

Sichere Intelligente Mobilität
Testfeld Deutschland



Deliverable D21.2

Konsolidierter Systemarchitekturentwurf

Version	3.0
Verbreitung	öffentlich
Projektkoordination	Daimler AG
Versionsdatum	09.10.2009



sim^{TD} wird gefördert und unterstützt durch
Bundesministerium für Wirtschaft und Technologie
Bundesministerium für Bildung und Forschung
Bundesministerium für Verkehr, Bau und Stadtentwicklung

Beiträge wurden verfasst von

Hagen Stübing – Adam Opel GmbH (Editor)

Marc Bechler – BMW Group Forschung und Technik (Editor)

Stephan Buchta – Adam Opel GmbH (Editor)

Bechir Allani – Hochschule für Technik und Wirtschaft des Saarlandes

Thomas Baum – Hochschule für Technik und Wirtschaft des Saarlandes

Thomas Biehle – Volkswagen AG

Norbert Bißmeyer – Fraunhofer SIT

Murat Caliskan – Volkswagen AG

Kurt Eckert – Robert Bosch GmbH

Jörg Freudenstein – Albrecht Consult GmbH

Manuel Fünfrocken – Hochschule für Technik und Wirtschaft des Saarlandes

Martin Goralczyk – Technische Universität Berlin

Matthias Haug – Robert Bosch GmbH

Florian Häusler – Technische Universität Berlin

Dieter Heussner – Hessisches Landesamt für Straßen- und Verkehrswesen

Andreas Hiller – Daimler AG

Arno Hinsberger – Hochschule für Technik und Wirtschaft des Saarlandes

Attila Jaeger – Technische Universität Darmstadt

Josef Kaltwasser – Albrecht Consult GmbH

Volker Kanngießer – Stadt Frankfurt am Main

Anselm Keil – Continental Teves AG & Co ohG

Carsten Kemper – Gevas Software GmbH

Sascha Kilb – T-Systems GEI GmbH

Oliver Klages – ICT Software Engineering Nord GmbH

Felix Klanner – BMW Group Forschung und Technik

José Luis Mateo – T-Systems GEI GmbH

Manuel Mattheß – Fraunhofer SIT

Thomas May – Robert Bosch GmbH

Marc Menzel – Continental Teves AG & Co ohG

Karl Naab – BMW AG

Carsten Neumann – Fraunhofer FOKUS

Anastasia Petrou – BMW Group Forschung und Technik

Ilya Radusch – Technische Universität Berlin

Horst Rechner – Fraunhofer FOKUS

Oliver Sawade – Fraunhofer FOKUS

Gunter Schaaf – Robert Bosch GmbH

Björn Schünemann – Technische Universität Berlin

Winfried Stephan – T-Systems GEI GmbH

Markus Trauberg – ICT Software Engineering Nord GmbH

Peter Vogel – Robert Bosch GmbH

Jonas Vogt – Hochschule für Technik und Wirtschaft des Saarlandes

Michael Wagner – Adam Opel GmbH

Sebastian Weber – Hochschule für Technik und Wirtschaft des Saarlandes

Peter Zahn – BMW Group Forschung und Technik

Jens Zech – TU Berlin

Projektkoordination

Dr. Christian Weiß

Daimler AG

HPC 050 – G021

71059 Sindelfingen

Germany

Telefon +49 7031 4389 550

Fax +49 7031 4389 210

E-mail christian.a.weiss@daimler.com

Das sim^{TD} Konsortium übernimmt keinerlei Haftung in Bezug auf die veröffentlichten Deliverables. Änderungen sind ohne Ankündigung möglich. © Copyright 2009 sim^{TD} Konsortium

The sim^{TD} consortium will not be liable for any use of the published deliverables. Contents are subject to change without notice. © Copyright 2009 sim^{TD} consortium

Inhaltsverzeichnis

Zusammenfassung.....	1
English summary.....	2
1 Einführung	3
1.1 Systemüberblick	3
1.2 Teilsysteme.....	5
1.2.1 Überblick Infrastruktur.....	5
1.2.1.1 Drahtgebundene Schnittstellen (Anbindung IRS, LSA, IGLZ, VsZ)	5
1.2.2 ITS Vehicle Station	14
1.2.3 ITS Roadside Station.....	15
2 Übergreifende Konzepte der Gesamtarchitektur	18
2.1 Kommunikation über ITS G5A.....	18
2.1.1 C2X Software stack	19
2.1.1.1 C2X Software-Stack (Cirquent Anteil).....	20
2.1.1.2 Schnittstelle RemoteInfoBase.....	21
2.1.1.3 Schnittstelle RemoteInfoConn	21
2.1.1.4 Schnittstelle GEO-ADHOC-NET	21
2.1.2 GEO-ADHOC-NET	21
2.1.2.1 Schnittstelle 802.11p MAC.....	22
2.1.3 Priorisierung von Nachrichten und Congestion control.....	22
2.1.4 Versenden von Nachrichten	22
2.1.5 Empfangen von Nachrichten	23
2.2 Kommunikation über Consumer-WLAN und Mobilfunk	25
2.2.1 Konzept für Mobilfunk	26
2.2.1.1 Übersicht.....	27
2.2.1.2 Schnittstellen.....	28
2.2.1.3 Protokolle	28
2.2.1.4 Quality of Service (QoS)	32
2.2.2 Konzept für Consumer-WLAN	32
2.2.2.1 Übersicht.....	32
2.2.2.2 Schnittstellen.....	33

2.2.2.3	Protokolle	33
2.3	Umgang mit Positionsdaten	35
2.4	Testsystem	36
2.4.1	Prüfstand	36
2.4.2	Versuchsdatenverarbeitung	42
2.4.2.1	Plattformen	42
2.4.2.2	Messdaten	45
2.4.2.3	Livedaten	53
2.4.3	Versuchsunterstützung	56
2.4.3.1	Flottenmanagement	58
2.4.3.2	Testablaufplanung	62
2.4.3.3	Versuchssteuerung	69
2.4.3.4	Fahrerbefragung	71
2.5	Security-Konzept	71
2.5.1	Grobe IT-Sicherheitsarchitektur	72
2.5.2	Einsatz von Pseudonymen	74
2.5.2.1	Wechsel von Pseudonymen	75
2.5.2.2	Fahrzeugseitige Pseudonymverwaltung	75
2.5.2.3	Zentrale Pseudonymverwaltung	76
2.5.3	ITS G5A WLAN 802.11p	76
2.5.3.1	Signierung und Verschlüsselung	76
2.5.3.2	Plausibilitätsprüfung	77
2.5.4	WLAN 802.11 b/g	77
2.5.4.1	C2X-Nachrichten über Consumer-WLAN	78
2.5.4.2	Consumer-WLAN zur Übertragung anwendungsspezifischer Nachrichten	78
2.5.4.3	Consumer-WLAN im Infrastrukturmodus	79
2.5.5	ITS IMT Public	79
2.5.5.1	Datenübertragung	80
2.5.5.2	Übertragung von C2X-Nachrichten	80
2.5.6	Absicherung der IP-basierten Kommunikationsverbindungen	80
2.6	Basisidentifikation der ITS Stations	80
3	Beschreibung der Teilsysteme	82

3.1	Vehicle/Roadside CCU	82
3.1.1	GPS	85
3.1.2	Bessere Ortung	86
3.1.2.1	Zeit und Positionsgenauigkeit	86
3.1.2.2	Verwendung von DGPS-Korrekturdaten	87
3.1.2.3	Positionsfilterung von Fahrzeugdaten über VAPI	88
3.1.2.4	Mapmatching und Digitale Karte	88
3.1.2.5	Mapmatching und Funk-Topologiedaten	88
3.1.2.6	Rückführung von Mapmatching-Daten zur Positionsverbesserung	89
3.1.2.7	Positions-Stempelung von C2X-Nachrichten	89
3.1.2.8	Zentrale Auswertung von Positionsdaten	89
3.1.2.9	OEM- und fahrzeugspezifische Parametrierung	90
3.1.2.10	Ausgangsdaten und deren Nutzung	90
3.1.3	C2C-Module	90
3.1.3.1	Management & Facilities auf der CCU	91
3.1.3.2	C2X Communication Component – SIM-NET (in English)	93
3.1.4	2G/3G/Consumer-WLAN	95
3.1.4.1	Wireless Manager (in English)	96
3.1.5	Fahrzeug-Datenzugriff (VAPI)	97
3.1.5.1	VAPI Server	97
3.1.5.2	CAN-Proxy	100
3.1.5.3	LLCF	100
3.1.5.4	CAN-Schnittstelle	100
3.1.6	CCU Security	101
3.1.7	Router-API	102
3.2	Vehicle Application Unit	102
3.2.1	Application Framework	104
3.2.1.1	Plattform und Framework	104
3.2.1.2	Komponentenklassifizierung	105
3.2.1.3	Zugriffsregeln bezüglich der Komponenten	106
3.2.2	System Manager	106
3.2.3	Kommunikations-API	109

3.2.4	Datenbasis	110
3.2.4.1	VAPI-Client	111
3.2.4.2	C2X-Nachrichten und ASN.1-Decoder	114
3.2.4.3	Umfeldtabelle	116
3.2.4.4	Relevanzfilter	124
3.2.5	Navi/Karte	128
3.2.5.1	Zieleingabe	128
3.2.5.2	Routenplanung	128
3.2.5.3	Routenberechnung	129
3.2.5.4	Routenführung	129
3.2.5.5	Zugriff auf die Route	129
3.2.5.6	Elektronischer Horizont.....	130
3.2.5.7	Zugriff auf Attribute von Segmenten/Kanten.....	130
3.2.5.8	Fahrzeugpositionierung	131
3.2.5.9	Grafische Kartenschnittstelle	131
3.2.5.10	Verwaltung von mehreren Karten	132
3.2.5.11	Verwaltung von POIs	133
3.2.5.12	Sonderfall Testgelände und Versuchsplanung	133
3.2.5.13	Logging	133
3.2.5.14	Architektur.....	133
3.2.5.15	Datenspezifikation.....	135
3.2.5.16	Schnittstellen.....	138
3.2.5.17	Offene Punkte	144
3.2.6	HMI	145
3.2.7	Security	145
3.2.7.1	Kommunikationsabsicherung Client.....	145
3.2.7.2	Plausibilitätsprüfer.....	145
3.2.7.3	Verteildienste für individuelle und allgemeine Sicherheitsparameter.....	146
3.2.8	Positionskettenverarbeitung (TraceHandling).....	147
3.2.9	Funktionskoordination.....	148
3.2.9.1	Koordination der Ausgaben	149
3.2.9.2	Koordination der Funktionsaktivierung.....	150

3.2.9.3	Anwendung der Funktionskoordination in den Hauptfunktionen.....	150
3.2.10	Verkehrslage.....	151
3.2.11	Fahrerverhaltensvorhersage.....	156
3.2.11.1	Importierte Schnittstellen	158
3.2.11.2	Exportierte Schnittstellen	159
3.2.12	Hauptfunktionen.....	159
3.2.13	OEM-Sonderfunktionen	159
3.2.13.1	Sonderfunktionen und OEM-Devices.....	159
3.2.13.2	Anforderungen	159
3.2.13.3	Konzept.....	160
3.2.13.4	Integrationsverantwortung und Integrationsregeln.....	160
3.3	Roadside Application Unit.....	161
3.3.1	Physikalischer Aufbau einer IRS	161
3.3.2	Logischer Aufbau IRS	163
3.3.3	FunctionFramework	164
3.3.3.1	Bundle Umfeldtabelle.....	165
3.3.3.2	Bundle Proxy	165
3.3.3.3	Zugriff auf Zeit: Java Time	165
3.3.3.4	Bundle IRS_GPSProxy	165
3.3.3.5	Bundle CoMa	166
3.3.3.6	Bundle IRSGeoCastClient	169
3.3.4	IRS Management Framework.....	170
3.3.4.1	CMC	170
3.3.4.2	FMC	170
3.3.4.3	NMC.....	171
3.3.4.4	IRS-LOG	171
3.3.4.5	CoMa	171
3.3.4.6	RessourceManager.....	171
3.3.4.7	IRS-Funktionen.....	171
3.3.5	IRS Management Center	173
3.3.5.1	Allgemeine Aufgaben eines IRS-Management	173
3.3.5.2	Physikalischer Aufbau des IRS Management Centers (IRSMC)	174

3.3.5.3 Subsysteme des IRSMC.....	175
3.3.6 ApplicationServer.....	178
3.3.6.1 AMIC	178
3.3.6.2 IRSProxy.....	178
3.3.7 Schnittstellen	179
3.3.7.1 Schnittstelle IRS – IRS.....	179
3.4 IGLZ.....	180
3.5 VZH.....	181
3.6 Versuchszentrale (VsZ)	181
3.6.1 Organisatorisches.....	182
3.6.2 Hardware	182
3.6.3 Software.....	182
3.6.4 Internet-Anbindung	182
3.6.5 Logging-Daten (nur Hinweis).....	183
3.6.6 IPv6-Fähigkeit.....	183
3.6.7 Netzwerkmanagement für IRS.....	183
3.6.8 ICS-seitiges Systemmanagement	183
3.6.9 Security	184
3.6.9.1 Security Server	184
3.6.9.2 Schnittstellen der Versuchszentrale.....	184
4 Zusammenfassung	187
Anhang 1 Die Schnittstelle zu OEM-Sonderfunktionen	188
Anhang 2 Schnittstellenspezifikation Komponente „Navi/Karte“	189
Anhang 3 Abkürzungen.....	210

Abbildungen

Abbildung 1-1: Gesamtarchitektur in sim ^{TD}	4
Abbildung 1-2: System- und Funktionskomponenten auf der IRS für die drahtgebundene Kommunikation zwischen IRS und IGLZ.....	6
Abbildung 1-3: System- und Funktionskomponenten in der IGLZ für die drahtgebundene Kommunikation zwischen IRS und IGLZ.....	7
Abbildung 1-4: Funktion Lichtsignalanlagen Netzsteuerung	7
Abbildung 1-5: Funktion Lokale verkehrsabhängige LSA-Steuerung.....	8
Abbildung 1-6: Funktion Lokale verkehrsabhängige LSA-Steuerung im Testfeld	9
Abbildung 1-7: Systemarchitektur Bereich Infrastruktur – Komponentensicht	10
Abbildung 1-8: Systemarchitektur Bereich Infrastruktur - Datensicht.....	12
Abbildung 1-9: UML-Spezifikation für die Schnittstelle IRS-LSA im Testfeld	13
Abbildung 1-10: Gesamtschaubild ITS Vehicle Station	14
Abbildung 1-11: IRS-Teilsysteme und deren Schnittstellen	16
Abbildung 2-1: Kommunikationskomponenten des Software stacks und deren Schnittstellen	20
Abbildung 2-2: Versenden von Nachrichten aus einer Funktion	23
Abbildung 2-3: Empfangen von Nachrichten.....	24
Abbildung 2-4 Übersicht ITS IMT Public	27
Abbildung 2-5: Protokollstack ITS IMT Public	29
Abbildung 2-6: Übersicht Kommunikation mittels ITS IMT Public	29
Abbildung 2-7: Übersicht IPv6 Tunneling bei ITS IMT Public.....	30
Abbildung 2-8: Übersicht Anpassung C2X Header an ITS IMT Public.....	31
Abbildung 2-9: Übersicht Kommunikation mittels Consumer-WLAN.....	33
Abbildung 2-10: Ad-hoc zwischen ITS Vehicle Stations.....	34
Abbildung 2-11: Ad-hoc zwischen ITS Vehicle Station und ITS Roadside Station	34
Abbildung 2-12: Ad-hoc zwischen ITS Vehicle Station und WLAN AccessPoint	35
Abbildung 2-13: Prüfstandinterface	37
Abbildung 2-14: Überblick über den Prüfstand.....	38
Abbildung 2-15 Interface TraceProvider.....	39
Abbildung 2-16 Interface TracePlayer.....	39
Abbildung 2-17 Interface TraceConnector	40

Abbildung 2-18: Ablaufdiagramm zur Tracewiedergabe	41
Abbildung 2-19: Komponentendiagramm der Versuchsdaten-Architektur	42
Abbildung 2-20: Vehicle/Roadside CU	43
Abbildung 2-21: Vehicle/Roadside AU	44
Abbildung 2-22: ITS Central Station.....	45
Abbildung 2-23: Sequenzdiagramm der Messdatenübermittlung	49
Abbildung 2-24: Prozessübersicht bei der Versuchsdurchführung im Bezug auf die Logdaten	51
Abbildung 2-25: Übersicht über die Datenströme zwischen den einzelnen Stations- und den Logdatenspeichern unter Berücksichtigung der eingesetzten Verschlüsselungen	52
Abbildung 2-26: Das Interface für die Übergabe von Werten an das Monitoring.....	53
Abbildung 2-27: Sequenzdiagramm der Livedatenübermittlung	55
Abbildung 2-28: Interfaces für die Darstellung auf dem Leitstand.....	56
Abbildung 2-29: Beispiel zum Bearbeiten eines Versuchsszenarios	56
Abbildung 2-30: Beispiel zur Versuchsdurchführung	57
Abbildung 2-31: Komponenten Übersicht für die Versuchsunterstützung.....	58
Abbildung 2-32: Interfaces des Flottenmanagements.....	59
Abbildung 2-33: Zugriff auf die Schnittstellen des Flottenmanagements	59
Abbildung 2-34: An- und Abmeldung von Fahrzeugen	60
Abbildung 2-35: An- und Abmeldung von Fahrern	61
Abbildung 2-36: Beispiel für Versuchsverwaltung	62
Abbildung 2-37: Zugriff auf die Schnittstellen der Versuchsplanung	63
Abbildung 2-38: Beispiel für endgültige Fahrzeugzuordnung.....	64
Abbildung 2-39: Beispiel für den Szenario Editor	64
Abbildung 2-40: Interfaces der Versuchssteuerung	66
Abbildung 2-41: Interfaces der Testablaufplanung.....	66
Abbildung 2-42: Interface des Map Servers	66
Abbildung 2-43: LifecycleInterface in der AU	67
Abbildung 2-44: Ablauf der Versuchsplanung	69
Abbildung 2-45: Interface der AU Steuerung	69
Abbildung 2-46: Versuchsdurchführung	71
Abbildung 2-47: Grobe IT-Sicherheitsarchitektur	73

Abbildung 2-48: Einsatz von kommerziellem WLAN IEEE 802.11 b/g.....	78
Abbildung 3-1: Hardwareüberblick IVS CCU.....	83
Abbildung 3-2: IVS CCU Systemarchitektur.....	84
Abbildung 3-3: GPS-Subsystem.....	85
Abbildung 3-4: Management und Facilities	91
Abbildung 3-5: C2X-Communication Subsystem – SIM-NET	93
Abbildung 3-6: CCU – UMTS- und Consumer-WLAN-Subsystem	96
Abbildung 3-7: Konzept des Fahrzeugzugriffs	98
Abbildung 3-8: Generierung der VAPI.....	99
Abbildung 3-9: Positionierung der Kommunikationsabsicherung durch den Security Daemon auf der CCU	101
Abbildung 3-10: sim ^{TD} Deployment	103
Abbildung 3-11: sim ^{TD} Deployment (mit Zusammenhängen zwischen den Komponenten).104	104
Abbildung 3-12: ID Teilsystem	106
Abbildung 3-13: Einordnung des SystemManager	107
Abbildung 3-14: Ablaufdiagramm bei ausgelasteter CPU	108
Abbildung 3-15: Ablaufdiagramm bei ausgelasteter Bandbreite	109
Abbildung 3-16: Komponenten der Datenbasis und Einbindung in die AU-Architektur.....	111
Abbildung 3-17: VapiClient-Bundle Interface	112
Abbildung 3-18: Sequenzdiagramm: Subscribe	112
Abbildung 3-19: Sequenzdiagramm: Request.....	113
Abbildung 3-20: Sequenzdiagramm zum Erstellen eines Java-Objektes für eine C2X-Nachricht und das Ablegen in die Umfeldtabelle.....	114
Abbildung 3-21: Sequenzdiagramm zum Erstellen einer C2X-Nachricht in einer Funktion; beispielhaft für eine Destination-Nachricht.....	115
Abbildung 3-22: Interface C2XMessageHandler	118
Abbildung 3-23: ComponentModel Umfeldtabelle	119
Abbildung 3-24: Ablauf Registrierung geänderte CAM	120
Abbildung 3-25: Ablauf Registrierung neue CAM.....	121
Abbildung 3-26: Ablauf Registrierung DEN	122
Abbildung 3-27: Ablauf Registrierung Nachbarschaftstabelle	123
Abbildung 3-28: Ablauf Registrierung andere Nachrichten	124
Abbildung 3-29: Übersicht Navigationskomponente	134

Abbildung 3-30: Übersicht Komponenten-Diagramm.....	135
Abbildung 3-31: Übersicht Klassendiagramm	136
Abbildung 3-32: Übersicht DB-Diagramm	137
Abbildung 3-33: Übersicht Datenflüsse	138
Abbildung 3-34: Übersicht Navigationsschnittstelle	139
Abbildung 3-35: Ablauf „Aktuelle Position aktualisieren“	140
Abbildung 3-36: Ablauf „Logging-Einträge“	141
Abbildung 3-37: Ablauf „Zieleingabe“	142
Abbildung 3-38: Ablauf „Routenkriterien“	142
Abbildung 3-39: Ablauf „Routenberechnung“	143
Abbildung 3-40: Ablauf „Routenführung“	143
Abbildung 3-41: Plausibilitätsprüfer auf der Vehicle AU	146
Abbildung 3-42: Einordnung der Komponente „Positionskettenverarbeitung“	147
Abbildung 3-43: Schema einer Prioritätentabelle mit Prioritätsbereichen für Funktionen/Anwendungsfälle (Quelle: Deliverable D22.2)	149
Abbildung 3-44: Struktur einer Funktionskoordination.	151
Abbildung 3-45: Einordnung der Komponente „Verkehrslage“	152
Abbildung 3-46: UML-Modell der Verkehrslage	152
Abbildung 3-47: Schnittstellen der Verkehrslagedatenbank.....	154
Abbildung 3-48: Ablaufdiagramm „SituationenErfragen“	154
Abbildung 3-49: Ablaufdiagramm „SituationHinzufügen“	155
Abbildung 3-50: Einordnung der „KantenEinaerbung“	155
Abbildung 3-51: Ablaufdiagramm „KantenEinaerbung_Einfärben“	156
Abbildung 3-52: Einordnung und Schnittstellen der Komponente „Fahrerverhaltensvorhersage“	157
Abbildung 3-53: Ablaufdiagramm der Komponente „Fahrerverhaltensvorhersage“.....	158
Abbildung 3-54: IRS Land Hessen mit Anbindung VsZ.....	161
Abbildung 3-55: IRS Stadt Frankfurt mit Anbindung an IGLZ und VsZ	162
Abbildung 3-56: Logischer Aufbau einer IRS	163
Abbildung 3-57: Roadside AU Deployment-Diagramm	164
Abbildung 3-58: Sequenzdiagramm IRS_GPSProxy auf IRS	166
Abbildung 3-59: Datenaustausch: VsZ – IRS	167

Abbildung 3-60: Datenaustausch: IRS – VsZ.....	168
Abbildung 3-61: CoMa-Kommunikation IRS – IGLZ.....	169
Abbildung 3-62: IRSGeoCast	170
Abbildung 3-63: IRS Management Center in der VsZ	174
Abbildung 3-64: Physikalische Übersicht des IRSMC	175
Abbildung 3-65: Deployment-Diagramm der Versuchszentrale	178
Abbildung 3-66: Empfang von C2XMessages über IRSProxy	179
Abbildung 3-67: IGLZ-Systemarchitektur	180
Abbildung 3-68: Absicherung der Versuchszentrale	185
Abbildung 4-1: Übersicht Position	190
Abbildung 4-2: Übersicht POI	191
Abbildung 4-3: Übersicht Karten-Segment.....	193
Abbildung 4-4: Übersicht Routendaten	194
Abbildung 4-5: Übersicht Instruction	195
Abbildung 4-6: Übersicht ADAS Horizon.....	196

Tabellen

Tabelle 1: Zuordnung von Kommunikationsarten zu Funktionen	25
Tabelle 2: Definition Position.....	189
Tabelle 3: Definition Address	189
Tabelle 4: Definition GeoPosition	189
Tabelle 5: Definition POI	190
Tabelle 6: Definition POICategory.....	190
Tabelle 7: Definition Layer.....	190
Tabelle 8: Definition Link	191
Tabelle 9: Definition ShapePoint	191
Tabelle 10: Liste von Link-Attributen	192
Tabelle 11: Definition Route	193
Tabelle 12: Definition RouteLeg	193
Tabelle 13: Definition RouteLink	194
Tabelle 14: Definition Fahranweisung	194
Tabelle 15: Definition Horizon	195
Tabelle 16: Definition HorizonLink.....	195
Tabelle 17: Definition Logging-Eintrag bei POIs	197
Tabelle 18: Definition Logging-Eintrag bei einer Routenberechnung	197
Tabelle 19: Definition Logging-Eintrag beim Aktivieren/Deaktivieren der Routenführung....	198
Tabelle 20: Definition Logging-Eintrag beim Starten/Stoppen der Navigation	198
Tabelle 21: Navigationsschnittstelle: Zieleingabe	199
Tabelle 22: Navigationsschnittstelle: Routenplanung, Routenberechnung, Routenführung	201
Tabelle 23: Navigationsschnittstelle: Positionierung	202
Tabelle 24: Navigationsschnittstelle: Kartenschnittstelle und POIs.....	207
Tabelle 25: Navigationsschnittstelle: Elektronischer Horizont und Kartenzugriff	209
Tabelle 26: Navigationsschnittstelle: Versuchsplanung	209
Tabelle 27 Navigationsschnittstelle: Events und Listener	210

Zusammenfassung

Deliverable D21.2 spezifiziert den konsolidierten Systementwurf für sim^{TD}, der die Grundlage für die Umsetzung des Gesamtsystems sowie die Implementierung der einzelnen Subsysteme bildet. Gleichzeitig ist Deliverable D21.2 eines der zentralen Ergebnisse für den Meilenstein MS2 („Systemarchitektur erstellt“), zusammen mit den folgenden weiteren vertiefenden Deliverables aus Teilprojekt TP2:

- D21.3: Spezifikation der sim^{TD}-Funkschnittstellen
- D21.4: Spezifikation der Kommunikationsprotokolle
- D21.5: Spezifikation der IT-Sicherheitslösung
- D22.2: Formale Spezifikation der HMI-Lösung
- D23.1: Beschreibung der Versuchszentrale

Die einzelnen Subsysteme der Systemarchitektur von sim^{TD} leiten sich aus der Projektstruktur von Teilprojekt TP2 ab:

- Fahrzeugseitiges Subsystem, bestehend aus ITS Vehicle Station und Human Machine Interface (HMI)
- Infrastrukturseitiges Subsystem, bestehend aus ITS Roadside Station (IRS) und der Versuchszentrale (ITS Central Station (ICS)), den LSA-Steuergeräten und der Integrierten Gesamtverkehrsleitzentrale der Stadt Frankfurt am Main (IGLZ)).
- Testsystem, integriert in die Versuchszentrale.

In diesem Deliverable werden die einzelnen Subsysteme spezifiziert, was sowohl Hardwarekomponenten als auch Software-/Systemkomponenten betrifft. Ebenfalls wird spezifiziert, wie diese Subsysteme und deren Komponenten miteinander interagieren. Die Entwicklung der Funktionalität der einzelnen Systemkomponenten wurde eng mit den Funktionsentwicklungsteams und den in Teilprojekt TP1 erarbeiteten Anforderungen abgestimmt. Damit kann davon ausgegangen werden, dass mit dem in diesem Deliverable spezifizierten Basissystem die erforderliche Funktionalität bereitgestellt wird, die von den umzusetzenden Funktionen benötigt wird.

Für die formale Modellierung des Systems und seiner Bestandteile wurde im Rahmen der Systemarchitekturentwicklung in Teilprojekt TP2 ein UML-Modell entwickelt. Die Abbildungen in Deliverable D21.2 sind „Ausleitungen“ aus diesem UML-Modell, das TP2-übergreifend für die Spezifikation der Komponenten Verwendung findet. In dieses Modell werden zukünftig auch die Feinspezifikation der einzelnen Funktionen integriert, wodurch mit dem finalen Stand ein einheitliches übergreifendes Systemmodell stehen wird, das für Konsistenzprüfungen und für die Umsetzung herangezogen wird.

Da Deliverable D21.2 die Grundlage für die Umsetzung des Gesamtsystems in sim^{TD} bildet, ist es (zusammen mit den oben genannten Deliverables) ein zentrales Dokument, das für zahlreiche Aktivitäten in sim^{TD} von Bedeutung ist.

English summary

Deliverable D21.2 specifies the consolidated system architecture design in sim^{TD}. This specification defines the basis for the implementation of the overall sim^{TD} system as well as the subsystems defined. Deliverable D21.2 also summarizes the major result for Milestone MS2 ("System Architecture specified"), together with the following deliverables in TP2:

- D21.3: Specification of the sim^{TD} Radio Interfaces
- D21.4: Specification of Communication Protocols
- D21.5: Specification of IT Security
- D22.2: Formal Specification of HMI
- D23.1: Description of Test Site

The very basic subsystems of the sim^{TD} system architecture are derived from the internal project structure of TP2:

- Vehicleside subsystem, consisting of ITS Vehicle Station and Human Machine Interface (HMI)
- Roadside subsystem, consisting of ITS Roadside Station and the so called Test Center (also called ITS Central Station)
- Test system, integrated into the Test Center.

This deliverable specifies the functionality the subsystems mentioned above including their basic interactions. This specification addresses hardware issues as well as software and component issues. The specification itself was developed in intensive discussions with the application development teams and also includes the requirements developed in TP1. Hence, the component specification in this deliverable can be considered as a commonly accepted basis system, which provides the functionality required by the applications being developed.

These agreements and discussions are the basis for the specification of the overall system, its subsystems, components and interfaces, which are described in this deliverable. Besides the specification activities, the system and its components were also modeled in an overall UML model. The figures used in this deliverable are derived from this UML model, which will be also used as a basis for the specification of the different applications deployed in sim^{TD}. This way, the UML model represents the overall system architecture, which will be used for consistency checks as well as for the implementation and deployment of the system architecture.

Together with the above mentioned deliverables, Deliverable D21.2 is the basis for the deployment of the overall system in sim^{TD}. Hence, it is an important document for several continuing activities in sim^{TD}.

1 Einführung

Das vorliegende Deliverable D21.2 beschreibt die finale Spezifikation der Systemarchitektur in sim^{TD}. Deliverable D21.2 entstand in mehreren Iterationsschritten (Arbeitsdokument W21.2, Deliverable D21.2 (31/05/2009 Draft), Deliverable D21.2 (31/07/2009 Draft)), in denen die Ergebnisse einer engen Abstimmung zwischen Teilprojekt TP2, Arbeitspaket AP11 und den Funktionsentwicklungsteams einflossen. So konnte ein konsistenter Stand zwischen den Anforderungen und den Systemkomponenten erreicht werden. Diese Iterationsschritte wurden begleitet von einem intensiven Austausch zwischen TP1 und TP2 im Rahmen des Architekturentwicklungsworkshops. Auf dem TP1/TP2-Synchronisationsworkshop am 19./20.08.2009 in Kaiserslautern fand die abschließende Abstimmung statt zwischen TP1 und TP2 statt, bei der die letzten offenen Fragen geklärt wurden. Die Systemarchitektur wurde im Vorfeld mit den Anforderungen aus Teilprojekt TP1 und den Funktionsentwicklungsteams abgestimmt. Damit kann davon ausgegangen werden, dass die Systemarchitektur vollständig ist und die Anforderungen der Partner in sim^{TD} abdeckt. Gleichzeitig ist diese Spezifikation auch die Grundlage für die Umsetzung des Gesamtsystems sowie die Implementierung der Teilsysteme und Komponenten; sie ist auch eine wichtige Grundlage für die weiteren Aktivitäten in den übrigen Teilprojekten.

In diesem Dokument werden zahlreiche Begriffe verwendet, die eine sim^{TD}-spezifische Bedeutung haben und in dieser Form auch projektweit verwendet werden. Diese Begriffe werden hier nicht weiter erläutert; ihre Definition findet sich im projektweiten Glossar, das eine übergreifende Verwendung sichergestellt.

Der Aufbau orientiert sich prinzipiell an der Strukturierung des gesamten Systems, wobei übergreifende Aspekte separat in einem Kapitel behandelt werden. Der Rest von Kapitel 1 gibt einen zusammenfassenden Überblick über das Gesamtsystem, dessen Bestandteile in den nachfolgenden Kapiteln vertieft werden. Abschnitt 2 beschreibt die systemübergreifenden Aspekte, die für das Verständnis der Zusammenhänge (sowie für Spezifikation der Funktionalität einzelner Komponenten) erforderlich sind. Kapitel 3 widmet sich schließlich den einzelnen Teilsystemen. Eine kurze Zusammenfassung samt einem Ausblick auf die zukünftigen Aktivitäten findet sich in Kapitel 4.

1.1 Systemüberblick

Die nahtlose Vernetzung von Fahrzeugen und Infrastruktur stellt eine technologische und organisatorische Herausforderung dar. Diese Herausforderung leitet sich aus der Komplexität der heutigen Transportsysteme ab und röhrt aus der Tatsache das verschiedenste Kommunikationswege, Datenformate und Akteure involviert sind.

Abbildung 1-1 zeigt die Systemarchitektur in sim^{TD} aus der abstrakten Perspektive der unterschiedlichen Subsysteme, deren Komponenten und Interaktionen. Generell lässt die Architektur in zwei Subsysteme einteilen:

- Das fahrzeugseitige Subsystem beschreibt alle Hardware- und Softwarekomponenten, die auf dem Fahrzeug zu Einsatz kommen.
- Das Infrastruktur-Subsystem setzt sich zusammen aus dem Infrastrukturteil der Stadt Frankfurt am Main und des Landes Hessen (HLSV).

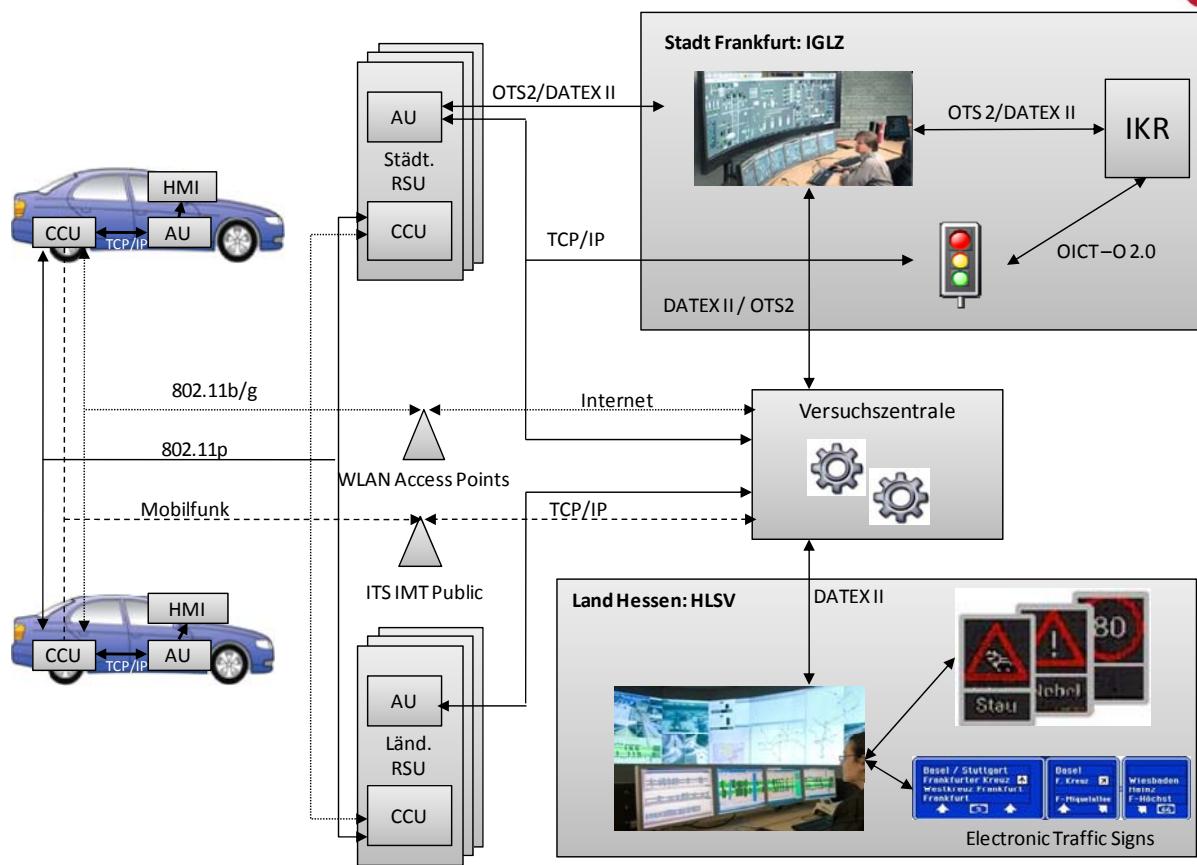


Abbildung 1-1: Gesamtarchitektur in sim^{TD}

Die sim^{TD}-Architektur integriert grundsätzlich drei verschiedene Kommunikationskanäle, die sich in Technologie und Funktionalität unterscheiden:

- Die direkte Kommunikation zwischen Fahrzeugen untereinander sowie Fahrzeugen und Infrastruktur erfolgt über den Kommunikationsstandard IEEE 802.11p (ITS G5A). Die meisten Meldungen zur Sicherheits- und Verkehrseffizienz erfordern eine niedrige Latenz und werden daher über diese Verbindung realisiert.
- Für die direkte Kommunikation sind alle Fahrzeuge weiterhin mit IEEE 802.11 b/g Modulen ausgestattet. Fahrzeuge nutzen IEEE 802.11 b/g über die RSUs auch, um IP-basierte Kommunikation über das Internet mit dem Test Management Center oder mit anderen Backend-Diensten herzustellen. Zusätzlich werden WLAN Access Points verwendet um große Datenmengen (z.B.: Messdaten) an die Versuchszentrale zu übermitteln.
- IP-basierte Kommunikation ist möglich über das gesamte sim^{TD}-Testgebiet unter Verwendung von GPRS, EDGE, UMTS oder HSPA. Zugangspunkte für zellularen Mobilfunk sind in Abbildung 1-1 ETSI-konform als ITS IMT Public gekennzeichnet. Über diesen Link wird eine Weiterleitung und Verteilung von Nachrichten zur Sicherheits- oder Verkehrseffizienz zwischen den Autobahn- und Stadt-Szenarien realisiert werden. Auch das Test Management Center nutzt Mobilfunk, um z.B. Fahreranweisungen an die Fahrer zu übertragen.

Die Verarbeitung aller Nachrichten bis zur Netzwerkschicht wird auf Fahrzeugseite (ITS Vehicle Station) von der Vehicle CCU (Communication & Control Unit) behandelt. Daten vom CAN-Bus eines Fahrzeugs werden von einer zentralen Komponente aufgearbeitet und den Funktionen zur Verfügung gestellt.

Für die ITS Roadside Station ist bis auf einige zusätzliche Schnittstellen die gleiche Architektur mit Aufteilung in CCU und AU (Application Unit) vorgesehen. Im Gegensatz zu ITS Roadside Stations entlang der Autobahnen können städtische ITS Roadside Stations das Verkehrsgeschehen direkt über eine Verbindung zu Lichtsignalanlagen (LSA) steuern.

Die Kommunikationswege in sim^{TD} sind prinzipiell verschiedenen Angriffen auf die Datensicherheit und Privatsphäre der Fahrer ausgesetzt. Um zu verhindern, dass verfälschte Nachrichten Messergebnisse verfälschen und um sicherheitskritische Angriffe zu verhindern, werden eingehende und ausgehende Nachrichten verschieden Sicherheitsüberprüfungen unterzogen.

In der Versuchszentrale laufen Datenströme aus IGLZ, VZH sowie die über Mobilfunk und WLAN transportierte Daten zur Auswertung und weiteren sim^{TD}-Versuchssteuerung zusammen. Als systemübergreifendes Konzept ist das Testsystem tief in den fahrzeug- und infrastrukturseitigen Subsystemen integriert.

1.2 Teilsysteme

1.2.1 Überblick Infrastruktur

1.2.1.1 Drahtgebundene Schnittstellen (Anbindung IRS, LSA, IGLZ, VsZ)

Die Systemsicht auf die drahtgebundenen Schnittstellen (d.h. auf die Infrastrukturseite) wird maßgeblich durch die beiden Funktionen „Lichtsignalanlagen Netzsteuerung“ (F 1.3.2) und „Lokale verkehrsabhängige LSA-Steuerung“ (F 1.3.3) bestimmt. Diese werden im Folgenden kurz dargestellt.

