die kontextsensitive Personalisierung geeigneten Wert. Zwar kann der künstliche Verfall dahingehend angepasst werden, dass bei der Korrelationsanalyse auch historisch weit zurückliegende Interaktionen berücksichtigt werden können, jedoch konterkariert dieses Vorgehen das Ziel, eine Benutzerschnittstelle bereitzustellen, die die kontextsensitiven Präferenzen möglichst schnell erlernt. Ein Bedienmuster sollte in den Vorlagen folglich immer nur mit einer geringen Zahl kontextueller Aspekte in Verbindung gebracht werden. Diese Vorgehensweise steigert aus Sicht des Benutzers überdies auch die Nachvollziehbarkeit der kontextsensitiven Personalisierungen. Inwiefern das Verfahren dahingehend modifiziert werden kann, die Erkennung kontextsensitiver Präferenzen in Bezug auf die zu berücksichtigenden Aspekte des Kontextes dennoch flexibler zu gestalten, wird im Abschnitt 8.4 erläutert.

Weitere Einschränkungen des Verfahrens gilt es bei der Formulierung der Interaktions- und Situationsmuster zu beachten. Die Komplexitätsanalyse in Unterabschnitt 4.2.5 ergab, dass sich nicht nur die Anzahl der zu erkennenden Interaktions- und Situationsmuster, sondern insbesondere die Zusammensetzungen der EPL-Anweisungen maßgeblich auf den Aufwand der Datenanalyse auswirken. Demnach sollten Subausdruck-Operatoren in einer EPL-Anweisung zum einen begrenzt, und zum anderen nur dann verwendet werden, wenn die zu erkennenden Ereignisse nicht allzu häufig im Strom der Sensordaten auftreten. Andernfalls führt dies während des Pattern Matchings zu einer hohen Anzahl parallel auszuführender Untersuchungen von gültigen Teilsequenzen eines Musters und der damit zusammenhängenden hohen Auslastung des Arbeitsspeichers. Auch der Zeitrahmen und damit die Länge des Musters sollten mit Bedacht gewählt werden, um den Aufwand in Grenzen zu halten und den Benutzer nicht mit sich z.B. über mehrere Tage erstreckenden Interaktions- oder Situationsmustern mental zu überfordern.

Die Skalierbarkeit des Prozesses zur Erkennung der Interaktions- und Situationsmuster ist unter der Annahme, dass die Ereignisse mit einer begrenzten Rate auftreten und die EPL-Anweisungen der Interaktions- und Situationsmuster sich nicht ändern, nur dann gegeben, wenn sich die parallel auszuführenden Untersuchungen für jede neue Instanz eines konkretisierten Situationsmusters, welches sich aus einer neuen kontextsensitiven Präferenz ableitet, auf unterschiedliche, möglichst unabhängig arbeitende CPUs verteilen lassen. Kritischer hingegen stellt sich die Komplexität des GDBSCAN-Algorithmus zur Gruppierung der Interaktionen dar. Liegen die Interaktionen nicht in geeigneter Form sortiert vor, so führt die Komplexität von $\mathcal{O}(n^2)$ dazu, dass das Verfahren hinsichtlich der Anzahl relevanter Interaktionen schlecht skaliert. In der Praxis lassen sich die Interaktionen jedoch aufgrund der durch den künstlichen Verfall ohnehin begrenzten Menge zu gruppierender Interaktionen und der daher geringen und begrenzten Anzahl möglicher kontextsensitiver Präferenzen mit Hilfe heute zur Verfügung stehender Hardware in einem vertretbaren Zeitrahmen verarbeiten. Das testweise mit der prototypischen Implementierung durchgeführte Gruppieren von 5000 Interaktionen und sich daraus ergebenden 100 kontextsensitiven Präferenzen eines Anwendungsfalls auf einem Rechner mit Doppelkernprozessor (jeweils 3,10 GHz), 8

GB Arbeitsspeicher und mit dem Betriebssystem Microsoft Windows 7[1] dauerte im Durchschnitt weniger als 10 Sekunden.

8.3 Offene Fragen

Ein kontextsensitiv personalisierbares In-Car-Infotainment-System ist trotz der im automotiven Bereich zahlreich vorhandenen Sensoren nicht immer in der Lage, die Benutzerpräferenzen hinreichend genau zu modellieren oder die Situationen in ihrer Gänze korrekt zu erfassen. Steht der Benutzer beispielsweise aus terminlichen Gründen, die das System nicht ermitteln kann, unter Zeitdruck, so werden die Situationen vom System möglicherweise nur unvollständig interpretiert. In einigen Situationen kann dies infolgedessen zu kontextsensitiven Personalisierungen der Benutzerschnittstelle führen, die der Benutzer weder erwartet, noch als angenehm empfindet. Der Einsatz neuartiger Sensoren und verbesserter Algorithmen zur Auswertung bestimmter kontextueller Aspekte wird die Fehlerrate in Zukunft zwar reduzieren, aber nicht gänzlich beseitigen können. Dabei ergibt sich die generelle Frage, mit welcher Fehlerrate beim hier vorgestellten Verfahren der kontextsensitiven Personalisierung zu rechnen ist, wenn ausschließlich Sensoren zum Einsatz kommen, die bereits heute im Fahrzeug verbaut werden. Über eine Studie im praktischen Gebrauch müssen dazu die aus Sicht des Benutzers als korrekt und hilfreich empfundenen Personalisierungen sowohl den als falsch und unbrauchbar wahrgenommenen Modifikationen der Benutzerschnittstelle, als auch den zwar erwarteten, aber vom System nicht durchgeführten Personalisierungen gegenübergestellt werden. Eine umfassende Einschätzung bzgl. der Gebrauchstauglichkeit einer Benutzerschnittstelle, deren kontextsensitive Personalisierungen auf Basis des hier vorgestellten Verfahrens durchgeführt werden, kann folglich erst auf Grundlage der Ergebnisse einer solchen Studie erfolgen. Es stellt sich außerdem die Frage, ob sich aus Sicht der Entwickler überhaupt alle vorstellbaren Anwendungsfälle der kontextsensitiven Personalisierung in den Vorlagen beschreiben bzw. formalisieren lassen. Auch hierzu muss eine Studie durchgeführt werden, in der darüber hinaus auch die Komplexität des Prozesses der Anfertigung und Modifikation der Vorlagen thematisiert werden kann.

Die Behandlung widersprüchlicher kontextsensitiver Präferenzen stellt sich zudem als schwierig dar. Solche Präferenzen können dann auftreten, wenn der Benutzer in einer bestimmten Situation mehrmals auf die gleiche Art und Weise, unter der Berücksichtigung verschiedener Einstellungen, über die Benutzerschnittstelle mit dem System interagiert. Ein Benutzer kann in der gleichen Region z.B. häufig sowohl den Radiosender A als auch den Radiosender B hören. Welcher Radiosender hingegen bevorzugt wird, ist dem Systemdienst, evtl. aufgrund fehlender Sensorik, nicht bekannt. Auf der Ebene der Benutzerschnittstelle lassen sich die mit widersprüchlichen Präferenzen einhergehenden Probleme, wie z.B. die gleichzeitige Ausführung sich widersprechender Automatisierungen, durch die Einführung der vorgestellten Feedbackansicht umgehen. Widersprüchliche Aktivierungen kontextsensitiver Automatisierungen könnten damit von der Benutzerschnittstelle bereits frühzeitig erkannt und geeignet abgefangen werden. In analoger Weise lassen sich damit auch die Empfehlungen, die durch die Schnellstartbuttons visualisiert werden, steuern. Z.B. ließe sich die Anzahl der in einer Situation zu empfehlenden widersprüchlichen Präferenzen auf drei beschränken. In beiden Fällen liegt es in der Hand des Benutzers, widersprüchliche Präferenzen durch Aktivierung bzw. Deaktivierung aufzulösen.

[1] http://www.microsoft.com

Ist jedoch der Einsatz einer Feedbackansicht nicht erwünscht, so tritt das oben erwähnte und im Zusammenhang mit der kontextsensitiv personalisierten Automatisierung auftretende Problem der Ausführung widersprüchlicher Automatisierungen auf. Im Fall der kontextsensitiv personalisierten Schnellstartbuttons führt eine hohe Anzahl widersprüchlicher Empfehlungen dazu, dass nur ein Teil dieser Empfehlungen angezeigt werden kann. Einige widersprüchliche Präferenzen, die in der jeweiligen Situation möglicherweise wichtiger sind als andere, können dem Benutzer daher unter Umständen verborgen bleiben[2]. Wäre der Systemdienst nun in der Lage, (widersprüchliche) kontextsensitive Präferenzen automatisiert zu priorisieren, so würden im Rahmen der kontextsensitiv personalisierten Schnellstartbuttons nur die Funktionen empfohlen, deren Priorisierung am höchsten ist. Im Fall der kontextsensitiv personalisierten Automatisierung wäre es damit für die Benutzerschnittstelle möglich, sich für eine von zwei oder mehreren sich widersprechenden Automatisierungen - auch ohne eine über die Feedbackansicht angegebene manuelle Priorisierung - zu entscheiden. Die Behandlung widersprüchlicher kontextsensitiver Präferenzen kann also entweder über eine Feedbackansicht oder durch eine automatische Priorisierung der kontextsensitiven Präferenzen erfolgen. Inwiefern sich allerdings eine automatische Priorisierung (widersprüchlicher) kontextsensitiver Personalisierungen innerhalb des Verfahrens umsetzen lässt, bleibt offen.

Im Laufe der Zeit unterliegen kontextsensitive Präferenzen diversen Änderungen, die sich entweder auf den künstlichen Verfall der aufgezeichneten relevanten Interaktionen oder auf das Hinzukommen neuer relevanter Interaktionen zurückführen lassen. Verändert werden dabei nicht die Einstellungen der kontextsensitiven Präferenzen, sondern die jeweiligen Ausprägungen der Situationen. Das Ausmaß der Ausprägungen einzelner Aspekte des Kontextes kann demnach im Zuge des künstlichen Verfalls verkleinert, oder durch das Auftreten neuer relevanter Interaktionen vergrößert werden. Wurde eine kontextsensitive Präferenz aufgrund der wiederholt auftretenden manuellen Ausführung einer bevorzugten Einstellung erkannt, dann kann diese, wie oben beschrieben, in der entsprechenden Situation entweder automatisiert oder empfohlen werden. Ab diesem Zeitpunkt wird die bevorzugte Einstellung in der entsprechenden Situation also nicht mehr überwiegend manuell, sondern größtenteils über die kontextsensitive Personalisierung ausgeführt. Löst aber nur das manuelle Einstellen eine relevante Interaktion im Systemdienst aus, dann verringert sich mit der Zeit aufgrund des künstlichen Verfalls die Anzahl der relevanten Interaktionen, die die weitere Existenz dieser kontextsensitiven Präferenz sicherstellen sollten. Wird das manuelle Einstellen in diesem Fall gänzlich eingestellt, dann verliert die kontextsensitive Präferenz nach einer gewissen Zeit ihre Gültigkeit, obwohl die dazugehörigen Automatisierungen nach wie vor in Gebrauch sein könnten.

Da einem Anwendungsfall in den Vorlagen mehr als ein Bedienmuster zugewiesen werden kann, wäre es alternativ möglich, die über die kontextsensitive Personalisierung erfolgte Einstellung ebenfalls als relevante Interaktion des gleichen Anwendungsfalls zu interpretieren. Alle neuen relevanten Interaktionen, die z.B. über die kontextsensitiv personalisierte Automatisierung ausgeführt werden, könnten dabei den sich mit der Zeit einstellenden Schwund an relevanten Interaktionen, die aus dem manuellen Einstellen heraus erkannt wurden, kompensieren und die Relevanz der kontextsensitiven Präferenz sicherstellen. Ein solcher Schritt wirkt sich mit der Zeit jedoch nachteilig auf die Ausmaße bzw. Struktur der jeweiligen Situation aus. Abbildung 8.1 zeigt beispielhaft den Verlauf der Veränderungen,

[2] Diese Problematik bezieht sich nicht nur auf widersprüchliche Präferenzen, sondern betrifft das gleichzeitige Anzeigen kontextsensitiver Präferenzen i. Allg..

