Technische Universität München
Fakultät für Informatik

Forschungs- und Lehreinheit XI
Angewandte Informatik / Kooperative Systeme

KOMODE
Ein semantisches Kontextmodell für kollaborative Anwendungen in automobilen ad-hoc Netzwerken

Robert Eigner

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

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. M. Bichler
Prüfer der Dissertation:	1.	Univ.-Prof. Dr. J. Schlichter
	2.	Univ.-Prof. Dr. U. Baumgarten

Die Dissertation wurde am 06.08.2009 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 15.02.2010 angenommen.

Gut war es, zu arbeiten. Dieses Sich-schweben-Fühlen, wenn Dinge und Menschen, alles, was man sah, dachte, las, lebte, hineinwuchs in den Plan. Gut sogar die Wut, der Ärger, wenn Stockungen kamen Widerstände, wenn sich zeigte, daß am Organismus etwas nicht in Ordnung war. Die Befriedigung dann, wenn es sich wieder einrenkte, so also, daß sich erwies: der Gedanke war wirklich lebendig, voll Widerhaken eben und Widerspruch. Prachtvoll die Arbeit an der Schreibmaschine, wenn die Buchstaben sich hineinwühlten ins Papier, Werk wurden, sichtbar. Die Lust, wenn plötzlich ein Einfall aufsprang, überraschend, irgendwo, unvermutet, im Bad, beim Essen, über dem Lesen eines Zeitungsblattes, mitten in einem albernen Gespräch. Willkommen auch jener finstere Zustand, wo man dahockte, fluchend, sich zusammenigelnd, weil man sich sagte: Es geht nicht, es läßt sich nicht zwingen. Da liegt dieser Berg vor einem, man kommt nicht hinauf. Recht haben die anderen, die einen auslachen. Es fehlt an Kraft, man hat sich übernommen. Man ist ein Stümper. Dann wieder das deprimierende aufstachelnde Gefühl, wenn man über den Werken derer saß, die es dennoch zwangen. Wenn aus Ihren Büchern das abgelebte Leben neu aufstand, mit dem eigenen sich mischend. [Feu93]

Lion Feuchtwanger – Erfolg

Abstract

With the raising propagation of sensors in vehicles, cars will increase their ability to draw a more accurate picture of their driving situation. At the same time also the number of control units and driver assistance applications using such information rises. Basis of these applications is always the same set of information provided by these sensors, even if they are implemented in a technically different way by different manufacturers. Examples are speed, position, acceleration and friction sensors, whose data is linked together and processed for e. g. ABS or ESP systems.

Recent developments make it possible to link this information at another level of networking: Vehicular Ad-Hoc Networks (VANETs) enable cars to exchange this data among different vehicles, which further aggravates the problems of information integration and heterogeneity. Since the WWW has been facing similar problems for a longer time and provides a promising solution using ontologies, a technique of the Semantic Web, the purpose of this work is to investigate whether ontologies can be used as a tool for context modeling in VANETs.

This work analyzes the specific requirements of context models in VANETs and presents KOMODE – a basic set of ontologies for context modeling. It has been used as a foundation for designing collaborative applications for VANETs, which are to demonstrate the capabilities and limitations of the approach. These sample applications include a context-aware recommender, an application for cooperative collision avoidance and procedures for using context for addressing and routing in VANETs.

Kurzfassung

Mit zunehmender Verbreitung von Sensoren in Fahrzeugen werden diese immer mehr in die Lage versetzt, ein genauer werdendes Bild ihrer Fahrsituation zu zeichnen. Gleichzeitig steigt auch die Zahl der Steuergeräte und Fahrerassistenzanwendungen, die diese Informationen verarbeiten sollen. Grundlage dieser Anwendungen ist stets der gleiche Satz an durch die Sensoren bereitgestellten Informationen, auch wenn diese bei verschiedenen Herstellern technisch unterschiedlich realisiert sind. Als Beispiele sollen hier Geschwindigkeits-, Positions-, Beschleunigungs- und Schlupfsensoren gelten, deren Daten z. B. von ABS- oder ESP-Systemen vernetzt wird.

Neue technische Entwicklungen machen es möglich, diese Informationen auf einer anderen Ebene zu vernetzen: Vehicular Ad-Hoc Networks (VANETs) machen den Austausch dieser Daten unter verschiedenen Fahrzeugen möglich, was die Probleme der Informationsheterogenität und -integration noch weiter verstärkt. Da das WWW schon seit längerer Zeit vor ähnlichen Problemen steht und mit Ontologien, einer Technik des Semantic Web, einen vielversprechenden Lösungsansatz zur Modellierung anwendungsübergreifender Konzepte, ihrer Eigenschaften und Beziehungen untereinander bietet, soll mit dieser Arbeit untersucht werden, ob sich Ontologien auch als Mittel zur Kontextmodellierung in VANETs eignen.

Diese Arbeit analysiert die speziellen Anforderungen an Kontextmodelle in VANETs und stellt KOMODE – einen Basissatz von Ontologien zu Kontextmodellierung – vor, auf dessen Grundlage kollaborative Anwendungen für VANETs entwickelt wurden, die die Leistungsfähigkeit und Grenzen des Ansatzes aufzeigen sollen. Diese Beispielanwendungen umfassen einen kontextsensitiven Recommender, eine kollaborative Kollisionsvermeidung und Verfahren zur kontextsensitiven Adressierung und Routing in VANETs.

Danksagung

Diese Arbeit ist im Rahmen meiner Zeit als wissenschaftlicher Assistent am Lehrstuhl Informatik XI: Angewandte Informatik / Kooperative Systeme der Technischen Universität München entstanden. Dem Leiter dieses Lehrstuhls, Herrn Prof. Dr. Johann Schlichter, sei an dieser Stelle als erstem gedankt: dafür, dass er mir die Möglichkeit gegeben hat, in seinem Team an diesem Thema arbeiten zu dürfen, für seine Begleitung meiner Arbeit, die konstruktive Unterstützung und für den großen Freiraum während der wunderbaren Zeit des Lernens, Arbeitens und auch des Spaßes an seinem Lehrstuhl. Herzlicher Dank gebührt auch meinem Zweitgutachter Herrn Prof. Dr. Uwe Baumgarten, den ich mit einigen Deadlines regelrecht überrumpelt habe.

Die wunderbare Zeit an der Universität ist aber auch dem großartigen Team zu verdanken, dass in jeder Hinsicht – fachlich wie auch persönlich – einmalig ist. Den Mitgliedern dieses Teams danke ich für die Diskussionen, Kaffeepausen, das gemeinsame Mittagessen und die dazu gehörenden Gespräche, für die Skiausflüge, die (legendären) Doktorandenseminare und Weihnachtsfeiern. Die Arbeit geht einfach leichter von der Hand, wenn man es mit einer solch hervorragenden Truppe von motivierten Kollegen und Freunden zu tun hat. Der Applaus geht an Roland Bader, Halgurt Bapierre, Michele Brocco, Friedl Bunke, Evelyn Gemkow, Dr. Georg Groh, Florian Forster, René Frieß, Jan Herrmann, Prof. Dr. Michael Koch, Dr. Matthias Kramm, Dr. Thomas Leckner, Walter Kammergruber, Hubert Kreuzpointner, Alexander Lehmann, Marlene Müller, Vivian Prinz, Dr. Jochen Reich, Dr. Rosmary Stegmann, Dr. Frank Schütz, Karlheinz Toni, Dr. Wolfgang Wörndl und Dr. Weilun Zhuang. Ohne Ihre Unterstützung wäre ich nie an diesen Punkt gelangt.

Diese Arbeit wäre nicht ohne Hilfe von externen Partnern entstanden. Deshalb möchte ich dankend darauf hinweisen, dass diese Arbeit vom deutschen Bundesministeriums für Bildung und Forschung (Förderkennzeichen 01AK064B) gefördert wurde. Diese Unterstützung kam mir im Rahmen des Forschungsprojekts „NOW - Network on Wheels“ zugute, welches mir die Möglichkeit gegeben hat, mein Thema in einem recht praxisnahen und neuartigen Umfeld und mit sehr kompetenten und freundlichen Kollegen zu bearbeiten. Besonders hervorgehoben und stellvertretend für alle anderen sei hier die Kooperation mit meinen Projektleitern Dr. Timo Kosch und Dr. Markus Straßberger von der BMW Forschung und Technik GmbH, die das Arbeiten im Projekt immer spannend und bereichernd

gestaltet haben, für viele Anregungen sorgten und auch die Finanzierung sichergestellt haben.

Bei der Durchführung der Arbeit wurde ich tatkräftig unterstützt von meinen Studenten, die mir viel Implementierungarbeit abgenommen haben und ohne die diese Arbeit nicht zustande gekommen wäre: stellvertretend für alle geht mein Dank hier an Andreas Gardener, Wolfgang Linsmeier, Georg Lutz, Christoph Mair, Daniel Saraci und Andreas Staffler.

Zuletzt möchte ich mich für die Unterstützung durch meinen Freundeskreis und durch meine Familie bedanken. Sie haben in „Dürreperioden“ für den nötigen Durchhaltewillen und Motivation gesorgt. Besonders gedankt sei meinen Brüdern Alexander und Markus, die immer dann Ablenkung geliefert haben, wenn sie nötig war – und ab und zu auch dann, wenn keine nötig war. Schließlich gebührt der größte Dank den wichtigsten Menschen in meinem Leben: meinen Eltern Helga und Heinz, denen ich alles verdanke, was ich bin und was ich kann.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABC	Active Body Control
ABS	Anti-Blockiersystem
ACC	Adaptive Cruise Control
AIFS	Arbitration Interframe Space
AODV	Ad hoc On-Demand Distance Vector [Routing]
ASR	Antriebsschlupfregelung
CAN	Controller Area Network
CBLR	Cluster-based location Routing
CC/PP	Composite Capabilities/Preference Profiles
CERN	Conseil Européen pour la Recherche Nucléaire
CoBrA	Context Broker Architecture
COMANTO	Context Management Ontology
CoNoN	Context Ontology
CoOL	Context Ontology Language
CSCP	Comprehensive Structured Context Profiles
CSMA	Carrier Sense Multiple Access
CW	Contention Window
DAML-OIL	= DAML-ONT + OIL
DAML-ONT	Darpa Agent Markup Language - Ontology
DCF	Distributed Coordination Function
DIFS	Distributed Interframe Spacing (DIFS)
DSR	Dynamic Source Routing
DTR	Distronic
EDCF	Enhanced Distributed Coordination Function
EDV	Elektronische Datenverarbeitung
ESP	Electronic Stability Program
ETSI	European Telecommunications Standards Institute
FCC	Federal Communications Commission
GPS	Global Positioning System
GSM	Global System for Mobile Communications
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol

IARP	Intra-zone Routing Protocol
IEEE	Institute of Electrical and Electronics Engineers
IERP	Inter-zone Routing Protocol
INVENT	Intelligenter Verkehr und Nutzergerechte Technik
ISO	International Organization for Standardisation
LAR	Location-aided Routing
LIN	Local Interconnect Network
MDDV	Mobility-Centric Data Dissemination Algorithm for Vehicular Networks
MOST	Media Oriented Systems Transport
MP3	MPEG-1 Audio Layer 3
MPEG	Moving Picture Experts Group
NoW	Network on Wheels
OFDM	Orthogonal Frequency-Division Multiplexing
OGC	Open Geospatial Consortium
OIL	Ontology Inference Layer
OLSR	Optimized Link State Routing
ORM	Object-Role Modeling
OSPF	Open Shortest Path First
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
SBC	Sensotronic Brake Control
SIFS	Short Interframe Spacing
SIM-TD	Sichere Intelligente Mobilität – Testfeld Deutschland
TMC	Traffic Message Channel
UML	Unified Modeling Language
UMTS	Universal Mobile Telecommunications System
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTRA-TDD	UMTS Terrestrial Radio Access – Time Division Duplexing
VANET	Vehicular Ad-hoc Network
VCM	Vehicle Context Model
W3C	World Wide Web Consortium
WAVE	Wireless Access in Vehicular Environments
WLAN	Wireless Local Area Network
WWW	World Wide Web
XML	Extensible Markup Language
ZRP	Zone Routing Protocol

1 Einführung

1.1 Automobile auf dem Weg vom Industrie- ins Informationszeitalter

Nur an wenigen „Industriegütern“ ist der Weg vom Industriezeitalter in die Informationsgesellschaft deutlicher ablesbar als am Automobil. Der „Benz Patent-Motorwagen Nummer 1“ (Abbildung 1.1) war bei seiner Konstruktion 1886 ein Automobil, das hauptsächlich mit viel Erfindergeist und handwerklichem Geschick konstruiert wurde, um Personen von einem Ort zum anderen zu befördern. Bei seiner Fertigstellung war noch nicht absehbar, dass man bei der Entwicklung seiner Nachkommen auch Elektroniker geschweige denn Informatiker benötigen würde. Die größten Weiterentwicklungen des Automobils beschränkten sich daher für lange Zeit auf die Perfektion des Verbrennungsprinzips und Verbesserungen am Karosseriebau.

Dennoch wird 1960 die erste (auf analogen Schaltungen beruhende) elektronische Einspritzanlage im VW 1600 eingesetzt und man mag es als Zufall bezeichnen, dass manche Soziologen den Beginn des Informationszeitalter in das Jahrzehnt zwischen 1970 und 1980 legen [NM78], aber ausgerechnet 1978 findet das erste vollelektronische digitale Steuergerät Einzug in die Automobilwelt: Die Mercedes S-Klasse der Baureihe W116 (Abbildung 1.2) wird mit einem Anti-Blockier-System (ABS) der Firma Bosch ausgestattet. Zum ersten Mal wurde es nötig und möglich, Informationen über den Fahrzustand eines Fahrzeuges zu sammeln und digital weiterzuverarbeiten. Damit hatte nicht nur die EDV (elektronische Datenverarbeitung) im wörtlichen Sinne ihren Einzug in das Auto gefunden, sondern Fahrzeuge waren zum ersten Mal im Stande – wenn auch in bescheidenem Umfang – Daten über ihre Umwelt, den *Kontext*, in dem sie sich bewegen, aufzunehmen und aufzubereiten.

Ab jetzt nahm die Entwicklung der Elektronik und der Informationsverarbeitung im Fahrzeug einen rasanteren Verlauf, ähnlich wie die „Informatisierung der Gesellschaft“ [NM78]: 1983 gab es dann schon genügend Steuergeräte im Fahrzeug, so dass man aus Ihrer Vernetzung zum Zweck des Informationsaustauschs einen Vorteil ziehen konnte. Dazu wurde von Bosch der CAN-(Controller Area Network)-Bus [Law00] entwickelt. Ein serieller asynchroner Bus, der zusammen mit Intel 1987 vorgestellt wurde und zum ersten Mal 1990 in der Mercedes S-Klasse (Baureihe W140, Abbildung 1.3) eingesetzt wurde. Zwar wurden auch schon vorher Steuergeräte miteinander vernetzt, aber diese Vernetzung wurde nur mit Hilfe von Punkt-zu-Punkt-Verbindungen realisiert, was die Kabelbäume in Fahrzeu-

Abbildung 1.1: Der von Carl Benz 1886 erbaute Patent-Motorwagen Nummer 1, Quelle: Daimler AG

gen schon damals auf eine Länge von bis zu zwei Kilometern anwachsen ließ, bei teilweise nur fünf Steuergeräten. Der CAN-Bus ist bis heute im Einsatz.
Heutzutage könnte man fast meinen, ein Fahrzeug dient gar nicht mehr der Beförderung von Personen: ein (relativ) aktuelles Modell der Mercedes S-Klasse (Baureihe W220, bis 2005) enthält, obwohl außer dem CAN-Bus noch weitere Bus-Systeme (MOST [Grz07], LIN [GW05], FlexRay [Rau07]) im Einsatz sind, ca. 1900 Leitungen mit einer Gesamtlänge von drei Kilometern, bis zu 50 Steuergeräte (vgl. [BS07]), vollwertige Computer und eine Vielzahl von Sensoren, die Informationen sammeln. Man könnte ein Fahrzeug fast mehr als „Computer auf Rädern“ denn als Fortbewegungsmittel interpretieren.
Neueste Entwicklungen [Bra95, Che02, FEL01, RMM+02, FNS+08] im Automobilbereich treiben die Vernetzung von Fahrzeugen auf einer anderen Ebene als der CAN-Bus weiter: Automobile werden in Zukunft auch im Stande sein, sich untereinander zu vernetzen und Daten auszutauschen. Das führt dazu, dass ein Fahrzeug nun nicht nur mit seiner eigenen lokalen Menge an Informationen zurecht kommen muss, sondern auch die Daten anderer Fahrzeuge, die über eine Funkschnittstelle empfangen wurden, verarbeiten muss. Da diese Vernetzung natürlich auch zwischen Fahrzeugen verschiedener Hersteller stattfindet, von Geräten geleistet wird, die nicht unbedingt vom Fahrzeughersteller selbst stammen oder

Abbildung 1.2: Mercedes-Benz S-Klasse, Baureihe W116, Quelle: Daimler AG

Abbildung 1.3: Mercedes-Benz S-Klasse, Baureihe W140, Quelle: Daimler AG

auch mit der Hilfe von Infrastruktur-Komponenten (also nicht ausschliesslich zwischen Fahrzeugen) durchgeführt wird, wird deutlich, dass man es hier mit einen hohen Maß an Heterogenität zu tun bekommt: die Flut an Informationen und ihre Integration muss gebändigt werden und die Kontextdaten eines Fahrzeugs müssen maschinenlesbar *modelliert* werden, um die Verarbeitbarkeit durch andere Fahrzeuge und Rechner zu ermöglichen.

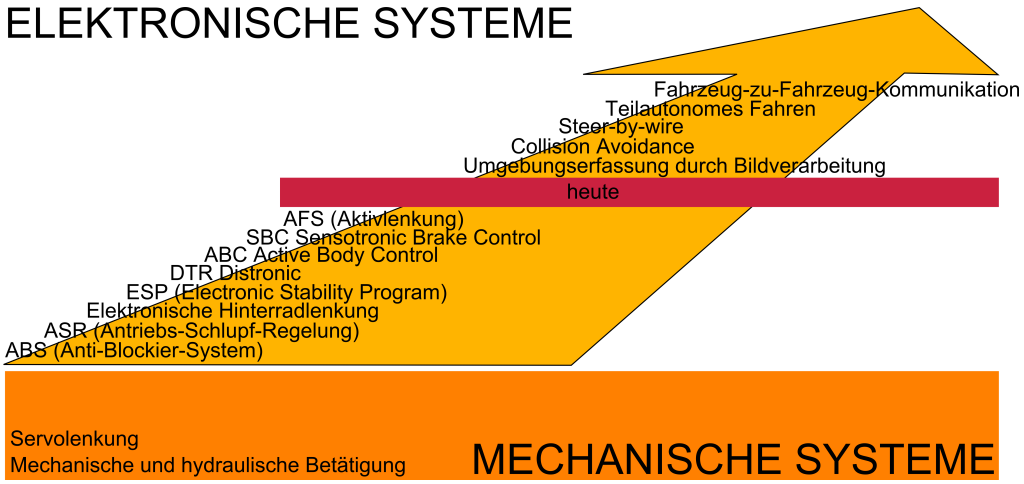


Abbildung 1.4: Entwicklung elektronischer Komponenten im Fahrzeug

Ähnlich wie sich dieser Wandel – wie in Abbildung 1.4 dargestellt – beim Automobil vom Meisterstück eines einzelnen Ingenieurs zur rollenden Informationsverarbeitungsmaschine vollzog, fand er auch in der Informationstechnologie selbst statt. Die Vernetzung von Rechnern unter Verwendung von einfachen Protokollen, machte es fast jedem Nutzer möglich, Informationen und Daten beliebiger Art einer breiten Öffentlichkeit zur Verfügung zu stellen, was zur Bildung des World Wide Web führte.

1.2 Das World Wide Web

Sehr deutlich ist der Übergang vom Industrie- ins Informationszeitalter in der Entwicklung des World Wide Web (WWW): Die zunehmende Digitalisierung des wissenschaftlichen Betriebes und dadurch anfallenden Mengen an Daten und Informationen machten am CERN in Genf ein System zur Verwaltung dieser Informationen nötig. Um allen Nutzern am CERN zugänglich zu sein, wurde daher ein vernetztes Informationssystem geschaffen [BL89]:

> The WorldWideWeb (W3) is a wide-area hypermedia information retrieval initiative aiming to give universal access to a large universe of documents.

Das WWW verließ bald die wissenschaftliche Welt am CERN und dank offener Protokolle wie HTTP, eindeutigen Adressen (URIs bzw. URLs) und relativ einfach verständlichen Sprachen, um Informationen zu strukturieren (HTML und andere Markup-Sprachen) trat es seinen Siegeszug als frei zugängliches *Internet* an. Besonderen Auftrieb für das Internet gab es 1993 als der erste freie grafikfähige Webbrowser „Mosaic“ für die Öffentlichkeit verfügbar wurde. Seitdem wächst das Internet rasant, die Anzahl der Internetseiten beläuft sich im November 2008 auf 185167897 Seiten (Abbildung 1.5).

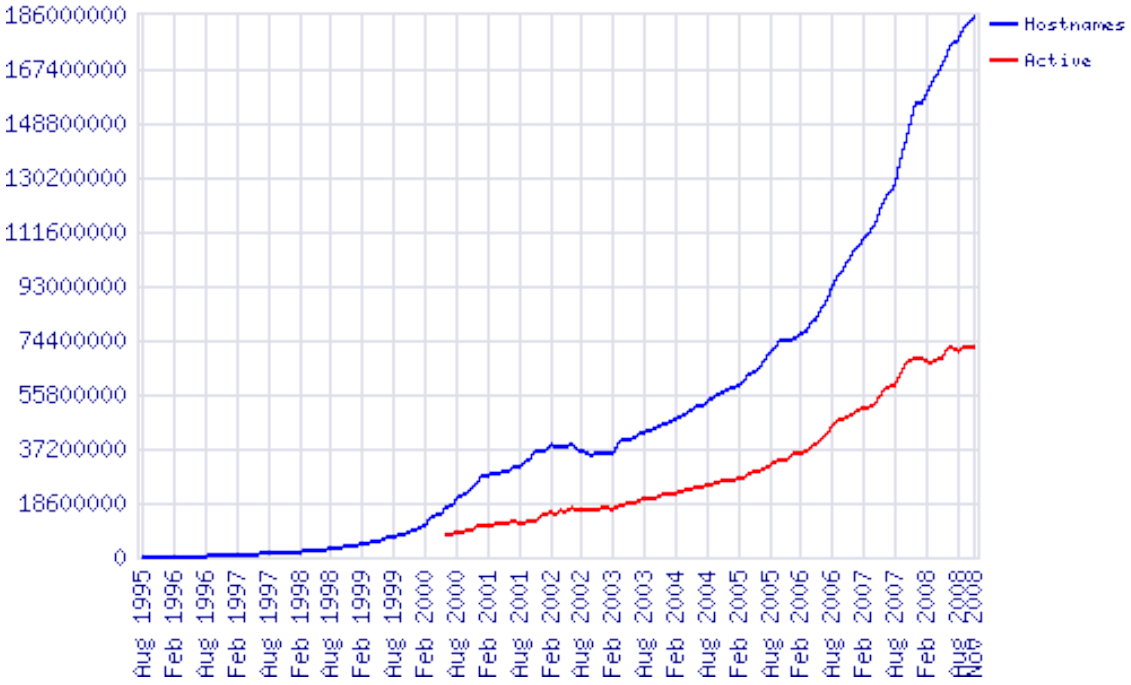


Abbildung 1.5: Anzahl der Webseiten im Internet, Stand: November 2008, Quelle: `http://news.netcraft.com/`

In der Abbildung zählt die blaue Kurve die Anzahl der bei den unterschiedlichen Internet-Registraren angemeldeten Hostnames. Da aber mehrere Adressen auf die gleiche Inter-

netseite verweisen können oder aber unter der registrierten Adresse gar keine Inhalte hinterlegt sind, ist die Anzahl der tatsächlich unterschiedlichen und aktiven Internet-Server kleiner (rote Kurve). Für eine Beschreibung der genauen Methode zur Ermittlung der aktiven Internet-Server sei auf `http://survey.netcraft.com/index-200007.html#active` verwiesen.
Mit der Hilfe des Internets war es nun möglich, Information ständig *aktuell* zu halten, und sie fast *universell zur Verfügung* zu stellen. Mit dieser „Universalisierung“ einher ging auch eine Liberalisierung der Informationsbereitstellung: die Verbreitung von Informationen war nun nicht mehr einigen wenigen „Versorgern“ wie Zeitungen, Radio und Fernsehen überlassen. Das WWW ermöglichte es jedem einzelnen, an der Informationsverbreitung teilzunehmen. Nicht zuletzt dadurch wurde die Menge an zur Verfügung stehenden Informationen unüberschaubar.
Verstärkt wurde dieser Trend durch das sogenannte *Ubiquitous* oder *Pervasive Computing*, das die Anzahl der Datenquellen noch weiter erhöhte.

1.3 Ubiquitous Computing

In der Frühzeit der Computertechnologie arbeiteten viele Personen an einem Rechner (Mainframes), dann bekam jeder Nutzer seinen eigenen Rechner, den „Personal Computer“ und schließlich arbeiten in der heutigen Zeit mehrere Computer mehr oder weniger sichtbar für eine einzelne Person. Als Beispiel seien hier nur Laptops, Mobiltelefone, MP3-Player und viele andere Geräte mit eingebetteten Mikroprozessoren genannt. Nicht nur, dass diese Geräte eine Vielzahl von Informationen verarbeiten können, sie erzeugen auch viel Daten oder ermöglichen zumindest den Zugriff auf sie, wenn er vorher nicht möglich war. Die allgegenwärtige Druchdringung unseres Alltags mit Computern ließ den Begriff des Ubiquitous Computing aufkommen.
Als Mark Weiser 1991 seine Idee vom *Ubiquitous Computing* darstellte [Wei91], von der Art und Weise, wie Computer unser Leben vollständig durchdringen werden, aber gleichzeitig unsichtbar werden, einerseits weil die Miniaturisierung sie tatsächlich so klein werden ließ, dass sie unsichtbar wurden, andererseits weil man sie nicht mehr wahrnehmen würde [Nor98], machte er schon darauf aufmerksam, dass der Begriff in seiner Vision mehrere Dimensionen umfasst:
Verteiltheit bezieht sich darauf, dass bei der Ausführung einer Anwendung nicht mehr nur eine Komponente beteiligt ist, sondern mehrere, die im Raum (und auch in der Zeit) verteilt sind, d. h. die – obwohl sie alle gemeinsam an der Erfüllung des Anwendungszwecks mitwirken – an verschiedenen Orten und zu verschiedenen Zeiten arbeiten.
Mit der Verteiltheit kommt auch *Heterogenität*: Geräte sind nicht gleichartig, sondern verfügen über unterschiedliche Ressourcen hinsichtlich Hardware, Kommunikationseinrichtungen, aber auch Datenformaten und Nutzerschnittstellen.

Eine zweite „Begleiterscheinung“ der Verteiltheit ist die Notwendigkeit zur Kommunikation: ohne eine Abstimmung untereinander ist Kollaboration nicht möglich. Eine Lösung für dieses Problem stellen offene Standards wie IEEE 802.11 WLAN [IEE07, IEE99, IEE05a, IEE], Bluetooth [IEE05b] u. ä. dar, mit deren Hilfe zumindest die technischen Grundlagen für einen Datenaustausch festgelegt werden. Ein besonderes Merkmal ubiquitärer Rechnernutzung ist die Tatsache, dass Geräte spontan miteinander kooperieren müssen, da sie mobil sind und eine Netzwerkinfrastruktur von vornherein nicht gegeben ist, sondern sich die Netzwerkknoten in dynamischer Art und Weise miteinander arrangieren. Durch die Mobilität und Dynamik ergeben sich neue Herausforderungen in allen Schichten der Kommunikationshierarchie einer verteilten Anwendung. Die Heterogenität der kooperierenden Geräte umfasst auch Unterschiede ihrer *Benutzerschnittstellen*. Neue Eingabe- und Ausgabemöglichkeiten werden mit den fortschreitenden Möglichkeiten der Technik realisierbar: Haptische Benutzerschnittstellen, virtuelle Realitäten und berührungssensitive Displays sind nur wenige der Mensch-Maschine-Schnittstellen, die Ubiquitous Computing erst möglich machen. Die Abbildungen 1.6 und 1.7 zeigen zwei Head-up-Displays in Fahrzeugen (die natürlich als rollende Rechner beste Beispiele für Ubiquitous Computing sind), die Informationen direkt auf die Windschutzscheibe eines Fahrzeugs werfen. Diese Displays sind sogar kontextsensitiv (was schwer im Bild darzustellen ist), da sich die Größe und Transparenz der angezeigten Grafik z. B. an die aktuelle Geschwindigkeit anpasst.

Abbildung 1.6: Head-Up-Display (Quelle: BMW AG)

Abbildung 1.7: Head-Up-Display (Quelle: Osram GmbH)

1.4 Die Probleme von World Wide Web und Ubiquitous Computing

Durch die allgegenwärtige Verfügbarkeit von Informationen aus einer unzählbaren Menge von Informationsquellen ergibt sich das Problem, dass ein Nutzer gar nicht mehr weiß,

welche Informationen für ihn relevant sind und wie er an sie kommen kann. Dazu ist er auf technische Unterstützung in Form von Filtern und Suchmaschinen angewiesen. Schwierig ist hierbei nur, dass diese Hilfsmittel auf den Menschen als Nutzer ausgerichtet sind[1]: er kann die Informationen einer Webseite leicht erfassen, transformieren und mit anderen für ihn relevanten Informationen verknüpfen: Maschinen können das nicht. Andererseits ist eine automatische Suche nach relevanten Informationen angesichts der Informationsfülle angebracht. Weiser hatte das Problem damals zwar noch nicht so deutlich artikuliert, hatte es aber in seinem eben schon angesprochenen Artikel [Wei91] bemerkt: er sprach vom Vorgang des Verstehens von Information als einer impliziten, unbewussten Absorption von Information. Er verwendete damals auch den von Michael Polanyi geprägten Begriff der „tacit dimension“, der *stillen Dimension* des Verstehens [Pol85]. Straßenschilder z. B. werden nicht gelesen, Ihre Information wird einfach ohne den Vorgang des Lesens wahrgenommen, gerade weil Menschen über die Fähigkeit verfügen, ihr implizites Wissen und die ihnen implizit innewohnenden Fähigkeiten anzuwenden. Auch die Einschätzung von Situationen aufgrund der über die Sinnesorgane wahrgenommenen Tatsachen fällt Menschen im Allgemeinen nicht schwer, er kann den *Kontext*, in dem er sich befindet, leicht erfassen. Rechner verfügen über diese Fähigkeiten nicht: ihnen muss man mit aufwendigen Verfahren wie neuronalen Netzen, Experten- und Wissensbasierten Systemen eine Art *künstlicher Intelligenz* beibringen.
Als Beispiel sei hier die Webseite zur Klärung des Begriffes „Jaguar“ auf Wikipedia gezeigt (Abbildung 1.8):
Ein Mensch, der auf der Suche nach Informationen über die Raubkatze Jaguar ist, wird sofort auf den ersten Link klicken. Ein Rechner, dem die Bedeutung, d. h. die *Semantik* hinter den Begriffen nicht geläufig ist, kann das nicht und würde vielleicht einen beliebigen und unpassenden Begriff wählen.
Tim Berners-Lee erkannte das Problem früh und schlug vor, auch die Semantik von Begriffen maschinenverarbeitbar zu *modellieren* und so für eine automatische Verarbeitung durch Computer zugänglich zu machen: er formulierte die Idee eines Semantic Web [BLHL01], welches in Kapitel 2.4 vorgestellt wird.

1.5 Zielsetzung dieser Arbeit

Das Semantic Web ist als Aufsatz für das stationäre WWW gedacht und bietet Werkzeuge an, Konzepte, ihre Semantik und Beziehungen untereinander zu modellieren, um die Probleme der Informationsheterogenität und Informationsintegration in den Griff zu bekommen. Da dieselben Probleme auch für (Kontext-)Informationen im Fahrzeug beste-

[1] Ob der beschriebene Sachverhalt problematisch ist, hängt natürlich vom Blickwinkel ab: natürlich ist es wünschenswert, wenn Suchmaschinen auf den Menschen als Nutzer zugeschnitten sind. Für die automatische Filterung und Erfassung von Sachverhalten durch Maschinen ist es natürlich hinderlich.

Abbildung 1.8: Wikipedia-Seite zum Begriff „Jaguar“

hen liegt es nahe, dieselben Techniken dazu zu verwenden, Kontextdaten im Fahrzeug zu modellieren und auch dort die Informationsflut zu bändigen. Kontext können Fahrzeuge über Sensoren wahrnehmen, die Daten aber zu einem Gesamtbild zusammenzusetzen, ist schwierig – man bedient sich verschiedener Modelle, die Kontextinformationen miteinander verknüpfen. Im Rahmen der Modellbildung muss aber derjenige, der das Modell erstellt, die Konzepte des Modells und ihre Beziehungen nachvollziehen und in einer maschinenlesbaren Form niederlegen, so dass ein Rechner dies rekonstruieren kann. Er muss also sein implizites Wissen (die schon angesprochene „tacit dimension“) externalisieren. Da Modelle i. A. auf eine bestimmte Anwendung zugeschnitten sind, muss also für jede Anwendung ein neues Modell geschaffen werden, da z. B. bestimmte Zusammenhänge im Modell für Anwendung A ausgespart wurden, für Anwendung B aber benötigt werden. Ein weiterer Abstraktionsschritt zur Findung anwendungsübergreifender Modelle wäre hier wünschenswert.
Das führt zu den folgenden Forschungsfragen:

- Genügen die Werkzeuge des Semantic Web den Anforderungen zur Kontextmodellierung in Fahrzeugen?
- Wie kann man die Kontext-Anforderungen von Anwendungen abstrahieren und in ein anwendungsübergreifendes Modell einfließen lassen?

- Wie mächtig hinsichtlich der Ausdrucksfähigkeit muss ein Kontextmodell mindestens sein?
- Wie mächtig darf ein Modell werden, um noch handhabbar zu sein?
- Welche besonderen Anforderungen ergeben sich durch die Randbedingungen in automobilen ad-hoc Netzen?
- Wie können Anwendungen vom Modell profitieren?

Ziel dieser Arbeit ist es, mit Semantic-Web-Techniken ein anwendungsübergreifendes Kontextmodell für mobile Anwendungen im Allgemeinen zu erstellen, dabei aber die besonderen Anforderungen, die sich durch das Einsatzgebiet ergeben, zu berücksichtigen. Dazu gehören vor allem die erhöhte Mobilität und Dynamik der Netzwerkteilnehmer, die dazu führt, dass sich der Kontext, in dem sie sich befinden, schnell ändert und die Tatsache, dass die Netzwerkbandbreite in drahtlosen Netzen stark begrenzt ist. Auch muss das Modell dem Umstand Rechnung tragen, dass eine Vielzahl von Anwendungen mit ganz unterschiedlichen Zielen sich seiner bedienen soll. Manche dieser Anwendungen stellen bestimmte Anforderungen an die Berechnungsgeschwindigkeit, d.h. Anfragen an das Modell müssen in engen Zeitrahmen beantwortet werden. Außerdem muss das Modell Erweiterungen und Überarbeitungen für das Hinzufügen und Verändern von Anwendungen zulassen. Mit der Hilfe des Kontextmodells können Applikationen auf eine Kontextbasis zugreifen, ohne dass jede Anwendung relevanten Kontext jeweils neu modellieren muss. Spezifische Erweiterungen und Veränderungen des Modells können von den Anwendungen selbst vorgenommen werden, um Konzepte in das Modell einzuführen, die auf Ihren eigenen Anwendungszweck zugeschnitten sind.

1.6 Inhalt und Aufbau

Diese Dissertation ist wie folgt aufgebaut:

Kapitel 2 stellt die Grundlagen dieser Arbeit vor. Es werden die Begriffe „Kontext“ und „Kontextbewusstsein“ definiert, technische Grundlagen von VANETs eingeführt und das Semantic Web – zumindest in Grundzügen – erläutert. Davon ausgehend werden Anforderungen an Kontextmodelle aufgestellt (Kapitel 3). Diese ergeben sich hauptsächlich aus einer Analyse der Fahraufgabe, deren Unterstützung eines der Hauptziele von VANET-Anwendungen ist. Weitere Anforderungen ergeben sich aus anderen Anwendungen aus dem Infotainment-Bereich und Besonderheiten bei der Markteinführung von VANET-Systemen. Kapitel 4 zeigt den aktuellen Stand der Forschung auf und Kapitel 5 stellt die KOMODE Basiskonzepte vor. Diese Konzepte werden in drei verschiedenen Anwendungen in Kapitel 6 zur Anwendung gebracht. Die mit Hilfe der Anwendungen simulativ

gemachten Erfahrungen werden in Kapitel 7 im Rahmen einer kurzen Evaluation zusammengefasst. Die Ergebnisse werden mit den vorher aufgestellten Anforderungen abgeglichen. Das letzte Kapitel – Kapitel 8 – rekapituliert die Ergebnisse dieser Arbeit und zeigt noch offene Fragen und Anknüpfungspunkte auf.

2 Grundlagen

Das folgende Kapitel soll die Grundlagen, auf die diese Arbeit aufbaut, kurz vorstellen. Es definiert wichtige Begriffe wie „Kontext“ und „Kontextbewusstsein“ (Context Awareness) und stellt die Besonderheiten von automobilen ad-hoc Netzwerken vor, die im Englischen vehicular ad-hoc networks (VANETs) genannt werden. Des Weiteren gibt das Kapitel einen tieferen Einblick in die Möglichkeiten und Techniken des Semantic Web und stellt dessen prominentestes Modellierungswerkzeug, die Ontologien, vor.

2.1 Kontext

Wenn man der ursprünglichen Bedeutung des Wortes „Kontext“ nachspürt, muss man die lateinische Sprache bemühen: das Verb „contexo“ bedeutet soviel wie „ich webe zusammen“ oder „ich setze zusammen“, das entsprechende Partizip „contextus“ soviel wie „verflochten“. So richtig die Anwendung der ursprünglichen Bedeutung auch heute in der Informatik noch ist, denn um nicht anderes als die Verflechtung und In-Beziehung-Setzung von Informationen geht es nach wie vor, so wenig hilfreich ist sie, wenn Begriffe wie Nutzerkontext im Sinne einer technischen Verwertbarkeit eindeutig gefasst werden müssen.
Wenn man sich von dieser ursprünglichen Bedeutung löst, und Kontext als „Kon Text“, also „mit Text“ auffasst, dann wird deutlicher, dass Kontext etwas ist, das parallel zu geschriebenen oder gesprochenen Texten existiert: erst durch die Interpretation eines Textes durch Personen kann ihm eine mehr oder weniger eindeutige Bedeutung zugeordnet werden, d.h. die Bedeutung eines Textes wird erst durch die Verknüpfung des Textes selbst mit seinem Kontext, in dem er gesprochen oder niedergeschrieben wurde, deutlich.
In der Informatik ist Kontext nicht mehr nur in der Verbindung mit Texten zu finden: Nutzer, Anwendungen, Geräte usw. befinden sich in Kontexten. Daraus ergibt sich die Notwendigkeit, neu zu definieren, was Kontext bedeutet, denn in [DAS01] stellen Dey et al. fest, dass es keine hinreichende Definition von (informatischem) Kontext gibt:

> (a) the notion of context is still ill defined, (b) there is a lack of conceptual models and methods to help drive the design of context-aware applications, and (c) no tools are available to jump-start the development of context-aware applications.

In der Frühzeit adaptiver Systeme wurde der Begriff „Kontext“ oft gleichgesetzt mit Ort

und Zeit, einfach weil das die einzigen beiden Informationen waren, die ohne großen Aufwand zur Verfügung standen. Dennoch stellten schon Schilit et al. in ihrer Kontextdefinition von 1994 [SAW94] fest, dass das nicht ausreichend ist:

> Three important aspects of Context are: where you are, who you are with, and what resources are nearby. Context encompasses more than just the user's location, because other things of interest are also mobile and changing.

Im weiteren Verlauf des Artikels wird ein System namens PARCTab vorgestellt, das sich auf der Basis von Infrarotsensoren dennoch „nur“ an den Aufenthaltsort des Nutzers anpassen kann. Vorausgesetzt, das System wüsste, welches PARCTab welchem Nutzer zuzuordnen wäre, wären noch weitere nutzerspezifische Anpassungen denkbar.
Die Unzulänglichkeit einer Kontextdefinition, die nur den Ort umfasst, wird auch von Brown et al. [BBC97] bemerkt:

> In general, the information may be tailored to several aspects of the user's context: location, time of day, season of the year, temperature, and so forth.

Diese Definition bezieht auch mehrere Daten in den Kontext mit ein. Die Formulierung „and so forth“ legt nahe, dass hier noch mehr Daten folgen könnten. Sie zeigt auch, dass eine Definition von Kontext offen bleibt und dass die zum Kontext gehörenden Daten veränderlich sind. Auch hier ist aber zu bemerken, dass die im zitierten Artikel vorgestellte Anwendung im Prinzip nur Orts- und Zeitinformationen verarbeitet. Hinzu kommt noch ein Datum das die Orientierung des Nutzers im Raum beschreibt.
Die Definition von Pascoe et al. [PRM98] geht in eine ähnliche Richtung, zählt aber nur andere Daten als „Kontext“ auf: Der Ort des Nutzers, seine Umgebung, Daten zu seiner Identität und die Zeit, ähnlich wie eine frühere Definition von Dey [Dey98], die dieser Definition noch den emotionalen Zustand des Nutzers und den Gegenstand seiner Aufmerksamkeit hinzufügt. Zur Umgebung zählt Dey auch andere Personen und Objekte in der Nähe des Nutzers.
Da der Versuch, einfach alles aufzuzählen, was Kontext sein könnte, augenscheinlich nicht zielführend ist, nimmt Winograd in seiner Definition [Win01] einen anderen Blickwinkel ein:

> Context is an operational term: Something is context because of the way it is used in interpretation, not due to its inherent properties.

Damit schlägt er den Bogen zur oben vorgestellten wortwörtlichen Bedeutung von Kontext: Etwas ist Kontext, weil es in der Interpretation [von Information] verwendet wurde, und nicht weil es bestimmte Eigenschaften aufweist. Problematisch im Sinne einer informatischen Verwendung ist die Tatsache, dass erst im Nachhinein klar ist, was der Kontext einer Information ist. Die Definition macht aber klar, dass jeder, der den Versuch unternimmt,

Kontext in Rechensystemen zu modellieren, für die Offenheit seiner Modelle sorgen muss, um zu einem späteren Zeitpunkt weitere Kontextdaten aufnehmen zu können. Gleichzeitig stellt Winograd weiter fest:

1. Context is not just more text. [...]
2. Context is effective only when it is shared. [...]
3. Context emerges in dialog. [...]

Daraus wiederum folgt, dass auch Kontextmodelle inhärent einer Verteilung unterliegen müssen, so dass die Interpretation von Information in ihrem Kontext durch unterschiedliche Personen oder Rechner zu (annähernd) gleichen Ergebnissen ohne Missverständnisse kommt, es muss also ein gemeinsames („shared“) Verständnis von Kontext geben. Dieses Verständnis kommt nur durch Austausch („dialog“) zu Stande: auf die Ebene von Kontextmodellen übertragen bedeutet das, dass es Möglichkeiten geben muss, sie auszutauschen. Im Sinne einer maschinellen Verarbeitbarkeit ist die folgende Definition nach Dey [Dey01] praktikabler, weil eher greifbar:

> Context [is] any information that can be used to characterize the situation of entities (i.e., whether a person, place, or object) that are considered relevant to the interaction between a user and an application, including the user and the application themselves. Context is typically the location, identity, and state of people, groups, and computational and physical objects.

Sie widerspricht der Definition von Winograd durch ihre Offenheit nicht wirklich, gibt aber konkreten Aufschluss darüber, welche Informationen als Kontext zu betrachten sind, und in welchen Domänen man nach weiteren Einheiten („entities“) suchen kann, die einem Kontext unterliegen, nämlich:

- interaction between a user and an application
- users
- applications

Im weiteren Verlauf dieser Arbeit folgen wir der Definition von Dey. Spätere Definitionen von Kontext fußen i. A. auf dieser Definition. Als Beispiel sei noch die Definition von Strang [Str04] gegeben:

> Eine Kontextinformation ist eine Information, die dazu benutzt werden kann, den Zustand einer Entität bzgl. eines Aspekts zu charakterisieren. Sie ist gegeben durch ein Element des Wertebereichs eines Aspekts. Eine Kontextinformation ist relevant für eine Aufgabe, wenn sie die Belegung einer Zustandsvariable

> eines relevanten Aspekts ist, der zu einer für diese Aufgabe relevanten Entität gehört.

und an anderer Stelle:

> Ein Kontext ist die Menge aller für eine Aufgabe relevanten Kontextinformationen.

Sie erweitert die Deysche Definition um den Begriff der „Kontextinformation“ (im Gegensatz zum allgemeineren Kontext), mit dem einzelnes Kontextdatum bezeichnet wird. Details zu den Begriffen „Entität“ und „Aspekt“ in dieser Definition können unter [Str04, S. 20 ff.] nachgelesen werden. Betont wird hierbei auch noch die Relevanz der Kontextinformation für eine Aufgabe. Diese Einschränkung fällt in der Definition von Derowski [Der06] (zitiert aus [Str07]) weg:

> Kontext ist jede Information, die den Zustand und die Umgebung, sei sie räumlich, zeitlich oder sozial, einer Entität, sowie die Entität selbst beschreibt oder spezifiziert.

Diese Definition steht bei genauer Betrachtung nicht im Gegensatz zu den vorherigen von Dey und Strang, wenn man Aufgaben als Entitäten betrachtet. Diese Sichtweise ist auch durch die Tatsache gedeckt, dass Anwendungen (die in Deys Definition sehr wohl in einem Kontext verwendet werden) immer zur Lösung einer bestimmten Aufgabe verwendet werden.
Kontext kann bei einigen Autoren noch weiter in zusätzliche Kategorien unterteilt werden. In [Rot05] und [DRD+00] wird Kontext differenzierter betrachtet:

- *Infrastrukturkontext* fasst alle Kontextvariablen zusammen, die mit der Kommunikationsinfrastruktur verbunden sind. Üblicherweise sind das Netzwerkbandbreite, -verzögerung und die Zuverlässigkeit. Von den oben erwähnten Autoren nicht explizit angesprochen, aber durchaus auch unter diesem Punkt einsortierbar ist auch das technische Mittel, mit dem auf ein Netzwerk zugegriffen wird, z. B. Zugriff über GSM, UMTS, WLAN oder andere Möglichkeiten

- *Systemkontext* betrifft die Verteilung einer mobilen Anwendung. Wenn andere Komponenten als das unmittelbar durch den Nutzer bediente Gerät an der Ausführung einer Anwendung beteiligt sind bezieht sich der Begriff auf den Zustand des Gesamtsystems.

- *Domänenkontext* stellt in [Rot05] „eine Beziehung zwischen den Geräten und ihren Benutzern auf der Basis der Anwendungsdomäne her.“ Die Definition ist etwas

schwammig und ist in [DRD+00] genauer gefasst: Er modelliert die Semantik von Konzepten einer bestimmten Anwendungsdomäne. Die im Artikel gemachte Einschränkung, dass sich das hauptsächlich auf die Interaktionsmöglichkeiten zwischen Nutzer und Gerät bezieht, ist eigentlich gar nicht vonnöten.

- *Physikalischer Kontext* bezeichnet die konkreten Umweltbedingungen, in denen sich ein Nutzer befindet oder ein Gerät eingesetzt wird. Üblicherweise sind das einerseits wenig veränderliche Informationen wie z. B. die Abmessungen und Beschaffenheit von Geräten, andererseits alle Daten, die von Sensoren für physikalische Messgrößen erfasst werden können.

Eine weitere Unterscheidung des Kontextes wird dahingehend getroffen, ob er unmittelbar von einem physikalisch vorhandenen Sensor ermittelt wurde oder ob weitere Vorverarbeitungsschritte nötig waren, um aus Sensordaten den Kontext zu ermitteln bzw. ob er aus anderen Daten logisch erschlossen wurde.
Niederwertiger Kontext (so bezeichnet bei [Str07] und [Dey00]) oder primärer Kontext [Str04] ist Kontext, der unmittelbar von Sensoren ermittelt werden kann. Er entspricht in dieser Definition also dem physikalischen Kontext aus [Rot05] und [DRD+00]. Wenn sich aus diesen Informationen durch logisches Schließen, Aggregieren oder anderen Methoden neue (Kontext-)Information ableiten lässt, spricht man von *höherwertigem* ([Str07] und [Dey00]) oder *sekundärem Kontext* [Str04]. Eine weitere Unterteilung des Kontexts findet sich in [Str07], verdeutlicht durch Abbildung 2.1:

- *Abstrahierte Sensorinformationen* versehen einzelne Messwerte mit einer Skala: GPS-Geräte z. B. ermitteln die Geschwindigkeit, indem sie die Frequenzverschiebung des GPS-Trägersignals berechnen (Doppler-Effekt) und daraus die eigene Geschwindigkeit über Grund kalkulieren. Sie bilden also Frequenzverschiebungen auf Geschwindigkeiten ab und versehen sie mit einer entsprechenden Skala (in diesem Fall km/h oder m/s) was für den Menschen besser lesbar ist. Diese abstrahierte Sensorinformation in Kombination mit einer semantischen Anreicherung (nämlich durch Angabe einer Skala) nennt [Str04] auch *Aspekt*. Da die Ausgabedaten von Sensoren im Allgemeinen nicht standardisiert oder für Menschen nicht unmittelbar verständlich sind, macht eine Vereinheitlichung und weitere Veredelung dieser Daten Sinn. Dadurch wird aber keine neue Information erzeugt, sondern die ursprünglichen Daten werden nur in eine andere Form transformiert. Sie sind deshalb weiterhin niederwertiger Kontext.

- *Aggregrierte Sensorinformation* liegt dann vor, wenn unterschiedliche Messwerte mit statistischen Methoden wie z. B. Durchschnittsbildung oder Ermittlung des Median o. ä. zusammengefasst werden.

- *Abgeleiteter Kontext* ist jeglicher Kontext der mit Methoden des logischen Schließens ermittelt wurde.

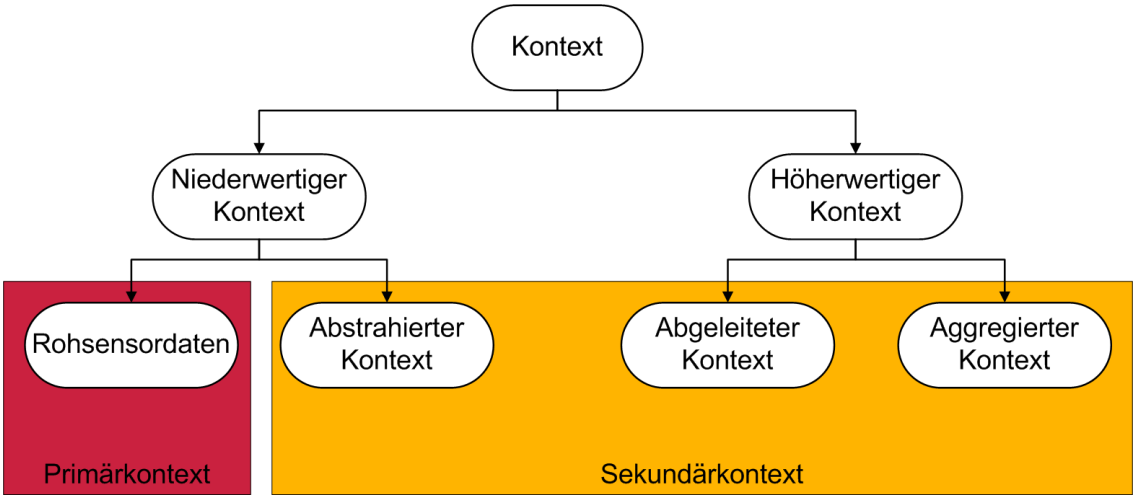


Abbildung 2.1: Klassifizierung von unterschiedlichen Kontexttypen nach [Str07] und [Str04]

Die Verwendung von Kontext durch Anwendungen ist ein zentrales Konzept von adaptiven (mobilen) Systemen. Gerade der unterschiedliche Kontext, in dem eine Anwendung verwendet macht den Unterschied aus: adaptive Anwendungen sind nicht einfach Anwendungen, die aus statischen Umgebungen auf mobile Geräte kopiert wurden, sondern Anwendungen, die für eine Anpassung an verschiedene – durch die Umwelt, den Ort, die Zeit oder den Nutzer gegebene – Aspekte angepasst wurden. Solche Anwendungen nennt man kontextadaptiv oder auch kontextbewusst („context aware“), ein Begriff, den näher zu fassen im nächsten Abschnitt versucht wird.

2.2 Kontextbewusstsein (Context Awareness)

Anwendungen werden als *context aware*[1] bezeichnet, wenn sie imstande sind, den Kontext (in allen vorher beschriebenen Dimensionen), in dem sie genutzt werden, zu erfassen und ihre eigene Arbeitsweise, d. h. ihre internen Berechnungen, die Präsentation oder auch Aktivierung, an ihn anzupassen. Schilit und Theimer [ST94] definieren den Begriff „Context Awareness“ wie folgt:

> Location information is necessary for users and applications that want to query and interact with nearby devices and services. Such information also allows

[1] im Deutschen hat sich „Kontextbewusstsein“ als Übersetzung für „Context Awareness“ nicht durchgesetzt. Die Arbeit verwendet daher weiter die englische Bezeichnung.

> stationary clients to track moving objects. In general, location information enables software to adapt according to its location of use, the collection of nearby people and objects, as well as the changes to those objects over time. We use the term context-aware computing to describe software exhibiting these general capabilities.

Ähnlich wie schon zuvor bei der Definition von Kontext bezieht sich die Anpassungsfähigkeit nur auf Ortsinformation, eine weitere Fassung des Begriffes findet sich dann später bei Chen und Kotz [CK00]. Dabei wird zusätzlich noch zwischen einer *aktiven* und *passiven* context awareness unterschieden:

- *Active context awareness*: an application automatically adapts to discovered context, by changing the application's behavior.
- *Passive context awareness*: an application presents the new or updated context to an interested user or makes the context persistent for the user to retrieve later.

Dieselbe Einschränkung, nämlich dass Kontextbewusste Anpassungen auch von einer gewissen Relevanz für den Nutzer sein müssen, wobei diese Relevanz vom Ziel des Nutzers abhängig ist, die Dey schon in seiner Definition von Kontext machte, wiederholt er auch bei seiner Definition von context awareness [Dey00]:

> A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Die eben genannte Definition ist die, auf die sich diese Arbeit stützt. Die spätere Definition von Strang [Str04], fügt eigentlich nichts Wesentliches mehr hinzu:

> Eine System ist kontextadaptiv (context aware), wenn es den Kontext vor oder während der Erfüllung einer Aufgabe berücksichtigt.

2.3 Automobile ad-hoc Netzwerke (VANETs)

Automobile ad-hoc Netze bilden das Einsatzumfeld für die in dieser Arbeit entwickelten Kontextmodelle. Sie weisen im Gegensatz zu fest verdrahteten Netzen einige Besonderheiten auf und bieten eine Grundlage für neue Anwendungen. Auf die Eigenheiten dieser Netze soll in den nächsten Abschnitten nach einer Beschreibung eines einführenden Szenarios eingegangen werden.

2.3.1 „Kazaa auf der Autobahn“

„Kazaa auf der Autobahn – Jeder für sich war gestern: In Zukunft sollen alle Autofahrer auf Deutschlands Straßen per Funk miteinander verbunden sein. Bei Unfall, Glatteis oder

Stau leitet jeder Warnhinweise an andere Verkehrsteilnehmer weiter - ohne Zentralrechner und Satelliten.“ ([Don07])

So begann im Juli 2007 ein Zeitungsartikel, der verdeutlichen sollte, was Fahrzeuge in Zukunft können sollten, sobald sie mit Kommunikationsvorrichtungen, die einen gegenseitigen Nachrichtenaustausch ermöglichen, ausgestattet sind. Mannigfache Anwendungen sind mit VANETs denkbar: eine gegenseitige Warnung bei Gefahren, bei uneinsehbaren Stauenden und schlechten Sichtverhältnissen. Auch eine kollaborative Optimierung des Verkehrsflusses zur besseren Auslastung und Vermeidung einer Überlastung von Straßen ist möglich. Denn der Passus „Jeder für sich war gestern“ zeigt schon, dass es um mehr als nur einfachen Datenaustausch geht: Ziel ist eine echte Kooperation der Fahrzeuge untereinander, um gemeinsam Situationen detektieren zu können, die sich einem einzelnen Fahrzeug nicht erschließen können, sei es, weil das eigene Fahrzeug nicht mit den entsprechenden Sensoren ausgestattet ist, oder weil sich das Fahrzeug noch gar nicht an der betroffenen Stelle im Straßenverlauf befindet. Abbildung 2.2 zeigt ein solches Anwendungsbeispiel.