Funktion Lichtsignalanlagen-Netzsteuerung (F 1.3.2)

Die Funktion Lichtsignalanlagen-Netzsteuerung besitzt Funktionsanteile sowohl in der IRS als auch in der IGLZ. Die städtischen IRS sind über das städtische Kommunikationsnetz an die IGLZ angeschlossen, wobei die Kommunikation über das TCP-Protokoll abgewickelt wird. In der IRS gibt es dafür neben der Funktionskomponente DRIVERSDatenaufbereitung die OTS2-Komponente, die die Kommunikation der IRS zur IGLZ übernimmt (Abbildung 1-2). Eingesetzt wird dabei das OTS2-Protokoll.

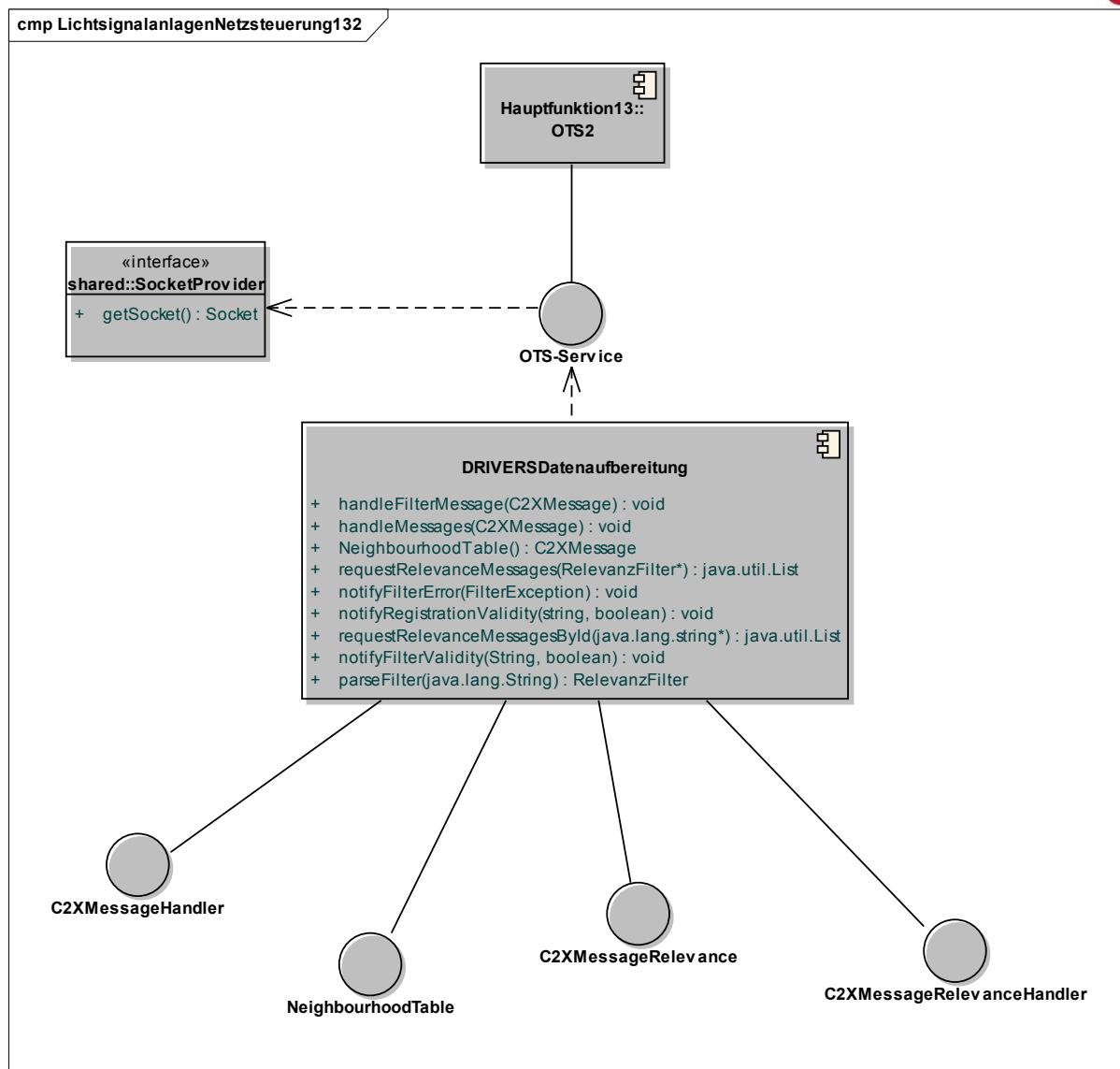


Abbildung 1-2: System- und Funktionskomponenten auf der IRS für die drahtgebundene Kommunikation zwischen IRS und IGLZ

In der IGLZ existiert eine entsprechende Gegenstelle der drahtgebundenen Kommunikationsschnittstelle (OTS2-Komponente), die die Kommunikation mit den städtischen IRS regelt (Abbildung 1-3). Ferner sind die Funktionskomponenten der Funktion 1.3.2 dargestellt (DRIVERS, BALANCE).

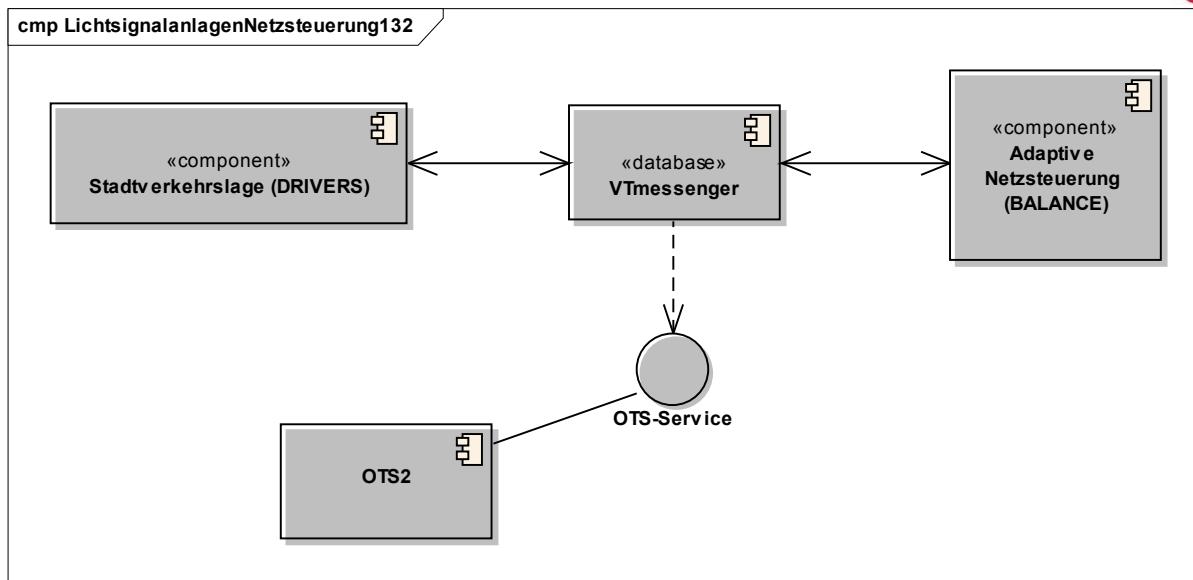


Abbildung 1-3: System- und Funktionskomponenten in der IGLZ für die drahtgebundene Kommunikation zwischen IRS und IGLZ

Einen Überblick über das Gesamtsystem im Rahmen der Funktion 1.3.2 gibt die folgende Grafik.

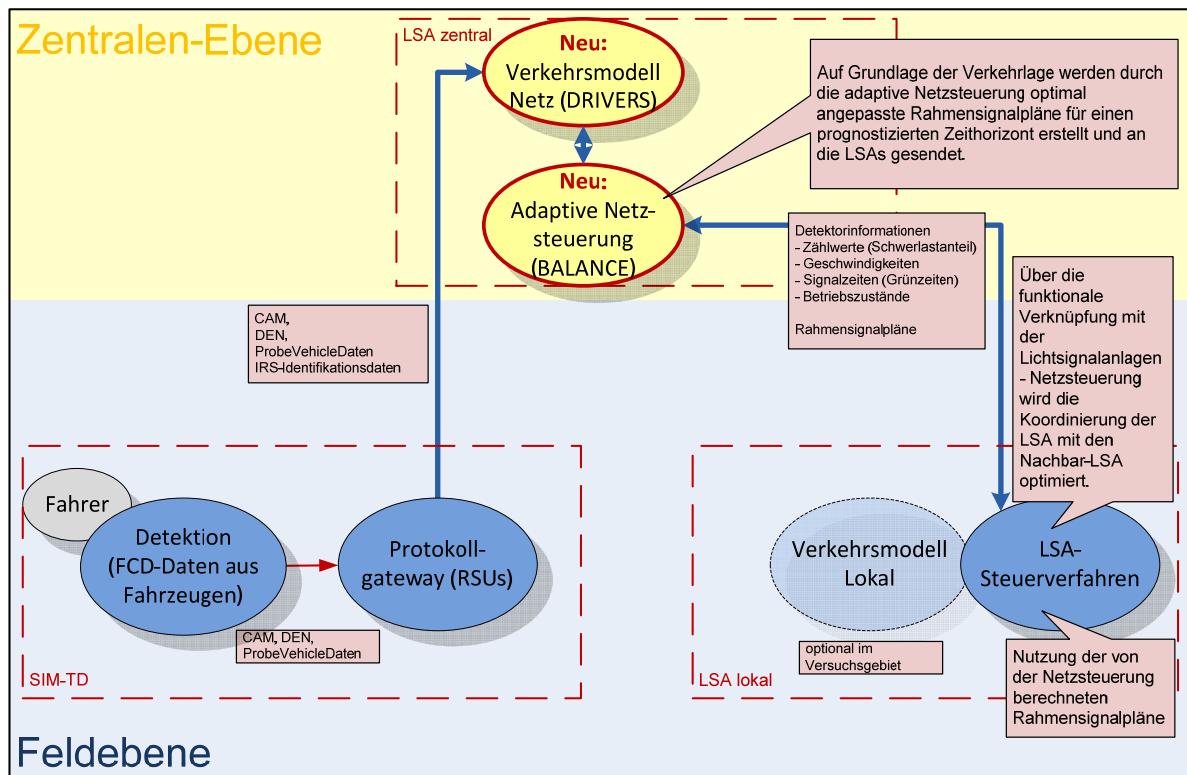


Abbildung 1-4: Funktion Lichtsignalanlagen Netzsteuerung

Über infrastrukturseitige und fahrzeugseitige Erfassungseinrichtungen (z.B. Detektoren oder IVS) wird die aktuelle Verkehrslage im Straßenverkehrsnetz erfasst und in einem Verkehrsmodell (DRIVERS) zu einer räumlich-zeitlichen Repräsentation der Verkehrslage fusioniert.

Mit Hilfe eines Wirkungsmodells (BALANCE) werden die Auswirkungen verschiedener Steuerungsalternativen für jede LSA für die nächsten 5 bis 15 Minuten prognostiziert und dahingehend optimiert, dass eine minimale Verlustzeit sowohl für den ÖPNV als auch für den motorisierten Individualverkehr (mIV) und den nicht motorisierten Individualverkehr (nmIV) entsteht. Die hieraus entstehenden optimalen Rahmensignalpläne werden an die Lichtsignalanlagen geschickt, die diese unmittelbar umsetzen und dadurch den Verkehrsfluss optimieren.

Die Einbindung neuartiger Detektionsverfahren (Fahrzeuge mit ITS Vehicle Stations (IVS)) in bestehende lokale und adaptive Steuerungsverfahren ist ein Schwerpunkt dieser Funktion. Besonderes Augenmerk muss auf der korrekten Lokalisierung der Fahrzeuge liegen und damit auf einer korrekten Zuordnung zur entsprechenden Signalgruppe.

Funktion Lokale verkehrsabhängige LSA-Steuerung (F 1.3.3)

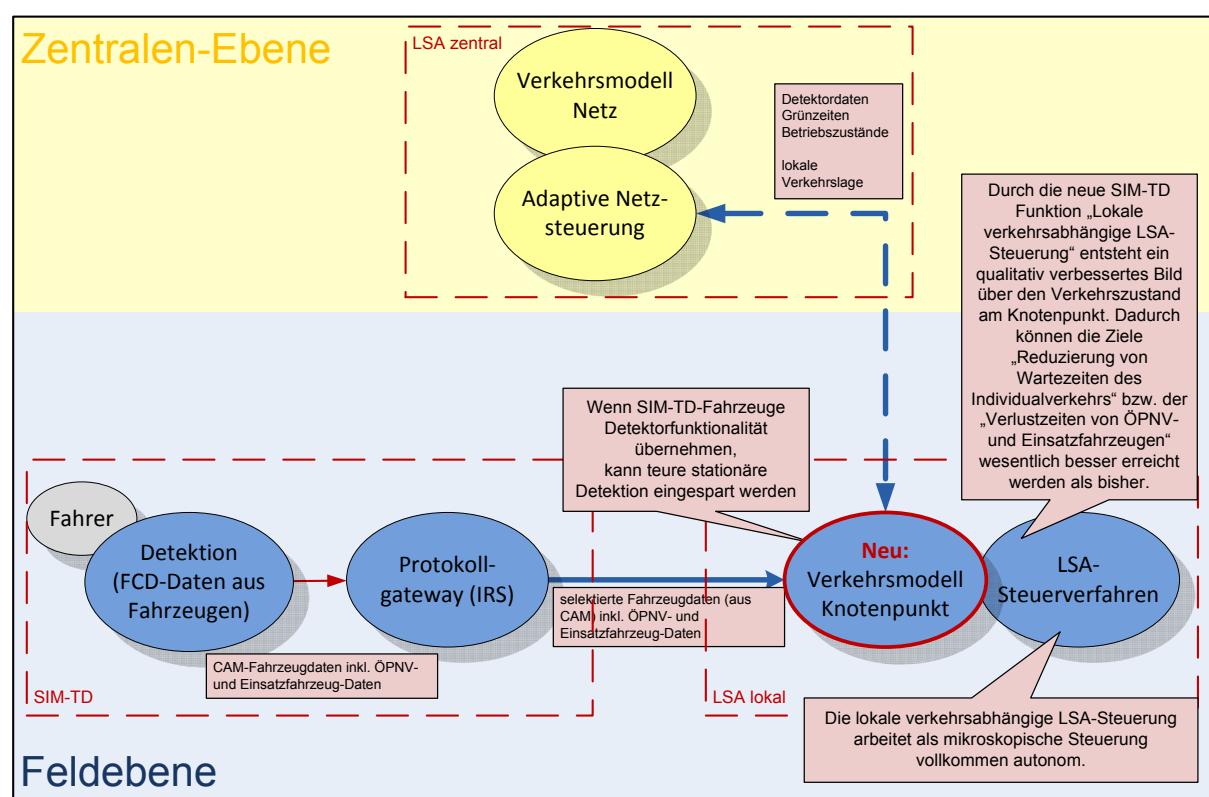


Abbildung 1-5: Funktion Lokale verkehrsabhängige LSA-Steuerung

Durch Kenntnis der genauen Position, Geschwindigkeit und weiterer Fahrzeugdaten des Individualverkehrs, öffentlichen Verkehrs und der Einsatzfahrzeuge (vgl. auch entsprechende Anwendungsfälle) kann die LSA für viele verkehrliche Situationen besser als bisher eine geeignete Steuerungsstrategie erzeugen und lokal umsetzen.

Die lokale verkehrsabhängige LSA-Steuerung arbeitet in Bezug auf ihren Einsatz am Knotenpunkt vollkommen autonom. Über die funktionale Verknüpfung mit der Lichtsignalanlagen Netzsteuerung wird ihre Zusammenarbeit mit den Nachbarknotenpunkten optimiert.

Ein Schwerpunkt der technischen Aspekte dieser Funktion ist die Integration von neuartigen Detektionsverfahren (hier: Fahrzeuge mit IVS) in bestehende lokale (z.B. VS-Plus) und netzweite Steuerverfahren. Zur Nutzung der Fahrzeuge als Detektoren muss ein besonderes Augenmerk des Forschungsprojektes auf der korrekten Georeferenzierung der Fahrzeuge

(Fahrstreifen, Fahrtrichtung, Geschwindigkeit) liegen sowie für diese Daten bzw. Parameter eine geeignete Kodierung auf Seiten von ITS Roadside Station (IRS), IVS und ggf. weiteren Teilsystemen spezifiziert werden.

Die folgende Abbildung zeigt die beteiligten Komponenten für die Funktion „Lokale Verkehrsabhängige LSA-Steuerung“ im Testgelände.

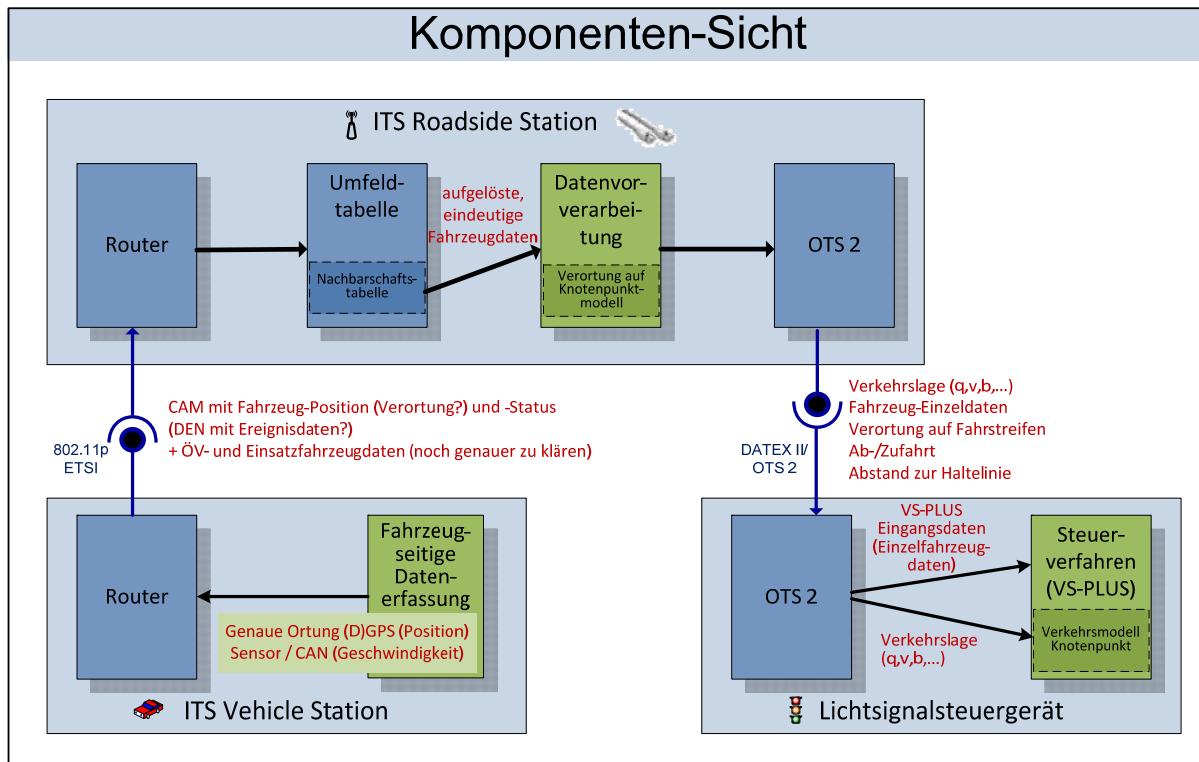


Abbildung 1-6: Funktion Lokale verkehrsabhängige LSA-Steuerung im Testfeld

Systemarchitektur: Komponentensicht

Der Bereich der Infrastruktur ist in der folgenden Abbildung dargestellt. Er ist aufgeteilt in die beiden Hoheitsbereiche der Stadt Frankfurt am Main (grün) und des Landes Hessen (grau).

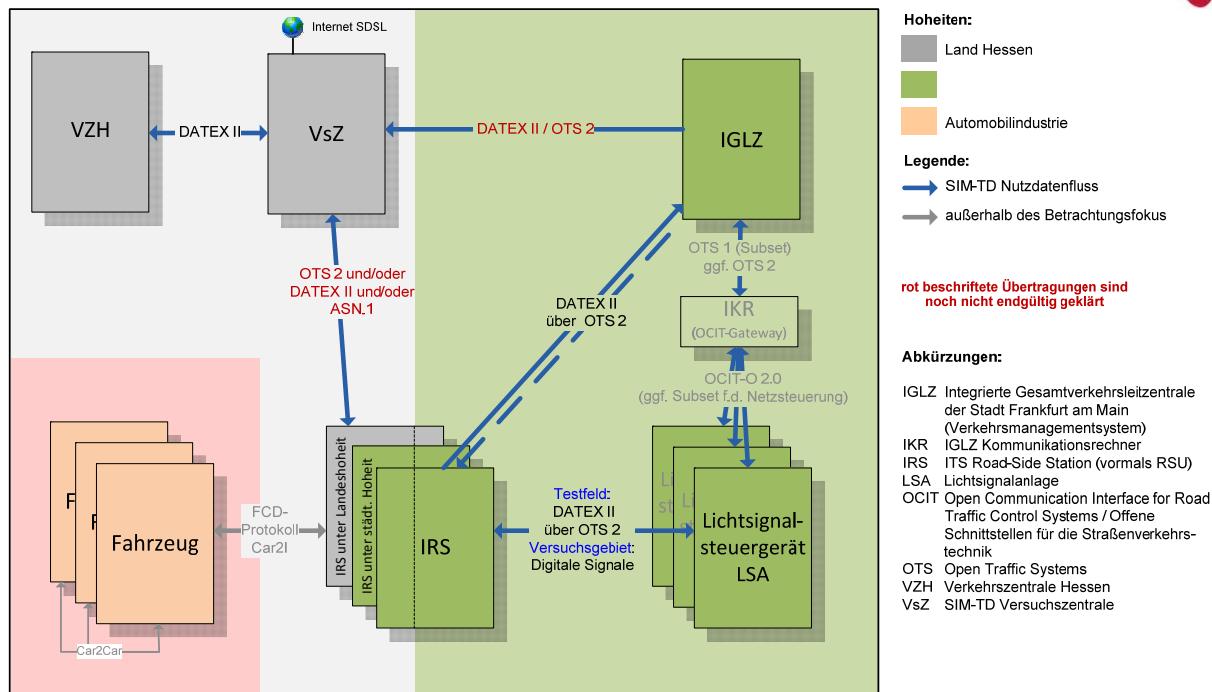


Abbildung 1-7: Systemarchitektur Bereich Infrastruktur – Komponentensicht

Neue Schnittstellen

Kernkomponenten auf der Infrastrukturseite sind die IRS, die jeweils entweder von der Stadt Frankfurt am Main oder dem Land Hessen verwaltet werden. Entsprechend dieser Varianten sind sie auch unterschiedlich angeschlossen:

- IRS-Hoheit der Stadt Frankfurt am Main
 - Die IRS wird einen Anschluss an das Verkehrsmanagement der Stadt Frankfurt am Main (IGLZ: Integrierte Gesamtverkehrsleitzentrale) besitzen. Diese Schnittstelle resultiert aus den Anforderungen der Netzsteuerungsfunktion und soll einen direkten Datenaustausch von Daten in die Zentrale ermöglichen. Diese Schnittstelle wird das OTS2-Protokoll und das Datenmodell DATEX II nutzen.
 - Die IRS ist an der jeweils zugehörigen Lichtsignalanlage (bzw. genauer deren Steuergerät) angeschlossen. Diese Verbindung dient der lokalen verkehrsabhängigen Steuerung, bei der das Steuergerät ohne direkte Beteiligung einer Zentrale auf neue Detektionsinformationen der Fahrzeuge zugreifen kann. Außerdem können über diese Schnittstelle LSA-Daten für Funktionen wie z.B. den Ampelphasen-Assistenten bereitgestellt werden. Diese Schnittstelle erfährt eine unterschiedliche Ausprägung in Testfeld und Versuchsgelände:

Testgelände:

Die Schnittstelle wird das OTS2-Protokoll und das Datenmodell DATEX II über eine Ethernet-Verbindung nutzen. Die Komponente „Verkehrsmodell Knotenpunkt“ wird im Lichtsignalsteuergerät angesiedelt.

Versuchsgelände:

Da keine umfangreichen Umrüstungen an den Steuergeräten durchgeführt werden können, kann keine OTS2-Schnittstelle realisiert werden. Stattdessen wird die Komponente „Verkehrsmodell Knotenpunkt“ abweichend zu oben in der IRS

realisiert. Die Schnittstelle zum Lichtsignalsteuergerät arbeitet nun mit digitalen Signalen, auf denen Detektorsignale simuliert werden. Diese werden vom Steuergerät mittels vorhandenen Optokopplern empfangen und an das Steuerverfahren weitergeleitet.

- IRS-Hoheit des Landes Hessen

Die IRS wird einen Anschluss an die sim^{TD}-Versuchszentrale (VsZ) besitzen. Zu beachten ist, dass die Ausführung dieser Schnittstelle noch nicht endgültig geklärt ist.

Bestehende Schnittstellen

Hinweis: Die folgenden beiden Abschnitte beschreiben Infrastruktur-Teile der Stadt Frankfurt am Main, welche nicht Bestandteil des eigentlichen Forschungsprojektes sim^{TD} sind. Sie dienen nur dem fachlich interessierten Leser als zusätzliche Information. Der beschriebene Bereich ist der Grafik verblasst dargestellt.

Die Lichtsignalanlagen der Stadt Frankfurt am Main sind über Kupferleitungen an eine Lichtsignalsteuerungszentrale (IKR) angeschlossen; neben einigen noch bestehenden proprietären Verbindungen wird ein Teil der Anlagen (ca. 120 von 660) über OCIT-Outstations 1.0 angesprochen. Im Zuge von sim^{TD} könnte diese Schnittstelle auf OCIT-Outstations 2.0 erweitert werden, um zusätzlich benötigte Daten für die Netzsteuerung zu transportieren. Statt einer vollständigen Implementierung der Version 2.0 ist als schmalere Variante auch ein geeignetes Subset denkbar.

Die Lichtsignalsteuerungszentrale ist über eine OCIT-Instations 1.1 Schnittstelle an die IGLZ angeschlossen, über die ebenfalls einerseits LSA-Daten transportiert und andererseits LSAs geschaltet werden können. Auch hier ist im Zuge des Projektes die Einführung einer OTS1 oder OTS2-Schnittstelle vorgesehen, ggf. als Subset.

Systemarchitektur: Datensicht

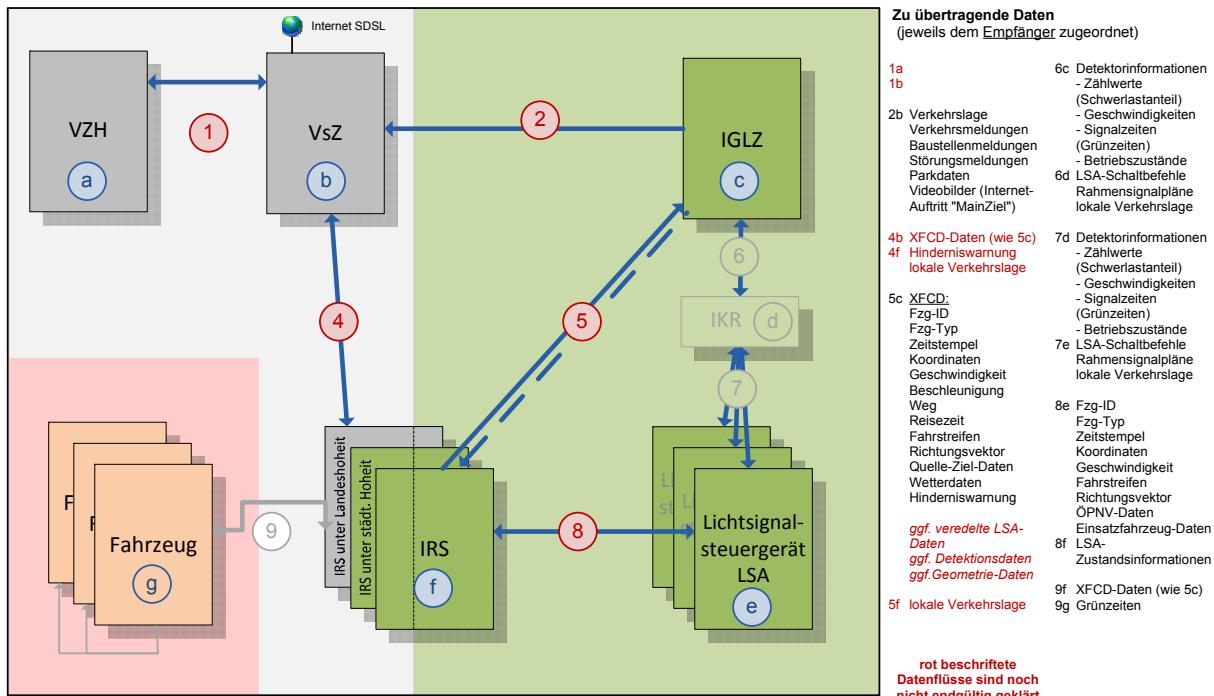


Abbildung 1-8: Systemarchitektur Bereich Infrastruktur - Datensicht

Abbildung 1-8 verdeutlicht die bislang identifizierten Datenflüsse, wobei zu beachten ist, dass es sich hierbei um eine vorläufige, ggf. noch unvollständige Darstellung handelt. Bestimmte Datenflüsse (insbesondere die rot markierten) sind noch in Abstimmung und können erst zu einem späteren Zeitpunkt endgültig festgelegt werden. Dies betrifft insbesondere die Datenflüsse IRS – VsZ – VZH aber auch bestimmte Datenarten von bzw. zur IRS.

Die Schnittstellen sind nummeriert und bilden jeweils mit einem nachgelagerten Kleinbuchstaben die Zielkomponente ab, die die Daten empfangen soll.

UML-Spezifikation

Abbildung 1-9 zeigt die UML-Spezifikation für die Schnittstelle IRS – LSA im Testfeld.

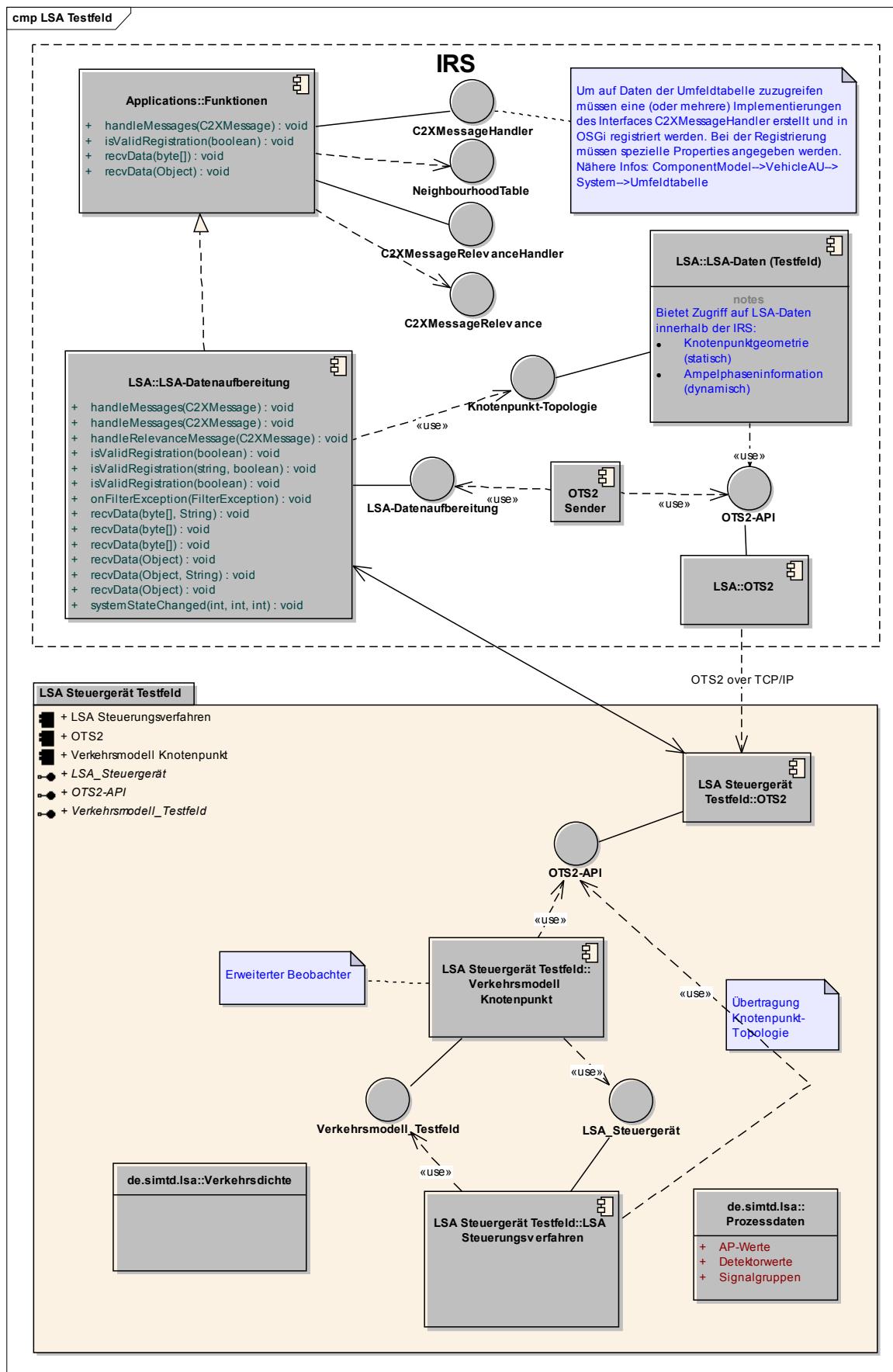


Abbildung 1-9: UML-Spezifikation für die Schnittstelle IRS-LSA im Testfeld

1.2.2 ITS Vehicle Station

Abbildung 1-10 zeigt die Grobarchitektur der ITS Vehicle Station mit ihren Subsystemen Communication Control Unit (CCU, im Folgenden auch: Router) und Application Unit (AU, im Folgenden auch: Host), sowie externen Zusatzgeräten, die an die Application Unit angebunden werden können.

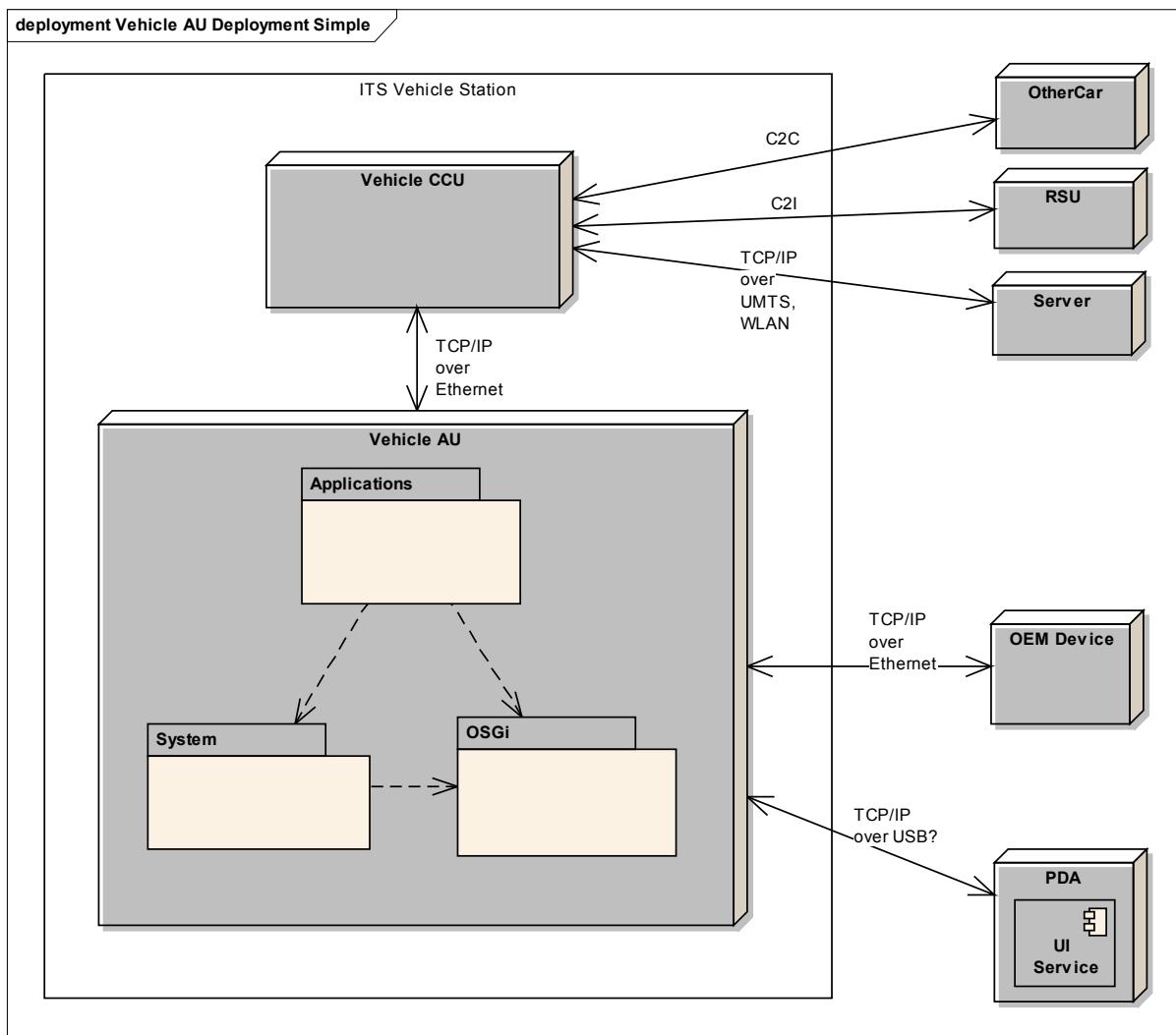


Abbildung 1-10: Gesamtschaubild ITS Vehicle Station

Alle Kommunikationsaufgaben werden im Router gebündelt. Neben Standard-Kommunikationsmodulen (z.B. WLAN, UMTS) wird hier vor allem das speziell für die Car2X-Kommunikation vorgesehene Funkmodul integriert (IST-G5A-basierte Kommunikation). Darüber hinaus werden hier die zugehörigen Protokolle und der Zugriff auf den CAN-Bus realisiert.

Auf dem Host sieht die Architektur eine Aufteilung der Software in Applikationen (weiter unterteilt in Hauptfunktionen und Funktionen), Systemkomponenten und OSGi-Komponenten mit definierten Zugriffsregeln vor, die alle innerhalb eines OSGi-Frameworks betrieben werden. Die Benutzerführung (HMI) erfolgt auf einem separaten Gerät.

Die Aufteilung in Router und Host folgt den Überlegungen von C2C-CC und ETSI. Aufgrund der vielfältigen Projektanforderungen (z.B. Preis, Robustheit, Leistung, Flexibilität, Erweiterbarkeit) wurde eine Entwicklung auf verschiedenen Hardware-Plattformen für

Router, Host und HMI beschlossen. Eine solche Aufteilung der Aufgaben gewährleistet zum einen, dass keine relevanten Projektziele aufgrund von mangelnden Systemressourcen (Rechnerleistung, Speicher, etc.) aufgegeben werden müssen. Zum anderen wird die Komplexität der Einzelsysteme in einem vertretbaren Rahmen gehalten.

Es ist vorgesehen für Host und HMI kommerziell erhältliche Komponenten (CarPC, PDA bzw. UMPC) zu verwenden. Für den Router gibt es kein entsprechendes kommerziell erhältliches System, weshalb hier ein neues Gerät entwickelt werden muss.

Sowohl Router als auch HMI sind jeweils über Ethernet mit dem Host verbunden. Die Kommunikation erfolgt über TCP/IP-Verbindungen, die jeweils direkt von Komponenten auf Router bzw. Host geöffnet werden können. Bestimmten Komponenten und dem zugehörigen Service werden dazu bestimmte Ports zugewiesen. In der Regel greifen Applikationen nicht direkt auf die Sockets zu, sondern über die zugehörigen Systemkomponenten.

Weitere externe Geräte (z.B. OEM-spezifische Hardware inklusive zugehöriger Sonderfunktionen), können ggf. über Ethernet angebunden werden. Der technische Ansatz wird im Abschnitt 3.2.13 und im Anhang beschrieben.