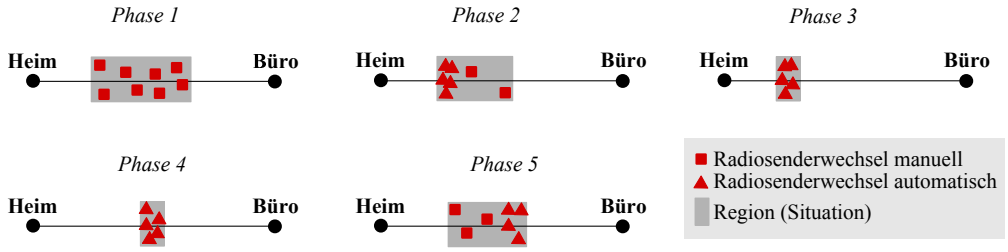


Abb. 8.1 Zeitbedingte Veränderungen des Ausmaßes der Situation am Beispiel des ortsabhängigen Radiosenderwechsels.

die eine Situation des ortsabhängigen Radiosenderwechsels erfährt, wenn der automatisch ausgeführte Radiosenderwechsel ebenfalls als relevante Interaktion interpretiert wird. In der ersten Phase tragen manuelle Radiosenderwechsel, die immer nur auf der Hinfahrt ins Büro ausgeführt werden, dazu bei, dass sich die kontextsensitive Präferenz herausbildet. Danach wird der daraus ableitbare ortsabhängige Radiosenderwechsel automatisiert und die manuellen Radiosenderwechsel gänzlich eingestellt. Aufgrund des künstlichen Verfalls der alten, aufgezeichneten manuellen Radiosenderwechsel und der nur am linken Rand beim Eintritt in die Region hinzugekommenen neuen automatisierten Radiosenderwechsel, verkleinert sich das Ausmaß der ursprünglichen Region, bis in der dritten Phase nur noch eine kleine Region übrig bleibt. Da die technische Erkennung der Situation auf Seiten des Systemdienstes und die damit zusammenhängende Automatisierung (angesichts der Verarbeitungsdauer) immer nur zeitversetzt zum eigentlichen Betreten der Region bzw. dem Eintreten der Situation stattfinden kann, werden die neuen, automatisierten Radiosenderwechsel im Laufe der Zeit immer später auf der Fahrt ausgeführt. Im Fall der kontextsensitiv personalisierten Schnellstartbuttons findet diese Verschiebung der Region sogar noch deutlich schneller statt, da der Benutzer die empfohlene Funktion nicht unmittelbar nach dem Erscheinen des kontextsensitiv personalisierten Schnellstartbuttons ausführt. Aus Sicht des Benutzers stellen die Veränderungen, die sich von der ersten bis zur dritten Phase ergeben, in den meisten Fällen kein Problem dar. Verschiebt sich die Region jedoch so weit, dass die automatisierten Radiosenderwechsel gefühlt zu spät eintreten, dann müssen, entgegensetzt dem Ziel der kontextsensitiven Personalisierung, wieder manuelle Radiosenderwechsel durchgeführt werden. Es bleibt also noch zu klären, welchen Einfluss die über die kontextsensitive Personalisierung durchgeführten Interaktionen im Einzelnen auf die Struktur der betreffenden kontextsensitiven Präferenzen ausüben sollen.

8.4 Ausblick

Wie oben beschrieben, liegt eine grundlegende Schwäche des vorgestellten Verfahrens zur kontextsensitiven Personalisierung in der Tatsache, dass die Erkennung von kontextsensitiven Präferenzen erst dann möglich ist, wenn nicht nur hinreichend viele relevante Interaktionen mit den gleichen Einstellungen aufgezeichnet wurden, sondern sich die relevanten Interaktionen hinsichtlich des Kontextes in allen Aspekten der Umgebung ähneln. Führt der Benutzer beispielsweise einen bestimmten Radiosenderwechsel immer nur in Abhängigkeit von der Position des Fahrzeugs aus, aber wird über die Vorlagen ein Radiosenderwechsel sowohl mit der Position als auch mit der Tageszeit in Verbindung gebracht, so bleibt dem Sys-

tem die kontextsensitive Präferenz des Benutzers verborgen. Zwar wäre es möglich, zwei unterschiedliche Anwendungsfälle in den Vorlagen zu definieren, die jeweils eine Variante beschreiben, jedoch führt dies möglicherweise zur Erkennung zwei unabhängiger kontextsensitiver Präferenzen, die sich nur hinsichtlich eines Aspektes der Umgebung unterscheiden und daher letztendlich zusammengefasst werden müssten. In einer weiteren Ausbaustufe des Verfahrens wäre es daher vorteilhafter, bereits während des Phase des Gruppierens die statistisch relevanten Aspekte der Umgebung anhand der spezifischen Ausprägungen automatisiert zu bestimmen. Dieser als *Feature Selection* bezeichnete Prozess kann entweder vor der eigentlichen Gruppierung erfolgen oder mit der Gruppierung einhergehen. Aufgrund der hohen Anzahl möglicher Aspekte der Umgebung und der damit verbundenen kombinatorischen Explosion des Aufwandes bei der Bestimmung einer geeigneten und statistisch relevanten Teilmenge wird die Suche oftmals durch Heuristiken unterstützt und nur auf bestimmte Aspekte der Umgebung beschränkt. Die Vorlagen könnten in diesem Fall dazu dienen, nicht nur die potentiell relevanten Aspekte der Umgebung zu beschreiben, sondern außerdem noch mögliche und erlaubte Teilmengen zu priorisieren.

Des Weiteren gilt es, insbesondere die Komplexität des Prozesses der Gruppierung insgesamt zu reduzieren. Eine Möglichkeit besteht darin, statt einen auf statischen Daten operierenden Clustering-Algorithmus für die Gruppierung, einen der vielen, überwiegend in den letzten Jahren entwickelten, datenstrom-basierten Clustering-Algorithmen zu verwenden. Diese zeichnen sich hauptsächlich dadurch aus, dass jeder Datenpunkt (hier relevante Interaktion) während des Gruppierens nur einmal gelesen wird und sich die Gruppen mit der Zeit verändern können. Cao u.a. stellen in [Cao u. a., 2006] dazu beispielsweise eine datenstrom-basierte Variante des DBSCAN-Algorithmus vor, der genauso wie der in diesem Verfahren genutzte GDBSCAN-Algorithmus Gruppen mit beliebiger Form erkennt. Da die Verarbeitung innerhalb des hier vorgestellten Verfahrens, bis auf die Gruppierung, ohnehin datenstrom-basiert durchgeführt wird und der künstliche Verfall impliziter Bestandteil datenstrom-basierter Clustering-Algorithmen ist, wäre es nicht mehr nötig, relevante Interaktionen zu speichern. Darüber hinaus sollte untersucht werden, ob es gar möglich ist, die in EPL zur Verfügung stehenden Konzepte zur Agglomeration gleicher konkreter Muster um datenstrom- und dichtebasierte Clustering-Verfahren zu erweitern. Die Formulierung des Interaktionsmusters und der damit korrelierten Aspekte der Umgebung könnten damit sogar, unter der Annahme, dass eine Erweiterung möglich ist, in den Vorlagen innerhalb einer EPL-Anweisung zusammengefasst werden.

Im Zuge einer Verbesserung der Nachvollziehbarkeit kontextsensitiver Personalisierungen wäre es außerdem möglich, die Vorlagen nicht nur durch einen Experten erstellen zu lassen, sondern auch den Benutzer in den Prozess der Parametrisierung des Systemdienstes zu involvieren. Dabei sollte der Benutzer in der Lage sein, über die automotive Benutzerschnittstelle Vorlagen zu kreieren, zu modifizieren und zu löschen. Voraussetzung dafür ist jedoch eine Benutzerschnittstelle, die die Komplexität der Vorlagen auf ein Minimum reduzieren und die Vorlagen statt in Textform in eine geeignete grafische Darstellung übertragen kann. In diesem Zusammenhang wäre es auch denkbar, die Verwaltung der Vorlagen und der kontextsensitiven Präferenzen alternativ auf dem Heim- oder dem Tablet-Rechner zu ermöglichen, um die Vorteile der flexibleren Eingabemöglichkeiten zu nutzen. Kontextsensitiv personalisierte Automatisierungen ließen sich somit, beispielsweise auf mobilen Geräten, bestimmten Profilen zuordnen, die vor einer Fahrt als Ganzes aktiviert oder deaktiviert werden könnten.

Alles im allem lässt sich das Verfahren der kontextsensitiven Personalisierung nicht nur im Bereich automotiver Benutzerschnittstellen, sondern auch bei der Heimautomati-

sierung, in mobilen Geräten wie Smartphones oder Tablet-Rechnern und sogar in Home-Entertainment-Systemen wie beispielsweise TV-Geräten einsetzen. In letzterem Fall ließen sich z.B. die zeit- und geschmacksabhängigen Gewohnheiten bei der Nutzung der TV-Geräte ermitteln. Die TV-Geräte könnten sich dann beispielsweise morgens zu einer bestimmten Uhrzeit automatisch einschalten und auf den bevorzugten Nachrichtenkanal wechseln, während sie Abends Sendungen empfehlen könnten, die der Benutzer üblicherweise zu dieser Zeit konsumiert. Dank der flexiblen Parametrisierung des Verfahrens kann die kontextsensitive Personalisierung z.B. im Fall der Heimautomatisierung sogar an die speziellen Gegebenheiten eines Hauses oder einer Wohnung angepasst werden. Ist beispielsweise eine Garage vorhanden, so ließe sich die Beleuchtung eines Hauses in Abhängigkeit von der Ankunftszeit, der Lichtintensität und diverser Sensordaten aus der Garage nur in dem Teil des Hauses einschalten, der der Garage unmittelbar angeschlossen ist. Betritt der Benutzer das Haus hingegen nur über eine Eingangstür, dann könnte sich neben der Beleuchtung auch der Anrufbeantworter, der sich z.B. unmittelbar an der Eingangstür befinden kann, automatisiert einschalten. Dem hier vorgestellten Verfahren der kontextsensitiven Personalisierung werden somit bzgl. des Anwendungsgebiets kaum Grenzen gesetzt.

Abkürzungen

ABS	Antiblockiersystem
API	Application Programming Interface
BNF	Backus-Naur-Form
CAN	Controller Area Network
CEP	Complex Event Processing
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DES	Diskretes, ereignisbasiertes System
DEVS	Discrete Event System Specification
EMS	Environmental Monitoring System
EPL	Event Pattern Language
GDBSCAN	Generalized Density-Based Spatial Clustering of Applications with Noise
GIS	Geoinformationssystem
GPS	Global Positioning System
HMI	Human Machine Interface
HTML	Hypertext Markup Language
IFC	Industry Foundation Classes
ISTAG	Information Society Technologies Advisory Group
K-SCM	K Symbol String Clustering Map
KNN	Künstliches Neuronales Netz
MP3	Moving Picture Experts Group-1 Audio Layer 3
OPC	One-Pass Clustering
PDA	Personal Digital Assistent
QoSDREAM	Quality of Service for Distributed Reconfigurable Adaptive Multimedia
REST	Representational State Transfer
RFID	Radio-frequency Identification
RSS	Really Simple Syndication
SCM	Symbol String Clustering Map
SMS	Short Message Service
STDEVS	Stochastic Discrete Event System Specification
TCP/IP	Transmission Control Protocol / Internet Protocol
TOSSIM	TinyOS Simulator
UML	Unified Modeling Language
URL	Uniform Resource Locator
USB	Universal Serial Bus
WGS84	World Geodetic System 1984

WLAN	Wireless Local Area Network
XML	Extensible Markup Language
ZBE	Zentrale Bedieneinheit
3G	Dritte Generation im Mobilfunk

Abbildungsverzeichnis

1.1 Bildschirmtastatur und zentrale Bedieneinheit 7

2.1 Kategorisierung offener Systeme 20
2.2 Abstraktionsmöglichkeiten bei der Änderung der Lautstärke 23
2.3 Sequentielles Interaktionsmuster bzgl. der Bedienung des Navigationssystems 26
2.4 Beispielhaftes Markov-Modell des Fahrverhaltens 27
2.5 Dendrogramm zur Visualisierung hierarchischer Gruppierungen 37
2.6 Dichtebasierende Aufteilung beispielhafter Positionsdaten 37
2.7 Beispielhaftes Benutzerverhalten Frau Müllers an Arbeitstagen 40

4.1 Aufbau des Betriebssystems eines In-Car-Infotainment-Systems 64
4.2 Grafische Benutzeroberfläche eines In-Car-Infotainment-Systems 64
4.3 Kommunikation mit dem Systemdienst 65
4.4 Verfahren der kontextsensitiven Personalisierung 67
4.5 Beispielhafte Ausprägungen des Kontextvektors 71
4.6 Unterschiedliche Evaluationen der gleichen Beispielsequenz 78
4.7 Übersicht über den Gruppierungsprozess 82
4.8 Unterschiedliche Aufteilungen durch unterschiedliche Distanzfunktionen . . 86
4.9 Beispielhafte Ausprägungen szenarienspezifischer Gruppenvektoren 89
4.10 Gegenüberstellung historischer und aktueller Gruppeneigenschaften 91
4.11 Verarbeitungsdauern der kontextsensitiven Personalisierung 96
4.12 Korrelation zwischen Anzahl der Subausdruck-Operatoren und Anzahl der Untersuchungen bei der Evaluation 98