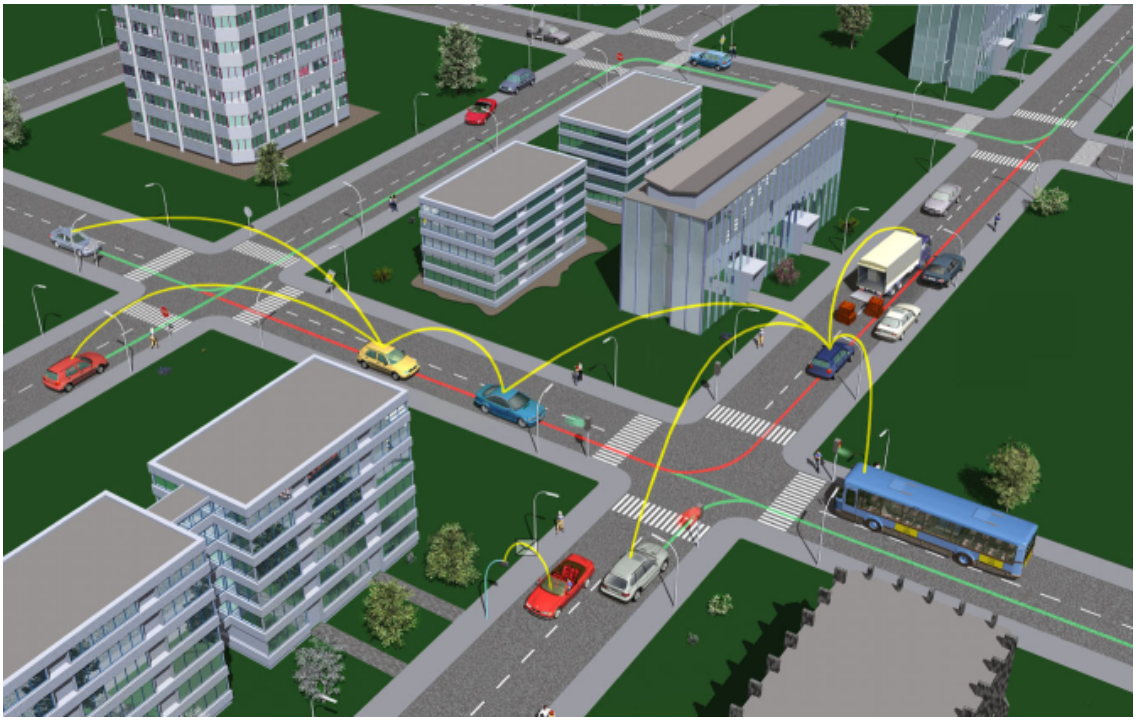


Abbildung 2.2: Spontane Vernetzung von Fahrzeugen (VANET) [Kos05]

Um dem in der Abbildung skizzierten Szenario näherzukommen, bemühen sich Wissenschaft und Industrie seit längerem in verschiedenen Kooperationsprojekten um die Erforschung zahlreicher Aspekte von VANETs. Die Fortschritte auf diesem Gebiet sollen im

nächsten Abschnitt kurz skizziert werden.

2.3.2 Historische Entwicklung von VANETs

Das erste Projekt, das sich mit der Kommunikation von Fahrzeug zu Fahrzeug beschäftigte, war das 1987 gestartete Projekt PROMETHEUS (Programme for a European Traffic of Highest Efficiency and Unprecedented Safety, [Bra95]), das bis 1995 dauerte und im Rahmen des Eureka-Programms gestartet wurde. Die zentrale Rolle spielten Fragen der Verkehrssicherheit und eine der dabei eingesetzten Methoden war die Fahrzeug-Fahrzeug-Kommunikation. PROMETHEUS diente – seinem Namen entsprechend[2] – als Ausgangspunkt vieler Nachfolgeprojekte, die einzelne Teilaspekte des Ursprungsprojekts weiter vertieften:

- Inter-Vehicle Hazard Warning (IVHW) [Che02] war in den Jahren 2001/2002 ein deutsch-französisches Projekt, das die Übermittlung von Warnnachrichten zwischen Fahrzeugen im 869 MHz-Band erforschte. Die niedrige Basisfrequenz ermöglichte Reichweiten bis zu 1 km. Nachrichten wurden nur per Broadcast übermittelt und Empfänger konnten selbst entscheiden, ob sie eine empfangene Nachricht erneut aussenden wollten.

- FleetNet [FEL01] – Internet on the Road wurde in den Jahren 2000 bis 2003 durchgeführt, um Algorithmen und Kommunikationsprotokolle für den Datenaustausch über mehrere Fahrzeuge hinweg (Multi-Hop-Verfahren) entwickeln, und zwar mittels eins ad-hoc Funknetzes, d. h. ohne Infrastruktur. Besonderer Schwerpunkt waren dabei neue Routingkonzepte. Ein weiterer Fokus des Projektes war mehr die Entwicklung einer Kommunikationsplattform als einzelner Anwendungen. Diese Plattform sollte so gestaltet werden, dass sie sowohl Anwendungen aus dem Bereich aktive Sicherheit als auch Informationsanwendungen ohne direkten Sicherheitsbezug unterstützt. FleetNet untersuchte drei mögliche Kandidaten für die technische Umsetzung: UTRA-TDD (ein Nebenstandard zu UMTS), Radarsysteme im 24-GHz-Bereich, die zur Kommunikation eingesetzt werden sollten und IEEE 802.11-WLAN-Systeme, die letztendlich verwendet wurden. Abbildung 2.3 zeigt den FleetNet-Demonstrator.

- Das Ziel von CarTALK2000 [RMM+02] war das Design, die Entwicklung und der Test von kooperativen Fahrerassistenzanwendungen, die der Fahrzeug-zu-Fahrzeug-Kommunikation basieren. Die Sicht von CarTALK2000 auf das Problem war ganzheitlich: neben Anwendungen im Fahrzeug und der Entwicklung der darunterliegenden Protokolle und Technik wurden im Projekt auch juristische und wirtschaftliche Aspekte der Fahrzeug-zu-Fahrzeug-Kommunikation betrachtet. CarTALK2000 führte wie FleetNet Untersuchungen zur Einsetzbarkeit von UTRA-TDD durch. Letzt-

[2] der griechischen Sage nach brachte PROMETHEUS den Menschen das Feuer

Abbildung 2.3: Der FleetNet Demonstrator, Quelle [Fra04]

endlich wurden WLAN-Technologien eingesetzt. Das Projekt lief von 2001 bis 2004 und wurde von europäischen Partnern durchgeführt.

- Softnet [BRS+02] zielte mehr auf neue Methoden des Software Engineering für die Erstellung von verteilten mobilen Systemen, in Speziellen im automobilen Umfeld. Ein Schwerpunkt war die Erforschung von Verfahren für die Entwicklung von Diensten in einem derart heterogenen Umfeld, abhängig von der Kommunikationstechnologie. Unter anderem wurden Routing-Verfahren für VANETs in Innenstadt-Szenarien entwickelt, Basistechnologie war IEEE 802.11 WLAN.

- INVENT war 2001 bis 2005 ein hauptsächlich anwendungsgetriebenes Projekt: Ziel war die Verbesserung des Verkehrsflusses, z.T. auch unter Verwendung von Fahrzeug-zu-Fahrzeug-Kommunikation, um z. B. den Fahrer über Stausituationen aufklären zu können und so zu ihrer Auflösung beizutragen. Als Funktechnologien kamen sowohl WLAN als auch zelluläre Netze zum Einsatz.

- Ziel des NoW-Projekts (Network on Wheels) [FNS+08] war die Spezifikation einer auf IEEE 802.11 basierenden Kommunikationsplattform, die es ermöglichen sollte, Sensordaten und allgemeine Informationen von Fahrzeug zu Fahrzeug auszutauschen. Die entwickelten Protokolle sollten Eingang in ein Referenzsystem und

einen Standard finden. Das im Jahr 2004 begonnene Projekt endete 2008 mit einem Demonstrator, der auch die im Projekt entwickelten Anwendungen zeigte. Das System unterstützt dabei Anwendungen aus dem Bereich aktive Sicherheit als auch Infotainment-Anwendungen. Letztere sind nötig geworden, da im Projekt durchgeführte Studien zeigten, dass sie zur Markteinführung eine Fahrzeug-zu-Fahrzeug-Kommunikationssystems unabdingbar sind. Weiterhin ist das System imstande, auch fest installierte Funkstationen in sein Netz zu integrieren. Abbildung 2.4 zeigt die in NoW vorgeschlagene Systemarchitektur.

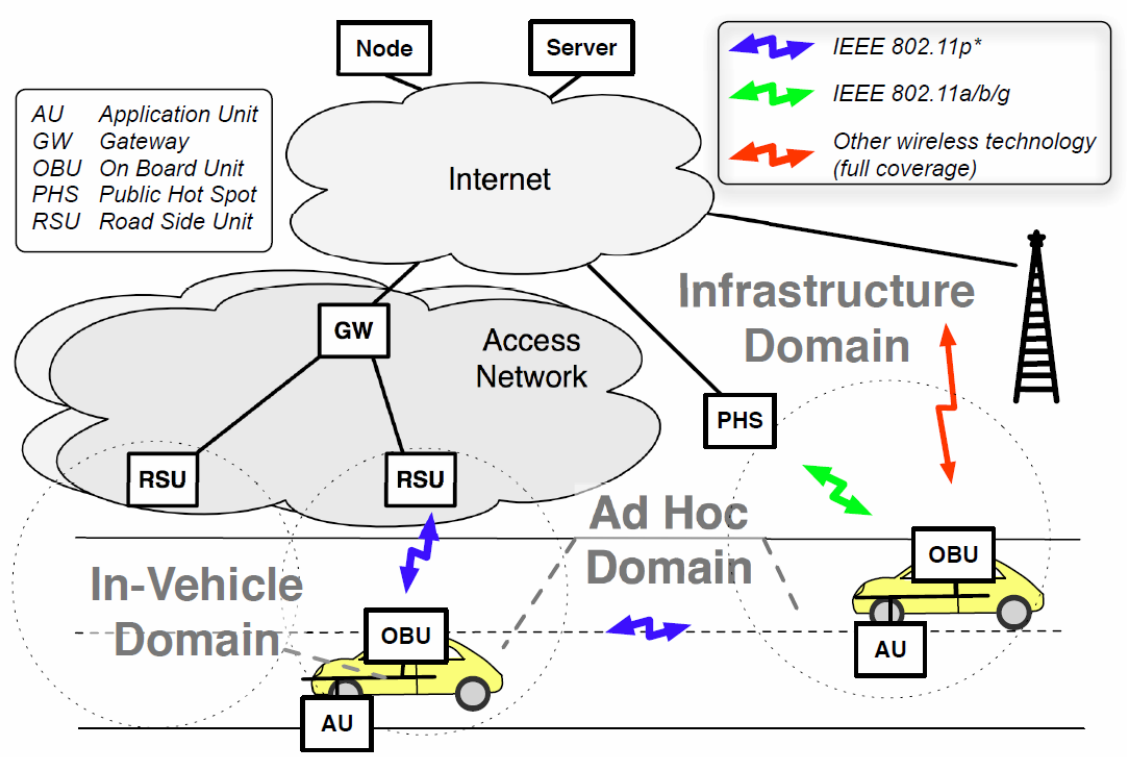


Abbildung 2.4: NoW Systemarchitektur, Quelle [FNS+08]

- PReVENT [SNB05] zielt darauf, die Anzahl der Todesfälle im Straßenverkehr bis 2010 (ein Ziel der Europäischen Kommission) um die Hälfte zu reduzieren. Dies soll mit der Hilfe von Fahrerassistenzsystemen gesehen, die z. T. auch auf Fahrzeug-zu-Fahrzeug-Kommunikation basieren. Fokus des Projektes ist eine frühzeitige Erkennung von Gefahren und rechtzeitige Warnung des Fahrers bei gleichzeitiger Einbeziehung des aktuellen Fahrverhaltens des Fahrers. Im Teilprojekt WILLWARN [HHSV07] ist die Übermittlung von Warnnachrichten über ein VANET im Fokus. Das Projekt lief von 2004 bis 2008.

- SIM-TD ist ein Projekt, das zum ersten Mal die Einsetzbarkeit und Wirksamkeit von VANETs im großen Maßstab überprüfen soll. Im Rhein-Main-Gebiet soll dazu eine große Flotte von Fahrzeugen mit VANET-Kommunikationssystemen ausgestattet werden. Kommunikation findet dabei nicht nur zwischen Fahrzeugen statt, sondern auch zwischen fest installierter Infrastruktur. Ziel ist die Erhöhung der Verkehrssicherheit und eine Verbesserung des Verkehrsflusses. SIM-TD startete im November 2008 und soll vier Jahre lang laufen.

2.3.3 Besonderheiten der Netzwerkarchitektur

Die im vorigen Kapitel angesprochenen Projekte haben einige Aspekte der Vernetzung in VANETs bearbeitet und neue Lösungsansätze für viele Probleme geliefert. Ein Teil von ihnen wird in den nächsten drei Abschnitten näher erläutert, wobei sich der Überblick auf die unteren drei Schichten des ISO/OSI-Referenzmodells (siehe Abbildung 2.5) beschränkt. Die Anwendungen (ISO/OSI-Schicht 7) in VANETs werden in Kapitel 3 vorgestellt, da aus ihnen die Anforderungen für die zugrunde liegenden Kontextmodelle abgeleitet werden.

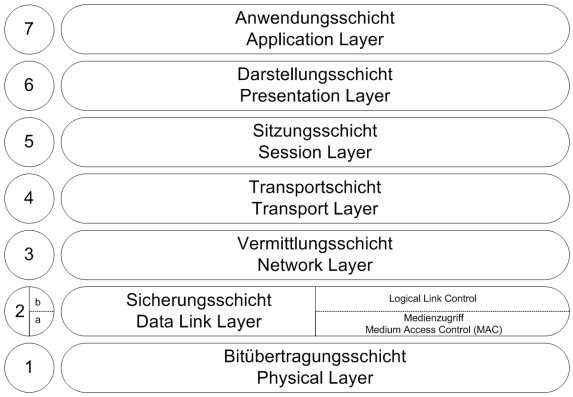


Abbildung 2.5: Das ISO/OSI-Referenzmodell für Netzwerkarchitekturen

1. Bitübertragungsschicht (Physical Layer)
2. Sicherungsschicht (Data Link Layer): Diese Schicht ist unterteilt in zwei Unterebenen: Medienzugriff (Medium Access Control, MAC) und Logical Link Control (LLC). In dieser Arbeit wird nur der Medienzugriff betrachtet, da sich die in Kapitel 6 vorgestellten Anwendungen nur auf diese Schicht auswirken.
3. Vermittlungsschicht (Network Layer)

Die Schichten 4-6 werden in der Betrachtung ausgespart, da sie für diese Arbeit nicht relevant sind. Die Anwendungen (Schicht 7) in VANETs werden gesondert in Kapitel 3

vorgestellt, da aus ihnen unmittelbar Anforderungen für die Kontextmodelle abgeleitet werden.

2.3.3.1 Bitübertragungsschicht

Auf der physikalischen Ebene unterscheiden sich VANETs wenig von herkömmlichen WLAN-Technologien. Der für Fahrzeugnetze ausschlaggebende Standard ist IEEE 802.11p [IEE][3], der wiederum auf IEEE 802.11a [IEE99] – einer PHY-Spezifikation für „normale“ drahtlose Netze – basiert. Der Standard ist aus einer Arbeitsgruppe namens WAVE (Wireless Access in Vehicular Environments) in den Jahren 2003/2004 hervorgegangen. Der Standard reserviert eigens für Fahrzeug-zu-Fahrzeug-Anwendungen ein 75 MHz breites Frequenzband von 5,850-5,925 GHz. Das Spektrum wird aufgeteilt in 7 Kanäle á 10 MHz, in Kombination mit OFDM (Orthogonal frequency-division multiplexing) als Modulationsverfahren führt diese Aufteilung in logische Kanäle zu einer Abschwächung von Multipath-Ausbreitungsphänomena. Abbildung 2.6 zeigt die Aufteilung.

Die Abbildung zeigt, dass zwei Dienst-Kanäle (in diesem Beispiel die Kanäle 180 und 182) nach Bedarf zusammengelegt werden können, um so höhere Bandbreiten erzielen zu können. Die Aufteilung auf mehrere logische Bänder führt dazu, dass bei Einsatz von nur einem Hardwareempfangsteil für Senden und Empfangen ein Frequenzsprungverfahren eingesetzt werden muss. Auf einem Kontrollkanal sind dann Ankündigungen zu empfangen, ob und auf welchem Dienstkanal welche Dienste zu empfangen sind. Dieser Kontrollkanal muss regelmäßig abgefragt werden. Laut Standard soll mit diesen Maßnahmen ein Datendurchsatz zwischen 3 und 27 Mbps erzielt werden können, die Latenz soll unter 50 ms liegen und die Reichweite bis zu 1 km.

2.3.3.2 Medienzugriff

Die Details des Medienzugriffs sind in den Standards [IEE07, IEE99, IEE05a, IEE] festgelegt und werden hier analog zu [KR03] kurz beschrieben. Weil der Zugriff auf das Medium bei drahtlosen Netzen immer nur exklusiv geschehen kann, muss er reguliert werden, d. h. eine sendewillige Station hört den Kanal ab, bevor sie selbst Daten senden will und prüft somit, ob er nicht schon durch eine andere Station belegt ist. Falls dies der Fall ist, dann muss sie warten. Dieses Verfahren nennt man Carrier Sense Multiple Access (CSMA).

[3] Die finale Version des Standards ist noch nicht erschienen, die Veröffentlichung ist für April 2009 geplant. Die hier gemachten Angaben stammen aus Vorabinfomationen aus Forschungsprojekten und schon getätigten Frequenzallokationen bei der FCC (Federal Communications Commission). Diese Allokationen sind bisher nur in den USA verbindlich. Über eine Allokation in Europa wird z. Zt. beim zuständigen Gremium (European Telecommunications Standards Institute, ETSI) noch verhandelt.

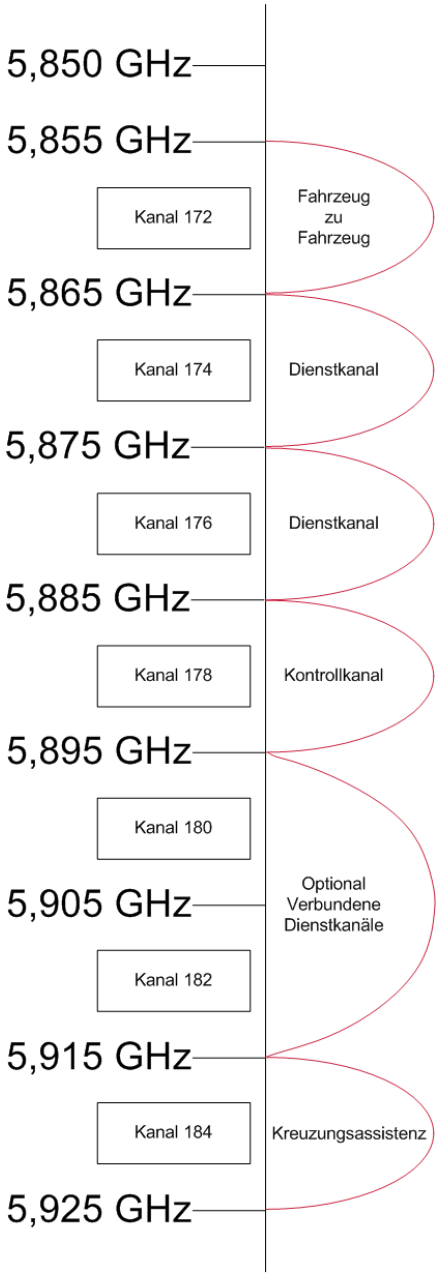


Abbildung 2.6: Die Frequenzaufteilung des IEEE 802.11p Standards

Um sicherzustellen, dass nach dem Ende einer erfolgten Übertragung nicht sofort alle wartenden Stationen zeitgleich beginnen zu senden, was zu Kollisionen führen würde, muss eine sendebereite Station noch einen gewissen Zeitraum lang warten. Dieser Zeitraum heißt Distributed Interframe Spacing (DIFS). Wenn der Kanal während des Ablaufs des DIFS von keinem anderen Netzwerkknoten belegt wird, kann die eigene Nachricht übertragen werden. Nach erfolgreicher Übertragung wartet der Empfänger der Nachricht einen kurzen

Zeitraum (die Pause wird mit Short Interframe Spacing (SIFS) bezeichnet) und bestätigt den Empfang der Nachricht. DIFS und SIFS sind konstante Werte, aber abhängig vom jeweiligen Übertragungsmedium bzw. dem Frequenzband, in dem gesendet wird. im IEEE 802.11a-Standard ist DIFS definiert als DIFS = SIFS + 2 · *Länge eines Zeitschlitzes*. Mit den Werten aus Tabelle 2.1 gilt also DIFS = 16µs + 2 · 9µs = 34µs. Die Bestätigung ist nötig, da im Gegensatz zu fest verdrahteten Netzen in drahtlosen Netzen nur der Empfänger und nicht auch der Sender wissen kann, ob eine Nachricht korrekt übermittelt wurde. Sollte der Kanal doch während des DIFS belegt werden, so muss die eigentlich sendewillige Station noch eine zusätzliche Wartepause einlegen und danach die Sendung wiederholen. Die Länge dieser zusätzlichen Pause wird zufällig gewählt, und zwar aus einem Intervall $[0, CW]$, wobei die maximale Wartezeit CW als Contention Window bezeichnet wird. Mit dem Zufallswert wird ein Zähler, der sogenannte Back-off timer, initialisiert, der die Anzahl der abzuwartenden Zeitschlitze vorgibt. Solange das Medium frei ist, wird der Zähler dekrementiert. Wenn der Zähler den Wert 0 erreicht, darf die Station senden. Sollte die Station beim Herabzählen des Back-off timers unterbrochen werden, dann wählt sie keine neue Zufallszahl, sondern setzt das Herabzählen des Timers nach einem weiteren DIFS mit dem aktuellen Wert fort. Eine Station, die von einem Konkurrenten verdrängt wurde, behält so ihren Wert und hat damit eine höhere Wahrscheinlichkeit, als nächste Zugriff auf das Medium zu erlangen. Das Vorgehen ist exemplarisch in Abbildung 2.7 veranschaulicht: vier Stationen konkurrieren um das Medium, Station 1 sendet als erste, Station 2 als zweite und verdrängt dabei Station 3. Deren Back-off timer, der zum Zeitpunkt der Verdrängung auf 3 stand, wird „unterbrochen“, nach dem Abwarten eines weiteren DIFS, zählt sie einfach ab 2 weiter herab.

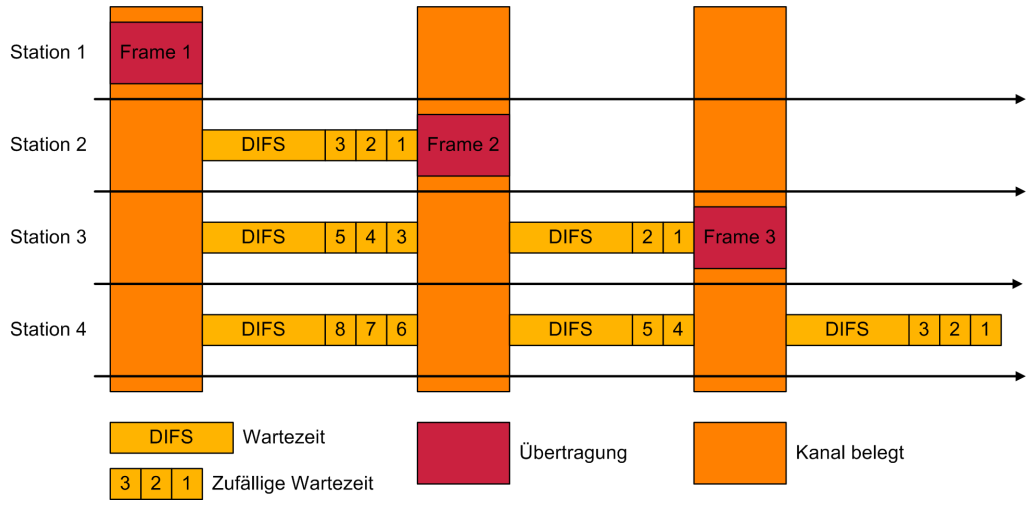


Abbildung 2.7: Konkurrenz beim Medienzugriff unter Verwendung von CSMA/CA gemäß IEEE 802.11

Mit dem Verfahren ist es nach wie vor möglich, das zwei sendewillige Stationen dieselben zufälligen Back-off timer wählen. Sie würden nach Ablauf des Zählers gleichzeitig anfangen zu senden, und es würde zu einer Kollision kommen. Daher verhindert das Verfahren Kollisionen nicht, sondern vermeidet sie nur. Es heißt deshalb Carrier Sense Multiple Access/Collision Avoidance (CSMA/CA).
Offensichtlich ist mit Hilfe des CW eine Veränderung der Zugriffswahrscheinlichkeit auf das Medium und damit eine Priorisierung von Nachrichten möglich: ein kleinerer Wert für CW führt dazu, dass statistisch gesehen die Wartezeit sinkt. Aus diesem Grund wird CW anfangs mit einem möglichst kleinen Wert CW_{min} initialisiert. Sollte das zu Kollisionen führen, so wird bei jedem Eintritt einer Kollision CW um einen festen Wert, den persistency factor PF, erhöht, und zwar bis ein vorher definierter maximaler Wert CW_{max} erreicht wird. Sobald eine Übertragung erfolgreich war, wird CW wieder auf den minimalen Anfangswert CW_{min} zurückgesetzt. Durch dieses Verfahren wird CW ständig an die aktuelle Netzlast angepasst. CW kann somit auch zur Abschätzung der Last dienen. Die Funktion zur verteilten Koordinierung von Netzwerkstationen beim Zugriff auf das Medium heißt Distributed Coordination Function (DCF).
Tabelle 2.1 listet die wichtigsten Parameter und Unterschiede von IEEE 802.11a und IEEE 802.11p auf.

	802.11p	**802.11a**
Kanalbandbreite	10 MHz	20 MHz
Datenraten	3 bis 27 Mbps	6 bis 54 Mbps
Länge eines Zeitschlitzes	16 µs	9 µs
Dauer eines SIFS	32 µs	16 µs
CW_{min}	15	15
CW_{max}	1023	1023

Tabelle 2.1: Grundlegende Parameter von IEEE 802.11a und IEEE802.11p

IEEE 802.11e [IEE05a] führt zusätzlich noch eine Priorisierung von Nachrichten *innerhalb* eines Netzwerkknotens ein, die sogenannte Enhanced Distributed Coordination Function (EDCF): während bei IEEE 802.11a Nachrichten von einer Station einfach in der Reihenfolge des Eintreffens in ihrem Sendepuffer abgearbeitet werden, bietet IEEE 802.11e zu diesem Zweck acht interne Warteschlangen an, jeweils mit unterschiedlicher Priorität. Die Auswahl der nächsten zu bedienenden Warteschlange geschieht dabei nicht statisch, sondern statistisch: je höher die Priorität einer Warteschlange ist, desto höher ist die Wahrscheinlichkeit, beim nächsten Versenden von Daten berücksichtigt zu werden. Das verhindert ein Verhungern einzelner Nachrichten in Warteschlangen mit niedriger Priorität. Das Vorgehen ist analog zum Verfahren beim Medienzugriff: beim internen Wettbewerb um den Zugriff auf den Funkkanal müssen die Sendepuffer eine bestimmte Zeit lang warten,

falls mehrere Puffer gleichzeitig senden möchten. Die zu wartende Zeitspanne heißt Arbitration Interframe Space (AIFS). Warteschlangen mit höherer Priorität werden kleinere AIFSs zugeteilt. IEEE 802.11p sieht eine Priorisierung mit EDCF vor, jeweils getrennt für Dienst- und Kontrollkanäle.

[SAE06, AESS06, ESKS06, SSEE06, KAE+06, Str07] schlagen ein Verfahren vor, das den Medienzugriff dahingehend verändert, so dass Nachrichten mit einer höheren Priorität ähnlich wie IEEE 802.11e mit kleineren Contention Windows ausgestattet werden und somit eine höhere Wahrscheinlichkeit haben, den Kanal früher als andere Knoten mit weniger relevanten Informationen zu belegen. Nachrichten können dabei nach mehreren Kriterien differenziert werden:

- Redundanz: haben potentielle Empfänger die Nachricht schon erhalten?
- Anwendungsdurchdringung: wie hoch ist die Wahrscheinlichkeit, dass eine empfangendes Fahrzeug eine Anwendung installiert hat, die die empfangene Nachricht überhaupt interpretieren kann?
- Verbindungscharakteristik: wie groß ist die Wahrscheinlichkeit, dass eine übertragene Nachricht von potentiellen Empfängern korrekt empfangen werden kann?
- Verbreitungscharakteristik: wie groß ist der potentielle Nutzen der Nachricht im Hinblick auf eine weitere Verbreitung?
- potentieller Nutzen: der Nutzen einer Information ist sehr gering oder Null, wenn z. B. die übertragene Information sich auf eine anderes Zielgebiet bezieht als das, in dem sich der Empfänger befindet. Gleiches gilt, falls die Information veraltet ist oder dem Empfänger schon bekannt ist.

Die fünf Kriterien sind lediglich heuristische Abschätzungen, die unter Annahme zahlreicher Parameter getroffen werden. Um eine Abschätzung möglichst genau vornehmen zu können, muss der Kontext des jeweiligen Fahrzeugs betrachtet werden. Zudem ist eine solche Abschätzung oft nur mit zusätzlichem Anwendungswissen zu treffen, das auf ISO/OSI-Schicht 2 eigentlich gar nicht vorzufinden ist und daher modelliert werden müsste. Zudem müssen Fahrzeuge über den Kontext eines anderen im Bilde sein, wenn sie den potentiellen Nutzern einer Nachricht für den anderen abschätzen möchten. [Str07] berechnet zu diesem Zweck eine heuristische Relevanzabschätzung, die heuristische Abschätzungen über verschiedene Kontextaspekte gewichtet und mittelt. Um das Anwendungswissen den unteren Netzwerkschichten verfügbar zu machen, schlägt [Str07] eine Cross-Layer-Architektur nach [FTMT+05] vor, die in Abbildung 2.8 gezeigt wird.

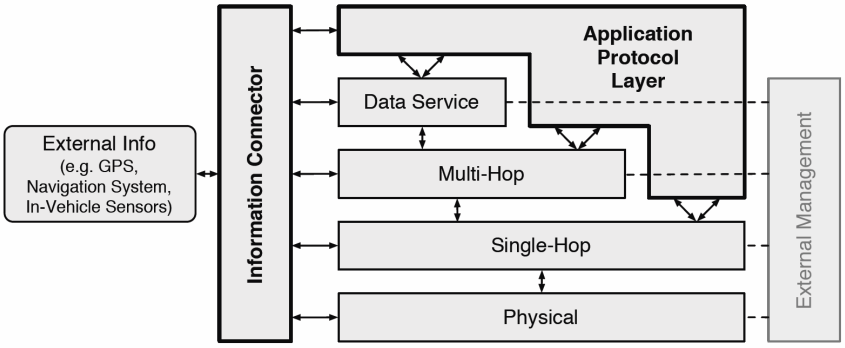


Abbildung 2.8: Eine Protokollarchitektur für VANETs, Quelle: [FTMT+05]

Der *Information Connector* ist eine gemeinsame Schnittstelle zum Austausch von Sensordaten (z. B. GPS-Informationen zur Positionsbestimmung, die in mehreren Protokollebenen von Interesse sein können), sowie Verwaltungs- und Zustandsdaten des Netzwerks, die aus Paketen extrahiert werden. Alle Protokollebenen können sich beim Information Connector registrieren und erhalten dann auf der Basis eines Publish/Subscribe-Mechanismus die gewünschten Daten. Somit stellt er den einzelnen Protokollen Kontextdaten zur Verfügung. Die *Single-Hop*-Ebene umfasst alle Funktionen zur Kommunikation mit direkten Funknachbarn, die *Multi-Hop*-Ebene die Kommunikation mit allen Knoten, die nicht in der unmittelbaren Funkreichweite eines Netzwerkteilnehmers liegen. Der *Data Service* enthält weitere Transportfunktionen wie z. B. einen zuverlässigen Bytestrom-Dienst oder Datagramm-Schnittstellen.

2.3.3.3 Routing

Die Aufgabe des Routing ist es, Netzwerkpakete von einem Startknoten zu einem Zielknoten zu vermitteln, auch wenn diese Knoten nicht in unmittelbarer Nachbarschaft stehen, so dass keine direkte Kommunikation untereinander möglich ist. Das bedeutet, dass ein Routing-Algorithmus geeignete Zwischenstation finden muss, die ein Netzwerkpaket dem Ziel schrittweise näher bringen. In VANETs wird das Problem durch einige Umstände komplexer:

- Die ständige Mobilität der Netzwerkknoten macht es nahezu unmöglich, die Topologie des Netzes abzufragen und darauf Routen aufzubauen. Das Ziel, eine stabile Unicast-Route zwischen zwei Netzwerkknoten aufrecht zu erhalten, wird dadurch ungleich schwieriger. Die Schwierigkeiten nehmen weiter zu, je größer die Distanz zwischen den beiden Knoten ist, da somit auch die Anzahl der zu wählenden Zwischenknoten größer wird.

- Ein weiteres Problem beim Aufbau von Unicast-Routen ist, das i. A. die Aufenthaltsorte der Zielknoten nicht bekannt sind und daher erst durch einen geeigneten Algorithmus ermittelt werden müssen.
- Durch den Zugriff auf ein gemeinsames Medium, das aber immer nur ein Knoten in einer Funknachbarschaft exklusiv belegen darf, muss die Anzahl der Zugriffe reduziert werden, so dass Wartezeiten beim Medienzugriff gering gehalten werden können.
- Aus dem gleichen Grund sollte die Anzahl der Weiterleitungen überhaupt minimiert werden, um redundante Übertragungen zu vermeiden. Das explosionsartige unkontrollierte Verbreiten von Nachrichten in VANETs durch Broadcast-Methoden wird auch Broadcast Storm genannt (vgl. [NTCS99]).

Routing-Protokolle für mobile Netze wurden schon vor dem Aufkommen von VANETs erforscht. Hierbei handelte es sich um Verfahren zum Aufbau von Routen zwischen zwei Knoten in dynamischen Netzen mit geringerer Mobilität. Der wesentliche Unterschied zu den Anforderungen in VANETs besteht darin, dass (a) die Mobilität höher ist und dass VANET-Anwendungen es häufig erfordern, eine Nachricht nicht nur zwischen zwei dezidierten Knoten zu übermitteln, sondern Nachrichten (z. B. Warnungen) erst in ein bestimmtes Zielgebiet zu bringen und dort zu verbreiten. Andererseits lässt sich die Netzwerktopologie aufgrund der Tatsache, dass Fahrzeuge an Straßen gebunden sind, zumindest für einen kurzen Zeitraum vorhersagen. Ein Umstand, den sich einige Unicast-Routing-Protokolle wie das in [LMFH05] beschriebene zunutze machen. Generell lassen sich Routing-Protokolle in die folgenden Kategorien unterteilen, die im Anschluss mit jeweils wichtigen Vertretern vorgestellt werden:

- Proaktive Verfahren
- Reaktive Verfahren
- Hybride Verfahren
- Positionsbasierte Verfahren
- Implizites Routing durch intelligentes Broadcasting

Weitere Unterteilungen von Routing-Protokollen, die eine Einsortierung aufgrund von anderen Kriterien unternehmen, finden sich in [KR03, Tan02].

Proaktives Routing (auch als global routing, link state routing oder table-driven routing bezeichnet) berechnet den Pfad mit den geringsten Kosten zwischen zwei Netzwerkknoten im Voraus. Solche Algorithmen benötigen daher vor der Routenberechnung schon globales und vollständiges Wissen über die Verbindungswege im Netz. Welcher Weg dann bei

Erreichen eines bestimmten Knotens gewählt werden muss, ist in einer Tabelle im Knoten selbst hinterlegt. Im Hinblick auf VANETs haben diese Verfahren zwei entscheidende Nachteile:

1. Es ist nicht klar, wie man bei derart mobilen Netzen an die vorab benötigte Information über die Verbindungskosten zwischen den Netzwerkknoten kommt.

2. Die Verbindungskosten und die Verbindungen an sich sind in VANETs äußerst volatil, so dass Verbindungsinformationen – so sie denn überhaupt vorliegen – sehr schnell wieder veraltet sind.

Aufgrund der genannten Unzulänglichkeiten können proaktive Protokolle in VANETs eigentlich nicht eingesetzt werden. Bekannte Vertreter dieser Klasse sind Open Shortest Path First (OSPF) [Moy98] und Optimized Link State Routing (OLSR) [CJ03].

Reaktives Routing (auch dynamisches oder on-demand routing genannt) hat kein globales Wissen über den Netzzustand. Stattdessen werden die Verbindungskosten in einem iterativen verteilten Prozess berechnet, indem jeder Knoten zuerst nur die Kosten der Verbindung zu seinen direkten Nachbarn abfragt. Durch weiteren Informationsaustausch berechnet ein Netzwerkknoten dann die Kosten einer Übertragung zu einer Menge von Zielknoten. Er hält dabei nicht alle Routen in seinem Speicher vor, sondern nur diejenigen, die tatsächlich gebraucht werden. Das bedeutet aber auch, dass Routen zu einem neuen bisher noch nicht verwendeten Ziel erst aufgebaut werden müssen, was einen zeitlichen Overhead beim Aufbau neuer Ende-zu-Ende-Verbindungen bedingt. Durch den reduzierten Datenverkehr eignen sich solche Protokolle besser für den Datenaustausch in VANETs. Ein prominenter Vertreter ist Dynamic Source Routing (DSR) [JHM07]. Bei einem neuen Verbindungswunsch zu einem Ziel mit (noch) unbekannter Route erzeugt ein Netzwerkknoten ein sogenanntes Route Request (`RREQ`) Paket und flutet das Netz damit. Jedes `RREQ`-Paket enthält eine Sequenznummer und Informationen über die bisher besuchten Knoten und jeder empfangende Knoten sendet das Paket erneut aus, es sei denn es hat das schon getan (was anhand der Sequenznummer erkennbar ist) oder er ist der Zielknoten. Um eine endlose Flutung des Netzes zu verhindern, sind `RREQ` mit einem Time-to-Live-(TTL)-Zähler ausgestattet, der bei jeder empfangenden Zwischenstation dekrementiert wird. Sollte der Zähler den Wert 0 erreichen, so wird das Paket vernichtet und nicht erneut ausgesendet. Falls das Paket dann beim Zielknoten ankommt, kann dieser anhand der gespeicherten Pfadinformationen dem Sender des Originalen `RREQ`-Pakets antworten, das Antwortpaket nimmt dabei genau den umgekehrten Weg. Die so gefundenen Routen werden zwischengespeichert und können z. T. auch beim Aufbau von neuen Routen verwendet werden. Nachteile bei diesem Protokoll bestehen darin, dass unterbrochene Verbindungen lokal nicht repariert werden und veraltete zwischengespeicherte Informationen zu Problemen bei erneuten Verwendungen von Routen führen können. Außerdem ist der Aufwand

zum Aufbau von Routen recht hoch, was das Verfahren für die Anwendung in VANETs letztlich disqualifiziert. Ad hoc On-Demand Distance Vector (AODV) Routing [PBRD03] verfährt insofern analog zu DSR, dass Routen erst dann aufgebaut werden, wenn sie benötigt werden. Der Hauptunterschied zu DSR besteht lediglich darin, dass DSR „source routing“ betreibt, d. h. einem Datenpaket schon beim Aussenden durch den Ursprungsknoten der komplette vorher ermittelte Pfad zum Ziel mitgegeben wird. Bei AODV speichern die Knoten nur die Verbindungsinformationen zum nächsten möglichen Knoten, die dafür ständig überwacht werden müssen. Zusätzlich kann bei AODV eine Aussage über das Alter einer Route gemacht werden. Einer der Nachteile (neben den wie bei DSR zu erwartenden Inkonsistenzen) ist die konstante Netzlast durch das sogenannte Beaconing, das zur Überprüfung des Verbindungsstatus zu den direkten Nachbarn benötigt wird. Zu AODV gibt es spezielle Erweiterungen für VANETs (vgl. hierzu [KSA02, SK02]).

Hybride Verfahren versuchen sowohl proaktive als auch reaktive Komponenten miteinander zu vereinen. Das Zone Routing Protocol (ZRP) [Haa97] versucht, den Verwaltungsaufwand proaktiver Routing Protokolle zu vermindern, ebenso wie die bei reaktiven Protokollen entstehenden Latenzzeiten beim Verbindungsaufbau. Dazu definiert ZRP eine k-Nachbarschaftszone, bestehend aus allen Knoten, die über maximal k Zwischenknoten erreichbar sind. Dabei beziehen sich die Zonen jeweils auf die Sicht der einzelnen Knoten. Innerhalb dieser Zone wird proaktiv mit Hilfe des Intra-zone Routing Protocol (IARP) geroutet, außerhalb reaktiv unter Verwendung des Inter-zone Routing Protocol (IERP). Das reaktive Routing verwendet dabei nur die Knoten im Grenzbereich, d. h. alle Knoten die genau k Hops vom Ursprungsknoten entfernt sind. Sobald ein `RREQ`-Paket einen Grenzknoten erreicht, fragt er erst in seiner eigenen Nachbarschaftszone nach, ob sich der gewünschte Zielknoten darin befindet. Falls nicht, leitet er das `RREQ`-Paket an seine eigenen Grenzknoten weiter, nachdem er seine eigene Adresse dem Paket hinzugefügt hat, so dass letztendlich der Pfad durch das Netz wie bei (DSR) nachverfolgt werden kann. In eine ähnliche Richtung geht Cluster-based location Routing (CBLR) [SES02], das auf VANETs zugeschnitten wurde: dieses Protokoll benennt dezidierte Cluster mit einem Verantwortlichen, der den Datenverkehr zwischen den Clustern regelt (den sogenannten Cluster-Head). Er verwaltet auch Adressen und Positionen aller an seinem Cluster beteiligten Knoten. Kommunikation läuft immer über den Cluster-Head, was ihn zu einem Schwachpunkt in diesem Protokoll macht, zusätzlich zum Kommunikationsaufwand, der durch die Verwaltung der Cluster entsteht.

Positionsbasierte Verfahren nutzen Kontextinformationen, nämlich Positionsdaten, für die Zustellung von Nachrichten in einem Netzwerk aus. Diese Daten müssen mittels GPS oder ein anderen Technik ermittelt werden können, d. h. ein Netzwerkknoten muss sich mindestens seiner geographischen Position bewusst sein, kann aber u. U. auch die ande-

rer Netzteilnehmer kennen. Sollte es möglich sein, Knoten auch anhand ihrer Position zu adressieren, so spricht man von *Geocasting* [MWH01]. Dadurch, dass diese Verfahren Kontextdaten (Position) verwenden, sind sie für diese Arbeit besonders interessant. Das Verfahren Location-aided Routing (LAR) [KV00] verwendet solche Information derart, dass es zwar ähnlich wie AODV oder DSR mittels Flooding von RouteRequests versucht, eine Route aufzubauen, die `RREQ` aber geographisch limitiert, so dass nur bestimmte Regionen geflutet werden. Netzwerkknoten leiten ein `RREQ`-Paket nur dann weiter, wenn sie sich innerhalb einer bestimmten request zone befinden. Die request zone selbst ist ein Teil einer größeren expected zone, in der ein Sender seinen Zielknoten vermutet. Die Vermutung wird gestützt durch Wissen über frühere Aufenthaltsorte eines Knoten, seine Bewegungsrichtung und -geschwindigkeit (weitere Kontextparameter). Die Zonen werden vom Sender festgelegt.
Greedy Perimeter Stateless Routing (GPSR) [KK00] verwendet dabei ein hybrides Verfahren. In einer ersten Phase wird ein Datenpaket greedy weitergeleitet, d. h. ein Paket wird explizit an denjenigen Netzwerkknoten weitergeleitet, der dem Ziel (der nicht unbedingt ein Knoten sein muss, sondern nur ein Punkt im Raum) hinsichtlich euklidischer Distanz am nächsten ist. Erst wenn ein Knoten selbst dem Zielpunkt am nächsten ist, wird der Knoten, der das Paket aktuell hält, als Ziel definiert. Danach wird das Paket an einen anderen Nachbarn weitergeleitet, und zwar in einer Art und Weise, dass das Paket um den Zielpunkt geroutet wird. Das stellt sicher, dass sich das Paket immer in der Nähe des Zielpunkts befindet. Hier kann der Eindruck entstehen, dass ein Paket immer um den Zielpunkt kreist, anstatt zum Zielpunkt selbst. Entscheidend ist aber die Tatsache, dass sich am geographischen Zielpunkt nicht unbedingt ein Knoten befinden muss: wichtig ist daher, dass das Paket immer in der Nähe des Ziels gehalten wird und der Knoten, der dem Ziel am nächsten ist, als Empfängerknoten definiert wird.
Positionsdaten und andere Kontextkriterien werden auch im in [LMFH05] vorgeschlagenen Verfahren verwendet: In Städten verfügen Fahrzeuge im Kreuzungsbereich über bessere Funkverbindungen, da ihre Signale nicht von Gebäudeabschattungen gestört werden. Daher werden sie als Netzwerkknoten für Weiterleitungen bevorzugt. Um dieses Kriterium nutzen zu können, muss jedoch Zugriff auf eine digitale Karte, die zudem noch über topografische Bebauungsdaten verfügt, vorhanden sein.

Die bisher vorgestellten Verfahren sind Routing-Verfahren im klassischen Sinne: sie bauen Routen auf, entweder proaktiv oder reaktiv und speichern diese für eine spätere Wiederverwendung. Die nun beschriebenen Verfahren bauen keine Routen mehr, sondern erledigen die Wegewahl implizit, d. h. sie wählen nur noch den nächsten Knoten geeignet aus und überlassen diesem dann die Wahl des übernächsten Knotens. Die so gewonnenen „Routen“ werden nicht mehr gespeichert, sondern jedes mal neu ermittelt. Man spricht in diesem Falle von „Forwarding“, das durch *intelligentes Broadcasting* umgesetzt wird.

Contention Based Forwarding (CBF) [Füß07] verwendet wie GPSR teilweise einen greedy Ansatz: Derjenige Netzwerkknoten wird zum Weiterleiter einer Nachricht, der die größte euklidische Distanz zwischen zwei Netzwerkknoten überbrückt. Von Vorteil ist hier, dass keinerlei Zusatzwissen über die geografischen Gegebenheiten oder über die Position anderer Netzwerkknoten vorhanden sein muss. Erreicht wird das dadurch, dass ein Netzwerkpaket immer die Position des letzten weiterleitenden Knotens und die Zielposition enthält. Mit Hilfe dieser Informationen berechnen empfangende Netzwerkknoten die Distanz die eine Weiterleitung des Paketes durch sie selbst überbrücken würde. Nun beginnt eine Wettbewerbsphase ähnlich zu der von CSMA/CA, nur mit dem Unterschied, dass derjenige Knoten das geringste Warteintervall zugeteilt bekommt, der die größte Distanz überbrückt:

Sei f die Position des letzten weiterleitenden Knotens,
z die Zielposition,
n die eigene Position,
r_{radio} die durchschnittliche Funkreichweite,
$dist$ eine Funktion, die euklidische Distanz zwischen zwei Knoten berechnet,
dann stellt

$$P(f,z,n) = \max\left\{0, \frac{dist(f,z) - dist(n,z)}{r_{radio}}\right\} \in [0,1]$$

ein Maß für das geographische Fortschreiten einer Nachricht dar. Die Wartezeit eine Knotens ist dann definiert als

$$t(P) = t_{max} * (1 - P)$$

in Abhängigkeit einer zu definierenden maximalen Wartezeit t_{max}. Nach Ablauf seiner Wartezeit sendet der erste Knoten das Paket. Alle anderen Knoten „hören“ die Übertragung (gemeinsames Medium) und unterdrücken daraufhin ihre eigenen Übertragungen, um dadurch Redundanz zu vermeiden.

Grundsätzlich besteht bei positionsbasiertem Routing und broadcastbasiertem Forwarding das Problem, dass Nachrichten in Sackgassen landen können. Das ist der Fall, wenn eine vermeintlich optimale Route eingeschlagen wurde, aber im weiteren Verlauf der Route kein Knoten mehr gefunden werden kann, der das Paket dem Ziel näher bringt. Wenn nun die Möglichkeit besteht, an einem früheren Punkt der Route eine Abzweigung zu nehmen, die auf einem lokal suboptimalen Weg zum Ziel führt, dann kann dieser Weg nicht mehr eingeschlagen werden, da eine Umkehr auf der Route eine Vergrößerung der Distanz zum Ziel bedeuten würde und daher nicht ausgewählt würde.

2.3.3.4 Informationsdissemination

Da in VANETs sehr oft gar nicht der Bedarf besteht, Unicast-Kommunikation zwischen zwei dezidierten Knoten zu betreiben, sondern Informationen einer ganzen Reihe von Fahrzeugen oder innerhalb eines Gebietes bereitzustellen, seien im Folgenden noch Verfahren zur Informationsverbreitung betrachtet, die ohne expliziten Aufbau einer Route auskommen und Informationen in 1-zu-n Beziehung zur Verfügung stellen.

Autonomous Gossiping [DQA04] ist ein Ansatz, bei dem die einzelnen Knoten Profile verwalten, in denen gespeichert wird, an welcher Art Information sie interessiert sind. Die dazu notwendigen Profile werden verteilt gespeichert, es ist also keinerlei Infrastruktur notwendig. Die Metrik für die Weiterleitung einer Information ist bei diesem Ansatz nicht mehr ein Maß, das unabhängig von der zu transportierenden Nachricht berechnet werden kann, sondern orientiert sich am Inhalt der Nachricht, was ein wesentlicher Unterschied zu den vorher vorgestellten Techniken ist. Zur Dissemination einer Nachricht verteilen die einzelnen Netzwerkknoten ihre Interessensprofile in der Nachbarschaft, so dass die umliegenden Knoten informiert sind, welche Informationstypen tatsächlich weitergeleitet werden müssen. Allerdings ist der durch die Verteilung der Profile induzierte Verwaltungs- und Nachrichtenaufwand nicht unerheblich. Das Verfahren kann außerdem keine Zustellung an alle interessierten Knoten garantieren.

Scott und Yasinac [SY04] verwenden grundsätzlich einfach Broadcast-basierte Kommunikation, passen die Kommunikation aber an ein weiteres Kontextkriterium an: falls das Netzwerk weniger dicht ist, wird die Wahrscheinlichkeit für das erneute Aussenden eines Pakets erhöht. Durch diese Anpassung an die lokale Netzwerktopologie wird auch eine implizite Kontrolle der Netzlast erreicht: da die Netzlast in dichten Netzen höher ist, wird durch dieses Verfahren eine automatische Verringerung der Anzahl der Retransmissionen erzielt.

Role-based multicasting [BH00] macht sich Kontextinformationen nutzbar, in dem es sowohl Nachbarschaftstabellen verwaltet als auch Informationen über alle Nachrichten speichert, die ein Knoten bereits an einen anderen zugestellt hatte. Jeder Knoten verwaltet eine Tabelle N, in der alle Nachbarn enthalten sind und pro Nachricht eine weitere Tabelle S, die speichert, von welchem Knoten die mit dieser Tabelle assoziierte Nachricht empfangen wurde. Durch einen Vergleich von N und S kann ein Knoten feststellen, ob er andere Knoten kennt, die die mit S assoziierte Nachricht noch nicht empfangen haben. Falls dies der Fall ist, wird der Knoten die Nachricht erneut aussenden. Der Retransmissionsmechanismus selbst basiert auf einer Mischung aus CBF [Füß07] und [KV98]. Falls N und S identisch sein sollten, dann kennen alle Knoten in der Nachbarschaft die Informa-

tion schon. Der Knoten wartet dann auf eine Veränderung von N, d. h. auf die Ankunft eines neuen Knotens in der Nachbarschaft. Bei diesem Verfahren ist es also wichtig, den zeitlichen Verlauf von Kontextinformationen zu speichern: Ein Knoten sendet eine Nachricht nur dann, wenn er genau weiß, dass es eine Nachricht noch nicht an einen Knoten zugestellt hatte.

Wu et al. stellen mit MDDV [WFGH04] einen anderen Ansatz vor, der zusätzlich zur Netzwerktopologie die Trajektorie eines Straßenverlaufs als Kontextkriterium verwendet. Wichtig hierbei sind die Länge von Straßenabschnitten, die Anzahl von Spuren und die Flussrichtung des Verkehrs auf diesen Abschnitten. Grundlegend ist dabei die Annahme, dass mehr Fahrzeugspuren zu höheren Fahrzeugdichten führen und somit zu einer besseren Netzwerkkonnektivität. Gleichzeitig ist eine Verkehrsfluss entgegengesetzt zur gewünschten Ausbreitungsrichtung einer Nachricht als nachteilig zu betrachten. Nachrichten sollen dann entlang von Trajektorien mit höheren Dichten und der der „richtigen“ Verkehrsflussrichtung geleitet werden.

Stored Geocast [MFE03] erweitert die in [KV98, KV99] vorgestellten Verfahren, um eine Nachricht an alle in einem spezifizierten geographischen Gebiet befindlichen Knoten zuzustellen. Um die Zustellung auch in dünn besiedelten Netzen sicherzustellen, wird die Möglichkeit eines physikalischen Nachrichtentransports in Betracht gezogen. Wenn sich im Zustellgebiet nur wenige Knoten befinden oder das einzige Fahrzeug, das eine bestimmte Nachricht in seinem Speicher hat, die Zielregion verlässt, so wird die Nachricht zwischengespeichert, um später gegebenenfalls wieder auszusenden. Der Empfänger soll dann die Nachricht physikalisch in ein Zielgebiet zurück transportieren. Um dieses Ziel zu erreichen, gibt es lokale Abstimmungsverfahren, die ein Fahrzeug zum Container für eine Nachricht machen. Zusätzlich wird Wissen über die Nachbarschaft dazu verwendet, die Nachricht an diejenigen Fahrzeuge zu senden, die die Zielregion wahrscheinlich wieder betreten werden. Zur Verbreitung in der Zielregion selbst wird periodisches Broadcasting in statischen Intervallen verwendet. Wegen der daraus resultierenden Netzbelastung zählt dieses Verfahren zu den Nachteilen des Ansatzes.
Verfeinerungen des Stored Geocasting wurden mit Hilfe von verschiedenen Techniken durchgeführt:

- *Tokens* werden in [SGM04] verwendet, um das Fahrzeug zu kennzeichnen, das für das Speichern und Wiederaussenden einer Nachricht zuständig ist. Das Token wird an die Fahrzeuge vergeben, die innerhalb eines bestimmten Radius (safety radius) die größte Entfernung zu einer Gefahrenstelle einnehmen. Das Token wird weitergegeben an diejenigen Fahrzeuge, die sich von der Gefahrenstelle entfernen (und zwar in beide Fahrtrichtungen), aber noch innerhalb des Radius befinden. Dadurch bewegen sich die Token vom Ursprung weg, gleichzeitig wird sichergestellt, dass alle Fahrzeu-

ge in unmittelbarer Nachbarschaft des Tokens die Nachricht erhalten haben, da ja die Fahrzeuge mit dem Token für das periodische Wiederaussenden einer Nachricht zuständig sind. Dadurch vergrößert sich die Verbreitungsregion einer Nachricht. Für dieses Verfahren ist Wissen über den Straßenverlauf und Fahrtrichtung nötig.

- Ein weiteres Verfahren verwendet eine *relevance zone* [BSH00] und basiert auch auf [KV98]. Der Ansatz legt eine Zone fest, in der eine Nachricht von potentiellem Interesse sein könnte und passt den Medienzugriff ähnlich wie CBF an.

- *Risk zones* werden in [Ben04] definiert. Nur innerhalb eines solchen Gebiets werden Nachrichten über ein bestimmtes Ereignis überhaupt übertragen, wobei die Übertragung selbst nicht von allen Fahrzeugen vorgenommen wird, sondern von einer speziell definierten Menge. Das Forwarding selbst basiert auf CBF, eine Klassifizierung der Nachrichten findet auf der Basis der Dringlichkeit ihres Inhalts statt.