Der für die ITS Vehicle Station entwickelte Router findet auch in der ITS Roadside Station Verwendung.

1.2.3 ITS Roadside Station

Die IRS ist eine zentrale Komponente der Gesamtarchitektur. Als Bindeglied zwischen Fahrzeugen und Versuchszentralen erfüllt sie verschiedenste Aufgaben. Sie ermöglicht den bidirektionalen Informationsfluss zwischen Fahrzeug und Zentrale. Dabei werden zwei stark divergierende Kommunikationstechnologien miteinander verbunden. Die Kommunikation zwischen Fahrzeugen und IRS ist verbindungslos, zeitlich sehr begrenzt und verfügt über geringe Bandbreite. Die Kommunikation von IRS in Richtung Zentrale dagegen ist verbindungsorientiert und verfügt konstant über eine vergleichsweise hohe Übertragungsrate. Der große Unterschied zwischen diesen beiden Übertragungskanälen wird ebenfalls in der eingesetzten Technologie deutlich. Für die Kommunikation zwischen IRS und Zentrale kommt Ethernet/LWL, UMTS oder SDSL mit TCP/IP zum Einsatz wohingegen zwischen IRS und den Fahrzeugen ITS G5A (IEEE 802.11p) IEEE 802.11b/g genutzt wird.

Neben den Aufgaben der Protokollumsetzung und Informationsweiterleitung findet auch eine Vorverarbeitung der Daten auf der IRS statt. Diese Aufgabe übernehmen IRS-seitige Funktionsanteile, die in Teilprojekt TP1 definiert werden. Die IRS-Plattform übernimmt keinerlei Grundfunktionalitäten, wie z.B. das Verdichten von Daten, sondern stellt lediglich eine Anwendungsplattform dar. Diese Plattform enthält ein so genanntes FunctionFramework, das allen Funktionsanteilen einheitliche Schnittstellen zu den Systemressourcen und Kommunikationskanälen zur Verfügung stellt. Eingeschränkt werden die IRS-seitigen Funktionsanteile dabei prinzipiell nur durch die begrenzten Ressourcen des IRS-Systems (Speicher, CPU, etc.) sowie die endliche Bandbreite auf den Kommunikationskanälen. Die gesamte ITS Roadside Application Unit inklusive des FunctionFramework kann vollständig von der Versuchszentrale über das IRS Management verwaltet werden. Hierdurch können Funktionsanteile installiert, gestartet, gestoppt und rekonfiguriert werden. Durch die Abstraktion aller Kommunikationsschnittstellen wird größtmögliche Flexibilität und Wartbarkeit gewährleistet.

Beispiele für mögliche IRS-seitige Funktionsanteile sind:

- Store and Forward von Ereignisinformationen (Decentralized Environmental Notification – DEN).
- Weiterleitung von Ereignisinformationen an Versuchszentrale (Testzentrale) → DEN.
- Aggregation von empfangenen Fahrzeugdaten zur Verbesserung der Wetter- und Verkehrslage erfassung.
- Neue Anwendungen bzgl. der Interaktion zwischen Fahrzeug und LSA.
- Kreuzungsassistenz sowie Assistenz im Baustellenbereich.
- Verteilung von Daten der ergänzenden Dienste aus der Versuchszentrale an die Fahrzeuge.
- Versendung von Daten zur Kreuzungstopologie.

Die IRS ist ebenso wie die IVS physikalisch in zwei Teilkomponenten (Communication and Control Unit – CCU, ITS Roadside Station Application Unit – RAU) aufgeteilt. Dies ist notwendig, um die Applikationen und die Steuerung der Kommunikationswege in Richtung Fahrzeug zu trennen. Alle Funktionsanteile werden auf der RAU installiert und ausgeführt. Sie finden hier eine Laufzeitumgebung, die ihnen alle benötigten Schnittstellen zur Verfügung stellt, die sie zur Sicherstellung ihrer Funktionslogik benötigen.

Die Anbindung der IRS an die Versuchszentrale und an eine LSA sind spezifische Schnittstellen, die nur in der IRS zur Verfügung gestellt werden. Aus diesem Grund sind diese Schnittstellen direkt auf der RAU realisiert. Die Architektur der CCU kann somit unangetastet bleiben. Die Art und Weise, wie eine IRS an die Versuchszentrale angeschlossen wird, ist abhängig von ihrer Position. Diese spezifischen Architekturen sind im Detail in diesem Dokument beschrieben. Im folgenden Bild sind jedoch alle Teilkomponenten und Schnittstellen aufgezeigt, die in jeder IRS zur Verfügung stehen. Die städtischen IRSs sind darüber hinaus an die IGLZ angeschlossen, auf der auch Funktionsanteile umgesetzt werden. Dies ist in Abbildung 1-11 nicht dargestellt.

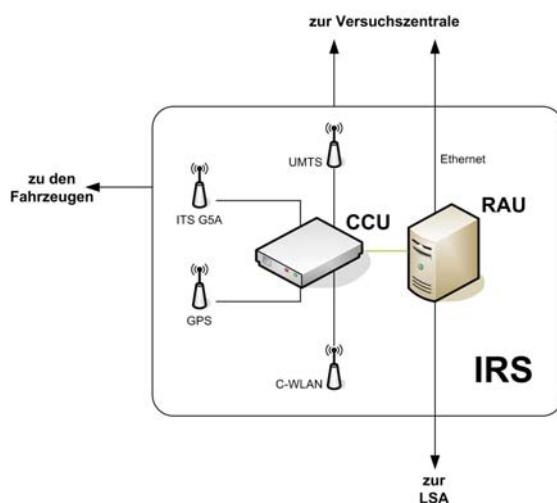


Abbildung 1-11: IRS-Teilsysteme und deren Schnittstellen

Die CCU-Komponente wird auf den beiden Plattformen IVS und IRS eingesetzt. Dabei wird bereits berücksichtigt, dass diese Komponente so generisch wie möglich konstruiert wird, um sie in beiden Systemen ohne große Änderungen einsetzen zu können. Dies betrifft jedoch

nur die Software, da die CCU-Hardware auf beiden Plattformen (IVS, IRS) identisch sein wird.

Der einzige Unterschied, aus Sicht der Funktionen zwischen IVS CCU und IRS CCU wird die Bereitstellung und Beschaffung der Positionsinformationen sein. Die Abfrage dieser Informationen über die VAPI steht in dieser Form auf der IRS CCU nicht zur Verfügung. Die VAPI ermöglicht den einfachen Zugriff auf Fahrzeugdaten (CAN), welche auf der IRS nicht zur Verfügung stehen. Aus diesem Grund, werden die Positionsinformationen über eine eigene Komponente (IRS_IRS_GPSProxy) auf der AU bereitgestellt. Die Abfrage der Position ist jedoch der einzige Unterschied zwischen VAU und RAU. Die Methodik, Daten von der IRS zur VsZ und umgekehrt zu übertragen, wird in Abschnitt 3.3 genauer beschrieben.

Die beiden Kommunikationswege zur LSA und zur Zentrale (LWL, UMTS) werden nur auf der IRS benötigt. Aus diesem Grund werden diese Verbindungen direkt an die RAU angeschlossen und laufen nicht über die CCU. Um diese Kommunikationswege nutzen zu können, wird je ein OSGi-Service angeboten, den die Funktionen nutzen können. Eine detaillierte Beschreibung der einzelnen Systemkomponenten auf der RAU wird im Abschnitt 3.3 gegeben.

2 Übergreifende Konzepte der Gesamtarchitektur

2.1 Kommunikation über ITS G5A

ITS G5A, d.h. funkbasierte Kommunikation basierend auf dem Standard IEEE 802.11p, stellt ein Kommunikationsmedium für sicherheits- und verkehrsrelevante Informationen dar. ITS Vehicle Stations und ITS Roadside Stations bilden ein dynamisches Ad-hoc-Netz, in dem die Bandbreite und die eingeschränkte Konnektivität der Partner eine entscheidende Rolle spielen. Der Informationsaustausch basiert im Wesentlichen auf verbindungsloser, paketorientierter Kommunikation.

Im C2X-Umfeld werden Nachrichten nicht unbedingt zwischen Softwarekomponenten der gleichen Funktion in den Systemen der Kommunikationspartner ausgetauscht. Es gibt eine klare Trennung zwischen der Funktionalität des Sendens und des Empfangens. Das Aussenden der Nachrichten wird von den Funktionen direkt veranlasst. Hierbei sind insbesondere die Funktionen der Hauptfunktion 1.1 „Datenerfassung“ zu nennen, die als Basisfunktionalität Daten zusammenstellen und anderen Kommunikationspartnern zur Verfügung stellen. Zumeist handelt es sich nicht um Nachrichten mit Informationen, die gezielt einem Kommunikationspartner zur Verfügung gestellt werden, sondern um Nachrichten, die nach festen Regeln per Geocast an alle Kommunikationspartner in einem definierten Adressbereich gesendet werden.

Nachrichten, die von einer ITS Station empfangen werden, werden teilweise von mehreren unterschiedlichen Funktionen der Application Unit genutzt. Das beste Beispiel hierzu sind die Cooperative Awareness Messages (CAM), das heißt Nachrichten, die Informationen über die Präsenz und den grundlegenden Status der umliegenden Kommunikationspartner beinhalten. Diese werden z.B. vom Kreuzungsassistent genutzt werden, um mögliche Konfliktpartner zu erkennen, von einer Stauerkennung, um die Verkehrslage in der Umgebung besser einzuschätzen, aber auch von der Infrastruktur, um Lichtsignalanlagen bedarfsgerecht zu steuern. Auf Empfängerseite werden die Nachrichten daher nicht direkt einer einzelnen Funktion zugeordnet. Sie werden zunächst in der Komponente „Umfeldtabelle“ abgelegt (siehe Abschnitt 3.2.4). Die Funktionen registrieren sich dort für bestimmte Datentypen oder sie fragen sie ab. Für einen spezifischeren Zugriff auf die Nachrichten in der Umfeldtabelle steht die Komponente „Relevanzfilter“ zur Verfügung (siehe Abschnitt 3.2.4). Zur Datenhaltung haben alle Nachrichten einige gemeinsame Datenelemente, z.B. die Gültigkeit der Nachricht und eine Action ID. Diese Attribute müssen von der aussendenden Funktion gesetzt werden. Zusätzlich werden in den Nachrichtenobjekten der Umfeldtabelle auch Informationen aus dem Netzwerkheader und zum Security-Status abgelegt.

Die Informationen sind entsprechend dieser Gedanken logisch in Nachrichten gegliedert, die als modulare Kommunikationsblöcke möglichst mehrfach verwendet und getrennt verwaltet werden können. Eine detaillierte Definition der Nachrichten ist in Deliverable D21.4 verfügbar. Hier soll nur ein kurzer Überblick und architekturelle Information gegeben werden. Die Definition der Daten orientiert sich an den Spezifikationen des C2C Communication Consortiums und der ETSI. Analog zu der Spezifikation in diesen Gremien und in den USA (Standard Draft SAE J2735) werden die Nachrichtenformate in ASN.1 definiert. Diese standardisierten Formate können automatisch und effizient nach den Packed Encoding Rules in Bytestream für die Übertragung umgesetzt werden. Neben der Darstellung der Nachrichtenformate wird die ASN.1-Spezifikation direkt zur Code-Generierung genutzt, um daraus Datenobjekte unter Java zu erstellen, die dem Zugriff auf die Nachrichteninhalte in den Funktionen dienen. Diese Nachrichtenobjekte greifen dann für

die Kodierung und Dekodierung auf eine Java-Bibliothek zu. Die Objektklassen und die Bibliothek werden in der Komponente C2XMessages zur Verfügung gestellt. Durch die Nutzung kommerzieller Tools wird die Fehlerwahrscheinlichkeit beim Kodieren in sim^{TD} definierter Nachrichten minimiert. Weitere Informationen zur Handhabung der Nachrichten über ITS G5A finden sich in Abschnitt 3.2.4.

Sollten entsprechende Nachrichten über Broadcast-Mechanismen anderer Medien (z.B. Consumer-WLAN) in eine ITS Station gelangen, so können diese entsprechend in die Umfeldtabelle eingestellt werden und von den Funktionen genutzt werden.

2.1.1 C2X Software stack

Die Kommunikation über ITS G5A wird softwaretechnisch durch den C2X Software stack gelöst. Der C2X Software stack ermöglicht die Nutzung der G5A Hardware für die simTD Funktionen.

Der C2X Software stack besteht aus:

- G5A-Hardware Interface,
- GEO-ADHOC-NET,
- Cirquent Anteil (unten „C2X Software stack-Cirquent“ genannt)

Diese Komponenten kommunizieren über die entsprechende Schnittstellen (siehe Abbildung 2-1).

Dieses Kapitel erklärt nur die C2X-relevanten Teile der Kommunikation. Die für die IP-basierte Kommunikation relevanten Komponenten (z.B. für Verbindungen mit einem Backend-System oder Web Service) werden hier nicht betrachtet.

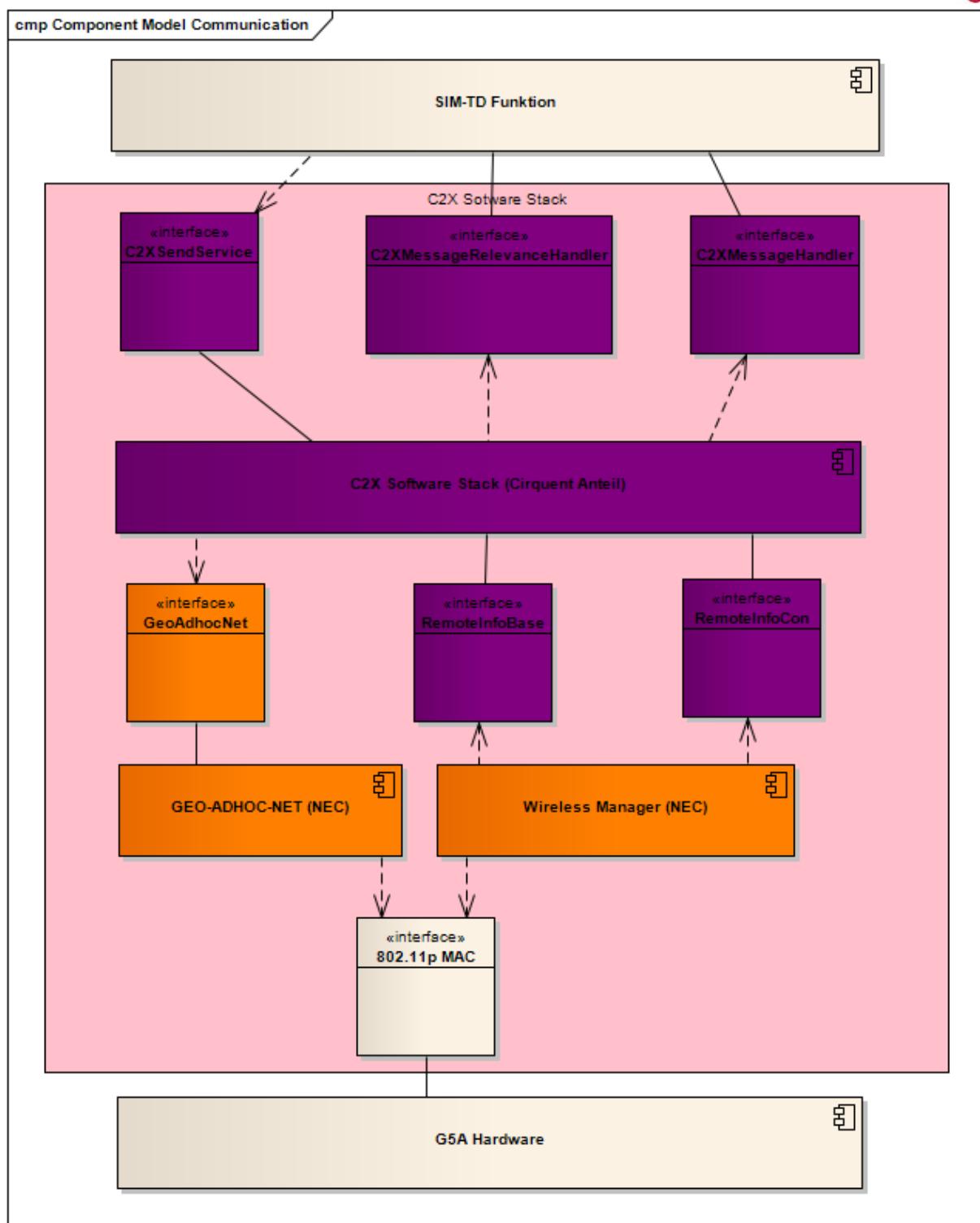


Abbildung 2-1: Kommunikationskomponenten des Software stacks und deren Schnittstellen

2.1.1.1 C2X Software-Stack (Cirquent Anteil)

Der C2X Software-Stack(Cirquent) realisiert das Versenden und Empfangen von C2X-Nachrichten für die Hauptfunktionen, den Informationsaustausch zwischen Host und Router

sowie das Generieren und Versenden von CAM-Nachrichten. Er realisiert zusätzlich die Komponenten Umfeldtabelle und Relevanzfilter auf dem Host, erfüllt Aufgaben wie Konfiguration, Austausch von Steuerungsinformationen, Kapseln der Schnittstelle SIM-NET des GeoRouting-Protokolls und die Verwaltung von Kommunikationssessions für mehrere Clients (Hauptfunktionen).

Der C2X Software-Stack(Cirquent) stellt der Komponente GEO-ADHOC-NET die Schnittstellen RemoteInfoBase und RemoteInfoConn zur Verfügung und greift auf die Schnittstelle SIM-NET API zu.

2.1.1.2 Schnittstelle RemoteInfoBase

Ermöglicht Zugriff auf die Information Base. Die Information Base ist ein hierarchisch organisierter Datenspeicher für Konfigurationsdaten und Betriebsdaten im C2X Software-Stack. Die Information Base speichert die Konfigurationsdaten für Initialisierung und Betrieb des gesamten CCU-seitigen Kommunikationsstacks und für die einzelnen Komponenten.

2.1.1.3 Schnittstelle RemoteInfoConn

Dient dem Austausch von Informationen (Events) zwischen dem C2X Software-Stack und davon getrennten Komponenten, insbesondere SIM-NET. Diese Schnittstelle repräsentiert die Komponente Information Connector im C2X Software-Stack. Der Information Connector dient zum „vertikalen“ Austausch von Informationen über unterschiedliche Kommunikationsschichten. Für eine effiziente Verteilung ist ein Notification-Mechanismus für Publish/Subscriber/Event vorgesehen.

2.1.1.4 Schnittstelle GEO-ADHOC-NET

Die Schnittstelle SIM-NET stellt Methoden zum Initialisieren der Kommunikation mit SIM-NET C2X und zum Versenden von Nachrichten mit entsprechendem Empfänger/Zielgebiet, Protocol ID, Transport- und Security-Optionen und dem Payload als Parameter zur Verfügung. SIM-NET stellt die Möglichkeit bereit, eine CallBack-Funktion zu registrieren, die im Fall einer empfangenen Nachricht aufgerufen wird.

Der C2X Software-Stack(Cirquent Anteil) greift auf diese Schnittstelle zu, um Nachrichten zu versenden und zu empfangen.

2.1.2 GEO-ADHOC-NET

GEO-ADHOC-NET realisiert die Netzwerkschicht und die Transportschicht für Car2X-Kommunikation. Es kommuniziert mit der Hardware auf MAC-Ebene. Unter anderem realisiert die Komponente auch das Forwarding, die Priorisierung von Paketen, Congestion/Communication Control und das Puffern von Paketen.

Die GEO-ADHOC-NET Komponente greift auf die Schnittstellen RemoteInfoBase und RemoteInfoConn zu und stellt die Schnittstelle GeoAdhocNet zur Verfügung. Die Kommunikation zwischen den beiden Komponenten (Netwerk-Client – Software-Stack-Cirquent) ist event basierend.

2.1.2.1 Schnittstelle 802.11p MAC

Versenden und Empfangen von Paketen auf dem MAC-Layer geschieht über die Schnittstelle 802.11p MAC (siehe Deliverable D21.3). Diese ist ebenfalls zuständig für die Konfiguration des Layers (z.B. der MAC-Adresse).

2.1.3 Priorisierung von Nachrichten und Congestion control

Für jede Hauptfunktion wird eine konfigurierbare Priorität gesetzt. Die Applikationen innerhalb der Hauptfunktionen haben die Möglichkeit eigene Priorisierungen ihrer Nachrichten (für jede Nachricht) zu definieren. Aus der von der Applikation gesetzten Priorität der einzelnen Nachricht und aus dem Prioritätswert der Hauptfunktion wird die globale Priorität der einzelnen Nachrichten im C2X Software stack berechnet. Beispiel: wenn 0 die niedrigste Priorität ist, und 100 die Höchste, und wenn eine Applikation den Wert 50 vorgibt, und die Hauptfunktion den globalen Wert zwischen 70 und 80 hat, ergibt der lokale Wert 50 einen globalen Wert 75. Für mehr Details siehe Deliverable Cirquent (link)

TP2 - Systementwurf/05 - AP21 - Gesamtarchitektur/A212 Entwicklung Kommunikationskonzept/V 2124
Entwickeln der Netzwerkprotokolle und des Nachrichtensystems

Ausser dem Prioritätswert setzen die Applikationen den TrafficClass für jede Nachricht. Der Traffic class bestimmt die Strategie für die Verbreitung einer Nachricht. Für alle Details bezüglich des Broadcastings, und die entsprechenden Congestion control Mechanismen, siehe Deliverable NEC, Kapitel Functional descriptions

[TP2 - Systementwurf/02 - Deliverables - Arbeitsdokumente/D21.x - Spezifizieren, Implementieren und Test des Nachrichtensystems A \(NEC\)](#)

2.1.4 Versenden von Nachrichten

Das Versenden von Nachrichten findet innerhalb der vorgesehenen Grundstruktur bei der Ausgabe statt:

- Erfassung (z.B. CAN) → Datenaufbereitung → Situationserkennung → Aktionsberechnung → Ausgabe (Versenden)

Beispiel: Anhand der Bordelektronik erkennt das Fahrzeug eine lokale Gefahrensituation. Daraufhin wird eine DEN-Nachricht generiert und an die anderen Verkehrsteilnehmer gesendet. Das Versenden einer Nachricht ist in Abbildung 2-2 dargestellt.

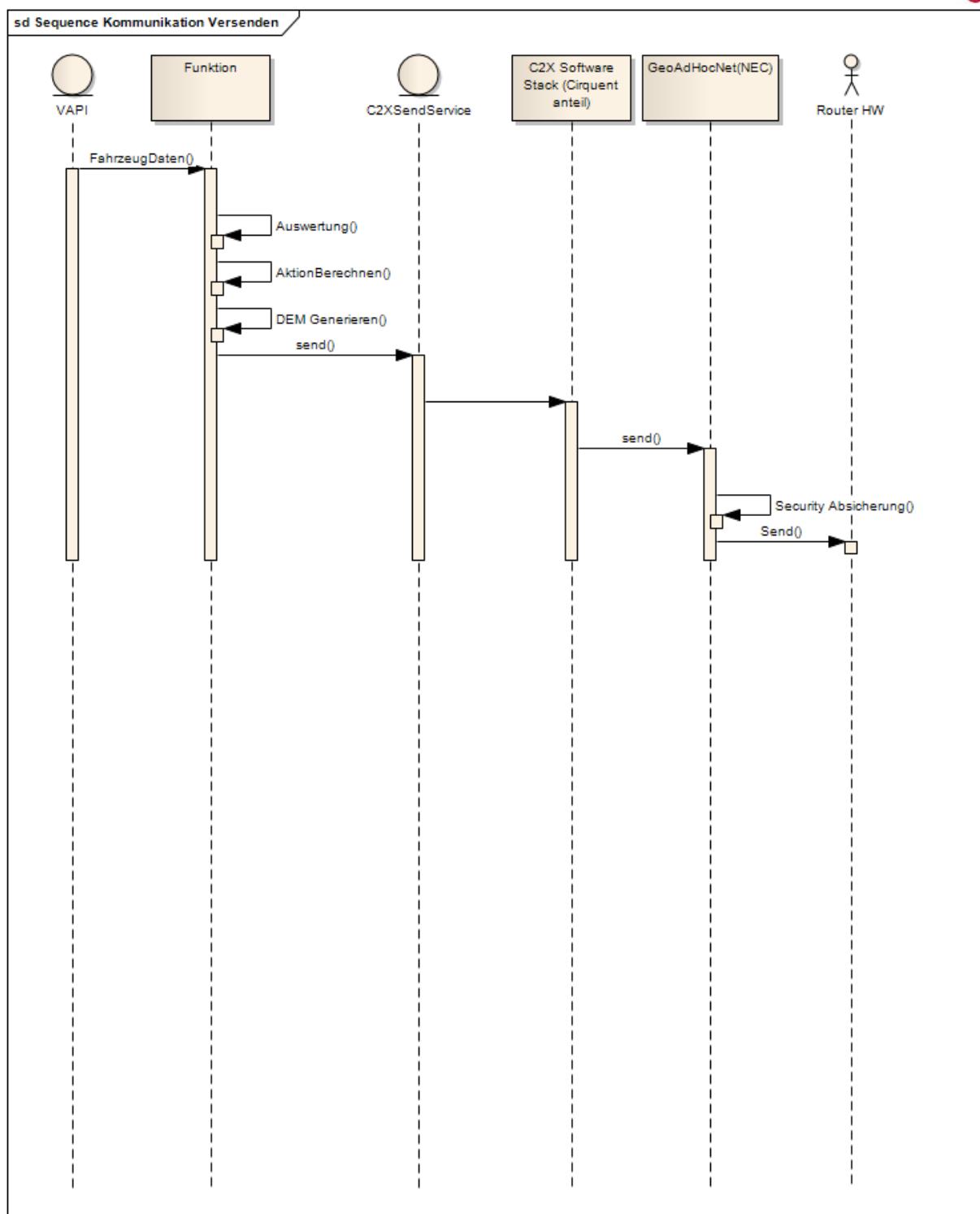


Abbildung 2-2: Versenden von Nachrichten aus einer Funktion

2.1.5 Empfangen von Nachrichten

Der Ablauf beim Empfang einer Nachricht ist wie folgt definiert:

- Erfassung (Empfangen) → Datenaufbereitung → Situationserkennung → Aktionsberechnung → Ausgabe (z.B. HMI)

Beispiel: Das Fahrzeug empfängt die Daten, die eine lokale Gefahr beinhalten, und nach einer Auswertung werden diese dem Fahrer angezeigt. Der prinzipielle Ablauf des Empfangs einer Nachricht ist in Abbildung 2-3 gezeigt.

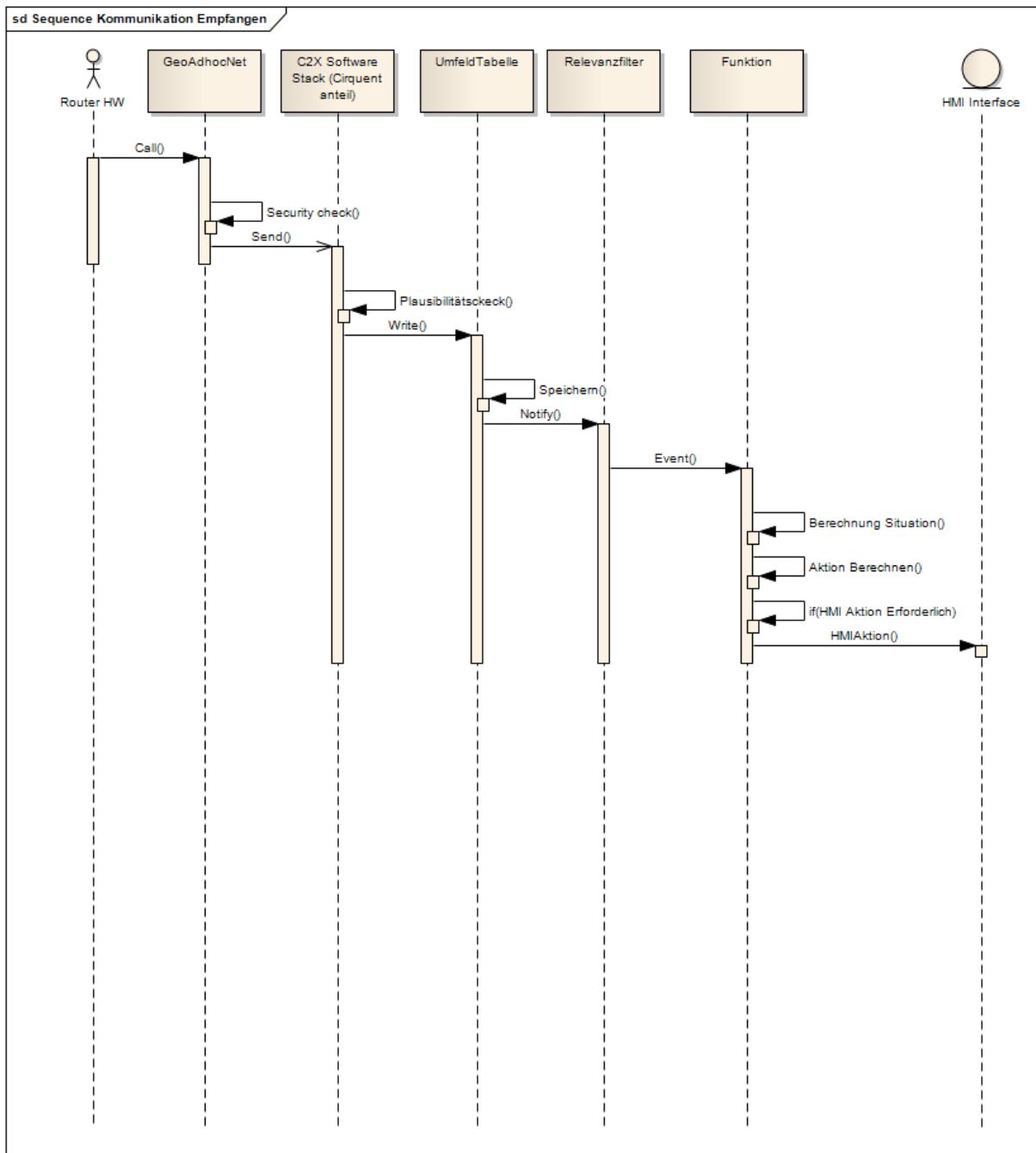


Abbildung 2-3: Empfangen von Nachrichten

2.2 Kommunikation über Consumer-WLAN und Mobilfunk

Als Grundlage für das Architekturkonzept für IP-basierte Datenübertragung dienen die in simTD für eine Untersuchung mit Mobilfunk bzw. Consumer-WLAN als Übertragungsstandard vorgesehenen Funktionen. Diese sind aus Tabelle 1 zu entnehmen.

Funktionen	ITS-G5	Consumer-WLAN	Zellular (GPRS, UMTS)
F_1.1.1 Infrastrukturseitige Datenerfassung			
F_1.1.2 Fahrzeugseitige Datenerfassung	X		X
F_1.1.3 Ermittlung der Verkehrswetterlage	X		X
F_1.1.4 Ermittlung der Verkehrslage	X		X
F_1.1.5 Identifikation von Verkehrereignissen	X		X
F_1.2.1 Straßenvorausschau	X	X	
F_1.2.2 Baustelleninformationssystem	X	X	X
F_1.2.3 Erweiterte Navigation	X	X	X
F_1.3.1 Umleitungsmanagement	X		X
F_1.3.2 Lichtsignalanlagen Netzsteuerung	X		
F_1.3.3 Lokale verkehrsabhängige LSA-Steuerung	X		
F_2.1.1 Hinderniswarnung	X		X
F_2.1.2 Stauendewarnung	X		
F_2.1.3 Straßenwetterwarnung	X		X
F_2.1.4 Einsatzfahrzeugwarnung	X		
F_2.2.1 Verkehrszeichen-Assistent/Warnung	X		X
F_2.2.2 Ampel-Phasen-Assistent/Warnung	X		
F_2.2.3 Längsführungsassistent	X		
F_2.2.4 Kreuzungs-/Querverkehrsassistent	X		
F_3.1.1 Internetbasierte Dienstnutzung		X	X
F_3.1.2 Standortinformationsdienste		X	X
F_4.1.1 Traceplayer		X	X
F_4.2.1 Livedaten			X
F_4.2.2 Messdaten		X	X
F_4.3.1 Flottenmanagement			X
F_4.3.2 Testablaufplanung			X
F_4.3.3 Versuchssteuerung			X
F_4.3.4 Fahrerbefragung			X
F_5.1.1 Verteildienst für individuelle Sicherheitsparameter			X
F_5.1.2 Verteildienst für allgemeine Sicherheitsparameter			X

Tabelle 1: Zuordnung von Kommunikationsarten zu Funktionen

2.2.1 Konzept für Mobilfunk

Zur Datenübertragung über Mobilfunk stehen einerseits GSM-basierte Dienste (GPRS und EDGE) und andererseits Dienste, die auf UMTS als Basistechnologie aufsetzen, zur Verfügung (HSDPA und HSUPA, gemeinsam als HSPA bezeichnet). Aus Sicht einer IP-basierten logischen Ende-zu-Ende-Verbindung beinhalten diese beiden Zugangstechnologien keine wesentlichen Unterschiede (auf Netzwerkprotokoll-Ebene wird jeweils eine Verbindung zu einem Access Point über die Festlegung eines Access Point Namens (APN) im Endgerät aufgebaut). Daher soll im Fall einer Verfügbarkeit von UMTS/HSPA eine Verbindung hierüber aufgebaut werden und im Fall von ausschließlicher Verfügbarkeit von GSM/EDGE auf diese Zugangstechnologie zurückgefallen werden. Die Auswahl der jeweiligen Zugangsart soll in der Vehicle CCU erfolgen und für die Applikationsschicht transparent sein. Detaillierte Ausführungen des Mobilfunkkonzepts finden sich in Deliverable D21.3.

Für die Betrachtung der Realisierung von C2X-Funktionen über Mobilfunk müssen auch die zur Verfügung stehenden Leistungsmerkmale des von T-Mobile als Provider angebotenen Systems und die Eigenschaften des vorgesehenen CCU-Moduls berücksichtigt werden. Insbesondere sind dies:

- Verfügbarkeit von EDGE als Datenübertragungsstandard flächendeckend im Versuchsgebiet.
- Verfügbarkeit von UMTS HSPA (HSDPA und HSUPA) als Datenübertragungsstandard in weiten Teilen des Versuchsgebietes.
- Keine Verfügbarkeit von öffentlichen festen IP-Adressen zur Adressierung der Mobilgeräte im Fahrzeug von Seiten des Mobilfunkbetreibers T-Mobile.
- Verfügbarkeit von privaten festen IP-Adressen zur Adressierung der Mobilgeräte im Fahrzeug von Seiten des Mobilfunkbetreibers T-Mobile.
- IPv4 als Netzwerkprotokoll innerhalb des Mobilfunksystems und an den Schnittstellen nach außen.
- Keine Verfügbarkeit von Geocast-, Broadcast- oder Multicast-Diensten als Basisdienst im Mobilfunknetz (weder MBMS noch Cell Broadcast).
- Verwendung eines Mobilfunkmoduls in der CCU mit GPRS Klasse 10 und HSDPA Kategorie 7.

2.2.1.1 Übersicht

Im Folgenden ist die Übersicht über die Architektur des in Abbildung 2-4 als „ITS IMT Public“ bezeichneten Funktionsblocks zu sehen.

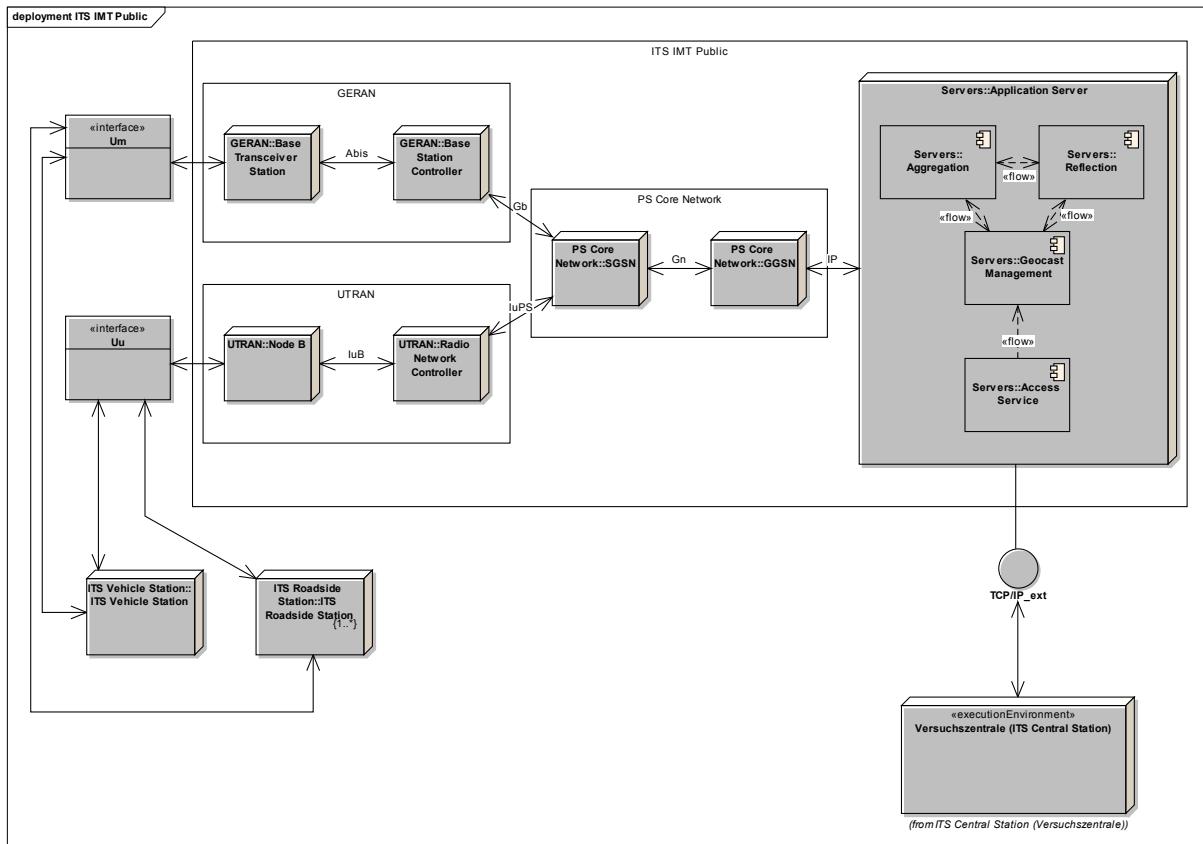


Abbildung 2-4 Übersicht ITS IMT Public

Die Abbildung zeigt zum einen das eigentliche Mobilfunksystem mit den Radio-Access-Teilen UTRAN und GERAN und dem Core-Netz. Um die für sim^{TD} vorgesehenen Funktionen über Mobilfunk realisieren zu können, ist es andererseits notwendig, zwischen dem GGSN (Gateway GPRS Support Node), der als Zugangsknoten zwischen dem Mobilfunknetz und dem Internet fungiert, und der Verkehrsinfrastrukturseite noch eine Serverkomponente vorzusehen. In diesem Server sind folgende Aufgaben angesiedelt:

1. Access Service: Der Access Server kümmert sich um die Adressierung der Fahrzeuge über IP. Abhängig von den Möglichkeiten des Mobilfunksystems (statisch/dynamisch IP Adressierung, öffentliche/interne IP-Adressierung, DynDNS, etc.) wertet er dazu die über IP gesendeten PositionUpdate-Nachrichten der Fahrzeuge aus und trägt die IP-Adresse für alle aktiven Node IDs in eine spezielle IMTUMfeldtabelle ein.
2. Aggregation: Da es nicht sinnvoll ist, wiederholte Nachrichten auch mehrfach weiterzuleiten, erfolgt eine Aggregierung und/oder Filterung der Nachrichten, bevor diese weitergegeben werden. Dabei werden redundante oder ungültige Nachrichten entfernt.
3. Reflektion: Um im Fall einer Warnmeldung, wie z.B. einer Hinderniswarnung von einem Fahrzeug, möglichst schnell eine Verbreitung der Information im geografisch relevanten Gebiet zu ermöglichen, werden solche Nachrichten nach einer Überprüfung durch die Aggregationskomponente an alle Fahrzeuge, die sich

im Relevanzgebiet der Nachricht befinden, geschickt (unter Nutzung des Geocast Management Services). Diese Aufgabe übernimmt die Komponente Reflektor.