5.1 Schematische Übersicht des Prototyps 101
5.2 Benutzeroberfläche während der manuellen Zieleingabe 102
5.3 Einige Ansichten und Menüs der Applikationen 103
5.4 Freiheisgrade des Dreh-Drück-(Schiebe)-Stellers 104
5.5 Curserverhalten am Beispiel der manuellen Zieleingabe 105
5.6 Radioapplikation kurz nach einem relevante Radiosenderwechsel 106
5.7 Kontextsensitiv personalisierte Schnellstartbuttons 107
5.8 Multifunktionstasten an der ZBE 108
5.9 Systemansicht und die dazugehörigen Untermenüs 109
5.10 Feedbackansicht in unterschiedlichen Konfigurationen 109
5.11 Signalisierung einer anstehenden kontextsensitiven Automatisierung 110

5.12 Hauptansicht des ContextSimulators 121
5.13 Einstellungen des Fahrsimulators im ContextSimulator 123
5.14 Fahreransicht des Fahrsimulators 124

6.1 Event Graph mit der Möglichkeit zur Verzweigung 129
6.2 Szenarienspezifisches DEVS-Modell 132
6.3 Szenarienspezifisches STDEVS-Modell 133
6.4 Szenarienspezifisches Teilmodell mit variablen Zeitangaben 134
6.5 Szenarienspezifisches Modell mit periodischen Ereignissen 136
6.6 Hauptansicht der Anwendung zur Generierung und Simulation szenarienspezifischer Ereignissequenzen 139

7.1 Ausschnitt aus dem Einführungsfragebogen 145
7.2 Bewertungsskala aus dem NASA-TLX Fragebogen 145
7.3 Ausschnitt aus den Fragen zu den Anforderungen 146
7.4 Fragen zur Effizienz der Bedienung 146
7.5 Ausschnitt aus dem „AttrakDiff 2“ Fragebogen 147
7.6 Prüfstand aus Sicht des Probanden 147
7.7 Virtuelle Teststrecke durch Berlin 148
7.8 Schematischer Ablauf der Testfahrten 149
7.9 Einstellung gegenüber adaptiven automotiven Benutzerschnittstellen 151
7.10 Arbeitsbelastungen während der Radiosenderwechsel 152
7.11 Ergebnisse zur Effizienz und den Anforderungen 153
7.12 Mittelwerte der subjektiven Qualitätswahrnehmung 154
7.13 Portfoliodarstellung der pragmatischen und hedonischen Qualität 155

8.1 Zeitbedingte Veränderungen des Ausmaßes der Situation am Beispiel des ortsabhängigen Radiosenderwechsels 166

Tabellenverzeichnis

2.1 Beispielhafte Schlüsselwertpaare 25

4.1 Szenarienspezifische Definition des Kontextes 72
4.2 Szenarienspezifische Definition der relevanten Interaktionsmuster 81
4.3 Szenarienspezifische Definition der exklusiven Vergleichseigenschaften 84
4.4 Szenarienspezifische Definition der nicht-exklusiven Vergleichseigenschaften 85
4.5 Szenarienspezifische Definition der generischen Situationsmuster 95

Literaturverzeichnis

[van der Aalst u. a. 2004] AALST, W. M. P. d. ; WEIJTERS, A. J. M. M. ; MARUSTER, L.: Workflow Mining: Discovering process models from event logs. In: *IEEE Transactions on Knowledge and Data Engineering* 16 (2004), S. 1128–1142

[Aamodt u. Plaza 1994] AAMODT, A. ; PLAZA, E.: Case-Based Reasoning: Foundational Issues, Methodological Variations, and System Approaches. In: *AI Communications* 7 (1994), S. 39–59

[Aarts 2004] AARTS, E.: Ambient Intelligence: A Multimedia Perspective. In: *IEEE Multimedia* 11 (2004), Nr. 1, S. 12 – 19

[Agrawal u. a. 2008] AGRAWAL, J. ; DIAO, Y. ; GYLLSTROM, D. ; IMMERMAN, N.: Efficient Pattern Matching over Event Streams. In: *Proc. of the 2008 ACM SIGMOD Int. Conf. on Management of Data*. New York, NY, USA : ACM, 2008 (SIGMOD '08), S. 147–160

[Agrawal u. Srikant 1994] AGRAWAL, R. ; SRIKANT, R.: Fast Algorithms for Mining Association Rules. In: *Proceedings of the 20th Int. Conf. on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994 (VLDB), S. 487–499

[Agrawal u. Srikant 1995] AGRAWAL, R. ; SRIKANT, R.: Mining Sequential Patterns. In: *Proc. of the 11th Int. Conf. on Data Engineering*. Washington, DC, USA : IEEE Computer Society, 1995 (ICDE '95), S. 3–14

[Anderson u. a. 2002] ANDERSON, C. R. ; DOMINGOS, P. ; WELD, D. S.: Relational Markov Models and their Application to Adaptive Web Navigation. In: *Proc. of the 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*. New York, NY, USA : ACM, 2002, S. 143–152

[Armentano u. Amandi 2009] ARMENTANO, M. G. ; AMANDI, A. A.: Recognition of User Intentions for Interface Agents with Variable Order Markov Models. In: HOUBEN, G.-J. (Hrsg.) ; MCCALLA, G. (Hrsg.) ; PIANESI, F. (Hrsg.) ; ZANCANARO, M. (Hrsg.): *User Modeling, Adaptation, and Personalization* Bd. 5535. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2009, S. 173–184

[Ashbrook u. Starner 2002] ASHBROOK, D. ; STARNER, T.: Learning Significant Locations and Predicting User Movement with GPS. In: *Proceedings of 6th Int. Symposium on*

Wearable Computers, 2002, S. 101 – 108

[Augusto u. McCullagh 2007] AUGUSTO, J. C. ; MCCULLAGH, P.: Ambient Intelligence: Concepts and Applications. In: *Computer Science and Information Systems* 4 (2007), Nr. 1, S. 1–27

[Aztiria u. a. 2009a] AZTIRIA, A. ; AUGUSTO, J. C. ; IZAGUIRRE, A. ; COOK, D.J.: Learning Accurate Temporal Relations from User Actions in Intelligent Environments. In: *Proc. of 3rd Symposium of Ubiquitous Computing and Ambient Intelligence* Bd. 51. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2009, S. 274–283

[Aztiria u. a. 2010a] AZTIRIA, A. ; IZAGUIRRE, A. ; AUGUSTO, J. C.: Learning patterns in ambient intelligence environments: a survey. In: *Artificial Intelligence Review* 34 (2010), S. 35–51

[Aztiria u. a. 2009b] AZTIRIA, A. ; IZAGUIRRE, A. ; BASAGOITI, R. ; AUGUSTO, J. C.: Learning about preferences and common behaviours of the user in an intelligent environment. In: *BMI Book*, IOS Press, 2009 (Ambient Intelligence and Smart Environments), S. 289–315

[Aztiria u. a. 2010b] AZTIRIA, A. ; IZAGUIRRE, A. ; BASAGOITI, R. ; AUGUSTO, J.C. ; COOK, D.J.: Automatic Modeling of Frequent User Behaviours in Intelligent Environments. In: *6th Int. Conf. on Intelligent Environments*, 2010, S. 7 –12

[Baldauf u. a. 2007] BALDAUF, M. ; DUSTDAR, S. ; ROSENBERG, F.: A survey on context-aware systems. In: *Int. Journal of Ad Hoc and Ubiquitous Computing* 2 (2007), S. 263–277

[Barton u. Vijayaraghavan 2003] BARTON, J. J. ; VIJAYARAGHAVAN, V.: UBIWISE, A Simulator for Ubiquitous Computing Systems Design / Hewlett-Packard Labs. 2003. – Forschungsbericht

[Battestini u. Flanagan 2005] BATTESTINI, A. ; FLANAGAN, J. A.: Modelling and Simulating Context Data in a Mobile Environment. In: *Workshop on Context Awareness for Proactive Systems(CAPS)*, 2005, S. 127–136

[Bauer 2004] BAUER, M.: Transparent User Modeling for a Mobile Personal Assistant. In: *12th GI-Workshop Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen*, 2004

[Bernsen u. Dybkjaer 2005] BERNSEN, NielsOle ; DYBKJAER, Laila: Enhancing the Usability of Multimodal Virtual Co-drivers. In: MINKER, W. (Hrsg.) ; BÜHLER, Dirk (Hrsg.) ; DYBKJÆR, Laila (Hrsg.): *Spoken Multimodal Human-Computer Dialogue in Mobile Environments* Bd. 28. Springer Netherlands, 2005, S. 269–285

[Berstel 2002] BERSTEL, B.: Extending the RETE Algorithm for Event Management. In: *Proc. of the 9th Int. Symposium on Temporal Representation and Reasoning*, 2002, S. 49 – 51

[Böhmer u. Bauer 2010] BÖHMER, M. ; BAUER, G.: Exploiting the Icon Arrangement on Mobile Devices as Information Source for Context-awareness. In: *Proc. of the 12th Int.*

Conf. on Human Computer Interaction with Mobile Devices and Services. New York, NY, USA : ACM, 2010 (MobileHCI '10), S. 195–198

[Broens u. van Halteren 2006] BROENS, T. ; HALTEREN, A. van: SimuContext: Simply Simulate Context. In: *Proc. of the Int. Conf. on Autonomic and Autonomous Systems*. Washington, DC, USA : IEEE Computer Society, 2006 (ICAS '06), S. 45

[Brown 1996] BROWN, P. J.: The stick-e Document: a framework for creating context-aware applications. In: *Proc. of Electronic Publishing*, Electronic Publishing, 1996, S. 259–272

[Broy u. a. 2009] BROY, M. ; LEUXNER, C. ; SITOU, W. ; SPANFELNER, B. ; WINTER, S.: Formalizing the Notion of Adaptive System Behavior. In: *Proc. of the 2009 ACM Symposium on Applied Computing*, ACM, 2009 (SAC '09), S. 1029–1033

[Bruneau u. a. 2009] BRUNEAU, J. ; JOUVE, W. ; CONSEL, C.: DiaSim: A Parameterized Simulator for Pervasive Computing Applications. In: *Proc. of the Int. Conf. on Mobile and Ubiquitous Systems: Computing, Networking and Services*, 2009 (Mobiquitous'09), S. 1–3

[Brusilovsky 1996] BRUSILOVSKY, P.: Methods and techniques of adaptive hypermedia. In: *User Modeling and User-Adapted Interaction* 6 (1996), Nr. 2-3, S. 87–129

[Brusilovsky 2001] BRUSILOVSKY, P.: Adaptive Hypermedia. In: *User Modeling and User-Adapted Interaction* 11 (2001), S. 87–110

[Burmester u. a. 2007] BURMESTER, M. ; HASSENZAHL, M. ; KOLLER, F.: Engineering attraktiver Produkte - AttrakDiff. In: *Interaktion mit komplexen Informationsräumen: Visualisierung, Multimodalität, Kooperation* (2007), S. 129–141

[Bylund u. Espinoza 2002] BYLUND, M. ; ESPINOZA, F.: Testing and Demonstrating Context-Aware Services with Quake III Arena. In: *Communications of the ACM* 45 (2002), Nr. 1, S. 46–48

[Cao u. a. 2006] CAO, F. ; ESTER, M. ; QIAN, W. ; ZHOU, A.: Density-Based Clustering over an Evolving Data Stream with Noise. In: *Proc. of the 6th SIAM Int. Conf. on Data Mining*, SIAM, 2006

[Chakravarthy u. a. 1994] CHAKRAVARTHY, S. ; KRISHNAPRASAD, V. ; ANWAR, E. ; KIM, S.-K.: Composite Events for Active Databases: Semantics, Contexts and Detection. In: *Proc. of the 20th Int. Conf. on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1994 (VLDB '94), S. 606–617

[Cheng u. a. 2008] CHENG, D. ; SONG, H. ; CHO, H. ; JEONG, S. ; KALASAPUR, S. ; MESSER, A.: Mobile Situation-Aware Task Recommendation Application. In: *2nd Int. Conf. on Next Generation Mobile Applications, Services and Technologies*, 2008, S. 228 –233

[Cheng u. a. 2009] CHENG, D. ; SONG, H. ; CHO, H. ; JEONG, S. ; KALASAPUR, S. ; MESSER, A.: Situation-aware User Interest Mining on Mobile Handheld Devices. In: *Proc. of the Workshop on Personalization in Mobile and Pervasive Computing*, 2009

[Coen 1998] COEN, M. H.: Design Principles for Intelligent Environments. In: *Proc. of the 15th nat. Conf. on Artificial Intelligence*. Menlo Park, CA, USA : American Association for Artificial Intelligence, 1998 (AAAI '98/IAAI '98), S. 547–554

