Modellierung von Entscheidungen beim Einfädeln auf die Autobahn

Lars Weber, Martin Baumann, Andreas Lüdtke und Rike Steenken

Schlüsselwörter: Statechart, Aufgabenmodell, Fahrerverhalten, Kognitive Architekturen

Zusammenfassung

Basierend auf einem hierarchischen Tasknetzwerk und zwei experimentellen Studien mit Versuchspersonen für das Fahrmanöver „Einfädeln auf die Autobahn“ wurde ein Statechart-Aufgabenmodell in Matlab Stateflow erstellt, welches mit Hilfe formaler Übersetzungsregeln in Wissensbasen für unterschiedliche kognitive Architekturen überführt werden soll. Ziel dieser Statechartdarstellung ist eine bessere Vergleichbarkeit von Simulationsergebnissen verschiedener kognitiver Architekturen, da das zugrundeliegende Aufgabenmodell identisch ist. Architekturspezifische Eigenschaften fließen durch den Übersetzungsprozess ein. Diese Publikation präsentiert Teile des Statechartmodells, sowie grundlegende Übersetzungsregeln für die kognitive Architektur CASCaS. Um in der Simulationsumgebung die Einfädelsituationen der Experimente nachzufahren, wurde das nach CASCaS übersetzte Modell noch manuell vervollständigt. Ein kurzer Vergleich der Einfädelentscheidungen von Modell und Experimentdaten wird ebenfalls vorgestellt.

Based on a hierarchical task network and two experimental studies with subjects for the driving maneouver “freeway merging” a statechart task model in matlab stateflow was constructed, which in future should be translated into knowledge bases for different cognitive architectures. Goal of the statechart model is a better comparability of simulation results of different cognitive architectures due to a single input representation of the task. Specific architectural properties are encapsulated in translation rules but are not part of the task model. This paper presents parts of the statechart model and basic translation concepts for the cognitive architecture CASCaS. Currently the translated task model needed to be extended manually to be uploaded onto CASCaS which was then able to drive the merging scenarios that were previously presented to real drivers in the same simulator. Recorded evaluation data will be presented here.

Einleitung

Im Projekt IMoST[1] werden Grundlagen für einen Entwicklungsprozess zum Entwurf moderner Fahrerassistenzsysteme geschaffen, der eine integrierte Simulation mit Fahrer-, Fahrzeug-, Assistenzsystem- und Umgebungsmodellen vorsieht. Hierzu wird eine Simulationsplattform entwickelt mit der es möglich sein soll, Interaktionsverhalten zwischen Fahrer und Assistenzsystem frühzeitig im Entwicklungsprozess zu testen. Die untersuchten Assistenzsysteme unterstützen, bzw. übernehmen Teile der primären Fahraufgabe (z.B. Adaptive Cruise Control). Dabei besteht die Gefahr, dass Fahrer ihre Aufmerksamkeitsverteilung ändern, z.B. den umliegenden Verkehr weniger genau beobachten. Resultiert daraus ein verändertes, eventuell schlechteres, Situationsbewusstsein, gerät der Fahrer im schlimmsten Fall „out of the loop“, und kann bei Systemausfall nicht mehr adäquat reagieren. In IMoST werden am Szenario „Einfädeln auf die Autobahn“, sowie mit einem Einfädelassistenten der unterschiedliche Automatisierungsgrade unterstützt, solche Aufgabenänderungen untersucht. Mit der Simulationsplattform soll ein Werkzeug entstehen, das Änderungen an Aufgabenverteilungen, sowie Auswirkungen auf das Situationsbewusstsein während der Assistenzsystementwicklung aufdeckt. Als Teil der Plattform soll neben einem kognitiven Fahrermodell basierend auf der kognitiven Architektur CASCaS zu Vergleichszwecken auch ein ACT-R Modell angebunden werden.