4. Geocast Management. Typischerweise sind ein Großteil der Nachrichten in ITS-Systemen Geocast-Nachrichten. Es werden also Nachrichten an alle Teilnehmer innerhalb eines geografisch begrenzten Gebietes geschickt. Der Geocast Manager muss diese Funktionalität im Falle der Nutzung von Mobilfunk für die Übertragung solcher Nachrichten bereitstellen. Da im vorliegenden Fall, wie eingangs erwähnt, keine Broadcast- oder Multicast-Fähigkeit des Systems gegeben ist, muss hierzu eine IMTUMfeldtabelle genannte Zuordnungstabelle aufgebaut und verwaltet werden, welche die geografischen Positionen (aus vom Fahrzeug regelmäßig gesendeten Statusnachrichten ähnlich der CAM Nachrichten bei ITS G5A) und IP-Adressen der Fahrzeuge in Beziehung setzt. Hierdurch können Geocast-Nachrichten in eine entsprechende Anzahl Unicast-Nachrichten umgesetzt werden. Dabei ist insbesondere auch auf die Einhaltung der IT-Sicherheitsvorgaben (z.B. Anonymität) zu achten.

Diese vier Aufgaben werden durch einen Communication Manager sowie einen Dispatcher unterstützt. Der Communication Manager ist für die Versendung von Nachrichten über die externen IP-Schnittstellen zuständig. Zusätzlich zu Nachrichten, die direkt über die externen Schnittstellen eintreffen, kann er auch Nachrichten vom Geocast Manager erhalten. Der Dispatcher verteilt die verschiedenen Nachrichtentypen (DEN, Probe Vehicle Data, etc.) an die jeweils zuständigen Komponenten.

2.2.1.2 Schnittstellen

Wie in Abbildung 2-4 zu sehen, bietet das ITS IMT Public Teilsystem drei Schnittstellen nach außen an. Zum einen sind dies die Funkschnittstellen der GERAN und UTRAN Funkzugangsnetze. Auf der anderen Seite steht eine Schnittstelle ins Internet zur Anbindung der Versuchszentrale oder zur Bereitstellung zusätzlicher Internet basierter Dienste zur Verfügung.

Die Funkschnittstellen sind in D21.3 genauer beschrieben. Deshalb wird hier nicht weiter auf sie eingegangen. Die Schnittstelle zum Internet setzt auf TCP/IP als Netz- und Transportprotokoll auf und muss hier insofern ebenfalls nicht genauer beschrieben werden.

2.2.1.3 Protokolle

Der Einsatz von zellularem Mobilfunk zur Datenübertragung hat in sim^{TD} verschiedene Anwendungsfälle. Bei all diesen Anwendungen dienen als Basis auf der Netzwerkschicht das IP-Protokoll und auf der Transportschicht die Protokolle UDP oder TCP. Bei Protokollen oberhalb der Transportschicht sollen soweit als möglich die gleichen Protokolle und Datenformate Verwendung finden wie bei der Funkübertragung mit ITS-G5. Abbildung 2-5 gibt hierzu einen Überblick.

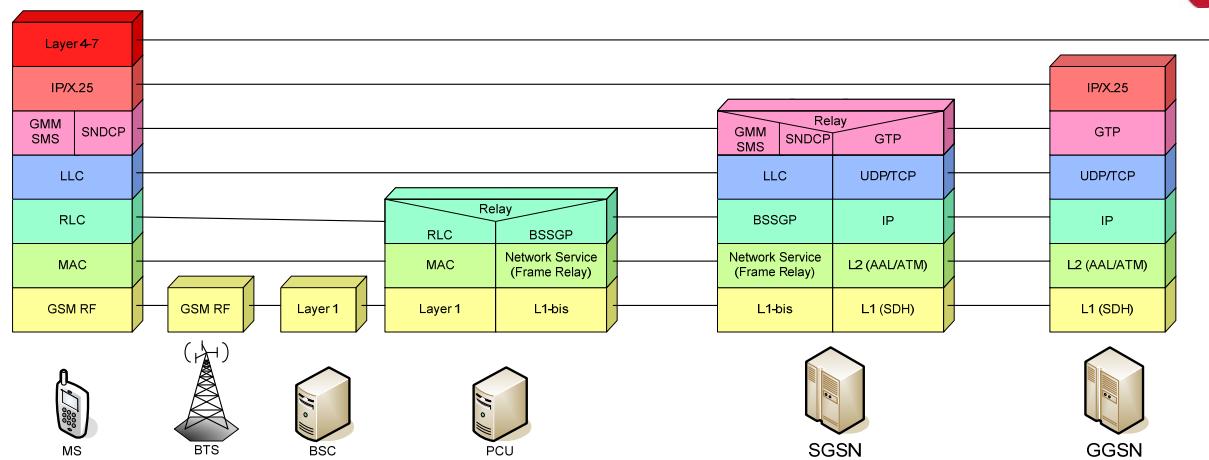


Abbildung 2-5: Protokollstack ITS IMT Public

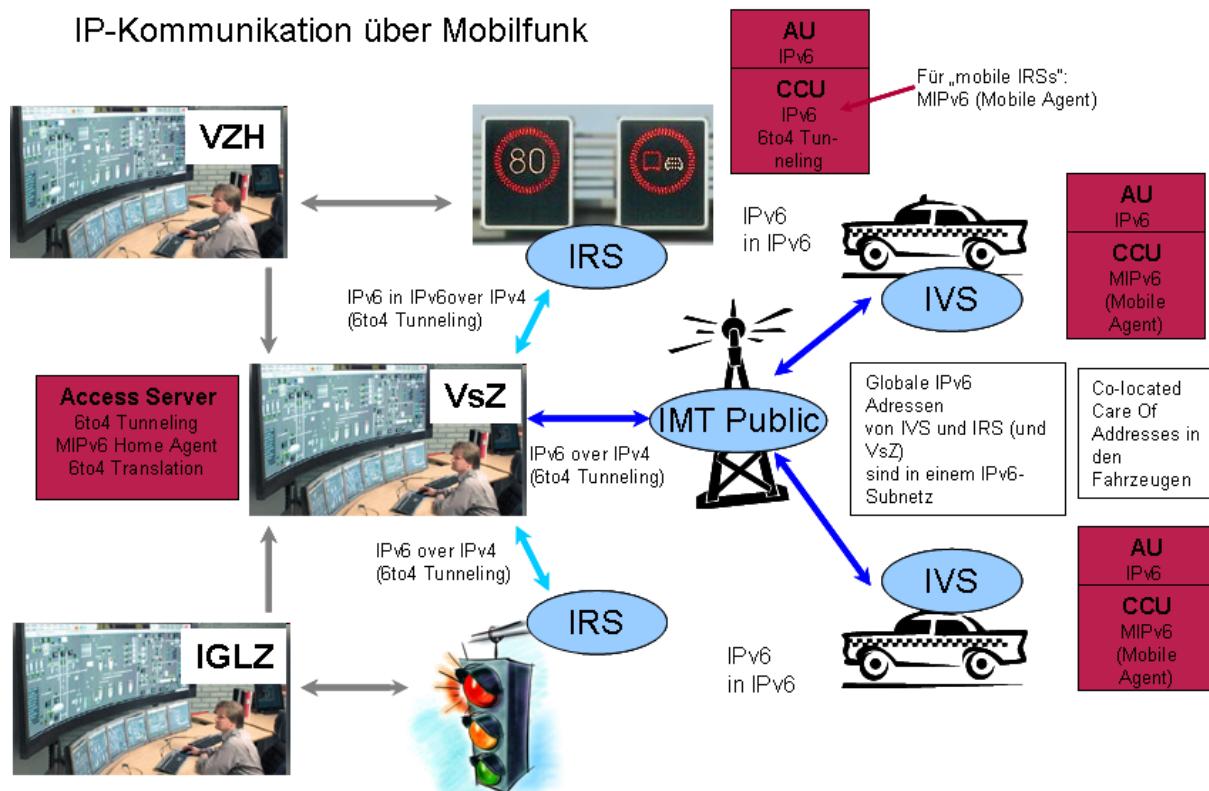


Abbildung 2-6: Übersicht Kommunikation mittels ITS IMT Public

Für die Adressierung der Fahrzeuge wird Mobile IPv6 verwendet. Damit ist gewährleistet, dass Fahrzeuge unabhängig vom Übertragungsmedium immer über die gleiche globale IPv6-Adresse erreichbar sind. Darüber hinaus können die Fahrzeuge auch beispielsweise

von der Versuchszentrale aus erreicht werden, ohne dass sie hierzu eine TCP-Verbindung aufbauen müssen, wenn zwischen Fahrzeug und Versuchszentrale ein IP-Tunnel etabliert ist¹. Die Integration von Mobile IPv6 in das Mobilfunkkonzept ist in Abbildung 2-6 dargestellt. Da die Fahrzeuge über Mobilfunk keine festen öffentlichen IP(v4)-Adressen innerhalb des T-Mobile Mobilfunknetzes zugewiesen bekommen können, muss die Erreichbarkeit der Fahrzeuge von Infrastrukturseite durch den Aufbau von Tunneln oder andere Mechanismen (z.B. dynamisches DNS) gewährleistet werden. Die Verwendung von IP-Tunneln sichert die Erreichbarkeit der Fahrzeuge aus der Versuchszentrale (unabhängig von der internen Vergabe von IP-Adressen beim Mobilfunk, feste oder dynamische IP-Adressen) und die Verwendung von IPv6. Eine Alternative dazu ist die Benutzung eines eigenen APN mit festen internen IP-Adressen für die Mobilstationen (im sim^{TD} die CCUs). Da für die Anbindung von Mobilstationen nur IPv4 verfügbar ist, wird dennoch ein 6to4 Tunnel bis zum Access Server für die Anwendung von IPv6 benötigt. Folgendes Bild stellt den vereinfachten Überblick zum IPv6 Tunneling dar. Als Ansatz für die Konzeption der IPv6-Adressvergabe wird die Autokonfigurationsmöglichkeit für IP-Adressen bei IPv6 verwendet (Erzeugen von link-lokalen Adressen, fe80::EUI/64/64).

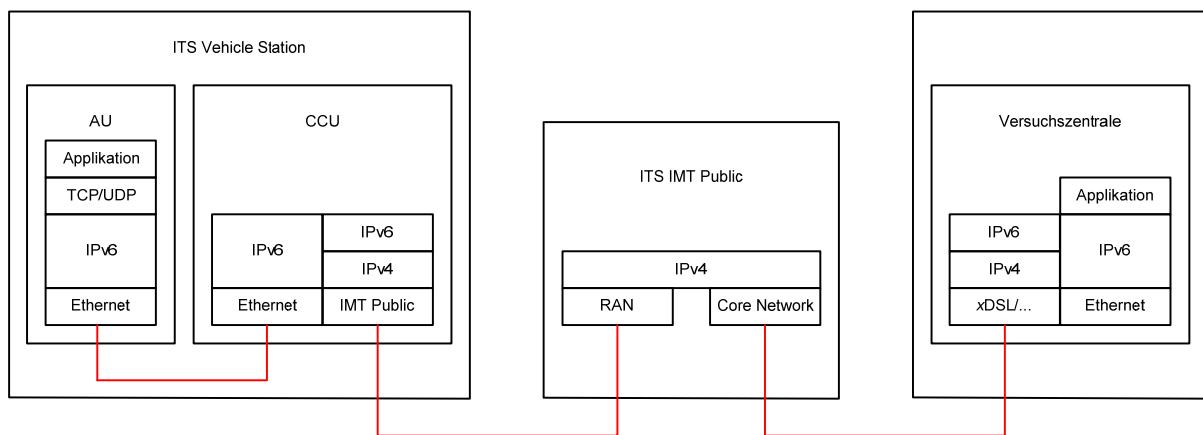


Abbildung 2-7: Übersicht IPv6 Tunneling bei ITS IMT Public

Der C2X-Header wird ersetzt durch einen C2XoverIMT-Header (angepasster C2X Header für C2X Nachrichten über Mobilfunk) mit folgender Struktur:

- Version
- Next Header
- Header Type
- Header Sub Type
- Flags
- Payload Length
- Sender Position Vector (Node ID, Timestamp, Latitude, Longitude, Speed, Heading, Altitude, Accuracy values)
- Sequence Number
- Repetition Interval
- Lifetime
- Addressing part (e.g. geographical region)

¹ Das setzt natürlich voraus, dass es eine Mobilfunk- oder Consumer-WLAN-Verbindung zwischen Fahrzeug und Versuchszentrale gibt.

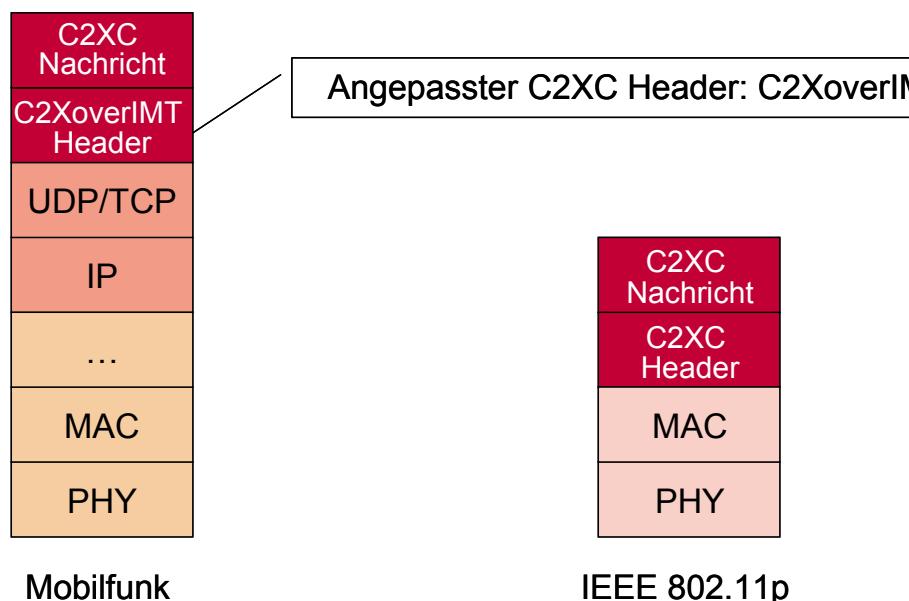


Abbildung 2-8: Übersicht Anpassung C2X Header an ITS IMT Public

Damit wird das Mobilfunksystem für folgende Szenarien eingesetzt.

1. Nutzung von Mobilfunk als Funkübertragungsmedium für funktionsbezogene Daten zwischen Fahrzeug und Verkehrszentrale sowie zwischen Verkehrszentrale und Fahrzeug.

Auf Applikationsebene wird angestrebt, die gleichen Protokolle zu verwenden, wie bei der Übertragung per ITS G5A. Eine Übertragung der CAM-Nachrichten ist nicht vorgesehen. Stattdessen ist es notwendig, um die oben genannten Server-Funktionalitäten realisieren zu können, eine spezielle Statusnachricht für ITS IMT Public von den Fahrzeugen zu erhalten. Diese Nachricht kann ähnliche Informationen wie die CAMs beinhalten, wird aber seltener gesendet und verzichtet auf ITS-G5 spezifische Inhalte. Alle periodischen, nicht sicherheitskritischen Nachrichten werden über UDP als Transportprotokoll übertragen. Alle anderen Nachrichten nutzen TCP auf Transportebene.

Außerdem werden die Daten der HF3.1 (ergänzende Dienste) über Mobilfunk verfügbar gemacht. Verkehrslage und Verkehrsereignisinformationen, Bilder von Verkehrswebcams, Kommunalinformationen und Parkrauminformationen können fahrzeug- und nutzerspezifisch aus der Versuchszentrale abgerufen und in das Fahrzeug übertragen werden. Für diesen Zweck wird eine Kommunikation über TCP/IP – in Einzelfällen auch UDP/IP – eingesetzt.

Ein Sonderfall stellt die Übertragung von Daten für die Steuerung und Auswertung der sim^{TD}-Versuchsergebnisse dar. Dazu stellt das ITS IMT Public Teilsystem die notwendigen Übertragungswege auf Basis von IP und TCP auf der Netz- bzw. Transportschicht bereit.

2. Nutzung von Mobilfunk als Funkübertragungsmedium für funktionsbezogene Daten zwischen Fahrzeugen.

Es besteht die Absicht über Services im Applikationsserver (siehe Abbildung 2-4) Ereignismeldungen direkt an die geografisch relevanten Verkehrsteilnehmer zurückzuspiegeln. Hierbei wird auf Applikationsebene keine Veränderung der erhaltenen Ereignismeldung durchgeführt, sondern nur eine Umsetzung des Nachrichteninhaltes auf Unicast-Nachrichten an alle als relevant bestimmten Fahrzeuge realisiert. Dabei werden

sowohl die C2X Nachrichteninhalte, als auch der C2XoverIMT Header unverändert beibehalten, und lediglich die darunter liegenden Schichten (Layer 1-4) verändert.

3. Nutzung von Mobilfunk zur Anbindung von ITS Roadside Stations an die Verkehrszentrale oder an andere Infrastrukturelemente als Alternative zu einer Anbindung über Kabel.

In diesem Fall dient das Mobilfunksystem als reines Datenübertragungsmedium, welches einen TCP/IP-basierten Übertragungsweg für die Nachrichten zur Verfügung stellt. Die Nachrichten auf höheren Protokollsichten unterscheiden sich insofern nicht von denen bei der Übertragung über kabelgebundenes Ethernet und sind im Kapitel für die Beschreibung der ITS Roadside Station genauer definiert.

2.2.1.4 Quality of Service (QoS)

Die Festlegung der Leistungsmerkmale einer Verbindung über den Paketvermittlungsteil von zellulären Mobilfunksystemen, wie sie sim^{TD} zur Verfügung stehen, geschieht durch Aufbau eines sogenannten PDP (Packet Data Protocol) Kontextes. Der PDP-Kontext stellt eine logische Verbindung von einem Mobilfunkteilnehmer zum Übergabepunkt ins Internet, dem GGSN, dar. Darüber hinaus erfolgt in diesem Kontext auch die Festlegung auf bestimmte QoS-Merkmale der Verbindung. Bei EDGE stehen dabei verschiedene Verzögerungsklassen, Verlässlichkeitssklassen, Dringlichkeitsklassen und Durchsatzklassen zur Verfügung. Bei UMTS existieren verschiedene QoS-Klassen.

Da die verschiedenen Datenarten, wie im vorigen Abschnitt beschrieben, unterschiedliche Anforderungen an die Übertragung bezüglich Kapazität, Verlässlichkeit und Latenz stellen, sollte unter Umständen für jeden (im Bezug auf QoS) unterscheidbaren Datentyp ein eigener PDP-Kontext aufgebaut werden, um bereits auf der Ebene des Übertragungssystems eine möglichst effiziente und auf die Anforderungen zugeschnittene Ressourcennutzung möglich zu machen.

2.2.2 Konzept für Consumer-WLAN

In sim^{TD} wird der Begriff „Consumer WLAN“ synonym für IEEE 802.11b/g verwendet. Dabei müssen zur Verfügung stehenden Leistungsmerkmale des im sim^{TD}-Projekt verfügbaren Funkinterfaces für 802.11b/g berücksichtigt werden; insbesondere, dass es nur im Ad-hoc-Modus operieren kann. Details hierzu finden sich in Deliverable D21.3. Die Gesamtübersicht über die Kommunikation mittels Consumer-WLAN zeigt Abbildung 2-9.

2.2.2.1 Übersicht

Bei der Kommunikation mittels Consumer-WLAN werden, je nach beteiligten Komponenten, drei Szenarien unterschieden:

1. WLAN 802.11b/g wird als Alternative zu 802.11p für bestimmte Funktionen (z.B. Gruppe F_1.2.x) getestet. Siehe Abbildung 2-10.
2. WLAN 802.11b/g als Übertragungsmedium für Funktionen, die eine Weiterleitung von lokalen Informationen (z.B. F_3.1.2) benötigen. Siehe Abbildung 2-11.
3. WLAN 802.11b/g als Medium für die Datenübertragung zwischen ITS Vehicle Station und Versuchszentrale über WLAN Access Points. Siehe Abbildung 2-12.

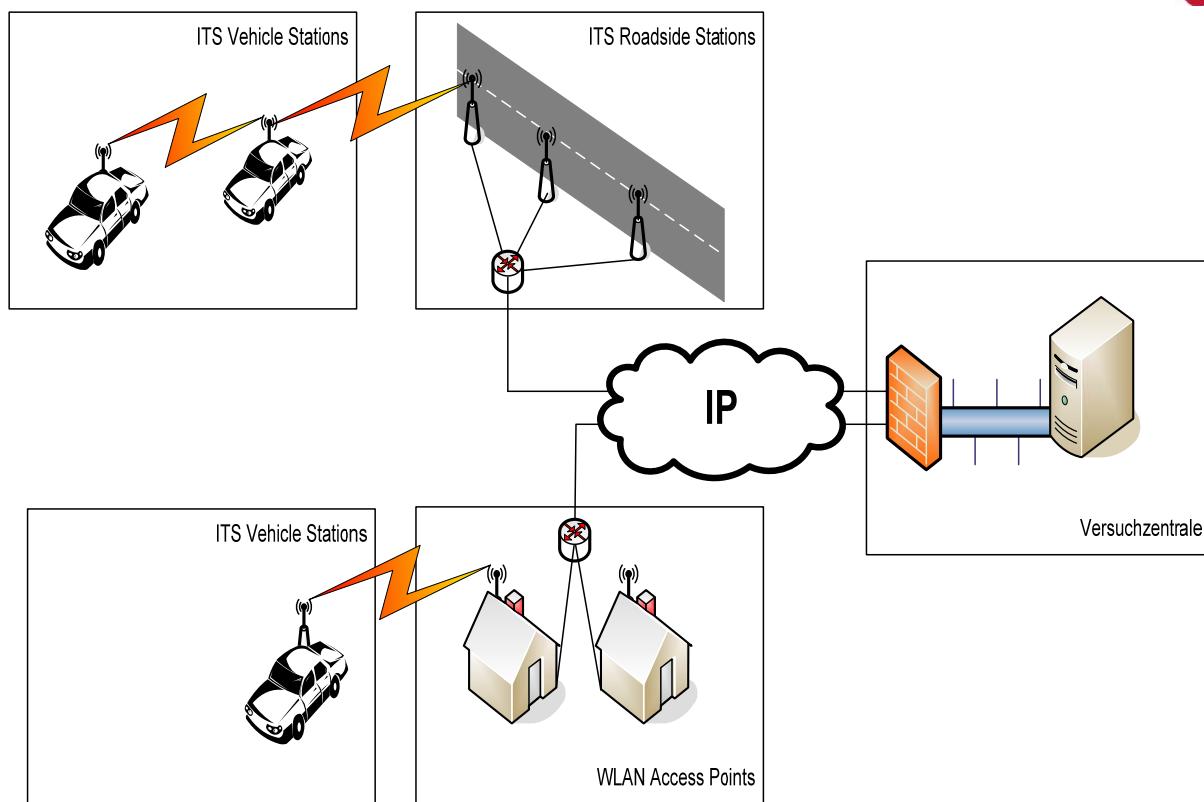


Abbildung 2-9: Übersicht Kommunikation mittels Consumer-WLAN

Konkrete Funktionsbeispiele für die oben aufgeführte Verwendung von WLAN wären bei 1. die Übertragung von Baustelleninformationen. Unter 2. können Standortinformationsdienste zur Anwendung kommen und zu 3. die Übertragung von im Fahrzeug gesammelten Protokolldaten zur Auswertung in die Versuchzentrale. Die hier beschriebenen WLAN Access Points gehören zur sim^{TD}-Infrastruktur. Eine Kommunikation mittels öffentlicher HotSpots (z.B. T-Mobile) ist nicht vorgesehen.

2.2.2.2 Schnittstellen

Wie in Abbildung 2-9 zu sehen, werden von den ITS Roadside Stations und von den WLAN Access Points 802.11b/g Schnittstellen in Ad-hoc-Modus zur Verfügung gestellt. Die Funkschnittstellen sind in D21.3 genauer beschrieben, deshalb wird hier nicht weiter auf sie eingegangen. Die Schnittstelle zum Internet setzt auf TCP/IP als Netz- und Transportprotokoll auf und muss hier insofern ebenfalls nicht genauer beschrieben werden.

2.2.2.3 Protokolle

Die Kommunikation auf 802.11b/g-Basis wird ausschließlich im Ad-hoc-Modus betrieben. Dabei kommunizieren die WLAN-Clients direkt miteinander und ohne einen WLAN Zugangspunkt (Infrastrukturmodus). Die ITS Roadside Station muss neben der Funktionalität des Ad-hoc-Clients auch ggf. als Gateway fungieren und die Daten des verbundenen Ad-hoc-Netzes in die Versuchszentrale routen.

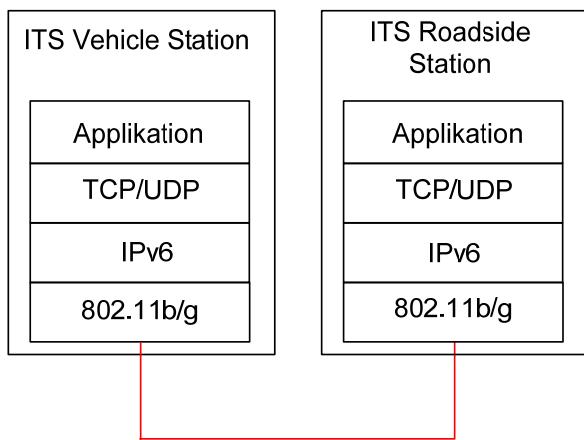


Abbildung 2-10: Ad-hoc zwischen ITS Vehicle Stations

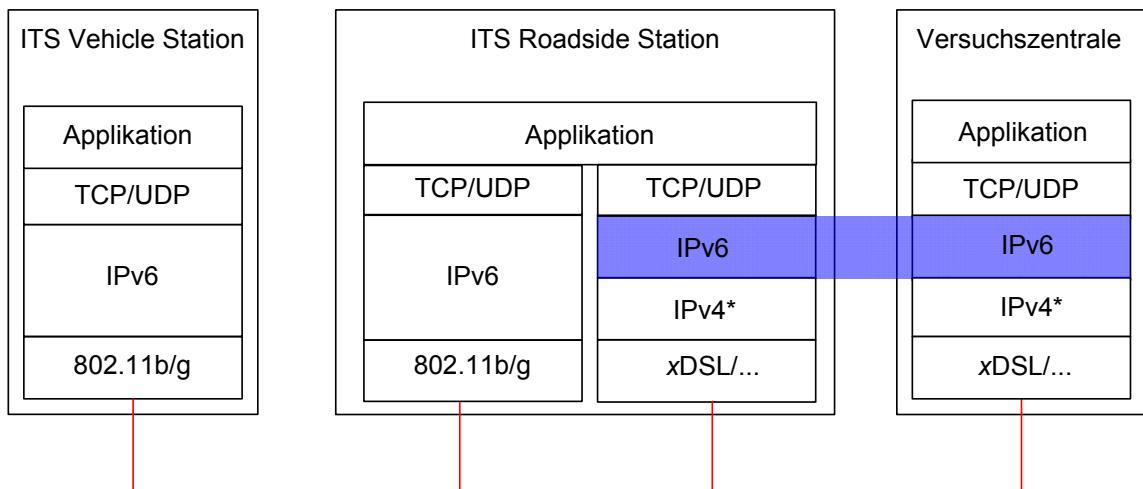


Abbildung 2-11: Ad-hoc zwischen ITS Vehicle Station und ITS Roadside Station

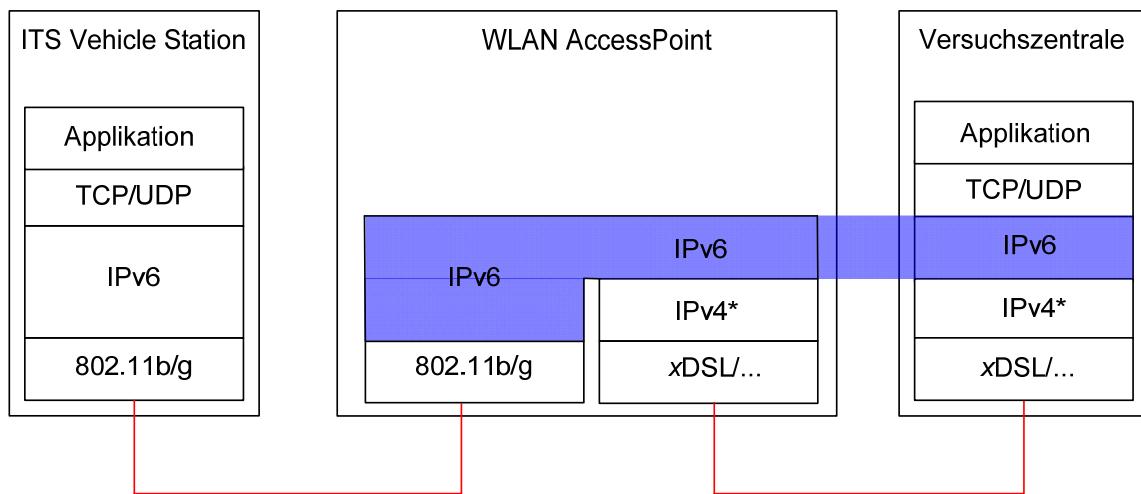


Abbildung 2-12: Ad-hoc zwischen ITS Vehicle Station und WLAN AccessPoint

Die WLAN Access Point fungieren als Gateway zwischen den ITS Vehicle Stations und der Versuchszentrale und obwohl der Begriff „Access Point“ dafür benutzt wird, werden diese auch in WLAN Ad-hoc-Modus operieren, und nicht in Infrastrukturmodus (Begrenzung durch das verwendete WLAN-Funkmodul). Die Kommunikation zwischen ITS Vehicle Station und WLAN Access Point ist auf Netzwerkebene identisch mit der zwischen ITS Vehicle Station und ITS Roadside Station. Der Unterschied liegt an den übertragenen Informationen. Für beide Fälle werden Broadcast Nachrichten gesendet, die alle Teilnehmer im Ad-hoc-Netzwerk erreichen sollen. Jede Entität (ITS Vehicle Station, ITS Roadside Station) entscheidet über die Handhabung der Nachrichten, z.B. ob sie dem Fahrer angezeigt oder an die Versuchszentrale und/oder andere Fahrzeuge weitergeleitet werden sollen, etc.

Der Einsatz von WLAN Access Points soll die Übertragung großer Datenmengen (z.B. Protokolldateien, Softwareupdates, etc.) zwischen den Fahrzeugen und der Versuchszentrale ermöglichen. Die Access Points werden voraussichtlich dort platziert, wo die Versuchsfahrzeuge für längere Zeit stehen bleiben (z.B. Parkhaus).

Bei der Nutzung von Consumer-WLAN findet ausschließlich IPv6 Verwendung. Die Verwendung von IPv6 wird durch den Ausschluss von öffentlichen WLAN Zugangspunkten vereinfacht, da WLAN aus Sicht von sim^{TD} rein „projektintern“ verwendet wird und somit alle zur Anwendung kommenden Schnittstellen (ITS Roadside Station, ITS Vehicle Station, etc.) auf IPv6 konfiguriert sind.

Als Ansatz für die Konzeption der IPv6-Adressvergabe wird die Autokonfigurationsmöglichkeit für IP-Adressen bei IPv6 verwendet (Erzeugen von link-lokalen Adressen, fe80::EU/64/64).

2.3 Umgang mit Positionsdaten

Für alle sim^{TD}-Funktionen ist die Verfügbarkeit möglichst genauer, zeitnäher und verlässlicher Positionsdaten von herausragender Bedeutung. Insbesondere die Hauptfunktionen zur aktiven Sicherheit und Fahrerassistenz sind bezüglich der erreichten Funktionsqualität von der Güte der Positionierung abhängig. Daher wurde ein Konzept entwickelt, das trotz limitierter Hardwarekosten ein Höchstmaß an Positionsgenauigkeit für die umfangreiche sim^{TD} -Basisflotte ermöglicht.

Als Ortungsverfahren kommt generell in allen Fahrzeugen das „Global Positioning System“ (GPS) mit Auswertung von Differenzsignalen (DGPS, dGPS) auf einem preiswerten GPS-Empfänger zum Einsatz. Das benötigte Korrektursignal soll allen Fahrzeugen im Versuchsgebiet so flächendeckend als möglich über UMTS oder ITS G5A zur Verfügung gestellt werden.

Die vom GPS-Receiver erhaltenen Positionsdaten können kurzzeitig ausfallen oder beeinträchtigt sein, beispielsweise durch Satelliten-Verdeckungen infolge höherer Gebäude am Straßenrandbereich, durch Mehrwegeausbreitung von Signalen („Multipath-Effekte“), durch Verdeckungen in Tunnels, Unterführungen und unter Brücken. Daher werden die DGPS-Positionsdaten in der sim^{TD}-Komponente „Bessere Ortung“ weiter aufgearbeitet. Die „Bessere Ortung“ fusioniert dabei die DGPS-Rohdaten mit den Fahrzeugdaten, die von der VAPI-Komponente bereitgestellt werden. Ein auf Kalman-Technik basierendes Positionsfilter erreicht durch Nutzung verfügbarer dynamischer Genauigkeitsinformationen eine optimale Integration der absoluten GPS-Daten mit verfügbaren Onboard-Daten zur relativen Fahrzeugbewegung. Kurzzeitige GPS-Ausfälle können damit durch Koppelnavigation vollständig überbrückt werden, und einzelne GPS-Messfehler beeinträchtigen nicht das stetige nahtlose Verfolgen der Fahrzeugbewegung. Die flächendeckende Verfügbarkeit des benötigten Korrektursignals oder teilweise bzw. kurzzeitige Ausfälle sind dabei für die Architektur unkritisch. In Abwesenheit eines Korrektursignals stellt das Modul „Bessere Ortung“ automatisch die Positionsdaten mit der Genauigkeit bereit, welche ohne oder mit einem „älteren“ Korrektursignal noch möglich ist.

Die Komponente „Bessere Ortung“ liefert im Ergebnis die aktuelle Position in Geokoordinaten als Länge und Breite zusammen mit dem Zeitstempel, zu dem diese Position gültig war, sowie mit einer Genauigkeitsschätzung in Form einer „Genauigkeitsellipse“ (beschreibt die maximale und die orthogonale minimale geschätzte Standardabweichung der Position sowie die Richtung der maximalen geschätzten Standardabweichung relativ zur Nordrichtung). Die Geokoordinaten können anschließend als die „gültige Position“ von allen anderen Komponenten und Funktionen verwendet werden, insbesondere auch für die Positionsstempelung von Nachrichten im sim^{TD} Car2X-Framework selbst. Über die VAPI werden sowohl die Ausgangsdaten der „Besseren Ortung“ als auch die DGPS-Positionsdaten zur Verfügung gestellt. Welche Positionsdaten eine Funktion nutzt, bleibt ihr überlassen. Weitere Informationen zur „Besseren Ortung“ sind in Abschnitt 3.1.2 zu finden.

2.4 Testsystem

2.4.1 Prüfstand

Als Ergänzung zur Testflotte wird ein Laborsystem aufgebaut, das alle Systemkomponenten beinhaltet und das eine detaillierte Systemanalyse und Systemoptimierung unter optimalen Bedingungen erlaubt. Der Prüfstand dient der Evaluation des integrierten Gesamtsystems, nicht dem Debugging einzelner Funktionen.

Um realitätsnahe Eingaben in das zu testende System zu haben, werden in einem ersten externen Schritt in echten Versuchen Daten aufgezeichnet (sog. „Traces“), die dann im eigentlichen Prüfstand mit Hilfe eines Wiedergabe-Tools („TracePlayer“) in das zu testende System eingespielt werden. Die Aufzeichnung der Traces erfolgt mit Hilfe der regulären Versuchsdatenverarbeitung. Ein spezielles Log-Profil definiert die für Traces relevanten aufzuzeichnenden Daten. Die Prüfstandskomponenten sind also an der Trace-Aufzeichnung

nicht beteiligt. Erste Traces vor Beginn der Versuchs- und Testphase werden voraussichtlich von Daimler gestellt.

In einem *Prüfstandsaufbau* werden eine oder mehrere ITS Vehicle Stations oder ITS Roadside Stations physikalisch (voraussichtlich per Ethernet) mit dem Prüfstandssystem verbunden.

Für einen *Prüfstandslauf* müssen nun ein oder mehrere Traces aus der Versuchsdaten-DB geladen werden. Dies geschieht mit Hilfe einer „TraceProvider“ Komponente, die direkt auf die Versuchsdaten zugreift und diese wenn nötig in ein TracePlayer-kompatibles Format wandelt.

Der eigentliche Traceplayer extrahiert zeitlich geordnet aus den Traces (komplette Fahrt) singuläre Trace-Informationen. Diese werden an die jeweils zugeordnete IVS/IRS übermittelt. Eine auf dem Prüfstand laufende „TraceConnector“ Komponente stellt diese Verbindung her und übermittelt die Trace-Informationen.

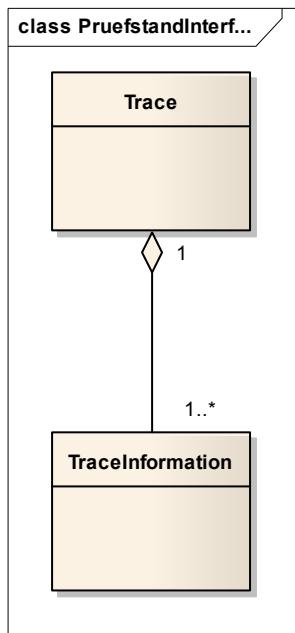


Abbildung 2-13: Prüfstandinterface

Die Steuerung der Komponenten, die prüfstandsseitig laufen, erfolgt mit einer grafischen Benutzeroberfläche.

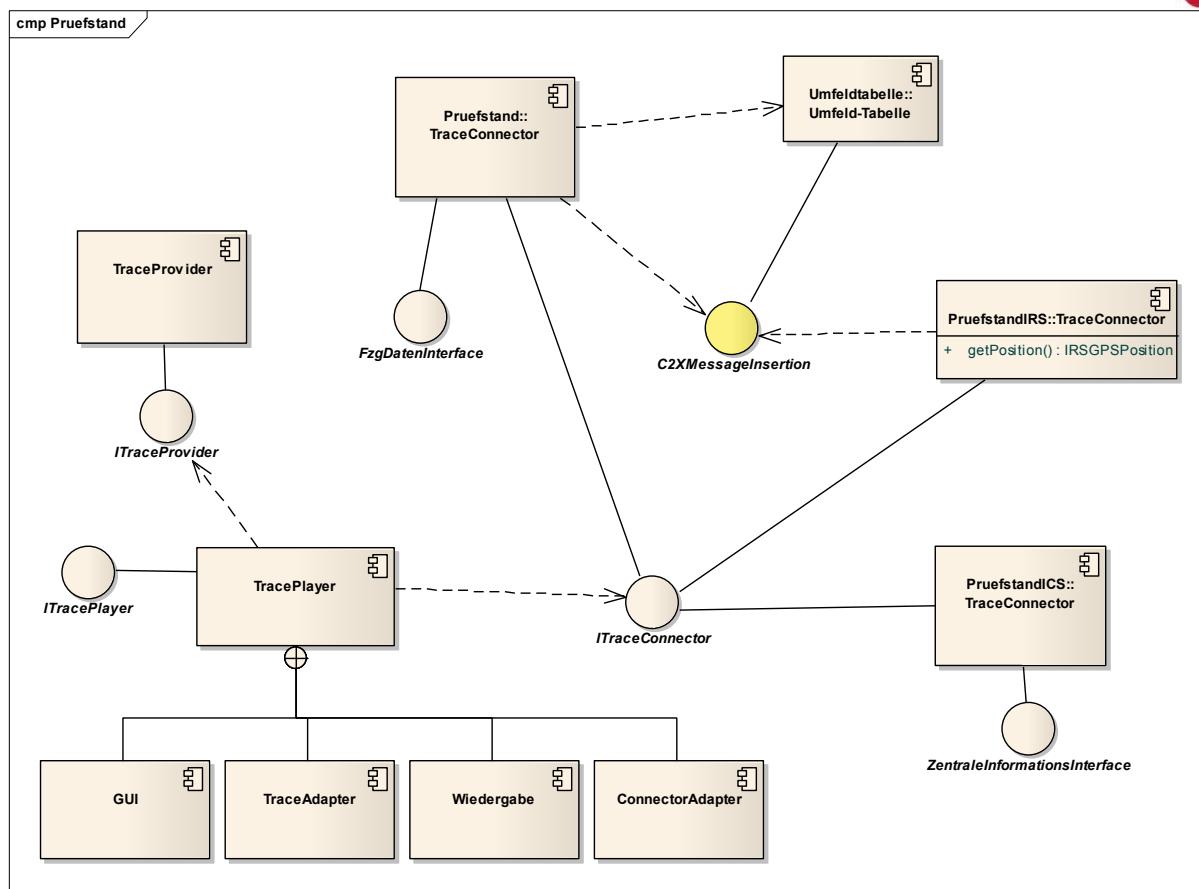


Abbildung 2-14: Überblick über den Prüfstand

TraceProvider

Der TraceProvider ist eine Komponente, die auf die Versuchsdaten-Datenbank zugreift und vorher gespeicherte Logdaten abruft. Diese Komponente dient als Adapter, falls die Art der Versuchsdatenspeicherung verändert wird.