[Cook u. a. 2003] COOK, D.J. ; YOUNGBLOOD, M. ; HEIERMAN, E.O. ; GOPALRATNAM, K. ; RAO, S. ; LITVIN, A. ; KHAWAJA, F.: MavHome: An Agent-Based Smart Home. In: *Proc. of the First IEEE Int. Conf. on Pervasive Computing and Communications*, 2003, S. 521 – 524

[Coutand 2008] COUTAND, O.: *A Framework for Contextual Personalised Applications*, Diss., 2008

[Dargie u. Poellabauer 2010] DARGIE, W. ; POELLABAUER, C.: *Fundamentals of Wireless Sensor Networks: Theory and Practice*. New York, NY, USA : John Wiley & Sons Inc., 2010

[Das u. a. 2002] DAS, S. K. ; COOK, D. J. ; BHATTACHARYA, A. ; HEIERMAN, E. O. ; LIN, T.-Y.: The Role of Prediction Algorithms in the MavHome Smart Home Architecture. In: *IEEE Wireless Communications* 9 (2002), S. 77–84

[Davison u. Hirsh 1998] DAVISON, B. D. ; HIRSH, H.: Predicting Sequences of User Actions, AAAI Press, 1998, S. 5–12

[Dey 2001] DEY, A. K.: Understanding and Using Context. In: *Personal Ubiquitous Computing Journal* 5 (2001), S. 4–7

[Dix u. a. 2004] DIX, A. ; FINLAY, J. E. ; ABOWD, G. D. ; BEALE, R.: *Human-Computer Interaction*. 3rd. Pearson Prentice Hall, 2004

[Doshi u. Trivedi 2011] DOSHI, A. ; TRIVEDI, M. M.: Tactical Driver Behavior Predcition and Intent Inference: A Review. In: *Proc. of 14th Int. IEEE Conf. on Intelligent Transportation Systems*, IEEE Publishing, 2011

[Duong u. a. 2006] DUONG, T.V. ; PHUNG, D.Q. ; BUI, H.H. ; VENKATESH, S.: Human Behavior Recognition with Generic Exponential Family Duration Modeling in the Hidden Semi-Markov Model. In: *18th Int. Conf. on Pattern Recognition* Bd. 3, 2006, S. 202 –207

[EsperTech Inc. 2010] ESPERTECH INC.: Complex Event Processing. (2010). – http://esper.codehaus.org/, Letzter Zugriff: 29.04.2013

[Ester u. a. 1996] ESTER, M. ; KRIEGEL, H.-P. ; SANDER, J. ; XU, X.: A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: *Proc. of 2nd Int. Conf. on Knowledge Discovery and Data Mining*, 1996, S. 226–231

[Etzion u. Niblett 2010] ETZION, O. ; NIBLETT, P.: *Event Processing in Action*. Manning Publications Co., 2010

[Flanagan 2003] FLANAGAN, J. A.: Unsupervised Clustering of Symbol Strings. In: *Proc. of the Int. Joint Conf. on Neural Networks* Bd. 4, 2003, S. 3250 – 3255

[Flanagan 2005] FLANAGAN, J. A.: Unsupervised Clustering of Context Data and Learning User Requirements for a Mobile Device. In: DEY, A. (Hrsg.) ; KOKINOV, B. (Hrsg.) ; LEAKE, D. (Hrsg.) ; TURNER, R. (Hrsg.): *Modeling and Using Context* Bd. 3554. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2005, S. 155–168

[Gajos u. Jameson 2009] GAJOS, K. ; JAMESON, A.: *New Paradigms for Adaptive Interaction*. Notes of Half-Day Tutorial at UMAP 2009, 2009

[Garofalakis u. a. 1999] GAROFALAKIS, M. N. ; RASTOGI, R. ; SHIM, K.: SPIRIT: Sequential Pattern Mining with Regular Expression Constraints. In: *Proc. of the 25th Int. Conf. on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999 (VLDB '99), S. 223–234

[Gatziu u. Dittrich 1994] GATZIU, S. ; DITTRICH, K.R.: Detecting Composite Events in Active Database Systems Using Petri Nets. In: *Proc. of the 4th Int. Workshop on Research Issues in Data Engineering*, 1994, S. 2 –9

[Gehani u. a. 1992] GEHANI, N. H. ; JAGADISH, H. V. ; SHMUELI, O.: Composite Event Specification in Active Databases: Model & Implementation. In: *Proc. of the 18th Int. Conf. on Very Large Data Bases*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1992 (VLDB '92), S. 327–338

[Greenberg 2001] GREENBERG, S.: Context as a Dynamic Construct. In: *Hum.-Comput. Interact.* 16 (2001), Nr. 2, S. 257–268

[Hansmann u. a. 2001] HANSMANN, U. ; MERK, L. ; NICKLOUS, M. S. ; STOBER, T.: *Pervasive Computing Handbook*. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2001

[Harel u. a. 1987] HAREL, D. ; PNUELI, A. ; SCHMIDT, J. P. ; SHERMAN, R.: On the Formal Semantics of Statecharts. In: *2nd IEEE Symp. on Logic in Computer Science*, 1987, S. 54–64

[Hart u. Stavenland 1988] HART, S. G. ; STAVENLAND, L. E.: Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In: HANCOCK, P. A. (Hrsg.) ; MESHKATI, N. (Hrsg.): *Human Mental Workload*. Elsevier, 1988, Kapitel 7, S. 139–183

[Hassenzahl 2003] HASSENZAHL, M.: The Thing and I: Understanding the Relationship Between User and Product. In: BLYTHE, M. A. (Hrsg.) ; OVERBEEKE, K. (Hrsg.) ; MONK, A. F. (Hrsg.) ; WRIGHT, P. C. (Hrsg.): *Funology* Bd. 3. Norwell, MA, USA : Kluwer Academic Publishers, 2003 (Human-Computer Interaction Series), S. 31–42

[Hassenzahl u. a. 2003] HASSENZAHL, M. ; BURMESTER, M. ; KOLLER, F.: AttrakDiff: Ein Fragebogen zur Messung wahrgenommener hedonischer und pragmatischer Qualität. In: SZWILLUS, G. (Hrsg.) ; ZIEGLER, J. (Hrsg.): *Mensch & Computer 2003: Interaktion in Bewegung*, Teubner, 2003, S. 187–196

[Hatscher 2001] HATSCHER, M.: Joy of use - Determinanten der Freude bei der Software-Nutzung. In: OBERQUELLE, H. (Hrsg.) ; OPPERMANN, R. (Hrsg.) ; KRAUSE, J. (Hrsg.): *Mensch & Computer*, B.G. Teubner, 2001, S. 445–446

[Hauser 2004] HAUSER, K.: *Entwicklung einer Methode und Pilotstudie zur Langzeitevaluation von adaptiven User Interface Elementen*, Fachhochschule Stuttgart - Hochschule der Medien, Diplomarbeit, 2004

[Hilbert u. Redmiles 2000] HILBERT, D. M. ; REDMILES, D. F.: Extracting Usability Information from User Interface Events. In: *ACM Computing Surveys* 32 (2000), S. 384–421

[Hritsevskyy 2012] HRITSEVSKYY, D.: *Entwurf und Implementierung eines Werkzeugs zum Testen von kontextsensitiven Mustererkennungssystemen*, Technische Universität Berlin, Diplomarbeit, 2012

[Huebscher u. McCann 2004] HUEBSCHER, M. C. ; MCCANN, J. A.: Simulation Model for Self-Adaptive Applications in Pervasive Computing. In: *Proc. of the 15th Int. Workshop on Database and Expert Systems Applications*. Washington, DC, USA : IEEE Computer Society, 2004 (DEXA '04), S. 694–698

[Hummel 2006] *Kapitel* 10. In: HUMMEL, B.: *Map Matching for Vehicle Guidance*. CRC Press, 2006 (Innovations in GIS)

[Indulska u. Sutton 2003] INDULSKA, J. ; SUTTON, P.: Location Management in Pervasive Systems. In: *Proc. of the Australian Information Security Workshop Conf. on ACSW Frontiers*. Darlinghurst, Australien : Australian Computer Society, Inc., 2003 (ACSW Frontiers '03), S. 143–151

[Iqbal u. a. 2011] IQBAL, S. T. ; HORVITZ, E. ; JU, Y.-C. ; MATHEWS, E.: Hang on a Sec! Effects of Proactive Mediation of Phone Conversations while Driving. In: *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. New York, NY, USA : ACM, 2011 (CHI '11), S. 463–472

[ISTAG 1999] ISTAG: *Orientations for Workprogramme 2000 and beyond*. 1999. – ftp://ftp.cordis.europa.eu/pub/ist/docs/istag-99-final.pdf, Letzter Zugriff:25.04.2013

[Jameson 2008] JAMESON, A.: Adaptive Interfaces and Agents. In: SEARS, A. (Hrsg.) ; JACKO, J. A. (Hrsg.): *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. 2nd. Boca Raton, FL : CRC Press, 2008, S. 433–458

[Jeong u. a. 2009] JEONG, S. ; CHENG, D. ; SONG, H. ; KALASAPUR, S.: Non-collaborative Interest Mining for Personal Devices. In: *Proc. of IEEE Symposium on Computational Intelligence and Data Mining*, IEEE, 2009, S. 179–186

[Kakousis u. a. 2010] KAKOUSIS, K. ; PASPALLIS, N. ; PAPADOPOULOS, G. A. ; RUIZ, P. A.: Testing self-adaptive applications with simulation of context events. In: *3rd Int. DisCoTec Workshop on Context-aware Adaptation Mechanisms for Pervasive and Ubiquitous Services*, 2010

[Kang u. a. 2004] KANG, J. H. ; WELBOURNE, W. ; STEWART, B. ; BORRIELLO, G.: Extracting Places from Traces of Locations. In: *Proceedings of the 2nd ACM Int. Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*. New York, NY, USA : ACM, 2004 (WMASH '04), S. 110–118

[Künzer u. a. 2004] KÜNZER, A. ; OHMANN, F. ; SCHMIDT, L.: Antizipative Modellierung des Benutzerverhaltens mit Hilfe von Aktionsvorhersage-Algorithmen. In: *MMI-Interaktiv* 7 (2004), S. 61–83

[Kobsa 1995] KOBSA, A.: Supporting User Interfaces for All Through User Modelling. In: *Proc. of the 6th Int. Conf. on Human-Computer Interaction*, 1995, S. 155–157

[Kofman u. Castro 2006] KOFMAN, E. ; CASTRO, R.: STDEVS. A Novel Formalism For Modeling and Simulation of Stochastic Discrete Event Systems. In: *Asociación Argentina de Control Automático*, 2006

[Korvemaker u. Greiner 2000] KORVEMAKER, B. ; GREINER, R.: Predicting UNIX Command Lines: Adjusting to User Patterns. In: *Proceedings of the 17th Nat. Conf. on Artificial Intelligence and 12th Conf. on Innovative Applications of Artificial Intelligence*, AAAI Press, 2000, S. 230–235

[Kosala u. Blockeel 2000] KOSALA, R. ; BLOCKEEL, H.: Web Mining Research: A Survey. In: *ACM SIGKDD Explorations Newsletter* 2 (2000), S. 1–15

[Krause u. a. 2006] KRAUSE, A. ; SMAILAGIC, A. ; SIEWIOREK, D.P.: Context-Aware Mobile Computing: Learning Context-Dependent Personal Preferences from a Wearable Sensor Array. In: *IEEE Transactions on Mobile Computing* 5 (2006), Nr. 2, S. 113 – 127

[Krumm 2008] KRUMM, J.: A Markov Model for Driver Turn Prediction. In: *SAE World Congress*, Springer-Verlag, 2008

[Krumm u. Horvitz 2005] KRUMM, J. ; HORVITZ, E.: The Microsoft Multiperson Location Survey / Microsoft Research (MSR-TR-2005-103). 2005. – Forschungsbericht

[Krumm u. Horvitz 2006] KRUMM, J. ; HORVITZ, E.: Predestination: Inferring Destinations from Partial Trajectories. In: *Proc. of the 8th Int. Conf. on Ubiquitous Computing*. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2006 (UbiComp'06), S. 243–260

[Laddaga 1997] LADDAGA, Robert: Self Adaptive Software SOL BAA 98-12. In: *DARPA Broad Agency Announcement (BAA-98-12)*, DARPA, 1997. – http://people.csail.mit.edu/rladdaga/BAA98-12excerpt.html, Letzer Zugriff: 2013-04-25

[Langley 1997] LANGLEY, P.: Machine Learning for Adaptive User Interfaces. In: *Proc. of the 21st Annual German Conf. on Artificial Intelligence: Advances in Artificial Intelligence*. Berlin/Heidelberg, Deutschland : Springer-Verlag, 1997 (KI '97), S. 53–62

[Lavie u. Meyer 2010] LAVIE, T. ; MEYER, J.: Benefits and costs of adaptive user interfaces. In: *Int. Journal Human-Compututer Studies* 68 (2010), Nr. 8, S. 508–524