[1] Integrated Modeling for Safe Transportation, gefördert durch das MWK-Niedersachsen (VW-Vorab)

Bisher werden Aufgabenmodelle für jede Architektur direkt in dessen Wissensrepräsentation spezifiziert, was zur Vermischung von Architekturspezifika und Aufgabenmodell führt und einen Ergebnisvergleich erschwert. Zielsetzung dieser Publikation ist eine Problemlösung vorzustellen, die ein einziges, formal spezifiziertes Aufgabenmodell beinhaltet, welches mit Übersetzern in Wissensbasen kompiliert werden soll, und somit architekturspezifische Eigenschaften in den Übersetzern vom Kern des Aufgabenmodells entkoppelt. Im folgenden Abschnitt wird zunächst die Wahl des Formalismus begründet. Die Ergebnisse zweier experimenteller Studien werden vorgestellt und fließen in das Aufgabenmodells ein. Simulationsergebnisse des nach CAScaS übersetzten Aufgabenmodells werden mit den Einfädelentscheidungen der Experimentdaten verglichen. Abschliessend erfolgt ein Ausblick auf zukünftige Arbeiten.

Formalismus zur Modellierung von Fahreraufgaben

Zur Auswahl eines geeigneten Formalismus für die Aufgabenmodellierung müssen die Besonderheiten von Fahraufgaben berücksichtigt werden: Cacciabue, Re und Macchi (2008) argumentieren, dass die Fahraufgabe kontinuierlich der lokalen Verkehrssituation angepasst werden muss. Viele Teilaufgaben werden bei Bedarf ausgeführt und lassen sich nicht statisch eingliedern. Klassische Taskanalysen und deren Ergebnisse in Form von hierarchischen Aufgabennetzwerken (HTN) sind daher weniger geeignet. Sie verwendeten eine deskriptive Darstellung des Goals-Means Task Analysis Ansatz (Hollnagel, 1993), mit Tabellen für Zielstrukturen und deren Vorbedingungen, sowie ein Flussdiagramm für die dynamische Interaktion.

Unser Ansatz verfolgt das Ziel einer vereinheitlichten Darstellung, wodurch der Übersetzungsprozess vereinfacht werden soll. Dazu muss der Formalismus sowohl hierarchische Zielstrukturen als auch dynamische Ablaufeigenschaften abbilden können. Aufbauend auf den Arbeiten von Delorme und Song (2001) die endliche Automaten zur Darstellung von Abläufen von Fahraufgaben nutzen, verwenden wir eine Erweiterung endlicher Automaten: Statecharts (Harel, 1987) bieten die Möglichkeit, serielle und parallele Aufgaben zu modellieren und können Unterbrechungen, beziehungsweise die Wiederaufnahme unterbrochener Aufgaben abbilden. Dynamische Aufgabenwechsel zwischen mehreren oder innerhalb eines parallelen Statecharts können über Events modelliert werden. Einzelne Aufgaben können entsprechend ihrer Teilaufgaben in mehrere Zustände innerhalb eines Statecharts untergliedert werden und hierarchische Gliederungen erlauben auch das Modellieren komplex strukturierter Aufgaben. Somit bieten sich Statecharts als Modellierungssprache für Fahraufgaben an.

Bezüglich automatisierter Übersetzung existieren bereits Ansätze, Aufgabenmodelle auf höherem Abstraktionsniveau zu spezifizieren und anschliessend für eine kognitive Architektur zu kompilieren: Basierend auf dem Ontologieeditor Protégé können mit dem Herbal Toolkit (Cohen, 2005) Aufgabenmodelle für SOAR erstellt werden. Mit G2A (Amant, 2004) lassen sich effizient GOMS Modelle nach ACT-R übersetzen. Bei G2A zeigte sich auch, dass Abstraktionen Probleme bereiten können: der LookAt() Operator von GOMS kann aufgrund der detaillierteren Modellierungsmöglichkeiten in ACT-R auf unterschiedliche Arten abgebildet werden. Im Gegensatz zu GOMS, das einige wenige kognitive Operatoren vorgibt, können kognitive Operatoren und Konzepte in Statecharts frei modelliert und bibliotheksartig eingebettet werden. Um einen einheitlichen Mechanismus für unterschiedliche kognitive Architekturen zu erstellen, definieren wir den semantischen Kern der geplanten Zielarchitekturen, der die minimale Schnittmenge gleichartiger Konzepte beinhaltet. Darüber hinaus werden Architekturspezifische Aspekte in Übersetzungsregeln gekapselt. Wir wollen auf dieser Basis, Übersetzungen nicht nur für eine einzige kognitive Architektur zu ermöglichen, sondern für CASCaS und ACT-R.