Das Interface zur Versuchsdatenverarbeitung wird nativ angepasst und hängt von der weiteren Implementierung der Datenbank oder der sonstigen Speichermethode ab. Beim Zugriff wird – wenn nötig – bereits unnötige Information gefiltert (z.B. durch entsprechende Datenbankabfrage). Eine weitere Aufbereitung erfolgt wenn nötig nach dem Extrahieren aus der Versuchsdaten-DB. So können z.B. Applikations-Lognachrichten entfernt werden, da diese für den TracePlayer nicht relevant sind.

Um für verschiedene TraceProvider eine einheitliche Methode des Zugriffs vom TracePlayer aus zu bieten ist das ITraceProvider Interface definiert:

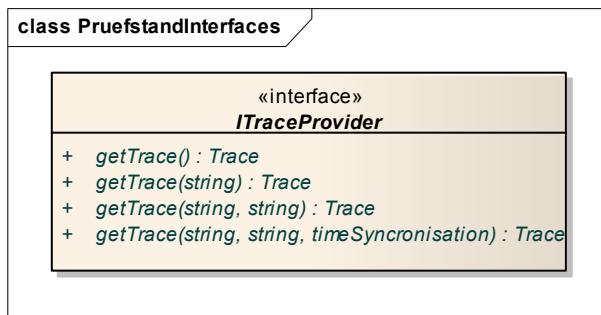


Abbildung 2-15 Interface TraceProvider

TracePlayer

Der TracePlayer ist die Hauptkomponente des Prüfstands. In dieser Komponente werden aus den Traces die einzelnen Trace-Informationen extrahiert und zeitlich geordnet an die vorher spezifizierten ITS Stations geleitet. Der TracePlayer dient also als *Abspieler* und der *Zuordnung*.

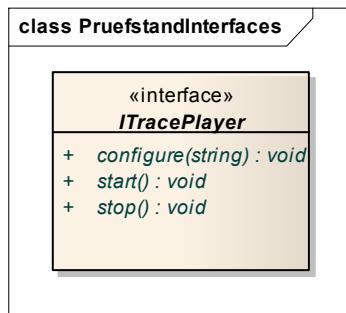


Abbildung 2-16 Interface TracePlayer

Durch mehrere Teilkomponenten werden die geforderten Funktionalitäten erfüllt:

- Die grafische Benutzerschnittstelle (GUI) erlaubt es dem Bediener des Prüfstands, die gewünschten Traces zu laden, Verbindung zu gekoppelten ITS Stations herzustellen und weitere Optionen eines Prüfstandlaufs einzustellen.
- Die TraceAdapter-Komponente lädt den vorher definierten TraceProvider über einen dynamischen Link und nutzt die einheitliche ITraceProvider Schnittstelle, um die gewünschten Traces zu laden oder einen Überblick über verfügbare Traces zu liefern.
- In der Wiedergabe-Komponente befindet sich die Grundfunktionalität des TracePlayers. Aus den geladenen Traces werden die zeitlich relevanten Informationen extrahiert. Dafür läuft ein paralleler Thread, der mit Hilfe eines Timers die passenden Trace-Informationen an den vorher konfigurierten ITS Station Adapter weiterleitet.
- Der ConnectorAdapter ist eine Klasse, die den Zugriff auf die TraceConnectoren bietet, die als Funktion in der IVS/IRS im System-Layer läuft. Wenn sich die entsprechende Station im Prüfstandsaufbau befindet kann über eine Netzwerkschnittstelle eine Verbindung vom ConnectorAdapter zum TraceConnector aufgebaut werden. Die Schnittstelle ist in ITraceConnector definiert.

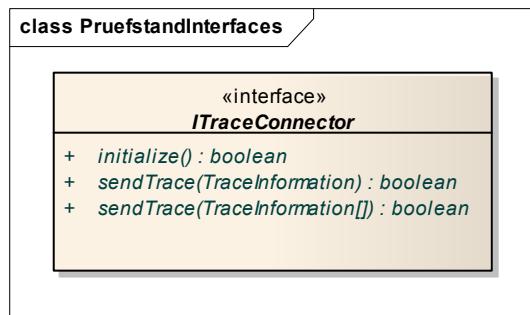


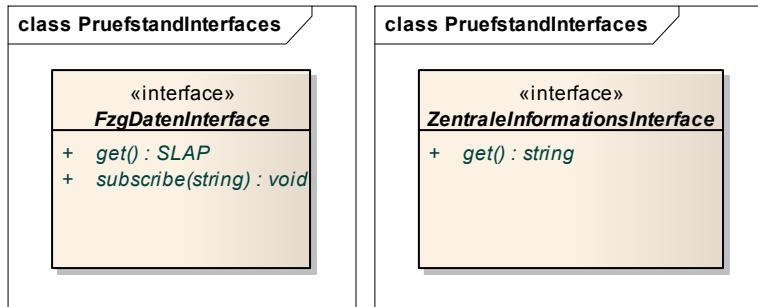
Abbildung 2-17 Interface TraceConnector

TraceConnector

Der TraceConnector läuft als Funktion auf der zu testenden IVS oder IRS. Er ermöglicht es dem TracePlayer, eine Verbindung herzustellen und künstliche Daten in die Umfeldtabelle und die Fahrzeugdaten-Komponente einzuspielen.

Um dies zu ermöglichen, ahmt die TraceConnector Komponente die regulär von der Fahrzeugdaten-Komponente und der Umfeldtabelle genutzten Interfaces nach.

Weiterhin kann eine TraceConnector-Komponente auch in der ICS genutzt werden. Diese bietet dann auch ein weiter zu spezifizierendes Interface für zu emulierende zentrale Daten an.



Ablauf einer Tracewiedergabe

Wenn eine Tracewiedergabe im Prüfstand gestartet wird, lädt er TracePlayer die definierten Traces aus der Versuchsdaten-DB. Dann wird eine Verbindung zu den angeschlossenen IVS/IRS hergestellt.

Sobald diese Initialisierungsphase beendet ist, beginnt die Wiedergabe des Traces, indem zeitlich geordnete TraceInformationen an die jeweils zugeordneten TraceConnectoren versandt werden. Diese nutzen die internen Interfaces, um den zu testenden Funktionen echte Daten vorzutäuschen. Die Reaktion dieser Funktionen wird mit Hilfe der Versuchsdatenverarbeitung geloggt.

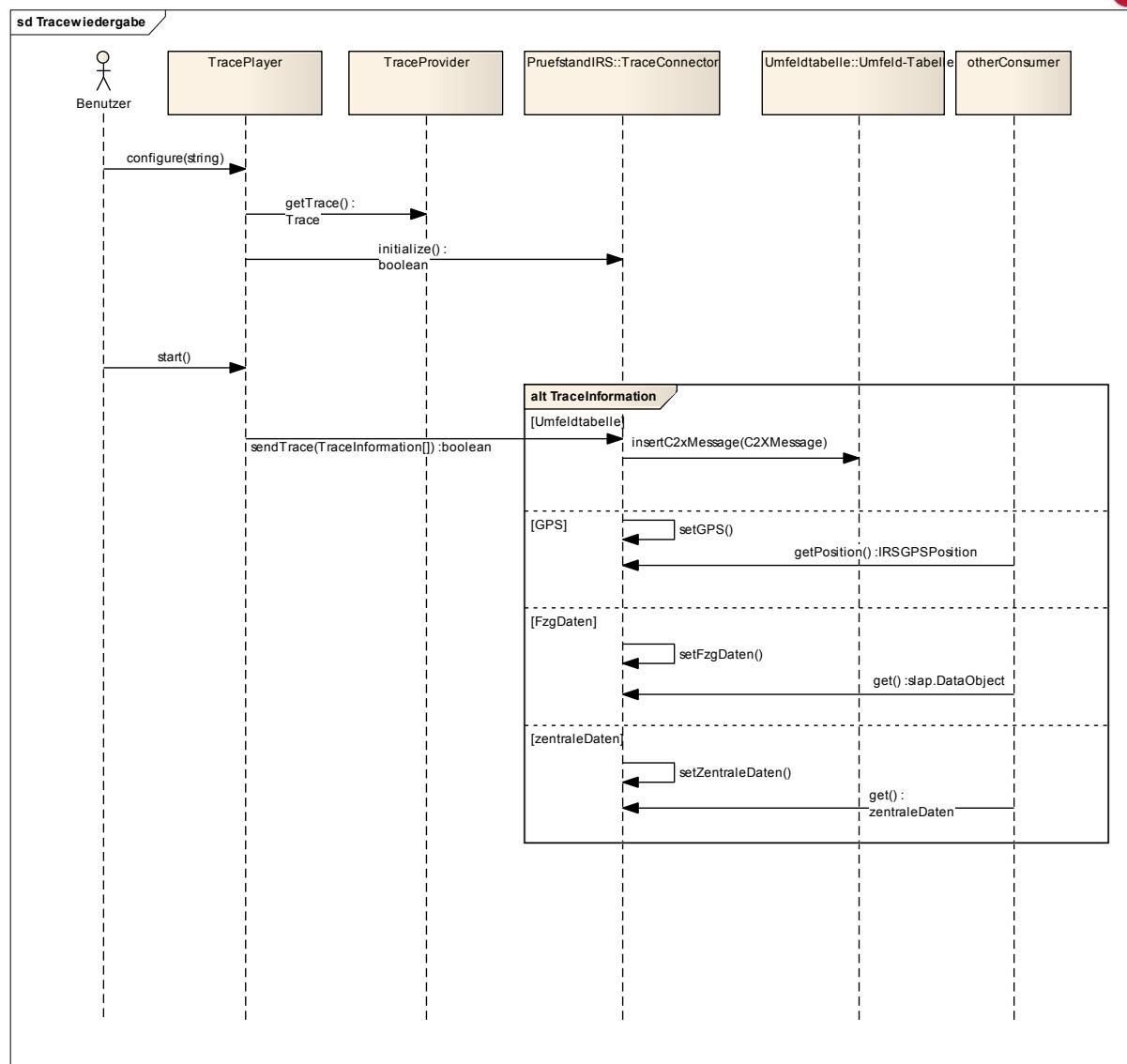


Abbildung 2-18: Ablaufdiagramm zur Tracewiedergabe

Testlauf Template

Das Prüfstandtemplate dient Funktionsentwicklern dazu, ihre Prüfstandsläufe hinsichtlich der Konfiguration, der Teststart und des gewünschten Logging-Profil festzulegen. Es sind folgende Daten erfassst:

- Name des Tests (oder ID)
- Autor
- Ansprechpartner
- Zu aktivierende Funktionen (ID aus FET-Liste)
- Zentralseitige Funktionen
- Zu testendes Anwendungsszenario
- Log-Profil

- Testaufbau (Hardware und System-Konfiguration)
- Traces und Zuordnung zu IVS/IRS (+Offset)
- Zu verwendende Systemzeit
- Dauer des Prüfstandlaufs (geschätzt)
- Anmerkungen

2.4.2 Versuchsdatenverarbeitung

Die Versuchsdatenverarbeitung teilt sich in Logging (Messdaten) und Monitoring (Livedaten) auf. Beides sind jeweils eigene Funktionen, trotzdem können sie nicht ganz getrennt betrachtet werden, weil sich beide Funktionsanteile in denselben Komponenten abspielen. Eine Reihenfolge in der Beschreibung zu finden fällt schwer, da es Abhängigkeiten in beide Richtungen gibt.

2.4.2.1 Plattformen

Abbildung 2-19 zeigt mit Hilfe eines Komponentendiagramms eine Übersicht aller vom Logging verwendeten Komponenten. In den folgen Unterkapiteln sind die einzelnen Plattformen näher beschrieben.

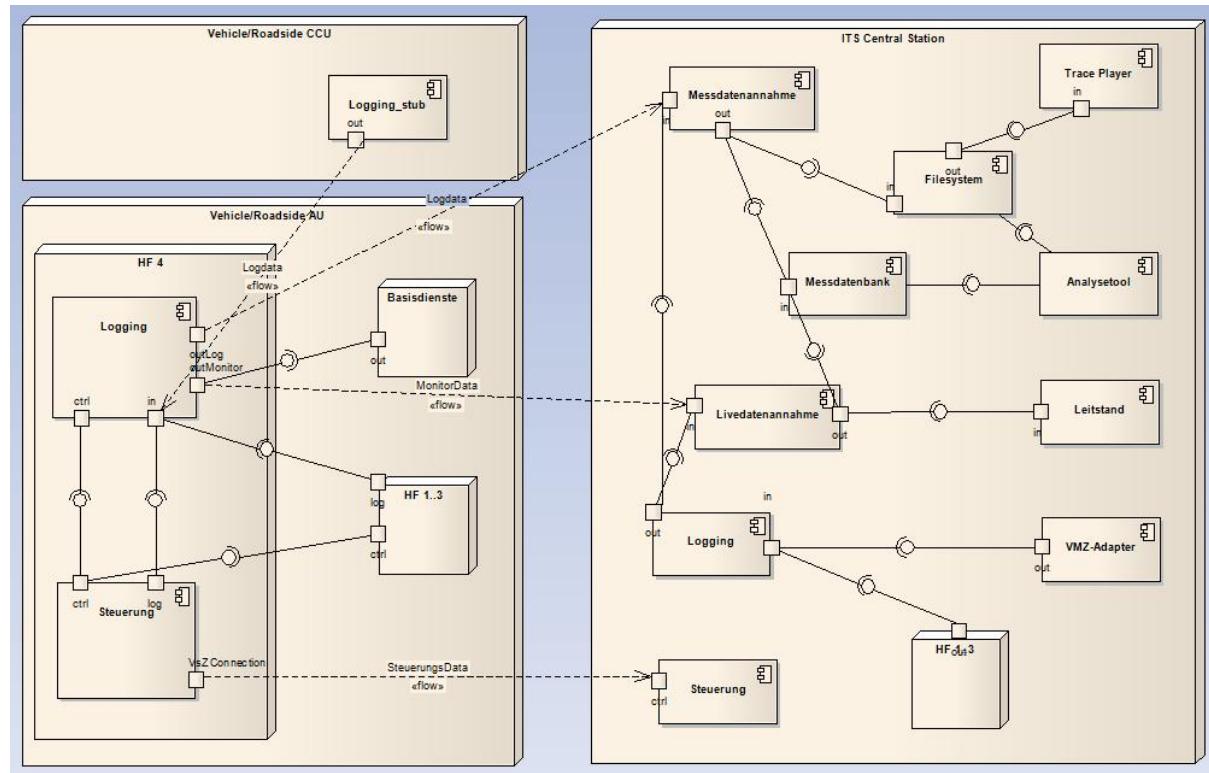


Abbildung 2-19: Komponentendiagramm der Versuchsdaten-Architektur

Vehicle/Roadside Communication Unit

Auf der Communication Unit, dargestellt in Abbildung 2-20, wird die Komponente Logging_stub zur Verfügung gestellt, die den anderen Komponenten dort ein Logging-Interface bietet. Der Logging_stub leitet diese Logs dann bei Auftreten über eine TCP Verbindung an die im lokalen Netzwerk liegende Application Unit weiter. Über diese Möglichkeit können dann auch Logs die die Communication Unit direkt betreffen, wie etwa Nachbarschaftstabellen, Ausnahmebehandlungen oder Softwareabstürze, mit in die zentralisiert abgelegten Logdateien aufgenommen werden. Die Logs werden als einzelne Strings ohne weitere Vorverarbeitung über die am Anfang zu öffnende TCP Verbindung geschickt. Weitere Logik ist im Logging_stub nicht notwendig. Der Logging_stub wird genau wie das Logging auf der AU nur die Logmethoden anbieten, die in einem Dictionary vordefiniert sind. Dies wird für die AU noch genauer erklärt und gilt auch für diese Komponente.

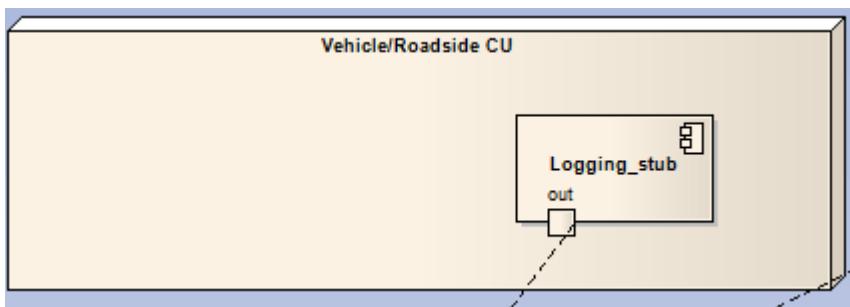


Abbildung 2-20: Vehicle/Roadside CU

Vehicle/Roadside Application Unit

Auf der Vehicle AU, dargestellt in Abbildung 2-21, läuft innerhalb einer OSGI-Umgebung das Logging Bundle. Dieses beinhaltet die meiste Logik im gesamten Logging-Ablauf. Es bietet der Steuerung und den anderen Funktionen aus den Hauptfunktionsgruppen 1 bis 3 ein Loginterface an, der Steuerung auch ein Controlinterface, über das das Logging gesteuert werden kann. Steuerung bedeutet für das Logging, das ein neues Logprofile, also eine Konfiguration der Daten-Abonnements, ausgewählt wird. Bei einigen Datenquellen, das sind Umfeldtabelle, VAPI und GPS, wird sich das Logging selbst mit den erforderlichen Parametern registrieren, um auf diese Weise Ressourcen zu schonen.

Logprofile sollen vordefiniert zur Verfügung gestellt werden und im Logging vorhanden sein. Solch ein Profil definiert die Konfiguration, welche Werte das Logging wie von Umfeldtabelle und VAPI bezieht. Application Logs, die an das Logging übergeben werden können nicht beeinflusst werden. Die Profile werden aufgrund der Anforderungen in den Versuchs und Testdefinitionen definiert und auf ein überschaubares Set im ein- bis niedrigen zweistelligen Bereich zusammengefasst.

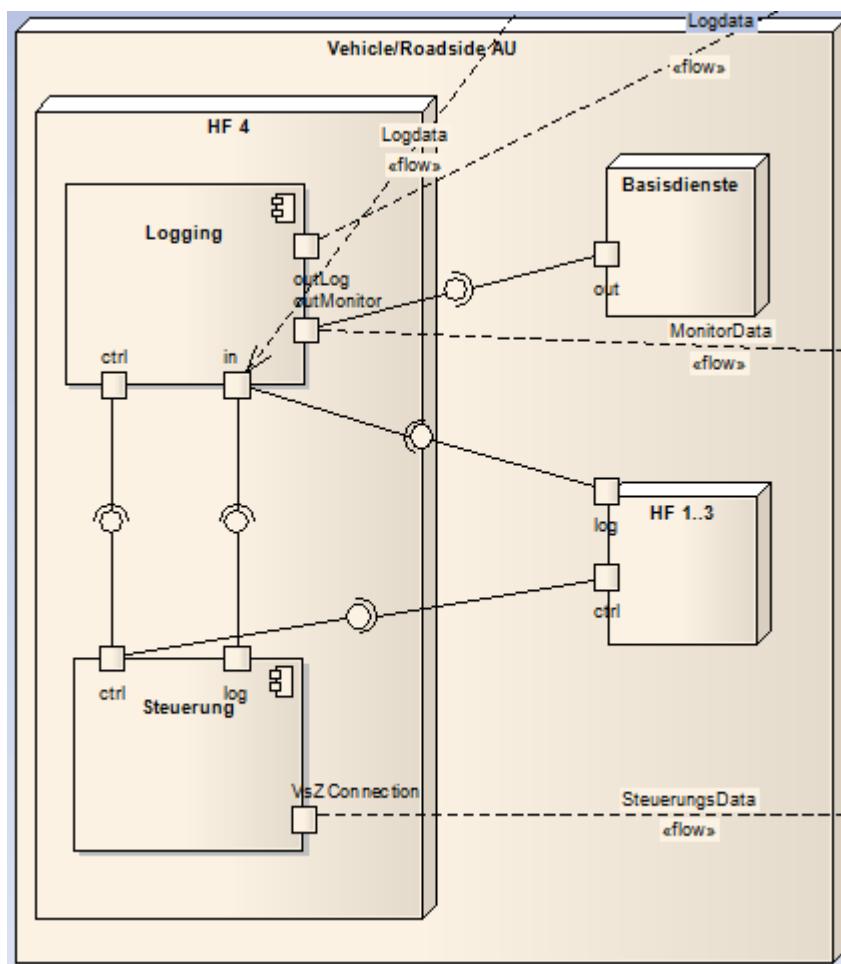


Abbildung 2-21: Vehicle/Roadside AU

ITS Central Station

Die ITS Central Station, dargestellt in Abbildung 2-22, nimmt die Daten vom Logging-Bundle an zwei Stellen entgegen. Die Logdaten an der Messdatenannahme und die Monitoringdaten an der Livedatenannahme. In der ITS Central Station läuft ebenfalls ein Logging-Bundle, ähnlich zu dem in den Application Units, das den in der ITS Central Station laufenden Funktionsanteilen der Funktionen aus den Hauptfunktionsgruppen 1 bis 3 ebenfalls ein Loginterface bietet. Des Weiteren bildet es eine Schnittstelle vom Loggingsystem zu der VMZ-Adapter genannten Komponente, die Verkehrs- und Wetterdaten aggregiert bereitstellt. Dieses Logging-Bundle liefert die Daten ebenfalls an Mess- und Livedatenannahme an. Konsumenten für die Mess- und Livedaten, wie etwa der Leitstand oder der Traceplayer, greifen entweder direkt auf die Datenbank zu, in Kombination mit Zugriff auf das Dateisystem oder aber nur auf das Dateisystem.

In der Datenbank, die in der ITS CS läuft, werden nur Daten aus dem Monitoring abgelegt, damit die Versuche relativ zeitnah validiert werden können. Die Logfiles werden nur im Filesystem auf dem Fileserver abgelegt. Sie sind mit Absicht in einem Format gehalten, in dem sie in Datenbanksysteme kopiert werden können, was um Größenordnungen schneller ist als sie einzufügen. Wer also die Logs in einer Datenbank halten will, um sie dort auszuwerten, der erhält dadurch eine Möglichkeit alle entstandenen Logs in Echtzeit zu importieren, muss dies aber vor Ort erledigen. Es wird keine zentrale rationale Datenbank geben, in der alle Logs abgespeichert sind.

Es gibt zurzeit den Vorschlag, beim Import in eine Datenbank für die Auswertung für jeden Tag eine neue Tabelle zu erstellen und dort die neuen Logfiles zu importieren. Der Aufwand die betreffenden Indizes zu erzeugen ist groß und kann erhalten bleiben wenn neue Daten in neue Tabellen eingespielt werden. Abfragen auf alle Daten müssten dann mittels union-Kommando gestellt werden.

Die Steuerungskomponente in der ITS Central Station ist der zentrale Bestandteil der Versuchsteuerung und für das Logging vor allem notwendig, um neue Logprofile zu übertragen und die Messdatenübertragung anzustoßen.

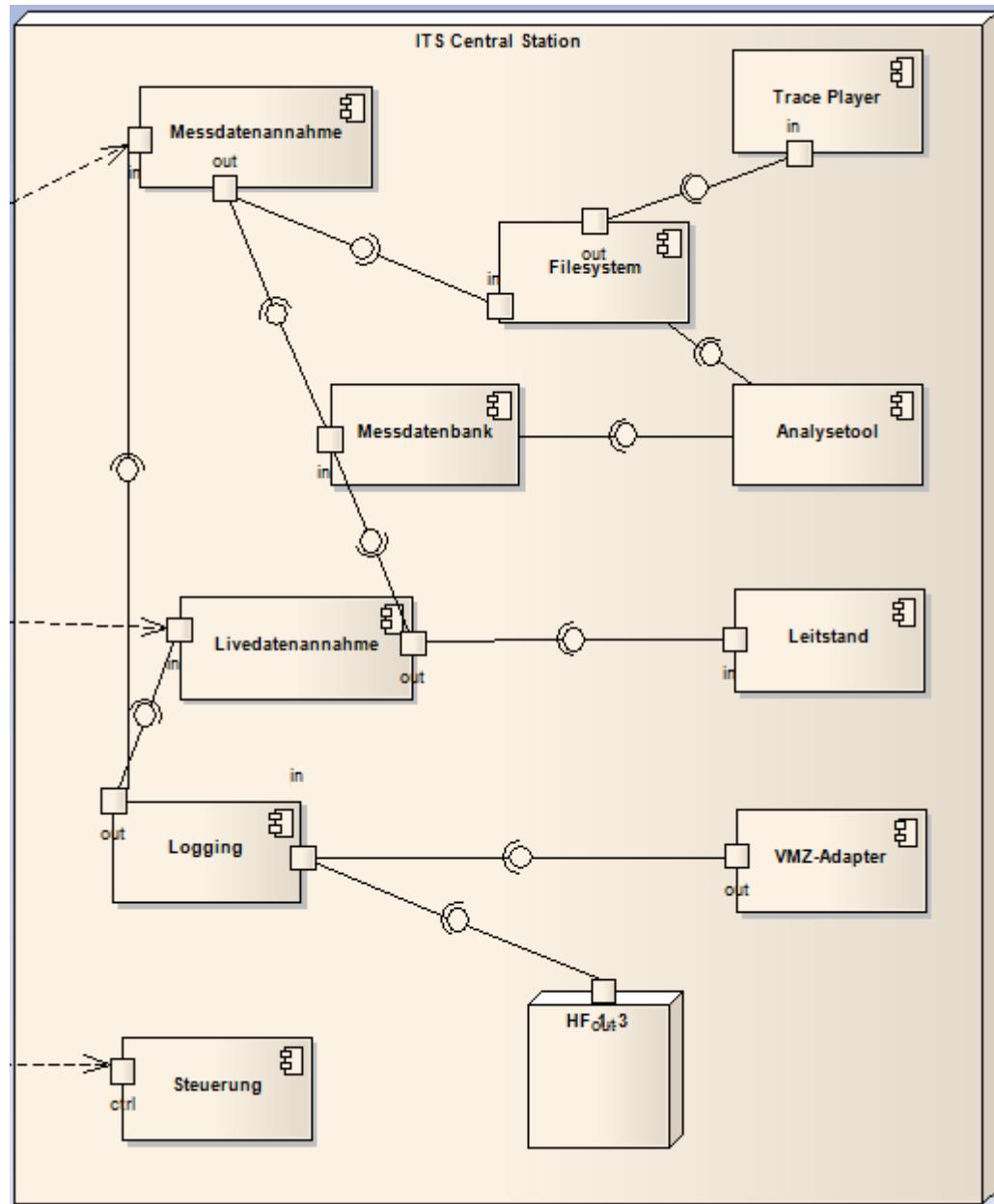


Abbildung 2-22: ITS Central Station

2.4.2.2 Messdaten

Das Logging ist in sim^{TD} die einzige Möglichkeit, um zentral detaillierte Rückschlüsse auf die Funktion aller Einheiten, also IVS, IRS und ICS, im Versuchsgebiet zu ziehen. Dafür werden

auf allen zu beobachtenden Systemen Komponenten des Loggings installiert, die dort als Datensenken dienen und die gesammelten Informationen dann zentral in der Versuchszentrale zur Verfügung stellen.

Messdatenerfassung

Das Logging-Bundle bildet eine Datensenke für die von der Communication Unit kommenden Logs, die Umfeldtabelle, die Fahrzeugdaten (CAN & GPS) und die auf der AU laufenden Funktionen und Komponenten. Um das Logging individuell für jede einzelne Einheit konfigurieren zu können, gibt es Logprofile. Diese sind in Absprache mit TP4 vordefiniert und beschreiben einen Filter, um die Menge der Logdaten an der Quelle anpassen zu können. Um in der Implementierung aufgrund der sehr limitierten Rechenleistung der AU den unnötigen Rechenaufwand zu minimieren, wird das Schema der Datensenke an zwei Stellen gebrochen:

Das Logging-Bundle abonniert sich mit den im Filter gesetzten Einstellungen bei Umfeldtabelle und Fahrzeugdaten, um genau so viele Datenobjekte zu erzeugen, wie für das Logging zum jeweiligen Zeitpunkt auch gebraucht werden. Bei den Logs von der CU und den Applikationen ist dies natürlich nicht möglich. Hier kann man nur nachträglich filtern, wobei das nur eingeschränkt sinnvoll ist, da es hier einen Trade-Off zwischen Rechenaufwand und Speicherplatz gibt. Auch bei diesen Logproduzenten sollte man schon an der Quelle einschränken und nur Logs produzieren, die für mindestens einen der Interessierten auch relevant sind.

Es wird vorgeschlagen einen Prozess innerhalb von sim^{TD} zu etablieren, der folgendermaßen aussieht:

Alle Dateninteressierten in sim^{TD}, das sind in erster Linie die Auswerter und diejenigen die die technischen Tests durchführen, müssen sich mit allen Funktionen und Komponenten auf CU und AU beschäftigen und beschreiben, welche Daten sie für ihre Arbeit brauchen. Die Funktions- und Komponentenentwickler entlehnen aus der Beschreibung das absolut notwendige Set an Lognachrichten. Dieses kann von den Entwicklern dann noch um weitere Logs erweitert werden und bildet dann eine Ressource des Loggingsystems. Es wird damit sichergestellt dass alle internen und externen Daten der Funktion die für die genannten Stakeholder notwendig sind auch über das Loggingsystem zentral für die Auswertung gespeichert werden.

Beispiel für die Funktion Ampelphasenassistent:

Auswerter beschreiben: Wir brauchen zum Zeitpunkt einer Fahrempfehlung auch die für die Empfehlung zugrundeliegenden Werte Ampel-Id, Entfernung zur Ampel, Dauer bis zum nächsten Phasenwechsel und geschätzte Rückstaulänge.

Hier sind auch funktionsinterne Werte gefordert, deren Auswertung notwendig ist, die ohne die formulierte Anforderung aber möglicherweise nicht geliefert werden würden, wenn es keine definierte Schnittstelle zwischen Auswerter und Entwickler gäbe. Üblicherweise sind das in der Softwareentwicklung dieselben Personen, in sim^{TD} findet aber eine besondere Auswertung unter nichttechnischen Gesichtspunkten statt, die allerdings fast ausschließlich auf technischen gemessenen Werten basiert.

Das Logging in sim^{TD} soll wie auch in anderen Systemen üblich in Textdateien erfolgen. Dabei ist vom Loggingsystem aus gewährleistet, dass die Dateien auch in kurzen Zeitabständen auf den Festspeicher geschrieben werden, damit im Falle eines Absturzes oder anderen Ausfalls zeitnahe Lognachrichten nicht im Hauptspeicher verloren gegangen sind. Die Logdateien erhalten eine feste Anzahl von Logs - geplant ist zwischen 1000 und 10000. Dies ist eine Variante die Logfiles mit wenig Rechenaufwand in ähnlicher Dateigröße

zu halten. Die Logfiles sollen in unkomprimierter Form kleiner als ein Megabyte sein, um bei Verbindungsabbrüchen bei der Übertragung im Mobilfunknetz bzw. WLAN nicht zu viel Zeit durch wiederholte Übertragungen zu verlieren. Außerdem sollen die Dateien bei der Auswertung gut benutzbar sein und dies wird mit zunehmender Dateigröße schlechter. Gegen zu kleine Dateien spricht die absolute Menge an Logs die insgesamt in sim^{TD} entstehen und die damit entstehenden Clusterverluste im Dateisystem.

Die Formatierung der Logs innerhalb der Logdateien verzichtet auf den Einsatz von Text und beschränkt sich auf ausschließlich auf numerische Mess- und Referenzwerte. Der Prozess, um die Referenzwerte (IDs) zu bekommen wurde schon mit dem Beispiel Ampelphasenassistent erwähnt und wird nun konkreter ausgeführt.

Jede Funktion innerhalb von sim^{TD} stellt dem Loggingsystem eine Textbasierte Ressourcendatei zur Verfügung. Diese Datei ist eine kommaseparierte Textdatei und enthält pro Zeile die Definition eines funktionsspezifischen Logtyps.

Beispiel:

Ampelphasenassistent.res

2200, ...

...

2253, Nachricht von LSA, LSA-ID, Entfernung in Dezimeter zum EgoFzg, [0=grün,1=rot]

...

An erster Stelle jeder Zeile steht die Logidentifikationsnummer, eine Konkatenation aus der Id für die Komponente oder Funktion und einer fortlaufenden Nummer. Zum aktuellen Zeitpunkt erscheinen 100 funktionsspezifische Logtypen und 100 Quellen als ausreichend und somit ist diese Nummer immer vierstellig. Die Ressourcendatei des FET Ampelphasenassistent „Ampelphasenassistent.res“ hat somit maximal 100 Zeilen.

An zweiter Stelle folgt eine Beschreibung des Logs, an den beliebig vielen weiteren Stellen folgen die Beschreibungen der Werte, die diesem Logtyp immer angehängt sein sollen. Diese Werte sind bis auf eine Ausnahme (für individuelle Eingaben vom Fahrer ist die Ausnahme String) immer vom Typ Integer, das heißt, dass die Beschreibung immer die Einheit enthalten muss, in der der Messwert abgespeichert wird. Will man also etwas im Zentimeterbereich messen können, muss man dies hier formulieren und beim Aufrufen des Logs darauf achten, dass man möglicherweise die funktionsintern benutzte Einheit (z.B. Meter als Double) in die Integereinheit umrechnet. Wenn man Funktionsintern also 12,2 Meter misst und die maximal definierte Messauflösung im Zentimeterbereich liegt, dann muss man an die Logfunktion den Integer 1220 übergeben. Messwerte die keine numerischen Messwerte sind, müssen in der Definition der Logtypen auf solche gemappt werden. Eine solche Enumeration wird innerhalb von eckigen Klammern zusammengefasst, kommasepariert und Werte zu Zahlen mit einem Gleichzeichen zugeordnet. Im Beispiel wird die Farbe grün der Zahl 0 und die Farbe rot der Zahl 1 zugeordnet.

Die Ressourcendatei wird benötigt, um beim Erstellen des Logging-Bundles die entsprechenden Einträge zu generieren, um diese Abstraktion für die Entwickler zu kapseln und innerhalb der IDE die natürlichsprachlichen Beschreibungen zu präsentieren.

Eine Standardressourcendatei wird vom Logging selbst zur Verfügung gestellt und enthält generelle Logtypen, die von allen Funktionen eingesetzt werden können.

Das Interface, das vom Logging bereit gestellt wird lässt sich noch nicht beschreiben, da es nicht eine generelle Logmethode geben wird, sondern für jedes Log eine spezifische. Die dafür verantwortliche Klasse wird aus den Ressourcendateien (Dictionary) generiert werden und einen Wrapper für das Logging darstellen. Dies verhindert Fehler bei der Eingabe, die später gar nicht mehr herausgefiltert werden könnten und generell Wildwuchs bei den Formulierungen der Logs. Wenn man nur eine Logmethode anbietet, die Logs aber automatisiert auswerten will, müssen die Identifikatoren für die Logs irgendwo definiert und benutzt werden. Die automatische Umwandlung dieses Wissens in einen Wrapper ist aufgrund der Verschleierung der Ids einfacher benutzbar und weniger fehleranfällig. Der Nachteil den man sich damit erkauf ist eine eingeschränkte Flexibilität, weil man das Logging-Bundle ändern und neu verteilen muss, wenn man neue Logmethoden hinzufügen will. Da dies allerdings keine bekannte Anforderung in sim^{TD} ist, wird dieser Nachteil in Kauf genommen. Die Dictionaries müssen durch die FETs und TP4 bereitgestellt werden. Erst dann wird das daraus generierte Interface verbreitet werden können. Bei diesem Prozess sollten die FETs einen Initialvorschlag machen den TP4 dann reviewt und erweitert. Letztendlich soll nichts gelogged werden, was TP4/AP33 nicht braucht.

Ein Log zum oben erwähnten Beispiel des Ampelphasenassistenten könnte in der resultierenden Logdatei folgendermaßen aussehen:

1211613815462, 2253, 123, 2535, 1

Die Logs sind ebenfalls kommasepariert und zeilenbasiert. Das Logging-Bundle setzt einen Unixzeitstempel (Longwert der Anzahl an verstrichenen Millisekunden seit dem 01.01.1970) als ersten Wert, darauf folgen die LogID und die Messwerte (in derselben Reihenfolge wie in der Logtypdefinition).

Die Logproduzenten werden in simTD folgendermaßen benannt:

IRS000 - IRS110

IVS000 - IVS400

ICS000

Abgelegt werden die Logfiles im Dateisystem der loggenden Einheit, benannt nach dem Schema <produzentenid>_<startzeitstempel>_<endzeitstempel>.zip in gepackter Form. Ein Beispiel hierfür ist IVS123_1322389105_1322389225.zip.

Die Vorteile dieses Verfahrens sind im wesentlichen die Platzersparnis in den Logfiles, die gute Packbarkeit der Logfiles, welche durch wiederkehrende Werte besser wird, und die Möglichkeit die Logs automatisiert auswerten zu können. Der wesentliche Nachteil ist die beschwerlichere Analyse durch Menschen, die mit Hilfe der vorausgesetzten Ressourcendateien und ein wenig Software wiederhergestellt werden kann. Dies ist aber bei der prognostizierten Datenmenge sowieso keine gute Option.

Die Homogenisierung der Logs ist eine Strategie um die Datengröße zu verringern. Die Datenmenge, also die Anzahl der Logs, wird mit folgenden Strategien reduziert:

Um die Redundanz innerhalb der Logs zu verringern wird die VAPI so abonniert, dass diese nur bei Veränderung eines Wertes diesen neu anbieten. Beim Schreiben in eine neue Datei wird das berücksichtigt und alle Werte werden einmal neu abgefragt.

Keine technische sondern eine inhaltliche Möglichkeit die Logs gering zu halten bietet die Möglichkeit Logevents einzuführen. Die Idee ist es Daten, die einerseits für eine spezielle

Untersuchung interessant sind aber andererseits sehr viele Logs produzieren nur für einen bestimmten kurzen Zeitraum aufzuzeichnen, um kompromisshaft die Interessen der Auswerter mit den technischen Grenzen zu vereinen. Ein Beispiel dafür könnte die Aufzeichnung aller Netzwerkkommunikation für einige Stunden sein. Für eine Analyse der Kommunikation braucht man sicher nicht Daten aus dem gesamten Versuchszeitraum und schränkt sich so auf das nötigste ein. Umgesetzt wird dies einfach mit einem entsprechenden Loggingprofil.

Messdatenübermittlung

Die Logdateien werden wie im Kapitel Messdatenerfassung beschrieben im Dateisystem der loggenden Einheit abgelegt. Hier warten sie darauf durch die Versuchssteuerung ein Kommando zu bekommen, um den Übertragungsprozess zu starten. Die Dateiübertragung wird nach aktuellem Stand mit mPRM, einer Überwachungssoftware von Prosys, durchgeführt. Die Messdatenannahme sendet nach der erfolgreichen Übertragung und Verifizierung mit Hilfe einer ebenfalls Übertragenen Prüfsumme über die Versuchssteuerung einen Läschauftag, der nach erfolgter Löschung von der loggenden Einheit auch bestätigt wird. Auf diese Weise werden alle vorliegenden Logdateien an die Messdatenannahme übertragen. Die Versuchssteuerung kann per Kommando die Übertragung der Logdateien auch wieder stoppen, obwohl noch weitere Dateien zu übertragen wären und der Auftrag noch nicht komplett abgearbeitet wurde, etwa weil ein neuer Versuch beginnt und der Kanal nun freigehalten werden soll. Das Übertragungsprotokoll wird genauer im Deliverable D21.4 Spezifikation der Kommunikationsprotokolle erläutert, ein Sequenzdiagramm dazu zeigt Abbildung 2-23.