[Lee u. a. 2012] LEE, J. D. ; ROBERTS, S. C. ; HOFFMAN, J. D. ; ANGELL, L. S.: Scrolling and Driving: How an MP3 Player and Its Aftermarket Controller Affect Driving Performance and Visual Behavior. In: *Human Factors: The Journal of the Human Factors and Ergonomics Society* (2012), S. 250–263

[Lei u. a. 2002] LEI, H. ; SOW, D. M. ; DAVIS II, J. S. ; BANAVAR, G. ; EBLING, M. R.: The Design and Applications of a Context Service. In: *SIGMOBILE Mobile Computing*

and Communications Review 6 (2002), Nr. 4, S. 45–55

[Luckham u. Vera 1995] LUCKHAM, D.C. ; VERA, J.: An Event-Based Architecture Definition Language. In: *IEEE Transactions on Software Engineering* 21 (1995), Nr. 9, S. 717 –734

[Ma u. a. 2005] MA, T. ; KIM, Y.-D. ; MA, Q. ; TANG, M. ; ZHOU, W.: Context-Aware Implementation based on CBR for Smart Home. In: *IEEE Int. Conf. on Wireless And Mobile Computing, Networking And Communications* Bd. 4, 2005, S. 112 – 115

[Maes 1994] MAES, P.: Agents that Reduce Work and Information Overload. In: *Communications of the ACM* 37 (1994), Nr. 7, S. 31–40

[Mak 2008] MAK, R. H.: *Design and Performance Analysis of Data-Independent Stream Processing Systems*, Diss., 2008

[Mannila u. a. 1997] MANNILA, H. ; TOIVONEN, H. ; VERKAMO, A. I.: Discovery of Frequent Episodes in Event Sequences. In: *Data Mining Knowledge Discovery* 1 (1997), S. 259–289

[Marmasse u. Schmandt 2002] MARMASSE, N. ; SCHMANDT, C.: A User-Centered Location Model. In: *Personal Ubiquitous Comput.* 6 (2002), Nr. 5-6, S. 318–321

[Martin u. Nurmi 2006] MARTIN, M. ; NURMI, P.: A Generic Large Scale Simulator for Ubiquitous Computing. In: *Proc. of 3rd Annual Int. Conf. on Mobile and Ubiquitous Systems*, 2006, S. 1 –3

[McGlinn u. a. 2010] MCGLINN, K. ; O'NEILL, E. ; GIBNEY, A. ; O'SULLIVAN, D. ; LEWIS, D.: SimCon: A Tool to Support Rapid Evaluation of Smart Building Application Design using Context Simulation and Virtual Reality. In: *Journal of Universal Computer Science* 16 (2010), Nr. 15, S. 1992–2018

[Mikalsen u. Kofod-Petersen 2005] MIKALSEN, M. ; KOFOD-PETERSEN, A.: Representing and Reasoning about Context in a Mobile Environment. In: *Revue d'Intelligence Artificielle* 19 (2005), S. 479–498

[Miyajima u. a. 2007] MIYAJIMA, C. ; NISHIWAKI, Y. ; OZAWA, K. ; WAKITA, T. ; ITOU, K. ; TAKEDA, K. ; ITAKURA, F.: Driver Modeling Based on Driving Behavior and Its Evaluation in Driver Identification. In: *Proc. of the IEEE* 95 (2007), Nr. 2, S. 427 –437

[Morla u. Davies 2004] MORLA, R. ; DAVIES, N.: Evaluating a Location-Based Application: A Hybrid Test and Simulation Environment. In: *IEEE Pervasive Computing* 3 (2004), Nr. 3, S. 48–56

[Nakashima u. a. 2009] NAKASHIMA, H. ; AGHAJAN, H. ; AUGUSTO, J. C.: *Handbook of Ambient Intelligence and Smart Environments*. 1st. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2009

[Nazari Shirehjini u. Klar 2005] NAZARI SHIREHJINI, A. A. ; KLAR, F.: 3DSim: Rapid Prototyping Ambient Intelligence. In: *Proc. of the Joint Conf. on Smart Objects and Ambient Intelligence*. New York, NY, USA : ACM, 2005 (sOc-EUSAI '05), S. 303–307

[Nazemi u. a. 2010] NAZEMI, K. ; STAB, C. ; FELLNER, D.W.: Interaction Analysis: An Algorithm for Interaction Prediction and Activity Recognition in Adaptive Systems. In: *IEEE Int. Conf. on Intelligent Computing and Intelligent Systems* Bd. 1, 2010, S. 607 –612

[Nguyen u. a. 2005] NGUYEN, N. T. ; PHUNG, D. Q. ; VENKATESH, S. ; BUI, H.: Learning and Detecting Activities from Movement Trajectories Using the Hierarchical Hidden Markov Models. In: *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*. Washington, DC, USA : IEEE Computer Society, 2005 (CVPR '05), S. 955–960

[Perkowitz u. Etzioni 1999] PERKOWITZ, M. ; ETZIONI, O.: Adaptive Web Sites: Conceptual Cluster Mining. In: *Proc. of the 16th Int. Joint Conf. on Artifical Intelligence* Bd. 1. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999, S. 264–269

[Persson u. a. 2002] PERSSON, P. ; ESPINOZA, F. ; FAGERBERG, P. ; SANDIN, A. ; CÖSTER, R.: GeoNotes: A Location-based Information System for Public Spaces. In: *Designing Information Spaces: The Social Navigation Approach*, Springer-Verlag, 2002 (Computer Supported Cooperative Work), S. 151–173

[Piechulla u. a. 2003] PIECHULLA, W. ; MAYSER, C. ; GEHRKE, H. ; KÖNIG, W.: Reducing drivers' mental workload by means of an adaptive man-machine interface. In: *Transportation Research Part F: Traffic Psychology and Behaviour* Bd. 6, Elsevier, 2003, S. 233–248

[Riener u. Ferscha 2008] RIENER, A. ; FERSCHA, A.: Supporting Implicit Human-to-Vehicle Interaction: Driver Identification from Sitting Postures. In: *Proc. of the 1st Annual Int. Symposium on Vehicular Computing Systems*. ACM Digital Library, 2008

[Rogers u. a. 2000] ROGERS, S. ; FIECHTER, C.-N. ; THOMPSON, C.: Adaptive User Interfaces for Automotive Environments. In: *Proc. of the IEEE Intelligent Vehicles Symposium*, 2000, S. 662 –667

[Sadeh u. a. 2005] SADEH, N. M. ; GANDON, F. L. ; KWON, O. B.: Ambient Intelligence: The MyCampus Experience / Carnegie Mellon University. 2005. – Forschungsbericht

[Salber u. a. 1999] SALBER, D. ; DEY, A. K. ; ABOWD, G. D.: The Context Toolkit: Aiding the Development of Context-Enabled Applications. In: *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*. New York, NY, USA : ACM, 1999 (CHI '99), S. 434–441

[Sander u. a. 1998] SANDER, J. ; ESTER, M. ; KRIEGEL, H.-P. ; XU, X.: Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and its Applications. In: *Data Mining Knowledge Discovery* 2 (1998), Nr. 2, S. 169–194

[Sanmugalingam u. Coulouris 2002] SANMUGALINGAM, K. ; COULOURIS, G.: A Generic Location Event Simulator. In: *Proc. of the 4th Int. Conf. on Ubiquitous Computing*. London, UK, UK : Springer-Verlag, 2002 (UbiComp '02), S. 308–315

[Schiaffino u. Amandi 2006] SCHIAFFINO, S. ; AMANDI, A.: Polite Personal Agents. In: *IEEE Intelligent Systems* 21 (2006), Nr. 1, S. 12 – 19

[Schmidt u. Laerhoven 2001] SCHMIDT, A. ; LAERHOVEN, K. V.: How to Build Smart Appliances? In: *IEEE Personal Communications* 8 (2001), Nr. 4, S. 6–11

[Schruben 1983] SCHRUBEN, L.: Simulation Modeling with Event Graphs. In: *Commun. ACM* 26 (1983), Nr. 11, S. 957–963

[Schulz u. a. 2000] SCHULZ, S. ; EWING, T.C. ; ROZENBLIT, J.W.: Discrete Event System Specification (DEVS) and StateMate StateCharts Equivalence for Embedded Systems Modeling. In: *Proc. of the 7th IEEE Int. Conf. and Workshops on the Engineering of Computer Based Systems*, 2000, S. 308 – 316

[Siewiorek u. a. 2003] SIEWIOREK, D. ; SMAILAGIC, A. ; FURUKAWA, J. ; MORAVEJI, N. ; REIGER, K. ; SHAFFER, J.: SenSay: A Context-Aware Mobile Phone. In: *Proc. of the 7th IEEE International Symposium on Wearable Computers*. Washington, DC, USA : IEEE Computer Society, 2003, S. 248 – 249

[Srikant u. Agrawal 1996] SRIKANT, R. ; AGRAWAL, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: APERS, Peter (Hrsg.) ; BOUZEGHOUB, Mokrane (Hrsg.) ; GARDARIN, Georges (Hrsg.): *Proc. of the 5th Int. Conf. on Extending Database Technology: Advances in Database Technology*. Berlin/Heidelberg, Deutschland : Springer-Verlag, 1996 (Lecture Notes in Computer Science), S. 3–17

[Tanca u. a. 2011] TANCA, L. ; BOLCHINI, C. ; QUINTARELLI, E. ; SCHREIBER, F. A. ; ORSI, G.: Problems and Opportunities in Context-Based Personalization. In: *Proc. of 5th Int. Workshop on Personalized Access, Profile Management, and Context Awareness in Databases*, 2011

[Tchankue u. a. 2011] TCHANKUE, P. ; WESSON, J. ; VOGTS, D.: The Impact of an Adaptive User Interface on Reducing Driver Distraction. In: *Proc. of the 3rd Int. Conf. on Automotive User Interfaces and Interactive Vehicular Applications*, 2011, S. 87–94

[Torkkola u. a. 2007] TORKKOLA, K. ; ZHANG, K. ; LI, H. ; ZHANG, H. ; SCHREINER, C. ; GARDNER, M.: Traffic Advisories Based on Route Prediction. In: *Workshop on Mobile Interaction with the Real World*, 2007

[Vetek u. a. 2009] VETEK, A. ; FLANAGAN, J. A. ; COLLEY, A. ; KERÄNEN, T.: SmartActions: Context-Aware Mobile Phone Shortcuts. In: *Proc. of the 12th Int. Conf. on Human-Computer Interaction – INTERACT 2009* Bd. 5726. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2009, S. 796–799

[Walzer u. a. 2007] WALZER, K. ; SCHILL, A. ; LÖSER, A.: Temporal Constraints for Rule-based Event Processing. In: *Proc. of the ACM first Ph.D. workshop in CIKM*. New York, NY, USA : ACM, 2007 (PIKM '07), S. 93–100

[Weiser 1991] WEISER, M.: The Computer for the 21st Century. In: *Scientific American* (1991), S. 94–104

[Weld u. a. 2003] WELD, D. S. ; ANDERSON, C. ; DOMINGOS, P. ; ETZIONI, O. ; GAJOS, K. ; LAU, T. ; WOLF, S.: Automatically Personalizing User Interfaces. In: *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence*. San Francisco, CA, USA : Morgan Kaufmann

Publishers Inc., 2003, S. 1613–1619

[Woolridge 2009] WOOLRIDGE, M.: *Introduction to Multiagent Systems*. New York, NY, USA : John Wiley & Sons, Inc., 2009

[Zeigler 1976] ZEIGLER, B. P.: *Theory of Modeling and Simulation*. John Wiley, 1976

[Zhou u. a. 2007] ZHOU, C. ; FRANKOWSKI, D. ; LUDFORD, P. ; SHEKHAR, S. ; TERVEEN, L.: Discovering Personally Meaningful Places: An Interactive Clustering Approach. In: *ACM Transactions on Information Systems* 25 (2007)

[Zhu u. Sethi 2001] ZHU, D. ; SETHI, A. S.: SEL, A New Event Pattern Specification Language for Event Correlation. In: *Proc. of 10th Int. Conf. on Computer Communications and Networks*, IEEE, 2001, S. 586–589

[Ziebart u. a. 2008] ZIEBART, B. D. ; MAAS, A. L. ; DEY, A. K. ; BAGNELL, J. A.: Navigate Like a Cabbie: Probabilistic Reasoning from Observed Context-Aware Behavior. In: *Proc. of the 10th Int. Conf. on Ubiquitous Computing*. New York, NY, USA : ACM, 2008 (UbiComp'08), S. 322–331

[Zimmer u. a. 2011] ZIMMER, B. ; BARRETT, G. ; METCALF, A.: *"App" 2010 Word of the Year, as voted by American Dialect Society*. : American Dialect Society, 2011