Empirische Untersuchungen zum Einfädelverhalten

Für die Erstellung des Aufgabenmodells wurden zwei Experimente mit dem Ziel durchgeführt genauere Kenntnis über die Einflussfaktoren auf die Einfädelentscheidung zu gewinnen. Das Szenario des ersten Experimentes wurde bewußt einfach gehalten: während Fahrer A auf der Beschleunigungsspur fuhr, näherte sich auf der rechten Autobahnspur von hinten ein Fremdfahrzeug B und der Fahrer sollte einfädeln. Manipuliert wurden die Distanz d_{AB} und die Differenzgeschwindigkeit $vdiff_{AB}$ zwischen Fremd- und Egofahrzeug zu dem Zeitpunkt, an dem der Fahrer den Beginn des Beschleunigungsstreifens passierte. Aus jeweils drei Abstufungen für d_{AB} (20, 30, 40m) und $vdiff_{AB}$ (20, 30, 40km/h) ergaben sich 9 Kombinationen, die jeder der 20 Versuchspersonen je 4 mal in randomisierter Abfolge dargeboten wurden. Als abhängige Variablen wurde das Entscheidungsverhalten des Fahrers betrachtet. Die Ergebnisse in Tab.1 zeigen, dass die Versuchspersonen sowohl d_{AB} als auch $vdiff_{AB}$ beim Einfädeln berücksichtigen. Mit zunehmender $vdiff_{AB}$ fädeln weniger Versuchspersonen vor dem Fremdfahrzeug ein. Bei $vdiff_{AB}$ von 20 und 30km/h spielt d_{AB} für die Entscheidung eine wesentliche Rolle. Hier steigt die Häufigkeit des vorher Einfädelns mit zunehmender Distanz an, so dass bei einer $vdiff_{AB}$ von 20km/h und d_{AB} von 40m in 91% der Durchgänge eingefädelt wird.

Tab.1: Einfädelvorgänge vor B. Max. 80 pro Eintrag: Absolutwert / prozentuale Umrechnung.

	20 m	30 m	40 m
20 km/h	7 / 8,75	49 / 61,25	73 / 91,25
30 km/h	0	2 / 2,5	35 / 43,75
40 km/h	0	0	0

Ziel des *zweiten* Experiments war es, den Einfluss des Verlaufs von d_{AB} und $vdiff_{AB}$, sowie einer abgeleiteten Grösse „Veränderungsrate der Objektgrösse von B im Aussenspiegel“ (rate_of_change_b, kurz roc_B) während des Einfädelvorgangs auf das Einfädelverhalten der Versuchspersonen zu untersuchen. Während in Experiment 1 d_{AB} und $vdiff_{AB}$ nur zu Beginn des Beschleunigungsstreifens festgelegt wurden, wurde der Verlauf nun kontinuierlich kontrolliert. Dazu wurde die interaktive Simulation durch Präsentation definierter Einfädelssequenzen ersetzt und die Versuchspersonen mussten nach jeder Videosequenz eine Einfädelentscheidung treffen. Die Ergebnisse aus Experiment 1 bezüglich der Abhängigkeit der Einfädelentscheidung von d_{AB} und $vdiff_{AB}$ wurden bestätigt. Zusätzlich wurde mit einer logistischen Regressionsanalyse über die drei Variablen d_{AB}, $vdiff_{AB}$ und roc_B gezeigt, das roc_B den grössten Einfluss bezüglich der Einfädelentscheidung hat. Daraufhin wurde der erste Modellierungsansatz (Weber & Lüdtke, 2007), die Einfädelentscheidung über d_{AB} und $vdiff_{AB}$ abzubilden verworfen und durch die Veränderungsrate der Objektgrösse im Aussenspiegel ersetzt. Das nachfolgend vorgestellte erweiterte Aufgabenmodell berücksichtigt diese Änderung.