Ein *nutzenbasiertes Verfahren* wird z. B. in [WR05a, WR05b] vorgestellt. Mit Hilfe einer Nutzenfunktion wird der zu erwartende Nutzen einer Information ermittelt. In Abhängigkeit des Nutzens wird den Netzwerkknoten dann der Kanal zugeteilt. Erst wenn die Kapazität des Kanals nicht ausreicht, um alle Pakete zu versenden, wird die Nutzenfunktion auch verwendet, um zu entscheiden, welche Nachricht verworfen wird. Nachrichten mit geringem Nutzen werden gelöscht. Interessant ist der Ansatz deshalb, weil er Nachrichten nicht isoliert betrachtet, sondern versucht, eine Vielzahl von Nachrichten innerhalb eines Knotens anhand des Nutzens und Kontexts zu bewerten und entsprechend zu koordinieren. Der Ansatz verwendet aber nur eine statische Nutzenfunktion und nicht auf verschiedene Anwendungstypen hin optimierte verschiedene Funktionen.

Ein weiteres nutzenbasiertes Verfahren wird in [Kos05] definiert, allerdings wird der Begriff des Nutzens hier etwas differenzierter gesehen: von Nutzen kann eine Information sein, wenn sie

1. in der Situation des Fahrers von Interesse sein könnte

2. wenn der Knoten aufgrund seiner eigenen Interessenslage explizit die Nützlichkeit einer Information spezifiziert hat.

Letzterer Fall führt dazu, dass Information vom Netzwerkknoten dediziert angefordert werden muss, weil bei stark divergierenden Interessenlagen verschiedener Knoten in einem Netzwerk nicht davon ausgegangen werden kann, dass solche Informationen für alle Knoten von Interesse sind. Daher ist in diesem Falle keine proaktive Verbreitung der Information vorzunehmen. Kontextinformationen sind jedoch von allgemeinem Interesse, so dass sie allgemein verbreitet werden sollen: hier wird der Nutzen einer Information durch den Grad der „Neuheit“ bewertet: wenn eine Information zu einer radikalen Neubewertung der

Umwelt führt, so wird ihr Nutzen als hoch eingestuft. Zusätzlich wird davon ausgegangen, dass alle Knoten in der Nähe sich in einem ähnlichen Kontext befinden. Eine Übermittlung derselben Information würde also zu einer ähnlich radikalen Neubewertung der Umwelt durch die benachbarten Knoten führen, womit die Information also auch für diese interessant wäre. Eine Nachricht wird bei diesem Verfahren also solange weitergeleitet, bis Ihr Nutzen und damit das Interesse der Knoten unter eine bestimmte Schwelle gesunken ist. Es gibt also kein beim Versenden einer Information explizit anzugebendes Verbreitungsgebiet mehr.

Die in [SAE06, AESS06, ESKS06, SSEE06, KAE+06, Str07] beschriebenen Verfahren beinhalten nicht nur eine Anpassung des Medienzugriffsverfahrens (siehe Kapitel 2.3.3.2), sondern auch ein Verfahren zur Informationsausbreitung: Information wird auch hier anhand seines Nutzens bewertet. Der Nutzen ist dabei abhängig von der Anwendung – jede Anwendung definiert seine eigene Heuristik zur Ermittlung des Nutzens einer Information. Dieser Wert wird dann normiert, um zu einer fairen Bewertung zu kommen. Die Verfahren sind darauf hin optimiert, einem bestimmten Netzziel zu dienen: die Unterstützung eines Fahrers durch kooperative Fahrerassistenzanwendungen, d. h. Netzwerkpakete, die fahrsicherheitsrelevante Informationen enthalten werden priorisiert. Dabei wird nicht der eigene Nutzen abgeschätzt, sondern der Nutzen für andere Fahrzeuge in der Nachbarschaft, so dass sich insgesamt ein altruistisches Netzwerkverhalten der einzelnen Knoten ergibt. Die Nutzenbewertung selbst passt sich dabei laufend an den neuen Kontext an, um der Dynamik in VANETs Rechnung zu tragen und sieht eine laufende Reorganisation der auszusendenden Informationen vor. Auch findet eine Anpassung an die aktuelle Last des Netzwerks statt: bei geringer Auslastung werden Informationen zwischengespeichert, um sie zu einem späteren Zeitpunkt doch noch an neue Fahrzeuge zustellen zu können. Genauso wird bei hoher Netzlast das Aussenden von Nachrichten mit wahrscheinlich geringem Nutzen unterdrückt, sie werden aber nicht gelöscht. Durch die Offenheit der Nutzenabschätzung ist es möglich, weitere Ansätze (z. B. die erhöhte Konnektivität in Kreuzungsbereichen) zur Abschätzung in die Nutzenfunktion mit einzubeziehen. Eine Integration anderer Netzwerkprotokolle (TCP/IP) ist auch möglich: diese werden dann als Anwendungen mit einer eigenen Nutzenfunktion betrachtet. Gleichzeitig ist diese Offenheit aber auch eine Herausforderung an die Modellierung des Kontextes, so dass er in Nutzenberechnung einbezogen werden kann: hier ist nicht klar, wie die Nutzenheuristiken aufeinander abgestimmt werden können, so dass es keiner Anwendung möglich ist, für sich einen unfair großen Nutzen zu definieren und alle anderen Anwendungen benachteiligt werden. Weiterhin ist nicht klar, wie das zur Bewertung nötige Anwendungswissen überhaupt unteren Netzwerkschichten zur Verfügung gestellt werden soll. Die Integration neuer Anwendungen mit eigenen Nutzenbewertungsfunktionen zur Laufzeit des Systems (also wenn keine erneute zentrale Abstimmung aller Nutzenfunktionen mehr möglich ist),

ist nicht klar.

2.4 Semantic Web

Um überprüfen zu können, ob die Techniken des Semantic Web geeignet sind, um Kontext in VANETs modellieren zu können, werden diese im Folgenden vorgestellt.

2.4.1 Semantik

Der Begriff „Semantik“ leitet sich vom Griechischen σημαίνειν ab, was soviel wie „anzeigen“ oder „bezeichnen“ bedeutet. Der Begriff wird je nach Forschungsgebiet unterschiedlich verwendet, zielt aber immer darauf ab, die Bedeutung oder den Sinn eines Wortes zu erfassen, bzw. bezeichnet dieser Begriff die Wissenschaft, die sich der Erforschung von Wortbedeutungen beschäftigt. Ein von Gottlob Frege verwendetes Beispiel [Fre02] veranschaulicht den Unterschied: die Begriffe „Morgenstern“ und „Abendstern“ beziehen sich (engl. „reference“) auf denselben Planeten, die Venus, haben aber unterschiedliche Bedeutungen (engl. „meaning“). Während dieser sprachwissenschaftliche Unterschied in der Informatik, in der die Begriffe Sinn und Bedeutung nahezu synonym verwendet werden, von geringem Belang ist, ist die semiotische Einordnung der Semantik von größerer Wichtigkeit: das semiotische Dreieck (Abbildung 2.9) stellt eine Beziehung her zwischen Syntax, Semantik und Pragmatik eines Begriffs. Worte (Symbole) werden benutzt, um bestimmte Dinge (Frege: „Bedeutung“) zu bezeichnen. Welches Konzept oder Begriff (Frege: „Sinn“) einem Wort aber tatsächlich durch denjenigen, der es aufnimmt, zugeordnet wird, ist für denjenigen, der ausspricht, nicht vollständig gesichert.

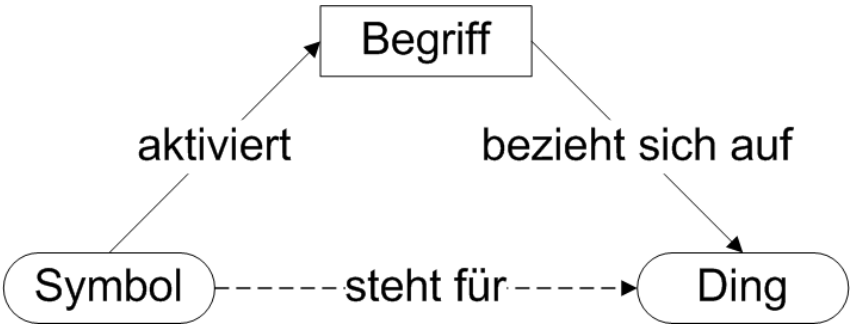


Abbildung 2.9: Das Semiotische Dreieck [ES06]

In der Informatik im speziellen ist das Gebiet der Semantik reduziert auf das der *formalen Semantik*, die versucht, Computerprogramme, Spezifikationen und andere maschinenlesbare Codes zu formalisieren und somit die Unsicherheit in der Zuordnung von Begriffen zu Dingen (um mit dem semiotischen Dreieck zu sprechen) möglichst weitgehend zu eliminie-

ren und ein Vokabular (Syntax) für die Bezeichnung von Konzepten festzulegen. Vokabular und Konzepte sollen dadurch maschinenverwertbar werden.

Allerdings wird in der Informatik Wissen auf vielerlei Art und Weise formalisiert und repräsentiert: UML, Entity-Relationship-Modelle, Datenbanken und dazugehörige Schemata, Dateistrukturen, Prozessbeschreibungssprachen – um nur einige Beispiele zu nennen – werden verwendet, um Vokabulare festzulegen, mit denen dann Konzepte beschrieben werden können. Allein die Vielzahl der Möglichkeiten zeigt, dass es hier ein großes Maß an Heterogenität gibt. Man versucht, diese Heterogenität dadurch zu überwinden, dass man neue Beschreibungssprachen entwickelt, die mit einem hohen Maß an semantischer Ausdrucksfähigkeit ausgestattet sind, in der Hoffnung, die Ausdrucksmöglichkeiten bisheriger Sprachen vereinigen zu können.

Eine jüngste Entwicklung ist dabei das Aufkommen des Semantic Web als eine weitere Stufe in der Entwicklung des Internet. In der von Tim Berners-Lee et al. entwickelten Vision des Semantic Web [BLHL01] wird die im Internet „gespeicherte“ Information mit einer wohldefinierten Bedeutung angereichert, so dass das Semantic Web kein Parallel-Netz, sondern eine Erweiterung des bestehenden Netzes darstellt.

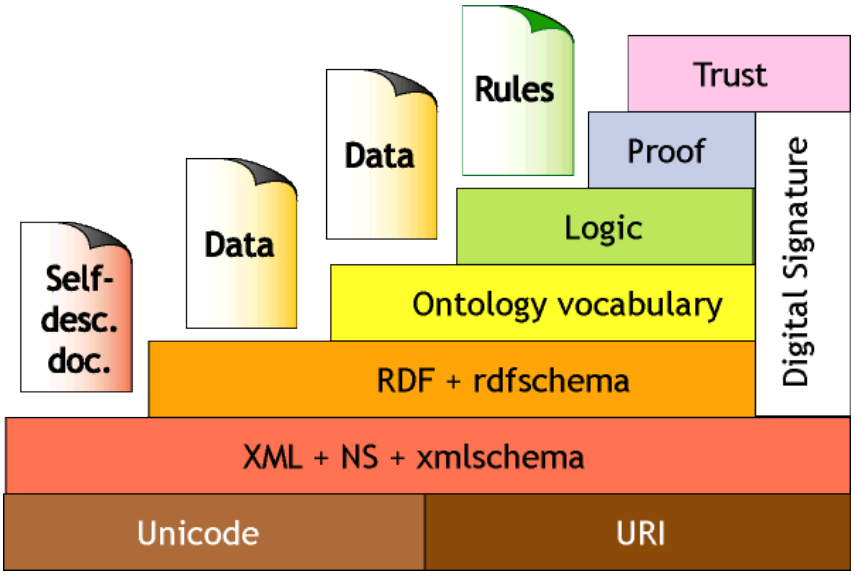


Abbildung 2.10: Schichtenmodell des Semantic Web [BL00]

Abbildung 2.10 zeigt, wie neue semantisch angereicherte Schichten in die Schichtenstruktur des bestehenden Internet integriert würden. Das Semantic Web zielt also darauf ab, ein Rahmenwerk zu erstellen, um Konzepte und Ihre Beziehungen in einer semantisch reichen

Art und Weise zu erfassen. Die im Semantic Web entstehenden Modelle werden *Ontologien* genannt und sind verwendbar über Anwendungs- und Unternehmensgrenzen hinweg, sowie in verschiedenen Nutzergruppen.

2.4.2 Ontologien

Der Begriff der Ontologie entstammt der theoretischen Philosophie und beschäftigt sich mit Fragestellungen um den Begriff des „Seins“ bzw. der Existenz. In klassischen Gliederungen der Philosophie umfasst die Ontologie eine „allgemeine Metaphysik“, wenn sie sich mit grundlegenden Entitäten und deren strukturellen Beziehungen beschäftigt, und eine „spezielle“ Metaphysik, wenn es um konkrete Gegenstandsbereiche geht. Erste philosophische Ansätze finden sich schon beim Vorsokratiker Parmenides (6./5. Jhdt. v. Chr.), eine Definition des Begriffs liefert Aristoteles in seiner „Metaphysik“ [Ros24]:

> ἔστιν ἐπιστήμη τις ἥ ϑεωρεῖ τὸ ὂν ᾗ ὂν [...]
> Es gibt eine Art Wissenschaft, die das Seiende an sich betrachtet [...]

Eine „informatische“ Definition findet sich in [Gru93]:

> An ontology is an explicit specification of a conceptualization

Diese Definition wurde später von Studer et al. [SBF97] erweitert zu

> An ontology is a formal, explicit specification of a shared conceptualisation.
> Eine Ontologie ist eine formale, explizite Spezifikation einer gemeinsamen Konzeptualisierung (also einer abstrakten, modellhaften Sicht auf einzelne Aspekte der Welt).

Aus diesen Definitionen folgt, dass „Ontologie“ in der Informatik keine Philosophische oder sonstige wissenschaftliche Disziplin meint, sondern eine Strukturierung von Wissen in einer bestimmten Domäne. Zentral sind dabei folgende Begriffe:

formal	Die Formalisierung von Wissen macht es maschinenverarbeitbar.
explicit specification	Konzepte, Ihre Eigenschaften und Beziehungen sind explizit festgelegt, d. h. Wissen über sie wurde externalisiert.
shared	Das derart festgehaltene Wissen steht auf der Basis eines Konsens innerhalb einer bestimmten Gemeinschaft.
conceptualization	Eine Ontologie ist ein abstraktes Modell eines Phänomens.

Tabelle 2.2: Zentrale Eigenschaften von Ontologien

2.4.2.1 Einsatzziele von Ontologien

Mit den oben genannten Eigenschaften können Ontologien in der Informatik als Wissensbasen verwendet werden, wie auch schon von [SPKR96] vorgesehen:

> An ontology is a set of structured terms that describes some domain or topic. The idea is that an ontology provides a skeletal structure for a knowledge base.

Zusätzlich zu diesem Einsatzzweck dienen Ontologien nach [UG96] auch noch den folgenden Zwecken. Jeder Einsatzzweck ist dabei schon durch die vier genannten Eigenschaften von Ontologien motiviert:

- Festlegung eines Vokabulars
- Kommunikation
- Inferenz
- Repräsentation von Wissen
- Wiederverwendung von Wissen

Vokabular Dadurch dass eine Ontologie eine explizite (also greifbar niedergeschriebene) Formalisierung von Wissen ist, legt sie ein Vokabular mit einer wohldefinierten Semantik für einzelne Konzepte fest, so dass ein Satz von Bezeichnern für diese Konzepte und ihre Beziehungen untereinander entsteht: ein Vokabular, das dem terminologischen bzw. Schemawissen einer Domäne entspricht. Durch die gemeinsame Anwendung des Vokabulars wird eine Ontologie somit zur Externalisierung des gemeinsamen Verständnisses dieser Konzepte und ihrer Bedeutung. Da das Semantic Web eine offene Architektur ist, an der prinzipiell jeder teilnehmen kann („Anyone can say Anything about Any topic – the AAA assumption“, vgl. [AH08].) und nicht ausgeschlossen werden kann, dass unterschiedliche Parteien dieselben Konzepte unterschiedlich bezeichnen, können nach wie vor Mehrdeutigkeiten hinsichtlich des Vokabulars auftreten. Durch Äquivalenzrelationen zwischen einzelnen Konzepten verschiedener Ontologien können solche unterschiedlichen Sichtweisen über Anwendungs- oder Komponentengrenzen hinweg integriert werden bzw. Abbildungen zwischen Konzepten gefunden werden.

Kommunikation Weil ein gemeinsames Verständnis des festzulegenden Vokabulars nur durch die Kommunikation des in ihm gespeicherten Wissens erreicht werden kann, dient eine Ontologie auch immer der Verbreitung. Bei der Erstellung einer Ontologie ist in iterativen Schritten immer wieder der Transport des Wissens zwischen allen bei der Erstellung beteiligten Parteien notwendig, um zum gemeinsamen Verständnis zu kommen. Eine Ontologie dient hier also als Vehikel zur Kommunikation von Wissen. Später, beim Einsatz

der Ontologie, wird Wissen ein zweites Mal kommuniziert. Falls sich ein Anwendungsdesigner entscheiden sollte, eine bestimmte Ontologie zu verwenden, dann wird ihm durch die Ontologie das in ihr externalisierte Wissen mitgeteilt, auch hier dient sie also als Kommunikationsmittel. Drittens ist sie auch dadurch ein Kommunikationsmittel, dass sie verwendet wird, um implizites Wissen zu externalisieren, also Wissen, das nur einer Person implizit zugänglich ist, durch Manifestation in einer Ontologie der Außenwelt zugänglich zu machen. Weil das Wissen selbst in kleinen Anwendungsdomänen zu komplex ist, um von einer einzelnen Person explizit niedergeschrieben zu werden, werden Ontologien üblicherweise von mehreren Personen verfasst, die zu diesem Zweck kommunizieren müssen.

Inferenz Durch die Verteiltheit der Ontologieerstellung und das Zustandekommen des gemeinsamen Verständnisses mit Hilfe verschiedener Ontologien mit gleichen oder ähnlichen Konzepten kann es bei der Überlagerung von mehreren Ontologien natürlich zu Inkonsistenzen kommen. Diese Inkonsistenzen können mit der Hilfe von bestimmten Werkzeugen, den sogenannten *Reasonern*, entdeckt und u. U. aufgelöst werden. Neben der Konsistenzprüfung können Reasoner durch automatisches Schließen auch aus vorhandenem Wissen und in der Ontologie gespeicherten Annahmen, Vermutungen und Regeln (*Axiome*) neues Wissen erschließen. Ein Reasoner wird also – ähnlich wie Ontologien selbst – dazu verwendet, implizit in der Wissensbasis gespeichertes Wissen in explizites Wissen umzuwandeln. Dieser Prozess heißt *Reasoning*, wobei das automatisierte, computergestützte Schließen auch *Inferenz* genannt wird. Im Allgemein sind Inferenzmaschinen im Bereich des Semantic Web nur in der Lage, deduktive Schlüsse zu ziehen, d. h. vom Allgemeinen auf das Besondere zu schließen, also neue speziellere Wissensaussagen zu treffen, die dann der Wissensbasis hinzugefügt werden können. Dazu bedienen sie sich eines formalen Logikkalküls (Im Falle von Ontologien: Prädikatenlogik unterschiedlicher Stufen – je nach Ausdrucksmächtigkeit der Ontologien, siehe auch Kapitel 2.4.2.2). Abduktive Schlüsse, also das Aufstellen neuer Hypothesen bzw. Auffinden neuer Axiome ist aufgrund „weicherer" Eingangsvoraussetzungen (z. B. Wahrscheinlichkeiten) schwierig durchzuführen. Aus ähnlichen Gründen sind Reasoner nicht in der Lage, induktive Schlüsse, also vom Speziellen zum Allgemeinen hin durchzuführen. Beispiele für Reasoner sind in Tabelle 2.3 aufgelistet.

Repräsentation von Wissen Ontologien modellieren Wissen als *Konzepte*, die wiederum instantiiert werden können. Derart entstehende *Instanzen* werden Fakten *genannt* und entsprechen Entitäten der realen Welt. Damit sind ontologische Modelle durchaus vergleichbar mit Konzepten aus der objektorientierten Modellierung. Das gleichermaßen verfolgte Ziel ist die Errichtung einer Klassenhierarchie, indem von allgemeineren Oberklassen speziellere Subklassen mit verfeinerten Eigenschaften abgeleitet werden, was somit zu einer Strukturierung des Wissensraumes dient. Instanzen wiederum werden an Klassen gebunden, um ihnen Typen zuzuweisen. Im Gegensatz zu den meisten objektorientierten Programmier-

CEL	http://lat.inf.tu-dresden.de/systems/cel/
Cerebra Engine	http://www.cerebra.com/
FaCT++	http://owl.man.ac.uk/factplusplus
KAON2	http://kaon2.semanticweb.org/wiki/KAON
MSPASS	http://www.cs.man.ac.uk/~schmidt/mspass/
Pellet	http://pellet.owldl.com/
RacerPro	http://www.racer-systems.com/de/index.phtml?lang
SimDL	http://sim-dl.sourceforge.net/

Tabelle 2.3: Übersicht über Reasoner-Implementierungen

sprachen unterstützen Ontologien die Mehrfachvererbung, so dass Konzepte und Instanzen von mehreren Oberklassen gleichzeitig erben können. Die Beschreibung der Beziehungen zwischen Konzepten geht bei ontologischen Modellen i. A. über das hinaus, was objektorientierte Programmiersprachen zulassen: hier ist oft nur eine Subklassen-Beziehung möglich, während Klassen in Ontologien z. B. auch als disjunkt, äquivalent, komplementär oder als Ergebnisklasse aus dem Durchschnitt zweier Klassen entstanden gekennzeichnet werden können. Ein weiterer Unterschied ergibt sich dadurch, dass in Ontologien nicht nur Klassen miteinander in Beziehung gesetzt werden können, sondern auch Instanzen. Reasoning ist sowohl auf Klassen- als auch auf Instanzebene möglich. Für die Modellierung birgt das einige Schwierigkeiten, da es möglich ist, bestimmte Entitäten sowohl als Klassen als auch als Instanzen zu betrachten: Zum Beispiel kann eine Entität „Pferd“ als Instanz oder als Subklasse der Klasse „Säugetier“ betrachtet werden. Keine der beiden Betrachtungsweisen ist definitiv falsch oder richtig, je nach Einsatzzweck ist eine der beiden eher gewünscht.

Wiederverwendung von Wissen Dadurch dass Ontologien gemeinsame Konzeptualisierungen sind, lässt sich ein weiterer Einsatzzweck direkt ableiten. Die „Gemeinsamkeit“ kommt erst dadurch zustande, dass Ontologien mehrfach von unterschiedlichen Parteien verwendet werden - Ontologien dienen also der Wiederverwendung von Wissen. Um diesem Zweck gerecht zu werden, ist eine gewisse Vollständigkeit der Ontologie notwendig, die sich aber erst im Laufe ihres Gebrauchs ergeben kann. Zugute kommt dem ontologischen Konzept hier, dass sie verfügbares Wissen in Domänen partitionieren können und dann nur innerhalb dieser Domäne gültig sind. Somit lässt sich die Qualität einer ontologischen Modellierung auch immer an ihrem Verbreitungsgrad messen. Je mehr Nutzer sich zur Verwendung einer bestimmten Ontologie bekennen, desto größer ist der gefundene Konsens über die in ihr modellierten Konzepte – Ontologien unterliegen also direkten Netzwerkeffekten, d. h. ihr „Wert“ steigt mit zunehmender Verbreitung.

2.4.2.2 Ontologiebeschreibungssprachen

Das Semantic Web selbst legt keine Sprache zur Modellierung von Ontologien fest, vielmehr ist die Leistungsfähigkeit des Reasoners und die Expressivität einer Beschreibungssprache bestimmt durch die Ausdrucksmächtigkeit der ihr zugrunde liegenden Logikkalküle, die hier kurz, aber keinesfalls vollständig vorgestellt werden sollen:

Aussagenlogik Nach [Sch00] verknüpft die Aussagenlogik atomare sprachliche Gebilde mit der Hilfe von logischen Operatoren. Zulässige Operatoren sind

$\neg$	logische Negation
$\wedge$	ODER-Verknüpfung
$\vee$	UND-Verknüpfung

Tabelle 2.4: Aussagenlogische Operatoren

Die Operatoren $\rightarrow$ (Implikation) und $\leftrightarrow$ (Bikonditional, hinreichende und notwendige Bedingung) lassen sich aus den in Tabelle 2.4 angegebenen ableiten. Für eine detaillierte Beschreibung der formalen Semantik der Aussagenlogik, ihrer Formeln und Operatoren sei auf [Sch00] verwiesen. Die Aussagenlogik ist nicht ausdrucksmächtig genug, um darzustellen, dass gewisse Entitäten in bestimmten Beziehungen stehen, und auch nicht, ob diese Beziehung für alle oder für mindestens eine Variable gültig ist. Diese Erweiterung wird durch die Prädikatenlogik vorgenommen.

Prädikatenlogik Die Prädikatenlogik ist eine Erweiterung der Aussagenlogik um neue Quantoren und Funktions- und Prädikatsymbole. Prädikate sind dabei Aussagen über bestimmte Subjekte der Form „Apoll ist ein Sohn des Zeus“. Das Prädikat ist hierbei „$\__1$ ist ein Sohn des $\__2$“. „$\__1$“ und „$\__2$“ bezeichnen die Stellen, an denen die Individuen, über die eine Aussage getroffen wird, eingesetzt werden. Da in diesem Beispiel zwei Individuen benötigt werden, spricht man von einem zweistelligen Prädikat, ein- oder mehrstellige sind ebenfalls möglich. Zusätzlich zu Prädikaten gibt es zwei Quantoren: denn Allquantor $\forall$, wenn eine Aussage für alle Individuen wahr sein soll und den Existenzquantor $\exists$ für mindestens ein Individuum gültig sein soll. Von Prädikatenlogik erster Stufe spricht man, wenn die Argumente aller verwendeten Prädikate Individuen sind, von einer Prädikatenlogik zweiter Stufe, wenn die Argumente auch Prädikate erster Stufe umfassen dürfen.

Ontologische Notationsformen in OWL sind – je nach Sprachausprägung – prädikatenlogische Ausdrücke erster oder höherer Stufe. Für eine detailliertere Einführung in die Prädikatenlogik und ihre formale Semantik sei wiederum auf [Sch00] verwiesen.

Beschreibungslogik Beschreibungslogiken sind Untermengen der Prädikatenlogik erster Stufe. Diese Untermengen werden deshalb gebildet, da die Prädikatenlogik erster Stufe i. A. nicht entscheidbar ist. Entscheidbar ist eine Aussage dann, wenn die Frage, ob sie aus einer Ontologie gefolgert werden kann, in endlicher Zeit beantwortbar ist [HKRS08]. Details zu den Einschränkungen, die den unterschiedlichen OWL-Ausprägungen gemacht wurden, sind im folgenden Kapitel zu finden.

2.4.3 OWL

Die *Web Ontology Language* (OWL)[4] wurde ab November 2001 als Nachfolger der schon bestehenden Ontologiesprache DAML-OIL entwickelt. Erste Versionen des W3C-Sprachstandards zu OWL gab es ab Juli 2002, ab Februar 2004 wurde OWL in Form einer finalen W3C Recommendation verabschiedet [MH04]. In OWL spezifizierte Sachverhalte können in weit stärkerem Maß von Maschinen automatisch interpretiert werden als das zuvor mit XML, RDF und RDF Schema (RDFS) möglich war. Zu diesem Zweck definiert OWL ein erweitertes Vokabular zusammen mit einer formalen Semantik für dieses Vokabular, basiert aber weiterhin auf XML, RDF und RDFS. Genauer gesagt: OWL ist in RDFS definiert. Somit stellt jedes gültige OWL-Dokument auch ein gültiges RDF-Dokument dar. Mit seinen Erweiterungen geht OWL aber in der Ausdrucksstärke über RDF hinaus. Der OWL-Standard spezifiziert drei Teilsprachen, die jeweils unterschiedliche Mächtigkeit haben (vgl. [HKRS08]). Grundlegend ist aber, dass alle auf dieselben Basisfähigkeiten zur Modellierung, nämlich Klassen und ihnen zugeordnete Eigenschaften (*Properties*), Instanzen (*Individuen*) und deren Beziehungen untereinander zurückgreifen. Falls Individuen mit Individuen in Beziehung gesetzt werden, spricht man von *abstrakten Rollen*. *Konkrete Rollen* setzen Individuen mit Datenwerten in Beziehung. Um z. B. Klassenhierarchien zu erstellen ist es nicht nötig, auf den OWL-Sprachschatz zurückzugreifen, das RDFS-Konstrukt `rdfs:subClassOf` wird in diesem Falle verwendet. Da es an dieser Stelle zu weit führen würde, eine Einführung in RDF und RDFS zu geben, sei nur auf entsprechende Literatur ([KC04], [Bec04], [BG04]) verwiesen.

[4] [HKRS08] macht mehrere Angaben über die Herkunft des Buchstabendrehers im Akronym OWL:

- Im Gegensatz zu WOL ist bei OWL sofort klar, wie die Abkürzung ausgesprochen werden muss, nämlich wie das englische Wort für „Eule“.
- OWL-Logos sind leicht zu erstellen.
- Eulen werden mit Weisheit assoziiert.
- In den 70er Jahren wurde am MIT schon von William A. Martin versucht, eine universelle Beschreibungssprache, die One World Language, zu spezifizieren.
- Die Eule in Alan Alexander Milnes Buch „Winnie the Pooh“ war auch schon nicht imstande, ihren Namen „Owl“ korrekt zu buchstabieren und schrieb immer „Wol“.

- **OWL Full**
  - enthält sowohl OWL DL und OWL Lite als Teilsprachen
  - enthält den kompletten RDFS-Standard
  - ist unentscheidbar (Nachweis in [HKS06] und [HS03])
  - ist ausdrucksstärkster OWL-Dialekt
  - entspricht der Beschreibungslogik $\mathcal{sROIQ}$:
- **OWL DL**
  - ist Teilsprache von OWL Full
  - enthält OWL Lite als Teilsprache
  - Komplexität der Entscheidbarkeitsfrage: NExpTime (Nachweis in [Tob96] und [Tob01])
  - entspricht der Beschreibungslogik $\mathcal{SHOIN}$:
- **OWL Lite**
  - ist Teilsprache von OWL Full und OWL DL
  - Komplexität der Entscheidbarkeitsfrage: ExpTime (Nachweis in [Tob01])
  - ist am wenigsten ausdrucksstärkster OWL-Dialekt
  - entspricht der Beschreibungslogik $\mathcal{SHIF}$:

Dabei entspricht jeder OWL-Dialekt einer bestimmten Beschreibungslogik. Diese Beschreibungslogiken werden durch Buchstaben beschrieben, wobei jeder Buchstabe für eine oder mehrere Spracheigenschaften steht. Jedem OWL-Dialekt liegt die Beschreibungslogik $\mathcal{ALC}$ zugrunde, die die folgenden Eigenschaften unterstützt [HKRS08]:

$\mathcal{ALC}$	Klassen, Rollen und Individuen
	Klassenzugehörigkeit und Instanzen
	die leere Klasse `owl:Nothing` und die universelle Superklasse `owl:Thing`
	Klasseninklusion, Klassenäquivalenz und Klassendisjunktheit
	Konjunktion von Klassen
	Negation von Klassen
	Rollenrestriktionen
	Wertebereiche und Definitionsbereiche für Rollen

Tabelle 2.5: Eigenschaften der Beschreibungslogik $\mathcal{ALC}$

OWL Lite erweitert diese Fähigkeiten und entspricht dann der Beschreibungslogik $\mathcal{SHIF}$:

$\mathcal{S}$	Alle Eigenschaften aus $\mathcal{ALC}$ plus transitive Rollen
$\mathcal{H}$	Rollenhierarchien
$\mathcal{I}$	inverse Rollen
$\mathcal{F}$	funktionale Rollen

Tabelle 2.6: Eigenschaften der Beschreibungslogik $\mathcal{SHIF}$

OWL DL stellt eine zusätzliche Erweiterung dar. Das Ergebnis entspricht der Beschreibungslogik $\mathcal{SHOIN}$:

$\mathcal{S}$	Alle Eigenschaften aus $\mathcal{ALC}$ plus transitive Rollen
$\mathcal{H}$	Rollenhierarchien
$\mathcal{O}$	abgeschlossene Klassen
$\mathcal{I}$	inverse Rollen
$\mathcal{N}$	Zahlenrestriktionen

Tabelle 2.7: Eigenschaften der Beschreibungslogik $\mathcal{SHOIN}$

Der mächtigste OWL-Dialekt ist OWL Full, er entspricht der Beschreibungslogik $\mathcal{sROIQ}$:

s	Alle Eigenschaften aus $\mathcal{ALC}$ plus transitive, disjunkte, reflexive, irreflexive und asymmetrische Rollen (nur für einfache Rollen); universelle Rolle; negierte Aussagen über Rollen; „lokal reflexive“ Rollen
$\mathcal{R}$	komplexe Rollenhierarchien
$\mathcal{O}$	abgeschlossene Klassen
$\mathcal{I}$	inverse Rollen
$\mathcal{Q}$	qualifizierende Kardinalitätseinschränkungen

Tabelle 2.8: Eigenschaften der Beschreibungslogik $\mathcal{sROIQ}$

Im folgenden sollen die OWL-Sprachelemente kurz vorgestellt werden. Dabei wird nur eine „intuitive“ Semantik der einzelnen Sprachkonstrukte angegeben, die leicht verständlich sein soll. Für genaue Definitionen und exakte formale Semantik sei auf [BHH+04], [PSHH04] und [HKRS08] verwiesen. Ein Stern in der ersten Spalte bedeutet, dass dieses Sprachelement in OWL Lite nur eingeschränkt oder überhaupt nicht zu Verfügung steht.

	`owl:AllDifferent`	Kennzeichnet die Verschiedenheit einer Menge von Individuen
	`owl:allValuesFrom`	Rolleneinschränkung: alle Rollen müssen einen Wert aus einer vorgegebenen Klasse annehmen, entspricht ungefähr dem Allquantor $\forall$

	owl:AnnotationProperty	für Annotationen
	owl:backwardCompatibleWith	Verweis auf eine weitere Ontologie, die als kompatible frühere Version der vorliegenden Ontologie aufgefasst wird
⋆	owl:cardinality	exakte Kardinalitätseinschränkung für Rollen, kann auch aus owl:maxCardinality und owl:minCardinality zusammengesetzt werden
	owl:Class	erzeugt eine Klasse
⋆	owl:complementOf	erzeugt eine komplexe Klasse als Komplement einer anderen Klasse
⋆	owl:DataRange	erzeugt einen Aufzählungsdatentypen oder eine Aufzählungsklasse
	owl:DatatypeProperty	erzeugt eine konkrete Rolle, also eine Eigenschaft mit einem Datenwert
	owl:DeprecatedClass	erzeugt eine Klasse, die aufgrund ihrer Veralterung nicht mehr verwendet werden sollte und nur noch aus Kompatibilitätsgründen in die Ontologie aufgenommen wurde
	owl:DeprecatedProperty	erzeugt eine Eigenschaft, die aufgrund ihrer Veralterung nicht mehr verwendet werden sollte und nur noch aus Kompatibilitätsgründen in die Ontologie aufgenommen wurde
	owl:differentFrom	kennzeichnet zwei Individuen als verschieden
⋆	owl:disjointWith	kennzeichnet zwei Klassen als disjunkt
	owl:distinctMembers	verknüpft eine Instanz von owl:AllDifferent mit einer Liste von Individuen und wird auch nur in diesem Fall gebraucht
	owl:equivalentClass	kennzeichnet zwei Klassen (mit unterschiedlichen Namen) als identisch, d. h. sie modellieren das gleiche Konzept
	owl:equivalentProperty	kennzeichnet zwei Eigenschaften (mit unterschiedlichen Namen) als identisch
	owl:FunctionalProperty	kennzeichnet eine Eigenschaft als funktional, so dass sie pro Instanz nur genau einen Wert annehmen kann

	owl:hasValue	ein Spezialfall von owl:someValuesFrom, bei dem eine konkrete Instanz als Wert der Rolle angegeben wird
	owl:imports	importiert eine andere Ontologie
	owl:incompatibleWith	Verweis auf eine weitere Ontologie, die als inkompatible frühere Version der vorliegenden Ontologie aufgefasst wird
	owl:intersectionOf	erzeugt eine komplexe Klasse als Schnittmenge mehrerer anderer Klassen
	owl:InverseFunctionalProperty	kennzeichnet die Inverse einer Rolle als funktional
	owl:inverseOf	kennzeichnet ein Rolle als invers zu einer anderen
⋆	owl:maxCardinality	maximale Kardinalitätseinschränkung für Rollen
⋆	owl:minCardinality	minimale Kardinalitätseinschränkung für Rollen
	owl:Nothing	die leere Klasse. Alle Klassen haben owl:Nothing als Unterklasse
	owl:ObjectProperty	erzeugt eine abstrakte Rolle
⋆	owl:oneOf	wird verwendet, um Klassenbeschreibungen durch Aufzählung zu realisieren, d. h. eine Klasse wird durch die Aufzählung ihrer Instanzen beschrieben
	owl:onProperty	wird verwendet, um eine Einschränkung (owl:Restriction) an eine bestimmte Eigenschaft zu binden
	owl:Ontology	zur Angabe allgemeiner Information über eine Ontologie (wird üblicherweise im Kopf einer Ontologie deklariert)
	owl:OntologyProperty	wird verwendet, um Aussagen über die Eigenschaften von Ontologien zu treffen. owl:imports, owl:priorVersion, owl:backwardCompatibleWith und owl:incompatibleWith sind z. B. Instanzen von owl:OntologyProperty

	owl:priorVersion	Verweis auf eine weitere Ontologie, die als frühere Version der vorliegenden Ontologie aufgefasst wird
	owl:Restriction	erzeugt eine Einschränkung, die mittels owl:onProperty an eine Eigenschaft gebunden wird
	owl:sameAs	kennzeichnet zwei Individuen als identisch
	owl:someValuesFrom	Rolleneinschränkung: mindestens eine Rolle muss einen Wert aus einer vorgegebenen Klasse annehmen, entspricht ungefähr dem Existenzquantor ∃
	owl:SymmetricProperty	kennzeichnet eine Eigenschaft als symmetrisch
	owl:Thing	die universelle Klasse. Alle Klassen sind Unterklassen von owl:Thing
	owl:TransitiveProperty	kennzeichnet eine Eigenschaft als transitiv
⋆	owl:unionOf	erzeugt eine komplexe Klasse als Vereinigung mehrerer anderer Klassen
	owl:versionInfo	wird verwendet, um Ontologien zu versionieren

Tabelle 2.9: Übersicht über die OWL-Sprachelemente

2.5 Zusammenfassung

Im Verlauf von Kapitel 2 wurden die wesentlichen Grundlagen dieser Arbeit vorgestellt: die Begriffe „Kontext“ und „Kontextbewusstsein“ wurden definiert und ein kurzer Abriss über die historische Entwicklung dieser Begriffe wurde gegeben. Ferner wurde auf die Eigenschaften von VANETs, insbesondere auf der Ebene der Bitübertragungsschicht, des Medienzugriffs und des Routings eingegangen. Der vorangegangene Abschnitt stellte auch Verfahren zur Informationsdissemination in VANETs vor. Die Zielsetzungen des Semantic Web und Ontologien als sein grundlegendes Modellierungskonzept wurden vorgestellt, inklusive einer Erläuterung der Verbindung zwischen OWL und Beschreibungslogiken. Zuletzt wurden die Sprachelemente von OWL dargestellt, um die Ausdrucksmächtigkeit von OWL zu illustrieren.

3 Anforderungsanalyse

Durch die Möglichkeit, Daten auszutauschen, sei es mit anderen Fahrzeugen oder mit Infrastrukturkomponenten, ergeben sich ganz neue Rahmenbedingungen zur Realisierung von Anwendung zur Umsetzung. Durch den Datenaustausch wird es nun möglich, Entscheidungen im Straßenverkehr nicht länger nur auf der Basis lokal vorhandenen Wissens zu treffen, sondern auch Daten entfernter Fahrzeuge zu verwenden. Dadurch können Anwendung kooperativ und kollaborativ miteinander arbeiten, aber nur, wenn die Datengrundlage eines anderen Fahrzeug der eigenen Anwendung verständlich ist. Daher müssen kollaborative Anwendungen über ein gemeinsames Verständnis der Kontextdaten, mit denen sie arbeiten verfügen, damit gesichert ist, in welcher Situation mit welchen Techniken für welche Arbeiten zu welchem Zweck kooperiert wird. Zu diesem Zweck sollen im folgenden Abschnitt exemplarisch VANET-Anwendungen betrachtet werden, auf deren Grundlage dann Anforderungen an das Kontextmodell abgeleitet werden. Zusätzlich zu den Anwendungsbeschreibungen wird in Kapitel 3.4 ein Einblick in Probleme bei der Markteinführung solcher Anwendungen gegeben, da diese auch Auswirkungen auf die Anforderungen haben.

3.1 Analyse der Fahraufgabe

In modernen Fahrzeugen kommen immer mehr Assistenzsysteme, die den Fahrer bei der Durchführung seiner Aufgabe unterstützen sollen, zum Einsatz. Je nach Assistenzsystem ist die Unterstützung dabei mehr oder weniger eng an die Erfüllung der Fahraufgabe gekoppelt, die von [Don99, Don76] wie folgt definiert wird:

Navigation (navigation) bezeichnet die Wahl der Fahrtroute und des zeitlichen Ablaufs der Fahrt. Navigation, also das Auswählen einer geeigneten Strecke, wird nicht nur vor der Fahrt durchgeführt, sondern muss unter Umständen vom Fahrer während der Fahrt aufgrund von z. B. Staus oder Unfällen erneut durchgeführt werden.

Führung (guidance) bezeichnet die Festlegung von Soll-Variablen, wie z. B. Soll-Kurs und Soll-Geschwindigkeit während des Fahrens und das Einhalten dieser Variablen während der Fahrt und Einbeziehung der gegebenen Fahrsituation. Auch das Durchführen von Fahrmanövern wie z. B. Überholmanövern oder Spurwechseln zählt dazu.

Stabilisierung (stabilisation) bezeichnet die Regelung der Längs- und Querbeschleunigungen des Fahrzeugs – auf diese Stufe der Fahraufgabe nimmt der Fahrer Handlungen vor, um eventuelle Abweichungen zwischen den Größen der Führungsebene ist Ist-Werten auszugleichen.

Die Zeithorizonte für die Durchführung dieser Aufgaben ändert sich mit zunehmender Spezialisierung der Aufgabe deutlich: während für die Durchführung der Navigation noch mehrere Stunden (Routenwahl vor der Fahrt) bis Minuten (spontane Routenänderung während der Fahrt) zur Verfügung stehen, muss die Führung des Fahrzeugs im Minuten- bis Sekundenbereich durchgeführt werden. Der Zeithorizont der Führungsebene ist vom Wahrnehmungshorizont des Fahrers abhängig: zum Beispiel nimmt bei guten Sichtverhältnissen der optische Wahrnehmungshorizont (Straßengeometrie) Einfluss auf die Führung: bewusste Lenkbewegungen, Brems- und Beschleunigungsmanöver werden dann innerhalb von wenigen Sekunden antizipiert. In der Stabilisierungsebene werden Eingriffe eher auf instinktiver und fertigkeitsbasierter Ebene (vgl. [BD06]) vorgenommen und liegen damit im Millisekundenbereich. Verdeutlicht werden die unterschiedlichen Ebene der Fahraufgabe und die mit ihnen assoziierten Zeithorizonte in Abbildung 3.1.

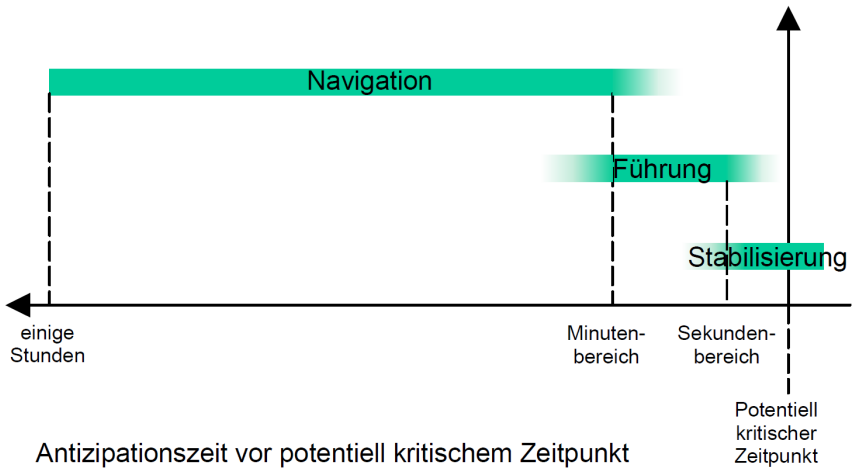


Abbildung 3.1: Die Einteilung der Fahraufgabe und assoziierte Zeithorizonte [BD06]

Aus diesen Fakten lässt sich ableiten, dass Relevanz der in jeder Ebene zu verarbeitenden Daten abhängig sind von ihrer „geographischen Herkunft“: Bei Durchführung der Navigation kann der Fahrer noch Daten miteinbeziehen, die sich nicht in seinem unmittelbaren Wahrnehmungshorizont befinden, wie z. B. Informationen aus dem Verkehrsfunk oder Verkehrsflussdaten, die ihm z. B. über ein VANET übermittelt wurden. Im Falle der Führung kann und sollte der Fahrer Daten aus seinem gesamten unmittelbaren Wahrnehmungshori-

zont einbeziehen, während die zu verarbeitenden Daten sich bei der Stabilisationsaufgabe nur noch auf einen sehr kleinen, eingeschränkten Wahrnehmungshorizont beziehen, nicht zuletzt weil im letzteren Fall schon die Übermittlung der Daten über ein VANET u. U. zu viel Zeit in Anspruch nehmen könnte. Die hier gemachten Annahmen hinsichtlich Zeithorizont und Lokalitätseigenschaften lassen sich natürlich auch vom Fahrer auf die ihn unterstützenden Assistenzsysteme übertragen. Auch für sie gilt:

- unterstützen sie ihn bei der Durchführung der Navigation, so können sie auch weit entfernte Daten miteinbeziehen und müssen nur schwache Anforderungen hinsichtlich Latenzzeiten einhalten
- je näher die Unterstützung des Fahrers an die Stabilisierungsebene gekoppelt ist, desto kleiner wird der Radius der einzubeziehenden Daten und desto härter werden die Anforderungen hinsichtlich Latenz

Tabelle 3.1 (in Anlehnung an [Str07]) verdeutlicht die Anforderungen von Fahrerassistenzanwendungen.

	Ebene der Fahraufgabe		
	Navigation	Führung	Stabilisierung
Relevanter Datenradius			
Latenzanforderung			

Tabelle 3.1: Latenz- und Lokalitätsanforderungen von Fahrerassistenzanwendungen in Abhängigkeit der Fahraufgabenebene

Nicht nur die Menge der für eine Fahrerassistenzanwendung relevanten Daten und die maximal zulässige Latenz sind abhängig vom Kontext eines Fahrzeugs bzw. des Fahrers, sondern auch die Art und Weise, wie ihm eine eventuelle Warnung präsentiert wird. So ist es z. B. denkbar, dem Fahrer je nach Fahrsituation, Entfernung oder Gefahrenpotential einer Gefahrenstelle oder eine Warnung unterschiedlich zu präsentieren:

- Optisch, z. B. durch Einblenden eines Warnungssymbols in das Navigationssystem oder direkt auf die Windschutzscheibe mittels Head-up-Display (siehe Abbildungen 1.6 und 1.7
- Akustisch durch Abspielen eines Warntones
- Haptisch, z. B. durch Vibrationen im Lenkrad oder Gegendruck an den Pedalen

Allerdings würde eine genaue Erörterung und Evaluation aller Möglichkeiten hinsichtlich der Benutzerinteraktion an dieser Stelle zu weit führen und liegt auch nicht im Fokus dieser Arbeit. Es sei lediglich festgehalten, dass eine optimale Präsentation von Nachrichten

im Fahrzeug vom aktuellen Nutzer- und Nutzungskontext abhängig ist.

Wie schon im Verlauf diese Abschnitts dargestellt, lassen sich Fahrerassistenzanwendungen in Abhängigkeit von Ihrer Relevanz für die Durchführung der Fahraufgabe einteilen. Eine gröbere Einteilung nehmen die folgenden beiden Kapitel vor: wenn Anwendungen direkten Einfluss haben auf die Fahrsicherheit, denn ist von *aktiven Sicherheitsanwendungen* die Rede, alle anderen Anwendungen, die eher aus dem Bereich Information, Unterhaltung und Erhöhung des Fahrkomforts stammen, werden unter dem Begriff *Deployment-Anwendungen* zusammengefasst. Zur näheren Erläuterung des letztgenannten Begriffes sei auf Kapitel 3.4 verwiesen.

3.2 Aktive Sicherheitsanwendungen

Diese Anwendungs-Klasse dient hauptsächlich der Erhöhung der Sicherheit im Straßenverkehr und der Verringerung der Anzahl und Schwere von Unfällen. Sie ist also recht eng mit der Fahraufgabe verknüpft. Für eine umfassendere Aufstellung an in VANETs denkbaren aktiven Sicherheitsanwendungen sei auf [Eig05a] verwiesen.

3.2.1 Lokale Gefahrenwarnung (Local Danger Warning)

Ziel der lokalen Gefahrenwarnung (vgl. [Str07, Kos04, ZSRC08]) ist es, den Straßen- und Verkehrszustand zu beobachten und sich eventuell ergebende Gefahrensituationen an den Fahrer und umliegende Fahrzeuge zu melden. Gefahrensituationen können auftreten, wenn sich z. B. Hindernisse auf der Fahrbahn befinden, die Sicht durch Nebel eingeschränkt ist, der Kraftschluss mit der Fahrbahn durch Öl oder Eis nicht mehr gegeben ist. In diesem Fall ist die Gefahr nur von temporärer Dauer und wird hauptsächlich durch das Überwachen und Kombinieren von lokal vorliegenden Sensorwerten detektiert. Anders liegt der Fall, wenn sich die Gefahr durch die Straßengeometrie selbst ergibt, z. B. bei engen Kurven oder Brücken mit einer maximalen Durchfahrtshöhe. Derartige Information ist eher mit einer digitalen Karte verknüpft und verortet und von dauerhafter Natur. Zu beachten ist auch, dass sich manche Situationen wie Staus und die Ausdehnung von Nebelbänken nur kooperativ ermitteln lassen, oder zumindest durch Kooperation qualitativ verbessern lassen.

All diesen Anwendungen liegt zugrunde, dass sie durch Sensoren ermittelte Kontextdaten verarbeiten müssen und gegebenenfalls mit anderen Fahrzeugen austauschen müssen. Konkret wäre als Datenbedarf zu nennen:

- Ort (Geokoordinaten) und somit Distanz zwischen Fahrzeugen
- Geschwindigkeit (eigene und die anderer Fahrzeuge)

- Richtungsvektoren
- Umweltbedingungen (Temperatur, Sichtweite, Regensensor)
- Beschleunigungs- oder Verzögerungswerte (sowohl Längs- als auch Querbeschleunigung)
- Lenkwinkel
- digitale Kartendaten
- Informationen über die Straßengeometrie
- Lageinformationen über Straßenbauwerke
- Fahrzeugausdehnungen

3.2.2 Kooperative Kollisionswarnung (Cooperative Collision Warning)

Durch den Datenaustausch in VANETS sind Fahrzeuge in der Lage, zu berechnen, wann sie sich – basierend auf aktuellen Informationen zu Ort, Geschwindigkeit, Beschleunigung bzw. Verzögerung und Lenkwinkel – auf Kollisionskurs befinden. Die Anwendung manifestiert oftmals in unterschiedlichen Ausprägungen: Anwendungen zur Vermeidung von Auffahrunfällen, Spurwechselassistenten, Kreuzungsassistenten basieren aber grundsätzlich auf dem gleichen Prinzip. Anzumerken sei noch, dass bereits erhältliche Systeme wie ACC (adaptive cruise control) eine ähnliche Funktionalität bieten, nämlich die Einhaltung eines definierten Sicherheitsabstandes zum Vordermann, ihre Entscheidungen aufgrund von lokal vorhandenem Wissen fällen. Durch den Einsatz von Radar- oder Lidar-Systemen haben diese Anwendungen aber keine Rundumsicht und können den Abstand nur in eine Richtung überwachen.

Der Informationsbedarf stellt sich bei diesen Anwendungen wie folgt dar:

- Ort (Geokoordinaten) und somit Distanz zwischen Fahrzeugen
- Geschwindigkeit (eigene und die anderer Fahrzeuge)
- Richtungsvektoren
- Beschleunigungs- oder Verzögerungswerte (sowohl Längs- als auch Querbeschleunigung)

Im Falle von Kreuzungsassistenten muss ein Fahrzeug auch noch über die Information verfügen, ob es sich an einer Kreuzung befindet und benötigt deshalb Zugriff auf eine digitale Karte. Der oben skizzierte Datenbedarf deckt nur die notwendigsten Daten ab. Die Anwendungen können durch den zusätzlichen Zugriff auf Daten wie Umweltbedingungen oder

die Straßengeometrie wesentlich verbessert werden. Die Kooperative Kollisionswarnung in VANETS wird z. B. in [EGH+06, YYFK03, MH02] beschrieben.

3.2.3 Kooperative Verkehrsflussinformationen (Cooperative Traffic Information)

Aufgrund der Tatsache, dass sich Fahrzeuge selbst im Verkehrsfluß befinden sind sie optimal geeignet, ihn zu beobachten und Daten wie die aktuelle Geschwindigkeit oder Durchschnittsgeschwindigkeiten über bestimmte Streckenabschnitte hinweg aufzuzeichnen und anderen Fahrzeugen zur Verfügung zu stellen, die sich somit ein Bild über die Verkehrsdichte machen können. Eine solche Anwendung (SOTIS) wird in [WER+03] beschrieben. Sie verknüpft auf eine dezentrale Art und Weise Sensordaten von mehreren Fahrzeugen, um Verkehrsflussinformationen zu gewinnen und diese an Fahrzeuge bis in 50 km Entfernung zu verteilen. Dadurch werden Beobachtungen des Verkehrs mit an Brücken oder eigens installierten Überführungen überflüssig. Auch Verkehrsfunksysteme, die auf Meldungen von Fahrern basieren und deren Meldungen noch redaktionell überarbeitet werden müssen, würden obsolet, da die Messung durch ein betroffenes Fahrzeug selbst genauere Daten mit größerer Zeitnähe liefert.

Anwendungen zur Verkehrsflussermittlung bräuchten Zugriff auf die folgenden Daten: Der Informationsbedarf stellt sich bei diesen Anwendungen wie folgt dar:

- Ort (Geokoordinaten)
- Geschwindigkeit (eigene und die anderer Fahrzeuge)
- Durchschnittsgeschwindigkeiten auf bestimmten Streckenabschnitten und somit
- digitale Kartendaten, um eine Strecke in Segmente zerteilen zu können und die eigenen Position eine Segment zuteilen zu können.
- Datum und Uhrzeit

3.3 Deployment-Anwendungen

Deployment-Anwendungen sind nicht direkt mit der Erledigung der Fahraufgabe verbunden, sondern sollen der Erhöhung des Fahrkomforts oder der Information und der Unterhaltung dienen. Weil sie nicht notwendigerweise eine Verbindung mit anderen Fahrzeugen benötigen, sondern auch mit Infrastrukturkomponenten alleine arbeiten, können sie bei der Markteinführung (engl. „Deployment“, siehe Kapitel 3.4) unterstützend wirken. Eine ausführlichere Darstellung der in VANETs denkbaren Deployment-Anwendungen findet sich in [Eig05b].