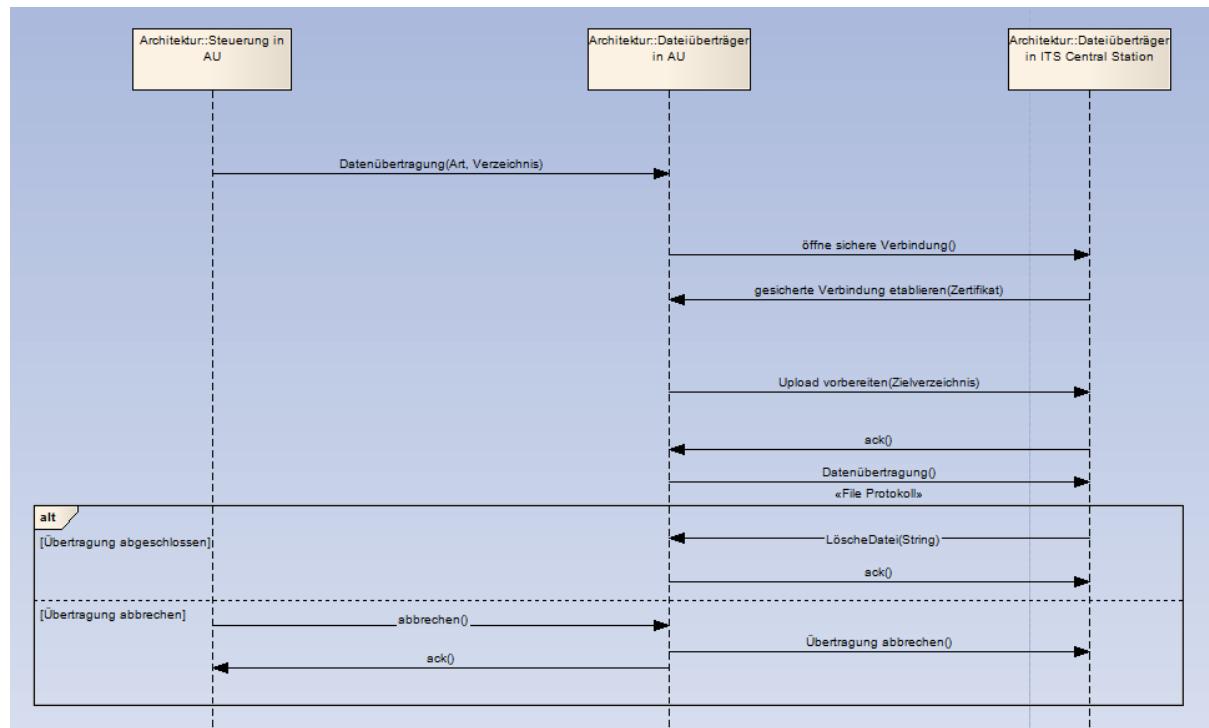


Abbildung 2-23: Sequenzdiagramm der Messdatenübermittlung

Das Übertragungsmedium wird in der Regel das Mobilfunknetz sein. WLAN nach Standard 802.11b/g könnte auf dem Testgelände alternativ genutzt werden. Die Möglichkeit private AccesPoints der Fahrer der freien Flotte nur für deren Auto zu nutzen bestünde ebenfalls. Auf dem Testgelände mit wenigen Autos und umfangreichen zum Debugging verwendeten

Logprofilen können auch USB-Sticks oder ähnliche Offline-Datenübertragungsverfahren genutzt werden.

Um die Menge der Daten sinnvoll einschränken zu können haben wir folgende Hochrechnung der möglichen Datenmenge gemacht:

Als Übertragungsverfahren kann EDGE als Minimum angenommen werden. Andererseits lasten 500 (400 IVS und 100 IRS) Datensender per GPRS den 32MBit Link der ITS Central Station aus. Höhere Datenübertragungsraten wären also nicht gleichzeitig möglich. Angesichts dessen, dass während der laufenden Versuche keine Messdaten übertragen werden sollen, um die Versuche nicht zu beeinflussen, und die Fahrer nach Beendigung der Versuche zu einem Stellplatz fahren werden an dem die AU eine Nachlaufzeit von einer halben Stunde (aktuelle Annahme) haben wird ist das Zeitfenster um Testdaten zu übertragen auf etwa 30 Minuten beschränkt. Techniken die die AU nachts aufwecken könnten, könnten diese Situation möglicherweise entspannen, sind aber zum jetzigen Zeitpunkt nicht vorgesehen. In einer Stunde können per EDGE im Idealfall ca. 36 MB an gepackten Daten übertragen werden, dafür muss man aber praktisch allein in der Zelle sein. Das entspricht in etwa 80 MB an Rohdaten, die pro Arbeitstag entstehen dürfen.

Ob die Übertragung der Logdaten im Netz der Telekom realistisch ist wird zum aktuellen Zeitpunkt geprüft. Die Lösung für die freie Flotte muss dann eine weitere Einschränkung des Loggings sein, bei den Hired Drivers wäre das WLAN verpflichtend.

Messdatenspeicherung

Die Messdatenspeicherung zeichnet sich im Wesentlichen dadurch aus die Logfiles im Dateisystem eines Fileservers zu hinterlegen. Das ursprünglich favorisierte Konzept alle Logs typisiert in einer großen Datenbank vorzuhalten lässt sich mit den in sim^{TD} dafür vorgesehenen Mitteln nicht realisieren. Die Gründe dafür sind die große Anzahl der Logs und auch der dafür vorgesehene Platz. Eine Hochrechnung des gewünschten Datenaufkommens ergibt ca. 500.000 Logs pro Stunde pro AU, das aktuell errechnete technische Limit liegt bei 144.000. Es gibt zum aktuellen Zeitpunkt von allen betroffenen Partnern Hochrechnungen, die je nach Wahl der Parameter mehr oder weniger hoch ausfallen. Hier beschrieben ist eine konkrete Hochrechnung deren Gesamtvolume im unteren Drittel liegt. Zum jetzigen Zeitpunkt gibt es hier noch keinen Konsens.

Einige Strategien die Anzahl der Logs einzuschränken wurden schon im Kapitel Messdatenerfassung beschrieben. Eine scharfe Grenze die verhindert alle Logs in der Datenbank zu speichern ist die Anzahl der Inserts die das verwendete Datenbanksystem pro Sekunde ausführen kann. Tests haben ergeben, dass diese Leistung sich in Größenordnungen von einigen hundert bis wenigen Tausend abspielen wird und wir damit bei realistisch geschätzten 500 möglichen Inserts pro Sekunde von jeder loggenden Einheit nur ein Log pro Sekunde in der Messdaten-DB speichern könnten.

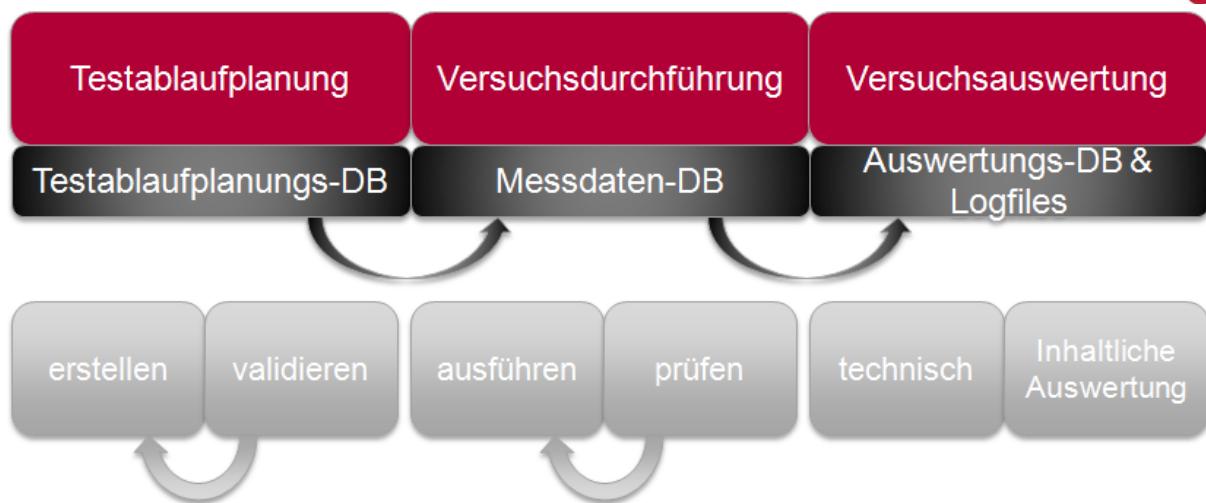


Abbildung 2-24: Prozessübersicht bei der Versuchsdurchführung im Bezug auf die Logdaten

Abbildung 2-24 zeigt in welcher Prozessphase im Versuchszeitraum welche Daten wo benötigt werden. In der Testablaufplanung geht es darum Tests und Versuche zu definieren. Dafür werden sie erstellt und validiert, was sich auf die Erstellung rückkoppeln kann. Die Definitionen der Versuche anhand fester Parameter finden sich dann in einer Testablaufplanungs-DB. Während des laufenden Versuchs entstehen wiederum Daten, die in der Messdaten-DB gespeichert werden, und deren Nutzen es ist überprüfen zu können, ob die Versuchsdurchführung zur Definition in der Versuchsplanungs-DB passt oder die Durchführung wiederholt werden muss. Wenn ein Versuch wie definiert durchgeführt wurde, kann man die Daten aus der Messdaten-DB in Kombination mit den gesammelten Logfiles zur technischen und inhaltlichen Auswertung nutzen.

Die Daten in der Messdaten-DB sind im wesentlichen auf etwa einen Datensatz pro AU und Sekunde beschränkt, sollen aber möglichst schnell einer Versuchsüberprüfung gewährleisten.

Für Logs ist das Dateisystem eine ideale Ablage, weil sie wegen der gepackten Ablage in untypisierter Form wenig Platz verbrauchen und der Zugriff auf die Dateien schneller ist als auf Daten in einer Datenbank.

Wie genau das Datenbankschema für die Messdaten-DB (der Name ist historisch begründet, sie enthält die Daten des Monitorings) inhaltlich sein wird, kann zum aktuellen Zeitpunkt nicht definiert werden, da dafür die Vorgaben der Nutzer noch fehlen, aber es wird sich voraussichtlich um Zeitstempel, QuellenId, Position, Statusflags und einige ausgewählte CAN-Daten handeln, alle Daten die das Monitoring überträgt.

Die geplante Vorgehensweise zur Befüllung der Messdaten-DB ist, die Livedaten zu nutzen, um in der Datenbank möglichst schnell alle verfügbaren Werte vorzuhalten.

Die Datenbank dient in erster Linie der AdHoc Validierung durchgeföhrter Versuche und nicht der späteren Analyse. Sie muss also nicht bei den auswertenden Partnern vorgehalten werden.

Nach aktuellem Planungsstand wird für die Synchronisierung der Logdateien ein Verteilserver eingesetzt, welcher im Folgenden kurz beschrieben wird.

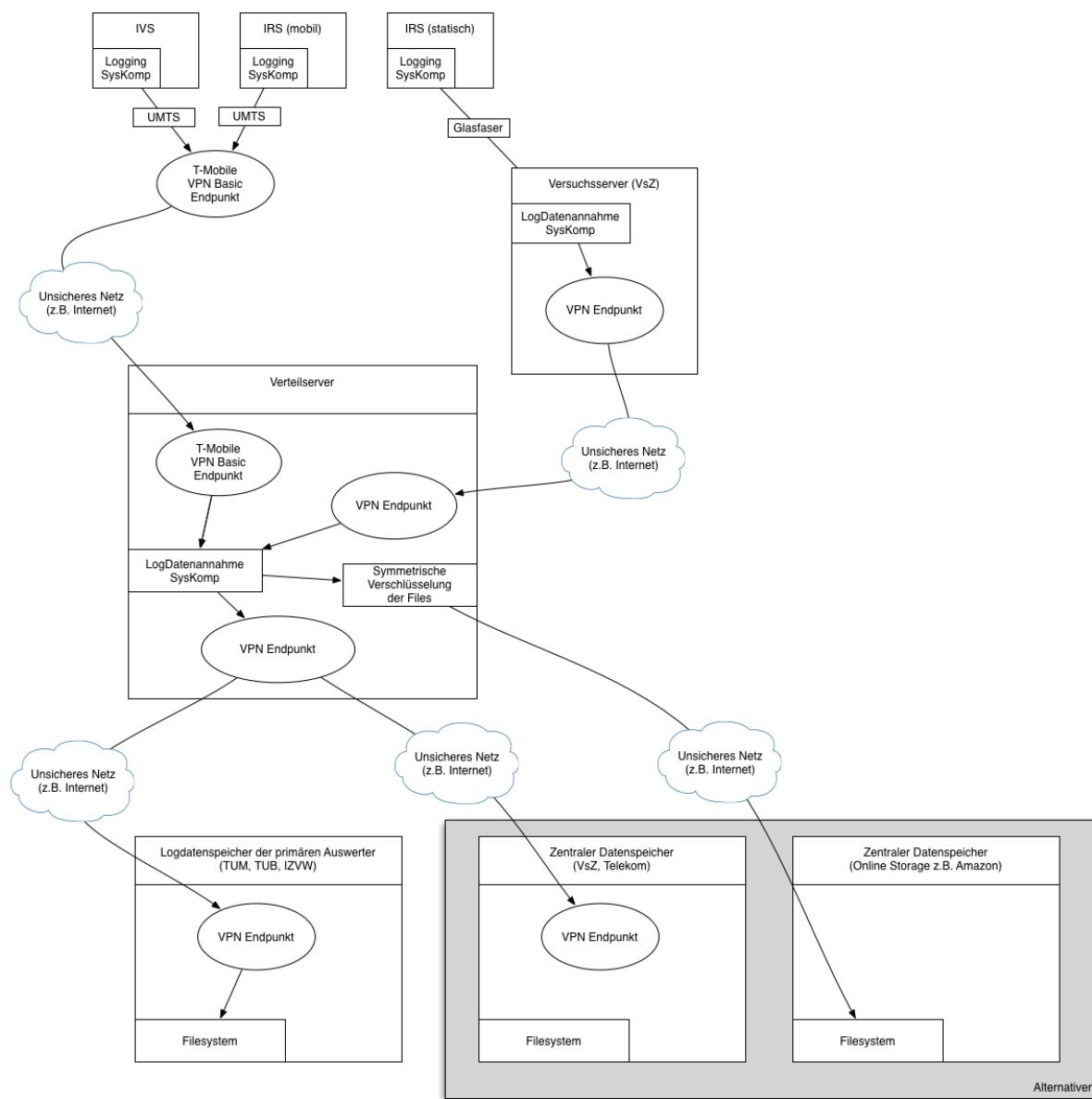


Abbildung 2-25: Übersicht über die Datenströme zwischen den einzelnen Stations- und den Logdatenspeichern unter Berücksichtigung der eingesetzten Verschlüsselungen

Der Verteilserver dient als Zwischenpuffer zwischen den einzelnen Stations (Abbildung 2-25 oben) und den Logdatenspeichern (unten).

Nach einem Versuch bzw. nach einem definierten Zeitraum werden durch die Stations die angefallenen Logdateien auf den Verteilserver übermittelt, welcher sie in erster Instanz entgegennimmt.

Der Verteilserver hält die aktuell für den Versuch (bzw. Zeitraum) für die Logfiles abonnierten Partner (z.B. TU München, TU Berlin, IZVW) vor, und verteilt die für diese Partner relevanten Logfiles dann direkt an den jeweiligen Logdatenspeicher. Hierbei werden 2 Dinge sichergestellt:

1. Die Logfiles gelangen nach Versuchende sofort für die an den Daten für die Analyse interessierten Partnern (primäre Auswerter).

2. Die Datenleitungen der VsZ (Anbindung und Details siehe D23.1) werden nicht durch einen einkommenden (durch die Stations) und danach durch einen ausgehenden (durch die Auswerter) Datenstrom belastet.

Um trotzdem zu gewährleisten, dass die Logfiles, die eine Hauptbasis der Analysen der Versuche in simTD sind, für alle Konsortialpartner an zentraler Stelle zur Verfügung stehen wird ein zentraler Datenspeicher immer mit den einkommenden Logfiles aller Stations beschickt. Dieser zentrale Datenspeicher kann im momentanen Planungsstand 3 Ausprägungen besitzen, welche alle durch die vorliegende Architektur sowohl übertragungs- als auch sicherheitstechnisch abgedeckt sind:

- Standort in der VsZ
- Standort bei einem infrastrukturtechnisch gut angebundenen Konsortialpartner
- Standort bei einem Drittanbieter (wie z.B. Amazon S3)

Momentan werden durch AP23 und AP24 alle 3 Möglichkeiten beleuchtet, und Ergebnisse dem Projekt nach Analyse zur Verfügung gestellt.

2.4.2.3 Livedaten

Das Monitoring soll gewährleisten dass der Versuchsregisseur in der Versuchszentrale die laufenden Versuche überwachen kann. Dazu zählt im Wesentlichen das Verifizieren der Parameter, die einen Versuch als valide ausgeführt auszeichnen. Das Monitoring, wie es hier beschrieben wird, ist nur für die ITS Vehicle Stations gültig, die ITS Roadside Stations besitzen ein eigenes Monitoring. Das Monitoring soll die relevanten Daten mit festem Zeitabstand an die Central ITS Station übertragen. Geplant ist zum aktuellen Zeitpunkt der Abstand von einer Sekunde, die technischen Gegebenheiten vor Ort können dies aber weiter einschränken.

Livedatenerfassung

Die Livedatenerfassung findet wie die Messdatenerfassung im Logging-Bundle statt. Für die Livedatenerfassung gilt es von TP4 aus eine Anzahl von Messpunkten zu definieren, die für die Überwachung als relevant gelten. Ein Messpunkt ist eine Stelle, die jederzeit einen Messwert liefern kann. Das kann ein Wert aus der VAPI sein, oder aber ein Wert der dem Logging über das dafür bereitgestellte Interface von einer Funktion oder Komponente übergeben wird. Auch für das Monitoring wird es ein Dictionary mit Monitoring-IDs geben und auch hier wird wie beim Logging der zu übergebende Wert ein Integer sein.



Abbildung 2-26: Das Interface für die Übergabe von Werten an das Monitoring.

Die drei folgenden Funktionen liefern dann mit dem Messwert als Eingabe die Ausgabe, die im Monitoring erfasst wird. Zur Erläuterung der Funktionen wurde das Beispiel Tankfüllung als Messpunkt gewählt (Sx=Schwellwert, E=Eingabe, A=Ausgabe).

1. Eine Binärfunktion liefert eine boolsche Ausgabe. Bei der Abfrage nach dem Messwert kann mit Hilfe eines zu definierenden Schwellwertes entschieden werden, ob es ein Problem gibt (true) oder nicht (false).
 - S1=10l
 - E=4l
 - A=True
 -
2. Eine Mappingfunktion liefert einen von vier Zuständen. Hier gibt es die vier Zustände „Good“, „Warn“, „Error“ und „Fatal“. Mit Hilfe von drei zu definierenden Schwellwerten kann nun der absolute Tankfüllungswert auf einen der Zustände gemapped werden.
 - S1=10l
 - S2=5l
 - S3=1l
 - E=4l
 - A=Error
3. Eine Direktfunktion leitet den Eingabewert direkt an die Ausgabe weiter, wenn ein Schwellwert überschritten ist. In dem Fall wird dann der Messwert direkt benutzt.
 - S1=10l
 - E=4l
 - A=4l

Jede dieser Funktionen muss für jeden Messpunkt implementiert werden, wenn sie genutzt werden soll. Die Messpunkte werden bis auf mögliche Ausnahmen (etwa freier Festplattenplatz) Logs sein, die im Rahmen der Messdatenerfassung im Logging-Bundle ankommen, woraus dann, konfigurierbar über das Logprofil, mit Hilfe einer der drei beschriebenen Funktionen ein Wert erstellt wird. In weiten Teilen ist das Monitoring also ein Subset des Loggings mit einer einzustellenden Unschärfe.

Livedatenübermittlung

Jedem Messpunkt wird außer der Funktion und den möglicherweise notwenigen Schwellwerten eine Relevanzstufe (1,2,3) zugeordnet, die aussagt wie relevant die Messung an dieser Stelle bezüglich der Versuchsdurchführung ist. Dies bietet die Möglichkeit dynamisch auf die zu übertragenden Werte einzugreifen und etwa nur die Messpunkte der Relevanzstufe 1 zu übertragen, wenn das Übertragen der Funktionsausgaben aller Messpunkte die Paketgröße überschritten hätte. Der Plan bezüglich der Größe ist, die im

Ethernet übliche MTU Größe von 1500Byte nicht zu überschreiten, damit die sekündlich gesendeten Nachrichten nicht in mehrere Pakete unterteilt werden müssen. Allein ein Paket pro Sekunde, wie es zum jetzigen Zeitpunkt geplant ist, lastet eine GPRS Verbindung mit optimalem Durchsatz (56 KBit/sec) zu ca. 22% aus, optimale Bedingungen sind aber sicher nicht immer zu erwarten. Grundsätzlich sieht das Übertragungsprotokoll, das in Deliverable D21.4 Spezifikation der Kommunikationsprotokolle genau definiert ist, neben der Übertragung eines Zeitstempels, der Quell-Id und der Position eine Menge an Stati vor. Von diesen werden jedoch nur die übertragen, die „nicht OK“ sind, also einen vordefinierten Schwellwert überschritten haben. Im Idealfall werden also nur Nachrichten mit wenig Informationen übertragen, sollten aber viele Werte zu übertragen sein und sollte damit das Paket drohen gesplittet zu werden, würden die Werte deren Messpunkte als weniger relevant definiert sind nicht übertragen.

Zurzeit ist eine sekündliche Übertragung geplant, allerdings kann es möglich sein dass aufgrund der technischen Gegebenheiten eine niedrigere Frequenz eingestellt werden muss.

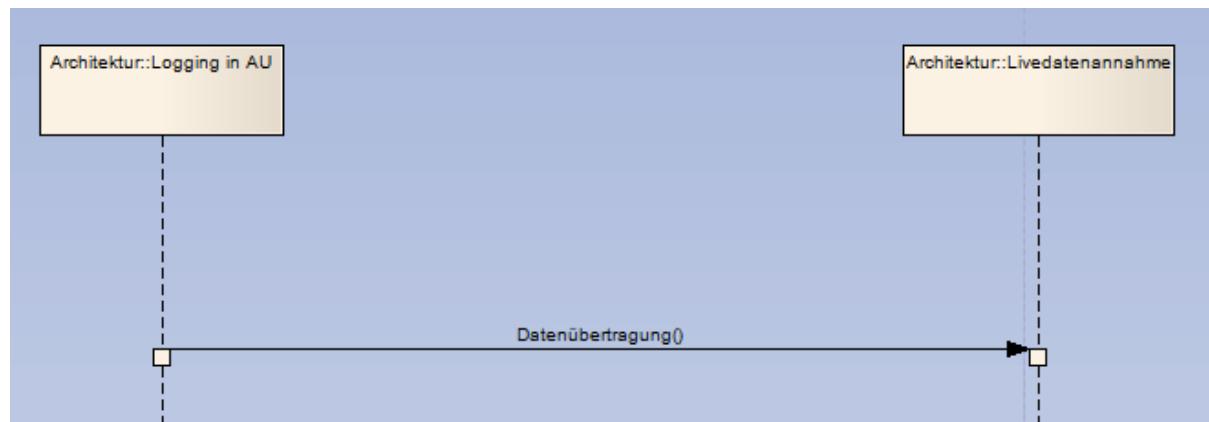


Abbildung 2-27: Sequenzdiagramm der Livedatenübermittlung

Die Übertragung selbst läuft per UDP ab. Wie im Sequenzdiagramm in Abbildung 2-27 modelliert wird ein aufwändiger Verbindungsauflauf damit eingespart, eine zuverlässige Übertragung ist allerdings nicht gewährleistet. Gesendet werden die Pakete an die Komponente Livedatenannahme, die als Server auf einem vordefinierten UDP Port die Pakete annimmt.

Livedatenverarbeitung

Die Livedatenannahme ist die Komponente, die die Livedatenpakete annimmt und dekodiert. Dort werden sie im Speicher vorgehalten um zügig an Interessenten weitergeleitet zu werden. Für diese Weiterleitung wird ein Interface zur Verfügung gestellt. Ein Interessent an diesen Daten ist der Leitstand, der die Daten grafisch aufbereitet darstellt. Die Versuchssteuerung ist ebenfalls an diesen Daten interessiert, um zeitnah überprüfen zu können ob ein Versuch korrekt durchgeführt worden ist. Die Daten werden dafür parallel in die Messdaten-DB eingespielt.

Für die Darstellung auf dem Leitstand werden neue Daten an das Interface des Leitstandes übergeben. Für den Fall, dass der Benutzer des Leitstandes weitere Details über ein bestimmtes Objekt erfahren möchte kann die Livedatenannahme entsprechend angefragt werden. Diese Interfaces werden in Abbildung 2-28 dargestellt.

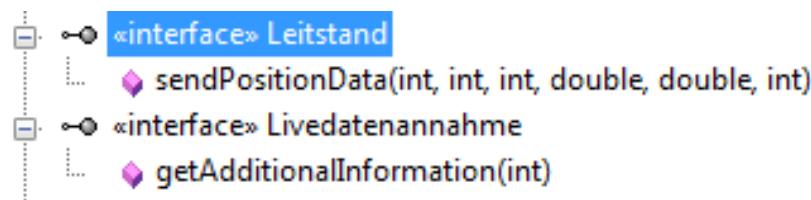


Abbildung 2-28: Interfaces für die Darstellung auf dem Leitstand

2.4.3 Versuchsunterstützung

Im Vorfeld der Versuchsdurchführung werden für die einzelnen Funktionen die entsprechenden Anwendungs-, Test- und Versuchsfälle definiert. Diese entstehen in mehreren Iterationen in den Arbeitspaketen AP11, AP13 und AP41, sowie anhand der Validierungsziele. Auf Basis dieser Anwendungsfälle entsteht eine Reihe an Versuchsszenarien, die einen bestimmten Anwendungsfall abdecken.

Für diese Versuchsszenarien werden durch den Versuchsplaner die entsprechenden Drehbücher mit Fahreranweisungen angelegt und die Versuchsdurchführung wird entsprechend der Verfügbarkeit von Fahrer und Fahrzeug eingeplant. Es ist vorgesehen, dass geplante Szenarien mehrfach durchgeführt werden.

Für die Versuchsdurchführung sind im Wesentlichen die folgenden Schritte notwendig:

- Eingeben der Flottendaten
 - Informationen über die Fahrer
 - Informationen über die Fahrzeuge
 - Erfassen der täglichen Verfügbarkeit von Fahrer und Fahrzeug
- Eingeben der Versuchsszenarien
 - Eingeben der einzelnen Routen
 - Eingeben der benötigten Fahrer und Fahrzeuge, sowie die Angabe von Anforderungen an beide

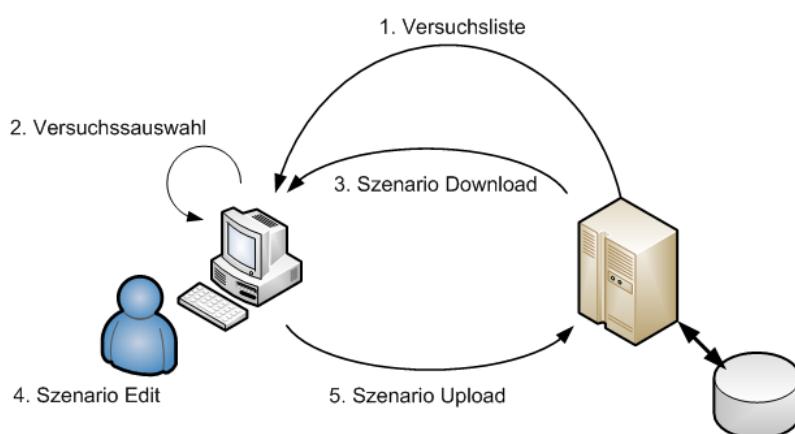


Abbildung 2-29: Beispiel zum Bearbeiten eines Versuchsszenarios

- Planen der Versuchsszenarien
 - Einplanen der verfügbaren Fahrer und Fahrzeuge für die Versuchsdurchführung
- Versuchsdurchführung
 - Auswählen des durchzuführenden Versuchsszenarios
 - Prüfen aller Vorbedingungen (Fahrer und Fahrzeuge verfügbar, etc.)
 - Versuchsdurchführung initiieren, starten und bei Bedarf stoppen
 - Überprüfen, ob die Versuchsdurchführung als „erfolgreich“ gewertet werden kann

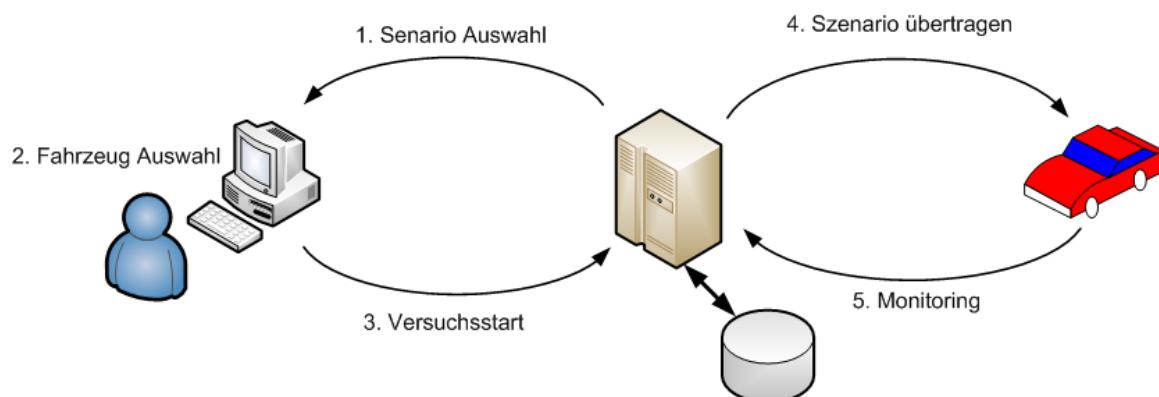


Abbildung 2-30: Beispiel zur Versuchsdurchführung

- Fahrerbefragung
 - Befragung während der Versuchsdurchführung (Gezielt über die zu testende Funktion)
 - Befragen nach Abschluss der Versuchsdurchführung

Die einzelnen Punkte lassen sich dabei gliedern in die folgenden vier Funktionen:

- Flottenmanagement
- Testablaufplanung
- Versuchssteuerung
- Fahrerbefragung

Diese vier Funktionen erstrecken sich dabei über die Vehicle AU, die Versuchszentrale und einem lokalen Arbeitsplatz für den Versuchsplaner/Versuchsregisseur.

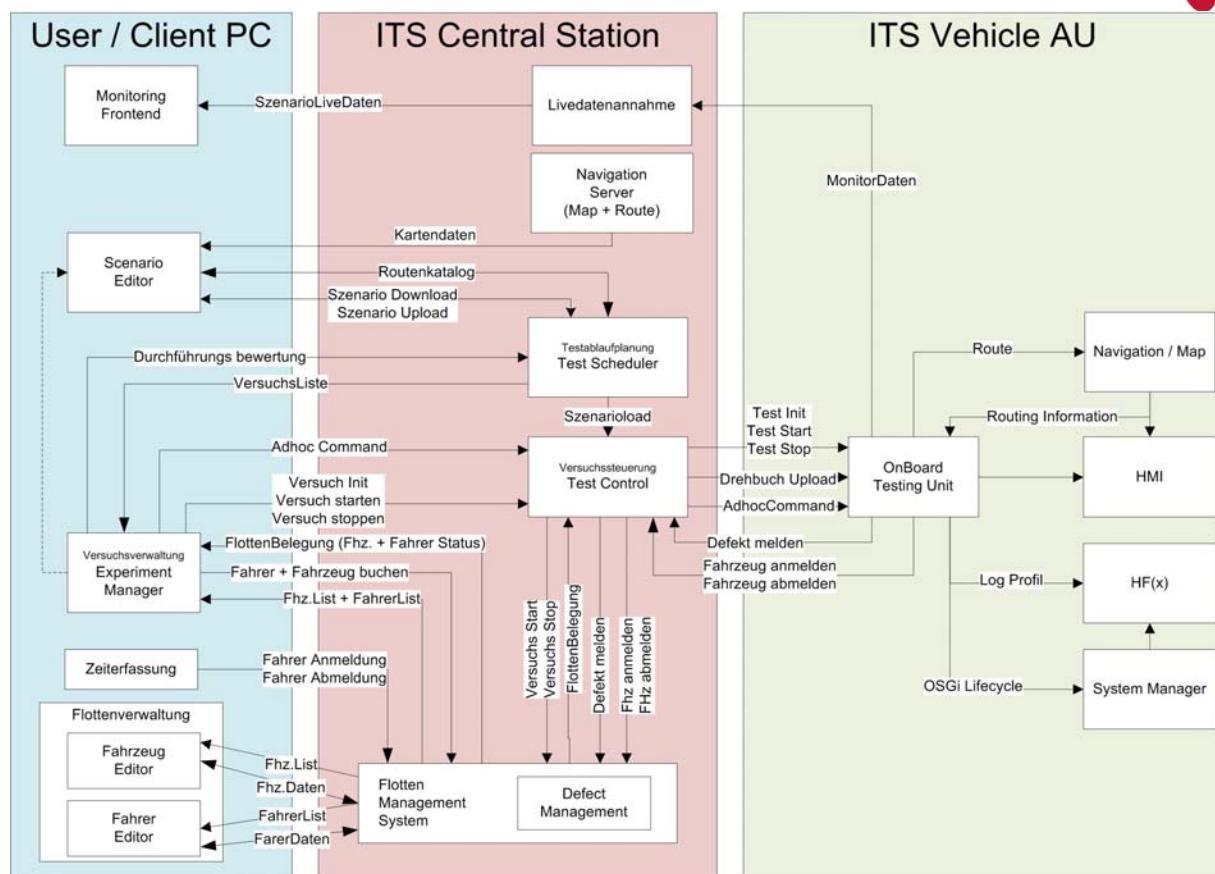


Abbildung 2-31: Komponenten Übersicht für die Versuchsunterstützung

2.4.3.1 Flottenmanagement

Für die Planung und Durchführung einzelner Versuchsszenarien erhält der Versuchsregisseur einen Überblick über die aktuell verfügbaren Fahrer und Fahrzeuge. Zu den einzelnen Fahrern und Fahrzeuge sind darüber hinaus Detailinformationen abrufbar.

Im Vorfeld benötigte Informationen:

- Welche Fahrer & Fahrzeuge gibt es?
- Verfügbarkeit der Fahrer und Fahrzeuge (bereits verplant / frei / defekt)

Am Versuchstag:

- Position der Fahrzeuge
- Verfügbarkeit der Fahrzeuge („online?“)

Um die skizzierten Funktionen und Werkzeuge (siehe Testablaufplanung) zu realisieren, sind die folgenden Funktionalitäten zu implementieren und in drei Interfaces anzubieten. Die Interfaces sind in Abbildung 2-32 dargestellt.

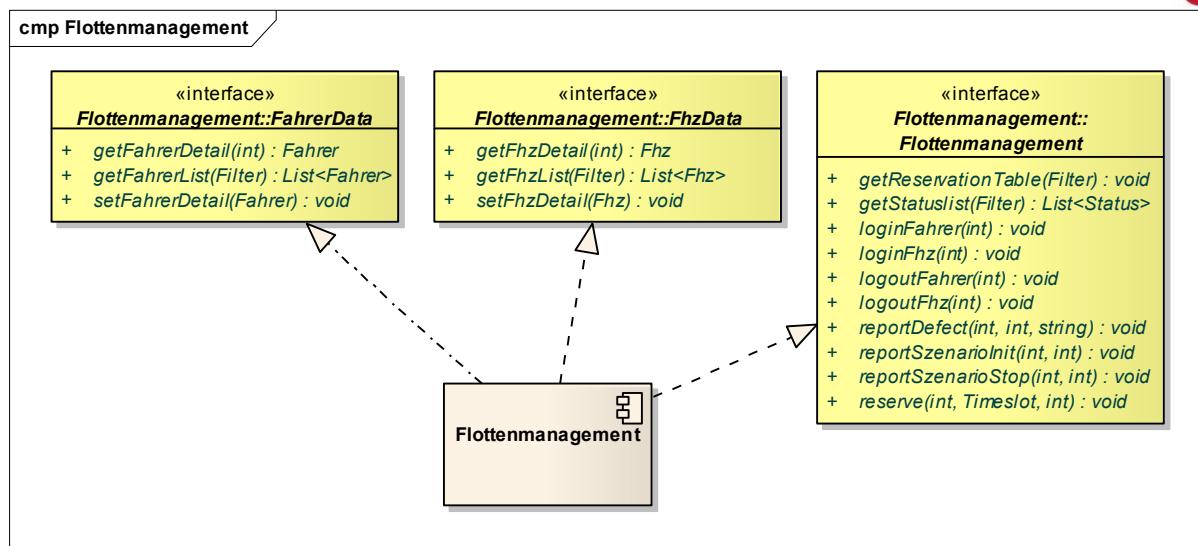


Abbildung 2-32: Interfaces des Flottenmanagements

Der Zugriff auf diese drei Interfaces erfolgt dabei von verschiedenen Komponenten. Zum einen direkt von der Versuchszentrale, zum anderen aber auch von diverseren Clients. Neben der lokalen Datenpflege von Fahrern und Fahrzeugen meldet noch die Arbeitszeiterfassung die Verfügbarkeit der einzelnen Fahrer für Versuchsszenarien. Für die Einplanung in die einzelnen Versuchsdurchführungen benötigt zusätzlich auch die Testablaufplanung einen Zugriff.

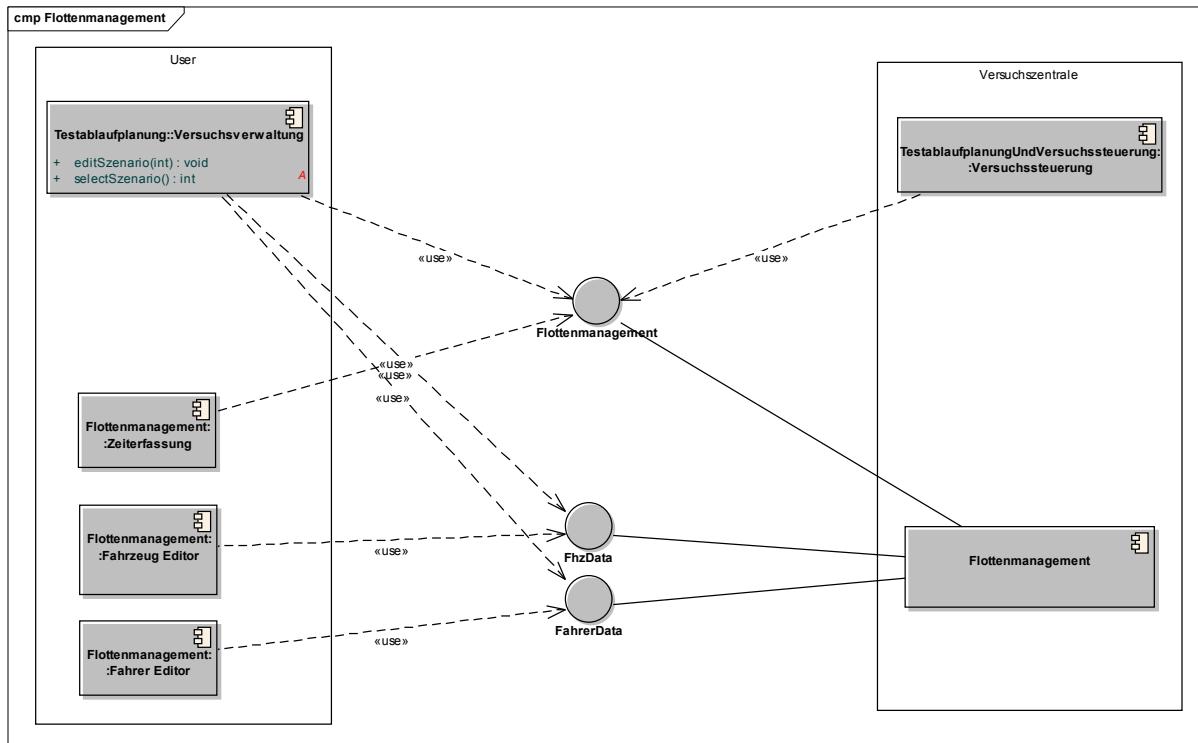


Abbildung 2-33: Zugriff auf die Schnittstellen des Flottenmanagements

Fahrzeugverwaltung

Ermöglicht die Eingabe und Verwaltung der Daten der Fahrzeugflotte in einer Datenbank (ID, CCU Version, Software Version, Reparaturstatus, etc.) über die Methode `setFhzDetail()` im Interface `FhzData`.