Erweitertes Aufgabenmodell

In die Erstellung des erweiterten Aufgabenmodells sind Ergebnisse einer Fahrlehrerbefragung von Kassner (2004), ein daraus abgeleitetes erstes Aufgabenmodell (Weber & Lüdtke, 2007), sowie die Erkenntnisse aus den beiden Experimenten eingeflossen. Abb.1 zeigt ein Matlab Stateflow Modell[2], welches den Statechart Formalismus verwendet und den Einfädelprozess im Bereich der Beschleunigungsspur modelliert (Mit dem rückwärtigen Fahrzeug B assoziierte Variablen sind mit ‚_b’ markiert). Auf oberster Ebene befindet sich der Hauptzustand *CarDriving*, der mehrere parallele Zustände (gestrichelt umrandet) beinhaltet: Neben den Fahrzeugkontrollaufgaben *LongitudinalControl* und *LateralControl* gibt es die Einfädelaufgabe

[2] Desweiteren existiert ein umgebendes Matlab Simulink Modell, welches zur Simulation des Aufgabenmodells genutzt wird, dies kann hier allerdings nicht mehr dargestellt werden.

ManeouverTask. Diese besteht aus zwei sich verschränkenden Teilaufgaben: *FindGap* ist mit dem Blick in den linken Aussenspiegel assoziiert, während *FrontView* die Blickkontrolle in Fahrtrichung auf den Fahrbahnverlauf (*CourseOfRoad*), sowie eventuell vorausfahrende Fahrzeuge (*LeadCar*) abbildet.