3.3.1 Point-of-Interest Notification

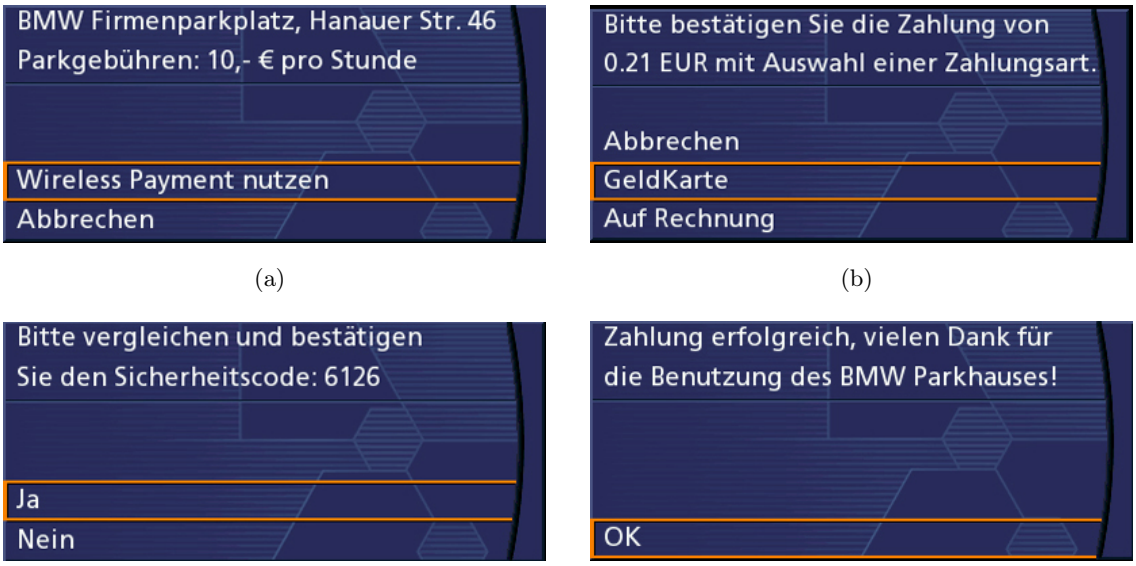
Diese Anwendung übermittelt dem Fahrer relevante Informationen über seinen aktuellen Aufenthaltsort, sein Zielort oder alle auf der Strecke befindlichen für den Fahrer interessanten Punkte. So können z. B. Tankstellen und Rastplätze ihre aktuellen Benzinpreise übermitteln oder auf ihr Serviceangebot hinweisen. Städte und Gemeinden können auf Sehenswürdigkeiten aufmerksam machen oder auf Veranstaltungen hinweisen. Diese Informationen können mit einer digitalen Karte verknüpft werden und müssen mit einer Gültigkeitsdauer versehen werden. Eine solche Anwendung namens „PoiLone“ wird z. B. in [FNS+08] beschrieben: hier wird ein Pylone mit einer Funkeinrichtung ausgestattet, der die Umgebung mit Informationen versorgt. Ein Nutzer kann dann bei Bedarf mehr Informationen anfordern. Der PoiLone kann auch dazu verwendet werden, örtlich gebundene (statische) Warnungsinformationen z. B. an Baustellen zu übermitteln oder als Weiterleitungsagent für Verkehrsfunkinformationen (TMC – Traffic Message Channel) zu dienen. Abgesehen vom eigentlichen Inhalt, braucht die Anwendung Zugriff auf die folgenden Informationen:

- Ort (Geokoordinaten)
- Datum und Uhrzeit
- evtl. digitale Karte

Zur Ortsinformation muss hier noch gesagt werden, dass es u. U. nicht ausreicht nur Geokoordinaten zu speichern. Da die Nutzer dieser Anwendungen nicht Informationen über bestimmte Ort im Sinne eines Paares aus Längen- und Breitengrad bekommen möchten, sondern über die Stadt oder das Land, in dem sie sich befinden, ist es nötig, die Koordinaten einer semantischen Ortsinformation zuzuordnen.

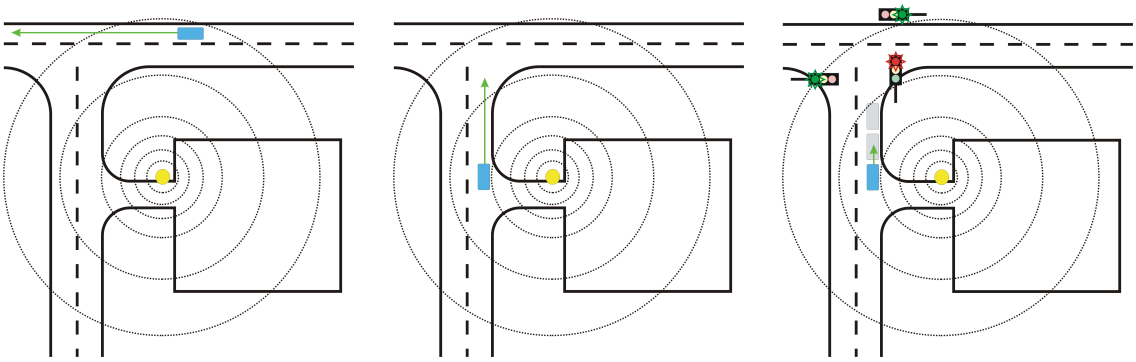
3.3.2 Mobiles Bezahlen

Da viele der Deployment-Dienste (u. a. auch die eben angesprochene Point-of-Interest Notification) eine Bezahlung erfordern, ist es auch im Sinne einer schnelleren Markteinführung einen Dienst zum drahtlosen Bezahlen direkt aus dem Fahrzeug heraus bereitzustellen. Primäre Anwendungsgebiete liegen in erster Stelle bei Parkhäusern und Tankstellen oder zur Mauterhebung. Abbildung 3.2 zeigt den exemplarischen Ablauf eines drahtlosen Bezahlvorgangs am Beispiel eines Parkhauses.
Wie in [LE06] erläutert, lässt sich auch in diesem Fall eine Verbesserung der Anwendung erreichen, wenn sie kontextsensitiv arbeitet: Ein Fahrzeug könnte sich einer Parkschranke mit drahtloser Bezahlmöglichkeit nähern, ohne dass tatsächlich ein Bezahlwunsch vorliegt. Abbildung 3.3 verdeutlicht die Situation.



(a) (b) (c) (d)

Abbildung 3.2: Ein drahtloser Bezahlvorgang



(a) (b) (c)

Abbildung 3.3: Kontextsensitive Dienstinitiierung im Falle von Drive-Through-Payment

(a) Das Fahrzeug befindet sich zwar in Funkreichweite des Dienstes, fährt aber an ihm vorbei

(b) Das Fahrzeug fährt zwar auf den Dienst zu und befindet sich Funkreichweite, will ihn aber nicht nutzen

(c) Das Fahrzeug befindet sich in Dienstreichweite und stoppt, was auf eine beabsichtigte Dienstnutzung schließen lassen könnte. Dennoch hält das Fahrzeug nur verkehrsbedingt und möchte den Dienst nicht nutzen.

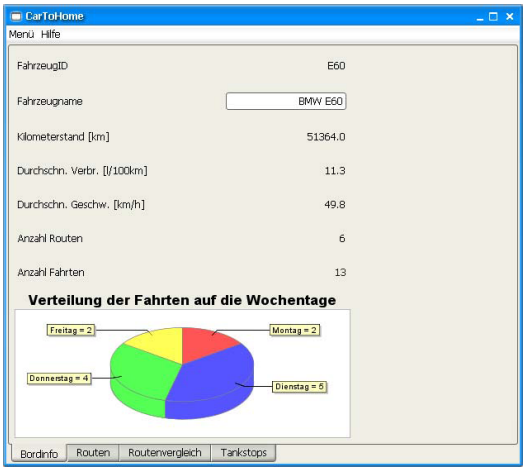
Solche unerwünschten Situationen lassen sich durch den Einsatz von zusätzlichen Kontextdaten vermeiden:

- Ort (Geokoordinaten)
- Geschwindigkeit
- Fahrtrichtung
- Lenkeinschlag
- Zündschlüsselstatus

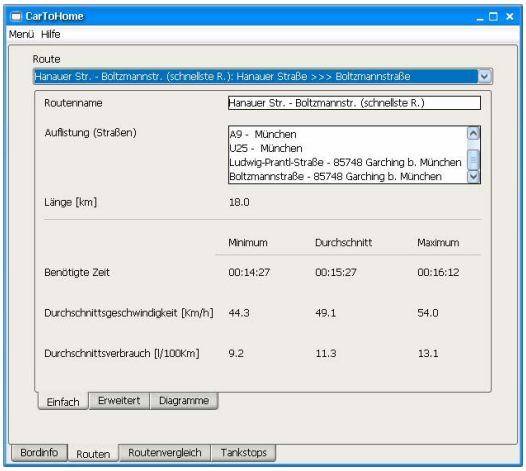
Durch die Kombination dieser Informationen lässt sich – wie in [Lin06] beschrieben – eine unerwünschte Dienstinitiierung unterdrücken.

3.3.3 Car2Home-Anwendungen

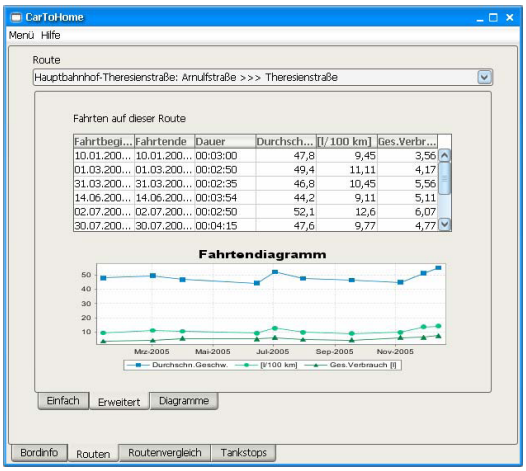
Unter Car2Home-Anwendungen versteht man eine ganze Klasse von Anwendungen, die vom heimischen PC aus auf das Fahrzeug über eine drahtlose Verbindung zugreifen. Sinn und Zweck ist es, den Zustand des Fahrzeugs bequem von zu Hause aus überwachen zu können und je nach Implementierung noch Licht ausschalten zu können, das Fahrzeug abzusperren oder die Standheizung einzuschalten. Zusätzlich ist es möglich, statistische Daten wie z. B. Durchschnittsgeschwindigkeiten, Verbrauch oder die zurückgelegte Strecke im Fahrzeug aufzeichnen zu lassen, um sie dann später am PC abzurufen, weiter zu verarbeiten oder ein Fahrtenbuch zu erstellen. Ein Beispiel für eine solche Anwendung ist in Abbildung 3.4 zu sehen.



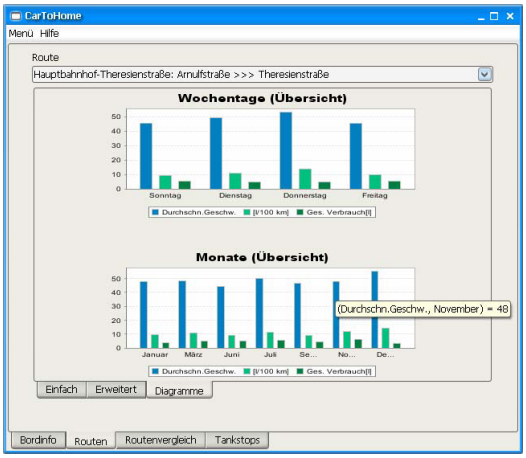
(a)



(b)



(c)



(d)

(e)

Abbildung 3.4: Car2Home-Anwendung zur Verwaltung von statistischen Fahrzeugdaten

Eine weitere Anwendungsmöglichkeit ist die Verwaltung von Musikdateien am PC, die dann ins Fahrzeug überspielt werden. Eine solche Anwendung ist am PC leichter zu realisieren, da diese über bessere Eingabemöglichkeiten verfügen als moderne Fahrzeuge mit ihren Drück-/Drehreglern. Eine Verwaltungsanwendung zeigt Abbildung 3.5.

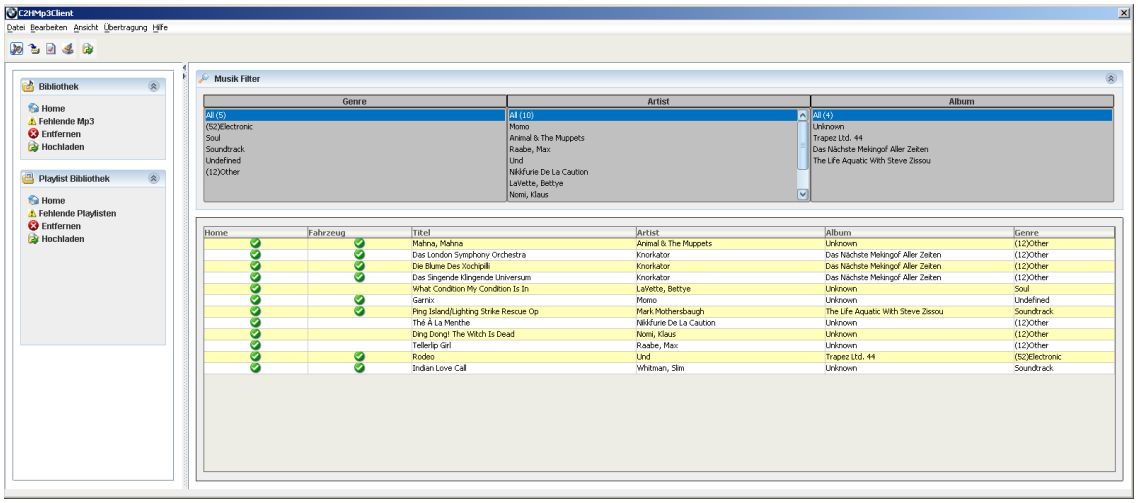


Abbildung 3.5: Eine Anwendung zur Verwaltung von Musikdateien

Während diese Anwendungen auf den ersten Blick nicht kontextsensitiv sind, so arbeitet die Statistik-Anwendung doch auf denselben Kontextdaten wie aktive Sicherheitsanwendungen. Auch in diesem Falle werden während der Fahrt anfallende Kontextdaten aufgezeichnet und gespeichert. Im Speziellen sind dies:

- Geschwindigkeit
- Kilometerstand
- Reichweite
- Nachtankmenge (zur Feststellung von Tankstops)
- Einspritzmenge (zur Berechnung des Verbrauchs)
- GPS-Koordinaten (eine Zuordnung der den Koordinaten entsprechenden Straße kann vom Navigationsgerät im Fahrzeug vorgenommen werden)
- Zündschlüsselstatus (zur Ermittlung ob ein Tankstop stattgefunden hat)

3.4 Markteinführungsszenarien

Die in 3.2 beschriebenen aktiven Sicherheitsanwendungen Anwendungen unterliegen sogenannten Netzwerkeffekten (in geringerem Maße trifft dies auch für Deployment-Anwen-

dungen zu), d. h. ähnlich wie beim Telefon erhöht sich ihre Nutzbarkeit mit dem Grad der Netzwerkdurchdringung. Tatsächlich sind die meisten dieser Anwendung erst dann funktionstüchtig, wenn genügend potentielle Kommunikationspartner im Netzwerk vorhanden sind, so dass ein sinnvoller Informationsaustausch stattfinden kann. Diese Durchdringung kann auf zwei Arten erzielt werden: einerseits durch die Erhöhung des Anteils der mit dem Kommunikationssystem ausgestatteten Fahrzeuge, andererseits durch den Einsatz von vorinstallierter Infrastruktur. Letzterer Ansatz eignet sich aber nur in bedingtem Umfang für aktive Sicherheitsanwendungen, da in diesem Fall Bewegungsdaten vorliegen müssen, die natürlich nur in anderen Fahrzeugen selbst erhoben und übermittelt werden können. Abbildung 3.6 (Quelle: [MM04]) setzt verschiedene VANET-Anwendungen in Beziehung zum benötigten Netzwerkausbau (y-Achse) und zur Marktdurchdringung (x-Achse).

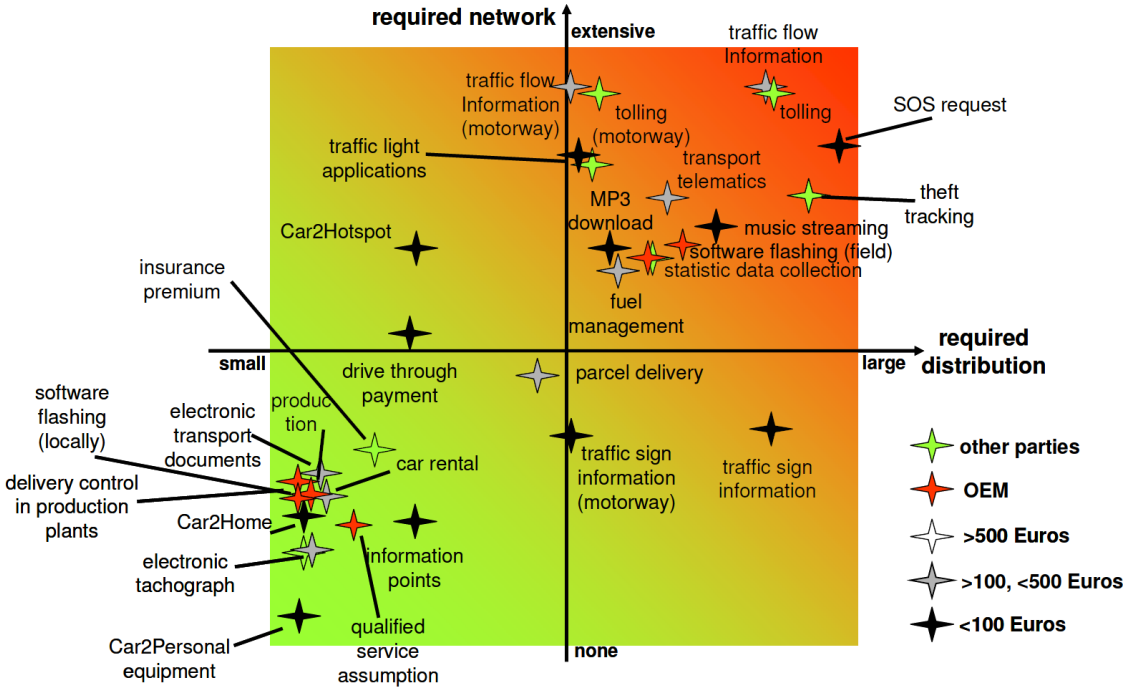


Abbildung 3.6: VANET-Anwendungen und notwendige Marktdurchdringung, Quelle: [MM04]

Dabei fällt auf, dass sich Infotainmentanwendungen eher im linken unteren Bereich befinden und Anwendungen mit Bezug zur Fahraufgabe im oberen rechten Bereich, also eine recht hohe Marktdurchdringung und einen großen Netzausbau benötigen. Selbst wenn in Deutschland alle Neuwägen mit einem solchen System ausgestattet werden würden, würde das Erreichen einer Marktdurchdringung von ca. 10% 1,5 Jahre dauern, wie Abbildung 3.7 (Quelle: [MMP+05, MM04]) zeigt.

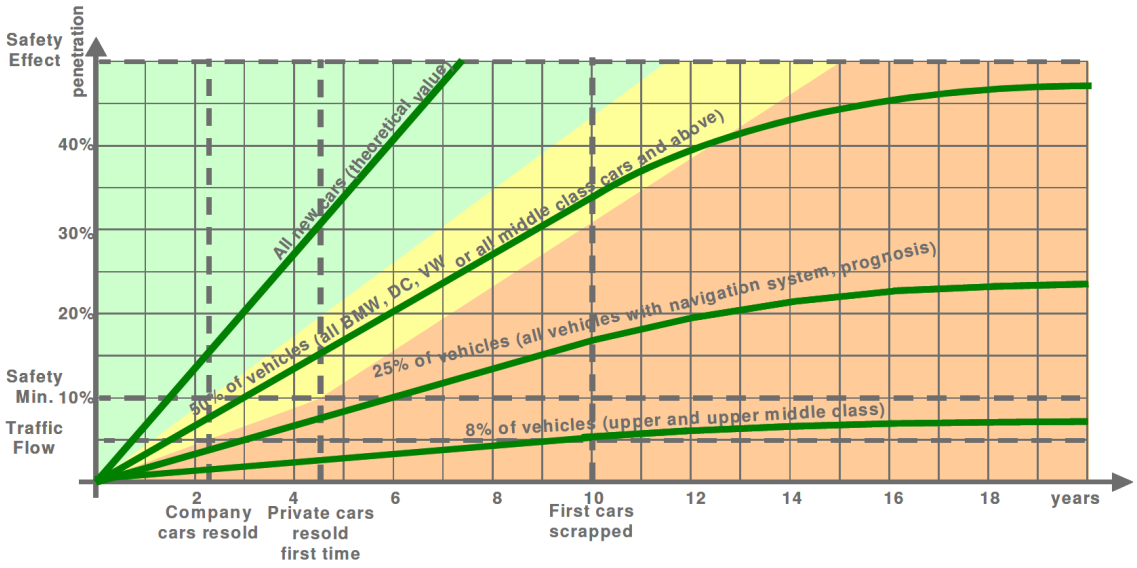


Abbildung 3.7: Verlauf der Marktdurchdringung in unterschiedlichen Szenarien, Quelle: [MMP+05, MM04]

Ein Szenario, bei dem alle Neuwägen ab Werk mit einem VANET-Kommunikationssystem ausgestattet werden ist aber aus den folgenden Gründen unwahrscheinlich:

- Kunden sind nicht bereit für ein System zu bezahlen, das in den ersten Jahren ohne eigentliche Funktion sein wird.
- Die Fahrzeughersteller sind nicht in der Lage, das System kostenlos anzubieten.
- Eine regulative Order durch den Gesetzgeber, die den Einbau verpflichtend macht, ist bei neuartigen Systemen unwahrscheinlich.

Daher schlägt [MMP+05, MM04] folgende Lösungsmöglichkeiten vor:

- Potentiellen Kunden wird mit Anwendungen, die auch schon ohne Netzwerkausbau und -durchdringung volle Funktionalität bieten, ein Anreiz geboten, das System zu kaufen. Solche Anwendung befinden sich in Abbildung 3.6 in der linken unteren Ecke. Weil sie die Einführung von VANET-Systemen in den Markt unterstützen, heißen sie *Deployment-Anwendungen* (siehe auch Abschnitt 3.3).
- Die Systeme werden schon bei Der Entwicklung auf einen nachträglichen Einbau (*After market solution*) ausgelegt. So können interessierte Kunden das System erst dann kaufen, wenn der Netzausbau z. B. durch den Aufbau von Infrastrukturkomponenten vorangeschritten ist.

- Das System wird zu einem niedrigen Preis und mit beschränkter Funktionalität von Anfang an eingebaut. Erst zu einem späteren Zeitpunkt wird das System durch den Zukauf von zusätzlichen Anwendungen auf seinen volle Funktionalität erweitert.

3.5 Resultierende Anforderungen

Aus den in Abschnitt 2.3 genannten technischen Eigenschaften, den Anwendungsbeschreibungen und den eben aufgeführten Besonderheiten bei der Markteinführung von VANET-Systemen ergeben sich nun folgende Anforderungen an ein Kontextmodell:

Ausdrucksmächtigkeit Die Analysen der Anwendungen haben gezeigt, dass der zu modellierende Kontext in VANETs recht umfangreich ist und komplexe Sachverhalte und Konzepte beinhaltet. Erschwerend kommt dazu, dass dasselbe Kontextmodell für sehr verschiedene Einsatzzwecke (Strukturierung eines Wissensraumes, automatisches Reasoning, Konsistenzprüfungen o. ä.) eingesetzt werden soll, teilweise mit diametral auseinander strebenden Anforderungen (siehe z. B. Latenzanforderungen und zeitlicher Horizont bei der Unterstützung der Fahraufgabe, Abschnitt 3.1). Das Kontextmodell sollte daher eine möglichst große Ausdrucksstärke bei gleichzeitig guter Handhabbarkeit bieten, um auch vielschichtige Zusammenhänge abbilden zu können.

Umfassendes Basismodell Die meisten Anwendungen arbeiten auf derselben Datengrundlage. Um nicht für unterschiedliche Anwendungen immer wieder die gleiche Modellierungsarbeit leisten zu müssen, sollte zumindest für Sensordaten ein generisches Modell erstellt werden, das alle Anwendungen gemeinsam verwenden. Das erleichtert die Kooperation bei der Verwendung von unterschiedlichen Sensoren, die dieselbe semantische Information liefern, aber anders repräsentieren. Bei genauer Betrachtung des Einsatzgebietes (verschiedene Fahrzeugmodelle von unterschiedlichen Herstellern; teilweise mit Sensoren, die von Drittherstellern zugeliefert werden) erschient dies auch sinnvoll. Gleichzeitig wird dadurch der Wissensraum für alle semantisch verbindlich strukturiert. Das Modell sollte dabei die folgenden Grundkonzepte umfassen:

- **Ortsbezug** Ein wichtiges Kontextdatum stellt der Ortsbezug dar, dabei verwenden die Anwendungen teilweise eine unterschiedliche Definition des Begriffes „Ort“: während es manchmal ausreichend ist, einen Ort durch zwei Geokoordinaten zu beschreiben, müssen andere Anwendungen Zugriff auf semantisch angereicherte Ortsinformationen (wie z. B. Städtenamen, Länder o. ä.) haben.
- **Zeitbezug** Ebenso wichtig wie der Ortsbezug ist die Zeitbezogenheit. Dabei muss das Modell mit unterschiedlichen Dimensionen des Zeitbezuges umgehen können: während sich einige Daten wie z. B. Geschwindigkeit sehr schnell verändern sind

andere Daten wie die Ausdehnungen von Fahrzeugen von konstanter Natur. Das Modell muss in der Lage sein, Kontextdaten eine gewisse Aktualität zuordnen zu können, aber auch eine Gültigkeitsdauer für kontextuelle Informationen angeben zu können.

- **Qualität von Kontextinformationen** Da Kontextinformationen von unterschiedlichen Sensoren in Fahrzeugen unterschiedlicher Hersteller geliefert werden, kann nicht davon ausgegangen werden, dass alle Informationen mit derselben Qualität vorliegen. Genauso kann fehlerhaftes Arbeiten von Sensoren zu einer Verschlechterung der Kontextqualität führen. Das Modell muss daher eine Angabe der Kontextqualität beinhalten.

Modellierung von Unsicherheit und Unvollständigkeit Genauso wenig wie von einer konstanten Kontextqualität ausgegangen werden kann, kann auch nicht angenommen werden, dass zu jedem Zeitpunkt alle Kontextinformationen vorhanden oder abfragbar sind. Dies rührt nicht zuletzt daher, dass sich die Fahrzeughersteller auch über die Ausstattungsmerkmale ihrer Fahrzeuge diversifizieren. Gerade beim automatischen Schließen muss das Modell auch imstande sein, mit unsicherer und unvollständiger Information umgehen zu können.

Höherwertiger Kontext Das Modell sollte mehrere Schichten an Kontext bieten, damit Anwendungen Zugriff auf Kontextinformationen unterschiedlicher Abstraktionsgrade haben. Dabei muss das Modell einen ausgewogenen Mittelweg zwischen dem Zugriff auf unverarbeitete Rohdaten als auch auf veredelten höherwertigen Kontext bieten. Dabei sollte ein Zugriff auf alle Schichten des Modells möglich sein. Das Modell sollte Anwendungen außerdem Schnittstellen bieten, eigene anwendungsspezifische Regeln für das Ableiten neuen höherwertigen Kontexts einzubringen.

offenes Modell, zeitliche Evolution des Modells, Erweiterbarkeit Durch technische Weiterentwicklungen oder wissenschaftlichen Fortschritt können sich die Charakteristika der Kontextermittlung oder die Struktur des Wissensraumes verändern. Damit Anwendungen, die auf der Basis von älteren Modellen arbeiten ihre Funktionalität erhalten können, sollte das Modell einen Versionsmechanismus bieten, um (In-)Kompatibilität zwischen verschiedenen Entwicklungsstufen eines Modells feststellen zu können. Auch die Markteinführungsszenarien und die langen Entwicklungszyklen in der Automobilindustrie, die u. U. über der typischen Entwicklungsdauer für VANET-Anwendungen liegen können, erfordern dies. Deshalb muss die Erweiterbarkeit des Modells auf struktureller Ebene gewährleistet sein. Gerade weil es die Markteinführungsszenarien (siehe Kapitel 3.4) erfordern, auch eine Nachrüstlösung anzubieten, sollte es möglich sein, das Modell zu jedem Zeitpunkt erweitern zu können, z. B. dann, wenn ein Fahrzeug nachträglich mit einem neuen Sensor,

der neue Möglichkeiten bietet, ausgestattet wird. Eine Erweiterung kann auch dann nötig werden, wenn eine neue Anwendung installiert wird, die neues domänenspezifisches Wissen erfordert.

Einfache Zugänglichkeit des im Modell gespeicherten Wissens Da das Wissen über Kontext bzgl. des ISO/OSI-Netzwerkmodells nicht nur auf Anwendungsebene benötigt wird (siehe Abschnitt 2.3), sondern auch auf den Schichten darunter, um eine möglichst gute Netzwerkperformance bieten können, muss das im Modell gespeicherte Wissen auf allen Ebenen zugänglich sein. Das Modell sollte daher fest definierte Schnittstellen für den Zugriff auf die Wissensbasis bieten.

Verteiltheit der Modellierung Da VANETs ihrer Natur nach verteilt sind, kann nicht davon ausgegangen werden, dass in einem Netzwerkknoten zu einem bestimmten Zeitpunkt ein vollständiges Modell vorliegt. Dies kann auch darin begründet sein, dass nur ein bestimmter Teil aller möglichen Sensoren und Anwendungen aufgrund von Kundenwünschen in einem Fahrzeug integriert wird. Es muss daher möglich sein, ein Modell auch nur teilweise validieren zu können, genauso müssen die Anwendungen imstande sein, mit nur teilweise vorliegender Information umzugehen. Diese Anforderung wird zusätzlich durch die eben angesprochene zeitliche Evolution der Modelle motiviert.

Formales Modell, Semantische Klarheit Das Modell muss einen hohen Grad an Formalität aufweisen, um maschinenverarbeitbar zu sein. Die Verteiltheit der Modelle und die Tatsache, dass einige Anwendungen über mehrere Fahrzeuge hinweg miteinander kooperieren müssen, erfordert es, dass die zugrunde gelegten Modelle semantisch eindeutig sind, was durch eine Formalisierung der Semantik erreicht werden kann. Anderenfalls würden Modelle auf unterschiedlichen Wissensgrundlagen operieren, was zu einer fehlerhaften Anwendungsverhalten führen kann. Je klarer das gemeinsame Verständnis der Anwendungsdomäne ist, desto höher ist der Grad der Interoperabilität, der erreicht werden kann.

Konsistenz von Modellen und Sensordaten, partielle Konsistenzüberprüfung Um Fehler in Modellierung ausschließen zu können, muss das Modell validierbar sein. Dies ist insbesondere notwendig, da Modelle zu einem späteren Zeitpunkt erweitert werden können, was zu Inkonsistenzen führen kann. Auch die Verteiltheit der Modellierung und die Tatsache, dass die Strukturierung der Anwendungsdomänen und des Wissensraumes nicht einer einzigen zentralen Instanz überlassen ist, macht diese Anforderung notwendig.

Echtzeit-Anforderungen, Handhabbarkeit Trotz einer hohen Ausdrucksmächtigkeit muss die Modellierung des Kontextes noch handhabbar sein. Ohne einfache Möglichkeiten zur Erstellung von Modellen können neue Anwendungen nur schwierig erstellt werden.

Außerdem misst sich die Qualität der Modellierung und ihre semantische Verbindlichkeit auch an ihrer Verbreitung. Diese Verbreitung wird nicht eintreten, wenn die Modelle zu komplex werden, nicht mehr les- und verstehbar oder überprüfbar sind. Weiterhin müssen die Modelle auch leicht und hinreichend schnell einsetzbar sein. Gerade Fahrerassistenzanwendungen haben oft Echtzeitanforderungen: die Erfüllbarkeit eines modellierten Aspekts muss dann noch in ausreichend schneller Zeit geschehen können.

Integrationsfähigkeit Die Modelle sollen sich einfach in schon bestehende Umgebungen integrieren lassen, um eine zügige Verbreitung zu erreichen.

4 Stand der Forschung

4.1 Alternative Modellierungstechniken

Ontologische Kontextmodelle stehen am (bisherigen) Endpunkt einer längeren Entwicklung von Kontextmodellen, die Hand in Hand ging mit der Entwicklung des Kontextbegriffs. In den folgenden Abschnitten soll auf alternative Möglichkeiten zur Kontextmodellierung eingegangen werden und schließlich auf schon vorhandene ontologische Kontextmodelle. Eine guten Überblick über verschiedene Techniken zur Kontextmodellierung geben [SLP04, BDR07, OA06].

Schlüssel-Wert-Paare Zur Kontextmodellierung wurden schon in [SAW94] Schlüssel-Wert-Paare verwendet, um kontextbewussten Anwendungen über Umgebungsvariablen Zugriff auf entsprechende Daten zu geben. Kontextuelle Daten wurden dabei in der Form `distance=20ft`, `name=claudia`, `room=35-2108` gespeichert. Aufgrund seiner Einfachheit wird der Ansatz oft verwendet, z. B. auch in [WE07] oder [SMLP01]. Im letzteren Ansatz werden Schlüssel-Wert-Paare dazu verwendet, die Rahmenbedingungen für kontextbasierte Dienstausführungen zu formulieren. Eine Einhaltung der Rahmenbedingungen kann dann sehr einfach durch ein einfaches Matching mit den tatsächlich herrschenden Kontextbedingungen durchgeführt werden. Schlüssel-Wert-Paare lassen sich zwar einfach erstellen und verwalten, aber durch ihre nicht festgelegte Semantik sind sie nicht allgemein von Maschinen interpretierbar und auch nur schwer erweiterbar, selbst durch menschliches Eingreifen. Eine automatische Interpretierbarkeit ist nur dann gegeben, wenn die Interpretationslogik durch den Ersteller in die Anwendung eingebaut wurde. Auch führt die beschränkte Ausdrucksmächtigkeit dazu, dass Informationsräume mit Schlüssel-Wert-Paaren nicht strukturierbar sind, nur um noch einen weiteren Nachteil zu nennen. Schlüssel-Wert-Paare liegen keinem Schema zugrunde, was die Modellierung sehr anfällig für Fehler macht.

Markup-Sprachen Markup-Sprachen wie SGML und XML werden verwendet, um in Dokumenten Inhalte von der Struktur zu trennen und damit die Präsentation eines Dokumentes von seinem Inhalt oder dem Rechner, auf dem es angezeigt wird zu trennen. Kontext-Profile können dann auf der Basis dieser Markup-Sprachen serialisiert werden. Ein Beispiel für eine solche Profil-Sprache ist *CC/PP* (Composite Capabilities / Preferences Profile) [KRW+04, Kis06], das verwendet wird, um die Struktur für Geräteprofile und An-

forderungen an Vokabulare zur Beschreibung von Geräten zu definieren. Diese Definition wird auch Auslieferungskontext eines Gerätes genannt und kann verwendet werden, um die Darstellung von Inhalten an ein Gerät anzupassen. CC/PP basiert auf RDF und wird verwendet, um mit der Festlegung eines Vokabulars auch dessen Semantik festzuschreiben. Ein auf der Basis von CC/PP entworfenes Vokabular ist *UAProf*, das User Agent Profile, das z. B,. ein Vokabular zur Beschreibung von WAP-Mobiltelefonen festlegt. Allerdings sind Profildaten von eher statischer Natur und daher ist dieser Ansatz weniger geeignet, um dynamische Daten in VANETs zu beschreiben.

Der *CSCP*-Ansatz (Comprehensive Structured Context Profiles) [HBS02] versucht diesen Nachteil zu umgehen und definiert ein ein einheitliches Repräsentationsformat für Kontextdaten in mobilen Umgebungen. Wie CC/PP basiert CSCP auf RDF und verfügt somit über eine gewisse Ausdrucksstärke. Mit der Hilfe von CSCP ist es möglich, Kontextprofile zu strukturieren und sie in einem austauschbaren Format niederzuschreiben, wie in Listing 4.1 dargestellt. CSCP-Profile sind zerlegbar und erweiterbar und im Gegensatz zu CC/PP nicht an eine vorgegebene Struktur gebunden. Weitere Vorteile bestehen in der Tatsache, dass sie nicht auf eine einheitlich Namensgebung für modellierte Konzepte bestehen, d. h. dasselbe Konzept darf, wenn es z. B. von unterschiedlichen Personen modelliert wurde, auch verschiedene Namen tragen – ein Vorteil den es mit OWL gemein hat. Dennoch ist das Einsatzgebiet von CSCP durch die Einschränkung auf die Beschreibung von Hard- und Softwareeigenschaften begrenzt.

```
<?xml version="1.0"?>
<rdf:RDF
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:cscp = "http://example.org/CSCPProfileSyntax#"
  xmlns = "http://example.org/SessionProfileSyntax#"
  xmlns:dev = "http://example.org/DeviceProfileSyntax#"
  xmlns:net = "http://example.org/NetworkProfileSyntax#">
  <SessionProfile rdf:ID="Session">
    <cscp:defaults rdf:resource=
      "http://localSessionContext/CSCPProfile/previous#Session"/>
    <device>
      <dev:DeviceProfile>
        <dev:hardware>
          <dev:Hardware>
            <dev:memory>9216</dev:memory>
          </dev:Hardware>
        </dev:hardware>
      </dev:DeviceProfile>
    </device>
```

```
  </SessionProfile>
</rdf:RDF>
```

Listing 4.1: Ein in CSCP formuliertes Profil, Quelle: [HBS02]

Grafische Modelle *ContextUML* [SB05] ist eine auf UML (Unified Modeling Language) basierendes grafische Sprache für die modellgetriebene Entwicklung von kontextbewussten Webservices. ContextUML bietet dabei ein Metamodell, um eine abstrakte Syntax für Kontextmodellierungen zu bieten und eine konkrete Syntax (Notation) als Sprachrepräsentation. Abbildung 4.1 zeigt das Metamodell.

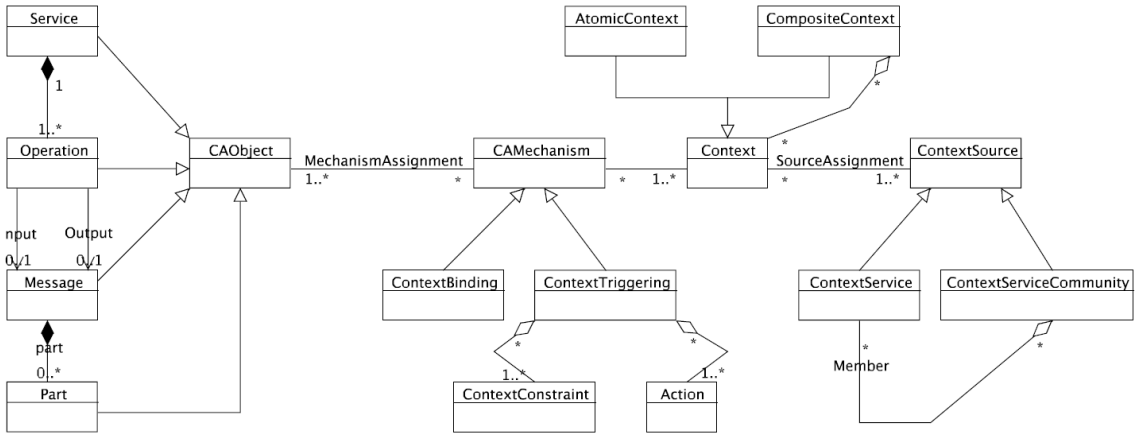


Abbildung 4.1: Das ContextUML Metamodell, Quelle: [SB05]

ContextUML unterscheidet dabei zwischen atomaren Kontexten, die direkt von Kontextquellen wie Sensoren geliefert werden und zusammengesetzten Kontexten, die nicht notwendigerweise direkt von Sensoren abgelesen werden können, sondern sich aus mehreren atomaren Kontexten zusammensetzen. Durch die Modellierung kann eine Trennung von Kontextmodell und den eigentlichen Diensten erreicht werden und damit ein gewisses Maß an Abstraktion. Allerdings ist die Erweiterbarkeit des Modells sehr eingeschränkt, da keine offenen Schnittstellen zur Veränderung des Metamodells vorhanden sind und die Modellierungsmöglichkeiten für Beziehungen zwischen Komponenten in UML beschränkt sind. Die Formalität der Sprache ist zwar durch die UML-Vorgaben vorhanden, aber schlecht ausnutzbar, da nicht maschinell weiterverarbeitbar.

[HIR03] schlägt eine Erweiterung des Object-Role Modeling [Hal01] auf der Basis eines informellen Kontextmodells [HIR02] vor. Das Modell erlaubt die Abbildung von komplexen Beziehungen zwischen Kontexteinheiten wie z. B. der Tatsache, dass Kontext direkt von Sensoren abgelesen, abgeleitet, aus einem Profil ermittelt werden kann oder grund-

sätzlich statisch ist. Der Ansatz erlaubt auch die Modellierung von Kontextqualität und zeitlichen Zusammenhängen, um Kontexthistorien aufzeichnen zu können. Die aus ORM übernommenen Begriffe beziehen sich auf grundlegende Konzepte: als Basis dienen *Fakten* und *Rollen*. Modellierung in ORM bedeutet also die Identifikation von Fakten, Entitäten und ihrer Rollen. Beziehungen zwischen Fakten müssen entsprechend der oben genannten Kategorien typisiert werden. Interessant bei diesem Modell ist die Möglichkeit, Abhängigkeitsbeziehungen zwischen Fakten modellieren zu können, d. h. wenn ein Faktum sich ändert, ändern sich auch alle abhängigen Fakten. Abbildung 4.2 gibt einen beispielhaften Einblick in die Notation.

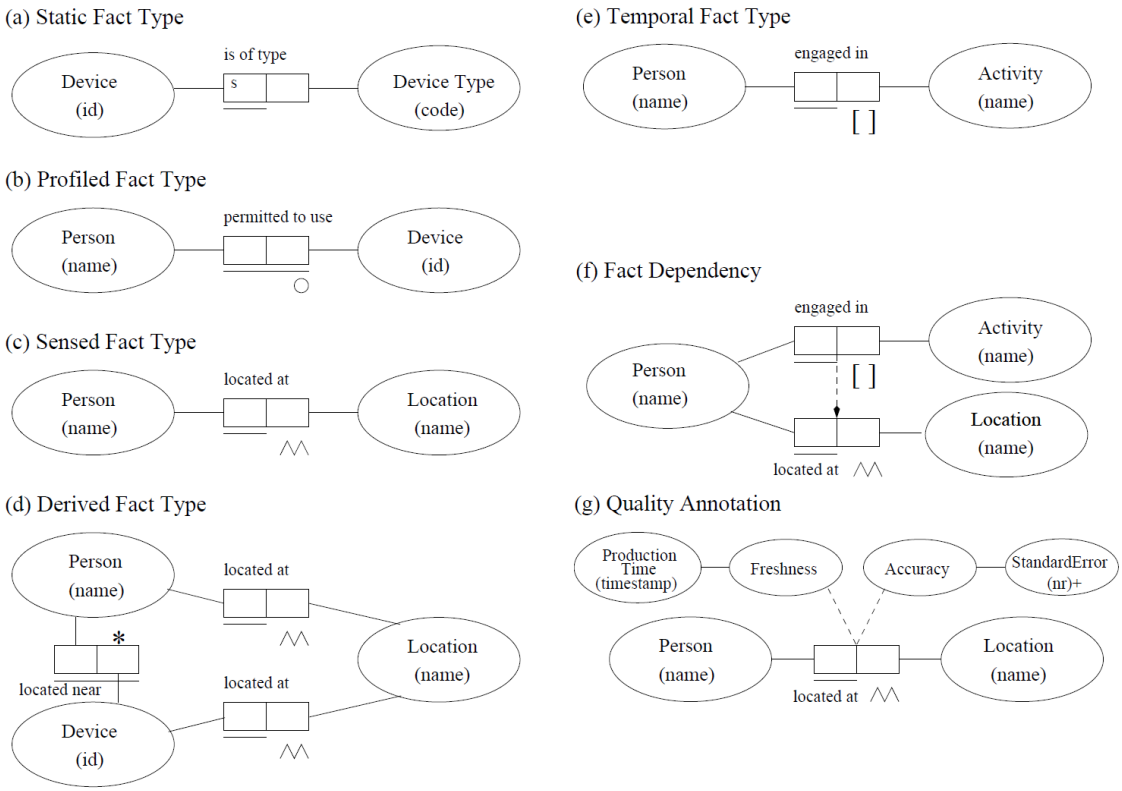


Abbildung 4.2: Ein grafisches Kontextmodell nach [HIR03]

Auch gelten die selben Einschränkungen wie bei den anderen grafischen Modellen, d. h. die Modelle sind i. A. nicht maschinell weiterverarbeitbar, sondern eher für die Interpretation und Kommunikation mit Menschen gedacht. Ein Vorteil des Modells ist der Herkunft aus dem Bereich der Datenbank-Modellierung geschuldet: da ORM-Modelle in Entity-Relationship-Modelle übersetzbar sind, kann derart modellierter Kontext leicht auf Datenbanktabellen abgebildet werden.

Objektorientierte Modellierung Die objektorientierte Modellierung bietet mit Werkzeugen zur Abstraktion, Kapselung und Vererbung von Konzepten schon grundlegende Techniken, um auch Kontextinformationen strukturieren, beschreiben und damit wiederverwendbar machen zu können. Die Informationen werden dabei in Objekten gekapselt und der Zugriff auf sie erfolgt über fest definierte Schnittstellen. Ein Vertreter der objektorientierten Kontextmodellierung sind *cues* [SBG99], die die physikalischen Eigenschaften eines Sensors kapseln und somit eine Abstraktion darstellen. Dabei hängt ein cue von genau einem Sensor ab, aber mehrere cues dürfen vom selben Sensor abhängig sein. Kontext selbst ist eine dann weitere Abstraktion von mehreren cues, die miteinander verknüpft wurden. Der Mangel an Formalisierung macht das Modell schwer zu erweitern und verhindert eine maschinelle Verarbeitung des Kontexts.

Das Hydrogen-Projekt [HSP+03] unterscheidet Kontext in eine lokalen und entfernten Kontext, die einheitlich als Objekte modelliert sind. Eine Superklasse `ContextObject` wird dann in weitere Kontexttypen, z. B. räumlichen oder Gerätekontext unterteilt, was zu einer Strukturierung von Kontextinformationen führt. Ein Adapterkonzept verdeckt dabei technische Eigenschaften von Sensoren und dient der Kontextermittlung. Alle Objekte müssen Methoden (`toXML` und `fromXML`) implementieren, um Daten in und aus einem XML-Datenstrom zu konvertieren. Diese Schnittstelle stellt den Zugriffspunkt auf die Kontextinformationen dar. Abbildung 4.3 zeigt das Kontextmodell. Die mangelnde Fähigkeit, Beziehungstypen zwischen Objekten zu modellieren, beschränkt die Ausdrucksmächtigkeit von objektorientierten Modellen.

Logik-basierte Modelle Logikbasierte Modelle verfügen üblicherweise über einen hohen Formalisierungsgrad und verwenden Konzepte wie Fakten, Prädikate und Regeln um Kontext zu repräsentieren. Daten werden verwaltet, indem dem System neue Fakten hinzugefügt werden. Automatische Inferenz wird dazu verwendet, um aus dem vorhandenen Daten basierend auf den vorhandenen Regeln neues Wissen abzuleiten. Einer der ersten Ansätze wird in [MB94] beschrieben: Grundlage sind zwei Relationen

- $c' : ist(c, p)$, falls eine Aussage (proposition) p im Kontext c wahr ist. Diese Zusicherung gilt dabei nur in einem äußeren Kontext c'.
- $value(c, e)$, die den Wert eines Terms e im Kontext c festlegt

Außerdem bietet der Ansatz sogenannte *lifting formulas*, die es ermöglichen, Aussagen und Terme von untergeordneten Kontexten in allgemeinere zu übertragen. Untergeordnete Kontexte sind hinsichtlich Zeit, Ort und Terminologie spezialisiert. Diese Art der Modellierung sollte es künstlichen Intelligenzen ermöglichen, die menschlichen Art und Wiese zur Repräsentation von Fakten nachzuahmen und den Prozess des menschlichen Schlussfolgerns nachzuahmen. Die oben beschriebenen Kontextaussagen konnten dabei in Form

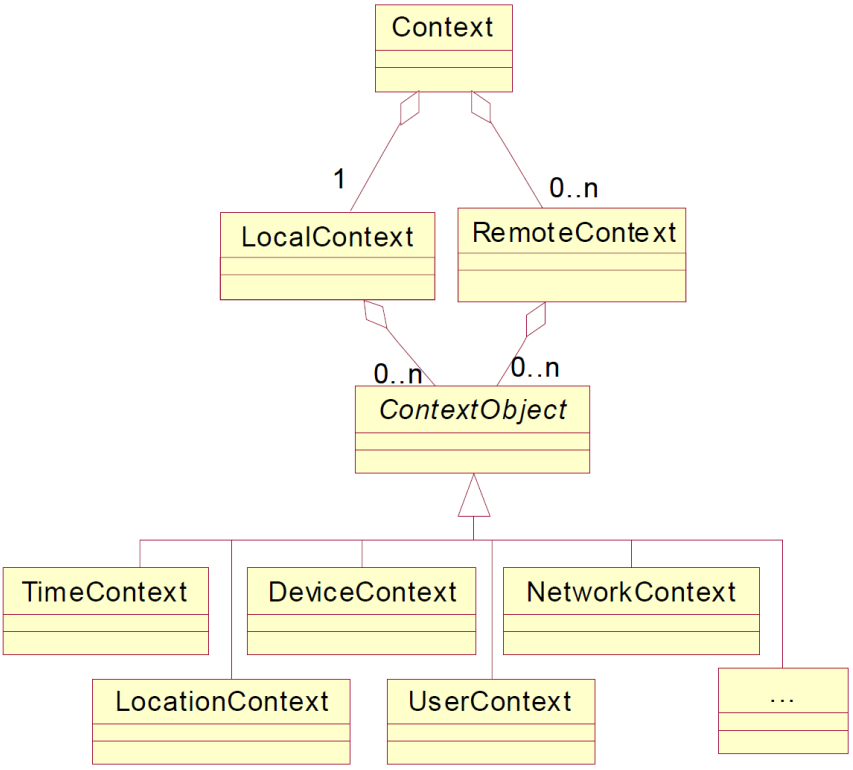


Abbildung 4.3: Objektorientiertes Kontextmodell des Hydrogen-Projekts, Quelle: [HSP+03]

von Beziehungen des Typs „innerer Kontext“ ↔ „äußerer Kontext“ miteinander verknüpft werden. In [AS97] wird der Ansatz weiter in einem situationstheoretischen Rahmenwerk formalisiert. Es ermöglicht die dynamische Erweiterung oder Verkleinerung von Kontexten durch Hinzufügungen oder Löschungen von Annahmen oder Regeln über den Kontext.

Ein informelles Modell, das an prädikatenlogische Ausdrücke erster Stufe erinnert, wird in [GS01] vorgeschlagen: es enthält Kategorien, um räumliche und zeitliche Bezüge eines Kontexts zu modellieren und diese mit einer Identität zu versehen. Außerdem können Informationsqualität, Auflösung, Genauigkeit, Zeitstabilität, Frequenz und Zeitnähe einer Sensorquelle für Kontextinformationen modelliert werden. Ein Modell zur Transformation von rohen Kontextdaten in andere (auch höhere) Formen der Repräsentation wird skizziert. Außerdem erlaubt das Modell eine Einschränkung der Kontextdomäne: die Menge der Subjekte, auf die sich ein Kontext bezieht, kann benannt werden.

Zumindest Ortsinformationen werden in [BBH97] in einer Art und Weise modelliert, die an prädikatenlogische Ausdrücke erinnert. Diese Ausdrücke werden aus einer Schnittstellenbeschreibungssprache generiert, die Schablonen für das Modell vorgibt. Damit kann

das Modell dann um andere Kontextdatentypen erweitert werden. Zur einfacheren Verwendung enthält der Ansatz auch noch eine Möglichkeit, das Modell erst in einer Entity-Relationship-Form niederzuschreiben und dann zu übersetzen. Die eigentliche Umsetzung des Systems erfolgte dann in Prolog.

4.2 Ontologische Kontextmodelle

Von logikbasierten Modellen ist dann der Weg nicht mehr weit zu ontologischen Modellen, die – wie in Kapitel 2.4.2 erläutert – auf Beschreibungslogiken basieren. Im Folgenden sollen Ansätze beleuchtet werden, die Kontext mit Hilfe von Ontologien modellieren, aber teilweise auf andere Einsatzzwecke zugeschnitten sind.

CoNoN Die Context Ontology CoNoN (vgl. [WZGP04, GWPZ04]) beschreibt eine obere Ontologie, die grundlegende Entitäten in mobilen Rechnerumgebungen, nämlich Ortsinformationen, Personen, Aktivitäten und Recheneinheiten wie z. B. Dienste, Geräte, Agenten, Netze u. ä.. Diese obere Ontologie ist eine gemeinsame Dachkonstruktion für mehrere spezialisierte Domänenontologien in den weitere spezifischere Konzepte modelliert werden. Die Ontologien werden in OWL DL serialisiert. Kontext wird unterschieden nach der Art seiner Ermittlung: Kontext kann von Sensoren abgelesen (sensed), definiert (defined), aggregiert oder abgeleitet sein. Außerdem enthält das Modell Konzepte zur Modellierung von Kontextqualitätskriterien wie z. B. Genauigkeit, Auflösung, Neuheit und Zuverlässigkeit. Abbildung 4.4 zeigt einen Ausschnitt der oberen CoNoN-Ontologie.

CoBrA Im Rahmen einer Context Broker Architecture (CoBrA) werden in [CFJ04] auch Kontextontologien für einen „intelligenten Besprechungsraum“ definiert. Basiskonzepte sind Ortsinformationen, menschliche Agenten, Softwareagenten, Ortsbeschreibungen an Universitäten wie z. B. Hörsäle, Büros, Gebäude und Aktivitäten. Zentrales Merkmal ist die Tatsache, dass es ein doppeltes Konzept für Ortsinformationen gibt: zum einen eine generelle `Place`-Klasse, die einen Ort als Koordinatenpaar `(longitude, latitude)` modelliert und diesen dann mit einen Namen versieht (geographischer Ort). Andererseits die Semantische Ortsinformation im Universitätsumfeld: Campus, Gebäude, Raum, Gang, Treppenhaus, Toilette, Parkplatz. Ein Beziehungskonzept verbindet diese beiden Klassen und Subklassen. Orte selbst können wieder unterteilt werden in kleinere Einheiten: so enthält z. B. ein Gebäude mehrere Flure, Treppenhäuser und Räume. Abbildung 4.5 gibt einen Überblick über die in CoBrA modellierten Klassen und Eigenschaften.

COMANTO Die Context Management Ontology [RSK+06] ist als übergreifende Ontologie zur Auflösung von Konflikten zwischen mehreren spezialisierten Domänen konzipiert

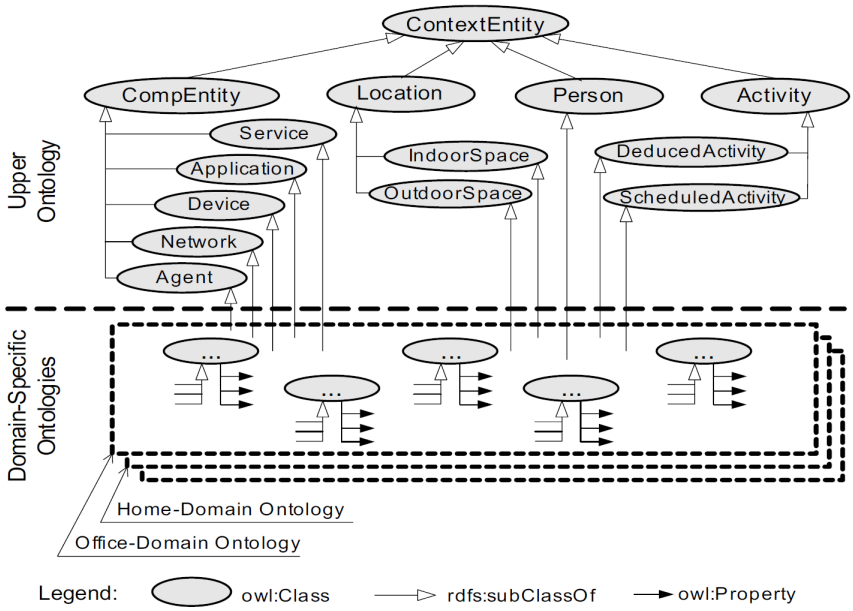


Abbildung 4.4: Ein Ausschnitt der oberen CoNoN-Ontologie, Quelle: [WZGP04]

und soll Wissen über Kontext synchronisieren. Wurzelkonzept ist die `Semantic Context Entity`, die wiederum in mehrere Subklassen unterteilt wird:

- `Person`
- `Legal Entity`
- `Actitvity`
- `Time`
- `Network`
- `Preferences`
- `PhysicalObject`
- `Agenda`
- `Place`
- `Sensor`
- `Service`

Abbildung 4.6 zeigt grundlegende Konzepte der COMANTO-Ontologie, spart aber Details der Ortsmodellierung aus. Orte sind sowohl über ihre geographischen Koordinaten charakterisiert als auch über über ihre symbolischen (semantischen) Bezeichner.
Damit ist die COMANTO Ontologie vielseitig einsetzbar, unabhängig von Anwendung, Anwendungsgebiet oder Situation.