Fahrerverwaltung

Ermöglicht die Eingabe und Verwaltung der Daten aller Fahrer in einer Datenbank (ID, Fahrerinformationen, Fahrerlevel, etc.) über die Methode `setFahrerDetail()` im Interface `FahrerData`.

Fahrzeugabfrage

Die Schnittstelle `FhzData` stellt die Methoden `getFhzList()` und `getFhzDetail()` zur Verfügung, mit der andere Anwendungen eine Liste aller Fahrzeuge oder ein einzelnes Fahrzeug und die entsprechenden Detaildaten abfragen können.

Fahrerabfrage

Die Schnittstelle `FahrerData` stellt die Methoden `getFahrerList()` und `getFahrerDetail()` zur Verfügung, mit der andere Anwendungen eine Liste aller Fahrer oder einen einzelnen Fahrer und die entsprechenden Detaildaten abfragen können.

Fahrzeug an-/abmeldung

Mit dem Einsteigen in das Fahrzeug erfolgt eine Anmeldung des Fahrzeugs in der Versuchszentrale. Damit ist das Fahrzeug für Versuche verfügbar. Die Anmeldung wird von der Versuchszentrale an das Flottenmanagement weitergeleitet, wo vermerkt wird, dass das Fahrzeug für Versuche bereit ist. Entsprechend erfolgt die Abmeldung des Fahrzeugs beim Verlassen.

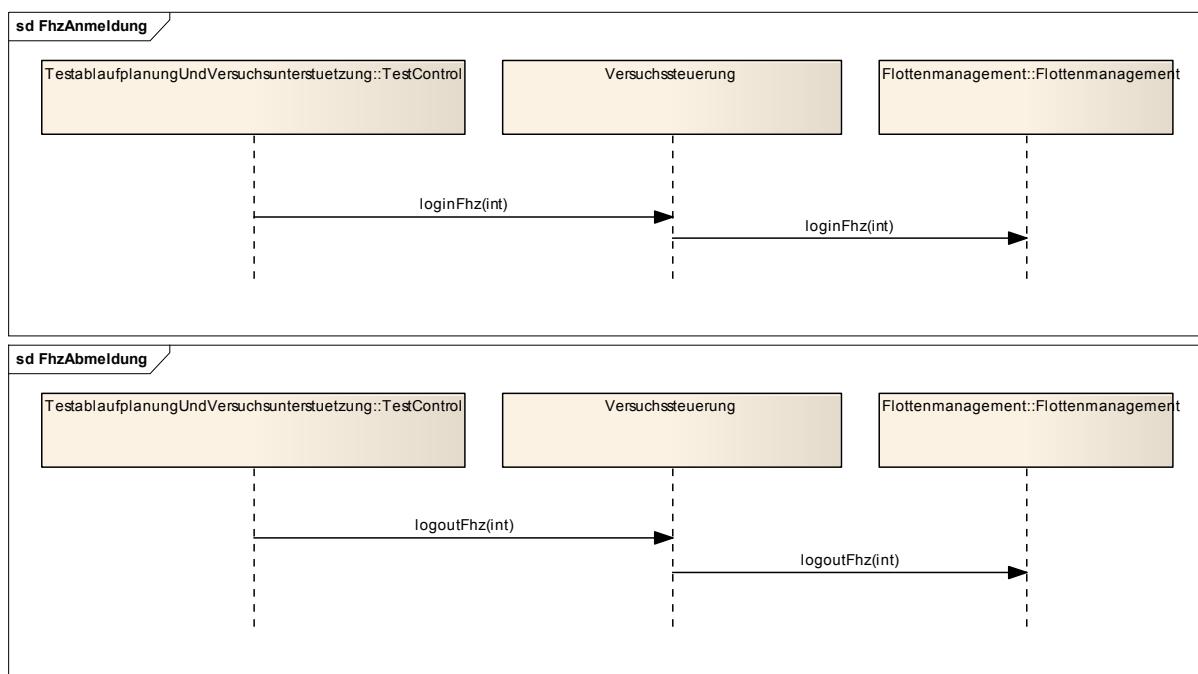


Abbildung 2-34: An- und Abmeldung von Fahrzeugen

Fahrer An-/Abmeldung

Zu Beginn/Ende seiner Arbeitszeit erfolgt eine An-/Abmeldung des Fahrers (dies kann je nach Ausbau des Versuchsceters manuell oder z.B. über Chipkarten Zeiterfassungsterminal erfolgen) in der Versuchszentrale. Dadurch kann der Versuchsregisseur rechtzeitig erkennen, welche Fahrer am Versuchstag tatsächlich verfügbar sind. Es muss eine Schnittstelle im Flottenmanagement zur Verfügung gestellt werden, über die die An-/Abmeldung an das Flottenmanagement weitergegeben werden kann.

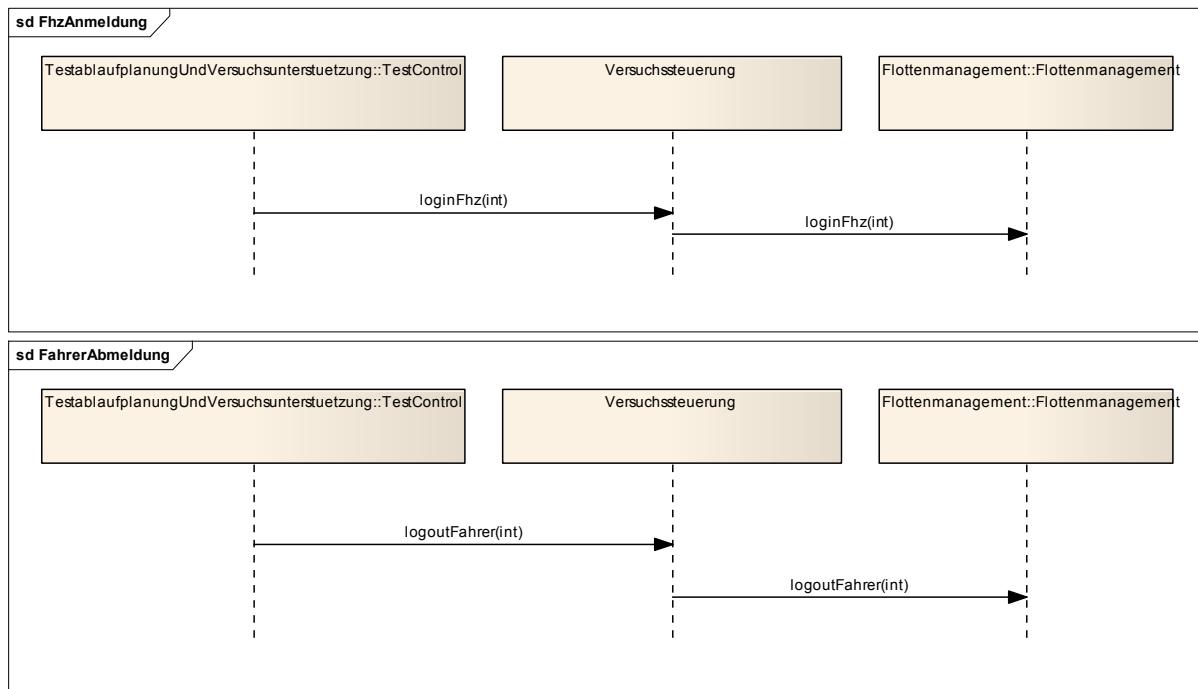


Abbildung 2-35: An- und Abmeldung von Fahrern

Flottenabfrage

Andere Anwendungen können über die Schnittstelle *Flottenmanagement* die geplante Belegung von Fahrzeugen und Fahrern sowie deren aktuellen Status über die Methoden *getReservatioTable()* und *getStatusList()* abrufen.

Fahrzeugreservierung

Über die Methode *reserve()* in der Schnittstelle *Flottenmanagement* ist es möglich, neue Reservierungen für eine Versuchsdurchführung vorzunehmen. Reserviert wird jeweils ein Fahrzeug zu einem Zeitpunkt mit einem Fahrer.

Defekt melden

Dem Flottenmanagement können über die Schnittstelle *Flottenmanagement* mögliche Defekte an einem Fahrzeug von der Versuchszentrale gemeldet werden. Hierfür wird die Methode *reportDefect()* implementiert.

Versuchsstatus melden

Die Versuchszentrale informiert das Flottenmanagement über die Methoden `reportSzenarioInit()` und `reportSzenarioStop()` darüber, ob ein Fahrzeug gerade an einem Versuch teilnimmt.

Die für die Testablaufplanung sind das An/Abmelden von Fahrern und Fahrzeugen hilfreich, da über diese Funktionen auswähle automatisch erkannt werden. Die Funktionalität der Ablaufplanung ist aber auch ohne die Funktionen zum an- und abmelden gegeben. Dann können lediglich, kurzfristige Ausfälle bei Fahrzeugen und Fahrer nicht so schnell berücksichtigt werden.

2.4.3.2 Testablaufplanung

Die Versuchsunterstützung soll die Testablaufplanung dadurch unterstützen, dass sie Werkzeuge für die Versuchsverwaltung und die Versuchsszenarioplanning zur Verfügung stellt.

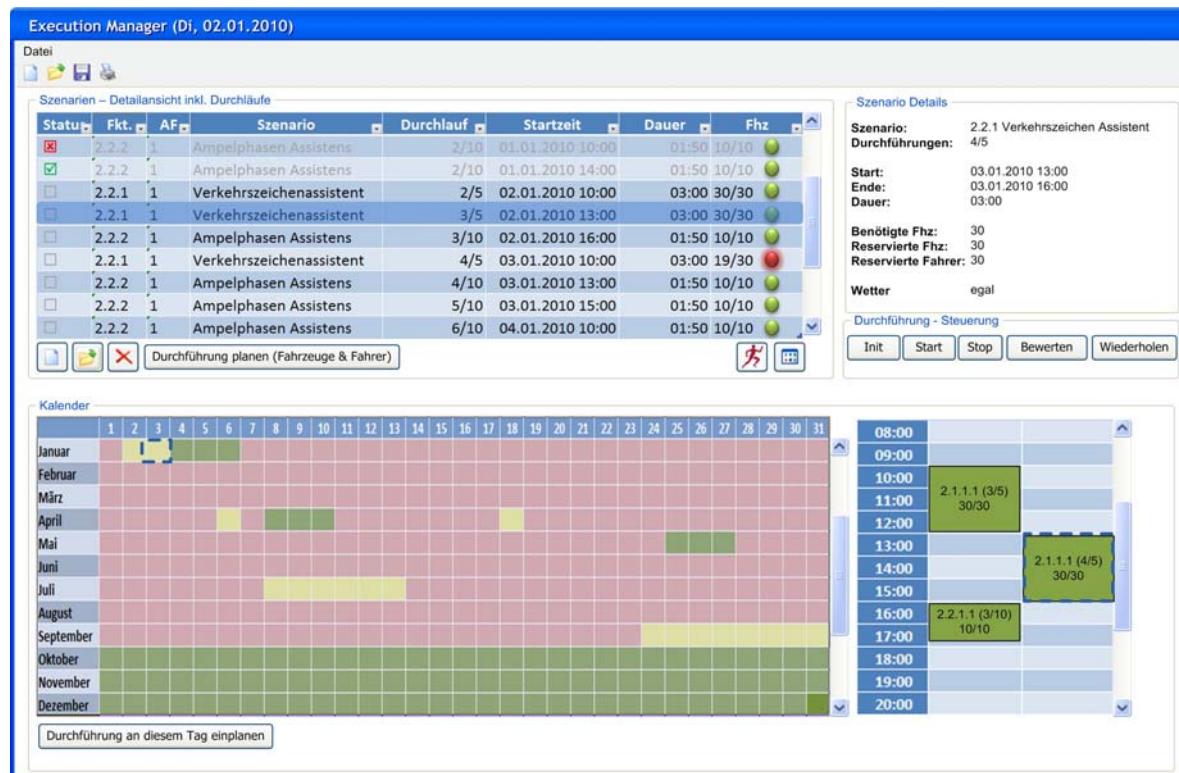
Die Versuchsszenarien werden in der Datenbank in der Versuchszentrale gespeichert und vom Versuchsplaner/Versuchsregisseur an einem lokalen Arbeitsplatz bearbeitet.

Die im Folgenden gezeigten Entwürfe der einzelnen Werkzeuge, dienen der beispielhaften Visualisierung und zeigen noch nicht den endgültigen Funktionsumfang.

Werkzeuge und Anwendungen

Für die Verwaltung und Durchführung der einzelnen Szenarien sind eine Reihe an Anwendungen geplant die hier kurz beschrieben werden sollen. Anhand dieser Anwendungen ergibt sich auch der Bedarf für die im Anschluss folgenden Teifunktionen.

Werkzeug – Execution Manager



The screenshot shows the Execution Manager interface for scenario planning. It includes:

- Szenarien - Detaillansicht inkl. Durchläufe:** A table listing scenarios with details like ID, name, duration, start time, and required Fhz.
- Szenario Details:** A panel showing the selected scenario (2.2.1 Verkehrszeichen Assistent) with its execution details: Start: 03.01.2010 13:00, End: 03.01.2010 16:00, Duration: 03:00, Required Fhz: 30, Reserved Fhz: 30, Reserved Driver: 30, Weather: egal.
- Durchführung - Steuerung:** Buttons for Init, Start, Stop, Bewertung, and Wiederholen.
- Kalender:** A monthly calendar showing scheduled events across months from January to December.
- Timeline:** A vertical timeline from 08:00 to 20:00 showing scheduled tasks for different drivers and their requirements.
- Buttons at the bottom:** Durchführung an diesem Tag einplanen and Durchführung planen (Fahrzeuge & Fahrer).

Abbildung 2-36: Beispiel für Versuchsverwaltung

Mit Hilfe des Execution Manager können alle Szenarien aufgelistet und die Durchführungen geplant und bewertet werden. Der Execution Manager greift dabei auf Schnittstellen von Szenario Editor, TestScheduler, TestControl und Flottenmanagement zu (siehe Abbildung 2-37).

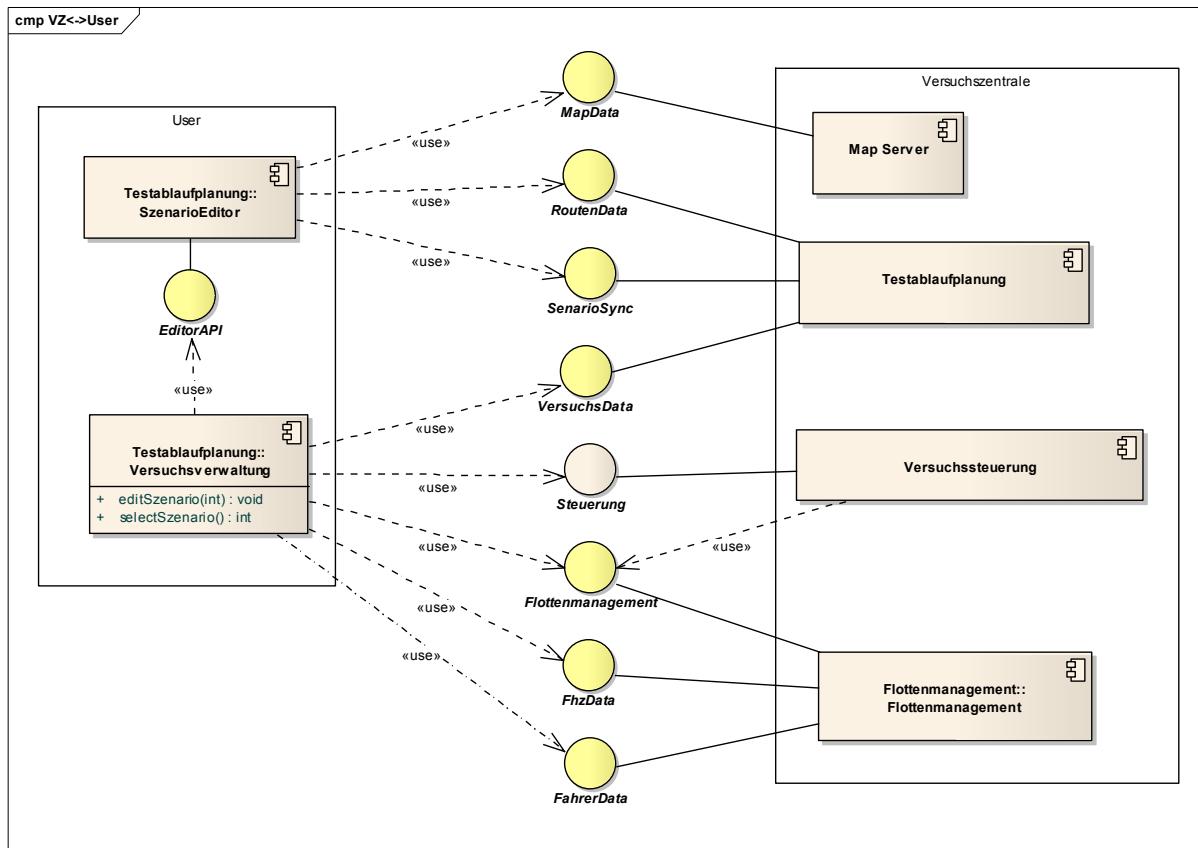


Abbildung 2-37: Zugriff auf die Schnittstellen der Versuchsplanung

Für die Planung hilft eine kalendarische Übersicht, die Auskunft darüber gibt, wie viele Fahrzeuge noch verfügbar sind. Dabei soll es später möglich sein, verschiedene Fahrzeugtypen (z.B. Motorisierung, PKW / LKW, Einsatzfahrzeug, etc.) zu unterscheiden. In der Versuchsplanung im Vorfeld wird dabei lediglich eine bestimmter Fahrzeugtyp für dieses Szenario reserviert. Erst an dem Tag der Versuchsdurchführung findet dann eine gezielte Auswahl des tatsächlichen Fahrzeugs und Fahrers zu den geplanten Strecken statt, da Fahrer krank oder Fahrzeuge defekt sein können.

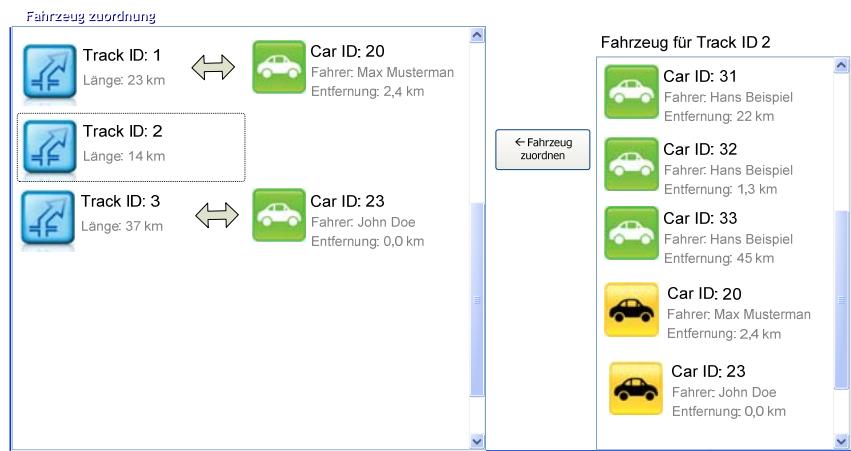


Abbildung 2-38: Beispiel für endgültige Fahrzeugzuordnung

Das ausgewählte Szenario kann vom Versuchsteam gestartet werden und wird im Anschluss als erfolgreich oder zu wiederholen eingestuft. Beim Starten eines Szenarios werden den beteiligten Fahrern ihre individuellen Drehbücher übermittelt.

Werkzeug – Szenario Editor

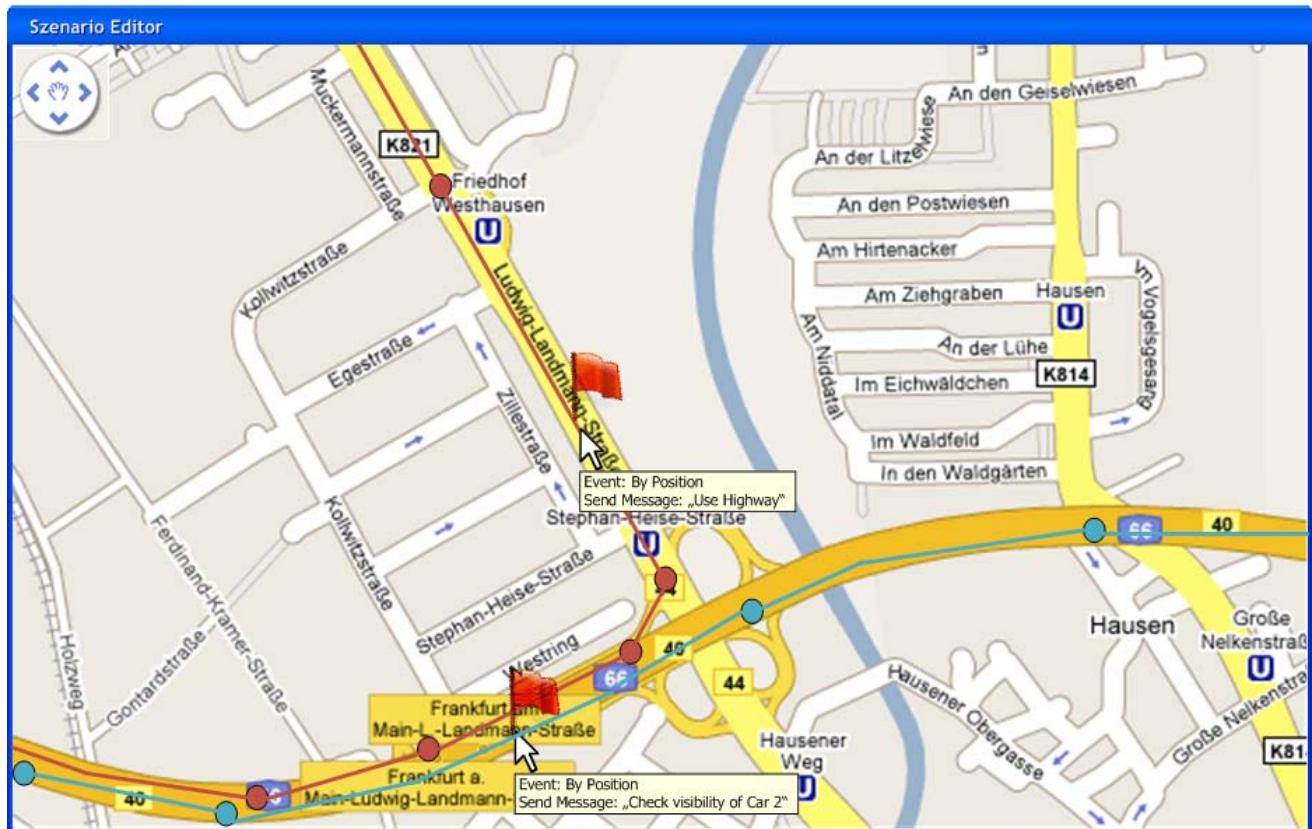


Abbildung 2-39: Beispiel für den Szenario Editor

Mit dem Szenario Editor können die verschiedenen Szenarien angelegt und bearbeitet werden. Der Szenario Editor bietet mindestens das folgende Interface.



Das Szenario umfasst dabei eine Route für jedes Fahrzeug (Schleifen sind dabei als Route zulässig), sowie die Planung von Ereignissen. Die Ereignisse beziehen sich auf ein bestimmtes Fahrzeug und können sowohl positions- als auch zeitgesteuert auftreten.

Als Ereignis sind dabei folgende Aktionen denkbar:

- HMI-Mitteilung an Fahrer (z.B. für die Navigation)
- Ändern des Loggingprofils

Die Navigation der Szenariorouten erfolgt nicht über das Navigationssystem, sondern über manuelle (per Ereignis) geplante HMI-Mitteilungen. Diese Maßnahme ist notwendig, da eine Reihe an Funktionen die Navigation verwenden und beeinflussen. Da genau diese Funktionen getestet werden sollen, muss auch die Reaktion des Navigationssystems mit getestet werden.

Werkzeug – Versuchsmonitor

Während der Versuchsdurchführung erhält das Versuchsteam Informationen über den Status der Fahrzeuge und deren Position. Er kann mit dieser Hilfe den geplanten mit dem realen Verlauf vergleichen und spontan Meldungen an einzelnen Fahrer oder an vordefinierte Gruppen senden.

Anwendungsfälle und Teilfunktionen

Für die Realisierung der beiden Werkzeuge sind die folgenden Anwendungsfälle relevant. Die Komponenten Testablaufplanung, Versuchssteuerung, Flottenmanagement und Map Server stellen hierfür Interfaces bereit, die in Abbildung 2-32, Abbildung 2-40, Abbildung 2-41 und Abbildung 2-42 dargestellt sind.

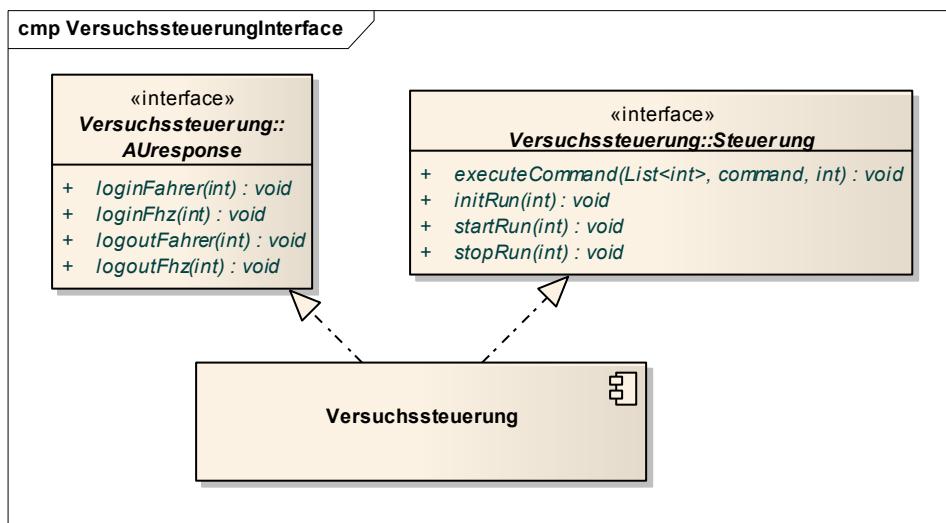


Abbildung 2-40: Interfaces der Versuchssteuerung

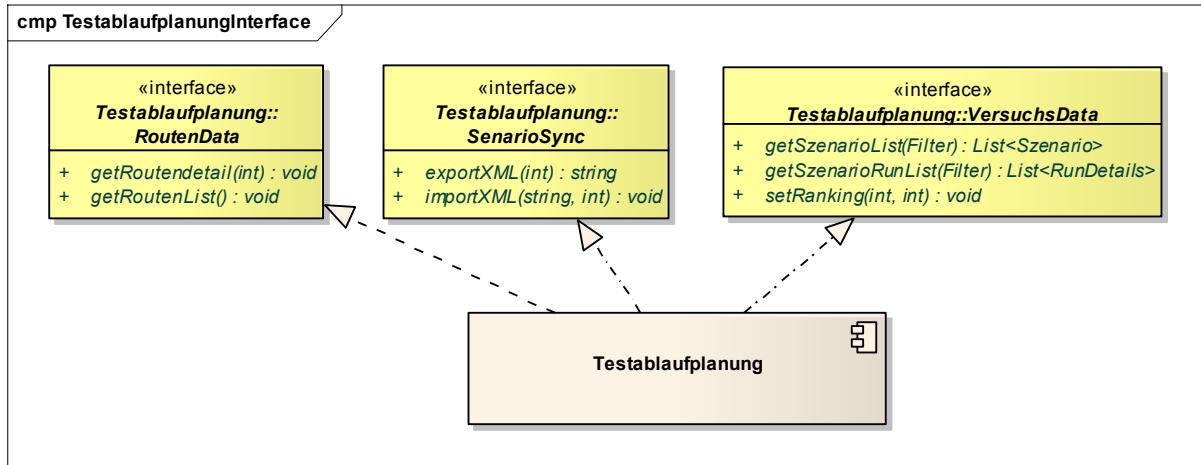


Abbildung 2-41: Interfaces der Testablaufplanung

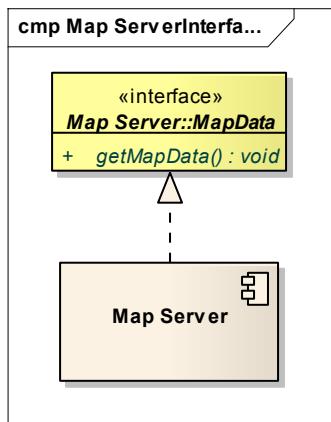


Abbildung 2-42: Interface des Map Servers

Versuchsmanagement

In der Datenbank werden alle Versuchsszenarien gehalten. Da einzelne Szenarien mehrfach durchgeführt werden können, muss zusätzlich die Versuchsspezifikation erfasst werden. Diese beschreibt, welche Szenarien wie oft durchgeführt werden müssen.

Szenario Auflistung

Über eine Methoden `getSzenarioList()` und `getSzenarioRunList()` der Schnittstelle `VersuchsData` können alle in der DB hinterlegten Versuchsszenarien abgerufen werden.

Szenario Sync

Über eine Schnittstelle `SzenarioSync` ist der Inhalt eines kompletten Szenarios abrufbar und kann auch wieder eingespielt werden (z.B. um ein Versuchsszenario im Editor zu bearbeiten). Dies geschieht über die Methoden `exportXML()` bzw. `importXML()`. Dabei kann das Szenario für andere geblockt werden (Abruf nur lesend / Abruf lesend und schreibend).

Szenario Edit

Mit dem Editor ist ein lokales Bearbeiten eines Versuchsszenarios möglich. Dies umfasst z.B. Planen von Routen, die über den `RoutenData` per `getRoutenList()` und `getRoutendetail()` zur Verfügung gestellt werden, Events und Anweisungen.

Auswahl und Reservieren von Fahrzeugen

Für die Durchführung eines Versuchsszenarios muss jeder Strecke ein Fahrzeug zugeordnet werden. Dies geschieht während des Editierens eines Szenarios über die Funktionalität des Flottenmanagements.

Versuchs Lifecycle Management

Für die Durchführung, muss das Versuchsszenario auf den Fahrzeugen initiiert, gestartet und gestoppt werden. Dafür werden sowohl von Versuchssteuerungsseite als auch von Fahrzeugseite Interfaces zur Verfügung gestellt, die Inititieren, Starten, Stoppen und Ausführen von Kommandos ermöglichen.

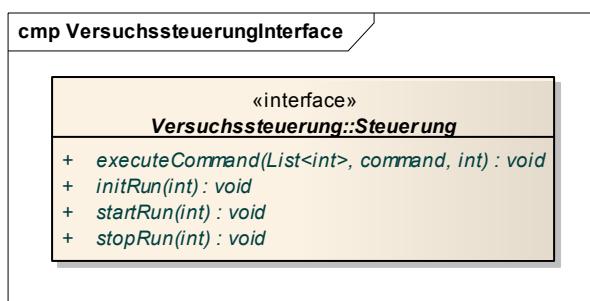


Abbildung 2-43: LifecycleInterface in der AU

Versuchsüberwachung

Dem Versuchsteam sollen Informationen zum aktuell durchgeführten Versuch zur Verfügung gestellt werden. (Ist- und Soll-Position der Fahrzeuge, evtl. Abweichung verdeutlichen).

Bewertung der Durchführung

Nach Abschluss des Versuchsszenarios muss der Versuchsregisseur die Durchführung bewerten (erfolgreich / muss erneut durchgeführt werden).

Dabei wird unter anderem beachtet:

- Sind alle Logfiles der Fahrzeuge vorhanden?
- Haben die Fahrzeuge die geplanten Routen eingehalten?
- Einhalten von vordefinierten Geschwindigkeiten auf einem bestimmten Streckenabschnitt.
- Manuelle Bewertung ob bestimmte vorher definierte Situationen auch eingetroffen sind.

Das Bewerten erfolgt über die Methode `setRanking()` im Interface `ExperimentData` der Testablaufplanung. Die Bewertung erfolgt dabei manuell durch den Versuchsregisseur. Ihm wird durch die Werkzeuge eine Reihe an Daten zur Verfügung gestellt, die ihm bei der Entscheidung helfen sollen.

Beispiel: Für verschiedene Szenarien reicht es aus, wenn das Fahrzeug die geplante Route einhält. Bei anderen kann es aber wichtig sein, dass die geplanten Punkte auch zu einer vorgegebenen Zeit erreicht werden (z.B. Kreuzungsfälle von Fahrzeugen). Bei diesen kann nur eine manuelle Prüfung zeigen, ob die zeitliche Abweichung trotzdem zu dem gewünschten Ergebnis geführt hat.

Drehbuch drucken

Im Vorfeld der Versuchsdurchführung kann das Szenario aus den in der Datenbank hinterlegten Daten ein Drehbuch pro Fahrzeug generieren und ausdrucken. Die eigentliche Versuchssteuerung soll automatisiert erfolgen. Die gedruckte Version dient als zusätzliche Information für die Fahrer

Gruppierung

Für ein Szenario können Gruppen angelegt werden, in denen die einzelnen Fahrzeuge organisiert werden können.

Routenkatalog

Die in den Szenarien verwendeten Routen werden gesondert abgelegt. Dadurch können sie in anderen Szenarien jederzeit wiederverwendet werden.

Ablauf der Versuchsplanung

Der geplante Ablauf der Versuchsplanung ist in Abbildung 2-44 dargestellt. Eine Liste der Szenarien wird aus der Datenbank der Versuchsablaufplanung geladen. Der Versuchsplaner wählt ein Szenario aus, welches ihm über `loadSzenario()` und `exportXML()` zur Verfügung gestellt wird und das er dann bearbeiten kann. Das Speichern des Szenarios erfolgt über `saveSzenario()` und `importXML()`.

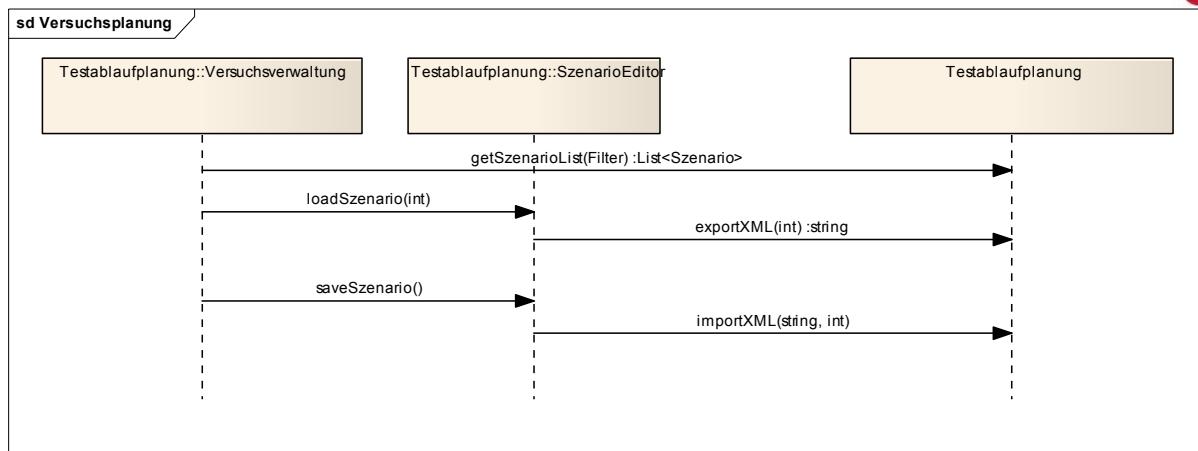


Abbildung 2-44: Ablauf der Versuchsplanung

2.4.3.3 Versuchssteuerung

Für die Versuchssteuerung sind die folgenden Anwendungsfälle geplant.

Versuchs Lifecycle Management

Auf dem Fahrzeug existiert eine Schnittstelle, über die Versuche initiiert, gestartet und gestoppt werden können. Außerdem kann die Vehicle AU spontan Anweisungen empfangen und ausführen (z.B. HMI Mitteilung ausgeben).

Die Komponente informiert auf der Vehicle AU andere Komponenten über ein Event, wenn sich der Status ändert.

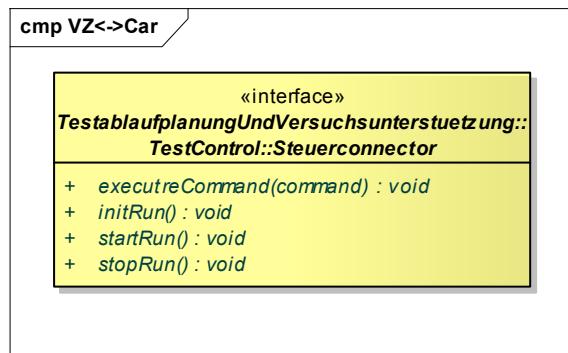


Abbildung 2-45: Interface der AU Steuerung

Drehbuch Download

Aufgrund einer Versuchsinitialisierung wird ein Drehbuchdownload ausgelöst. Das Fahrzeug fordert daraufhin von der Versuchszentrale das aktuelle Drehbuch an. Die Dateiübertragung findet über ein noch zu spezifizierendes Protokoll statt.

Ad-hoc-Anweisungen

Über diese Schnittstelle erhält das Fahrzeug Ad-hoc-Mitteilungen vom Versuchsteam. Mit den Ad-hoc-Anweisungen können sowohl Teile des Drehbuchs überschrieben werden, also

auch direkte Versuchsanweisungen an den Fahrer erfolgen. Die Übertragung erfolgt dabei über ein Protokoll, das den Erhalt der Nachricht bestätigt.

Versuchsüberwachung

Rückmeldung von Überwachungsinformationen an das Versuchsteam.

Versuchsanweisungen

In dem Drehbuch sind verschiedene Anweisungen für den Fahrer enthalten (z.B. Routenanweisungen). Diese Informationen müssen zu dem geplanten Zeitpunkt an das HMI übergeben werden.

Defekt melden

Der Fahrer soll direkt aus dem Fahrzeug heraus Defekte und Probleme melden können. Dies geschieht während der normalen Kommunikation zwischen Fahrzeug und Versuchszentrale über Kommandos im Protokoll.

Navigation zum Startpunkt

Für die Versuchsinitiierung muss das Fahrzeug zum Ausgangspunkt des Versuchsszenarios navigiert werden. Dafür wird das interne Navigationssystem verwendet und über die vorhandenen Schnittstellen mit Navigationszielen gefüllt.

Hauptfunktion steuern

In den Drehbüchern wird definiert, welche Funktionen aktiv sein müssen. Zusätzlich kann festgelegt werden, dass bestimmte Hauptfunktionen bewusst abgeschaltet sein sollen. Diese Hauptfunktionen werden dann über die OSGi-Schnittstellen gestartet bzw. gestoppt.

Ablauf der Versuchsdurchführung

Die Durchführung eines Versuchs ist in Abbildung 2-46 dargestellt. Aus der von `getSzenarioList()` gelieferten Liste von Szenarien wird eines ausgewählt und per `initRun()` in der Versuchszentrale initiiert. Die Versuchszentrale initiiert das Szenario auf den zugehörigen Fahrzeugen per `initRun()`. Der Start des Versuchs geschieht über `startRun()` in der Versuchszentrale und allen beteiligten Fahrzeugen. Während eines Versuchs können Ad-hoc-Anweisungen an einzelne Fahrzeuge oder Gruppen von Fahrzeugen gegeben werden. Dafür wird der Methode `executeCommand()` beim Aufruf in der Versuchszentrale neben dem Kommando eine Liste von Fahrzeugen übergeben, die dort aufgelöst wird, bevor das Kommando an die einzelnen Fahrzeuge weiterleitet wird. Nachdem das Beenden der Versuchsdurchführung per `stopRun()` über die Versuchszentrale an die Fahrzeuge propagiert wurde, kann noch eine Bewertung des Versuchs per `setRanking()` durchgeführt werden. Der letzte Schritt ist auch später noch möglich, da direkt nach Beendigung des Versuchs eventuell noch nicht alle Daten vorliegen, um den Versuch bewerten zu können.