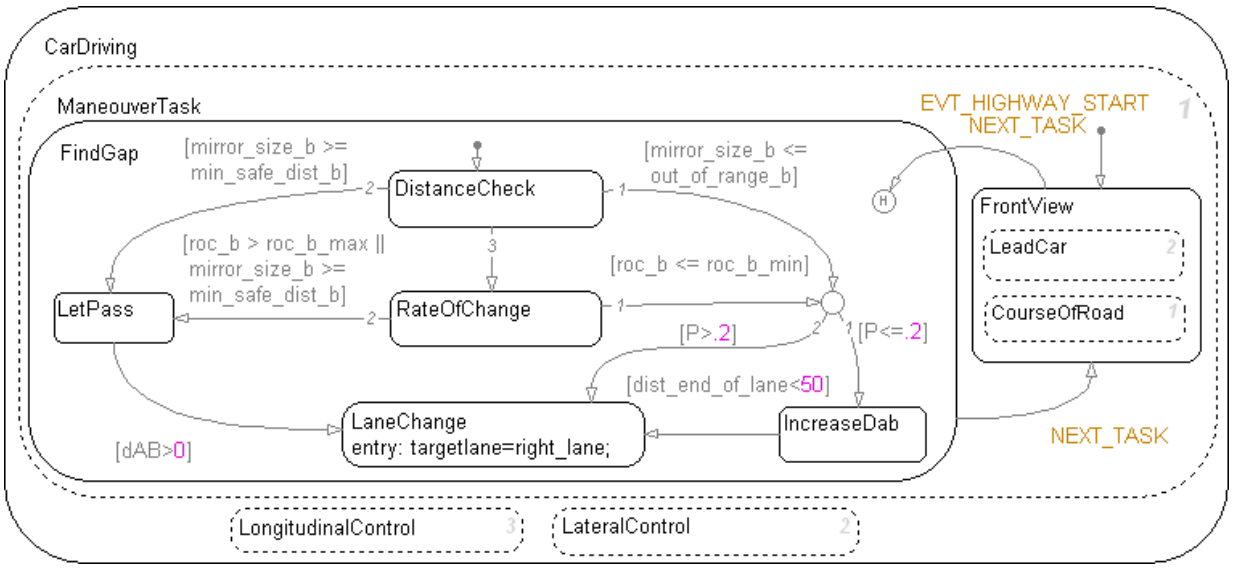

Abb.1: Abbildung erstellt mit Matlab Stateflow 7.5: Statechart für das Einfädeln auf die Autobahn. (Das vollständige Matlab Simulink Simulationsmodell konnte nicht abgebildet werden)

Das Modell beginnt in Zustand *FrontView*, wenn das Signal *EVT_HIGHWAY_START* generiert wird. Zwischen *FindGap* und *FrontView* wird periodisch umgeschaltet. Beim ersten Start von *FindGap* beginnt *DistanceCheck*, mit einem Orientierungsblick im linken Aussenspiegel bezüglich der Entfernung rückwärtigen Verkehrs. Als Entscheidungsvariable wird die Grösse des nachfolgenden Fahrzeugs im Spiegel benutzt (*mirror_size_b*). Es gibt drei Transitionsmöglichkeiten: 1) Transition über *LetPass*: Ist Fahrzeug B sehr nahe, entscheidet sich das Modell passieren zu lassen und wartet auf das Vorbeifahren (*LetPass*) bis es den Spurwechsel in *LaneChange* durchführen kann. 2) Transition über *ConnectiveJunction* (rechte Transition): Befindet sich B weit entfernt, entscheidet sich das Modell vorher einzufädeln. Anhand der Experimentdaten wurden zwei unterschiedliche Fahrerstrategien identifiziert: In 20% der Fälle „vorher Einfädeln“ nutzten Fahrer die Einfädelspur zur Erhöhung des Abstandes aus (*IncreaseDab*), in 80% der Fälle fadelten sie sofort vorher ein. In der *ConnectiveJunction* wird dabei zufällig entschieden, welche Fahrstrategie ausgewählt wird. 3) Transition über *RateOfChange*: Eine Einfädelentscheidung kann nicht unmittelbar anhand des Abstandes getroffen werden. Das Modell schätzt anhand der Veränderungsrate der Objektgrösse im Aussenspiegel (*roc_b*) ab, wie schnell Fahrzeug B näher kommt. Solange keine Entscheidung gefällt werden kann, verharrt das Modell im Zustand.

Zwei Besonderheiten müssen noch erläutert werden: 1) Im letzten Zustand *LaneChange* wird mit dem *entry* statement die Zielvariable für den automatischen Prozess *LateralControl* verändert. 2) Intuitiv läßt sich der Historyconnector (eingekreistes H) in *FindGap* verwenden: Wird *FindGap* unterbrochen und anschliessend wiederaufgenommen, wird in dem Zustand fortgefahren, der zum Zeitpunkt des Verlassens aktiv war.

Modellierung in CASCaS

In diesem Abschnitt werden die grundlegenden Konzepte für die Übersetzung des Statechartmodells in eine CASCaS Wissensbasis vorgestellt. Wie im zweiten Abschnitt erwähnt, soll zunächst ein gemeinsamer semantischer Kern von CASCaS und ACT-R erstellt