CoOl Die Context Ontology Language CoOL [Str04, SLPF03] basiert im wesentlichen auf dem Aspect-Scale-Modell (ASC): Jeder Aspekt verfügt über eine oder mehrere Skalen, jede Skala über eine oder mehrere Kontextinformationen, die mit einer (physikalischen) Einheit versehen werden können. Die Konzepte werden über die Relationen `hasAspect`, `hasScale`, `constructedBy` miteinander in Beziehung gesetzt, wie Abbildung 4.7 verdeutlicht.
Ein Aspekt bündelt dabei semantisch äquivalente Skalen, d. h. Geschwindigkeit könnte z. B. ein Aspekt sein, der mit Hilfe unterschiedlicher Skalen, nämlich m/s oder km/h

CoBrA Ontology Classes		CoBrA Ontology Properties	
"Place" Related	Agents' Location Context	"Place" Related	Agent's Location Context
Place AtomicPlace CompoundPlace Campus Building AtomicPlaceInBuilding AtomicPlaceNotInBuilding Room Hallway Stairway OtherPlaceInBuilding Restroom Gender LadiesRoom MensRoom ParkingLot	ThingInBuilding SoftwareAgentInBuilding PersonInBuilding ThingNotInBuilding SoftwareAgentNotInBuilding PersonNotInBuilding	latitude longitude hasPrettyName isSpatiallySubsumedBy spatiallySubsumes accessRestricted-ToGender lotNumber	locatedIn locatedInAtomicPlace locatedInRoom locatedInRestroom locatedInParkingLot locatedinCompoundPlace locatedInBuilding locatedInCampus
"Agent" Related	Agent's Activity Context	"Agent" Related	Agent's Activity Context
Agent Person SoftwareAgent Role SpeakerRole AudienceRole IntentionalAction ActionFoundInPresntation	PresentationSchedule Event EventHappeningNow PresentationHappeningNow RoomHasPresentationHappeningNow ParticipantOfPresentation-HappeningNow SpeakerOfPresentationHappeningNow AudienceOfPresentationHappeningNow PersonFillsRoleInPresentation PersonFillsSpeakerRole PersonFillsAudienceRole	hasContactInformation hasFullName hasEmail hasHomePage hasAgentAddress fillsRole isFilledBy intendsToPerform desiresSomeone-ToAchieve	participatesIn startTime endTime Location hasEvent hasEventHappeningNow invitedSpeaker expectedAudience presentationTitle presentationAbstract presentation eventDescription eventSchedule

Abbildung 4.5: Die im Rahmen von CoBrA modellierten Klassen und Eigenschaften, Quelle: [CFJ04]

ausgedrückt werden könnte. Zur Abbildung von Entitäten einer Skala auf einer anderen, bietet das Modell `IntraOperations` an, die eine Vorschrift enthalten, um eine Skala in eine andere Skala des selben Aspekts umzurechnen. Möchte man eine Skala eines Aspekts in eine Skala eines anderen Aspekts umrechnen, dann muss eine `InterOperation` verwendet werden. Die Qualität von Kontextinformationen kann im ASC-Modell angegeben werden, indem man die Konzepte `meanError` verwendet, um die Genauigkeit einer Messung zu spezifizieren und `timestamp` für den Zeitpunkt der Messung. Zusätzlich bietet das Modell eine frei zu definierende Qualitätseigenschaft `hasQuality`, bei der der Designer ein eigenes Qualitätskriterium definieren kann, freilich dann mit unzureichend definierter Semantik. Ein derart spezifiziertes Qualitätskriterium muss selbst nur wieder vom Typ `ContextInformation` sein, d. h. es stellt selbst eine Kontextinformation dar (wie der Zeitstempel und `meanError` auch).

CoOL und ASC bieten sehr detaillierte und mächtige Modellierungswerkzeuge für eine semantisch tief liegende Kontextschicht nah an der Sensorik, da sie sich mit eher technischen Eigenschaften beschäftigen. Sie bieten somit eher einen Interpretationsleitfaden für die von

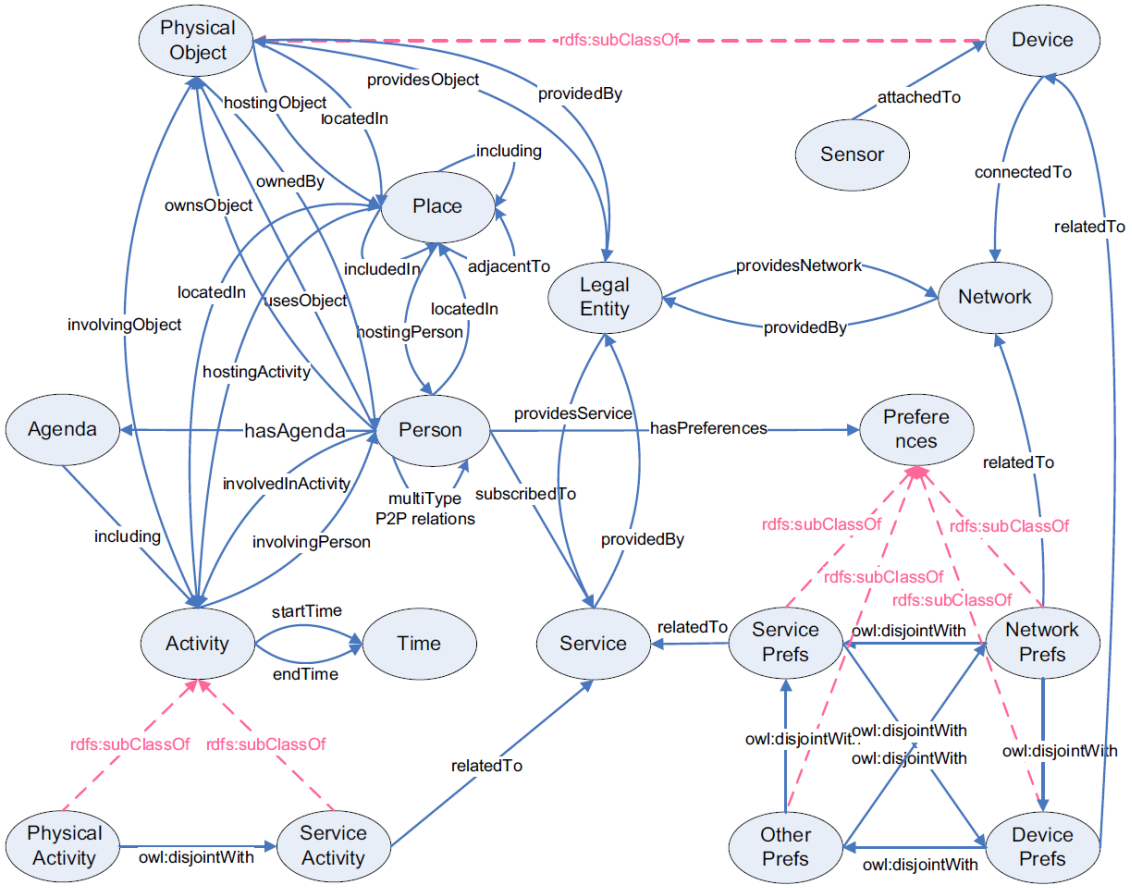


Abbildung 4.6: Die im Rahmen von CoBrA modellierten Klassen und Eigenschaften, Quelle: [RSK+06]

Sensoren gelieferten Werte, aber keine Möglichkeiten, diese zu höherwertigem Kontext zu verknüpfen.

VCM Das Vehicle Context Model [Str07] ist eine Instanz eines Context Meta Models [FHKB05] und dient zur Kontextverwaltung in Fahrzeugen, behandelt aber die unteren Aspekte wie z. B. im ASC-Modell nicht. Wesentliche Konzepte in VCM sind:

- `Beobachtung`: eine Beobachtung repräsentiert konkret ermittelte Werte von Sensorsystemen und macht Angaben hinsichtlich Ort und Zeit der Beobachtung, sowie zur Qualität, Zuverlässigkeit und Genauigkeit der Beobachtung. Es gibt keine Unterscheidung zwischen höherwertigen oder unmittelbaren Beobachtungen.
- `Aggregat`: ein Aggregat bündelt mehrere einzelne Beobachtungen, die sich auf dasselbe Ereignis beziehen. Dabei ist das Kriterium, das entscheidet, ob zwei Beobachtungen miteinander korreliert sind, ausschließlich räumlich (d. h. über ein Ausbreitungsgebiet). Im VCM können Beobachtungen nicht zeitlich aggregiert werden.

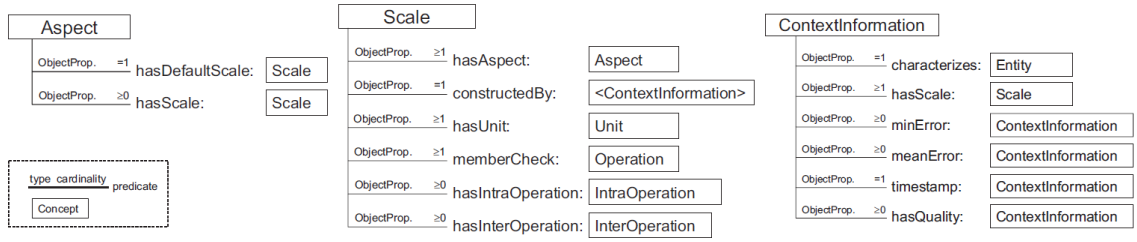


Abbildung 4.7: Das ASC-Modell, Quelle: [SLPF03]

- `Aspekt`: ein Aspekt ist eine aus strukturellen Gründen eingeführte Oberklasse zu `Beobachtung` und `Aggregat`. Sie wird dazu verwendet, um weitere Eigenschaften wie Dynamik, Distanz oder Priorität eines Aspekts nicht doppelt an die Klassen `Beobachtung` und `Aggregat` binden zu müssen. Dabei können aber noch zusätzlich Hierarchien zwischen Aspekten in Form von Generalisierungen oder Spezialisierungen und Kausalitätsketten modelliert werden, d. h. ein Aspekt ist Ursache für einen anderen.

Obere Ontologien Top-level Ontologien wie z. B. SUMO[1], GUMO [HSB+05], Cyc[2] bzw. OpenCyc[3], um nur einige Beispiele zu nennen, sind große Ontologien, die auf einer sehr generischen Ebene viele Konzepte modellieren. Sie sind nicht auf spezielle Anwendungsdomänen zugeschnitten, und können aufgrund Ihrer Größe nur selten direkt eingesetzt werden. Ihr Nutzen liegt vielmehr darin, als gemeinsame Basis für viele andere Domänenontologien zu dienen, die mit ihrer Hilfe miteinander in Beziehung gesetzt werden können.

[1] `http://www.ontologyportal.org/`
[2] `http://www.cyc.com/cyc`
[3] `http://opencyc.org/`

5 Konzept für Kontextmodelle auf der Basis von Ontologien

Das KOMODE Kontextmodell besteht aus mehreren Schichten: einerseits Basisontologien zur Modellierung von grundlegenden Konzepten, die allgemeine Begrifflichkeiten in VANETs modellieren und auf die sich alle weiteren Schichten stützen (können). Darauf aufbauend eine Schicht von höherwertigem Kontext, der aus den Basisschichten abgeleitet werden kann. Zusätzlich können Anwendungen eigene Modelle in Form von anwendungsspezifischen Ontologien einbringen.

Die grundlegenden Modelle erstrecken sich auf die folgenden Bereiche:

- **Zeitkonzepte**
- **Ort**
- **Umweltwahrnehmungen**

5.1 Zeitkonzepte

Um Abläufe in Kontextdaten, Dynamik und Aggregationen von Kontextinformationen über bestimmte Zeiträume hinweg modellieren zu können, ist es notwendig grundlegende Konzepte aus der Domäne „Zeit“ zu modellieren. Die hier vorgestellten Klassen und Beziehungen basieren im Wesentlichen auf den Konzepten aus der Zeit-Ontologie des W3C [HP06]. Die Konzepte bestehen aus den folgenden Klassen:

- `Zeitinstanz` als Oberklasse für die Klassen
- `Zeitpunkt`
- `Zeitraum`

Dabei kann man einen `Zeitraum` der Länge 0, also mit identischem Startzeitpunkt und Endzeitpunkt, als `Zeitpunkt` betrachten. Gleichzeitig gilt: die Klasse `Zeitinstanz` umfasst ausschliesslich die Unterklassen `Zeitpunkt` und `Zeitraum` und keine weiteren Unterklassen. Sie stellt also die Vereinigung der Klassen `Zeitpunkt` und `Zeitraum` dar. Um

Zeitinstanzen entsprechend beschreiben zu können, hat die Klasse Zeitinstanz die folgenden Eigenschaften:

hatBeginn
hatEnde
vor
nach

Die Relationen vor und nach werden verwendet, um entsprechende Beziehungen zwischen Zeitinstanzen herzustellen. hatBeginn und hatEnde beschreiben Zeitpunkte, die den Beginn eines Zeitraumes bzw. eines Zeitpunktes markieren. Mit diesen Konzepten ist es nun möglich einen Zeitraum der Länge 0 zu definieren. Um dennoch diese „falschen" Zeiträume von echten unterscheiden zu können, wird die Klasse EchterZeitraum als Unterklasse von Zeitraum und zusätzlich disjunkt von Zeitpunkt definiert.

Die Klasse EchterZeitraum verfügt über die folgenden Relationen:

identischerZeitraum	beschreibt die Identität zweier Zeiträume	
vorZeitraum	sagt aus, dass ein Zeitraum vor dem anderen liegt, aber nicht zwangsläufig unmittelbar davor	
aneinanderZeitraum	sagt aus, dass sich ein Zeitraum unmittelbar an den anderen reiht	
ueberlappenderZeitraum	sagt aus, dass sich zwei Zeiträume überlappen	
beginntZeitraum	sagt aus, dass ein Zeitraum den Beginn eines anderen darstellt, d. h. hatBeginn ist für beide identisch, hatEnde nicht	
waehrendZeitraum	sagt aus, dass ein Zeitraum komplett in einem anderen enthalten ist	

`beendetZeitraum`	sagt aus, dass ein `Zeitraum` das Ende eines anderen darstellt, d. h. `hatEnde` ist für beide identisch, `hatBeginn` nicht	
`nachZeitraum`	sagt aus, dass ein `Zeitraum` nach dem anderen liegt, aber nicht zwangsläufig unmittelbar danach	

Alle diese Relationen nehmen wiederum einen `EchtenZeitraum` als Argument. Zusätzlich existiert die Relation `innerhalb` für `EchteZeiträume`, die einen `Zeitpunkt` als Argument nimmt, um anzugeben, dass ein `Zeitpunkt` in einen `Zeitraum` liegt.

Um die Dauer von `Zeiträumen` beschreiben zu können, wird die Klasse `Zeitdauer` eingeführt. Sie verfügt über acht Eigenschaften, die jeweils höchstens einmal vorkommen dürfen:

`jahre`
`monate`
`wochen`
`tage`
`stunden`
`minuten`
`sekunden`
`millisekunden`

Jede dieser Relationen nimmt eine Dezimalzahl als Argument. Mit Hilfe dieser Klasse ist es möglich, für die Klasse `Zeitinstanz` eine neue Relation `hatZeitdauer` zu definieren, mit deren Hilfe man die Dauer eines `Zeitraumes` oder `Zeitpunktes` bestimmen kann. Das widerspricht nicht der Tatsache, dass `Zeitpunkte` die Länge 0 haben, sondern lässt sich mit der Granularität der Information erklären: so kann man z. B. den 25.12.1977 als einen `Zeitpunkt` betrachten, der die `Zeitdauer` 1 Tag hat und gleichzeitig den `Zeitpunkt` 25.12.1977 als Beginn und Ende.

Als Subklassen von `Zeitdauer` können jetzt die Klassen `Jahr`, `Monat`, `Woche` usw. definiert werden, die einfach entsprechende Einschränkungen der Klasse `Zeitdauer` vornehmen.

Um festzulegen, zu welcher Zeit und Datum ein `Zeitpunkt` tatsächlich stattfindet, wird die Klasse `ZeitDatum` definiert. Sie gibt an, an welchem Punkt auf dem Zeitstrahl ein `Zeitpunkt` verankert werden kann. Sie hat 11 Eigenschaften:

`einheit`
`jahr`
`monat`
`woche`
`tag`
`tagDesJahres`
`tagDerWoche`
`stunde`
`minute`
`sekunde`
`millisekunde`

Das Feld `einheit` legt dabei fest, ob man sich z. B. auf einen Tag oder einen Stunde bezieht. Zu diesem Zweck wird die Klasse `Zeiteinheit` definiert, die nur aus eine Aufzählung von ihren eigenen Instanzen (`einheitMillisekunde`, `einheitSekunde`, `einheitMinute`, `einheitStunde`, `einheitTag`, `einheitWoche`, `einheitMonat`, `einheitJahr`) besteht. Die Subklassen von `Zeitdauer` sind hier nicht verwendbar, da sie ja das Konzept einer Dauer und nicht einer Einheit modellieren. Wenn man z. B. genau die Minute um 12:30 Uhr eines Tages adressieren möchte, so ist die `einheit` mit `einheitMinute`, `stunde` mit 12 und `minute` mit 30 belegt. Alle kleineren Felder (in diesem Falle `einheitMillisekunde` und `einheitSekunde`) können dann ignoriert werden.

Mit der Hilfe der Klasse `ZeitDatum` können nun `Zeitpunkte` mit Hilfe der Eigenschaft `anZeitDatum` an ein `ZeitDatum` gebunden werden.

5.2 Ortskonzepte

Grundsätzlich muss bei der Modellierung von Ortskonzepten zwischen zwei Modelltypen unterschieden werden:

- *Symbolische* Ortsbeschreibungen identifizieren die modellierten Konzepte mit einem bedeutungstragenden Bezeichner. Beispielsweise bezeichnet der Ausdruck „Stadt“ einen abgegrenzten Ort innerhalb eines „Staates“. Neben all den politischen und anderen Bedeutungen, die diese Bezeichner haben mögen, benennen sie auch einen Ort.

- *Geometrische* oder *Geographische Modelle* identifizieren einen Ort, indem sie seine Koordinaten innerhalb eines Koordinatensystems angeben. Ausgehend von einem Nullpunkt wird die Lage des Ortes oder die der Eckpunkte seines Umrisses angegeben. Auch die Konzepte von geographischen Modellen können miteinander durch Relationen in Bezug gesetzt werden, genauso wie geographische Konzepte zu symbolischen Konzepten.

Im Rahmen dieser Arbeit soll beides versucht werden: ein geographisches Modell wird benötigt, weil Ortsinformationen in Fahrzeugen (oder auch bei anderen Verkehrsteilnehmern) gewöhnlicherweise per GPS ermittelt werden und geographische Koordinaten somit als Ausgangs-Datenmaterial dienen. Ein symbolisches Modell ist reicher an semantischen Informationen und kann daher später besser von Anwendungen verwendet werden: im Grunde genommen erfolgt erst durch die Verbindung von geographischen Koordinaten mit einem semantischen Modell eine „Aufladung“ und Aufwertung der Rohinformation mit Bedeutung. Erst durch die Zuordnung erfolgt eine „Verortung“ in der durch das Modell geformten Umgebung.

5.2.1 Geographische Ortskonzepte

Geographische Ortskonzepte werden schon seit längerer Zeit erforscht und haben im Rahmen des Open Geospatial Consortium (OGC) eine Standardisierung erfahren. Die Modelle des OGC dienen als Grundlage des hier aufgestellten Modells für geometrische Objekte (vgl. [Her06]).
Basiskonzept ist die `Geometrie`, die als Oberklasse für alle weiteren Unterkonzepte dient. `Geometrien` verfügen über die folgenden Eigenschaften:

- `hatKoordinatenSystem` gibt das Koordinatensystem an, in das diese Geometrie eingebettet ist. Beispiele hierfür sind zwei- oder dreidimensionale Koordinatensysteme als kartesische oder Polarkoordinatensysteme

- `hatDimension` gibt die Dimension dieses Objekt an: dabei muss die Dimension (natürlich) kleiner oder gleich der Dimension des verwendeten Koordinatensystems sein: dreidimensionale Objekte lassen sich z. B. nicht in zweidimensionalen Koordinatensystemen referenzieren

Geometrien selbst verfügen über drei Subklassen:

- `Punkte` verfügen über `hatDimension` 0 und haben zusätzlich eine `xKoordinate` und `yKoordinate`. Sollten `Punkte` in einem mehr als zweidimensionalen Koordinatensystem verwendet werden, so können noch weitere Koordinaten hinzugefügt werden.

- `Linien` sind `Geometrien` mit `hatDimension` 1 und genau 2 `Punkten`. Diese werden über die Relationen `hatStartpunkt` und `hatEndpunkt` an die entsprechenden Klassen gebunden. Zusätzlich verfügt `Linie` noch über die Eigenschaft `hatLänge`.
- `Polygone` werden aus mindestens drei `Linien` zusammengesetzt, die einen Ring bilden müssen, d. h. Endpunkt einer Linie muss zugleich immer wieder Startpunkt einer weiteren sein. Der letzte Endpunkt ist Startpunkt der ersten Linie. Zu diesem Zweck verfügt ein `Polygon` über die Relation `hatLinie`, die in mindestens dreifacher Ausprägung vorhanden sein muss.

Mit Hilfe dieser drei Basistypen lassen sich natürlich noch weitere komplexere Typen wie z. B. Zick-Zack-Linien definieren, die auch aus mehreren Linien bestehen, aber deren letzter Endpunkt nicht Startpunkt der ersten Linie ist. Weitere komplexere Typen sind Polygone mit Löchern, die noch gesondert definiert werden müssten, aber für den vorliegenden Anwendungsfall nicht weiter von Interesse sind.

Zur Definition der Relationen, mit denen sich zwei `Geometrien` miteinander in Beziehung setzen lassen, benötigt man noch zusätzlich Hilfsdefinitionen:

- $I(a)$ bezeichnet in den folgenden Definitionen der Relationen den Innenraum einer `Geometrie`
- $A(a)$ den Außenraum und
- $dim(a)$ die Dimension einer Geometrie

Damit lassen sich die `Geometrien` über die folgenden Relationen miteinander in Beziehung setzen, wobei jede der Relationen wieder eine andere Geometrie als Argument nimmt:

`identischMitGeometrie`	zwei `Geometrien` a, b sind vollkommen deckungsgleich: $a \subseteq b \wedge b \subseteq a$
`verschiedenMitGeometrie`	zwei `Geometrien` a, b haben keinen gemeinsamen Punkt $a \cap b = \emptyset$
`berührtGeometrie`	zwei `Geometrien` berühren sich $(I(a) \cap I(b) = \emptyset) \wedge (a \cap b \neq \emptyset)$
`schneidetGeometrie`	zwei `Geometrien` schneiden sich $dim(I(a) \cap I(b)) < max(dim(I(a)), dim(I(b))) \wedge$ $(a \cap b \neq a) \wedge (a \cap b \neq b)$
`enthältGeometrie`	eine `Geometrie` ist vollständig in einer anderen enthalten $(a \cap b = a) \wedge (I(a) \cap A(b) = \emptyset)$

5.2.2 Symbolische Ortskonzepte

Symbolische Ortskonzepte werden unter einem Oberkonzept `Ort` zusammengefasst. Es verfügt nur über eine Relation: `hatGeometrie` bindet eine `Geometrie` an einen Ort. Somit können jeder symbolischen Ortsbeschreibung auch ein geographischer Ort und damit letztlich auch die Koordinaten, die ihn im Raum verankern, zugewiesen werden. Das Oberkonzept `Ort` hat zwei Unterkonzepte: `BenannterOrt` und `UnbenannterOrt`. Der `BenannteOrt` verfügt über Unterkonzepte wie `Land`, `Region`, `Verwaltungseinheit`, `AdressierbarerOrt` und `Straße`. Das Konzept `Straße` wird in Kapitel 6.2.2 noch mit anwendungsspezifischen Ontologien weiter verfeinert. `Verwaltungseinheit` kann noch weiter verfeinert werden in `Staat`, `Gemeinde` und `Stadt`. `AdressierbarerOrt` sind alle Orte, die mit Adressen versehen werden können, z. B. Hotels, Restaurants, Tankstellen, Gebäude, usw.

Symbolische Ortskonzepte sind in hohem Maße anwendungsabhängig. Eine Ontologie aller in diese Kategorie einzuordnender Konzepte würde diese Arbeit sprengen und wäre im Hinblick der Zielsetzung dieser Arbeit auch nicht sinnvoll. Für weitere Konzepte sei auf die allgemeine obere Ontologie SUMO verwiesen, die viele weitere definiert. Die hier aufgezählten Konzepte genügen für die in Kapitel 6 vorgestellten Anwendungen und sind daher ausreichend.

5.3 Umweltwahrnehmungen

Dieser Bereich umfasst alle Daten, die in modernen Fahrzeugen mit Sensoren erfasst werden können. Diese Daten sind dynamischer Natur und umfasst auch ein Modell für Kontextqualität. Für die Modellierung der im Fahrzeug mit Hilfe von Sensoren anfallenden Umweltwahrnehmungen kommt das schon angesprochene ASC-Modell [Str04] zum Einsatz.
`Aspekte` dienen der Aggregation semantisch Äquivalenter `Skalen`. So kann die Geschwindigkeit z. B. mit Hilfe einer $\frac{m}{s}$-`Skala` gemessen werden, genauso wie mit einer $\frac{km}{h}$-`Skala`, je nach Bauart der Sensoren. Beide beziehen sich aber auf denselben `Aspekt`, nämlich die Geschwindigkeit. Für jede `Skala`, die einem `Aspekt` zugeordnet wird, muss die Bedingung gelten, dass jedem Wert der `Skala` S_1 ein Wert der `Skala` S_2 zugeordnet werden kann. Dies wird im Rahmen des ASC-Modells mit Hilfe von `IntraOperations` erreicht, die – auf das Beispiel des Geschwindigkeitsapekts bezogen – sozusagen $\frac{m}{s}$ in $\frac{km}{h}$ umrechnen und/oder umgekehrt. `Skalen` werden mit Hilfe der Relation `hasScale` an einen `Aspekt` gebunden, eine Standardskala mit der Relation `hasDefaultScale`. Letztere Relation darf pro `Aspekt` höchstens einmal definiert sein. Abbildung 5.1 soll die Modellierung eines Aspekts verdeutlichen:

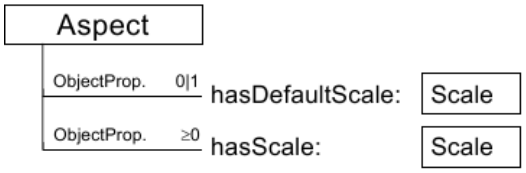


Abbildung 5.1: Ein `Aspekt` nach dem ASC-Modell, Quelle: [Str04]

`Skalen` formalisieren Wertebereiche von `ContextInformationen`. Die Beziehung zwischen `ContextInformationen` und `Skalen` werden durch die Relation `hasMember` ausgedrückt. Es gilt zusätzlich, dass diese `ContextInformationen` vom gleichen Typ sind, damit sie im Hinblick auf die `Skala` als gültig bezeichnet werden können: Beispielsweise kann eine $\frac{m}{s}$-Skala nur „zahlenartige“ Kontextinformationen unter einer Skala subsumieren, ob diese Zahlen aber Ganzzahlen oder Gleitkommazahlen sind, ist für die `Skala` unerheblich. String-Werte können einer $\frac{m}{s}$-Skala natürlich nicht zugeordnet werden, wohl aber einer anderen `Skala`, die Geschwindigkeiten nur in die Bereiche „schnell“, „mittel“ und „langsam“ unterteilt. Um überprüfen zu können, ob eine bestimmte `ContextInformation` auch Teil einer `Skala` ist, gibt es die Operation `memberCheck`. Operationen des Typs `IntraOperation` bilden Werte zwischen unterschiedlichen `Skalen` desselben `Aspekts` ab, `InterOperations` zwischen `Skalen` *unterschiedlicher* Aspekte. Zusätzlich sollte jede `Skala` eine Aussage über die von ihr verwendete Einheit mittels der Relation `hasUnit` machen. Abbildung 5.2 zeigt das Modell grafisch:

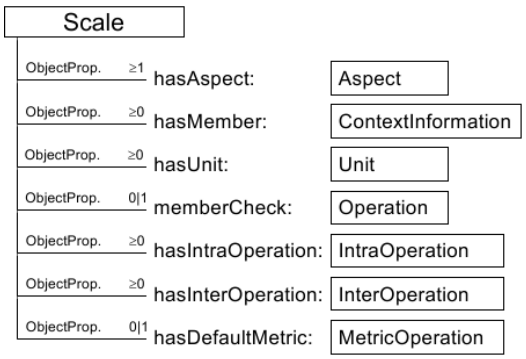


Abbildung 5.2: Eine `Skala` nach dem ASC-Modell, Quelle: [Str04]

`ContextInformationen` sind die eigentlich von Sensoren gelieferten Werte – sie charakterisieren einen Entität bezüglich eines bestimmten `Aspekts`, der mittels der Relation `characterizes` angegeben werden kann. Die Beziehung `usedByScale` ordnet ihr eine `Skala` zu. Abweichend vom originalen ASC-Modell wird auf den VANET-Anwendungsfall bezogen nicht eine Relation `hasQuality` definiert, die der Information eine (nicht näher definierte) Qualität beistellt, sondern spezifische Qualitätsrelationen:

- `Abdeckung`: dieser Wert gibt an, zu welchem Grad ein Sensor überhaupt den gesamten Wertebereich eines Aspekts abdecken kann. Beispielsweise gibt es Sensoren, die nicht imstande sind, Temperaturen über 100° Celsius zu erfassen – ihre Abdeckung wäre eingeschränkt

- `Auflösung`: die kleinstmögliche Einheit einer Kontextinformation, die ein Sensor erfassen kann: Temperatursensoren können beispielsweise eine Auflösung von einem Grad haben, d. h. er ist nicht imstande, Werte zwischen z. B. 3 und 4 Grad zu erfassen, sondern würde derartige Werte immer einem der beiden Messwerte zuordnen.

- `Genauigkeit`: Ein Qualitätsmaß, das angibt, wie weit ein Messpunkt von vom tatsächlichen Wert abweichen kann. GPS-Sensoren können z. B. einen mittleren Fehler von 5 Metern haben, so dass ein von einem GPS-Sensor ermittelter Wert bis zu 5 Meter von der eigentlichen Messposition abweicht.

- `Zeitgenauigkeit` betrifft den Zeitraum, der zwischen der Messung und der „Bekanntgabe“ des Messwerts durch den Sensor vergeht. Manche Sensoren brauchen intern Zeit, um einen Messwert erfassen zu können. Dieser kann dann schon wieder veraltet sein.

Zusätzlich zu diesen Qualitätsrelationen verfügt eine `ContextInformation` auch noch über einen Zeitstempel, der angibt, wann die Kontextinformation gemessen wurde.

In der Praxis sieht die Verwendung des ASC-Modells an einem Beispiel wie folgt aus:

```
<owl:Class rdf:ID="Integer">
  <rdfs:subClassOf rdf:resource="#ContextInformation"/>
</owl:Class>

<owl:DatatypeProperty rdf:ID="hasIntegerValue">
  <rdfs:domain rdf:resource="#Integer"/>
  <rdf:range rdf:resource="xsd:integer"/>
</owl:DatatypeProperty>

<Integer>
  <hasIntegerValue rdf:datatype="xsd:integer">120</hasIntegerValue>
  <characterizes rdf:resource="#OilTemperature"/>
  <usedByScale rdf:resource="#CelsiusScaleOfTemperatureAspect"/>
<Integer/>
```

Listing 5.1: Eine konkrete Ausprägung des ASC-Modells

Das Beispiel ist wie folgt zu lesen: im ersten Block wird deutlich gemacht, dass man es mit einem `Integer`-Wert zu tun hat, der als `ContextInformation` zu interpretieren ist, da er als Subklasse von `ContextInformation` modelliert wurde. Im zweiten Block wird eine Relation `hasIntegerValue` definiert, die besagt, dass die Klasse `Integer` auch über einen Integer-Wert (`xsd:Integer`) verfügt. Im dritten Block wird eine anonyme Instanz der `Integer`-Klasse angelegt und ihr der Wert 120 zugewiesen. Gleichzeitig wird klar gemacht, dass damit eine `OilTemperature`-Entität charakterisiert wird, und zwar auf einer Celsius-Skala eines (nicht näher definierten) Temperatur-Aspekts. Qualitätsrelationen und Zeitstempel wurden hier aus Einfachheitsgründen weggelassen.

Mit diesem grundlegenden Satz an Modellen für Umweltwahrnehmungen können nun beliebige Kontextinformationen, die durch Sensoren mit bestimmten Skalen und Qualitätseinschränkungen zu verschiedenen Zeitpunkten gemacht werden, modelliert werden.

6 Anwendungen

Um den Nachweis der Leistungsfähigkeit ontologischer Kontextmodelle zu führen, sollen im Folgenden ihr Nutzwert und auch ihre Grenzen anhand dreier exemplarischer Anwendungen aus dem Bereich von VANETs näher erläutert werden. Die Anwendungen sind folgende:

1. ein kontextbewusstes Recommendersystem für Tankstellen
2. Kooperative Kollisionsvermeidung
3. Kontextbasiertes Adressieren und Routing in VANETs anhand eines Windböenwarners

Die Anwendungen wurden in Simulationsumgebungen umgesetzt. Die Architektur und Implementierung sowie die Ergebnisse der Simulationsläufe werden im folgenden Kapitel vorgestellt.

6.1 Context-aware Recommendersysteme

Die folgende Anwendung setzt ein kollaboratives Empfehlungssystem (Recommendersystem) um, das Fahrern Tankstellen empfiehlt, die von ihm und anderen Benutzern als gut bewertet wurden. Zusätzliche Kriterien aus dem Kontext des Fahrers bzw. des Fahrzeugs sind hier bei der aktuelle Tankstand und somit die aktuelle Reichweite, so dass dem Fahrer auch nur die Tankstellen empfohlen werden, die tatsächlich noch für ihn erreichbar sind. Die Empfehlungen werden dadurch laufend an die dynamische Situation angepasst. Die zur Empfehlungsgenerierung notwendigen Bewertungsinformationen anderer Nutzer werden dabei über das VANET ausgetauscht.

Detailliertere Beschreibungen der Anwendung finden sich in [Bro07, WBE09, WBE08, BEW08, WE07].

6.1.1 Grundlagen

6.1.1.1 Recommendersysteme

Eine Grundlage für die Anwendung bilden Recommendersysteme: im verbreitetsten Fall dienen sie dazu, das Interesse von Benutzern an bestimmten Produkten vorherzusagen

und ihnen dann geeignete zum Kauf vorzuschlagen[1]. Um das Interesse eines Kunden an anderen Produkten vorhersagen zu können, muss im ersten Schritt eine Erfassung seines „Geschmacks“ versucht werden: ein Benutzerprofil wird erstellt. Dabei unterscheidet man zwischen zwei Möglichkeiten:

- *explizite* Methoden: der Benutzer wird ausdrücklich um die Bewertung eines von ihm gekauften Gegenstandes gebeten.
- *implizite* Methoden: im Hintergrund führt der Betreiber eines Webshops Analysen durch, die aufzeigen sollen, welche Objekte von einem Nutzer im Onlineshop beobachtet werden, wie oft sie beobachtet werden und welche er kaufte. Denkbar sind auch Methoden, die das soziale Netzwerk eines Nutzers untersuchen und Nutzer mit ähnlichen Vorlieben und Abneigungen suchen.

Mit der Hilfe dieser Daten wird dann mit unterschiedlichen Methoden versucht, neue Objekte zu finden, die der Interessenslage des Nutzers entsprechen. In [Bur02] werden die Methoden, diese aufzufinden, nach drei Kriterien unterschieden und dann in fünf Kategorien eingeteilt. Die drei Kriterien sind:

- die verwendeten *Hintergrunddaten*, die schon vor der Empfehlungsgenerierung verfügbar sind
- die etwaige Verwendung und Art von *Eingabedaten* über den Benutzer, die entweder explizit oder implizit ermittelt werden müssen
- der verwendete *Algorithmus*, der dann zur Empfehlungsgenerierung benutzt wird

Die Kategorien und Kriterien sind in Tabelle 6.1 erläutert. Dabei entspricht U der Menge aller Nutzer im System, I der Menge aller Objekte, für die Empfehlungen ausgesprochen werden können, u dem einen Nutzer, für die die Empfehlung generiert werden soll und i steht für das Objekt, dessen Bewertung vorhergesagt werden soll.

[1] vgl. das Merkmal auf der Amazon-Webseite: „Kunden, die verwandte Artikel gekauft haben, haben auch die folgenden Artikel gekauft“ (`http://www.amazon.de/`)

Technik	**Hintergrunddaten**	**Eingabedaten**	**Algorithmus**
Kollaborativ	Bewertungen der Nutzer in U für Objekte in I	Von u bewertete Objekte in I	Identifiziere die Nutzer in U, die ein u ähnliches Bewertungsprofil haben und extrapoliere dann von ihren Bewertungen ausgehend eine Bewertung von i.
Inhaltsbasiert	Metadaten über Objekte in I	Die Bewertungen von u für Gegenstände aus I	Errechne einen Klassifikator, der zum Bewertungsverhalten von u passt und wende ihn dann auf i an.
Demographisch	Demographische Informationen über U und Bewertungsinformationen für Objekte aus I	Demographische Informationen über u	Identifiziere Nutzer, die demographisch ähnlich zum Nutzer u sind und extrapoliere dann von ihren Bewertungen ausgehend.
Nutzenbasiert	Eigenschaften der Objekte in I	Eine Nutzenfunktion, die Präferenzen des Nutzers u hinsichtlich der Objekte in I beschreibt	Wende die Nutzenfunktion auf alle Objekte in I an und ermittle dann die Platzierung von i.
Wissensbasiert	Eigenschaften der Objekte in I und Information darüber, wie diese die Bedürfnisse eines Nutzers befriedigen können	Eine Beschreibung der Bedürfnisse des Nutzers u	Leite eine Überdeckung der durch i befriedigten Bedürfnisse und denen des Nutzers u ab.

Tabelle 6.1: Techniken zur Empfehlungsgenerierung nach [Bur02]

Kollaborative Filter versuchen Ähnlichkeiten zwischen Benutzern (benutzerbasierte Recommender) oder zwischen Objekten (itembasierte Recommender) zu errechnen und verwenden diese Information dann, um Bewertungen unter Heranziehung der Bewertungen ähnlicher Nutzer oder ähnlicher Objekte für noch nicht bewertete Objekte vorherzusagen. Diese Klasse der Recommender leidet unter dem sogenannten *Anlaufproblem* (engl.

cold start problem), da für neu in das System eintretende Nutzer und Objekte keine Bewertungen vorliegen und somit kein Ähnlichkeitsprofil erstellt werden kann. Algorithmen zur Bewertungsvorhersage sind z. B. Slope One (vgl. [LM05]) und Tree Clustering (z. B. [OH01]). Vorteil von kollaborativem Filtern ist, dass diese Algorithmen in der Lage sind, „Nischen“ (Items, die ähnlich bewertet wurden) zu erkennen, kein Domänenwissen notwendig ist, die Bewertung hinsichtlich Qualität besser werden, je mehr Bewertungen ins System einfließen und diese Bewertungen personalisiert, d. h. auf den einzelnen Benutzer zugeschnitten sind. Nachteil sind außer dem Anlaufproblem die starke Abhängigkeit von der Qualität der gesammelten Bewertungen und die Beeinflussbarkeit durch statistische Abweichungen (und damit die Manipulierbarkeit durch Nutzer).

Inhaltsbasierte Recommender versuchen, Objekte aufgrund von Metainformationen (bei Büchern z. B. Autor, Erscheinungsjahr, Verlag, usw.) zu klassifizieren. Je nach der Art der Metainformationen über bereits gekaufte Produkte kann der aus den Metainformationen erstellte Klassifikator dem Nutzer neue Objekte vorschlagen. Ein Beispiel für solche inhaltsbasierten Filter sind Spamfilter, die mit Bayeschen Filtern arbeiten und so E-Mails aufgrund ihres Inhalts und anderer Metainformationen (Header) in zwei Klassen einteilen: Spam und Nicht-Spam. Nachteil dieser Verfahren ist die Tatsache, dass eine möglichst große Anzahl an Informationen über die zu empfehlenden Objekte vorhanden sein sollte, die aber oft nur schwer zu erfassen und modellieren sind. Problematisch ist hierbei auch, dass Metadaten trotz gleicher Semantik unterschiedlich bezeichnet werden, was die Klassifikation erschwert.

Demographische Recommender funktionieren im Prinzip wie kollaborative Filter, verwenden als Ausgangsmaterial aber nicht Bewertungen von Nutzern, sondern demographische Informationen (Alter, Geschlecht, usw.). Aufgrund dieser Informationen wird dann versucht, eine Ähnlichkeitsmatrix zwischen Nutzern zu erstellen. Problematisch ist, dass die Preisgabe solcher Informationen vom Nutzer oft nicht erwünscht ist.

Regelbasierte Recommender sind recht einfach aufgebaut. Als Grundlage verwenden sie ein Regelwerk, das mit Hilfe von einfachen Wenn-Dann-Entscheidungen einem Nutzer neue Objekte vorschlägt. Ein Nachteil dieser Systeme besteht darin, dass zur Erstellung der Regeln Domänenwissen notwendig ist und dass die Empfehlungen statisch sind. Vorteilhaft ist aber, dass man die Art der Empfehlungen gezielt steuern kann.

Wissensbasierte Recommender nutzen die von Nutzern explizit vorgegeben Vorlieben [Bur00]. Dazu wird ihm zunächst ein Referenzitem i vorgeschlagen und er muss dann versuchen, seine Vorlieben dadurch preiszugeben, indem er angibt, inwiefern das Referenzitem von seinen Vorstellungen abweicht („teurer“, „schneller“, „größer“, usw.). Daraufhin werden

ihm entsprechende Objekte vorgeschlagen. Um ähnliche Objekte finden zu können, müssen Maße angegeben werden, hinsichtlich derer die Objekte in ihrer Ähnlichkeit bewertet werden. Maße müssen unterschiedlich gewichtet werden, da in unterschiedlichen Anwendungsdomänen verschiedene Maße dominant sein können und da Nutzer durch die Gewichtung versuchen können, die Ergebnismenge hinsichtlich mehrerer Zielvorgaben optimieren zu können. Nachteil an diesem Verfahren ist die große Menge an nötigem Domänenwissen über die zu empfehlenden Objekte und Unklarheit darüber, welche Bewertungsmaße für diese sinnvoll sind. Zusätzlich dazu wird noch funktionales Wissen verwendet, um genau entscheiden zu können, in welche Richtung sich ein Objekt auf der Skala „bewegt“, wenn die Gewichtung eines Maßes an die Vorlieben des Nutzers angepasst wird, dazu noch Wissen über den Nutzer selbst. Der größte Vorteil besteht darin, dass diese Arten von Recommendern keine Probleme in der Anlaufphase haben.

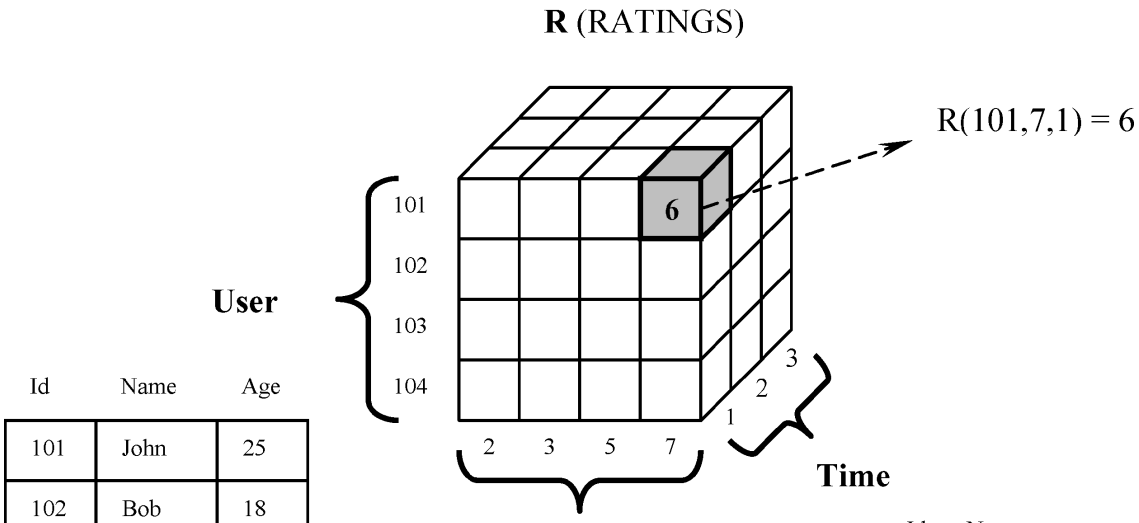
Um die Vor- und Nachteile der einzelnen Techniken auszugleichen, werden sie oft kombiniert: z. B. wissensbasierte Recommender, um die Probleme der Anlaufphase zu umgehen, und kollaborative Filter, um die Qualität der Empfehlungen bei wachsender Bewertungszahl zu nutzen. Durch die Vermischung, Kaskadierung, Gewichtung oder das Umschalten zwischen verschiedenen Techniken innerhalb eines Recommendersystems entsteht dann ein *hybrider Recommender* wie der in dieser Anwendung vorgestellte Ansatz.

6.1.1.2 Kontext in Recommendersystemen

Um die Qualität von Empfehlungen im Allgemeinen zu erhöhen, verfolgt [Che05] den Ansatz, den Kontext, in dem sich ein Nutzer befindet, mit in die Empfehlungsgenerierung einzubeziehen und Empfehlungen an den Kontext des Nutzers anzupassen. Die Empfehlungen werden somit „schärfer“, da sie nur noch in einem bestimmten Kontext gültig sind. Allgemein erweitert der Ansatz kollaboratives Filtern um eine weitere Dimension: zum Nutzer und zum Item kommt noch Kontext hinzu, so dass sich eine dreidimensionale Matrix ergibt. Empfehlungen können jetzt nicht nur für ähnliche Nutzer gegeben werden, sondern auch für unterschiedliche Nutzer, die sich aber in ähnlichen Kontexten befinden. Eine Empfehlung wird nur dann ausgesprochen, wenn die ihr zugrunde liegenden Bewertungen in einem ähnlichen Kontext gemacht wurden wie der Kontext des aktuellen Nutzers, für den die Empfehlung generiert werden soll.

Ähnlich wird in [ASST05] vorgegangen, nur dass hierbei auch noch die Multidimensionalität des Kontexts selbst berücksichtigt wird. Nun kann nicht nur mehr eine Art von Kontext betrachtet werden, sondern mehrere. Grundsätzlich werden Daten als Tupel mit den Dimensionen $D_1, \ldots, D_n$ beschrieben, jede einzelne Dimension kann wiederum mehrere Attribute haben. Als Beispiel gilt die Dimension Zeit mit den Attributen <Sekunden, Minuten, Stunden, ...>. Weitere Dimensionen sind z. B. Nutzer und Item mit ihren jewei-

ligen Attributen. Abbildung 6.1 soll einen solchen mehrdimensionalen Empfehlungsraum verdeutlichen:



Id	Name	Age
101	John	25
102	Bob	18
103	Alice	27
104	Mary	21

Id	Name	Cost
2	AB17	250.00
3	AB23	299.95
5	XY70	150.00
7	ZZ55	115.50

Id	Name
1	Weekday
2	Weekend
3	Holiday

Abbildung 6.1: Ein Modell für den Empfehlungsraum <Nutzer×Item×Zeit> [ASST05]

Mehr als drei Dimensionen sind machbar, dabei sind die Attribute einzelner Dimensionen auch aggregierbar (Durchschnittsbildung, Median, usw.) und hierarchisch gliederbar. Die Zeitdimension mit ihren Attributen stellt z. B. gleichzeitig eine Hierarchie dar. Weitere Hierarchien können explizit für verschiedene Attribute vorgegeben werden. Um letztendlich zu einer Empfehlung zu kommen, werden mehrdimensionale Strukturen auf 2-dimensionale mit Hilfe eines reduktionsbasierten Ansatzes abgebildet. Im zweidimensionalen Fall sind dann klassische Techniken wieder anwendbar, die schon gut erprobt sind.

6.1.2 Konzept

Umgesetzt wird eine Anwendung, die folgenden Ereignisfluss unterstützt:

- Ermitteln des aktuellen Kontexts. Dazu zählen die folgenden Dimensionen:

  - Benutzer
  - Tankstelle mit den Attributen <Betreiber, Position>
  - Tankstand mit den Attributen <Füllstand (prozentual), Reichweite>
  - Ort (eigene Position). Dabei ist unerheblich, mit welchem Attribut (Stadt, PLZ, GPS-Position) der Ort ermittelt wird, da sich mit Hilfe der Ortsontologie diese Informationen aufeinander abbilden lassen.

  Zu diesem Zweck werden die entsprechenden Ontologien (vgl Kapitel 5) mit Daten vom Fahrzeugbus instantiiert und dienen somit als Wissensbasis, auf die dann zugegriffen wird.

- Nach Analyse der Kontextdaten wird nach Tankstellen in der Nähe der aktuellen Position gesucht.
- Es werden alle Tankstellen heraus gefiltert, die nicht den Vorgaben des Nutzers entsprechen.
- Nun werden Bewertungen und Preisinformationen von relevanten Tankstellen über die VANET-Schnittstelle angefordert, sofern nicht schon welche empfangen wurden, die bereits verarbeitet werden können. Es ist irrelevant, ob diese Informationen von den Tankstellen selbst stammen oder von anderen Fahrzeugen im VANET verschickt werden.
- Mit dem Erhalt der Bewertungen kann der kollaborative Filter starten. Nun werden einige Tankstellen empfohlen und zusätzlich (falls vorhanden) Preisinformationen angezeigt
- Der Nutzer wählt eine Tankstelle aus und übernimmt sie als Ziel ins Navigationssystem

Die grobe Architektur der Anwendung ergibt sich wie in Abbildung 6.2 dargestellt. Der `HybridRecommender` vereinigt zwei Basisrecommender: einen inhalts- bzw. wissensbasierten `ContentRecommender`, der zusätzlich vor einen multidimensionalen kollaborativen Filter `MDRecommender` geschaltet wird. Diese Kombination ermöglicht es, im ersten Schritt sehr viele unsinnige Empfehlungen auszuschalten, die sich nicht mehr in Reichweite befinden, um im zweiten Schritt mit reduzierter Eingabemenge dann auf die Präferenzen des Nutzers einzugehen. Um das Anlaufproblem zu minimieren, kann der `ContentRecommender` die Eingabemenge mit zusätzlichen Empfehlungen auffüllen, sofern sie nicht den eingestellten Nutzervorlieben und dem Fahrzeugkontext widersprechen. Der `BusMonitor` ist für die Ermittlung des Fahrzeugkontextes und des Ortes zuständig, indem er auf die Fahrzeugdaten zugreift und die dort ermittelten Daten in Instanzen der Basisontologien einsortiert. Außerdem ist er für den Start des Recommenders zuständig, der nur dann ausgelöst wird,

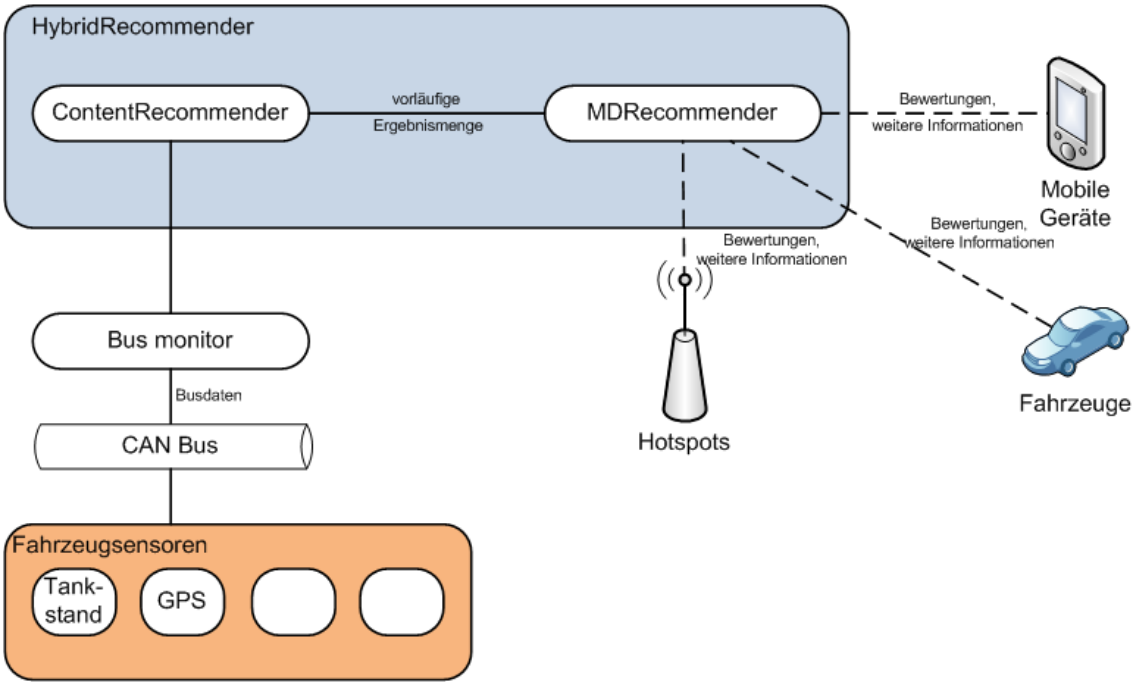


Abbildung 6.2: Architekturübersicht des Tankstellenrecommenders [WBE09]

wenn der Tankstand eine bestimmte Schwelle unterschreitet. Bewertungen und Preisinformationen werden über die VANET-Schnittstelle von Hotspots, mobilen Geräten und anderen Fahrzeugen eingesammelt.

Der `MDRecommender` macht sich dabei eine Eigenschaft des ontologischen Kontextmodells ganz besonders zu Nutze: für den Fall, dass in einem bestimmten Kontext (im vorliegenden Fall: Ortskontext) nicht ausreichend Bewertungen für kollaboratives Filtern vorliegen, kann er durch die Ober-Unterklassen-Struktur der Ortsontologie eine Hierarchieebene nach oben wechseln und somit zwar unschärfere, weil allgemeinere, Bewertungen erhalten, aber die Anzahl der zu verwendenden Bewertungen vergrößern: so kann der Recommender im ersten Schritt alle Bewertungen für die aktuellen GPS-Position verwenden, dann die für den aktuellen PLZ-Bereichs, dann für die aktuelle Stadt, usw. Denkbar wäre auch das Ausnutzen anderer Ähnlichkeitsbeziehungen, die explizit in die Ontologie modelliert werden müssten.

6.1.3 Implementierung und Evaluation

Implementiert wird das System in Java unter Zuhilfenahme der *Taste*-Recommender-Bibliothek[2]. Die Simulationsumgebung besteht aus mehreren Komponenten: eine Klasse `VehicleBus` simuliert den CAN-Bus des Fahrzeugs und liest aus einer Textdatei Daten aus, die die simulierten Sensoren mit GPS- und Tankstandsinformationen (`GPSMessage` und `FuelKMMessage` versorgen. Am `VehicleBus` können sich gemäß Observer-Pattern mehrere Listener registrieren, die den Empfehlungsprozess bei Unterschreiten einer (einstellbaren) Schwelle starten. Die Daten der Textdatei stammen dabei aus der Aufzeichnung von CAN-Busdaten einer echten Fahrt durch die Münchener Innenstadt und sind somit realistisch. Während der Fahrt durch die Stadtbezirke Milbertshofen, Moosach und Feldmoching reduziert sich der Tankstand langsam von 22 Liter auf 20 Liter.

Die VANET-Kommunikation wurde durch P2P-TCP-Kommunikation simuliert, wobei jeder Peer über drei Klassen verfügt: `VanetServer`, `VanetClient` und `VanetService`. Der `VanetServer` nimmt TCP-Verbindungen auf Port 6000 an und nimmt dort Anfragen zu Preisinformation und Bewertungen von `VanetClients` entgegen, die er beantwortet. Als Datengrundlage dient eine Tankstellendatenbank[3] mit ca. 2300 Einträgen für Deutschland. Die Nachrichten werden im XML-Format ausgetauscht, Listing 6.1 zeigt exemplarisch das XML-Schema für eine Bewertungsnachricht.