[Zimmermann 2003] ZIMMERMANN, A.: Context-Awareness in User Modelling: Requirements Analysis for a Case-Based Reasoning Application. In: *Proc. of the 5th Int. Conf. on Case-Based Reasoning*. Berlin/Heidelberg, Deutschland : Springer-Verlag, 2003 (Lecture Notes in Artificial Intelligence 2689), S. 718 – 732

[Zukerman u. Albrecht 2001] ZUKERMAN, I. ; ALBRECHT, D. W.: Predictive Statistical Models for User Modeling. In: *User Modeling and User-Adapted Interaction* 11 (2001), S. 5–18

Anhang A
Systemdienst

A.1 Anwendungsfallabhängige Instanziierung der Situationsnachrichten

```
<Events xmlns="de.dcaiti.hmi.personalization.contextmanager.events
    .xml" root="Event" timeSource="ts">

  <Event name="Event">
    <Property name="ts" type="long"/>
  </Event>

  <Event name="TimeEvent" base="Event"/>

  <Event name="Location" base="Event">
    <Property name="latitude" type="double"/>
    <Property name="longitude" type="double"/>
    <Property name="direction" type="double"/>
  </Event>

  <Event name="EngineStateChanged" base="Event">
    <Property name="running" type="boolean"/>
  </Event>

  <Event name="FuelLevelChanged" base="Event">
    <Property name="newLevel" type="double"/>
  </Event>

  <Event name="RadioStationChanged" base="Event">
    <Property name="stationFrequency" type="String"/>
    <Property name="stationName" type="String"/>
  </Event>

  <Event name="RadioStatusChanged" base="Event">
    <Property name="status" type="String"/>
  </Event>

  <Event name="BrowserAddressChanged" base="Event">
    <Property name="newAddress" type="String"/>
  </Event>
```

```
  <Event name="NavigationStarted" base="Event">
    <Property name="country" type="String"/>
    <Property name="city" type="String"/>
    <Property name="street" type="String"/>
    <Property name="streetNr" type="String"/>
    <Property name="latitude" type="double"/>
    <Property name="longitude" type="double"/>
  </Event>

  <Event name="CurrentPositionMap" base="Event"/>

</Events>
```

A.2 Schematische Spezifikation der Situations- und Dienstnachrichten

```
<xsd:schema xmlns="de.dcaiti.hmi.personalization.manager.hmi.
    message" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    attributeFormDefault="unqualified" elementFormDefault="
    qualified" targetNamespace="de.dcaiti.hmi.personalization.
    manager.hmi.message">

  <xsd:element name="HMIMessage" type="MSGType"/>

  <xsd:complexType name="MSGType">
    <xsd:choice>
      <xsd:element name="ListQuery" type="MSGListQueryType"/>
      <xsd:element name="ListResult" type="MSGListResultType"/>
      <xsd:element name="UseCaseQuery" type="MSGUseCaseQueryType"/
          >
      <xsd:element name="UseCaseResult" type="MSGUseCaseResultType
          "/>
      <xsd:element name="CaseQuery" type="MSGCaseQueryType"/>
      <xsd:element name="CaseResult" type="MSGCaseResultType"/>
      <xsd:element name="GeometryQuery" type="MSGGeometryQueryType
          "/>
      <xsd:element name="GeometryQueryResult" type="
          MSGGeometryQueryResultType"/>
      <xsd:element name="UseCaseModeChange" type="
          MSGUseCaseModeChangeType"/>
      <xsd:element name="UpgradeUseCase" type="
          MSGUpgradeUseCaseType"/>
      <xsd:element name="DowngradeUseCase" type="
          MSGDowngradeUseCaseType"/>
      <xsd:element name="UpdateViewed" type="MSGUpdateViewedType"/
          >
      <xsd:element name="NewRecommendation" type="
          MSGNewRecommendationType"/>
```

```
      <xsd:element name="NewDiscovery" type="MSGNewDiscoveryType"/
         >
      <xsd:element name="CaseNotification" type="
         MSGCaseNotificationType"/>
      <xsd:element name="Reset" type="xsd:string"/>
      <xsd:element name="ResetConfirmation" type="
         MSGResetConfirmationType"/>
      <xsd:element name="LoggedEvent" type="MSGLoggedEventType"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="MSGResetConfirmationType">
    <xsd:attribute name="success" type="xsd:boolean" use="required
       "/>
  </xsd:complexType>

  <xsd:complexType name="MSGNewRecommendationType">
  </xsd:complexType>

  <xsd:complexType name="MSGNewDiscoveryType">
    <xsd:attribute name="case" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGLoggedEventType">
    <xsd:sequence minOccurs="0">
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="
         Property" type="MSGEventPropertyType"/>
    </xsd:sequence>
    <xsd:attribute name="user" type="xsd:string" use="optional"/>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGEventPropertyType">
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="value" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGListQueryType">
    <xsd:attribute name="listType" type="xsd:string" use="required
       "/>
  </xsd:complexType>

  <xsd:complexType name="MSGListResultType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="
         UseCase" type="MSGUseCaseSignatureListType"/>
    </xsd:sequence>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGUseCaseSignatureListType">
    <xsd:sequence>
```

```
      <xsd:element maxOccurs="1" minOccurs="0" name="Properties"
         type="MSGPropertiesType"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:int" use="required"/>
    <xsd:attribute name="active" type="xsd:boolean" use="required"
       />
    <xsd:attribute name="update" type="xsd:boolean" use="required"
       />
    <xsd:attribute name="type" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGPropertiesType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="
         Property" type="MSGPropertyType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGPropertyType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" type="xsd:string" use="required
           "/>
        <xsd:attribute name="type" type="xsd:string" use="required
           "/>
        <xsd:attribute name="show" type="xsd:boolean" use="
           required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="MSGUseCaseQueryType">
    <xsd:attribute name="id" type="xsd:int" use="required"/>
    <xsd:attribute name="parameter" type="xsd:boolean" use="
       required"/>
    <xsd:attribute name="situation" type="xsd:boolean" use="
       required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGUseCaseResultType">
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="1" name="UseCase" type
         ="MSGUseCaseType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGCaseQueryType">
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGCaseResultType">
    <xsd:sequence>
```

```
      <xsd:element maxOccurs="unbounded" minOccurs="0" name="Case"
          type="MSGCaseType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGCaseType">
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="0" name="Properties"
         type="MSGPropertiesType"/>
      <xsd:element maxOccurs="1" minOccurs="0" name="Situation"
         type="MSGSituationType"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:int" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGUseCaseType">
    <xsd:sequence>
      <xsd:element maxOccurs="1" minOccurs="0" name="Properties"
         type="MSGPropertiesType"/>
      <xsd:element maxOccurs="1" minOccurs="0" name="Situation"
         type="MSGSituationType"/>
    </xsd:sequence>
    <xsd:attribute name="id" type="xsd:int" use="required"/>
    <xsd:attribute name="group" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGSituationType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="
         Factor" type="MSGFactorType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGFactorType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="
         Fragment" type="MSGFragmentType"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGFragmentType">
    <xsd:choice>
      <xsd:element name="FragmentNominal" type="xsd:string"/>
      <xsd:element name="FragmentInterval" type="
         MSGFragmentIntervalType"/>
    </xsd:choice>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGFragmentIntervalType">
    <xsd:sequence>
```

```
      <xsd:element maxOccurs="1" minOccurs="1" name="From" type="
         xsd:string"/>
      <xsd:element maxOccurs="1" minOccurs="1" name="To" type="xsd
         :string"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGGeometryQueryType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="
         Geometry" type="MSGGeometryType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGGeometryType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="id" type="xsd:int" use="required"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="MSGGeometryQueryResultType">
    <xsd:sequence>
      <xsd:element maxOccurs="unbounded" minOccurs="1" name="
         Geometry" type="MSGGeometryType"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="MSGUseCaseModeChangeType">
    <xsd:attribute name="id" type="xsd:int" use="required"/>
    <xsd:attribute name="mode" type="xsd:boolean" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGUpgradeUseCaseType">
    <xsd:attribute name="id" type="xsd:int" use="required"/>
    <xsd:attribute name="mode" type="xsd:boolean" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGDowngradeUseCaseType">
    <xsd:attribute name="id" type="xsd:int" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGUpdateViewedType">
    <xsd:attribute name="id" type="xsd:int" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="MSGCaseNotificationType">
    <xsd:attribute name="id" type="xsd:int" use="required"/>
  </xsd:complexType>

</xsd:schema>
```

A.3 UML-Sequenzdiagramm der Kommunikation zwischen dem HMI-Prototyp und dem Systemdienst



A.4 Schematische Spezifikation der Vorlagen

Hauptschema der Vorlagendatei:

```
<xsd:schema xmlns="de.dcaiti.hmi.personalization.usecases"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="de.dcaiti.hmi.personalization.usecases"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

  <xsd:include schemaLocation="EventPattern.xsd"/>

  <xsd:element name="Configuration" type="ConfigurationType"/>

  <xsd:complexType name="ConfigurationType">
    <xsd:all>
      <xsd:element name="ContextSignature" type="
         ContextSignatureType"/>
      <xsd:element name="Discovery" type="DiscoveryType"/>
      <xsd:element name="Detection" type="DetectionType"/>
    </xsd:all>
  </xsd:complexType>

  <xsd:complexType name="ContextSignatureType">
    <xsd:sequence>
      <xsd:element name="Attribute" type="AttributeType" minOccurs
         ="0" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="AttributeType">
    <xsd:sequence>
      <xsd:element name="Input" type="InputType" minOccurs="1"
         maxOccurs="unbounded"/>
    </xsd:sequence>
    <xsd:attribute name="name" type="xsd:string" use="required"/>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="InputType">
    <xsd:sequence>
      <xsd:element name="EventAssignment" type="
         EventAssignmentType" minOccurs="1" maxOccurs="unbounded"/
         >
    </xsd:sequence>
    <xsd:attribute name="event" type="xsd:string" use="required"/>
  </xsd:complexType>

  <xsd:complexType name="EventAssignmentType">
    <xsd:attribute name="source" type="xsd:string" use="required"/
       >
    <xsd:attribute name="target" type="xsd:string"/>
```

```
</xsd:complexType>

<xsd:complexType name="DiscoveryType">
  <xsd:sequence>
    <xsd:element name="Case" type="CaseType" minOccurs="0"
       maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="CaseType">
  <xsd:sequence>
    <xsd:element name="Action" type="ActionType"/>
    <xsd:element name="Properties" type="PropertiesType"
       minOccurs="0" maxOccurs="1"/>
    <xsd:element name="Context" type="ContextType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="ActionType">
  <xsd:sequence>
    <xsd:element name="Signature" type="SignatureType"/>
    <xsd:element name="Pattern" type="ActionPatternType"
       minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SignatureType">
  <xsd:sequence>
    <xsd:element name="Event" type="EventSignatureType"
       minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="EventSignatureType">
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="type" type="xsd:string" use="required"/>
</xsd:complexType>

<xsd:complexType name="PropertiesType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="Property" type="PropertyType"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="PropertyType">
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="event" type="xsd:string" use="required"/>
  <xsd:attribute name="eventProperty" type="xsd:string" use="
     required"/>
</xsd:complexType>
```

```
<xsd:complexType name="ContextType">
  <xsd:sequence>
    <xsd:element name="Scope" type="ScopeType" minOccurs="0"
       maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="groupingInterval" type="xsd:int" use="
     optional"/>
  <xsd:attribute name="keepingInterval" type="xsd:int" use="
     optional"/>
</xsd:complexType>

<xsd:complexType name="ScopeType">
  <xsd:sequence>
    <xsd:element name="Attribute" type="ScopeAttributeType"
       minOccurs="1" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="event" type="xsd:string"/>
  <xsd:attribute name="minCases" type="xsd:int"/>
</xsd:complexType>

<xsd:complexType name="ScopeAttributeType">
  <xsd:sequence>
    <xsd:element name="Similarity" type="SimilarityType"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string"/>
  <xsd:attribute name="type" type="xsd:string"/>
  <xsd:attribute name="feature" type="xsd:string"/>
  <xsd:attribute name="result" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="SimilarityType">
  <xsd:attribute name="epsilonDistance" type="xsd:double"/>
  <xsd:attribute name="distanceFunction" type="xsd:string"/>
</xsd:complexType>

<xsd:complexType name="DetectionType">
  <xsd:sequence>
    <xsd:element name="Situation" type="SituationType" minOccurs
       ="0" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="SituationType">
  <xsd:sequence>
    <xsd:element name="Trigger" type="TriggerType" minOccurs="0"
       maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="id" type="xsd:string" use="required"/>
  <xsd:attribute name="feedback" type="xsd:boolean" use="
     required"/>
```

```
    <xsd:attribute name="caseGroup" type="xsd:string" use="
       required"/>
  </xsd:complexType>

</xsd:schema>
```