werden. In einer ersten Version beinhaltet dieser Ziele, sowie Regeln mit Vorbedingungen und Perzeptaktionen. Die Grundstruktur einer Wissensbasis kann damit realisiert werden. In beiden Architekturen sind Ziele ein Strukturelement, um Handlungen in einer bestimmten, der Aufgabe angemessenen Reihenfolge durchzuführen. Regeln beinhalten Handlungsanweisungen (z.B. Perzeptaktionen), die an Vorbedingungen geknüpft sein können. Vorbedingungen sind Boolesche Ausdrücke und beschreiben eine bestimmte Situation die vorliegen muss (z.B. linke Autobahnspur frei), damit der Handlungsteil der Regel ausgeführt werden kann (z.B. Spurwechsel durchführen). Jede Regel kann genau einem Ziel zugeordnet sein, wobei das zur Laufzeit aktuell verfolgte Aufgabenziel dann eine separate Vorbedingung für die Auswahl dieser Regel darstellt. Ist eine Regel keinem Ziel zugeordnet, kann sie benutzt werden um z.B. die aktuelle Aufgabe zu unterbrechen.

Mit diesem initialen semantischen Kern lassen sich grosse Teile des Aufgabenmodells in die regelbasierte Darstellung der assoziativen Ebene von CASCaS (Lüdtke et al., 2009) übersetzen. Jeder Zustand wird auf ein Ziel abgebildet, während Transitionen in „reguläre Regeln“ für dieses Ziel übersetzt werden. Reguläre Regeln sind zielgebunden und sobald sie feuern gilt das Ziel als abgearbeitet. In CASCaS müssen die Variablen der Bedingungen dieser Regeln zur Auswertung aus dem Gedächtnis erinnerbar sein, andernfalls kann diese Regel nicht ausgewählt werden. Wenn keine reguläre Regel feuern kann, versucht CASCaS fehlende Informationen mit „Perzeptregeln“ zu sammeln. Zu diesem Zweck muss während der Übersetzung für jede Variable aller Transitionen eines Zustandes eine solche Perzeptregel generiert werden. Sollte nach dem Feuern aller Perzeptregeln keine ‚reguläre Regel’ auswählbar sein, versucht CASCaS einen Taskwechsel und feuert eine Warteregel für das aktuelle Ziel. Diese kann automatisch für jedes Ziel generiert werden. Reaktionen auf Events können in CASCaS in Form von „reaktiven Regeln“ abgebildet werden. Diese Regelart ist keinem Ziel direkt zugeordnet, sondern wird aktiviert, sobald die Perzeption entsprechende Trigger ‚bottom-up’ wahrnimmt und ins Gedächtnis schreibt.

Mit diesen Übersetzungsregeln konnte eine noch unvollständige Regelbasis für CASCaS erzeugt werden, bei der noch 1) die Anbindung an die autonome Ebene fehlte und 2) architekturspezifische Eigenschaften bezüglich Multitasking nicht integriert sind, wobei CASCaS zum Beispiel Prioritäten verwendet, um Aufgaben bevorzugt zu behandeln. Diese Erweiterungen wurden manuell eingefügt, um eine Fallstudie mit dem Fahrermodell durchführen zu können.

Fallstudie

Um die Modellierung des Entscheidungsprozesses und das durch Anwendung der Übersetzungsregeln resultierende Aufgabenmodell zu prüfen, wurde das CASCaS Fahrermodell in der Simulationsplattform getestet. Dazu musste es die die gleichen Szenarien fahren wie die Versuchsfahrer in Experiment 1.

Tab.2: Einfädelvorgänge vor B, max. 32 pro Eintrag: Absolut / prozentuale Umrechnung / prozentuale Umrechnung aus Tab.1 der Versuchspersonen.

	20 m	30 m	40 m
20 km/h	4 / 12,5 / 8,75	23 / 71,875 / 61,25	29 / 90,625 / 91,25
30 km/h	0	1 / 3,125 / 2,5	12 / 37,5 / 43,75
40 km/h	0	0	0

Tab.2 verdeutlicht, dass trotz der geringeren Anzahl an Simulationsfahrten des Modells die Prozentwerte der Einfädelungen pro Kombination eine gute Übereinstimmung mit den Fahrten der Versuchspersonen besitzen. Die gewählte Modellierung der Einfädelentscheidung anhand der Variablen *mirror_size_b* und *roc_b* lieferte mit relativ wenig Einstellaufwand für die mi-

nimalen und maximalen Werte dieser zwei Variablen (siehe Transitionen in Abb.1, z.B. *min_safe_dist_b* oder *roc_b_max*) gute Ergebnisse.