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.w3schools.com"
 xmlns="http://www.w3schools.com"
 elementFormDefault="qualified">
  <xs:element name="RatingRelations">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="PStationRating" type="PStationRating"/>
      </xs:sequence>
    </xs:complexType>
  </xs:complexType>
  <xs:element name="PStationRating">
    <xs:complexType>
      <xs:all>
        <xs:element name="UserID" type="xs:integer"
         maxOccurs="1" minOccurs="1"/>
        <xs:element name="FuelLevelID" type="xs:integer"
```

[2] http://taste.sourceforge.net/
[3] `http://www.poigps.com`

```
        maxOccurs="1" minOccurs="1"/>
       <xs:element name="PetrolStationID" type="xs:integer"
        maxOccurs="1" minOccurs="1"/>
       <xs:element name="LocationLAT" type="xs:decimal"
        maxOccurs="1" minOccurs="1"/>
       <xs:element name="LocationLONG" type="xs:decimal"
        maxOccurs="1" minOccurs="1"/>
       <xs:element name="Rating" type="xs:integer"
        maxOccurs="1" minOccurs="1"/>
     </xs:all>
   </xs:complexType>
 </xs:element>
</xs:schema>
```

Listing 6.1: XML-Schema für Bewertungsnachrichten [Bro07]

Zur Veranschaulichung der Ergebnisse und zur Konfiguration der Anwendung wurde eine grafische Oberfläche umgesetzt, die die Position des Fahrzeugs anzeigt, die empfohlenen Tankstellen in dessen Nähe und die Restreichweite des Fahrzeugs. Zusätzlich können in einem weiteren Fenster die Ergebnisse des Empfehlungsprozesses in tabellarischer Form eingesehen werden und bestimmte Nutzerpräferenzen gesetzt werden.

Die Simulation testet vier Szenarien:

1. **Zu niedrige Angabe für den Tankstandtrigger:** bei diesem Test wird die Schwelle für das Auslösen der Anwendung auf 20 Liter festgelegt: nach den ersten Metern der Fahrt sinkt der Tankstand noch nicht auf diese Schwelle (Abbildung 6.3, Fahrzeug in der linken oberen Ecke, die Restreichweite wird in der rechten oberen angezeigt), erst bei halber Strecke wird die Schwelle erreicht und der Recommender aktiviert: er empfiehlt dann zwei Tankstellen in der Reichweite (Abbildung 6.4). Der inhaltsbasierte Recommender ist also schon ohne Eingabe von Nutzerbewertungen aktiv.

   Die Ergebnisse der Filterung werden tabellarisch in Abbildung 6.5 dargestellt: Zusätzliche Information ist der Preis von 1,00 EUR pro Liter, welche bei der anderen empfohlenen Tankstelle nicht vorliegt.

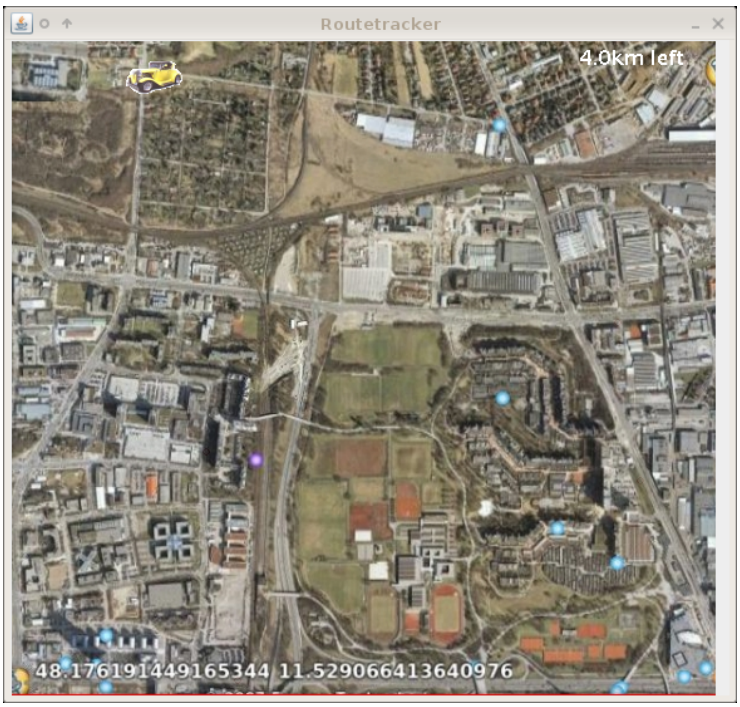


Abbildung 6.3: Die Simulationsumgebung vor dem Start des Recommenders [Bro07]

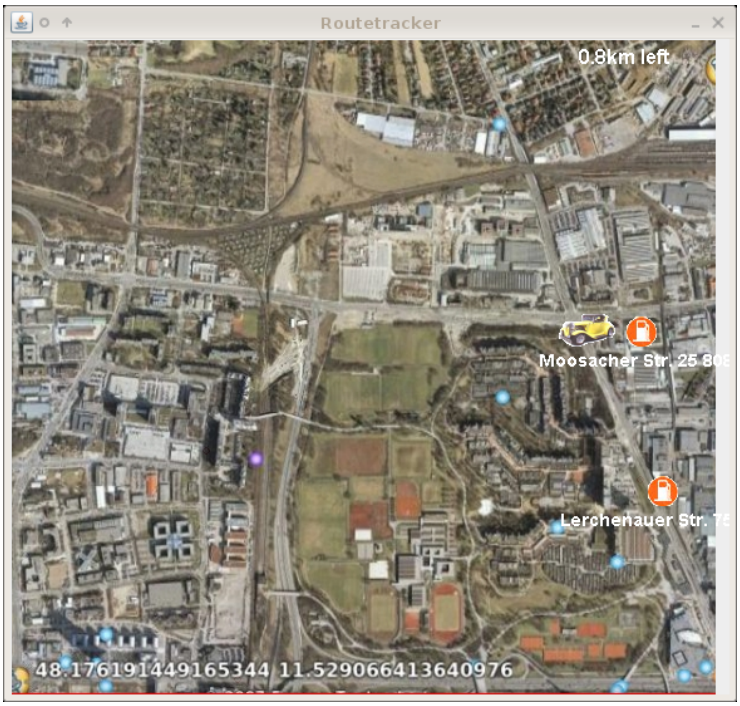


Abbildung 6.4: Die ersten Tankstellen werden empfohlen [Bro07]

Abbildung 6.5: Die Ergebnisse des Empfehlungsprozesses in tabellarischer Form [Bro07]

2. **Preisfilterung:** Nun gibt der Benutzer ein Preislimit von 0,90 EUR pro Liter vor (Abbildung 6.6). Abbildung 6.7 zeigt, dass die Tankstelle mit 1,00 EUR pro Liter nun nicht mehr empfohlen wird, die Tankstelle ohne Preisinformation hingegen schon.

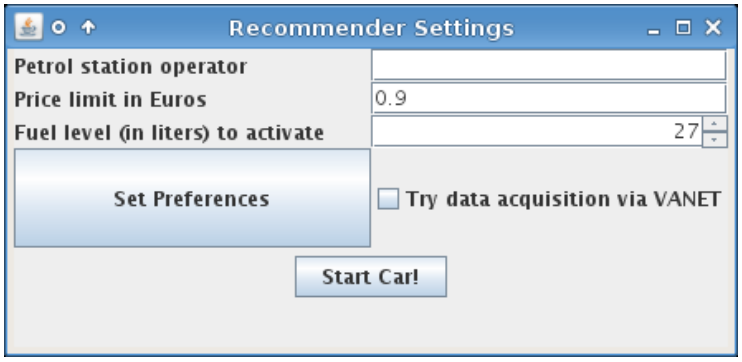


Abbildung 6.6: Der Preisfilter [Bro07]

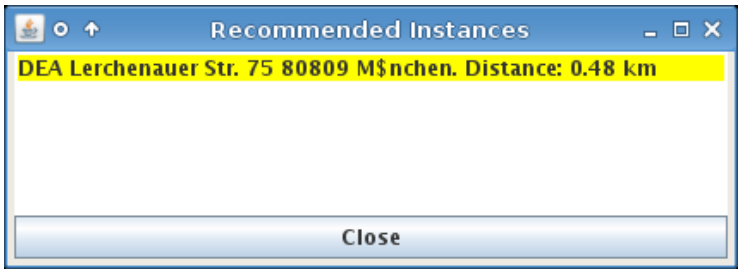


Abbildung 6.7: Durch den Preis gefilterte Ergebnisse [Bro07]

3. **Bewertungserfassung durch VANET-Kommunikation:** Dieser Test simuliert nun den kollaborativen Filter, der Bewertungen über die VANET-Schnittstelle erhält. Dazu werden in die Bewertungsdatenbank Datensätze eingefügt: einer für eine Tankstelle, die zwar relativ nah liegt, dafür vom eigenen Fahrer relativ schlecht bewertet wurde: sie wird trotzdem eher empfohlen als die Tankstellen ohne Bewertung (Abbildung 6.8, in rot). Im Lauf der Simulation kommt dann eine weitere – recht gute

– Bewertung für eine weiter entfernte Tankstelle hinzu. Diese Bewertung wird über die VANET-Schnittstelle von einem anderen Teilnehmer empfangen. Durch diese Zusatzinformation wird die zuvor unbewertete Tankstelle auf einmal höher eingestuft und führt dann die Empfehlungstabelle an (Abbildung 6.9).

Abbildung 6.8: Der kollaborative Filter empfiehlt eine Tankstelle mit Bewertung [Bro07]

Abbildung 6.9: Der kollaborative Filter empfiehlt eine besser bewertete Tankstelle trotz größerer Entfernung [Bro07]

4. **Preis- und Betreiberfilter:** Im letzten Test werden alle Filter aktiviert (Abbildung 6.10), die Simulation enthält drei Bewertungen. Abbildung 6.11 zeigt, dass die DEA-Tankstelle, die vorher auch Teil der Empfehlung war, nicht mehr empfohlen wird: der inhaltsbasierte Filter hat sie entfernt und sie zählt nicht mehr zu den Eingabedaten für den kollaborativen Prozess.

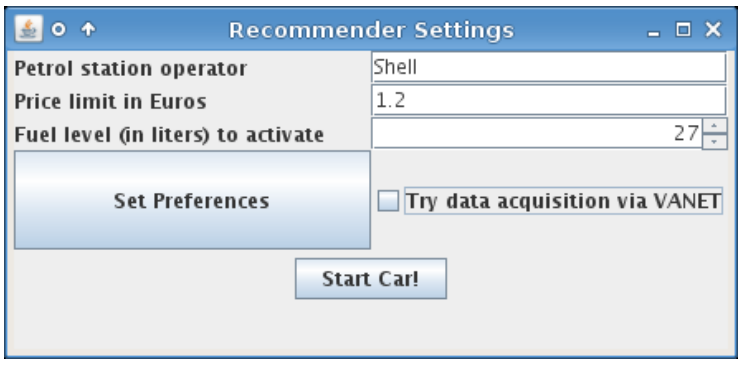


Abbildung 6.10: Alle Filter sind aktiviert [Bro07]

Abbildung 6.11: Filterergebnis mit allen Filterstufen [Bro07]

6.1.4 Zusammenfassung

Die gezeigte Anwendung erfüllt eine Anforderung, die den Einsatz von Ontologien zur Kontextmodellierung sinnvoll machen: ontologische Modelle können sehr einfach zusätzliches Domänenwissen in maschinenverarbeitbarer Form zu bestehenden Recommendern hinzufügen und sie damit um weitere Dimensionen erweitern. Bestehende Ansätze wie multidimensionale Recommender können somit weiterhin eingesetzt werden. Mit ihrer Strukturierung eines Wissensraumes mit Ober- und Unterklassenbeziehungen dienen sie auch dazu, diese neuen Dimensionen in kleinere „Attribute“ aufzuspalten. Dieselben Hierarchieebenen können sehr einfach die Strukturen abbilden, die die Empfehlungsgenerierung verbessern können, indem man bei einem Mangel an Bewertungen eine Hierarchiestufe nach oben steigt. Durch die so erhaltene Verallgemeinerung können mehr Bewertungen zur Empfehlungsgenerierung einbezogen werden – Ontologien helfen also im vorliegenden Fall, das Anlaufproblem für kollaborative Filter zu mildern. Des weiteren bieten ontologische Modelle auch hier die Grundlage für Kooperation: da im späteren Einsatz Fahrzeuge über Herstellergrenzen kommunizieren müssen (in diesem Fall über Kontextdaten wie Tankstände, Preisinformationen, Orte, usw.) bieten Ontologien eine einfache Plattform zur gemeinsamen Modellierung des nötigen Kontexts. Außerdem könnten sich (wenn auch in diesem

Szenario nicht geschehen) wissens-, regel- und inhaltsbasierte Recommender auf Ontologien stützen, indem sie die nötigen Regeln und Metainformationen als Ontologien modellieren. Durch das so zustande kommende gemeinsame Verständnis der Anwendungsdomäne kann eine Austauschbarkeit des Wissens z. B. über mehrere Fahrzeug- oder Dritthersteller hinweg ermöglicht werden. Grundsätzlich besteht auch die Möglichkeit, durch die erhöhte Ausdrucksmächtigkeit von Ontologien Ähnlichkeitsbeziehungen zwischen Konzepten explizit zu modellieren (z. B. implizit über Teilmengenbeziehungen oder explizit durch eine je nach Anwendungsfall genauer zu definierende Relation `isSimilar`), was Recommendern sehr zugute kommen würde, denn wie in 6.1.1.1 gezeigt, muss jeder Recommender ähnliche Objekte erst aufwendig identifizieren. Dieser Schritt könnte dann entfallen.

6.2 Kooperative Kollisionsvermeidung

Ziel der kooperativen Kollisionsvermeidung ist, durch konstante Überwachung des eigenen Fahrzeugkontexts und durch die Überwachung der Daten, die von anderen Fahrzeugen empfangen werden, Kollisionen zu vermeiden. Zur Strukturierung der Daten werden Ontologien eingesetzt; dabei wurde die Anwendung in zwei unterschiedlichen Versionen implementiert: einmal in einer simplen reduzierten Version, die schon die Kernfunktionalität bereitstellt und eine weitere Version, die ein erweitertes Kontextmodell einsetzt, um die Anwendungseffizienz zu erhöhen. Die Kernlogik bleibt dabei unverändert, nur die zugrunde liegenden Modelle ändern sich, was die Flexibilität der ontologischen Modelle zeigen soll. Außerdem soll die Anwendung nachweisen, dass Berechnungen auf ontologischen Modellen schnell genug sein können, um auch Echtzeitanforderungen im sicherheitskritischen Bereich, also im Stabilisierungsbereich der Fahraufgabe, einhalten zu können.

Eine ausführlichere Beschreibung der Anwendung kann in [Lut07, EL08, EL07] nachgelesen werden.

6.2.1 Grundlagen

Grundlage ist die Kollisionsberechnung für zwei Fahrzeuge, die als zweidimensionale Rechtecke im Raum modelliert werden. Ziel dieser Berechnung ist die Ermittlung einer Zeit t in Sekunden bis zur Kollision oder eines negativen Wertes für t, falls es zu keiner Kollision kommt. Die zur Verfügung stehenden Daten sind:

- Länge der Fahrzeuge
- Breite der Fahrzeuge
- Position
- Geschwindigkeit

- Beschleunigung
- Richtung

Gemäß Schulphysik lässt sich die zurückgelegte Strecke eines Punktes bei beschleunigter Bewegung mittels

$$s = t \cdot v + t^2 \cdot \frac{a}{2} \tag{6.1}$$

beschreiben, wobei s die Strecke bezeichnet, t die benötigte Zeit, v die Anfangsgeschwindigkeit und a die Beschleunigung. Werden Geschwindigkeit und Beschleunigung als Vektoren aufgefasst, um die Bahn in der Ebene beschreiben zu können, so ergibt sich folgende Gleichung:

$$\vec{s}(t) = \vec{y} + t \cdot \vec{v} + t^2 \cdot \frac{1}{2}\vec{a} \tag{6.2}$$

Zur Überprüfung, ob zwei Bahnen sich schneiden, setzt man $t_1 = t_2$. Nun erfolgt die Erweiterung auf sich schneidende Rechtecke: folgende Gleichung beschreibt die Fläche eines Rechtecks:

$$\vec{A} = \vec{y} + m \cdot \vec{l} + n \cdot \vec{b} \tag{6.3}$$

Hierbei beschreibt $|\vec{b}|$ die Breite des Fahrzeugs und $|\vec{l}|$ die Länge, $\vec{y}$ den Mittelpunkt des Rechtecks. Daher muss zusätzlich gelten:

$$-0.5 \leq m \leq 0.5 \tag{6.4}$$

$$-0.5 \leq n \leq 0.5 \tag{6.5}$$

Wenn man sich bewegende Rechtecke als Rechtecke, deren Mittelpunkt sich bewegt, modelliert, dann gilt für die Bahnvektoren durch Einsetzen von 6.3 in 6.2 folgende Gleichung:

$$\vec{s}(t) = \vec{y} + t \cdot \vec{v} + t^2 \cdot \frac{1}{2}\vec{a} + m \cdot \vec{l} + n \cdot \vec{b} \tag{6.6}$$

Die Ungleichungen 6.4 und 6.5 gelten gleichermaßen. Zur Bestimmung der Schnittpunkte zweier Rechtecke müssen nun zwei solcher Gleichungen gleichgesetzt und vereinfacht werden und somit ergibt sich dann

$$\vec{0} = \vec{y_1} - \vec{y_2} + t \cdot (\vec{v_1} - \vec{v_2}) + t^2 \cdot \frac{1}{2}(\vec{a_1} - \vec{a_2}) + m_1 \cdot \vec{l_1} + n_1 \cdot \vec{b_1} - m_2 \cdot \vec{l_2} - + n_2 \cdot \vec{b_2} \tag{6.7}$$

Diese Gleichung lässt sich nun in zwei Ungleichungen ≤ 0 und ≥ 0 aufspalten – mit den Ungleichungen 6.4 und 6.5 ergibt sich dann ein Ungleichungssystem mit fünf Unbekannten: m_1, m_2, n_1, n_2 und t, wobei t nichtlinear ist. Nun gilt es, dieses Ungleichungssystem durch Elimination der ersten vier Unbekannten zu lösen, dazu werden die Ungleichungen allgemein formuliert:

$$\vec{y} + \vec{v}t + \vec{a}t^2 + m_1\vec{l_1} + n_1\vec{b_1} + m_2\vec{l_2} + n_2\vec{b_2} \leq c \tag{6.8}$$

Je nach zu lösender Ungleichung nimmt c den Wert 0 oder 0.5 an. Jetzt ergibt sich $\vec{y}$ aus der Subtraktion der Ortsvektoren $\vec{y_1}$ und $\vec{y_2}$, $\vec{v}$ aus der Subtraktion der Geschwindigkeitsvektoren $\vec{y_1}$ und $\vec{y_2}$ und $\vec{a}$ aus $\frac{1}{2} \cdot (\vec{a_1} - \vec{a_2})$. Ersetzt man dann noch die Unbekannten m_1, m_2, n_1 und n_2 durch Variablen x_4 bis x_7 und erweitert die ersten Summanden mit $x_1 = x_2 = x_3 = 1$, dann hat man

$$\vec{y}x_1 + \vec{v}tx_2 + \vec{a}t^2x_3 + \vec{l_1}x_4 + \vec{b_1}x_5 + \vec{l_2}x_6 + \vec{b_2}x_7 \leq c \tag{6.9}$$

Nun werden noch folgende Substitutionen vorgenommen:

- der Ortsvektor $\vec{y}$ wird zerlegt in eine x-Komponente y_x und y-Komponente y_y.
- der Geschwindigkeitsvektor $\vec{v}$ multipliziert mit der Zeit t wird zerlegt in eine x-Komponente v_x und y-Komponente v_y.
- der Beschleunigungsvektor $\vec{a}$ multipliziert mit $\frac{1}{2}t^2$ wird zerlegt in eine x-Komponente a_x und y-Komponente a_y.
- l_{1x}, l_{2x}, l_{1y}, l_{2y}, b_{1x}, b_{1y}, b_{2x} und b_{2y} beschreiben jeweils die x- und y-Komponenten der Flächenvektoren $\vec{l_1}$, $\vec{l_2}$, $\vec{b_1}$ und $\vec{b_2}$

Jetzt hat man ein rein lineares Ungleichungssystem, da sich der nichtlineare Faktor t^2 hinter den Konstanten a_x und a_y verbirgt. Damit ergibt sich das folgende lineare Ungleichungssystem, wobei die ersten acht Zeilen den acht Ungleichungen 6.4 und 6.5 entsprechen (vier für jedes Fahrzeug, zwei Fahrzeuge sind an der potentiellen Kollision beteiligt), die letzten vier Zeilen den eben aufgestellten allgemeinen Ungleichungen (≤ 0, ≥ 0, x-Komponente, y-Komponente) entspricht:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ y_x & v_x & a_x & l_{1x} & b_{1x} & l_{2x} & b_{2x} \\ -y_x & -v_x & -a_x & -l_{1x} & -b_{1x} & -l_{2x} & -b_{2x} \\ y_y & v_y & a_y & l_{1y} & b_{1y} & l_{2y} & b_{2y} \\ -y_y & -v_y & -a_y & -l_{1y} & -b_{1y} & -l_{2y} & -b_{2y} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{pmatrix} \leq \begin{pmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{6.10}$$

Wenn dieses lineare Ungleichungssystem zum Zeitpunkt t erfüllbar ist, kommt es zu einer Kollision. Daher gilt es zu berechnen, ob es ein t gibt, das dieses System erfüllt. Zu diesem Zweck werden mit Hilfe der Fourier-Motzkin-Elimination die Unbekannten x_4 bis x_7

eliminiert. Das Ergebnis ist ein System quadratischer Ungleichungen der Form

$$ax_1 + bx_2 + cx_3 \leq d \tag{6.11}$$

Die Elimination von vier Variablen durch die Fourier-Motzkin-Elimination erkauft man sich durch die Erzeugung einer Vielzahl neuer Ungleichungen der obigen Form. Zusätzlich gilt noch:

- a ist aus Umformungen der ersten Spalte von 6.10 entstanden, d. h. aus Vielfachen der x- und y-Komponenten der Ortsvektoren $\vec{y_1}$ und $\vec{y_2}$. Somit ist a nicht von t abhängig und kann daher aus bekannten Werten berechnet werden.
- b ist aus Umformungen der zweiten Spalte entstanden und ist ein Vielfaches der Zeit t. Somit gilt: $b = b't$.
- c ist aus Umformungen der dritten Spalte hervorgegangen und ist ein Vielfaches des Quadrates der Zeit t. Damit gilt: $c = c't^2$
- d ist konstant.

Damit lässt sich 6.11 umformen zu

$$0 \leq d - a - b't - c't^2 \tag{6.12}$$

Der Vorteil dieser Darstellung ist nun, dass als einzige Unbekannte nur noch die Zeit t enthalten ist. Dieses quadratische Ungleichungssystem lässt sich durch die konjunktive Verknüpfung aller möglichen Lösungsintervalle für t, deren Umwandlung in disjunktive Normalform und anschließenden Test auf Erfüllbarkeit lösen. Als Ergebnis erhält man alle Intervalle für t, für die eine Kollision der Fahrzeuge auftritt. Das minimale t bezeichnet dabei den Zeitpunkt bis zum Eintreten der Kollision. Möglich ist auch, dass der Test auf Erfüllbarkeit keine Lösung ergibt: in diesem Falle kommt es zu keiner Kollision.

6.2.2 Konzept

Der obige Algorithmus wurde in der Anwendung umgesetzt, wobei die Daten zur Berechnung des Kollisionszeitpunktes aus instantiierten Ontologien stammen. Zum Einsatz kommen in der ersten Variante der Anwendung eine Ontologie, die Konzepte `Fahrzeug` und `Verkehrsteilnehmer` modelliert. Ein Fahrzeug verfügt über die vier Eigenschaften `GeschwindigkeitsAspekt`, `BeschleunigungsAspekt`, `RichtungsAspekt` und noch einen `PositionsAspekt`, die wiederum als Aspekte nach dem ASC-Modell der Context Ontology Language CoOL (siehe Kapitel 5.3 und 4.2 (also als `owl:FunctionalProperty`) konzipiert sind. Statische Daten über das Fahrzeug sind als Properties definiert; sie umfassen Werte für `Breite`, `Länge`, `ID`, `Maximalgeschwindigkeit`, `Maximalbeschleunigung`

und `Maxmialverzögerung`.

`Fahrzeuge` sind weiter in vier Unterklassen eingeteilt: `PKW`, `Bus`, `LKW` und `Motorrad`, die nicht weiter definiert werden. Abbildung 6.12 zeigt die Beziehung zwischen der Klasse `Fahrzeug` und weiteren Konzepten.

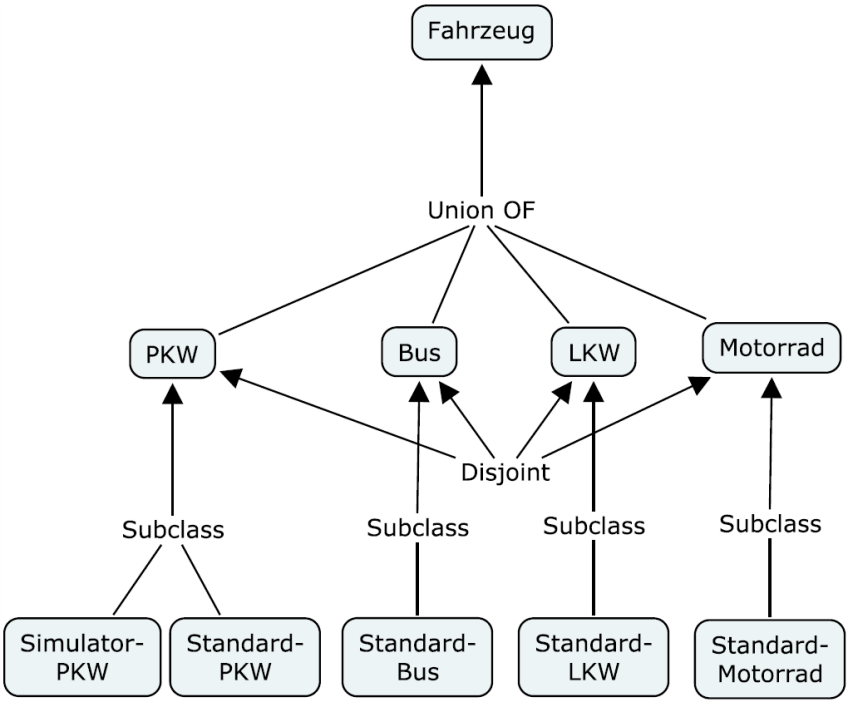


Abbildung 6.12: Die Klasse `Fahrzeug` [Lut07]

Jeder der vier Unterklassen hat Standardvertreter, die später in der Simulation eingesetzt werden. Bei ihnen sind die Eigenschaften mit sinnvollen Werten vorbelegt.

Zur Unterscheidung zwischen dem eigenen Fahrzeug und den Fahrzeugen anderer Verkehrsteilnehmer wurde entsprechende Konzepte der Ontologie hinzugefügt: eine Klasse `Eigenes_Fahrzeug` und eine Klasse `Andere_Fahrzeuge`, die als voneinander disjunkt definiert wurden. Das ermöglicht der Kollisionsberechnung, geeignete Anfragen an die Fahrzeugontologie zu stellen, da für die Assistenzanwendung nur die potentiellen eigenen Kollisionen von Interesse sind und nicht die der anderen Fahrzeuge. Aus diesem Grund wird eine Relation `Kollisionspartner` eingeführt, die von der Kollisionsberechnung unter Umständen mit einer Instanz der Klasse `Andere_Fahrzeuge` belegt wird – nämlich dann, wenn eine Kollision bevorsteht. Unter diesen Bedingungen wird dann auch die Eigenschaft `Kollisionszeit` mit einem Wert belegt. Abbildung 6.13 verdeutlicht diese Konzepte.
Die zweite Variante verwendet zusätzlich eine Ontologie, die das Straßennetz als modelliert.

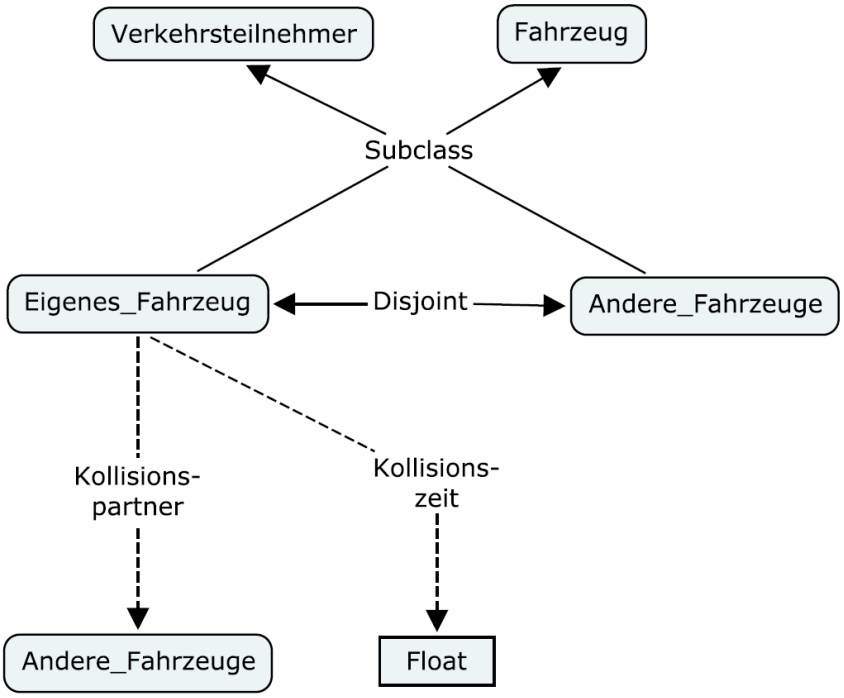


Abbildung 6.13: Zwei Klassen zur Unterscheidung zwischen dem `Eigenen_Fahrzeug` und den `Anderen_Fahrzeugen` [Lut07]

Kreuzungen sind Knoten des Graphen, Straßen die Kanten. Die Modellierung als Graph ermöglicht es, bekannte Algorithmen zur Ermittlung des kürzesten Weges zwischen zwei Punkten zu ermitteln. Für die Kollisionsvermeidung ist es interessant, zu ermitteln, welche Fahrzeuge überhaupt als potentielle Kollisionskandidaten in Frage kommen. Dazu wird der Straßengraph in seine Zusammenhangskomponenten zerlegt. Diese Zusammenhangskomponenten stellen maximale Teilgraphen dar, die miteinander über Kanten verbunden sind. Übertragen auf das Straßennetz bedeutet das, dass zwei Straßenteilnetze nur dann verbunden sind, wenn sie sich in der gleichen Zusammenhangskomponente befinden. Nur wenn sich zwei Fahrzeuge in derselben Zusammenhangskomponente befinden können sie überhaupt miteinander kollidieren, da nur dann eine Verbindung zwischen ihnen über das Straßennetz bestehen. Somit lassen sich viele unnötige Kollisionsberechnungen vermeiden.

Die einzelnen Konzepte der Straßenontologie sind wie folgt:

- Das `Strassennetz` selbst enthält alle `Abschnitte` als Eigenschaft. Diese werden über die Relation `hatAbschnitt` miteinander in Verbindung gesetzt.
- `Abschnitte` wiederum bestehen entweder aus den Klassen `Kreuzung` oder `Strasse` und werden durch vier Eigenschaften näher spezifiziert:

  - die Eigenschaft `hatZustand` ordnet ihm einen `Zustand` zu: er kann entweder `guterZustand` oder ein `schlechterZustand` sein.
  - die Eigenschaft `hatBelag` definiert den Belag eines Abschnitts: `AsphaltBelag`, `BetonBelag` oder `SchotterBelag` sind hier möglich.
  - über die Relation `erreichbar` kann angegeben werden, welcher Abschnitt von welchem anderen aus erreichbar ist. Diese Relation nimmt also wieder einen `Abschnitt` als Argument. Diese Eigenschaft ist als `owl:TransitiveProperty` gekennzeichnet
  - eine ganzzahlige `AbschnittID` dient der eindeutigen Identifikation

- die Klasse `Kreuzung` ist eine Unterklasse von `Abschnitt` und mit dieser disjunkt. Die boolesche Eigenschaft `hatAmpel` gibt an, ob die Kreuzung mit einer Ampel versehen ist oder nicht. Sie verfügt weiterhin über die Eigenschaft `hatZufahrt`, die die in sie einmündenden Straßen angibt. Argument dieser Relation ist also die Klasse `Strasse`. Es gibt drei unterschiedliche (disjunkte) Arten von Kreuzungen, die als Unterklassen modelliert sind:
  - `EchteKreuzung` ist jede `Kreuzung`, die mindestens drei Zufahrten hat – also das, was man sich gemeinhin als gewöhnliche Kreuzung vorstellt
  - `UnechteKreuzung` ist jede `Kreuzung`, die genau zwei `Strassen` miteinander verbindet. Sie kann dazu verwendet werden, einen `Strasse` in mehrere `Abschnitte` zu zerteilen, z. B. wenn sich die Eigenschaft `hatBelag` im Verlauf einer `Strasse` ändert.
  - `Endpunkt` ist eine `Kreuzung` dann, wenn sie nur eine Zufahrt hat und ist der Tatsache geschuldet, dass eine `Strasse` immer zwei `Kreuzungen` miteinander verbinden muss.

  Andere Kreuzungskonzepte wie Kreisverkehre können mit Hilfe der `EchtenKreuzung` abgebildet werden. Man kann einen solchen Kreisverkehr entweder als Kreuzung betrachten, bei der jede Einmündung als `EchteKreuzung` mit drei Zufahrten (eine von links, eine von rechts, und eine für die Zufahrt selbst) modelliert wird oder – etwas unschärfer – als eine große sternförmige `EchteKreuzung` mit einer entsprechenden Anzahl Zufahrten.

- `Strasse` bildet die andere Unterklasse von `Abschnitt`. Ihre Unterklassen wiederum sind `Autobahn`, `Landstrasse`, und `Stadtstrasse`. Jede Strasse verfügt über die Eigenschaften `hatKreuzung1` und`hatKreuzung2`, die die beiden `Kreuzungen` repräsentiert, die durch sie verbunden werden. Welche der beiden Vorfahrt wird durch die booleschen Eigenschaften `hatVorfahrt1` und `hatVorfahrt2` festgelegt. Jede Straße

verfügt außerdem über die Eigenschaften `hatNamen`, `hatHoechstgeschwindigkeit`, `hatLaenge`, `hatKurve` und `hatSpur`, die z. T. auch mehrfach vorliegen können.

Abbildung 6.14 soll die Straßenontologie in einer vereinfachten Form veranschaulichen:

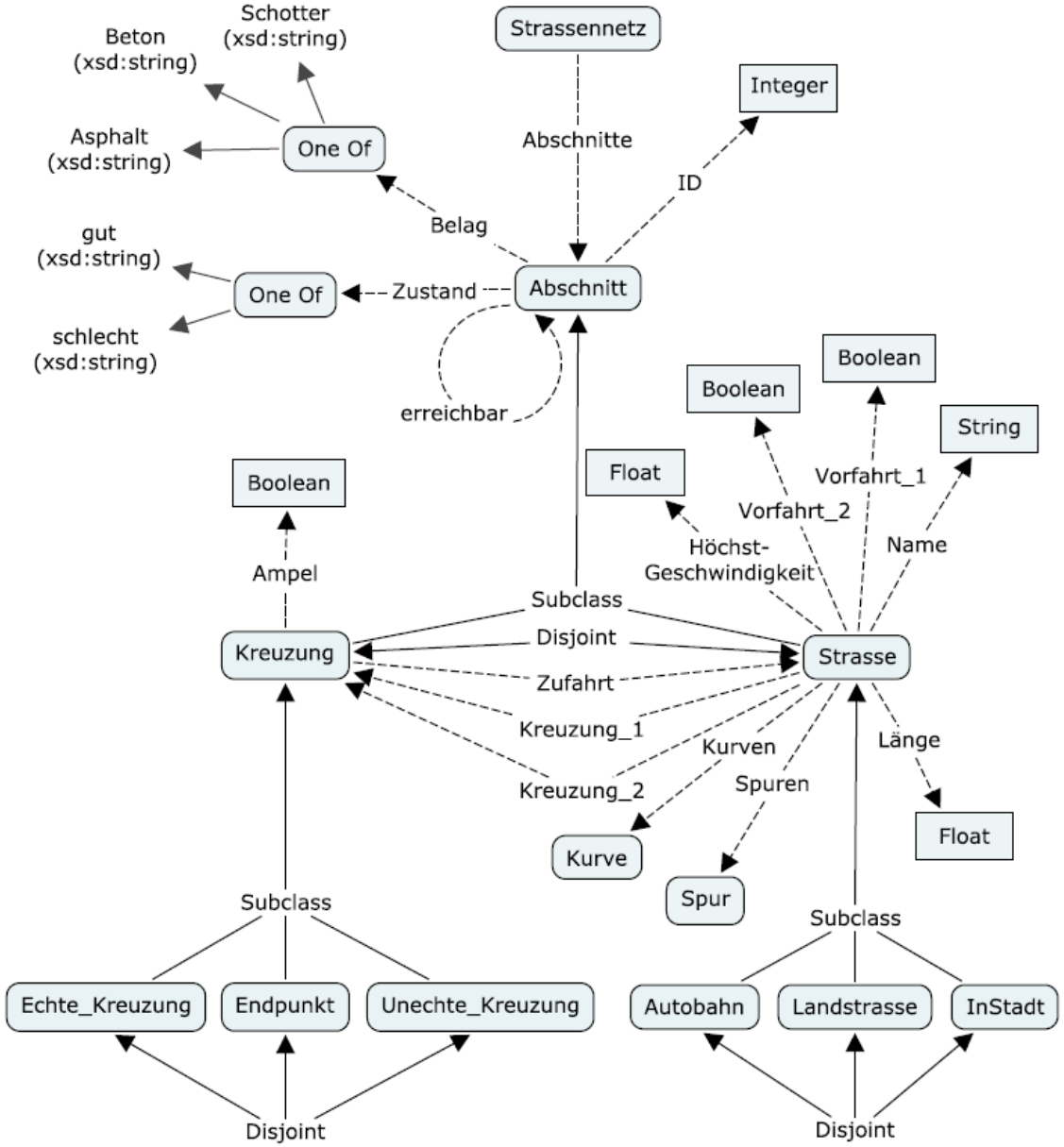


Abbildung 6.14: Die Ontologie des Straßennetzes [Lut07]

Erweitert wird die Straßenontologie durch folgende Konzepte:

- Eine Straße hat mindestens eine `Spur`, die mit einer `SpurID` eindeutig identifizierbar ist. Außerdem verfügen sie über eine Eigenschaft `hatFahrtrichtung`, die entweder

von Kreuzung 1 zu Kreuzung 2 führt, in die umgekehrte Richtung oder für beide Richtungen gleichzeitig benutzt werden kann. Zusätzlich sind die für eine Spur gültigen Verkehrsschilder über die Eigenschaft `hatVerkehrsschild` (nicht näher spezifiziert) erfasst. Spuren sind disjunkt mit `Abschnitten` und `Kurven`.

- Eine `Kurve` kann in drei Arten unterteilt werden:
  - eine `EinfacheKurve` ist nur in eine Richtung gekrümmt
  - eine `SKurve` ist zuerst in die eine, dann in die andere Richtung gekrümmt
  - eine `Serpentine` ist eine Kurve mit einer Krümmung von fast 180°.

  Kurven sind grundsätzlich durch ihre Eigenschaften `hatLaenge`, `hatKruemmung` und `hatPosition`, die als Argument eine `Geometrie` aus dem Basismodell verwendet, charakterisiert.

Veranschaulicht sind die Erweiterungen in Abbildung 6.15.

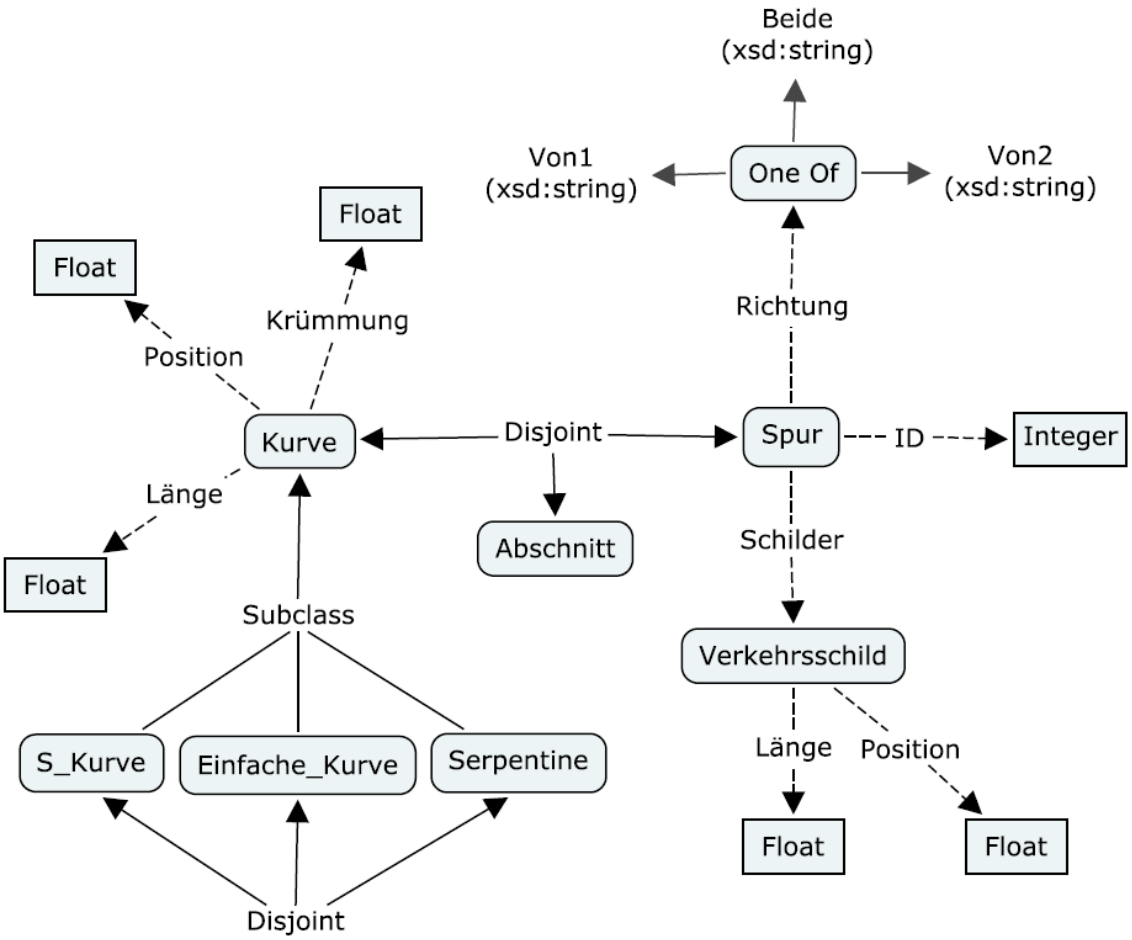


Abbildung 6.15: Die Erweiterung der Straßennetz-Ontologie [Lut07]

Um Fahrzeuge Straßenabschnitten zuordnen zu können, muss die Fahrzeugontologie der ersten Variante um eine Eigenschaft `hatAbschnitt` erweitert werden, die einen `Abschnitt` als Argument nimmt – dies stellt auch schon die einzige Veränderung dar.

6.2.3 Implementierung und Evaluation

Das System selbst ist in Java implementiert, die eigentliche Architektur der Anwendung ist übersichtsmäßig in Abbildung 6.16 zu sehen.

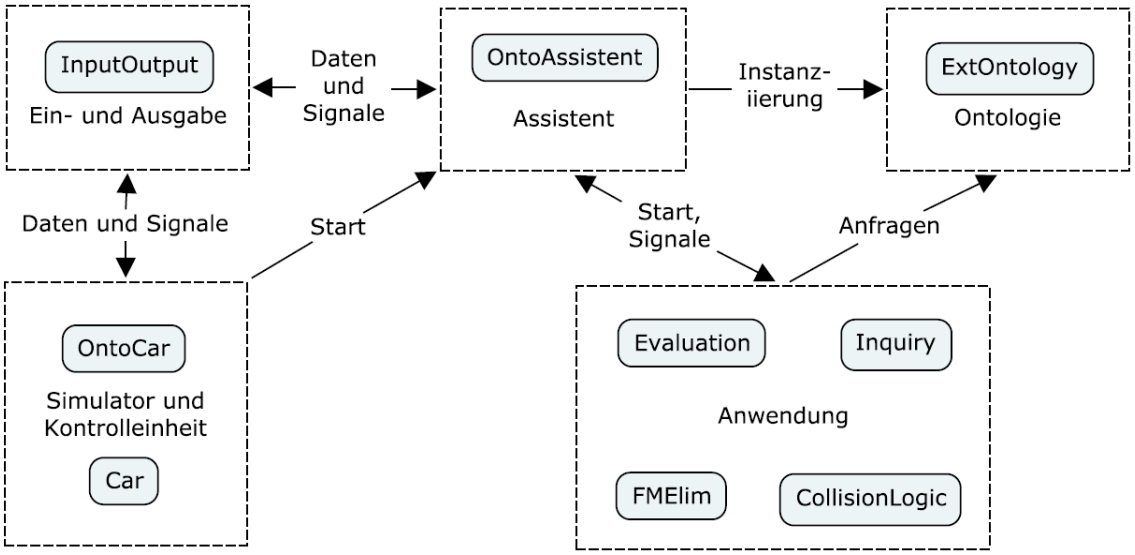


Abbildung 6.16: Die Architektur der Anwendung zur Kollisionsvermeidung [Lut07]

- Der *Simulator* stellt Eingabedaten zur Bewegung und Position der Fahrzeuge und zum Straßennetz zur Verfügung. Er reagiert außerdem auf Ein- und Ausgaben der Kollisionsvermeidungsanwendung, bremst also Fahrzeuge gegebenenfalls ab. Im vorliegenden Falle kommt der Simulator VIS_SIM[4] als mikroskopischer Verkehrssimulator zum Einsatz. Über die Klasse `OntoCar` ist der Fahrerassistent in den Verkehrssimulator eingebunden, der die Klasse `Car` verwendet. Zur Verknüpfung der beiden wird `Car` um je ein `OntoCar`-Objekt erweitert. Das `OntoCar`-Objekt wird nach einer bestimmten Anzahl von Simulationsschritten aktiv und es werden drei Methoden der Reihe nach aufgerufen:

  1. `setOntoCar()` und
  2. `updateOntoCar()` werden dazu verwendet, Fahrzeugdaten einzulesen. Dabei wird darauf geachtet, dass nicht alle sich in der Simulation befindlichen Fahrzeuge berücksichtigt werden, sondern nur die, die sich in einer bestimmten Reich-

[4] `http://mysite.wanadoo-members.co.uk/tomfotherby/Contents/Uni/Project/index.html`

weite befinden. Auf diese Weise wird die beschränkte Reichweite der VANET-Kommunikation berücksichtigt.

3. `startOntoCar()` startet nun den eigentlichen Assistenten, der in der Klasse `OntoAssistant` implementiert ist. Die von dieser Klasse erzeugten Kontrollsignale werden ausgewertet und an das `Car`-Objekt weitergeleitet, wodurch die spezifische Reaktion des Fahrzeugs eingeleitet wird.

Die `OntoCar`-Klasse wird also dazu verwendet, die Simulationsumgebung vor der Anwendung zu verbergen. Aus diesem Grund darf der Fahrerassistent auch nicht die Klasse `OntoCar` zur Kommunikation verwenden – Aufrufe sind nur in umgekehrter Richtung möglich. Falls der Assistent mit anderen Fahrzeugen kommunizieren möchte, dann muss er dazu die `InputOutput`-Klasse verwenden.

- Über *Ein- und Ausgabe* werden durch den Assistenten die Daten der anderen Fahrzeuge und die eigenen Sensordaten eingelesen. Es simuliert die VANET-Schnittstelle und die Schnittstelle zur Umgebungswahrnehmung. Über diese Schnittstelle werden auch Information über das eigene Fahrzeug an andere Verkehrsteilnehmer versendet. Die `InputOutput`-Klasse dient als Datenversandzentrale für die folgenden Daten:
  - `sensorData`: die Sensordaten des eigenen Fahrzeugs
  - `incoming`: Die Fahrzeugdaten, die von anderen Fahrzeugen empfangen werden
  - `outputMessage`: Die Fahrzeugdaten, die an andere verschickt werden sollen
  - `controlOutput`: Die vom Fahrerassistenten erzeugten Signale zur Steuerung des eigenen Fahrzeugs

- Der *Assistent* und die *Anwendung* sind die Hauptkomponenten der Anwendung und koordinieren die einzelnen Teilkomponenten. Für jedes Fahrzeug existiert zu diesem Zweck ein eigenes `OntoAssistant`-Objekt, das über die `InputOutput`-Klasse mit Daten versorgt wird und an die er seine eigenen Signale weitergibt. Der `OntoAssistant` liest bei jedem Aufruf Daten aus dem `InputOutput`-Objekt und veranlasst dann `ExtOntology`, das Modell mit diesen Daten zu instantiieren. Dann stellt er eine Anfrage an dieses Modell, deren Ergebnis die modellierten Daten des eigenen Fahrzeugs und der anderen in Reichweite befindlichen Fahrzeuge sind. Diese Daten dienen als Grundlage der Berechnung der Anwendungskomponente. Die Anwendung stellt mit Hilfe der `Inquiry`-Klasse und der Anfragesprache SPARQL[5] Anfragen an die durch die Ontologien modellierte und instantiierte Wissensbasis. Die vom Assistenten ermittelten Daten werden mit Hilfe der Klassen `Evaluation`, `FMElim` (die die Fourier-Motzkin-Elimination implementiert) und `CollisionLogic` (die die logischen

[5] `http://www.w3.org/TR/rdf-sparql-query/`

Umformungen in disjunktive Normalform und den Test auf Erfüllbarkeit enthält) für zwei Kollisionsberechnungen verwendet:

1. Kommt es zu einer Kollision bei gleichbleibenden Parametern?
2. Kommt es zu einer Kollision bei Beschleunigung?

Das Ergebnis (Kollisionspartner und Kollisionszeitpunkt) wird in einer Klasse `Result` gespeichert und es wird eine `OutputMessage` für die `InputOutput`-Klasse erzeugt. Unter Umständen werden dann Signale zum Abbremsen eines Fahrzeugs generiert.

- die *Ontologie* modelliert das Straßen- und Fahrzeugmodell aus Kapitel 6.2.2. Die Ontologie wird laufend mit neuen Daten aus der Simulationsumgebung instantiiert. Aus dem Modell kann der verwendete Reasoner (JENA[6]) neue Fakten und Daten ableiten. Zur Verwaltung der Ontologien wird die Klasse `ExtOntology` entwickelt – sie stellt unter anderem die Verbindung zwischen Reasoner und Ontologie her und bietet Methoden zur Instantiierung der Ontologien.

Die graphische Oberfläche (Abbildung 6.17) visualisiert den Simulationslauf, bei der Darstellung der Fahrzeuge haben die Farben die folgende Bedeutung:

- Blau: Kein Warnsignal vom Fahrerassistenten
- Gelb: Geringe Warnstufe
- Orange: Mittlere Warnstufe: Das Fahrzeug wird nicht weiter beschleunigt
- Rot: Höchste Warnstufe: eine Notbremsung erfolgt
- Grün: Das Notbremsmanöver ist abgeschlossen
- Magenta: Beschleunigungswarnung: Fahrzeug wird abgebremst.

[6] `http://jena.sourceforge.net/`

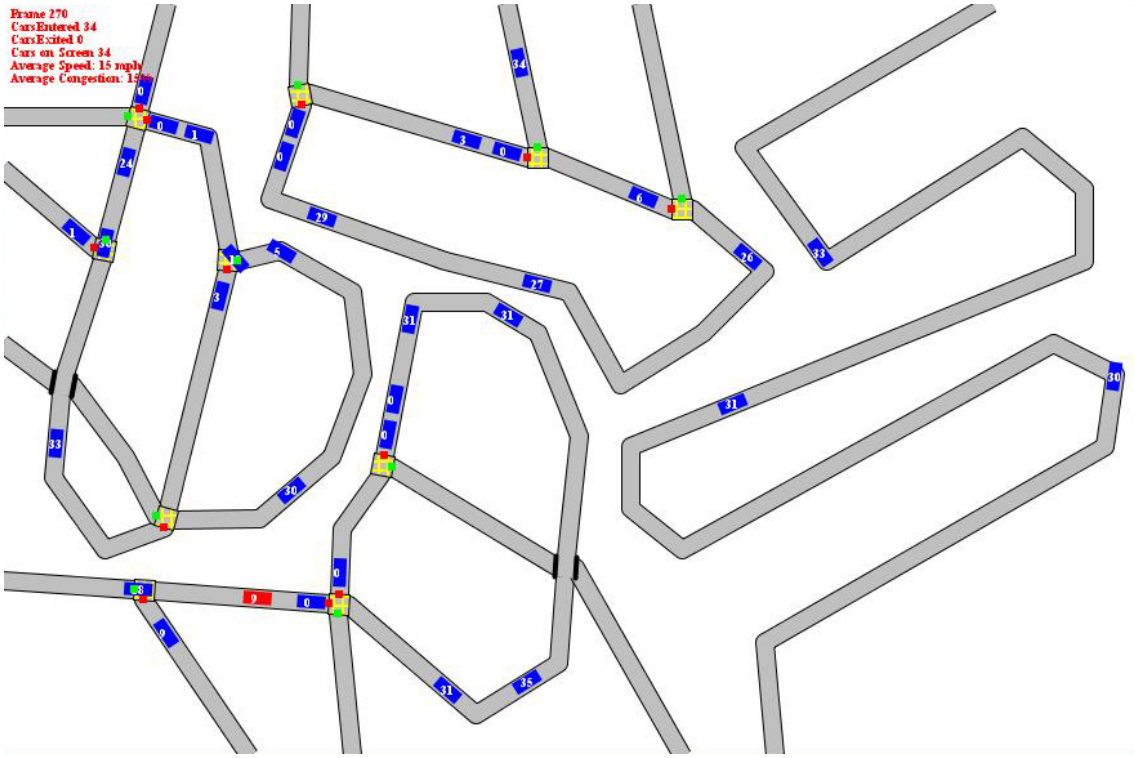


Abbildung 6.17: Grafische Oberfläche zur Simulation der Kollisionsvermeidung [Lut07]

Die Simulationen der Anwendung wurden auf einem AMD Athlon 64 3500+ Prozessor mit 1024 MB Arbeitsspeicher durchgeführt. Ein Simulationsschritt dauert 750 ms. Abbildung 6.18 zeigt einen Geschwindigkeitsvergleich des Simulators mit und ohne Fahrerassistent in der ersten Modellvariante ohne Straßennetz. Es fällt sofort auf, dass der Assistent signifikanten Mehraufwand erzeugt: ohne Assistenzanwendung ist die Simulation in der Laufzeit (y-Achse) eigentlich unabhängig von der Anzahl der simulierten Fahrzeuge (x-Achse) – zumindest für kleinere Szenarien mit unter 100 Fahrzeugen in der Simulation. Sobald der Assistent aktiviert wird, steigt der Aufwand, und zwar quadratisch: der Rechner muss schließlich die Kollisionsberechnungen für alle Fahrzeuge in der Simulation durchführen und nicht nur für ein einzelnes Fahrzeug. Da n Fahrzeuge $n-1$ potentielle Kollisionspartner haben, liegt die Anzahl der Kollisionsberechnungen in $O(n^2)$. Ein einzelnes Fahrzeug müsste nur $n-1$ Kollisionsberechnungen durchführen, der Aufwand würde mit steigender Anzahl der Fahrzeuge nur linear wachsen.
Wenn man versucht, den Effizienzgewinn zwischen Anwendungsversion eins (ohne Straßenontologie) und Version zwei (mit Straßenontologie) zu vergleichen, dann zeigt eine Gegenüberstellung der Laufzeit der unterschiedlichen Simulationsvarianten (Abbildung 6.19, dass die Laufzeit der komplexeren (!) Variante leicht gesunken ist. Auf den ersten Blick mag das widersprüchlich sein, jedoch ist die Anwendung durch das Zusatzwissen über das

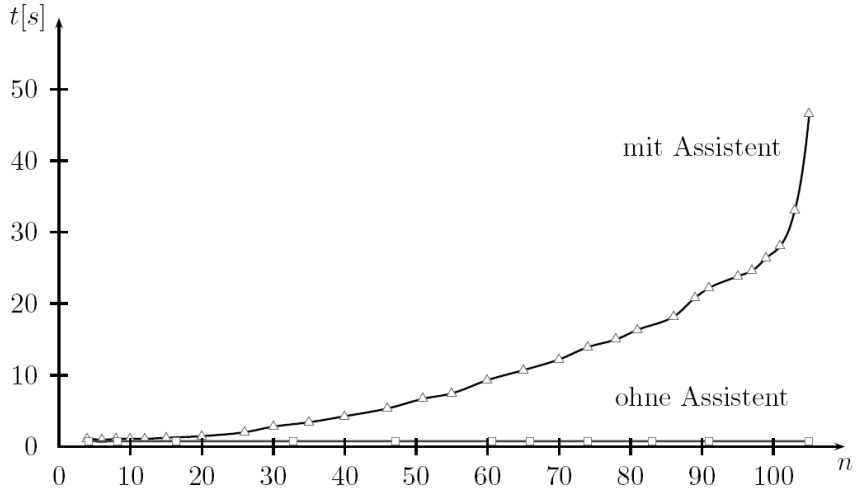


Abbildung 6.18: Laufzeitvergleich Simulation mit/ohne Fahrerassistent [Lut07]

Straßennetz in der Lage, festzustellen, ob zwei Fahrzeuge überhaupt potentielle Kollisionspartner sind, indem es überprüft, ob sie sich in denselben Zusammenhangskomponenten des Straßengraphen befindet. Wenn das nicht der Fall ist, können Kollisionsberechnungen vermieden werden. Der Gewinn ist also dadurch zu erklären, dass eine große Anzahl an Kollisionsberechnungen gar nicht erst durchgeführt wird. Abbildung 6.19 zeigt die Laufzeit der Simulation (y-Achse) im Zusammenhang mit der Anzahl der in der Simulation befindlichen Fahrzeuge (x-Achse). In Abbildung 6.20 kann die Anzahl der durchgeführten Kollisionsberechnungen (y-Achse) in Abhängigkeit der Anzahl der Fahrzeuge in der Simulation (x-Achse) abgelesen werden – sie hat sich ungefähr halbiert.

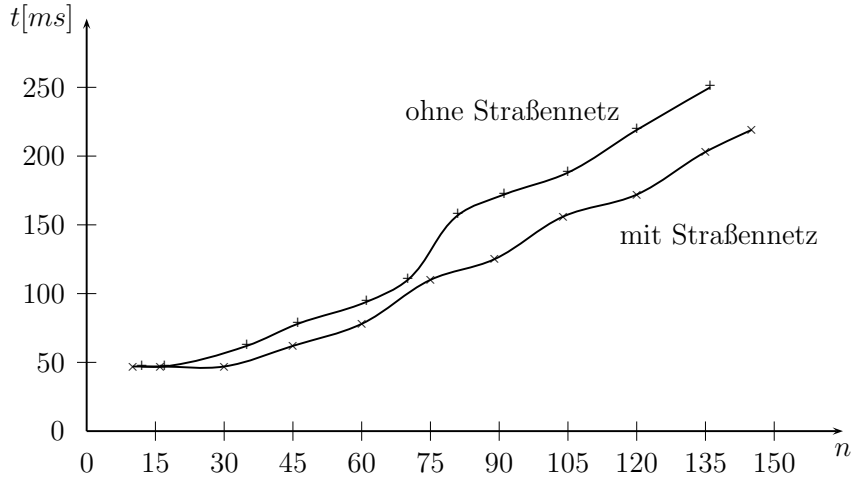


Abbildung 6.19: Laufzeitvergleich der beiden Kollisionsvermeidungs-Varianten [Lut07]

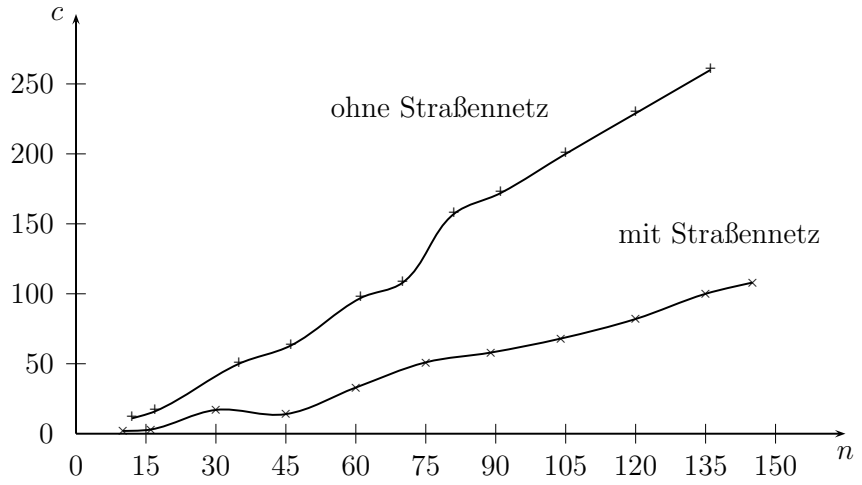


Abbildung 6.20: Vergleich der Anzahl durchgeführter Kollisionsberechnungen [Lut07]

Abbildung 6.19 zeigt auch, dass die Kollisionsberechnung schnell genug funktioniert, um Echtzeit-Ansprüchen genüge zu tun: bei ca. 140 Kollisionspartnern braucht die Anwendung in der komplexen Variante ca. 200 ms (Abbildung 6.19). Bei einer sehr hohen Geschwindigkeit von 200 km/h legt ein Fahrzeug in dieser Zeit 11,11 m Strecke zurück. Am Ende dieser Zeitspanne ist der Assistent mit seinen Berechnungen fertig und gibt u. U. eine Warnung an den Fahrer aus, der dann eine Vollbremsung einleitet. Bei einer angenommenen Reaktionszeit von ca. einer Sekunde legt ein Fahrzeug beim Durchführen einer Vollbremsung nach dem Erhalt der Warnung mit einer Bremsverzögerung von $-8\frac{m}{s^2}$ ca. 260 m zurück, in der Summe also ca. 271,11 m. Mit der in [TMSH06] angenommenen Empfangswahrscheinlichkeit von ca. 75% in 300 m Entfernung kann also noch rechtzeitig gewarnt werden.

6.2.4 Zusammenfassung

Die Anwendung hat gut gezeigt, wie flexibel Ontologien im Hinblick auf ihre Erweiterbarkeit sind: die Fahrzeugontologie der zweiten Variante wurde einfach durch Hinzufügen einer neuen Eigenschaft zur Ontologie der ersten Variante gewonnen, die Anwendung blieb dabei vollkommen unverändert. Selbst beim Hinzufügen einer komplett neuen Straßenontologie wurden keine Änderungen an der Anwendung nötig. Das Einbeziehen von Basismodellen ist einfach realisierbar, indem man eigene anwendungsspezifische Konzepte mit anderen aus Basisontologien in Beziehung setzt, wie z. B. die Positionsangaben für Fahrzeuge oder Straßen. Der Ansatz zeigt auch, dass die Verwendung komplexerer Modelle nicht zwangsläufig zu höheren Laufzeiten führen muss, da man das neue Zusatzwissen gewinnbringend einsetzen kann – ein weiterer Vorteil ontologischer Modelle. Komponenten wie JENA und die darin eingebauten Reasoner machen das in den Ontologien gespeicherte Wissen leicht

zugänglich und handhabbar, und das auch bei noch vertretbaren Laufzeiten. Insgesamt haben sich Ontologien bei dieser Anwendung als ausdrucksmächtiges, offenes, integrierbares und erweiterbares Modellierungswerkzeug erwiesen. Grundsätzlich sind sie hier von Vorteil, weil an einer solchen Kollisionsvermeidungsanwendung unter realen Bedingungen mehrere Hersteller beteiligt wären, die die gleichen Sensor- und Kontextkonzepte unterschiedlich realisieren würden. Zur Kooperation merherer Hersteller ist aber ein gemeinsames Verständnis dieser Konzepte nötig. Ontologien können das bieten.

6.3 Kontextbasierte Adressierung und kontextbasiertes Routing

Während bei „klassischen“ verdrahteten Netzen die Fluktuationsrate der Netzteilnehmer doch recht niedrig ist und somit der Adressat einer Kommunikation leichter zu ermitteln und zu erreichen ist, nehmen die solche Probleme bei drahtlosen Netzen zu: Kommunikationspartner wechseln oft die Netze, erhalten dann u. U. neue Adressen oder verlieren ihre Konnektivität ganz. Im Vergleich zu mobilen Nutzern mit Notebooks wird das Problem bei automobilen ad-hoc Netzwerken durch die hohe Mobilität und Geschwindigkeit der Knoten noch weiter verstärkt. Auf der anderen Seite machen VANET-Anwendungen mit ihren veränderten Anforderungen (vgl. Kapitel 3) eine dedizierte 1-zu-1-Kommunikation nicht mehr dringend notwendig: vielmehr werden Methoden zur Gruppenkommunikation gebraucht, d. h. ein Teilnehmer möchte eine Nachricht (z. B. eine Warnnachricht) an eine bestimmte Gruppe von Empfängern senden. Problematisch bleibt jedoch nach wie vor die Adressen der Empfänger oder der Empfängergruppe festzulegen: sie kann u. U. dem Sender im Voraus gar nicht bekannt sein und zusätzlich kann sie sich verändern, z. B. dadurch, dass Fahrzeuge an der mitzuteilenden Gefahrenstelle vorbeifahren oder ihren Fahrtweg so verändern, dass diese nicht mehr auf der Route liegt. Die Nachricht wird in solchen Fällen uninteressant und müsste nicht mehr zugestellt werden. An diesem Beispiel lässt sich gut erkennen, dass der Sender zwar die Empfänger nicht kennen kann, aber doch eine „Ahnung“ davon hat, in welchem Kontext sie sich befinden müssen: nämlich in der Nähe der Gefahrenstelle oder auf dem Weg dorthin.

Die in diesem Abschnitt vorgestellte Anwendung verwendet Kontextinformationen als Adresse und als Zusatzwissen für die Routingentscheidung. Dieses Zusatzwissen wird als Ontologie modelliert, genaueres findet sich in [Mai08, EM09, EM08].

6.3.1 Szenario

Für den Test des kontextbasierten Adressierens und Routings wird ein Windböenwarner gewählt: an einem windigen Tag fährt ein Fahrzeug über die Autobahn. Gerade an Brücken, Tunnelausfahrten oder Hügelkuppen kann starker Seitenwind auftreten und das Fahrzeug je nach gefahrener Geschwindigkeit aus der Spur drücken. Um den Fahrer vor solchen Gefahrensituationen aufmerksam zu machen, soll eine Warnung an andere sich auf die Gefahrenstelle hin bewegende Fahrzeuge zugestellt werden, die sich zudem in einem gewissen Umkreis um das Ereignis befinden. Zur Erkennung einer Windböe werden die folgenden Kontextdaten verwendet:

1. Aus der *Querbeschleunigung* eines Fahrzeugs lässt sich ermitteln, ob ein Fahrzeug zur Seite abgelenkt wird. In modernen Fahrzeugen ist diese Informationen verfügbar, da sie such für ESP-Systeme verwendet wird.

2. Der *Lenkeinschlag* ist insofern wichtig, da er verwendet werden kann, um willkürlich vom Fahrer induzierte Seitwärtsbewegungen von unwillkürlichen (durch Seitenwind verursachte) zu unterscheiden.

3. die *Geschwindigkeit* wird nicht unmittelbar zur Erkennung der Situation verwendet. Da aber die Gefahr umso höher ist, je schneller gefahren wird, wird die Warnung nur dann ausgelöst, wenn die Geschwindigkeit über einer bestimmten Schwelle liegt.

Eine Warnung durch ein Fahrzeug wird genau dann ausgelöst, wenn eine signifikante Seitwärtsbewegung ohne zugehörigen Lenkradeinschlag detektiert wird und gleichzeitig die Geschwindigkeit über einer bestimmten Schwelle liegt. In diesem Falle wird eine Windböenwarnung mit den Koordinaten des Auftretens an weitere Fahrzeuge, die sich auf derselben Route und in der Nähe befinden, versendet. Adressaten der Nachricht sind damit alle Fahrzeuge, die potentiell in dieselbe Gefahrensituation geraten könnten. Um die Anzahl der gewarnten Fahrzeuge zu erhöhen, ist die Warnung für beide Fahrtrichtungen gültig. Außerdem wird die Warnung von den Fahrzeugen, die sie erhalten haben, wieder ausgesendet und durch eigene Messwerte ergänzt, wodurch sich die Anzahl der Messpunkte im Bereich der Warnung erhöht. Durch dieses Verfahren erhalten Fahrzeuge, die neu in den Gefahrenbereich eintreten, schon ein aktualisiertes Bild des gesamten Bereichs. Um das Ad-hoc-Netz nicht durch diese Broadcast-Meldungen zu überlasten, werden die Nachrichten nur unter bestimmten Bedingungen wiederholt, die jeder einzelne Knoten immer wieder überprüft. Dadurch verschiebt sich die Entscheidung, ob ein Paket erneut ausgesendet wird, auf den Zwischenknoten.

6.3.2 Konzept

Für das Szenario kommen zwei in OWL modellierte Ontologien zum Einsatz: eine modelliert die Warnung, eine weitere den Kreis der Adressaten.

6.3.2.1 Warnungsontologie

Die Warnung enthält primär die Messdaten, also Daten zum Lenkeinschlag, zur Geschwindigkeit und zur Querbeschleunigung, die zur Warnung führten. Wenn die Messwerte in einer bestimmten Zusammensetzung auftreten, dann wird eine neue Warnungsontologie instantiiert und die entsprechenden Werte werden in sie eingetragen. Die Instanz kann im Fahrzeug ausgewertet werden (um sie z. B. dem Fahrer anzuzeigen) und kann an andere Fahrzeuge verschickt werden.

Die einzelnen Messwerte werden als `rdfs:subPropertyOf` von `hatMesswert` als Datentyp-Eigenschaft `hatGeschwindigkeit`, `hatQuerbeschleunigung` und `hatLenkwinkel` modelliert. Listing 6.2 zeigt dies am Beispiel einer Geschwindigkeitsmessung. Die Definition von `hatMesswert` dient dazu, einen oder mehrere Messwerte an eine Instanz der Klasse `Messung` zu binden. Dies wird zur weiteren Auswertung durch den Reasoner benötigt.