Schema bzgl. der sequentiellen Interaktions- und Situationsmuster:

```
<xsd:schema xmlns="de.dcaiti.hmi.personalization.usecases"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    targetNamespace="de.dcaiti.hmi.personalization.usecases"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">

  <xsd:complexType name="ActionPatternType">
    <xsd:choice maxOccurs="1">
      <xsd:element name="Event" type="EventType"/>
      <xsd:element name="ContextChanged" type="ContextChangedType"
         />
      <xsd:element name="ContextEntry" type="ContextEntryExitType"
         />
      <xsd:element name="ContextExit" type="ContextEntryExitType"/
         >
      <xsd:element name="FollowedBy" type="UnboundedOperatorType"/
         >
      <xsd:element name="And" type="UnboundedOperatorType"/>
      <xsd:element name="Or" type="UnboundedOperatorType"/>
      <xsd:element name="Every" type="UnaryOperatorType"/>
      <xsd:element name="Not" type="UnaryOperatorType"/>
      <xsd:element name="TimerInterval" type="FunctionOperatorType
         "/>
      <xsd:element name="TimerWithin" type="FunctionOperatorType"/
         >
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="TriggerType">
    <xsd:complexContent>
      <xsd:extension base="ActionPatternType">
        <xsd:attribute name="name" type="xsd:string" use="required
           "/>
      </xsd:extension>
    </xsd:complexContent>
  </xsd:complexType>

  <xsd:complexType name="ContextEntryExitType">
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="Equal" type="CompareOperatorType"/>
      <xsd:element name="NotEqual" type="CompareOperatorType"/>
      <xsd:element name="UserDefinedExpression" type="
         UserDefinedOperatorType"/>
```

```
      <xsd:element name="Not" type="LogicalUnaryOperatorType"/>
      <xsd:element name="InContext" type="InContextType"/>
      <xsd:element name="And" type="LogicalUnboundedOperatorType"/
         >
      <xsd:element name="Or" type="LogicalUnboundedOperatorType"/>
    </xsd:choice>
    <xsd:attribute name="scope" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="FunctionOperatorType">
    <xsd:choice minOccurs="1" maxOccurs="unbounded">
      <xsd:element name="Constant" type="ConstantType"/>
      <xsd:element name="Event" type="EventType"/>
      <xsd:element name="ContextChanged" type="ContextChangedType"
         />
      <xsd:element name="ContextEntry" type="ContextEntryExitType"
         />
      <xsd:element name="ContextExit" type="ContextEntryExitType"/
         >
      <xsd:element name="FollowedBy" type="UnboundedOperatorType"/
         >
      <xsd:element name="And" type="UnboundedOperatorType"/>
      <xsd:element name="Or" type="UnboundedOperatorType"/>
      <xsd:element name="Every" type="UnaryOperatorType"/>
      <xsd:element name="Not" type="UnaryOperatorType"/>
      <xsd:element name="TimerInterval" type="FunctionOperatorType
         "/>
      <xsd:element name="TimerWithin" type="FunctionOperatorType"/
         >
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="ConstantType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="type" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="EventPropertyType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>

  <xsd:complexType name="ContextPropertyType">
    <xsd:attribute name="attribute" type="xsd:string" use="
       required"/>
  </xsd:complexType>
```

```
<xsd:complexType name="CaseGroupPropertyType">
  <xsd:attribute name="property" type="xsd:string" use="required
     "/>
</xsd:complexType>

<xsd:complexType name="UnaryOperatorType">
  <xsd:choice maxOccurs="1">
    <xsd:element name="Event" type="EventType"/>
    <xsd:element name="ContextChanged" type="ContextChangedType"
       />
    <xsd:element name="ContextEntry" type="ContextEntryExitType"
       />
    <xsd:element name="ContextExit" type="ContextEntryExitType"/
       >
    <xsd:element name="Context" type="ContextChangedType"/>
    <xsd:element name="FollowedBy" type="UnboundedOperatorType"/
       >
    <xsd:element name="And" type="UnboundedOperatorType"/>
    <xsd:element name="Or" type="UnboundedOperatorType"/>
    <xsd:element name="Every" type="UnaryOperatorType"/>
    <xsd:element name="Not" type="UnaryOperatorType"/>
    <xsd:element name="TimerInterval" type="FunctionOperatorType
       "/>
    <xsd:element name="TimerWithin" type="FunctionOperatorType"/
       >
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="BinaryOperatorType">
  <xsd:choice maxOccurs="2">
    <xsd:element name="Event" type="EventType"/>
    <xsd:element name="ContextChanged" type="ContextChangedType"
       />
    <xsd:element name="ContextEntry" type="ContextEntryExitType"
       />
    <xsd:element name="ContextExit" type="ContextEntryExitType"/
       >
    <xsd:element name="FollowedBy" type="UnboundedOperatorType"/
       >
    <xsd:element name="And" type="UnboundedOperatorType"/>
    <xsd:element name="Or" type="UnboundedOperatorType"/>
    <xsd:element name="Every" type="UnaryOperatorType"/>
    <xsd:element name="Not" type="UnaryOperatorType"/>
    <xsd:element name="TimerInterval" type="FunctionOperatorType
       "/>
    <xsd:element name="TimerWithin" type="FunctionOperatorType"/
       >
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="UnboundedOperatorType">
  <xsd:choice minOccurs="2" maxOccurs="unbounded">
```

```
      <xsd:element name="Event" type="EventType"/>
      <xsd:element name="ContextChanged" type="ContextChangedType"
         />
      <xsd:element name="ContextEntry" type="ContextEntryExitType"
         />
      <xsd:element name="ContextExit" type="ContextEntryExitType"/
         >
      <xsd:element name="FollowedBy" type="UnboundedOperatorType"/
         >
      <xsd:element name="And" type="UnboundedOperatorType"/>
      <xsd:element name="Or" type="UnboundedOperatorType"/>
      <xsd:element name="Every" type="UnaryOperatorType"/>
      <xsd:element name="Not" type="UnaryOperatorType"/>
      <xsd:element name="TimerInterval" type="FunctionOperatorType
         "/>
      <xsd:element name="TimerWithin" type="FunctionOperatorType"/
         >
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="EventType">
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="Equal" type="CompareOperatorType"/>
      <xsd:element name="NotEqual" type="CompareOperatorType"/>
      <xsd:element name="UserDefinedExpression" type="
         UserDefinedOperatorType"/>
      <xsd:element name="Not" type="LogicalUnaryOperatorType"/>
      <xsd:element name="InContext" type="InContextType"/>
      <xsd:element name="And" type="LogicalUnboundedOperatorType"/
         >
      <xsd:element name="Or" type="LogicalUnboundedOperatorType"/>
    </xsd:choice>
    <xsd:attribute name="type" type="xsd:string" use="required"/>
    <xsd:attribute name="id" type="xsd:string" use="optional"/>
  </xsd:complexType>

  <xsd:complexType name="ContextChangedType">
    <xsd:choice minOccurs="0" maxOccurs="1">
      <xsd:element name="Equal" type="CompareOperatorType"/>
      <xsd:element name="NotEqual" type="CompareOperatorType"/>
      <xsd:element name="UserDefinedExpression" type="
         UserDefinedOperatorType"/>
      <xsd:element name="Not" type="LogicalUnaryOperatorType"/>
      <xsd:element name="InContext" type="InContextType"/>
      <xsd:element name="And" type="LogicalUnboundedOperatorType"/
         >
      <xsd:element name="Or" type="LogicalUnboundedOperatorType"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="InContextType">
    <xsd:attribute name="scope" type="xsd:string"/>
```

```
  </xsd:complexType>

  <xsd:complexType name="LogicalUnaryOperatorType">
    <xsd:choice maxOccurs="1">
      <xsd:element name="Equal" type="CompareOperatorType"/>
      <xsd:element name="NotEqual" type="CompareOperatorType"/>
      <xsd:element name="UserDefinedExpression" type="
          UserDefinedOperatorType"/>
      <xsd:element name="Not" type="LogicalUnaryOperatorType"/>
      <xsd:element name="InContext" type="InContextType"/>
      <xsd:element name="And" type="LogicalUnboundedOperatorType"/
          >
      <xsd:element name="Or" type="LogicalUnboundedOperatorType"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="LogicalUnboundedOperatorType">
    <xsd:choice maxOccurs="unbounded">
      <xsd:element name="Equal" type="CompareOperatorType"/>
      <xsd:element name="NotEqual" type="CompareOperatorType"/>
      <xsd:element name="UserDefinedExpression" type="
          UserDefinedOperatorType"/>
      <xsd:element name="Not" type="LogicalUnaryOperatorType"/>
      <xsd:element name="InContext" type="InContextType"/>
      <xsd:element name="And" type="LogicalUnboundedOperatorType"/
          >
      <xsd:element name="Or" type="LogicalUnboundedOperatorType"/>
    </xsd:choice>
  </xsd:complexType>

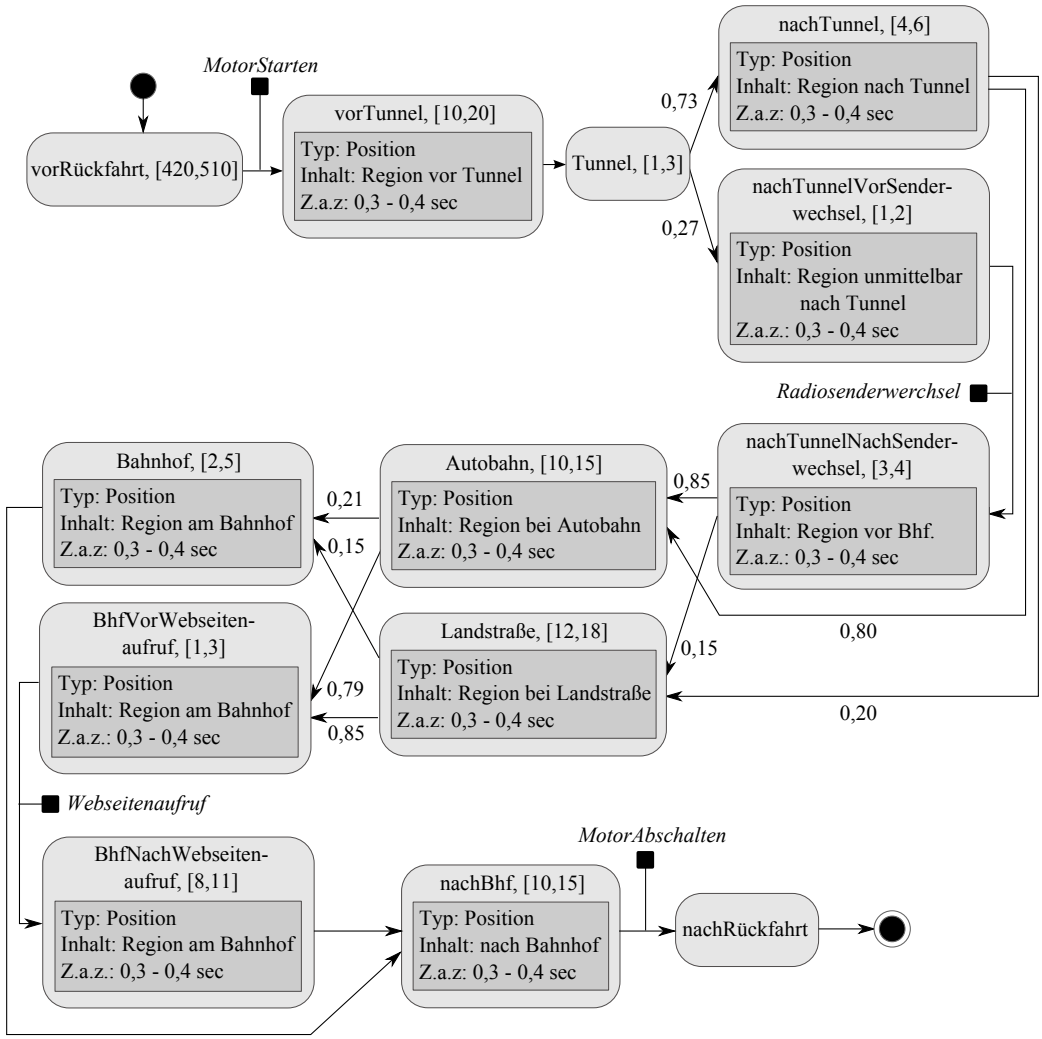
  <xsd:complexType name="CompareOperatorType">
    <xsd:choice minOccurs="2" maxOccurs="2">
      <xsd:element name="EventProperty" type="EventPropertyType"/>
      <xsd:element name="CaseGroupProperty" type="
          CaseGroupPropertyType"/>
      <xsd:element name="Context" type="ContextPropertyType"/>
      <xsd:element name="Constant" type="ConstantType"/>
    </xsd:choice>
  </xsd:complexType>

  <xsd:complexType name="UserDefinedOperatorType">
    <xsd:choice minOccurs="0" maxOccurs="unbounded">
      <xsd:element name="CaseGroupProperty" type="
          CaseGroupPropertyType"/>
      <xsd:element name="Context" type="ContextPropertyType"/>
      <xsd:element name="Constant" type="ConstantType"/>
    </xsd:choice>
    <xsd:attribute name="postprocess" type="xsd:string"/>
  </xsd:complexType>

</xsd:schema>
```

Anhang B
Simulation

B.1 Szenarienspezifisches Modell der Rückfahrt



Anhang C
Benutzerevaluation

C.1 Fragebogen: Einführung

Fragebogen: Einführung *Teilnehmernr.:_____*

Altersgruppe ☐ 20 bis 29 ☐ 30 bis 39 ☐ 40 bis 49 ☐ ab 50

Geschlecht ☐ weiblich ☐ männlich

Höchster Ausbildungsabschluss

☐ Schulabschluss (Abitur, Realschulabschluss, ...)
☐ Ausbildungsberuf, Volontariat
☐ Weiterführende Ausbildung (Meister, Techniker, ...)
☐ Hochschulstudium
☐ Weiterführende akademische Ausbildung (Promotion, PhD-Programm, Habilitation, ...)
☐ Andere/Sonstige

Derzeitige Tätigkeit

☐ Schüler
☐ Auszubildender
☐ Student
☐ Angestellter
☐ Unternehmer
☐ Arbeitslos

Besitzen Sie ein Auto? ☐ Ja ☐ Nein

Nutzen Sie ein Auto?