Zusammenfassung & Ausblick

In dieser Arbeit wurde eine Modellierung von Fahraufgaben vorgestellt, die als einheitliche Grundlage für die automatische Ableitung von Wissensbasen für unterschiedliche kognitive Architekturen dienen soll. Da bei der Modellierung von Fahreraufgaben kleine, interagierende und mit weniger festen Reihenfolgen versehene Aufgaben auftreten, ist die Wahl eines Verhaltensdiagrammes (Statechart) gegenüber der eines Strukturdiagrammes (HTN) sinnvoll. Die vorgestellten Übersetzungskonzepte sind in der Lage die verschiedenen Regelvarianten von CASCaS zu erzeugen, andere Teile wie z.B. die Anbindung der autonomen Ebene werden in anschließenden Arbeitsschritten ergänzt. Da CASCaS Regeln stark an GOMS angelehnt sind und für GOMS Regeln mit G2A ein Übersetzer nach ACT-R existiert, nehmen wir an, das Modell zukünftig für beide Architekturen übersetzen zu können. Neben einer formalen Spezifikation des semantischen Kerns, soll dieser um Konzepte für z.B. motorische oder Gedächtnisaktionen erweitert werden und es muss geklärt werden, wie die Zwei-Ebenen Struktur abgebildet werden kann, bzw., wie dies für ACT-R umgesetzt werden kann. Nach der Durchführung dieser Erweiterungen, sollen Vergleichsstudien mit beiden Architekturen durchgeführt werden.

Literatur

Amant, St. und Ritter, F.E. (2004). Automated GOMS to ACT-R model generation. In Proceedings of the 6. ICCM, Mahway, NJ: Lawrence Erlbaum [Best applied paper prize]

Anderson, J. (2000). *Learning and Memory: An integrated Approach*. Wiley

Cacciabue, P.-C., Re, C. und Macchi, L. (2008). Simple Simulation of Driver Performance for Prediction and Design Analysis. In P.C. Cacciabue, (Hrsg.), *Modeling Driver Behaviour in Automotive Systems* (S. 344-375), Springer Verlag.

Cohen, M.A., und Ritter, F.E. (2005). *Herbal Tutorial*. (Tech Report No. ACS 2004-2), Applied Cognitive Science Lab, School of Information Sciences and Technology, Penn State, http://acs.ist.psu.edu/reports/cohenR04.pdf.

Delorme, D. und Song, B. (2001). *Human Driver Model for SmartAHS*. California Research Report, UCB-ITS-PRR-2001-12.

Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming* (S. 231-274) Elsevier Science Publishers B.V.: 8

Hollnagel, E. (1993). *Human Reliability Analysis: Context and Control*. Academic Press, London.

Kassner, A. (2004). *Vergleich von idealem und tatsächlichem Fahrverhalten als Ansatzpunkt für Fahrerassistenzsysteme.* Diplomarbeit, TU Braunschweig, Institut für Psychologie.

Lüdtke, A., Weber, L., Osterloh, J.-P. und Wortelen, B. (2009). *Modeling Pilot and Driver Behavior for Human Error Simulation.* Conference Proceedings HCI 2009, Digital Human Modeling, San Diego, Springer: Lecture Notes in Computer science (LNCS).

Weber, L. und Lüdtke, A. (2007). Modellierung von Aufmerksamkeitsverteilung beim Einfädeln auf die Autobahn. In M. Rötting, u.a. (Hrsg.), *Prospektive Gestaltung von Mensch-Technik-Interaktion* (S. 35-40). ZMMS Spektrum: 22 (25).