```
<owl:DatatypeProperty rdf:about="#hatMesswert">
  <rdfs:domain rdf:resource="#Messung"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#hatGeschwindigkeit">
  <rdf:type rdf:resource="&owl;FunctionalProperty"/>
  <rdfs:subPropertyOf rdf:resource="#hatMesswert"/>
  <rdfs:range rdf:resource="&xsd;double"/>
</owl:DatatypeProperty>
```

Listing 6.2: Modellierung von Messwerten [Mai08]

Kurz gesagt verfügt eine `Messung` über eine Eigenschaft `hatMessung` und diese wiederum über Untereigenschaften (z. B. `hatGeschwindigkeit`). Geschwindigkeit ist in diesem Falle aus Gründen der Einfachheit als Double-Wert modelliert - eine genauere Modellierung hätte sich unter Verwendung des ASC-Modells ergeben, was aber im Hinblick auf die Ziele der Anwendung nicht unbedingt nötig ist.
Eine spezielle Warnung (in diesem Falle ein Windböenwarnung) ist als Unterklasse einer allgemeineren Klasse `Ereignis` modelliert, die zwingenderweise über mindestens eine Objekteigenschaft `hatMessung` (Listing 6.3) verfügen muss und somit eine `Messung` an ein `Ereignis` bindet. Der Wertebereich ist auf Instanzen des Typs `Messung` eingeschränkt.

Listing 6.4 zeigt die Modellierung der Klasse `Ereignis`.

```
<owl:ObjectProperty rdf:about="#hatMessung">
  <rdfs:domain rdf:resource="#Ereignis"/>
  <rdfs:range rdf:resource="#Messung"/>
</owl:ObjectProperty>
```

Listing 6.3: die Objekteigenschaft `hatMessung` [Mai08]

```
<owl:Class rdf:about="#Ereignis">
  <rdfs:subClassOf>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatMessung"/>
          <owl:someValuesFrom rdf:resource="#Messung"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatMessung"/>
          <owl:allValuesFrom rdf:resource="#Messung"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdfs:subClassOf>
  <owl:disjointWith rdf:resource="#Messung"/>
</owl:Class>
```

Listing 6.4: die Klasse `Ereignis` [Mai08]

Auf dieser Grundlage wird nun die Klasse `Seitenwind` (Listing 6.5) als Subklasse von `Ereignis` modelliert. Diese Klasse wird gleichzeitig auch als Warnung verwendet. Die Definition geschieht über eine Äquivalenzrelation (Schlüsselwort `owl:equivalentClass`): Sie definiert, dass die Klasse `Seitenwind` gleichwertig ist mit einer anonymen inneren Klasse, die aus der Schnittmenge dreier Messungen entsteht:

- eine `Messung` mit der Eigenschaft `hatGeschwindigkeit` ≥ 8.
- eine `Messung` mit der Eigenschaft `hatLenkwinkel` ≤ 5.
- eine `Messung` mit der Eigenschaft `hatQuerbeschleunigung` ≥ 10.

Die Werte sind willkürlich gewählt und entsprechen nicht unbedingt einem realen Szenario. Zum Schluss wird noch modelliert, dass `Seitenwind` eine Subklasse von `Ereignis` ist und mit der hier nicht näher spezifizierten Klasse `Glatteis` nicht identisch ist.

```
<owl:Class rdf:about="#Seitenwind">
  <owl:equivalentClass>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatMessung"/>
          <owl:someValuesFrom>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hatGeschwindigkeit"/>
              <owl:allValuesFrom>
                <rdf:Description>
                  <rdf:type rdf:resource="&rdfs;Datatype"/>
                  <owl:onDatatype rdf:resource="&xsd;double"/>
                  <owl:withRestrictions rdf:parseType="Collection">
                    <rdf:Description>
                      <xsd:minInclusive rdf:datatype="&xsd;int">8
                      </xsd:minInclusive>
                    </rdf:Description>
                  </owl:withRestrictions>
                </rdf:Description>
              </owl:allValuesFrom>
            </owl:Restriction>
          </owl:someValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatMessung"/>
          <owl:someValuesFrom>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hatLenkwinkel"/>
              <owl:allValuesFrom>
                <rdf:Description>
                  <rdf:type rdf:resource="&rdfs;Datatype"/>
                  <owl:onDatatype rdf:resource="&xsd;double"/>
                  <owl:withRestrictions rdf:parseType="Collection">
                    <rdf:Description>
                      <xsd:maxInclusive rdf:datatype="&xsd;int">5
                      </xsd:maxInclusive>
                    </rdf:Description>
                  </owl:withRestrictions>
                </rdf:Description>
              </owl:allValuesFrom>
            </owl:Restriction>
          </owl:someValuesFrom>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hatMessung"/>
          <owl:someValuesFrom>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hatQuerbeschleunigung"/>
              <owl:allValuesFrom>
                <rdf:Description>
                  <rdf:type rdf:resource="&rdfs;Datatype"/>
                  <owl:onDatatype rdf:resource="&xsd;double"/>
                  <owl:withRestrictions rdf:parseType="Collection">
                    <rdf:Description>
```

```
                    <xsd:minInclusive rdf:datatype="&xsd;int">10
                    </xsd:minInclusive>
                  </rdf:Description>
                </owl:withRestrictions>
              </rdf:Description>
            </owl:allValuesFrom>
          </owl:Restriction>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
 </owl:equivalentClass>
 <rdfs:subClassOf rdf:resource="#Ereignis"/>
</owl:Class>
```

Listing 6.5: die Klasse `Seitenwind`

Ein Reasoner kann nun mit einem solchen Modell aus einer Instanz, die über die geforderte Messwert-Belegung verfügt, ableiten, dass es sich um ein `Seitenwind-Ereignis` handeln muss und kann u. U. eine Warnung an den Fahrer veranlassen.

6.3.2.2 Adressontologie

Die ausgesendeten Warnungen sind nicht für jedes Fahrzeug relevant. Damit nicht jeder Netzwerkknoten alle empfangenen Pakete auswerten muss, werden die versendeten Pakete zusätzlich mit einer „Adressontologie“ ausgestattet, die Aufschlüsse darüber geben soll, für wen eine Information von Nutzen sein könnte. Diese Adressontologie enthält den Kontext, in dem sich ein Fahrzeug befinden sollte, um als Adressat für die Information gelten zu können. Ob sich ein Knoten an diese Hinweise hält, bleibt letztlich ihm überlassen - er kann nach wie vor die ganze Warnungsontologie auswerten, was aber viel umfangreicher ist.

Für das vorliegende Szenario wird eine Adressierung anhand der zukünftigen Route des Fahrzeugs und der Entfernung vom übermittelten Ereignis konzipiert: eine Warnung wird an all diejenigen Fahrzeuge verschickt, die sich auf dem Weg zur Gefahrenstelle befinden. Die Route wird durch die von ihr verwendeten Straßenabschnitte identifiziert, eine Information, die in den meisten modernen Navigationsgeräten vorliegt. Da Fahrzeuge im Verlauf der Zeit ihre Route verändern und anpassen können, verändert sich der Kreis der Adressaten genauso – durch den ontologischen Ansatz kann dieser Dynamik Rechnung getragen werden.

Die zukünftige Route wird in OWL als Liste von Streckenabschnittsnummern dargestellt werden: ein Attribut `hatStrSegmentID` nimmt die ID eines Streckenabschnitts als Argument und ist entsprechend mehrfach vorhanden. Ein Beispiel für eine Route findet sich in Listing 6.6.

```
<owl:DatatypeProperty rdf:about="#hatStrSegmentID">
  <rdfs:range rdf:resource="&xsd;int"/>
</owl:DatatypeProperty >
<rdf:Description rdf:about="#aktuelleRoute">
  hat_StrSegmentID rdf:datatype="&xsd;int">276</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">265</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">278</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">264</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">256</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">286</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">273</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">258</hatStrSegmentID>
  hat_StrSegmentID rdf:datatype="&xsd;int">275</hatStrSegmentID>
</rdf:Description>
```

Listing 6.6: Eine aus Streckenabschnitten bestehende Route [Mai08]

6.3.2.3 Routing

Als „Routing“-Verfahren kommt das sogenannte *Selektive Fluten* zum Einsatz, eine Art Broadcast-Routing: dadurch, dass die Funkschnittstelle mit einem gemeinsamen Medium arbeitet und alle Stationen – auch nicht explizit adressierte – eine Übertragung einer Station in Funkreichweite mithören können, kann man auch dem Empfänger der Nachricht die Entscheidung überlassen, ob er eine Nachricht erneut aussenden will oder nicht. Es kommt also zu keinem expliziten Routenaufbau (deshalb „Routing“ in Anführungszeichen), sondern eher zu einer Art Informationsverbreitung entlang relevanter Stationen, da die Entscheidung über eine erneute Aussendung davon abhängt, ob eine Station von der Information betroffen ist: im vorliegenden Falle heißt das, ob sie auf der entsprechenden Fahrtroute liegt oder nicht. Um zusätzlich redundante Übertragungen zu vermeiden, werden generell folgende Kriterien mit berücksichtigt:

- Eine bereits zum zweiten Mal empfangene Nachricht wird nicht wiederholt, da sie schon bekannt ist. Zur Identifikation einer Nachricht wird ein Merkmal, bestehend aus einem Hashwert der Nachricht und einem Zeitstempel für den Messwert, erzeugt. Dadurch können Duplikate erkannt werden.

- Die Warnung ist interessant für Fahrzeuge, die die Gefahrenstelle wahrscheinlich passieren werden. Dies kann mit Hilfe des Navigationsgeräts entschieden werden

- Da sich Routen ändern können, sollten auch Fahrzeuge berücksichtigt werden, die zwar nicht den exakten Streckenabschnitt auf der zukünftigen Route haben, sich aber zumindest auf ihn zu bewegen. Es besteht z. B. durch falsche Abbiegemanöver die

Wahrscheinlichkeit, dass ein Route so abgeändert wird, dass der gefährdete Abschnitt doch auf der Strecke liegt.

Die Bedingungen zum Weiterleiten von Nachrichten werden wieder in OWL implementiert. Die Auswertung erfolgt, wie bereits bekannt, als Matching von Instanzen zu Klassen (vgl. `Seitenwind`-Erkennung) mit Hilfe der Definition äquivalenter Klassen, denen vom Reasoner die entsprechenden Instanzen zugeordnet werden. Ein Reasoner kann durch die Auswertung der Ontologie entscheiden, ob sich ein Punkt, spezifiziert durch seine Längen- und Breitenattribute, innerhalb des jeweiligen Wertebereichs befindet und veranlasst dann die Weiterleitung.

Um die Anzahl der Nachrichten im Netzwerk einzudämmen, darf eine Nachricht höchstens einmal pro Fahrzeug versendet werden. Zusätzlich wird der Nachrichtenspeicher als Ringpuffer, der bei Erreichen seiner Kapazitätsgrenze die ältesten Nachrichten wieder überschreibt, realisiert. Die Implementierung dieses Verfahrens für Selektives Fluten ist auf ISO/OSI-Schicht drei angesiedelt, was dazu führt, dass jedes Paket sowohl die Adressauswertung (zur Entscheidung, ob das Paket für die eigene Station relevant ist) als auch die Routingimplementierung (zur Entscheidung, ob das Paket wieder erneut ausgesendet werden soll) durchlaufen muss. Die Routingkomponente speichert neu empfangene, bisher noch unbekannte Nachrichten und plant eine erneute Aussendung nach einem (zufällig beeinflussten) Warteintervall. Die Wartezeit wird verwendet, damit ein durch zwei Stationen gleichzeitig empfangenes Paket nicht sofort erneut durch beide versendet wird, da dies zu einer Kollision führen würde. Wird während der Wartezeit ein neues Paket empfangen, durchläuft auch dieses denselben zweistufigen Entscheidungsprozess. Wird es als Dublette einer noch in der Warteschlange befindliches Pakets erkannt, so wird es gelöscht.

6.3.3 Implementierung und Evaluation

Implementiert wird das System in Java unter Verwendung des OWL-API-Frameworks [HBN07] zur Verwaltung und Verarbeitung der Ontologien. Als Netzwerksimulator kommt JiST/SWANS[7] zum Einsatz. JiST (Java in Simulation Time) ist ein Framework für die diskrete und ereignisgesteuerte Simulation von Java-Programmen. Es verändert fertig kompilierten Java-Bytecode, um die eigene Simulationssemantik in das Programm zu integrieren: so kann der Original-Quellcode unverändert bleiben. SWANS (scalabale wireless network simulator) baut auf dem JiST-Framework auf und kann zum Simulieren von drahtlosen Netzwerken verwendet werden. Die Architektur von SWANS orientiert sich dabei am ISO/OSI-Modell und bringt schon eine große Anzahl an implementierten Netzwerkprotokollen für die unteren Schichten mit. SWANS verfügt auch über eine Erweiterung, die auf die Simulation von Fahrzeug-zu-Fahrzeug-Kommunikation zugeschnitten ist: SWANS++

[7] `http://jist.ece.cornell.edu/index.html`

integriert STRAW, das Street Random Waypoint Modell, mit dessen Hilfe digitale Karten als Grundlage für die Bewegung von Knoten verwendet werden können. Das Straßennetz muss im TIGER[8]-Format vorliegen. Für die Simulation ist STRAW ausreichend, da die Bewegung nicht ganz zufällig geschieht, sondern nur Start- und Endpunkt einer Route zufällig ermittelt werden. Der kürzeste Weg zum Endpunkt wird dann mit dem A*-Algorithmus berechnet. Die zu fahrende Route liegt also schon beim Beginn der Fahrt fest, was für die Adressauswertung notwendig ist. Auf Schicht eins kommt das Shadowing-Pathloss-Modell zum Einsatz, auf Schicht zwei der Medienzugriff nach IEEE802.11.

Die Auswertung der Ontologien wird auf Schicht drei vorgenommen, wozu jeweils eine Instanz des Pellet[9]-Reasoners verwendet wird. Die Abarbeitung einer Nachricht wird in Abbildung 6.21 schematisch skizziert und zeigt den Weg einer Nachricht oberhalb von Schicht zwei: empfangene Nachrichten folgen grünen Pfeilen, zu sendende roten.

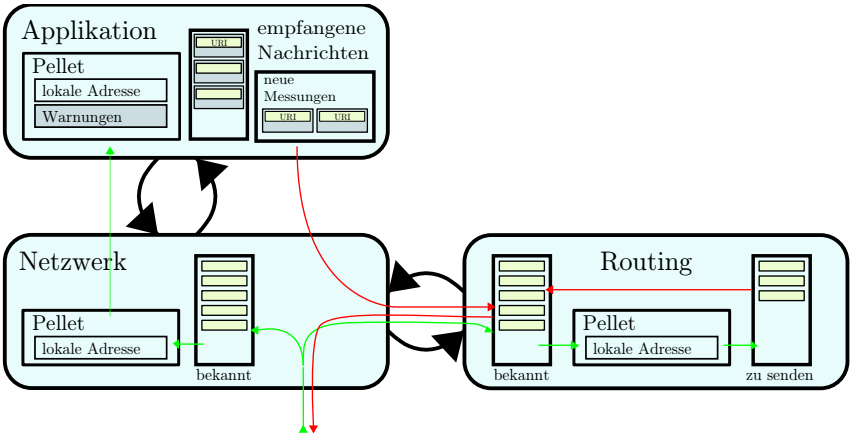


Abbildung 6.21: Datenfluss bei kontextbasierter Adressierung und kontextbasiertem Routing [Mai08]

Wenn die Netzwerkkomponente der Schicht drei ein Paket der darunterliegenden Schicht erhält, leitet sie eine Kopie des Pakets an die Routingkomponente weiter. Falls es sich um ein noch unbekanntes Paket handelt, wird der Pellet-Reasoner damit beauftragt, herauszufinden, ob sich ein in der Adressontologie gespeicherter Straßenabschnitt mit einem Abschnitt der eigenen zukünftigen Route überschneidet. Falls dies der Fall ist, so ist die Warnung relevant und wird dem Fahrer angezeigt und wird an die Applikationsschicht weitergeleitet. Falls nicht, so wird sie verworfen. Die Routingkomponente verwaltet ihre Kopie unabhängig davon, ob sie als relevant eingestuft wurde und führt ihre eigene Überprüfung auf Dubletten und Einhaltung der Kriterien für die Wiederaussendung durch.

[8] `http://www.census.gov/geo/www/tiger/`
[9] `http://pellet.owldl.com/`

Den Simulationen wurde ein Ausschnitt des Straßennetzes von Boston zugrunde gelegt, die Netzwerkparameter entsprechen den Standardvorgaben der SWANS++ Simulationsumgebung:

- Sendefrequenz 2,4 GHz
- Übertragungsrate 2 MBit/s
- Sendeleistung 15 dBm
- Empfangsempfindlichkeit -91 dBm

In der Simulation werden einige Einschränkungen vorgenommen, um das Auftreten von Windböen zu steuern: SWANS++ kann keine Kurvenfahrten simulieren und somit keinen Lenkwinkeleinschlag erzeugen, wie er zur Detektion einer Windöensituation notwendig wäre. Zu diesem Zweck wird angenommen, dass während der gesamten Simulation jede vorgenommene Lenkwinkelmessung einen Einschlag von 2 Grad aufweist. Dieser Wert liegt unterhalb der geforderten Grenze von 5 Grad, Windböen können somit erkannt werden. Die Querbeschleunigung liegt auf einem als gefährlich markierten Teilstück mit einer Wahrscheinlichkeit von 40% über 10. Damit führt nicht jeder Messvorgang automatisch zu einer Warnung. Eine weitere von SWANS++ simulierte Größe ist die Geschwindigkeit, die im Mittel über den Verlauf der Simulation gerechnet bei 7,9 liegt, also knapp unter der Schwelle von 8, die für eine Warnung notwendig ist. Das nur knappe Unterschreiten der Schwelle führt dazu, dass die Grenze recht häufig überschritten wird, so dass es zu zahlreichen Warnungen kommt. Die Visualisierung der Simulation erfolgt über SWANS++.

Die Simulation startet mit einer zufälligen Verteilung von Fahrzeugen im Straßennetz. Die Farben in den Abbildungen 6.22 bis 6.25 sind wie folgt zu interpretieren:

roter Rahmen	Nachricht erzeugt und versendet
schwarzer Rahmen	Nachricht empfangen und ignoriert
grauer Rahmen	Nachricht empfangen und gespeichert für späteres Wiederaussenden
grüner Rahmen	Nachricht empfangen und verarbeitet (für relevant erachtet)
gelbes Fahrzeug	Fahrzeug wurde erfolgreich gewarnt
orangenes Straßenstück	Gefahrenstelle

Sobald ein Fahrzeug eine Windböe registriert, wird eine Meldung an umliegende Fahrzeuge weitergeleitet. Abbildung 6.22 rechts zeigt das an Fahrzeug 8 (roter Rahmen). Die Nachricht wird von zwei Fahrzeugen empfangen, aber nicht ausgewertet, da ihre eigene Route sie nicht an der Gefahrenstelle vorbeiführen wird. Weil sie sich aber in der Nähe

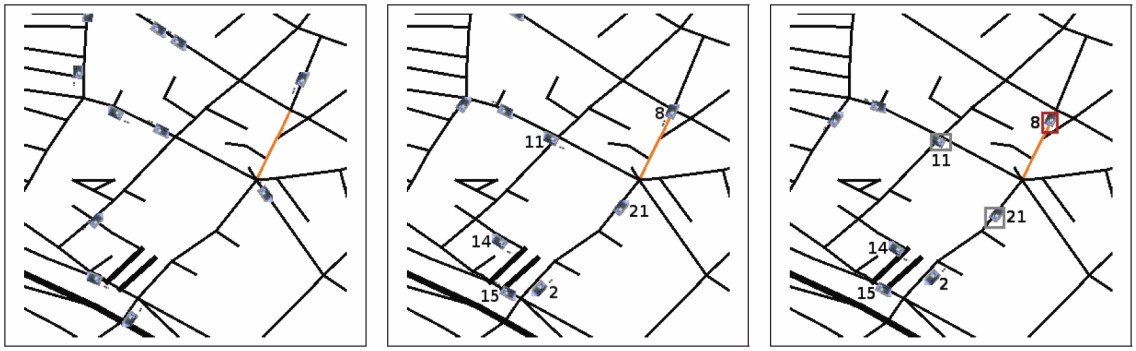


Abbildung 6.22: Visualisierung der Simulation – Simulationsschritt 1 [Mai08]

der Gefahrenstelle befinden, speichern sie die Nachricht für ein späteres Wiederaussenden (Fahrzeuge 11 und 21, grauer Rahmen).

Kurz darauf erzeugt Fahrzeug 8 erneut eine Nachricht, die aber aufgrund veränderter Funkbedingungen nur noch von Fahrzeug 11 empfangen wird (Abbildung 6.23, links). Fahrzeug 21 beginnt mit dem Wiederaussenden der ersten Nachricht (Abbildung 6.23, Mitte, roter Rahmen), die von den Fahrzeugen 2, 11 und 15 empfangen wird und von Fahrzeug 15 auch zur Weiterleitung gespeichert wird. Interessant ist, dass der Sender – Fahrzeug 21 – die Nachricht selbst nicht verarbeitet, obwohl es sich näher an der Gefahrenstelle befindet als Fahrzeug 2, welches die Nachricht verarbeitet (Abbildung 6.23, Mitte, grüner Rahmen). Das liegt daran, dass sich Fahrzeug 21 von der Windböe entfernt, während Fahrzeug 2 noch darauf zu fährt.

Abbildung 6.23: Visualisierung der Simulation – Simulationsschritt 2 [Mai08]

Bisher wurde noch kein Fahrzeug erfolgreich gewarnt, da die Nachrichten bei zu niedrigen Geschwindigkeiten versendet wurden (die Schwelle von 8 muss für eine Warnung überschritten werden), erst im weiteren Verlauf der Simulation stellt sich eine solche Situation

ein (Abbildung 6.24, rechts, gelb gefärbte Fahrzeuge 8 und 2).

Abbildung 6.24: Visualisierung der Simulation – Simulationsschritt 3 [Mai08]

Ohne Nachrichtenquellen beruhigt sich das Netz wieder, bis sich Fahrzeug 2 erneut in den Gefahrenabschnitt begibt und erneut Nachrichten erzeugt (Abbildung 6.25, links, roter Rahmen). Schließlich verlässt Fahrzeug 8 den Senderadius und empfängt auch keine Nachrichten von anderen Fahrzeugen mehr. Am Ende verliert es das „Gedächtnis“ über die Situation (Abbildung 6.25, rechts, Graufärbung von Fahrzeug 8).

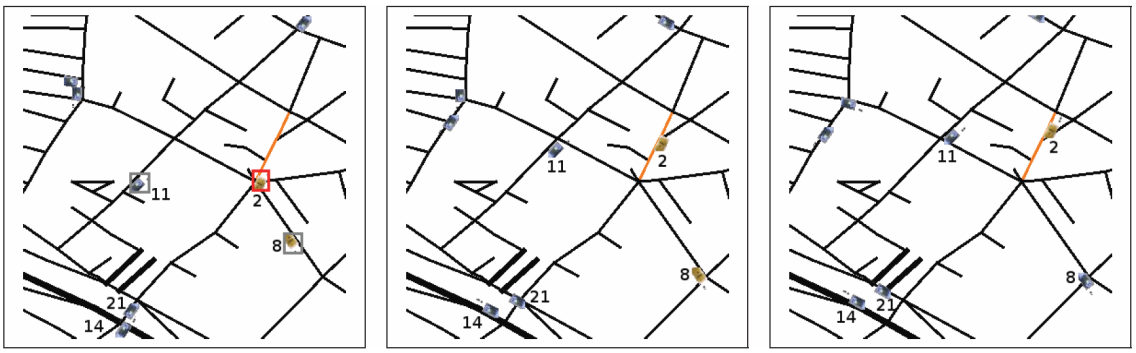


Abbildung 6.25: Visualisierung der Simulation – Simulationsschritt 4 [Mai08]

Um die Effizienz des kontextbasierten Adressierens und selektiven Flutens zu messen, wurde der Ansatz mit einer auf simplem Fluten und Broadcast basierenden Lösung verglichen. Zu erwarten ist beim kontextbasierten Adressieren ein leichtes Nachlassen des Warnungserfolges, da nicht mehr alle Fahrzeuge erfasst werden, sondern nur noch ausgewählte. Gleichzeitig sollte aber der Nachrichtenaufwand reduziert sein. Abbildung 6.26 zeigt die Anzahl der während der Simulation erzeugten Nachrichten (y-Achse) in Abhängigkeit von der Anzahl der sich in der Simulation befindlichen Fahrzeuge (x-Achse). Auffallend ist, dass die Anzahl der Nachrichten beim selektiven Fluten im Vergleich zum einfachen Fluten auf ca. die Hälfte reduziert ist.

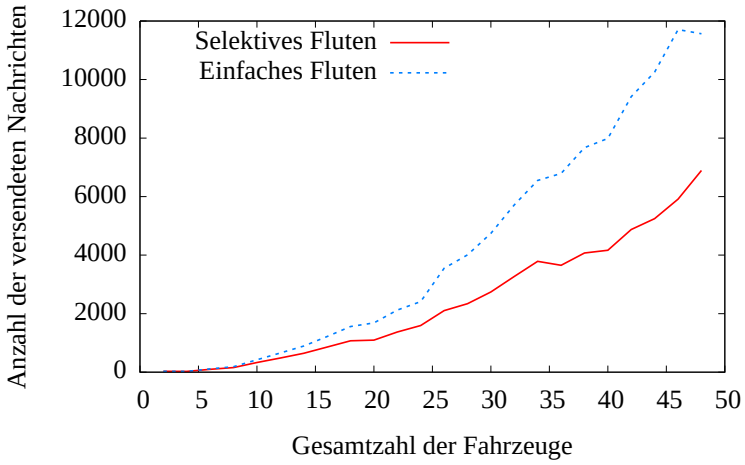


Abbildung 6.26: Anzahl der während der Simulation erzeugten Nachrichten [Mai08]

Wenn man nun den Warnungserfolg der beiden Verfahren miteinander vergleicht, dann ergibt sich Abbildung 6.27. Auf der x-Achse ist wieder die Anzahl der in Simulation befindlichen Fahrzeuge angetragen, auf der y-Achse die Quote der erfolgreich gewarnten Fahrzeuge (im Vergleich zur Gesamtzahl). Deutlich zu sehen ist, dass der „Gewinn“ des simplen Flutens bei maximal 15% liegt und somit in keinem Verhältnis zum um ca. 100% erhöhten Nachrichtenmehraufwand. Hier wird deutlich, dass sich die Einschränkung der Nachrichtenkommunikation auf beteiligte Stationen lohnt, da dadurch eine starke Entlastung des Netzes erreicht werden kann und somit Kapazität für andere Informationen frei wird. Durch die Entlastung sinkt auch die mittlere Wartezeit beim Zugriff auf den Kanal, da weniger Stationen um ihn konkurrieren.

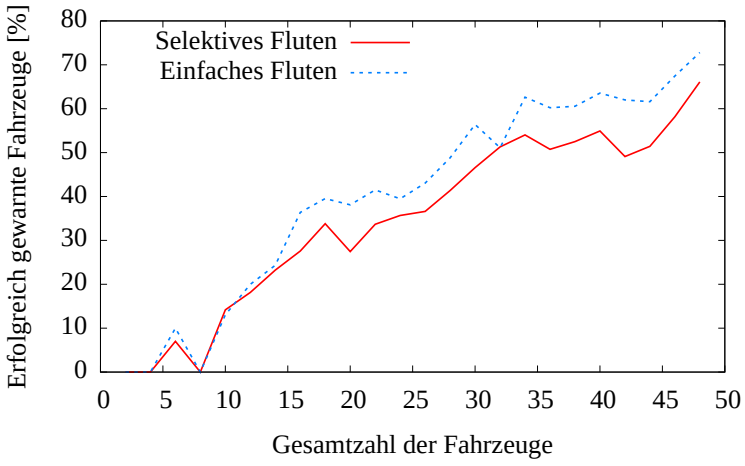


Abbildung 6.27: Warnungserfolg [Mai08]

6.3.4 Zusammenfassung

Generell zeigt der Ansatz, dass Kontextinformationen in VANETs als Adressen eingesetzt werden können, was den Anforderungen nach einer flexiblen und dynamischen Gruppenkommunikation eher gerecht wird. Ein weiterer Vorteil besteht in der Selbstorganisation des Netzes, die Adressaten und „Routen“ an sich verändernde Kontextkriterien laufend anpasst. Ontologien sind auch hier ein geeignetes Mittel, um das dafür nötige Anwendungswissen den unteren Schichten des ISO/OSI-Protokollstapels zugänglich zu machen. Informationen können dadurch orts- und zeitbezogen verarbeitet werden. Die Verfahren zur Erkennung der Windböensituation haben gezeigt, wie mit Hilfe von Reasonern aus einfachen Messdaten ein höherwertiger Kontext – nämlich die Erkennung eines Ereignisses – erzeugt werden kann. Die Offenheit des ontologischen Ansatzes ermöglicht es, jederzeit neue Kontextkriterien zur Adressierung heranzuziehen, in den unteren Schichten weitere Daten für die Routingentscheidung hinzuzuziehen oder neue und feinere „Schablonen“ für die Situationserkennung zu definieren.

Mit einem noch zu definierenden Modell für Zustand des Netzes wäre es auch möglich, Wiederholraten, Sendeleistung (und damit die Funkreichweite) an die Belastungssituation, die Netzwerkdichte und andere Parameter anzupassen, so dass eine noch bessere Auslastung erreicht wird.

7 Evaluation

Die Evaluation wurde schon auf Anwendungsebene durchgeführt, deshalb findet hier nur noch ein Abgleich mit den Anforderungen an das Kontextmodell (siehe Kapitel 3.5) statt:

Ausdrucksmächtigkeit Durch die Verwendung von Ontologien als Modellierungssprache steht ein relativ ausdrucksmächtiges Werkzeug zur Modellierung von Kontextkonzepten und deren Beziehungen untereinander zur Verfügung. Im Gegensatz zu den in Kapitel 4.1 vorgestellten Möglichkeiten, verfügt der in dieser Arbeit verfolgte Ansatz über eine erweiterte maschinenverarbeitbare Semantik. So wäre es z. B. XML nicht möglich gewesen, semantische Beziehungen zu modellieren. Während für Menschen Tags sofort mit Bedeutung aufgeladen sind, stellen sie für Rechner nur Zeichenketten ohne weitere Bedeutung dar. Auf semantischer Ebene besteht zwischen

```
Der <Modell>XKR</Modell> ist ein Fahrzeugmodell der Firma
<Hersteller>Jaguar</Hersteller>
```

und