☐ Ja, mehrmals wöchentlich
☐ Ja, mehrmals monatlich
☐ Selten

*Teilnehmernr.:*_____

Welche Funktionen des Entertainment Systems im Auto nutzen Sie regelmäßig?

[Mehrfachauswahl möglich]

- ☐ Radio
- ☐ Musik(CD, MP3, …)
- ☐ Navigationssystem
- ☐ Internet
- ☐ Telefon
- ☐ Bordcomputer (Klimaanlage, Sitzheizung, …)

Würden Sie sich wünschen, dass das Entertainment System im Auto sich Ihren individuellen Benutzungsgewohnheiten anpasst?

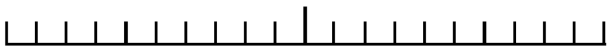
[Mehrfachauswahl möglich]

- ☐ Ja, sollte sich automatisch anpassen…
  - ☐ … auch ohne mein Zutun.
  - ☐ … aber nur wenn ich über die vorgenommenen Anpassungen informiert werde.
  - ☐ … aber nur mit meiner ausdrücklichen Zustimmung
- ☐ Ja, durch die Möglichkeit, selbst Anpassungen vornehmen zu können.
- ☐ Nein, ich bin zufrieden mit der Standardversion.

C.2 Fragebogen: Laufend

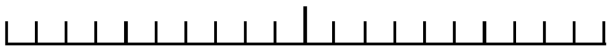
Fragebogen: Laufend *Task:____* *Teilnehmernr.:_____*

Geistige Anforderung:
Wie viel geistige Anstrengung war für die Aufgabe erforderlich?



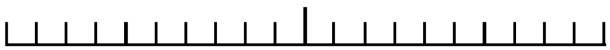
Sehr niedrig Sehr hoch

Physische Anforderung:
Wie hoch war die erforderliche körperliche Aktivität?



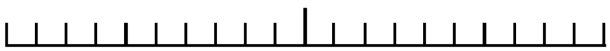
Sehr niedrig Sehr hoch

Zeitdruck:
Wie viel Zeitdruck empfanden Sie während der Aufgabenbearbeitung?



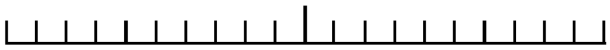
Sehr niedrig Sehr hoch

Erfolg:
Wie erfolgreich haben Sie Ihrer Meinung nach die Aufgabe bewältigt?



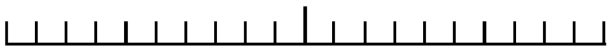
Perfekt Gescheitert

Aufwand:
Wie hart mussten Sie arbeiten, um die Aufgabenstellung zu erfüllen?



Sehr niedrig Sehr hoch

Frustration:
Waren Sie unsicher, irritiert, gestresst oder frustriert?



Sehr niedrig Sehr hoch

C.3 Fragebogen: Abschluss

Fragebogen: Abschluss *Teilnehmernr.:_____*

Würden Sie sich wünschen, dass das Entertainment System im Auto sich Ihren individuellen Benutzungsgewohnheiten anpasst?

[Mehrfachauswahl möglich]

- ☐ Ja, sollte sich automatisch anpassen…
  - ☐ … auch ohne mein Zutun.
  - ☐ … aber nur wenn ich über die vorgenommenen Anpassungen informiert werde.
  - ☐ … aber nur mit meiner ausdrücklichen Zustimmung
- ☐ Ja, durch die Möglichkeit, selbst Anpassungen vornehmen zu können.
- ☐ Nein, ich bin zufrieden mit der Standardversion.

Mit der personalisierten Schnellstartleiste kann ich die Aufgaben effizient erledigen	Völlig Falsch	☐	☐	☐	☐	☐	Völlig richtig
Mit den personalisierten Automatisierungen kann ich die Aufgaben effizient erledigen	Völlig Falsch	☐	☐	☐	☐	☐	Völlig richtig

Teilnehmernr.:_____

Die nun folgenden Fragen beziehen sich auf Anpassungen des Systems in Form der personalisierten Schnellstartleiste:

Die Anpassungen haben mich…

Unterstützt	☐	☐	☐	☐	☐	Behindert

Die Darstellung/Veranschaulichung der Anpassungen empfand ich als…

Angenehm	☐	☐	☐	☐	☐	Störend

Aufgrund der Anpassungen sehe ich meine Privatsphäre durch das System…

Gewahrt	☐	☐	☐	☐	☐	Gefährdet

Das Systemverhalten empfinde ich als…

Zuverlässig	☐	☐	☐	☐	☐	Unzuverlässig

Meine Bedürfnisse werden vom System…

Verstanden	☐	☐	☐	☐	☐	Nicht Verstanden

Ich fühle mich dem System gegenüber…

In Kontrolle	☐	☐	☐	☐	☐	Ausgeliefert

Teilnehmernr.:_____

Insgesamt beurteile ich die Anpassungen des Systems in Form der personalisierbaren Schnellstartleiste…

menschlich	☐	☐	☐	☐	☐	☐	☐	technisch
isolierend	☐	☐	☐	☐	☐	☐	☐	verbindend
angenehm	☐	☐	☐	☐	☐	☐	☐	unangenehm
originell	☐	☐	☐	☐	☐	☐	☐	konventionell
einfach	☐	☐	☐	☐	☐	☐	☐	kompliziert
fachmännisch	☐	☐	☐	☐	☐	☐	☐	laienhaft
hässlich	☐	☐	☐	☐	☐	☐	☐	schön
praktisch	☐	☐	☐	☐	☐	☐	☐	unpraktisch
sympathisch	☐	☐	☐	☐	☐	☐	☐	unsympathisch
umständlich	☐	☐	☐	☐	☐	☐	☐	direkt
stilvoll	☐	☐	☐	☐	☐	☐	☐	stillos
voraussagbar	☐	☐	☐	☐	☐	☐	☐	unberechenbar
minderwertig	☐	☐	☐	☐	☐	☐	☐	wertvoll
ausgrenzend	☐	☐	☐	☐	☐	☐	☐	einbeziehend
bringt mich den Leuten näher	☐	☐	☐	☐	☐	☐	☐	trennt mich von Leuten
nicht vorzeigbar	☐	☐	☐	☐	☐	☐	☐	vorzeigbar
zurückweisend	☐	☐	☐	☐	☐	☐	☐	einladend
phantasielos	☐	☐	☐	☐	☐	☐	☐	kreativ
gut	☐	☐	☐	☐	☐	☐	☐	schlecht
verwirrend	☐	☐	☐	☐	☐	☐	☐	übersichtlich
abstoßend	☐	☐	☐	☐	☐	☐	☐	anziehend
mutig	☐	☐	☐	☐	☐	☐	☐	vorsichtig
innovativ	☐	☐	☐	☐	☐	☐	☐	konservativ
lahm	☐	☐	☐	☐	☐	☐	☐	fesselnd
harmlos	☐	☐	☐	☐	☐	☐	☐	herausfordernd
motivierend	☐	☐	☐	☐	☐	☐	☐	entmutigend
neuartig	☐	☐	☐	☐	☐	☐	☐	herkömmlich
widerspenstig	☐	☐	☐	☐	☐	☐	☐	handhabbar

Teilnehmernr.:_____

Die nun folgenden Fragen beziehen sich auf die personalisierten Automatisierungen:

Die Anpassungen haben mich…

Unterstützt	☐	☐	☐	☐	☐	Behindert

Die Darstellung/Veranschaulichung der Anpassungen empfand ich als…

Angenehm	☐	☐	☐	☐	☐	Störend

Aufgrund der Anpassungen sehe ich meine Privatsphäre durch das System…

Gewahrt	☐	☐	☐	☐	☐	Gefährdet

Das Systemverhalten empfinde ich als…

Zuverlässig	☐	☐	☐	☐	☐	Unzuverlässig

Meine Bedürfnisse werden vom System…

Verstanden	☐	☐	☐	☐	☐	Nicht Verstanden

Ich fühle mich dem System gegenüber…

In Kontrolle	☐	☐	☐	☐	☐	Ausgeliefert

Teilnehmernr.:_____

Insgesamt beurteile ich die Anpassungen des Systems in Form von der personalisierten Automatisierungen…

menschlich	☐	☐	☐	☐	☐	☐	☐	technisch
isolierend	☐	☐	☐	☐	☐	☐	☐	verbindend
angenehm	☐	☐	☐	☐	☐	☐	☐	unangenehm
originell	☐	☐	☐	☐	☐	☐	☐	konventionell
einfach	☐	☐	☐	☐	☐	☐	☐	kompliziert
fachmännisch	☐	☐	☐	☐	☐	☐	☐	laienhaft
hässlich	☐	☐	☐	☐	☐	☐	☐	schön
praktisch	☐	☐	☐	☐	☐	☐	☐	unpraktisch
sympathisch	☐	☐	☐	☐	☐	☐	☐	unsympathisch
umständlich	☐	☐	☐	☐	☐	☐	☐	direkt
stilvoll	☐	☐	☐	☐	☐	☐	☐	stillos
voraussagbar	☐	☐	☐	☐	☐	☐	☐	unberechenbar
minderwertig	☐	☐	☐	☐	☐	☐	☐	wertvoll
ausgrenzend	☐	☐	☐	☐	☐	☐	☐	einbeziehend
bringt mich den Leuten näher	☐	☐	☐	☐	☐	☐	☐	trennt mich von Leuten
nicht vorzeigbar	☐	☐	☐	☐	☐	☐	☐	vorzeigbar
zurückweisend	☐	☐	☐	☐	☐	☐	☐	einladend
phantasielos	☐	☐	☐	☐	☐	☐	☐	kreativ
gut	☐	☐	☐	☐	☐	☐	☐	schlecht
verwirrend	☐	☐	☐	☐	☐	☐	☐	übersichtlich
abstoßend	☐	☐	☐	☐	☐	☐	☐	anziehend
mutig	☐	☐	☐	☐	☐	☐	☐	vorsichtig
innovativ	☐	☐	☐	☐	☐	☐	☐	konservativ
lahm	☐	☐	☐	☐	☐	☐	☐	fesselnd
harmlos	☐	☐	☐	☐	☐	☐	☐	herausfordernd
motivierend	☐	☐	☐	☐	☐	☐	☐	entmutigend
neuartig	☐	☐	☐	☐	☐	☐	☐	herkömmlich
widerspenstig	☐	☐	☐	☐	☐	☐	☐	handhabbar

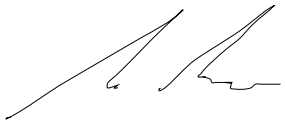
Teilnehmernr.:_____

Hatten Sie Probleme die Fragen zu beantworten? Oder haben Sie weitere Kommentare zu dem Fragebogen?

Vielen Dank

Eidesstattliche Erklärung

Ich versichere die vorliegende Dissertation mit dem Titel „Kontextsensitive Personalisierung automotiver Benutzerschnittstellen“ selbstständig und ohne unerlaubte Hilfe angefertigt zu haben. Bei der Verfassung der Dissertation wurden keine anderen als die im Text aufgeführten Hilfsmittel verwendet. Ein Promotionsverfahren wurde zu keinem früheren Zeitpunkt an einer anderen Hochschule oder bei einem anderen Fachbereich beantragt.



Berlin, Dezember, 2013 *Sandro Rodriguez Garzon*