```
Der <Qwert>XKR</Qwert> ist ein Fahrzeugmodell der Firma
<Asdfg>Jaguar</Asdfg>
```

für einen Rechner kein Unterschied. Für einen Menschen hingegen tragen die Tags im ersten Beispiel eine Bedeutung. Auch einfache Konzepthierarchien wie sie z. B. mit RDF und RDFS hätten erstellt werden können, reichen für komplexe Konzepte nicht aus: gravierendster Mangel von RDFS ist, dass es nicht möglich ist, auszudrücken, dass ein Sachverhalt *nicht* gilt (vgl. [HKRS08]). Angewendet auf die vorliegende Arbeit, wäre es z.Ḃ. nicht möglich gewesen auszudrücken, dass die Mengen `Eigenes_Fahrzeug` und `Andere_Fahrzeuge` der Kollisionsvermeidungsanwendung (siehe Kapitel 6.2, Abbildung 6.13) disjunkt sind – dafür ist die erweiterte Semantik von OWL nötig.

Zusätzlich bietet OWL noch den Vorteil, in mehreren Ausprägungen mit unterschiedlicher Mächtigkeit vorzuliegen (OWL Lite, OWL DL und OWL Full) und damit durch besseren „Zuschnitt“ auf einen Anwendungsbereich auch das automatische Reasoning auf Wissensbasen in OWL effizienter zu gestalten.

Umfassendes Basismodell Die in Kapitel 5 vorgestellten KOMODE-Basismodelle bilden die Grundlage für kontextsensitive Anwendungen in VANETs. Sie modellieren alle relevanten Daten, die bisher in Fahrzeugen zur Verfügung stehen. Durch die Verwendung des ASC-Modells ist es möglich, dasselbe Kontextkonzept durch unterschiedliche Sensoren zu erfassen – wer der Hersteller des Sensors war, ist auf Aspekt-Ebene nicht mehr relevant, ebenso wenig wie technische Details der Kontexterfassung. Eine Anwendung kann mit den im Aspekt modellierten Informationen arbeiten. Gleichzeitig stellt das KOMODE-Modell Definitionen für die im VANET-Bereich wichtigsten Konzepte, nämlich zeitlichen und örtlichen Kontext sowie Umweltwahrnehmungen zur Verfügung. Durch die Modellierung der Qualität einer Information innerhalb des ASC-Modells als erneute Kontextinformation werden Anwendungen in die Lage versetzt, die Güte einer Informationen in ihre Berechnungen mit einzubeziehen. So muss die zu einem Aspekt gehörende Kontextinformation nur mit einer weiteren Kontextinformation (z. B. dem im ASC-Modell verwendeten `meanError`) versehen werden, damit die Qualität einer Information erfasst werden kann. Eine Anwendung kann dann so realisiert werden, dass sie eine Kontextinformation nur dann berücksichtigt, wenn ihre Qualität ein bestimmtes Kriterium nicht unterschreitet.

Modellierung von Unsicherheit und Unvollständigkeit Durch den ontologischen Ansatz wird die mögliche Unvollständigkeit der durch Sensoren erfassten Information in einem gewissen Maß abgebildet: die Situationscharakterisierung wie sie z. B. bei der Windböenwarnung in Kapitel 6.3 realisiert wurde, funktioniert auch dann, wenn die zur Charakterisierung der Situation nötigen Daten nur unvollständig vorliegen. Sobald die Daten, die zur Feststellung einer solchen Situation nötig sind, in entsprechender Ausprägung vorhanden sind, kann einer Reasoner auf die aktuelle Situation schließen. Dies liegt daran, dass der dazu nötige Rückschluss von einer ontologischen Instanz auf das dazugehörige Konzept hauptsächlich eigenschaftsbasiert geschieht und dann durchgeführt werden kann, wenn die Eigenschaften, die das Konzept charakterisieren, in ausreichender Zahl (aber nicht notwendigerweise vollständig) vorliegen. Vielmehr ist der Reasoner dann in der Lage, nach dem Erkennen einer Situation zumindest den Rahmen der fehlenden Eigenschaften vorgeben zu können, da diese ja im Konzept modelliert sein müssen. Problematisch ist hier bei nur, dass diese Rückschlüsse nicht notwendigerweise korrekt sein müssen. Wenn man sich im Beispiel des Windböenwarners vorstelle, dass zwar Geschwindigkeit und Querbeschleunigung auf eine Windböensituation zutreffen, aber der Lenkwinkel aufgrund eines technischen Defekts nicht vorliegt, dann kann der Reasoner diese Situation zwar als Windböe klassifizieren, da die unvollständigen Daten auf die „Schablone“ das Windböen-Konzepts passen. Andererseits kann er damit falsch liegen: er kann ja nicht erkennen, dass u. U. eine vom Fahrer durch eine entsprechende Lenkbewegung beabsichtigte Abweichung vom Fahrweg vorliegt.

Höherwertiger Kontext Durch die Verwendung von mehreren Ontologieschichten ist es möglich, niederwertigen Kontext (wie z. B. die Aspekte Lenkwinkel, Querbeschleunigung und Geschwindigkeit) zu einem höherwertigen Konzept zu verdichten (in diesem Fall eine Windböe). Die Verbindung niedriger Konzepte mit anderen Ontologien durch explizites Angeben einer Relation macht es weiterhin möglich, höherwertigen Kontext in die eigene Modellhierarchie einzuführen. Zusätzlich mit Zeitkonzepten können dann auch Aspekte über längere Zeiträume hinweg beobachtet werden, um beispielsweise Durchschnittswerte, Maximal- oder Minimalwerte für eine Kontextvariable (z. B. Geschwindigkeit) zu ermitteln. Damit lässt sich dann nicht nur der aktuell gültige Kontextzustand ermitteln, sondern es können auch Kontexthistorien erzeugt oder zeitliche Abläufe modelliert werden. Zur Generierung des höherwertigen Kontextes können im Falle von Ontologien mehrere Verfahren angewendet werden. In dieser Arbeit wurden zwei verwendet: ein Reasoner verwendet Instanzen von niederwertigen Kontext, um sie einem höherwertigen Konzept zuzuordnen, das durch diesen niederwertigen Kontext charakterisiert wird (Windböenerkennung). Eine weitere Möglichkeit besteht darin, niederwertigen Kontext z. B. an Objekte einer anderen Programmiersprache zu binden und diese dann intern weiter zu verarbeiten und sie später wieder an einen evtl. errechneten höherwertigen Kontext zu binden. Im Rahmen dieser Arbeit wurde das bei der Kollisionserkennung (Abschnitt 6.2) durchgeführt, da die Errechnung des Kollisionszeitpunkts für einen Reasoner zu komplex gewesen wäre. Eine dritte – im Rahmen dieser Arbeit nicht untersuchte – Möglichkeit ist das Angeben expliziter Regeln, die einen niederwertigen Kontext in höherwertigen überführt. Beispiel für eine solche Regelsprache ist SWRL, die Semantic Web Rule Language [HPSB+04], eine Kombination von OWL (nur DL und Lite) und RuleML. SWRL kann verwendet werden, um aussagenlogische Horn-Formel-artige Regeln zu definieren, mit deren Hilfe man niederwertigen Kontext auf höherwertigen abbilden kann. Regeln haben die Form einer Implikation zwischen Antezedenz (Vorbedingung) und Konsequenz: Wenn die Vorbedingung gilt, dann ist auch die Konsequenz wahr.

offenes Modell, zeitliche Evolution des Modells, Erweiterbarkeit Ontologien sind offene Modelle, da Konzepte, Instanzen und Beziehungen zu jedem Zeitpunkt hinzugefügt werden können. Es ist beim späteren Addieren von neuen Konzepten lediglich darauf zu achten, dass diese widerspruchsfrei in das bestehende Modell integrierbar sind. Diese Widersprüche können automatisch durch Reasoner ermittelt werden. Im Falle eines Konflikts muss dieser manuell aufgelöst werden – entweder durch eine Anpassung eines bestehendes Konzepts oder durch die Anpassung des hinzuzufügenden Konzepts. Sollte ein bestehendes Konzept verändert werden, so kann sich u. U. eine Inkompatibilität mit schon bestehenden Ontologien und Anwendungen ergeben. Eine solche kann aber geduldet werden, da mit der `owl:versionInfo`-Eigenschaft in OWL immerhin eine rudimentäre Versionierung von Ontologien durchgeführt werden kann. So lassen sich kompatible und inkompatible

Versionen einer Ontologie voneinander unterscheiden. Die Anwendung zur Kollisionsvermeidung zeigte, wie die spätere Erweiterung eines Modells zu einen Verbesserung der Anwendungseffizienz führen kann.

Einfache Zugänglichkeit des im Modell gespeicherten Wissens Die Zugreifbarkeit des in Ontologien gespeicherten Wissens hängt stark von der Anwendungsumgebung und den verwendeten Werkzeugen für den Umgang mit Ontologien ab. OWL ist ein mittlerweile fünf Jahre alter Standard, der recht gut mit Werkzeugen unterstützt wird. Dadurch existiert eine Vielzahl von Möglichkeiten, auf das in OWL-Wissensbasen gespeicherte Wissen zuzugreifen. Im Rahmen dieser Arbeit wurde das hauptsächlich mit JENA (Kollisionsvermeidung) und OWL-API (Kontextbasierte Adressierung und kontextbasiertes Routing) gearbeitet, mit deren Hilfe über Java-Schnittstellen auf OWL-Modellen gearbeitet werden kann. Zusätzlich zu diesen Möglichkeiten existieren noch explizite Anfrage und Manipulationssprachen für Ontologien wie SPARQL und SPARUL – diese sind aber auf RDF zugeschnitten und unterstützen daher nur einen Teil der Ausdrucksfähigkeiten von OWL. Spezielle Anfragesprachen für OWL wie OWL-QL [FHH03] haben sich aber bisher nicht als Standard durchsetzen können.

Verteiltheit der Modellierung Dadurch, dass Ontologien in anwendungsspezifische und allgemeinere obere Ontologien aufgeteilt werden können, deren Konzepte miteinander in Verbindung gesetzt werden können, ergibt sich, dass Ontologien eine Form der verteilten Modellierung unterstützen. So ist es möglich, dass in bestimmten Fahrzeugen nur die anwendungsrelevanten Teilaspekte eine Modells geladen werden müssen, weil z. B. nur ein Teil der Sensoren zur Verfügung steht, die in einem anderen Fahrzeug oder bei einem anderen Hersteller verbaut wurden. Auch ist das automatische maschinelle Zusammensetzen von verteilten Modellen zu einem größeren integrierten Modell möglich, falls Konzepte zwischen zwei Teilontologien schon mit Hilfe von z. B. Äquivalenzrelationen (`owl:equivalentClass`) oder anderen Relationen miteinander in Beziehung gesetzt wurden. Falls dies nicht der Fall ist, muss das dazu notwendige Wissen noch mit menschlicher Unterstützung explizit formuliert werden.

Formales Modell, Semantische Klarheit Schon durch die Verwendung von Ontologien kann diese Anforderung erreicht werden. Sie stellen eine Formalisierung expliziten Wissens dar, die ein Vokabular festlegt, dessen Semantik eindeutig festgeschrieben wurde. Die Bedeutung der mit OWL modellierten Ontologien ist daher semantisch eindeutig und so weit formalisiert, dass sie von Rechner weiterverarbeitet werden können. Trotz des hohen Formalisierungsgrades ist das Vokabular nach wie vor von Menschen gut verständlich (die Bedeutung der in Kapitel 2.4.3 vorgestellten OWL-Sprachelemente erschließt sich auch menschlichen Lesern noch recht intuitiv), was die Modellierung einfacher macht.

Konsistenz von Modellen und Sensordaten, partielle Konsistenzüberprüfung Durch den hohen Grad an Formalisierung ist es nun möglich, die Konsistenz eines Modell auf Widerspruchsfreiheit zu testen – und zwar sowohl auf Instanzebene als auch auf Konzeptebene. Das macht es möglich, mit Hilfe von Validatoren zu testen, ob schon zum Zeitpunkt der Modellierung Fehler gemacht wurden, die dazu führen könnten, dass ein Modell nie instantiiert werden könnte, weil keine Instanzen allen Anforderungen des Modells genügen könnten. Weiterhin ist es möglich, real auftretende Kontextinformationen (Sensorwerte) auf Konsistenz hinsichtlich eines bestimmten Modells zu testen und somit bestimmte Werte speziellen Situationen zuzuordnen (vgl. Windböenwarnung). Allerdings ist die Einsetzbarkeit solcher Validatoren stark abhängig von der eingesetzten Ontologiesprache. Bei der Umsetzung von Ontologien in XML/RDF/OWL steht einen große Anzahl an Parsern und Validatoren zur Verfügung, die bei dieser Aufgabe unterstützen.

Echtzeit-Anforderungen, Handhabbarkeit Die Handhabbarkeit von ontologischen Modellen wurde u. a. bei der Kollisionsvermeidung getestet: nicht alle Konzepte und Herleitungen von höherwertigem Kontext ließen sich hier in OWL abbilden (z. B. die Kollisionsberechnung). Jedoch war auch hier die allgemeingültige Modellierung von Konzepten aufgrund des Effizienzgewinns vorteilhaft: mächtigere Modelle führten nicht unbedingt zu längeren Berechnungszeiten. Aufgrund dieser Tatsache konnte auch die Kollisionsberechnung noch Echtzeitanforderungen (d. h. Anfragen an das Modell können in für den Einsatz in Fahrzeugen sinnvollen Zeiten beantwortet werden) einhalten. Natürlich ist die Erfüllung dieser Anforderung schwer vom Einsatzziel der Anwendung abhängig – mit dieser Einschränkung können aber folgende Feststellungen gemacht werden:

- Auch auf der Stabilitätsebene der Fahraufgabenmodellierung (vgl. Kapitel 3.1) haben Ontologien ihre Berechtigung
- Um die Anzahl der Konzepte und Instanzen auf dieser Ebene klein zu halten (und somit das Reasoning zu beschleunigen), kann man die Eigenschaften der Teilbarkeit und Zuschneidbarkeit von Ontologien auf einen Anwendungsbereich gut nutzen.
- Auf den höheren Ebenen des Fahraufgabenmodells werden mehr Daten (mit geringeren Echtzeitanforderungen) benötigt: hier können Ontologien noch größere Stärken ausspielen, da die Strukturierung eines Wissensraumes ihre eigentliche Aufgabe ist.

Integrationsfähigkeit Zunächst sind ontologische Modelle allgemeingültig definiert, was es leicht macht, sie auch für andere (wenngleich verwandte) Zwecke einzusetzen. Die Ortshierarchie, die der kontextbasierte Recommender verwendet (siehe Kapitel 6.1) kann leicht auch in anderen Anwendungen verwendet werden. Auch die Tatsache, dass Ontologien nicht als Ganzes verwendet werden müssen, sondern durch Weglassen von Konzepten ganz

einfach auf andere Ziele optimiert werden können, macht es leicht, Ontologien auch in weitere Anwendungen zu integrieren. Auch die zugrunde liegenden Basistechnologien wie XML und RDF erleichtern die die Verwendung von Ontologien in anderen Szenarien: viele vernetzende Technologien wie Web Services nutzen z. B. auch XML als Grundlage, was die Integration in solche Umgebungen vereinfacht. Wenn Anwendungen schon Ontologien verwenden, ist die Integration neuer Modelle noch leichter, wie die Erweiterung des Modells bei der Kollisionsvermeidung nachgewiesen hat. Außerdem kommt der Integration von Ontologien auch noch die inzwischen große Verbreitung von Werkzeugen zugute. Zum Zeitpunkt der Erstellung dieser Arbeit (2009) gibt es mehrere Werkzeuge zu Erstellung von Ontologien:

- Protégé der Universität Stanford (wurde in dieser Arbeit verwendet): `http://protege.stanford.edu/`
- Semantic Works des kommerziellen Herstellers Altova: `http://origin.altova.com/products/semanticworks/semantic_web_rdf_owl_editor.html`
- SWeDE, ein Plugin für Eclipse: `http://owl-eclipse.projects.semwebcentral.org/`
- SWOOP: `http://code.google.com/p/swoop/`

Die Anzahl der OWL-Reasoner ist schon mehr als zweistellig:

- Bossam: `http://bossam.wordpress.com/`
- DLog: `http://sintagma.szit.bme.hu/dlog`
- FaCT: `http://www.cs.man.ac.uk/~horrocks/FaCT/`
- FaCT++: `http://owl.man.ac.uk/factplusplus/`
- Hoolet: `http://owl.man.ac.uk/hoolet/`
- JENA verfügt über mehrere integrierte Reasoner (wurde in dieser Arbeit verwendet): `http://jena.sourceforge.net/`
- KAON2: `http://en.wikipedia.org/wiki/KAON2`
- OWLIM: `http://www.ontotext.com/owlim/index.html`
- Pellet (wurde in dieser Arbeit verwendet): `http://pellet.owldl.com/`
- RacerPro: `http://www.racer-systems.com/`
- SHER: `http://www.alphaworks.ibm.com/tech/sher`

- SweetRules: `http://sweetrules.projects.semwebcentral.org/`

Auch existieren in inzwischen zwei Frameworks zur Weiterverarbeitung von Ontologien in anderen Programmiersprachen, die beide in dieser Arbeit verwendet wurden:

- JENA: `http://jena.sourceforge.net/`
- OWL-API: `http://owlapi.sourceforge.net/`

Insgesamt lässt sich damit feststellen, das die gute Unterstützung beim Erstellen und Verarbeiten von Ontologien durch Werkzeuge zu einer guten Integrationsfähigkeit führt.

8 Zusammenfassung und Ausblick

In diesem Abschnitt werden die Ergebnisse der Arbeit zusammengefasst. Offene Fragen, die nicht im Rahmen dieser Arbeit betrachtet werden konnten oder sich erst im Verlauf der Bearbeitung ergaben, werden aufgezeigt.

8.1 Zusammenfassung

Die anwendungsübergreifende Modellierung von Kontextinformationen in automobilen ad-hoc Netzen ist ein wichtiger Beitrag für die Realisierung von kooperativen Anwendungen in dieser Domäne. Im Rahmen dieser Dissertation wird aufgezeigt, wie solche Kontextinformationen so modelliert werden können, dass sie auch für die automatische Weiterverarbeitung durch Rechner geeignet sind.

Zu diesem Zweck wird in Kapitel 1 die Herausforderungen dargestellt, die sich durch die zunehmende Verbreitung von Sensorik in Fahrzeugen und die Vernetzung der Fahrzeuge untereinander ergibt: Informationsvielfalt und -heterogenität erfordern einen ausdrucksmächtigen Ansatz zur Modellierung des durch die Sensoren ermittelten Kontexts. Ein Ansatz, der im WWW schon seit längerem zu diesem Zweck verfolgt wird, sind Ontologien, eine Technik des Semantic Web.

Kapitel 2 stellt die wesentlichen Grundlagen der Arbeit vor: der Kontextbegriff wird definiert und hier zeigt sich, dass er im Sinne einer formalen Definition nur schwer zu fassen ist: vielmehr wird alles, was im Laufe einer Anwendung an Kontextwissen nützlich sein könnte, auch als Kontext betrachtet. Dieser (operationale) Kontextbegriff macht deutlich, dass auch Kontextmodelle eine gewisse Offenheit und Erweiterbarkeit unterstützten müssen. Weiterhin wird auf die Besonderheiten der Vernetzungstechnologien in VANETs auf den unteren Schichten (1 bis 3) des OSI/OSI-Modells eingegangen: auch hier wird augenfällig, dass Kontextwissen zur Verbesserung des Durchsatzes, zur Flusskontrolle, zur effizienten Informationsausbreitung und für Verkürzungen von Wartezeiten beim Medienzugriff eingesetzt werden kann und muss. Im selben Kapitel werden auch die Fundamente des Semantic Web und eines seines herausragendsten Konzepte vorgestellt: Ontologien sind „formale explizite Spezifikationen einer gemeinsamen Konzeptualisierung“. Oft werden sie in OWL formalisiert, einer auf XML und RDF(S) basierenden Sprache, deren wesentlichen

Elemente vorgestellt wurden.

Kapitel 3 analysiert zur Herleitung von Anforderungen für Kontextmodelle in VANETs die folgenden Faktoren:

- die Fahraufgabe, deren drei unterschiedliche Ebenen (Navigation, Führung und Stabilisierung) teilweise diametral unterschiedliche Anforderungen an das Kontextdatenmodell stellen
- Aktive Sicherheitsanwendungen, die in VANETS zur Kooperation von Fahrzeugen zum Zweck der Erhöhung der Fahrsicherheit und Verringerung der Unfallzahlen und -schwere eingesetzt werden
- Deployment-Anwendungen, die bei der Markteinführung von VANET-Systemen unterstützend wirken sollen
- Markteinführungsszenarien für VANET-Systeme im Allgemeinen

Daraus ergeben sich Anforderungen, die nur von einem offenen, flexiblen und ausdrucksmächtigen Modellierungsinstrument erfüllt werden können.

Auf den Stand der Forschung mit alternativen Kontextmodellen wird in Kapitel 4 eingegangen. Dabei werden nicht nur andere Modellierungstechniken beleuchtet, sondern auch schon existierende ontologische Kontextmodelle sowie deren Stärken und Schwächen.

Die KOMODE-Basismodelle, die das gemeinsame Verständnis von grundlegenden Kontextkonzepten und deren Relationen untereinander modellieren, werden in Kapitel 5 vorgestellt. Die Basismodelle umfassen Konzeptualisierungen für Ortsinformationen, Zeitkonzepte, Umweltwahrnehmungen und Kontextqualität.

In Kapitel 6 werden die eben vorgestellten Modelle zur Anwendung gebracht:

- Ein kontextsensitiver Recommender (Abschnitt 6.1) empfiehlt Anwendern in Fahrzeugen Tankstellen basierend auf dem aktuellen Tankstand (also der Restreichweite), aufgrund von Nutzerpräferenzen und Bewertungen von anderen Nutzern, die über die Luftschnittstelle eingesammelt werden. Als Technik zur Empfehlungsgenerierung werden inhaltsbasierte und wissensbasierte Recommender, kombiniert mit einem kollaborativen Filter eingesetzt. Zur Vermeidung des bei kollaborativen Filtern auftretenden Anlaufproblems werden die semantischen Strukturen, die sich mit Hilfe von Ontologien (in sogenannten Kontexthierarchien) ausdrücken lassen, ausgenutzt. Genauso kann das semantische Modell zum Auffinden von ähnlichen Gegenständen für das kollaborative Filtern verwendet werden.

- Eine kontextbasierte Anwendung zur Kollisionsvermeidung (Abschnitt 6.2) verwendet Ontologien zur Modellierung des aktuellen Fahrzeugkontexts und von bestimmten Umgebungsparametern. Fahrzeugdaten wie Länge, Breite, Position, Geschwindigkeit, Beschleunigung und Richtung werden in Ontologien verwaltet, die als Wissensbasis für die Kollisionsberechnung dienen. Die Berechnungen selbst werden nicht „innerhalb“ von Ontologien durchgeführt, sondern durch eine in Java geschriebene Programmlogik. Die Anwendung demonstriert, wie einfach Ontologien um neue Konzepte erweitert werden können, da sie in zwei Varianten existiert: einmal mit einem graphentheoretisch modellierten Straßennetz, einmal ohne. Dass das komplexere Modell nicht unbedingt zu längeren Berechnungen führen muss, wird dadurch gezeigt, dass das Zusatzwissen im komplexeren Modell dazu verwendet werden kann, unnötige Berechnungen zu vermeiden und so die Gesamtlaufzeit zu verringern. Des weiteren zeigt die Anwendung, dass Ontologien auch im Fahraufgabenbereich der Stabilität, also im echtzeitkritischen Bereich eingesetzt werden können.

- Die Verwendung von Kontextinformationen zur Unterstützung der Vernetzung (Abschnitt 6.3) sollen demonstrieren, wie Kontextinformationen nicht nur auf Anwendungsebene gewinnbringend eingesetzt werden können, sondern auch in den unteren Schichten des ISO/OSI-Modells zur effizienteren Informationsausbreitung. Dazu werden Kontextinformationen zu einer Art Adresse „verdichtet“. Das heißt, Fahrzeuge werden im Netz nicht mehr anhand eines vorher festgelegten Identifikators (wie z. B. einer IP-Adresse) identifiziert, sondern anhand ihres Kontexts adressiert. Damit soll es möglich sein, eine Nachricht an alle Fahrzeuge zuzustellen, die sich z. B. auf der Autobahn A7 in Richtung Norden befinden. Zusätzlich wird das Szenario dazu verwendet, Ontologien als „Schablonen“ für Situationen zu verwenden, die durch bestimmte Kontextkriterien charakterisiert werden. Verifiziert werden die Verfahren anhand eines Windböenwarners.

Kapitel 7 gleicht die in den Anwendungen simulativ nachgewiesenen Eigenschaften der Kontextmodelle mit den Anforderungen aus Kapitel 3 ab: zusammenfassend kann gesagt werden, dass die speziellen Anforderungen, die in VANETs durch die hohe Dynamik und Verteiltheit gestellt werden, durch Ontologien sehr gut abgebildet werden. Ihrem Wesen nach sind Ontologien aber „gemeinsame Konzeptualisierungen“, d. h. je mehr Verbreitung sie finden, desto nützlicher werden sie. Genauso wie die Anwendungen in VANETs unterliegen auch sie Netzwerkeffekten. Je mehr Personen sich an der Modellierung von Ontologien beteiligen, sich auf ein gemeinsames Kontextverständnis einigen können und evtl. ihre eigenen Ontologien auf größere abbilden, desto weiter wird sich der Ansatz durchsetzen können. Bedenken beim Einsatz von Ontologien ergeben sich durch die Komplexität der Reasoning-Berechnungen, wenn diese eine Vielzahl von Konzepten und Instanzen beinhalten: diese können aber durch die Eigenschaft, dass sich Ontologien leicht in kleinere,

auf einen Anwendungszweck besser zugeschnittene, zerteilen lassen, etwas abgeschwächt. Auch wird die die Rechenleistung der eingebetteten Systeme in Fahrzeugen in den nächsten Jahren steigen, so dass Rechenzeitanforderungen immer leichter erfüllt werden können. Im Hinblick auf das Potential, das aktive Sicherheitsanwendungen bezüglich Verringerung der Unfallzahlen und -schwere darstellt, ist die Kooperation von Verkehrsteilnehmern fast schon eine zwingende Maßnahme. Damit sie weitgehend automatisiert ablaufen kann, ist aber ein gemeinsames Verständnis der Daten, auf denen gearbeitet wird, unumgänglich: Ontologien bieten sich hier als Möglichkeit an.

8.2 Offene Fragen und weitere Anknüpfungspunkte

Da Kontext seiner Definition nach anwendungsgetrieben ist, kann eine Modellierung von Kontext nie vollständig sein. Die vorliegende Arbeit hat einen Ausschnitt angesprochen, der für eine einzelne Domäne, nämlich Anwendungen in VANETs, von Relevanz ist. Die Betrachtung von Lösungen für diesen Teil des Problems wirft naturgemäß weitergehende Fragen und Anknüpfungspunkte auf:

- Ein Modellierung anderer Kontextkonzepte: hier wäre z. B. eine ontologische Modellierung der Fahraufgabe (die in dieser Arbeit nur zur Ableitung von Anforderungen verwendet wurde) oder des Netzwerkzustandes sinnvoll. Solche Zusatzmodelle könnten die Entwicklung eines „cognitive radio“, das Sende- und Empfangsparameter basierend auf Nutzerverhalten und Netzwerkzustand verändern kann, vereinfachen.
- Der Einsatz der Modelle in weiteren Anwendungen
- Die Integration von anderen Verkehrsteilnehmern wie z. B. Fußgängern und Fahrradfahren in das Modell
- Die Einbeziehung des Fahrers ins Kontextmodell
- Detailliertere und realistischere Simulationen (z. B. unter Zuhilfenahme eines etablierten Verkehrssimulators wie VISSIM[1], vgl. [PBW09])
- Ein Einsatz der Modelle auf einem realen Versuchsträger könnte Erfahrungswerte aus der Praxis sammeln
- Um die Berechnungskomplexität des Reasonings auf Ontologien klein zu halten, müssen diese je nach Anwendungsziel zerteilt werden. Gibt es geeignete Heuristiken, an welchen Stellen Ontologien geeignet partitioniert werden können? Kann diese Zerteilung im laufenden Betrieb geschehen?

[1] `http://www.ptv.de/software/verkehrsplanung-verkehrstechnik/software-und-system-solutions/vissim/`

- Eine Infrastruktur zur Verwaltung der Ontologien: sowohl in einem Backend-System, das relevante Ontologien speichert als auch im Fahrzeug selbst, so dass neue Modelle im Fahrzeug zur Laufzeit ins (lokale) Gesamtmodell integriert werden können
- eine geeignete Schnittstelle zur Anfrage von Kontextdaten anderer Fahrzeuge über die Luftschnittstelle – hier wäre überhaupt erst eine geeignete Anfragesprache zu finden oder zu entwickeln, die die volle Komplexität von OWL unterstützt
- Ein automatisches Auffinden von Fehlern und Inkonsistenzen in verteilten Ontologien mit Methoden des Machine Learning kann helfen, die Qualität der Modelle zu verbessern. Ebenso denkbar ist ein automatisches Abbilden von Konzepten eigener, isolierter Ontologien auf große oder andere Ontologien.
- Erarbeitung von Richtlinien für „gutes“ Ontologiedesign („Ontology design patterns“)
- Generell eine Infrastruktur für die Verwaltung von Ontologien unterschiedlicher Versionen im Fahrzeug – Methoden zur Ontologieevolution

Literaturverzeichnis

[AESS06] Adler, Christian; Eigner, Robert; Schroth, Christoph; Strassberger, Markus: Context-Adaptive Information Dissemination VANETs – Maximizing the Global Benefit. In: *Proceedings of the 5th IAESTED International Conference on Communication Systems and Networks (CSN 2006)*, 2006, S. 7–12

[AH08] Allemang, Dean; Hendler, Jim: *Semantic Web for the Working Ontologist*. Burlington, MA, USA: Morgan Kaufmann Publishers, 2008

[AS97] Akman, Varol; Surav, Mehmet: *The Use of Situation Theory in Context Modeling*. August 1997

[ASST05] Adomavicius, Gediminas; Sankaranarayanan, Ramesh; Sen, Shahana; Tuzhilin, Alexander: Incorporating contextual information in recommender systems using a multidimensional approach. In: *ACM Trans. Inf. Syst.* 23 (2005), Nr. 1, S. 103–145. `http://dx.doi.org/http://doi.acm.org/10.1145/1055709.1055714`. – DOI http://doi.acm.org/10.1145/1055709.1055714. – ISSN 1046–8188

[BBC97] Brown, Peter J.; Bovey, John D.; Chen, Xian: Context-aware applications: from the laboratory to the marketplace. In: *Personal Communications* 4 (1997), October, Nr. 5, S. 58–64. `http://dx.doi.org/doi:10.1109/98.626984`. – DOI doi:10.1109/98.626984. – ISSN 1070–9916

[BBH97] Bacon, Jean; Bates, John; Halls, David: Location-oriented multimedia. In: *IEEE Personal Communications* 4 (1997), Oktober, Nr. 5, S. 48–57. `http://dx.doi.org/10.1109/98.626983`. – DOI 10.1109/98.626983. – ISSN 1070–9916

[BD06] Braess, Hans-Herrmann; Donges, Edmund: Technologien zur aktiven Sicherheit von Personenkraftwagen – „Konsumierbare“ oder echte Verbesserungen? In: *2. Tagung Aktive Sicherheit durch Fahrerassistenz*, 2006

[BDR07] Baldauf, Matthias; Dustdar, Schahram; Rosenberg, Florian: A survey on context-aware systems. In: *International Journal Ad Hoc and Ubiquitous Computing* 2 (2007), Nr. 4, S. 263–277

[Bec04] Beckett, Dave; McBride, Brian (Hrsg.): *RDF/XML Syntax Specification (Revised).* `http://www.w3.org/TR/rdf-syntax-grammar/`. Version: Februar 2004

[Ben04] Benslimane, Abderrahim: Optimized Dissemination of Alarm Messages in Vehicular Ad-Hoc Networks (VANET). In: *High Speed Networks and Multimedia Communications* Bd. 3079. Berlin/Heidelberg, Deutschland: Springer, 2004, S. 655–666

[BEW08] Brocco, Michele; Eigner, Robert; Wörndl, Wolfgang: Ein hybrides, kontextsensitives Recommender System für mobile Anwendungen in vernetzten Fahrzeugen. In: *Tagungsband der Teilkonferenz Äutomotive Servicesäuf der Multi-Konferenz Wirtschaftsinformatik 2008 (MKWI 2008).* München, 2008

[BG04] Brickley, Dan; Guha, R. V.; McBride, Brian (Hrsg.): *RDF Vocabulary Description Language 1.0: RDF Schema.* `http://www.w3.org/TR/rdf-schema/`. Version: Februar 2004

[BH00] Briesemeister, Linda; Hommel, Günter: Role-based multicast in highly mobile but sparsely connected ad hoc networks. In: *Proceedings of the 1st ACM international symposium on Mobile ad hoc networking & computing (MobiHoc 2000).* Piscataway, NJ, USA: IEEE Press, 2000. – ISBN 0–7803–6534–8, S. 45–50

[BHH+04] Bechhofer, Sean; Harmelen, Frank van; Hendler, Jim; Horrocks, Ian; McGuinness, Deborah L.; Patel-Schneider, Peter F.; Stein, Lynn A.: *OWL Web Ontology Language Reference.* `http://www.w3.org/TR/owl-ref/`. Version: Februar 2004

[BL89] Berners-Lee, Tim: *Information Management: A Proposal.* `http://www.w3.org/History/1989/proposal.html`. Version: 1989

[BL00] Berners-Lee, Tim: *Semantic Web–XML2000.* `http://www.w3.org/2000/Talks/1206-xml2k-tbl/`, 2000. – Stand: 27.08.2007

[BLHL01] Berners-Lee, Tim; Hendler, James; Lassila, Ora: The Semantic Web. In: *Scientific American* (2001), May

[Bra95] Braess, Hans-Hermann: PROMETHEUS, Contribution to a Comprehensive Concept for Future Road Traffic. (1995)

[Bro07] Brocco, Michele: *Konzeption und prototypische Umsetzung eines kontextsensitiven Recommender Systems für mobile Anwendungen in Fahrzeugen.*

München, Deutschland, Institut für Informatik der Technischen Universität München, Diplomarbeit, 2007

[BRS+02] Bauch, Monika; Reicher, Thomas; Schwingenschlögl, Christian; Sties, Peter; Kosch, Timo: SOFTNET – Software Engineering for Soft Networking. In: *Proceedings of the 6th World Multi-Conference on Systemics, Cybernetics and Informatics*, 2002

[BS07] Braess, Hans-Hermann (Hrsg.); Seiffert, Ulrich (Hrsg.): *Vieweg Handbuch Kraftfahrzeugtechnik*. 5. Vieweg Verlagsgesellschaft, 2007 (ATZ-MTZ Fachbuch). – ISBN 978–3–8348–0222–4

[BSH00] Briesemeister, Linda; Schäfers, Lorenz; Hommel, Günter: Disseminating messages among highly mobile hosts based on inter-vehicle communication. In: *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2000, S. 522–527

[Bur00] Burke, Robin: Knowledge-based recommender systems. In: *Encyclopedia of Library and Information Systems* Bd. 69. New York, NY, USA: Marcel Dekker, 2000, S. 180–200

[Bur02] Burke, Robin: Hybrid Recommender Systems: Survey and Experiments. In: *User Modeling and User-Adapted Interaction* 12 (2002), Nr. 4, S. 331–370

[CFJ04] Chen, Harry; Finin, Tim; Joshi, Anupam: An Ontology for Context-Aware Pervasive Computing Environments. In: *Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review* (2004), Nr. 18, S. 197–207

[Che02] Chevreuil, Martial: IVHW: An Inter-vehicle Hazard Warning System Concept within the DEUFRAKO program. In: *Proceedings of the e-Safety Congress and Exhibition*, 2002

[Che05] Chen, Annie: Context-Aware Collaborative Filtering System: Predicting the User's Preference in the Ubiquitous Computing Environment. Version: 2005. `http://dx.doi.org/10.1007/11426646_23`. In: *Location- and Context-Awareness* Bd. 3479. Berlin/Heidelberg, Deutschland: Springer, 2005. – DOI 10.1007/11426646_23, 244–253

[CJ03] Clausen, Thomas H.; Jacquet, Philippe: *Optimized Link State Routing Protocol (OLSR)*. RFC 3626 (Experimental). `http://www.ietf.org/rfc/rfc3626.txt`. Version: October 2003 (Request for Comments)

[CK00] Chen, Guanling; Kotz, David: A Survey of Context-Aware Mobile Computing Research / Department of Computer Science, Dartmouth College. 2000 (TR2000-381). – Forschungsbericht

[DAS01] Dey, Anind K.; Abowd, Gregory D.; Salber, Daniel: A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. In: *Human-Computer Interaction* 16 (2001), Nr. 2, S. 97–166. – ISSN 0737–0024

[Der06] Derowski, Stefan: *Dynamische Preisbildung bei der Vermittlung von kontextsensitiven Diensten im B2B Bereich.* München, Deutschland, Institut für Informatik der Ludwig-Maximilians-Universität München, Diplomarbeit, 2006

[Dey98] Dey, Anind K.: Context-aware computing: The CyberDesk project. In: *AAAI 1998 Spring Symposium on Intelligent Environments.* Palo Alto: AAAI Press., 1998, 51-54

[Dey00] Dey, Anind K.: *Providing Architectural Support for Building Context-Aware Applications*, Georgia Institute of Technology, Diss., 2000

[Dey01] Dey, Anind K.: Understanding and Using Context. In: *Personal Ubiquitous Computing* 5 (2001), Nr. 1, S. 4–7. `http://dx.doi.org/http://dx.doi.org/10.1007/s007790170019`. – DOI http://dx.doi.org/10.1007/s007790170019. – ISSN 1617–4909

[Don76] Donges, Edmund: *Experimentelle Untersuchung und regelungstechnische Modellierung des Lenkverhaltens von Kraftfahrern bei simulierter Straßenfahrt.* Darmstadt, Deutschland, Technische Hochschule Darmstadt, Diss., 1976

[Don99] Donges, Edmund: A Conceptual Framework for Active Safety in Road Traffic. In: *Vehicle System Dynamics* 32 (1999), Nr. 2–3, S. 113–128. `http://dx.doi.org/http://dx.doi.org/10.1076/vesd.32.2.113.2089`. – DOI http://dx.doi.org/10.1076/vesd.32.2.113.2089

[Don07] Donner, Susanne: Kazaa auf der Autobahn. In: *Spiegel Online* (2007), Juli. `http://www.spiegel.de/auto/aktuell/0,1518,492896,00.html`

[DQA04] Datta, Anwitaman; Quarteroni, Silvia; Aberer, Karl: Autonomous Gossiping: A self-organizing epidemic algorithm for selective information dissemination in mobile ad-hoc networks. In: *Proceedings of the International Conference on Semantics of a Networked World* Bd. 3226. Berlin/Heidelberg, Deutschland: Springer, 2004, S. 126–143

[DRD+00] Dix, Alan; Rodden, Tom; Davies, Nigel; Trevor, Jonathan; Friday, Adrian; Palfreyman, Kevin: Exploiting space and location as a design framework for interactive mobile systems. In: *ACM Trans. Comput.-Hum. Interact.*

7 (2000), Nr. 3, S. 285–321. http://dx.doi.org/http://doi.acm.org/10.1145/355324.355325. – DOI http://doi.acm.org/10.1145/355324.355325. – ISSN 1073–0516

[EGH+06] ElBatt, Tamer; Goel, Siddhartha K.; Holland, Gavin; Krishnan, Hariharan; Parikh, Jayendra: Cooperative collision warning using dedicated short range wireless communications. In: *Proceedings of the 3rd international workshop on Vehicular ad hoc networks (VANET2006)*. New York, NY, USA: ACM Press, 2006. – ISBN 1–59593–540–1, S. 1–9

[Eig05a] Eigner, Robert: *NOW – Network On Wheels AP3.1 Deliverable: Active Saftey Applications Milestone 1.* Juli 2005

[Eig05b] Eigner, Robert: *NOW – Network On Wheels AP3.2 Deliverable: Deployment Applications.* März 2005

[EL07] Eigner, Robert; Lutz, Georg: Steigerung der Anwendungseffizienz in mobilen Netzen durch ontologische Kontextmodelle. In: *Tagungsband 4. GI/ITG KuVS Fachgespräch: Ortsbezogene Anwendungen und Dienste.* München, 2007

[EL08] Eigner, Robert; Lutz, Georg: Collision Avoidance in VANETs – An Application for ontological context models. In: *Proceedings of the 5th IEEE Workshop on Context Modeling and Reasoning (CoMoRea 2008).* Hong Kong, China, 2008

[EM08] Eigner, Robert; Mair, Christoph: Kontextbasierte Adressierung und Routing in mobilen Ad-hoc-Netzwerken. In: *Tagungsband 5. GI/ITG KuVS Fachgespräch: Ortsbezogene Anwendungen und Dienste.* Nürnberg, 2008

[EM09] Eigner, Robert; Mair, Christoph: Using Context Ontologies for Addressing and Routing in Mobile Ad-Hoc Networks. In: *Proceedings of the 8th International Conference on Networks (ICN 2009).* Cancún, Mexico, 2009

[ES06] Ehrig, Marc; Studer, Rudi: Wissensvernetzung durch Ontologien. In: Pellegrini, Tassilo (Hrsg.); Blumauer, Andreas (Hrsg.): *Semantic Web.* Springer, 2006, S. 469–484

[ESKS06] Eichler, Stephan; Schroth, Christoph; Kosch, Timo; Strassberger, Markus: Strategies for Context-Adaptive Message Dissemination in Vehicular Ad Hoc Networks. In: *Proceedings of the 2nd International Workshop on Vehicle-to-Vehicle Communications (V2VCOM 2006)*, 2006

[Füß07] Füssler, Holger: *Position-Based Packet Forwarding for Vehicular Ad-Hoc Networks*. Mannheim, Deutschland, Universität Mannheim, Diss., 2007

[FEL01] Franz, Walter; Eberhardt, Reinhold; Luckenbach, Thomas: FleetNet – Internet on the Road. In: *Proceedings of the 8th World Congress on Intelligent Transportation Systems*, 2001

[Feu93] Feuchtwanger, Lion: *Erfolg*. Berlin, Deutschland: Aufbau, 1993

[FHH03] Fikes, Richard; Hayes, Patrick; Horrocks, Ian: OWL-QL – A Language for Deductive Query Answering on the Semantic Web. Stanford, CA, USA: Stanford University, 2003. – Forschungsbericht

[FHKB05] Fuchs, Florian; Hochstatter, Iris; Krause, Michael; Berger, Michael: A metamodel approach to context information. In: *Proceedings of the 3rd IEEE International Conference on Pervasive Computing and Communications (PerCom 2005)* (2005), März, S. 8–14. `http://dx.doi.org/10.1109/PERCOMW.2005.9`. – DOI 10.1109/PERCOMW.2005.9

[FNS+08] Festag, Andreas; Noecker, Gerhard; Strassberger, Markus; Lübke, Andreas; Bochow, Bernd; Torrent-Moreno, Marc; Schnaufer, Sascha; Eigner, Robert; Catrinescu, Catrinel; Kunisch, Jürgen: NoW – Network on Wheels: Project Objectives, Technology and Achievements. In: *Proceedings of 5th International Workshop on Intelligent Transportation (WIT 2008)*, 2008

[Fra04] Franz, Walter: Car-to-Car Communication – Anwendungen und aktuelle Forschungsprogramme in Europa, USA und Japan. In: *Proceedings VDE Kongress 2004 Berlin*, VDE, 2004

[Fre02] Frege, Gottlob: *Funktion, Begriff, Bedeutung. Fünf logische Studien*. Vandenhoeck & Ruprecht, 2002. – ISBN 3–52533–377–3

[FTMT+05] Füssler, Holger; Torrent-Moreno, Marc; Transier, Matthias; Festag, Andreas; Hartenstein, Hannes: Thoughts on a Protocol Architecture for Vehicular Ad-Hoc Networks. In: *Proceedings of the 2nd International Workshop in Intelligent Transportation (WIT 2005)*, 2005, S. 41–45

[Gru93] Gruber, Thomas R.: A translation approach to portable ontology specifications. In: *Knowl. Acquis.* 5 (1993), Nr. 2, S. 199–220. `http://dx.doi.org/http://dx.doi.org/10.1006/knac.1993.1008`. – DOI http://dx.doi.org/10.1006/knac.1993.1008. – ISSN 1042–8143

[Grz07] Grzemba, Andreas (Hrsg.): *MOST: Das Multimedia-Bussystem für den Einsatz im Automobil*. Franzis Verlag, 2007

[GS01] Gray, Philip; Salber, Daniel: Modeling and Using Sensed Context Information in the Design of Interactive Applications. In: *Engineering for Human-Computer Interaction* Bd. 2254. Springer, Januar 2001, S. 317–335

[GW05] Grzemba, Andreas; Wense, Hans-Christian von d.: *LIN-Bus*. Franzis Verlag, 2005

[GWPZ04] Gu, Tao; Wang, Xiao H.; Pung, Hung K.; Zhang, Da Q.: An Ontology-based Context Model in Intelligent Environments. In: *Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2004

[Haa97] Haas, Zygmunt J.: A new routing protocol for the reconfigurable wireless networks. In: *Proceedings of the 6th IEEE International Conference on Universal Personal Communications* Bd. 2, 1997, S. 562–566

[Hal01] Halpin, Terry: *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann, 2001. – ISBN 1558606726

[HBN07] Horridge, Matthew; Bechhofer, Sean; Noppens, Olaf: Igniting the OWL 1.1 Touch Paper: The OWL API. In: *Proceedings of the 3rd OWL: Experiences and Directions Workshop (OWLED 2007)*, 2007

[HBS02] Held, Albert; Buchholz, Sven; Schill, Alexander: Modeling of Context Information for Pervasive Computing Applications. In: *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, 2002, S. 14–18

[Her06] Herring, John R.: *OpenGIS Implementation Specification for Geographic Information – Simple feature access - Part 1: Common architecture*. 2006

[HHSV07] Hiller, Andreas; Hinsberger, Arno; Strassberger, Markus; Verburg, Dirk: Results from the WILLWARN Project. In: *Proceedings of the 6th International Conference on Intelligent Transportation Systems Telecommunications (ITS 2007)*, 2007

[HIR02] Henricksen, Karen; Indulska, Jadwiga; Rakotonirainy, Andry: Modeling Context Information in Pervasive Computing Systems. In: *Proceedings of the 1st International Conference on Pervasive Computing (Pervasive 2002)*. London, UK: Springer-Verlag, 2002. – ISBN 3–540–44060–7, S. 167–180

[HIR03] Henricksen, Karen; Indulska, Jadwiga; Rakotonirainy, Andry: Generating Context Management Infrastructure from High-Level Context Models. In: *Proceedings of the 4th International Conference on Mobile Data Management (MDM 2003)*, 2003, S. 1–6

[HKRS08] Hitzler, Pascal; Krötzsch, Markus; Rudolph, Sebastian; Sure, York: *Semantic Web*. 1. Springer, 2008

[HKS06] Horrocks, Ian; Kutz, Oliver; Sattler, Ulrike: The Even More Irresistible $\mathcal{SROIQ}$. In: *Proceedings of the 10th International Conference of Knowledge Representation and Reasoning (KR 2006)*, 2006

[HP06] Hobbs, Jerry R.; Pan, Feng: *Time Ontology in OWL*. `http://www.w3.org/TR/owl-time/`. Version: September 2006

[HPSB+04] Horrocks, Ian; Patel-Schneider, Peter F.; Boley, Harold; Tabet, Said; Grosof, Benjamin; Dean, Mike: *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. `http://www.w3.org/Submission/SWRL/`. Version: May 2004

[HS03] Horrocks, Ian; Sattler, Ulrike: Decidability of $\mathcal{SHIQ}$ with Complex Role Inclusion Axioms. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 2003)*, Morgan-Kaufmann Publishers, 2003

[HSB+05] Heckmann, Dominik; Schwartz, Tim; Brandherm, Boris; Schmitz, Michael; Wilamowitz-Moellendorff, Margeritta von: GUMO – The General User Model Ontology. In: *User Modeling 2005* Bd. 3538. Springer, Januar 2005, S. 428–432

[HSP+03] Hofer, Thomas; Schwinger, Wieland; Pichler, Mario; Leonhartsberger, Gerhard; Altmann, Josef; Retschitzegger, Werner: Context-awareness on mobile devices - the hydrogen approach. In: *Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS 2003)* (2003), Januar. `http://dx.doi.org/10.1109/HICSS.2003.1174831`. – DOI 10.1109/HICSS.2003.1174831

[IEE] IEEE 802.11p Working Group: *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. Wireless Access in Vehicular Environments (WAVE)*. New York, NY, USA, . – to be published

[IEE99] IEEE 802.11a Working Group: *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. High-speed Physical Layer in the 5 GHz Band*. New York, NY, USA, 1999

[IEE05a] IEEE 802.11b Working Group: *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements.* New York, NY, USA, 2005

[IEE05b] IEEE 802.15.1 Working Group: *Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs).* New York, NY, USA, 2005

[IEE07] IEEE 802.11 Working Group: *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.* New York, NY, USA, 2007

[JHM07] Johnson, David B.; Hu, Yih-Chun; Maltz, David A.: *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4.* RFC 4728 (Experimental). `http://www.ietf.org/rfc/rfc4728.txt`. Version: February 2007 (Request for Comments)

[KAE+06] Kosch, Timo; Adler, Christian; Eichler, Stephan; Schroth, Christoph; Strassberger, Markus: The scalability problem of vehicular ad hoc networks and how to solve it. 13 (2006), 22-28. `http://dx.doi.org/10.1109/WC-M.2006.250354`. – DOI 10.1109/WC–M.2006.250354. – ISSN 1536–1284

[KC04] Klyne, Graham; Carroll, Jeremy J.; McBride, Brian (Hrsg.): *Resource Description Framework (RDF): Concepts and Abstract Syntax.* `http://www.w3.org/TR/rdf-concepts/`. Version: Februar 2004

[Kis06] Kiss, Cédric: *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0.* `http://www.w3.org/TR/CCPP-struct-vocab2/`. Version: Dezember 2006

[KK00] Karp, Brad; Kung, H. T.: GPSR: greedy perimeter stateless routing for wireless networks. In: *Proceedings of the 6th annual international conference on Mobile computing and networking (MobiCom 2000).* New York, NY, USA: ACM, 2000. – ISBN 1–58113–197–6, S. 243–254

[Kos04] Kosch, Timo: Local Danger Warning based on Vehicle Ad-hoc Networks: Prototype and Simulation. In: *Proceedings of 1st International Workshop on Intelligent Transportation (WIT 2004)*, 2004

[Kos05] Kosch, Timo: *Situationsadaptive Kommunikation in Automobilen Ad-hoc Netzen.* München, Deutschland, Technische Universität München, Diss., März 2005

[KR03] Kurose, James F.; Ross, Keith W.: *Computer Networks.* Pearson Education, Inc., 2003

[KRW+04] Klyne, Graham; Reynolds, Franklin; Woodrow, Chris; Ohto, Hidetaka; Hjelm, Johan; Butler, Mark H.; Tran, Luu: *Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0.* `http://www.w3.org/TR/CCPP-struct-vocab/`. Version: Januar 2004

[KSA02] Kosch, Timo; Schwingenschlögl, Christian; Ai, Li: Information dissemination in multihop inter-vehicle networks. In: *Proceedings of the 5th IEEE International Conference on Intelligent Transportation Systems (ITS 2002)* (2002), S. 685–690. `http://dx.doi.org/10.1109/ITSC.2002.1041301`. – DOI 10.1109/ITSC.2002.1041301

[KV98] Ko, Young-Bae; Vaidya, Nitin H.: Location-based Multicast in Mobile Ad Hoc Networks / Texas A & M University. 1998 (TR98-018). – Forschungsbericht

[KV99] Ko, Young-Bae; Vaidya, Nitin H.: Geocasting in mobile ad hoc networks: Location-based multicast algorithms. In: *Proceedings of the 2nd IEEE Workshop on Mobile Computer Systems and Applications*, 1999, S. 101–110

[KV00] Ko, Young-Bae; Vaidya, Nitin H.: Location-aided routing (LAR) in mobile ad hoc networks. In: *Wireless Networks* 6 (2000), Nr. 4, S. 307–321. `http://dx.doi.org/http://dx.doi.org/10.1023/A:1019106118419`. – DOI http://dx.doi.org/10.1023/A:1019106118419. – ISSN 1022–0038

[Law00] Lawrenz, Wolfhard: *CAN Controller Area Network. Grundlagen und Praxis.* 4. Hüthig-Verlag, 2000

[LE06] Linsmeier, Wolfgang; Eigner, Robert: Design Criteria for Wireless Payment Applications in Vehicular Ad-Hoc Networks. In: *Proceedings of the IADIS International Conference e-Commerce.* Barcelona, Spanien: IADIS Press, 2006, S. 127–134

[Lin06] Linsmeier, Wolfgang: *Konzeption und Realisierung einer Anwendung für drahtlose Bezahlvorgänge in ad-hoc Fahrzeug-Netzwerken.* München, Deutschland, Institut für Informatik der Technischen Universität München, Diplomarbeit, 2006

[LM05] Lemire, Daniel; Maclachlan, Anna: Slope One Predictors for Online Rating-Based Collaborative Filtering. In: *SIAM Data Mining (SDM 2005)*, 2005

[LMFH05] Lochert, Christian; Mauve, Martin; Füssler, Holger; Hartenstein, Hannes: Geographic Routing in City Scenarios. In: *ACM SIGMOBILE Mobile Computing and Communications Review (MC2R)* 9 (2005), Jan, Nr. 1, S. 69–72. http://dx.doi.org/10.1145/1055959.1055970. – DOI 10.1145/1055959.1055970

[Lut07] Lutz, Georg: *Kollisionsvermeidung für Fahrzeuge als Beispielanwendung für ein ontologisches Kontextmodell.* München, Deutschland, Institut für Informatik der Technischen Universität München, Diplomarbeit, 2007

[Mai08] Mair, Christoph: *Entwicklung und Simulation eines kontextbasierten Adressierungs- und Routing-Verfahrens in mobilen Ad-hoc-Netzwerken.* München, Deutschland, Institut für Informatik der Technischen Universität München, Diplomarbeit, 2008

[MB94] McCarthy, John; Buvač, Saša: Formalizing Context (Expanded Notes). Stanford, CA, USA: Stanford University, 1994. – Forschungsbericht

[MFE03] Maihöfer, Christian; Franz, Walter; Eberhardt, Reinhold: Stored Geocast. In: *13. GI/ITG-Fachtagung Kommunikation in verteilten Systemen (KiVS 2003)*, 2003

[MH02] Miller, Ronald; Huang, Quinfeng: An Adaptive Peer-to-Peer Collision Warning System. In: *Proccedings of the IEEE Vehicular Technology Conference (VTC 2002)*, 2002, S. 317–321

[MH04] McGuinness, Deborah L.; Harmelen, Frank van: *OWL Web Ontology Language Overview.* http://www.w3.org/TR/owl-features/. Version: Februar 2004

[MM04] Matheus, Kirsten; Morich, Rolf: *NOW – Network On Wheels AP3.4 Deliverable: Business Model and Market Introduction for Car-to-Car Communication.* November 2004

[MMP+05] Matheus, Kirsten; Morich, Rolf; Paulus, Ingrid; Menig, Cornelius; Lübke, Andreas; Rech, Bernd; Specks, Will: Car-to-Car Communication – Market Introduction and Success Factors. In: *Proccedings of the 5th European Congress and Exhibition on Intelligent Transport Systems and Services (ITS 2005)*, 2005

[Moy98] Moy, John T.: *OSPF Version 2.* RFC 2328 (Standard). http://www.ietf.org/rfc/rfc2328.txt. Version: April 1998 (Request for Comments)

[MWH01] Mauve, Martin; Widmer, Jörg; Hartenstein, Hannes: A survey on position-based routing in mobile ad hoc networks. In: *Network, IEEE* 15 (2001), November/December, Nr. 6, S. 30–39. http://dx.doi.org/10.1109/65.967595. – DOI 10.1109/65.967595. – ISSN 0890–8044

[NM78] Nora, Simon (Hrsg.); Minc, Alain (Hrsg.): *L'Informatisation de la société*. Paris, Frankreich: La Documentation française, 1978

[Nor98] Norman, Donald A.: *The Invisble Computer*. MIT Press, 1998

[NTCS99] Ni, Sze-Yao; Tseng, Yu-Chee; Chen, Yuh-Shyan; Sheu, Jang-Ping: The Broadcast Storm Problem in a Mobile Ad Hoc Network. In: *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking (MobiCom 1999)*. New York, NY, USA: ACM, 1999. – ISBN 1–58113–142–9, S. 151–162

[OA06] O'Brien, Philip; Abidi, Syed Sibte R.: Contextual Knowledge Sharing in a P2P Network. In: *IEEE International Conference on Engineering of Intelligent Systems* (2006), S. 1–6. http://dx.doi.org/10.1109/ICEIS.2006.1703169. – DOI 10.1109/ICEIS.2006.1703169

[OH01] O'Connor, Mark; Herlocker, Jon: *Clustering items for collaborative filtering*. http://citeseer.ist.psu.edu/connor01clustering.html. Version: 2001

[PBRD03] Perkins, Charles E.; Belding-Royer, Elizabeth M.; Das, Samir R.: *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561 (Experimental). http://www.ietf.org/rfc/rfc3561.txt. Version: July 2003 (Request for Comments)

[PBW09] Prinz, Vivian; Brocco, Michele; Woerndl, Wolfgang: Distributed Information Management and Publish/Subscribe in VANETs: Requirements, State of the Art and a Novel P2P-based Approach. In: *Mobile Multimedia* 5 (2009), Nr. 2, S. 158–180

[Pol85] Polanyi, Michael (Hrsg.): *Implizites Wissen*. Suhrkamp Verlag KG, 1985

[PRM98] Pascoe, Jason; Ryan, Nick S.; Morse, David R.: Human-Computer-Giraffe Interaction: HCI in the Field. In: *Workshop on Human Computer Interaction with Mobile Devices* University of Glasgow, 1998 (GIST Technical Report G98–1)

[PSHH04] Patel-Schneider, Peter F.; Hayes, Patrick; Horrocks, Ian: *OWL Web Ontology Language Semantics and Abstract Syntax*. http://www.w3.org/TR/owl-semantics/. Version: Februar 2004

[Rau07] Rausch, Mathias: *FlexRay. Grundlagen, Funktionsweise, Anwendung.* Hanser Fachbuch Verlag, 2007

[RMM+02] Reichardt, Dirk; Miglietta, Maurizio; Moretti, Lino; Morsink, Peter; Schulz, Wolfgang: CarTALK 2000 – Safe and Comfortable Driving Based Upon Inter-Vehicle-Communication. In: *Proceedings of Intelligent Vehicle Symposium* Bd. 2, IEEE, 2002, S. 454–550

[Ros24] Ross, William D. (Hrsg.): *Aristotle's Metaphysics.* Oxford, UK: Clarendon Press, 1924

[Rot05] Roth, Jörg: *Mobile Computing.* Bd. II. 2. dpunkt.verlag GmbH, 2005. – ISBN 3–89864–366–2

[RSK+06] Roussaki, Ioanna; Strimpakou, Maria; Kalatzis, Nikos; Anagnostou, Miltos; Pils, Carsten: Hybrid context modeling: A location-based scheme using ontologies. In: *Proceedings of the IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom)* (2006), S. 2–7. http://dx.doi.org/http://doi.ieeecomputersociety.org/10.1109/PERCOMW.2006.65. – DOI http://doi.ieeecomputersociety.org/10.1109/PERCOMW.2006.65

[SAE06] Strassberger, Markus; Adler, Christian; Eigner, Robert: Situationsadaptive Verbreitung von Kontextinformationen in automobilen Ad-hoc-Netzen. In: *PIK - Praxis der Informationsverarbeitung und Kommunikation* 29 (2006), Nr. 1, S. 43–49. http://dx.doi.org/10.1515/PIKO.2006.43. – DOI 10.1515/PIKO.2006.43

[SAW94] Schilit, Bill N.; Adams, Norman; Want, Roy: Context-Aware Computing Applications. In: *Proceedings Workshop on Mobile Computing Systems and Applications* IEEE, 1994, S. 85–90

[SB05] Sheng, Quan Z.; Benatallah, Boualem: ContextUML: A UML-Based Modeling Language for Model-Driven Development of Context-Aware Web Services Development, IEEE, 2005, S. 206–212

[SBF97] Studer, Rudi; Benjamins, Richard; Fensel, Dieter: Knowledge engineering: principles and methods / Institut für Angewandte Informatik und Formale Beschreibungsverfahren, Universität Karlsruhe. 1997 (372). – Forschungsbericht

[SBG99] Schmidt, Albrecht; Beigl, Michael; Gellersen, Hans-W.: There is more to context than location. In: *Computers and Graphics* 23 (1999), S. 893–901

[Sch00] Schöning, Uwe: *Logik für Informatiker*. 5. Heidelberg, Deutschland: Spektrum, Akademischer Verlag, 2000

[SES02] Santos, R. A.; Edwards, R. M.; Seed, N. L.: Using the cluster-based location routing (CBLR) algorithm for exchanging information on a motorway, 2002, S. 212–216

[SGM04] Sun, Qixiang; Garcia-Molina, Hector: Using Ad-hoc Inter-vehicle Networks For Regional Alerts / Computer Science Department, Stanford University. 2004. – Forschungsbericht

[SK02] Schwingenschlögl, Christian; Kosch, Timo: Geocast enhancements of AODV for vehicular networks. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 6 (2002), Nr. 3, S. 96–97. `http://dx.doi.org/http://doi.acm.org/10.1145/581291.581307`. – DOI http://doi.acm.org/10.1145/581291.581307. – ISSN 1559–1662

[SLP04] Strang, Thomas; Linnhoff-Popien, Claudia: A Context Modeling Survey. In: *First International Workshop on Advanced Context Modelling, Reasoning And Management*, 2004

[SLPF03] Strang, Thomas; Linnhoff-Popien, Claudia; Frank, Korbinian: CoOL: A Context Ontology Language to enable Contextual Interoperability. In: *4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS 2003)* Bd. 2893. Springer, 11 2003, S. 236–247

[SMLP01] Samulowitz, Michael; Michahelles, Florian; Linnhoff-Popien, Claudia: CAPEUS: An Architecture for Context-Aware Selection and Execution of Services. In: *Proceedings of the IFIP TC6 / WG6.1 Third International Working Conference on New Developments in Distributed Applications and Interoperable Systems*. Deventer, The Netherlands, The Netherlands: Kluwer, B.V., 2001. – ISBN 0–7923–7481–9, S. 23–40

[SNB05] Schulze, Matthias; Nöcker, Gerhard; Böhm, Konrad: PReVENT: A European program to improve active safety. In: *Proceedings of the 5th International Conference on Intelligent Transportation Systems Telecommunications (ITS 2005)*, 2005

[SPKR96] Swartout, Bill; Patil, Ramesh; Knight, Kevin; Russ, Tom: Towards distributed use of large-scale ontologies. In: *Proceedings of the 10th Workshop On Knowledge Acquisition for Knowledge-Based Systems*, 1996

[SSEE06] Schroth, Christoph; Strassberger, Markus; Eigner, Robert; Eichler, Stephan: A Framework for Network Utility Maximization in VANETs. In:

Proceedings of the 3rd ACM International Workshop on Vehicular Ad Hoc Networks (VANET 2006), 2006

[ST94] Schilit, Bill N.; Theimer, Marvin M.: Disseminating active map information to mobile hosts. In: *Network* 8 (1994), Nr. 5, S. 22–32. `http://dx.doi.org/doi:10.1109/65.313011`. – DOI doi:10.1109/65.313011. – ISSN 0890–8044

[Str04] Strang, Thomas: *Service-Interoperabilität in Ubiquitous Computing Umgebungen*, Ludwig-Maximilians-Universität München, Diss., 2004

[Str07] Strassberger, Markus: *Kontextbereitstellung in Automobilen Ad-Hoc Netzen.* München, Deutschland, Institut für Informatik der Ludwig-Maximilians-Universität München, Diss., 2007

[SY04] Scott, Donald J.; Yasinsac, Alec: Dynamic Probabilistic Retransmission in Ad Hoc Networks. In: *Proceedings of the International Conference on Wireless Networks*, 2004, S. 158–164

[Tan02] Tanenbaum, Andrew S.: *Computer Networks.* 4. Prentice Hall, 2002

[TMSH06] Torrent-Moreno, Marc; Santi, Paolo; Hartenstein, Hannes: Distributed Fair Transmit Power Adjustment for Vehicular Ad Hoc Networks. In: *Proceedings of the 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks (SECON 2006)*, IEEE, 2006

[Tob96] Tobies, Stephan: The Complexity of Reasoning with Cardinality Restrictions and Nominals in Expressive Description Logics. In: *Artificial Intelligence Research* 12 (1996), S. 199–217

[Tob01] Tobies, Stephan: *Complexity Results and Practical Algorithms for Logics in Knowledge Representation.* Aachen, Deutschland, Rheinisch-Westfälisch Technische Hochschule Aachen, Diss., Mai 2001

[UG96] Uschold, Mike; Gruninger, Michael: Ontologies: Principles, Methods and Applications. In: *Knowledge Engineering Review* 11 (1996), S. 93–136

[WBE08] Woerndl, Wolfgang; Brocco, Michele; Eigner, Robert: A Context-Aware Gas Station Recommender System for Vehicular Ad-Hoc Networks. In: *Proceedings of the IADIS International Conference Wireless Applications and Computing (WAC2008).* Amsterdam, Niederlande: IADIS Press, 2008

[WBE09] Woerndl, Wolfgang; Brocco, Michele; Eigner, Robert: Context-Aware Recommender Systems in Mobile Scenarios. In: *Information Technology and Web Engineering* 4 (2009), Nr. 1, S. 67–85

[WE07] Woerndl, Wolfgang; Eigner, Robert: Context-Aware, Collaborative Applications for Inter-Networked Cars. In: *Proceedings of the 5th IEEE International Workshop on Distributed and Mobile Collaboration (DMC 2007)*. Paris, Frankreich, 2007

[Wei91] Weiser, Mark: The Computer for the 21st Century. In: *Scientific American* (1991), September

[WER+03] Wischhof, Lars; Ebner, André; Rohling, Hermann; Lott, Matthias; Halfmann, Rüdiger: SOTIS – A Self-Organizing Traffic Information System. In: *Proceedings of the 57th IEEE Vehicular Technology Conference (VTC 03 Spring)* IEEE, 2003, S. 2442–2446

[WFGH04] Wu, Hao; Fujimoto, Richard; Guensler, Randall; Hunter, Michael: MDDV: a mobility-centric data dissemination algorithm for vehicular networks. In: *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks (VANET 2004)*. New York, NY, USA: ACM, 2004. – ISBN 1–58113–922–5, S. 47–56

[Win01] Winograd, Terry: Architectures for context. In: *Human-Computer Interaction* 16 (2001), Nr. 2–4, S. 401–419. `http://dx.doi.org/doi:10.1207/S15327051HCI16234_19`. – DOI doi:10.1207/S15327051HCI16234_19. – ISSN 1044–7318

[WR05a] Wischhof, Lars; Rohling, Hermann: Congestion control in vehicular ad hoc networks. (2005), October, S. 58–63. `http://dx.doi.org/10.1109/ICVES.2005.1563614`. – DOI 10.1109/ICVES.2005.1563614

[WR05b] Wischhof, Lars; Rohling, Hermann: On Utility-Fair Broadcast in Vehicular Ad Hoc Networks. (2005), March, S. 47–51

[WZGP04] Wang, Xiao H.; Zhang, Da Q.; Gu, Tao; Pung, Hung K.: Ontology Based Context Modeling and Reasoning using OWL. (2004), March, S. 18–22. `http://dx.doi.org/10.1109/PERCOMW.2004.1276898`. – DOI 10.1109/PERCOMW.2004.1276898

[YYFK03] Yang, Lee; Yang, Ji H.; Feron, Eric; Kulkarni, Vishwesh: Development of a Performance-Based Approach for a Rear-End Collision Warning and Avoidance system for Automobiles. In: *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2003)*, 2003, S. 316–321

[ZSRC08] Zang, Yunpeng; Stibor, Lothar; Reumerman, Hans-Jürgen; Chen, Hui: Wireless local danger warning using inter-vehicle communications in highway

scenarios. In: *14th European Wireless Conference (EW 2008)* (2008), June, S. 1–7. http://dx.doi.org/10.1109/EW.2008.4623905. – DOI 10.1109/EW.2008.4623905