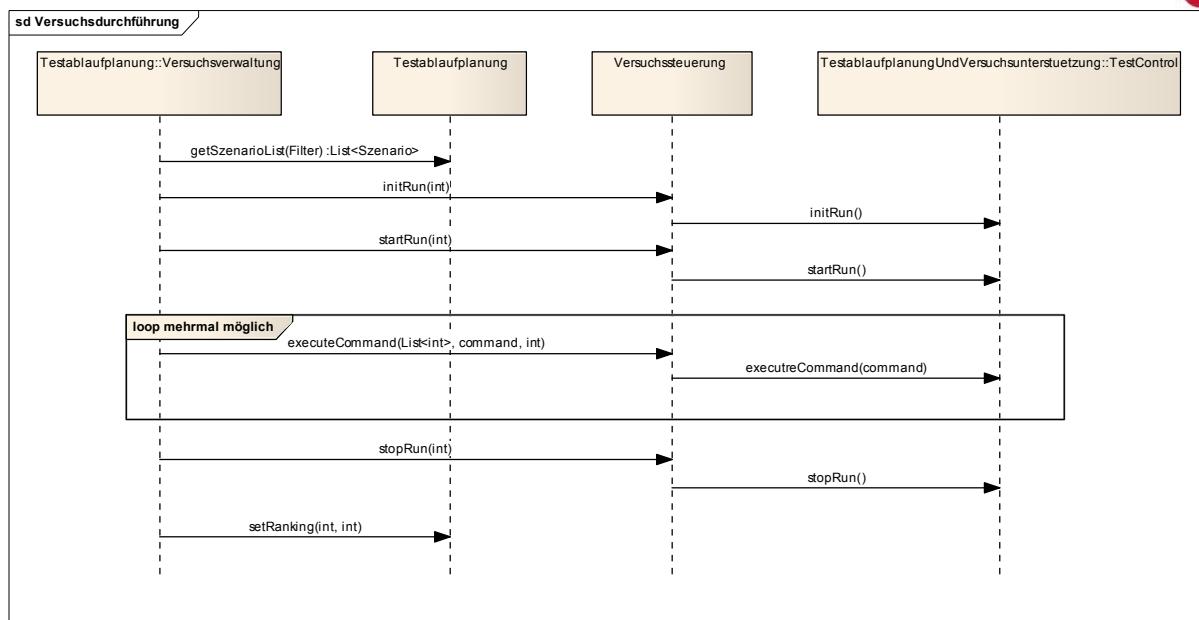


Abbildung 2-46: Versuchsdurchführung

2.4.3.4 Fahrerbefragung

Die Fahrerbefragung ist Teil der Versuchsunterstützung, wird jedoch im Rahmen einer Funktion modelliert, da sie keine Querabhängigkeiten zu anderen Komponenten des Systems hat, sondern lediglich die Funkschnittstellen verwendet.

2.5 Security-Konzept

Der unautorisierte Zugriff auf Daten und das Einschleusen gefälschten Informationen sind zwei Bedrohungen, denen die Systeme und Kommunikationsverbindungen in sim^{TD} ausgesetzt sind. Um diesen Bedrohungen zu begegnen wurde ein IT-Sicherheitskonzept für sim^{TD} entwickelt, dass Schutzmaßnahmen gegen diese und andere Bedrohungen berücksichtigt. Das IT-Sicherheitskonzept sieht vor, dass im Feldversuch jede C2X-Nachricht eindeutig verifizierbar sein muss und vertrauliche Daten zusätzlich verschlüsselt sein müssen. Ein passiver Angreifer darf verkehrssicherheitsrelevante Nachrichten mitlesen, da diese per Broadcast ausgesendet werden. Es muss aber unmöglich für den Angreifer sein, eigene Informationen in das System einzubringen. Das Einspielen von gefälschten oder alten Nachrichten kann die Funktionssicherheit des Gesamtsystems stören und muss somit verhindert werden. Vertrauliche Kommunikation über unsichere öffentliche Netze wie dem Internet muss verschlüsselt durchgeführt werden, damit ein Angreifer den Inhalt der Nachricht nicht mitlesen kann. Des Weiteren sieht das IT-Sicherheitskonzept vor, dass der Schutz der Privatsphäre mit Hilfe von wechselnden Pseudonymen adressiert wird, die das automatisierte langfristige Verfolgen eines Fahrzeugs verhindern. Eine umfassende Analyse der relevanten Bedrohungen und der dazugehörigen Gegenmaßnahmen sowie die daraus resultierende IT-Sicherheitsarchitektur für sim^{TD} und ein späteres Wirksystem finden sich in Deliverable D21.5.

In den folgenden Abschnitten wird auf die Absicherungsmethoden und dessen Komponenten eingegangen. Außerdem werden in Abschnitt 2.5.3, 2.5.4 und 2.5.5 die Absicherungsmechanismen für die verschiedenen Kommunikationswege genauer betrachtet. Abschnitt 2.5.6 adressiert die Absicherung der infrastrukturbasierten IP-Kommunikation.

2.5.1 Grobe IT-Sicherheitsarchitektur

Basierend auf der IT-Sicherheitsanalyse aus Deliverable D21.5 werden Maßnahmen ergriffen um den Schutzbedarf für das sim^{TD} System zu adressieren. Zusätzlich befindet sich die Diskussion der Absicherungsmöglichkeiten der unteren Netzwerkschichten (Layer 1 und Layer 2) im Deliverable D21.3 und die Absicherung der Kommunikationsprotokolle (Layer 3 bis Layer 7) im Deliverable D21.4.

Abbildung 2-47 zeigt eine grobe Übersicht aller wichtigen Komponenten, die in sim^{TD} miteinander kommunizieren. Hierbei lassen sich verschiedene Kommunikationskanäle identifizieren, die IT-sicherheitstechnisch relevant sind:

- Ad hoc ITS Kommunikation (ITS G5A WLAN 802.11p)
- Ad hoc Nicht-ITS Kommunikation (WLAN 802.11 b/g)
- Verbindungen über öffentliche Netze (ITS IMT Public)

Alle Bestandteile der Abbildung in roter Schrift, mit rotem Hintergrund oder mit roter Umrandung werden durch das IT-Sicherheitskonzept abgesichert.

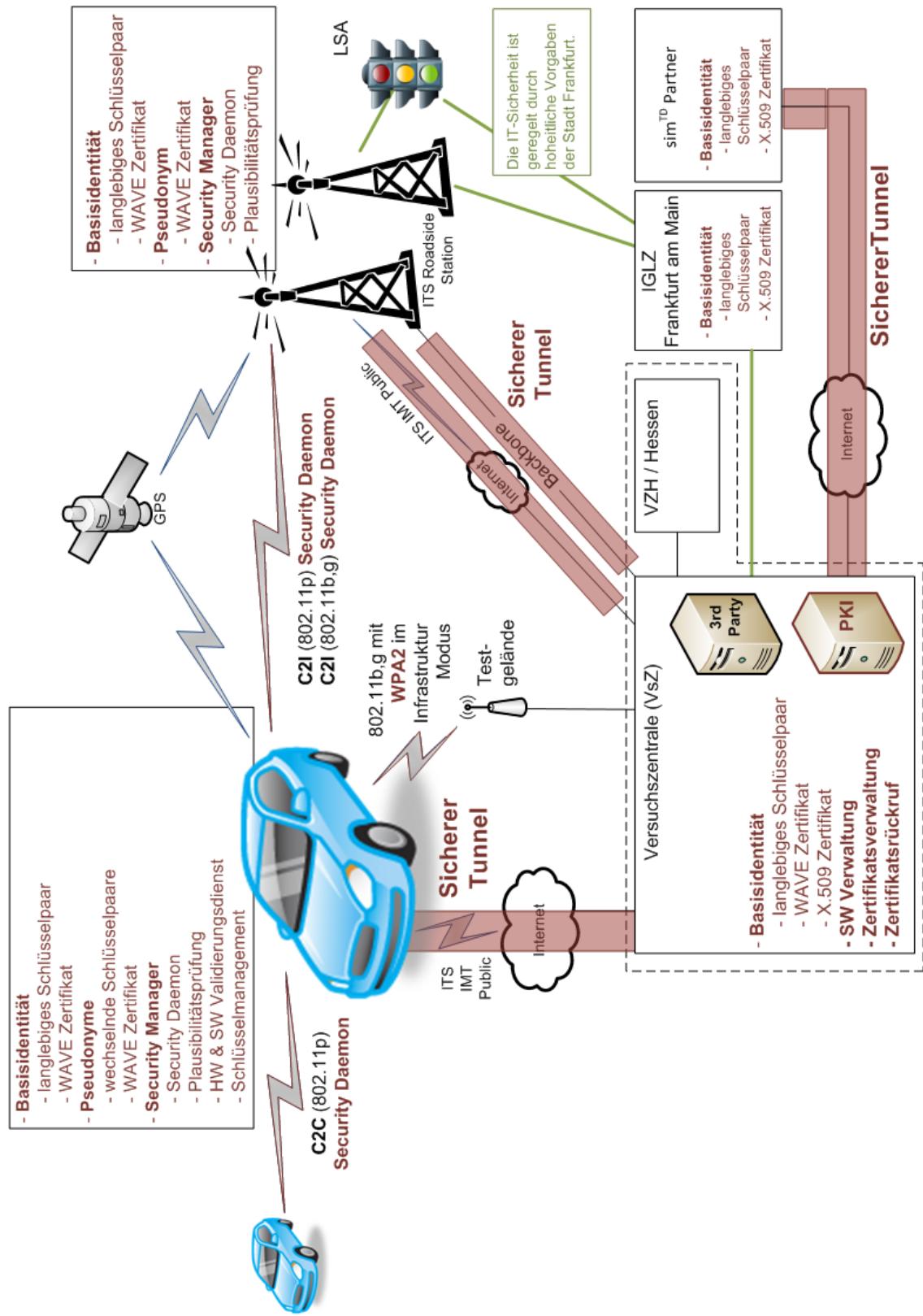


Abbildung 2-47: Grobe IT-Sicherheitsarchitektur

Alle Funktionen, die die Ad hoc ITS Kommunikation über IEEE 802.11p nutzen, werden durch einen speziell für diese Kommunikationsart entworfenen Sicherheitsdienst (Security Daemon) abgesichert. Diese Kommunikation findet statt zwischen Fahrzeugen (C2C) sowie Fahrzeugen und ITS Roadside Stations (C2I). Die Absicherung der Ad hoc ITS Kommuni-

kation über IEEE 802.11p mit der Komponente „Security Daemon“ erfolgt in enger Anlehnung an den existierenden Standard IEEE 1609.2. Dieser Standard basiert auf dem Einsatz asymmetrischer Kryptographie und wird im folgenden Abschnitt 2.5.2 näher beschrieben.

Beim Austausch von C2X-Nachrichten werden alle Informationen über ein sim^{TD}-spezifisches Netzwerkprotokoll ausgetauscht, wie in Deliverable D21.4 beschrieben. Da auf Vermittlungsschicht Mobilitätsdaten in die Nachrichten integriert werden, muss die Integrität bzw. Vertraulichkeit dieser Daten auf Vermittlungsschicht mit dem Security Daemon abgesichert werden.

Über ITS G5a dürfen in sim^{TD} nur verkehrssicherheitsrelevante Nachrichten und Nachrichten zur Verkehrsoptimierung ausgetauscht werden. Alternativ werden Nachrichten mit Consumer-WLAN (IEEE 802.11b/g) übertragen. Eine detaillierte Beschreibung der Absicherungsmaßnahmen für diese Kommunikation findet sich in Kapitel 2.5.4.

Alle Verbindungen über öffentliche Netze und dem Internet müssen prinzipiell mit einem sicheren Tunnel abgesichert werden. Auch die Verbindungen der ITS Roadside Station zu den Zentralen über das eigene Backbone werden mit Hilfe von sicheren Tunneln abgesichert. Dieses betrifft die drahtlose Verbindung per ITS IMT Public von den Fahrzeugen zur Versuchszentrale, die ITS Roadside Stations bei der Kommunikation mit der Zentrale und die sim^{TD}-Partner, die auf die Versuchszentrale von extern zugreifen.

Zu den nicht explizit abgesicherten Kommunikationskanälen gehören die Verbindung von den ITS Roadside Stations zu den LSAs/VBAs, die über einen internen drahtgebundenen Anschluss realisiert werden und somit ausreichend durch physische Schutzmaßnahmen gesichert sind und in sim^{TD} nicht weiter beachtet werden müssen. Die Positionsinformationen von den GPS-Satelliten werden nicht mit kryptographischen Maßnahmen abgesichert. Beim Empfang von C2X Nachrichten werden jedoch die Positionen mit Hilfe der Plausibilitätsprüfung verifiziert.

Die zentralen IT-Sicherheitskomponenten wie Pseudonymverwaltung und PKI (Public Key Infrastructure) sind in der Versuchszentrale untergebracht, zu der alle Teilnehmer des Systems Zugriff haben.

2.5.2 Einsatz von Pseudonymen

Die C2X-Kommunikationsabsicherung beruht auf dem Einsatz asymmetrischer kryptografischer Algorithmen. Prinzipiell wird jede ITS Station mit einer Basisidentität und mehreren wechselnden Pseudonymen ausgestattet, die jeweils aus einem privaten und öffentlichen Schlüssel bestehen. Eine Public-Key-Infrastruktur (PKI) erstellt mit Hilfe des öffentlichen Schlüssels und weiteren Attributen des Teilnehmers ein digitales Zertifikat. Das langlebige Zertifikat der Basisidentität wird in dem IT-Sicherheitskonzept für sim^{TD} ausschließlich für die Authentifizierung gegenüber der PKI bei der Anfrage neuer Pseudonyme verwendet. Die Zertifikate der ausgestellten Pseudonyme hingegen werden für die aktive C2X-Kommunikation verwendet. Die Anfrage neuer Pseudonyme wird über automatisierte Verfahren der Funktion 5.1.1 realisiert. Bei dieser automatisierten Anfrage wird neben den öffentlichen Schlüsseln auch die Fahrzeugkonfiguration in Form einer Versionsliste der installierten OSGi-Bundles übertragen, mit dessen Hilfe eine Prüfung vorgenommen werden kann, sodass Teilnehmer mit falschen Konfigurationen oder unbekannten Softwarekomponenten identifiziert und anschließend manuell überprüft werden können.

Die Pseudonyme werden nur eine kurze Zeitspanne gültig sein und können daher mit sehr kurzen kryptografischen Schlüsseln ausgestattet werden. Das reduziert drastisch den Berechnungsaufwand in den ITS Stations sowie die Länge der Sicherheitsinformationen die jeder Nachricht angehängt werden müssen. Zum Schutz der Privatsphäre werden die Pseudonyme regelmäßig gewechselt – gleichzeitig mit allen anderen Identifikatoren wie zum Beispiel der MAC-Adresse oder der IP-Adresse. Damit jeder Teilnehmer ständig mit mindestens einem gültigen Pseudonym ausgestattet ist, werden die ITS Stations zu Beginn des Feldtestes mit einer größeren Menge von Pseudonymen ausgestattet werden. Die für die Pseudonyme eingesetzten WAVE Zertifikate sind sehr klein (ca. 200 Byte) und sollten wegen der geringen Schlüssellänge nicht länger als ca. 24 Stunden gültig sein. Zum einen ist vorgesehen die ITS Stations für einen längeren Zeitraum mit Pseudonymen auszustatten, die jeweils einen festen Zeitraum gültig sind. Diese Pseudonyme werden in sim^{TD} verwendet wenn keine neuen Zertifikate abgefragt werden können. Zusätzlich zu dieser Grundausstattung fragen die ITS Stations von der PKI neue Pseudonyme ab, um mit wechselnden Pseudonymen kommunizieren zu können.

Eine detaillierte Beschreibung zur Nutzung von digitalen Identitäten und Pseudonymen im Allgemeinen findet sich in Deliverable D21.5 Kapitel 5.1.1. Der sim^{TD}-spezifische Einsatz von digitalen Identitäten und Pseudonymen wird in Kapitel 5.3.1 von Deliverable D21.5 im Detail erläutert.

2.5.2.1 Wechsel von Pseudonymen

Wie bereits erläutert werden die ITS Stations mit mehreren Pseudonymen automatisiert ausgestattet werden, damit zum Schutz der Privatsphäre ein regelmäßiger Wechsel des verwendeten Zertifikats der C2X-Kommunikation erfolgen kann. Aus Sicht der Funktionen ist es sehr wichtig, dass der ID-Wechsel benachbarter Teilnehmer nicht die Funktionalität der Anwendungen, die oft Node Traces verwenden, beeinträchtigt und damit nicht zu einem Ausfall in kritischen Situationen führt. Zu diesem Zweck bietet der Security Daemon Client auf der AU eine Schnittstelle an, mit der die Hauptfunktionen kritische Situationen melden können, in denen kein Pseudonymwechsel durchgeführt werden darf. Nur wenn alle Hauptfunktionen die aktuelle Situation als unkritisch ansehen, wird durch den Security Daemon in regelmäßigen Abständen ein Pseudonymwechsel angestoßen der auf allen Ebenen der Kommunikation die Identifizierer gleichzeitig wechselt.

Jeder eigene Pseudonymwechsel, sowie jeder identifizierte Wechsel bei benachbarten Teilnehmern wird durch die IT-Sicherheitskomponenten geloggt und für eine spätere Versuchsauswertung in sim^{TD} zur Verfügung gestellt.

2.5.2.2 Fahrzeugseitige Pseudonymverwaltung

Im Security Daemon wird eine Datenbank verwaltet, in der die eigenen Schlüssel und Zertifikate verwaltet werden sowie die der benachbarten Fahrzeuge und Roadside Stations. Zusätzlich wird an dieser Stelle das Zertifikat der PKI hinterlegt, mit dem alle Pseudonyme aller sim^{TD}-Teilnehmer verifiziert werden können. Für die Revokation ungültig gewordener Zertifikate werden in dieser Datenbank auch Revokationslisten gespeichert, die über den Verteildienst allgemeiner Sicherheitsparameter (Funktion F_5.1.2) empfangen werden. In sim^{TD} ist nur eine Revokation von Pseudonymen notwendig, die für einen längeren Zeitraum im Voraus ausgegeben wurden. Die Revokation muss jedoch nur so lange aufrechterhalten werden, bis die Pseudonyme abgelaufen sind. Der Missbrauch von Zertifikaten kann in sim^{TD} mit Hilfe der Plausibilitätsprüfung automatisch erkannt werden. Die Revokation der entsprechenden Pseudonyme wird jedoch manuell aktiviert.

Viele Anwendungen/Funktionen der AU basieren auf den feststehenden IDs aller benachbarten C2X-Teilnehmer. Durch den Pseudonymwechsel wird jedoch in regelmäßigen Abständen die ID gewechselt so dass eventuell eine Unterbrechung bei der Verhaltensberechnung bei den Anwendungen vorkommen kann. Um das zu verhindern wird die Plausibilitätsprüfung (Abschnitt 2.5.3.2) eine interne ID für jeden benachbarten Teilnehmer im Empfangsradius vergeben und die Zuordnung auch über einen Pseudonymwechsel hinaus aufrechterhalten. Mit Hilfe der verwendeten Trackingalgorithmen im Plausibilitätsprüfer kann das Fahrverhalten vorausberechnet werden und somit auch einem Knoten eine feste ID anhand seiner Mobilität zuordnen. Zum Schutz der Privatsphäre wird diese intern vergebene ID nicht nach außen kommuniziert.

2.5.2.3 Zentrale Pseudonymverwaltung

Die zentrale PKI in der Versuchszentrale übernimmt die Ausstellung der Basisidentitäten und Pseudonyme. Grundsätzlich wird in sim^{TD} jedes asymmetrische Schlüsselpaar auf dem System erstellt, wo es auch später eingesetzt werden soll. Das heißt die Fahrzeuge und ITS Roadside Stations erstellen den privaten und öffentlichen Schlüssel lokal und übertragen nur den öffentlichen Schlüssel zur Erstellung des Zertifikates. Da jeweils die Anfrage neuer Pseudonyme mit der Basisidentität signiert werden muss, kann die Pseudonymverwaltung in der VsZ eine Zuordnung zwischen Basisidentität und Pseudonym in einer Datenbank abspeichern, um eine spätere Auflösung zu gewährleisten.

Eine weitere Aufgabe der PKI ist die Erstellung und Ausgabe von Revokationslisten, die per F_5.1.2 an alle Teilnehmer verteilt werden. Wie in Abschnitt 2.5.2.2 beschrieben, werden nur Pseudonyme revoziert, die für einen zukünftigen Zeitraum ausgestellt wurden. Basisidentitäten brauchen nicht mit Revokationslisten gesperrt werden da sie nicht für die aktive C2X Kommunikation genutzt werden können.

2.5.3 ITS G5A WLAN 802.11p

In diesem Abschnitt wird die preventive Absicherung durch asymmetrische Verfahren in Abschnitt 2.5.3.1 vorgestellt sowie die Erkennung von fehlerhaften Nachrichten und Teilnehmern in Abschnitt 2.5.3.2.

2.5.3.1 Signierung und Verschlüsselung

Die Absicherung der ITS G5A Kommunikation wird von dem Security Daemon übernommen. Dieser wird auf der CCU platziert sein, aber auch mit Hilfe eines Clients von der AU aus erreichbar sein. Auf der CCU befindet sich ebenso die Komponente SIM-NET, die unter anderem den IEEE 802.11p Kommunikationsstack implementiert.

Die Platzierung des Security Daemon auf der CCU ist technisch bedingt und wird aus folgendem Grund in dieser Form umgesetzt. Die zu schützenden Informationen wie Position, Geschwindigkeit und Fahrtrichtung werden auf Vermittlungsschicht durch SIM-NET auf der CCU in die Pakete integriert. Da der Security Daemon diese Daten genauso schützen muss wie auch die Informationen in den Paketen der Anwendungsebene, kann der Schutz erst nach dem Hinzufügen der Informationen durchgeführt werden. Dazu arbeitet SIM-NET eng mit dem Security Daemon zusammen. Eine Verteilung von SIM-NET und Security Daemons auf unterschiedliche Systeme würde die Komplexität der Kommunikation und damit die Fehleranfälligkeit des gesamten ITS Ad hoc Kommunikationssystems erhöhen.

SIM-NET ist zuständig für den Aufruf des Security Daemons, um die Absicherung der Nachrichteninhalte auf Vermittlungsschicht sicher zu stellen. Der Security Daemon stellt dafür über seine Schnittstelle mit folgenden Operationen zur Verfügung:

- Signieren
- Verifizieren
- Verschlüsseln
- Entschlüsseln

Details zur Absicherung der C2X-Kommunikation per IEEE 802.11p werden in Deliverable D21.5 in Kapitel 5.3.3 zur Verfügung gestellt.

2.5.3.2 Plausibilitätsprüfung

Die Plausibilitätsprüfung ist für die Erkennung von fehlerhaften Informationen innerhalb der Nachrichten zuständig. Die Komponente ist in sim^{TD} auf der AU in dem Bundle „Plausibilitätsprüfer“ untergebracht. Das Bundle ist in den Kommunikationsweg aller C2X-Nachrichten integriert und kann somit die Inhalte, primär die Mobilitätsdaten, bewerten.

Eine grundlegende Prüfung der Wertebereiche der Mobilitätsdaten stellt die Basis der Plausibilitätsprüfung dar. Weiterhin kann eine Prüfung der Sendefrequenz von Nachrichten des gleichen Absenders einen Denial of Service (DoS) Angriff erkennen und die Umfeldtabelle unterstützen eine Überfüllung zu verhindern. Weiterhin können die Mobilitätsdaten der Nachrichten zu einer Verfolgung (tracking) aller benachbarten Fahrzeuge im direkten Empfangsbereich genutzt werden. Mit dieser Prüfung kann festgestellt werden, ob das Verhalten des Absenders plausibel ist oder er durch nicht-plausibles Auftreten bösartig oder ungewollt falsche bzw. störende Informationen verbreitet. Eine zentrale Plausibilitätsprüfung kann die Positionen aller C2X-Nachrichten zu einem Bewegungsmodell umsetzen und somit zum einen fehlerhafte Daten identifizieren und zum anderen nicht plausible Absender entdecken.

Da der Plausibilitätsprüfer ein Tracking aller benachbarten Teilnehmer im Empfangsbereich durchführt, wird diese genutzt, um eine interne ID zu vergeben, die auch über einen Pseudonym Wechsel hinweg bestand hat. Wie auch das Ergebnis der Plausibilitätsprüfung wird die Knoten ID in das C2X-Message-Objekt geschrieben und vom Communication Client in der Umfeldtabelle abgelegt. Mit Hilfe dieser ID können die Funktionen auf eine konstante ID der benachbarten Teilnehmer aufbauen.

Weiterhin ist mit dem Plausibilitätsprüfer die Erstellung einer CAM-Prädiktion möglich, die eine oder mehrere zukünftige Positionen benachbarter Fahrzeuge vorausberechnen kann. Mit Hilfe dieser Prädiktion kann unter anderem der zurückgelegte Weg eines CAM-Absenders zwischen Aussendung und dem Zeitpunkt bei der Verarbeitung der CAM beim Empfänger berechnet werden.

2.5.4 WLAN 802.11 b/g

In sim^{TD} wird zwischen drei verschiedenen Kommunikationsvarianten über Consumer-WLAN unterschieden. Abbildung 2-48 zeigt eine Übersicht der Kommunikationskanäle die mit Consumer-WLAN genutzt werden.

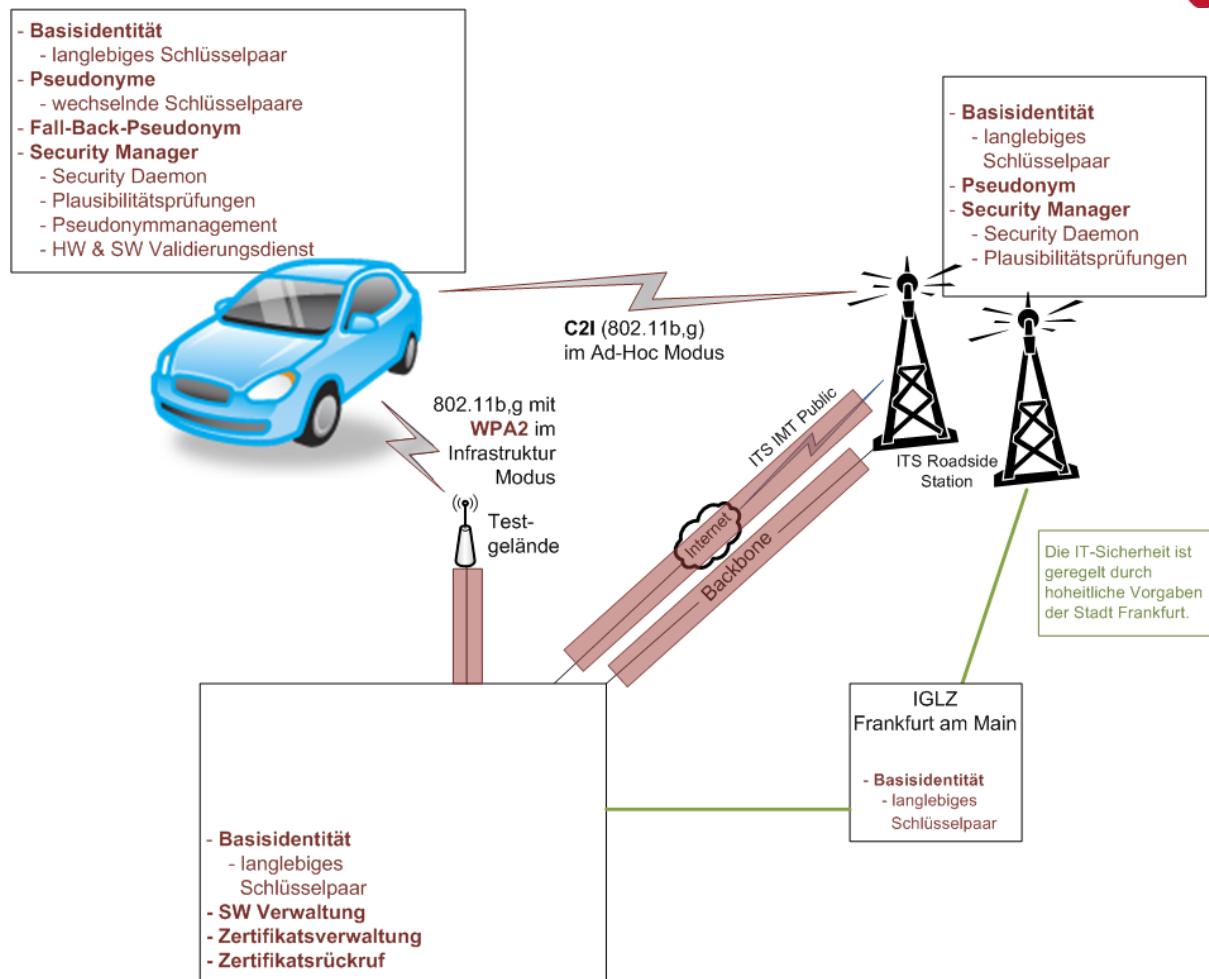


Abbildung 2-48: Einsatz von kommerziellem WLAN IEEE 802.11 b/g

2.5.4.1 C2X-Nachrichten über Consumer-WLAN

Zum einen sollen C2X-Nachrichten per Consumer-WLAN zwischen ITS Roadside Station und ITS Vehicle Station ausgetauscht werden. Die ITS Roadside Station stellt ein offenes Ad-hoc-WLAN-Netz zu Verfügung, zu dem sich jedes Fahrzeug verbinden kann. Da eine Absicherung dieser Kommunikation auf Layer 1 und 2 in sim^{TD} nicht sinnvoll ist, erfolgt die Kommunikation über einen unverschlüsselten WLAN-Kanal.

Für die Absicherung werden in diesem Fall die gleichen Mechanismen eingesetzt wie sie auch bei der IEEE-802.11p-Kommunikation Verwendung finden. Für den Security Daemon ist der Übertragungskanal transparent, da SIM-NET die gleichen Protokolle auf Vermittlungsschicht einsetzt. In Abbildung 2-48 wird die Kommunikation von C2X-Nachrichten zwischen Fahrzeugen und Roadside Stations dargestellt und mit C2I im Ad-hoc-Modus bezeichnet.

2.5.4.2 Consumer-WLAN zur Übertragung anwendungsspezifischer Nachrichten

Standortinformationsdienste sollen mit Hilfe von anwendungsspezifischen Nachrichten übertragen werden, die per UDP/IP über WLAN 802.11b/g zwischen einer ITS Roadside Station und einer ITS Vehicle Station übertragen werden. Da keine Absicherung auf Layer 1 und 2 möglich ist, müssen Maßnahmen auf Anwendungsebene die Integrität bzw.

Vertraulichkeit der Nachrichten sicherstellen. Des Weiteren kann die Authentizität des Absenders verifiziert werden, wenn die gleichen Pseudonyme wie in der IEEE 802.11p Kommunikation eingesetzt werden.

Anwendungsspezifische Nachrichten werden signiert, falls sie per Broadcast verschickt werden und optional verschlüsselt, wenn sie per Unicast versendet werden. Diese Absicherung erfolgt nicht automatisch, sondern muss von jeder Anwendung beim Versand und beim Empfang durch einen Aufruf des Security Daemons erfolgen. Darüber hinaus sollte eine Funktion darauf achten, dass die Nachrichtengröße nicht die MTU des zu Grunde liegenden WLAN-Kanals überschreitet². Ansonsten muss die Übertragung über den verbindungslosen Kanal in mehrere zusammenhängende Pakete aufgeteilt werden, was die Wahrscheinlichkeit für eine erfolgreiche Übertragung signifikant reduziert und so zu Problemen führen kann.

Bei dieser Kommunikationsvariante muss die Funktion bei jedem Versand von anwendungsspezifischen Nachrichten den Security Daemon aufrufen. Dieser erstellt eine sichere Nachricht nach IEEE 1609.2 und gibt diese an die Anwendung zurück. Die Funktion kann nun die sichere Nachricht per IP versenden. Beim Empfang muss nun analog jede anwendungsspezifische Nachricht durch den Security Daemon geleitet werden, um die eigentliche Nachricht aus dem sicheren Nachrichtenformat zu extrahieren.

Details zu der Absicherung von Ad hoc Consumer-WLAN-Verbindungen sind in Deliverable D21.5 in Kapitel 5.3.5 zu finden.

2.5.4.3 Consumer-WLAN im Infrastrukturmodus

Auf dem Testgelände oder den Parkplätzen der Hired-Driver von sim^{TD} werden Access Points installiert, die für die Kommunikation mit den Fahrzeugen im Infrastrukturmodus betrieben werden. Die Kommunikationsabsicherung findet nach IEEE 802.11i im Personal Mode (WPA2) statt und befindet sich auf der Sicherungsschicht des ISO/OSI-Modells. Eine detaillierte Beschreibung ist in Deliverable D21.5 in Kapitel 5.3.5 zu finden.

2.5.5 ITS IMT Public

Die Datenübertragung über das Mobilfunknetz wird in sim^{TD} für zwei unterschiedliche Informationsklassen genutzt. Zum einen sollen per ITS IMT Public anwendungsspezifische Daten per TCP/IP zwischen Fahrzeugen bzw. ITS Roadside Stations und der Versuchszentrale übertragen werden. Die Absicherungsmechanismen werden im Abschnitt 2.5.5.1 näher erläutert.

Des Weiteren sollen aber auch C2X-Nachrichten über Mobilfunk zu ITS Stations in einem bestimmten geografischen Gebiet übertragen werden. Hierzu wird in der VsZ ein Server installiert werden, der die Verteilung dieser Nachrichten übernimmt. Die Absicherung dieser Datenübertragung wird in Abschnitt 2.5.5.2 beschrieben.

² Von der MTU muss zusätzlich noch die Länge des Security-Headers abgezogen werden, der die für die Absicherung notwendigen Schlüssel und Informationen enthält.

2.5.5.1 Datenübertragung

In sim^{TD} wird aus Performancegründen auf eine Absicherung durch einen sicheren Ende-zu-Ende-Tunnel verzichtet. Da die Luftschnittstelle des Mobilfunknetzes verschlüsselt ist müssen lediglich die Kommunikationswege zwischen dem Mobilfunknetz und der Versuchszentrale, welche evtl. über das Internet laufen, verschlüsselt werden. Um diese zu gewährleisten wird das System *Mobile IP VPN basic* von T-Mobile eingesetzt mit dessen Hilfe die Daten durch ein VPN zwischen dem Zugangspunkt (Access Point Name, APN) des Mobilfunkbetreibers und der Versuchszentrale geschützt werden. Da es sich bei dieser Absicherungsvariante um keine Ende-zu-Ende-Verschlüsselung handelt, muss dem Mobilfunkbetreiber vertraut werden.

2.5.5.2 Übertragung von C2X-Nachrichten

Für die Übertragung von C2X-Nachrichten zwischen Fahrzeugen und zwischen VsZ und Fahrzeugen wird ein Geo-Server eingesetzt, der eingehende Nachrichten per ITS IMT Public an Fahrzeuge in einem bestimmten geografischen Gebiet aussendet. Damit dieser Server stets die IP-Adressen aller Fahrzeuge zu einem geografischen Gebiet zuordnen kann, müssen regelmäßige Nachrichten (Beacons) von den Fahrzeugen zum Geo-Server übertragen werden. Jedes Beacon wird wie eine CAM-Nachricht durch den Security Daemon auf der CCU signiert und anschließend per ITS IMT Public versendet. Auch die Absicherung der C2X-Nachrichten wird wie bei der IEEE 802.11p Kommunikation durch den Security Daemon signiert. Eine verschlüsselte Übertragung von C2X-Nachrichten per Unicast ist in sim^{TD} nicht angedacht.

2.5.6 Absicherung der IP-basierten Kommunikationsverbindungen

Neben der Datenübertragung per TCP/IP über das Mobilfunknetz werden Daten zwischen sim^{TD}-Partnern und der Versuchszentrale über das Internet übertragen. Des Weiteren muss die Kommunikation zwischen Testgelände und Versuchszentrale abgesichert werden. Dazu wird mit Hilfe von X.509v3 Zertifikaten eine sichere VPN-Verbindung erstellt. Diese kann bei Bedarf auch für die externen sim^{TD}-Partnerzugriffe genutzt werden. Wenn jedoch nur der Zugriff auf bestimmte Dienste der VsZ notwendig ist, sollte vorzugsweise eine SSL Verbindung wie zum Beispiel per HTTPS oder FTPS genutzt werden.

Weitere Details zur Absicherung der IP-basierten Kommunikation in sim^{TD} kann in Kapitel 5.3.5 von Deliverable D21.5 nachgelesen werden.

2.6 Basisidentifikation der ITS Stations

Für die Auswertung der Messdaten der ITS Stations wird eine Basisidentifikations ID benötigt, welche sich während der kompletten sim^{TD} Versuchslaufzeit nicht ändert. Das Basiszertifikat der IT Sicherheit stellt eine Identifikation der jeweiligen Einheit dar. Die Hash-Werte dieser Zertifikate sollen jedoch nicht für die Auswertung der Messdaten verwendet werden. Aus diesem Grunde wird eine zusätzliche neue Identifikation eingeführt.

Diese Basisidentifikation wird lokal auf den jeweiligen ITS Stations von den Integratoren (Bosch für Vehicle Station, HTW für Roadside Station und HLSV für Central Station) gesetzt, und kann von allen Komponenten und Funktionen auf der jeweiligen Station durch ein einfaches Interface für verschiedene Zwecke abgefragt werden.

Eine Verwendung der Basisidentifikation darf nicht beliebig erfolgen. Um eine Gefährdung der Privatsphäre der Benutzer zu reduzieren gelten folgende Einschränkungen:

- Die Basisidentifikation darf nur lokal verwendet werden
- Daten welche die Basisidentifikation enthalten dürfen nur dann zwischen Systemen übertragen werden, wenn die Vertraulichkeit gewährleistet ist (das bedeutet, dass der Kanal gegen unberechtigtes mithören gesichert ist)

Bereits bisher identifizierte Verwendungszwecke sind:

- Zuordnung der Messdaten Logfiles bei der Übertragung und Speicherung auf den Logdatenspeichern zu einzelnen ITS Stations
- Verknüpfung zwischen der Identität in den Domänen IT Security und Logging

Hierbei wird folgendes Nummerierungsschema verwendet:

- Für die Roadside Stations: IRS001 – IRS110 (voraussichtliche Anzahl von Roadside Stations)
- Für die Vehicle Stations: IVS001 – IVS400
- Für die Central Station: ICS001 (die Versuchszentrale)

Bei Bedarf kann der Zahlenraum für die Identifier erweitert werden. Es ist hier explizit zu vermerken, dass diese Identifier lediglich für den Feldtest sim^{TD} verwendet werden um eine einfache Identifikation der ITS Stations zu ermöglichen.

Die Schnittstelle zur Abfrage der Basisidentifikations ID wird über den Systemmanager realisiert, welcher eine einfache Methode zur Abfrage der ID als String zur Verfügung stellt.

3 Beschreibung der Teilsysteme

3.1 Vehicle/Roadside CCU

Die CCU der ITS Vehicle Station und der ITS Roadside Station ist das zentrales Kommunikationsgerät von sim^{TD}, da es sowohl in den IVS als auch in den IRS verwendet wird und alle verwendeten Funk-Kommunikationsstandards implementiert. Dies sind im Einzelnen:

- ITS G5A
- Consumer-WLAN gemäß IEEE 802.11b/g
- UMTS (inklusive HSDPA, GSM und Edge)

Für alle Funkstandards wird jeweils Physical Layer, MAC Layer und Netzwerk/Transport Layer auf der CCU implementiert. Für Consumer-WLAN und UMTS ist das Netzwerk/Transport Protokoll TCP/IP für IEEE 802.11p ist es SIM-NET.

Die Anbindung an die AU der IVS bzw. IRS wird über Fast Ethernet, TCP/IP Netzwerkprotokoll und (teilweise) über sim^{TD}-spezifische Applikationsprotokolle abgewickelt.

In einer IVS wird das Fahrzeug über zwei CAN Interfaces und zwei digitale Eingänge und zwei digitale Ausgänge realisiert. Zusätzlich steht ein weiteres CAN-Interface für projektspezifische Zwecke zur Verfügung. Die Fahrzeugdaten werden auf der CCU in ein vom Fahrzeugtyp unabhängiges Datenformat umgesetzt und der AU zur Verfügung gestellt.

Die CCU stellt weiterhin für interne Zwecke und für die AU eine unterbrechungsfreie Eigenposition zur Verfügung. Hierzu wird ein GPS Modul verwendet und in einer IVS wird dessen Position durch die Fahrzeugdaten verbessert und extrapoliert.

Für die Zwecke der Versuchssteuerung und für die Versuchsdatenerhebung stellt die CCU weiterhin noch eine Fernwartungsfunktion zur Verfügung und stellt Messwerte aus den Kommunikationssystemen für den Logging Stub auf der AU bereit.

Folgendes UML-Diagramm gibt einen Überblick über die Hardware der CCU.

[Back to CCU-Router](#)

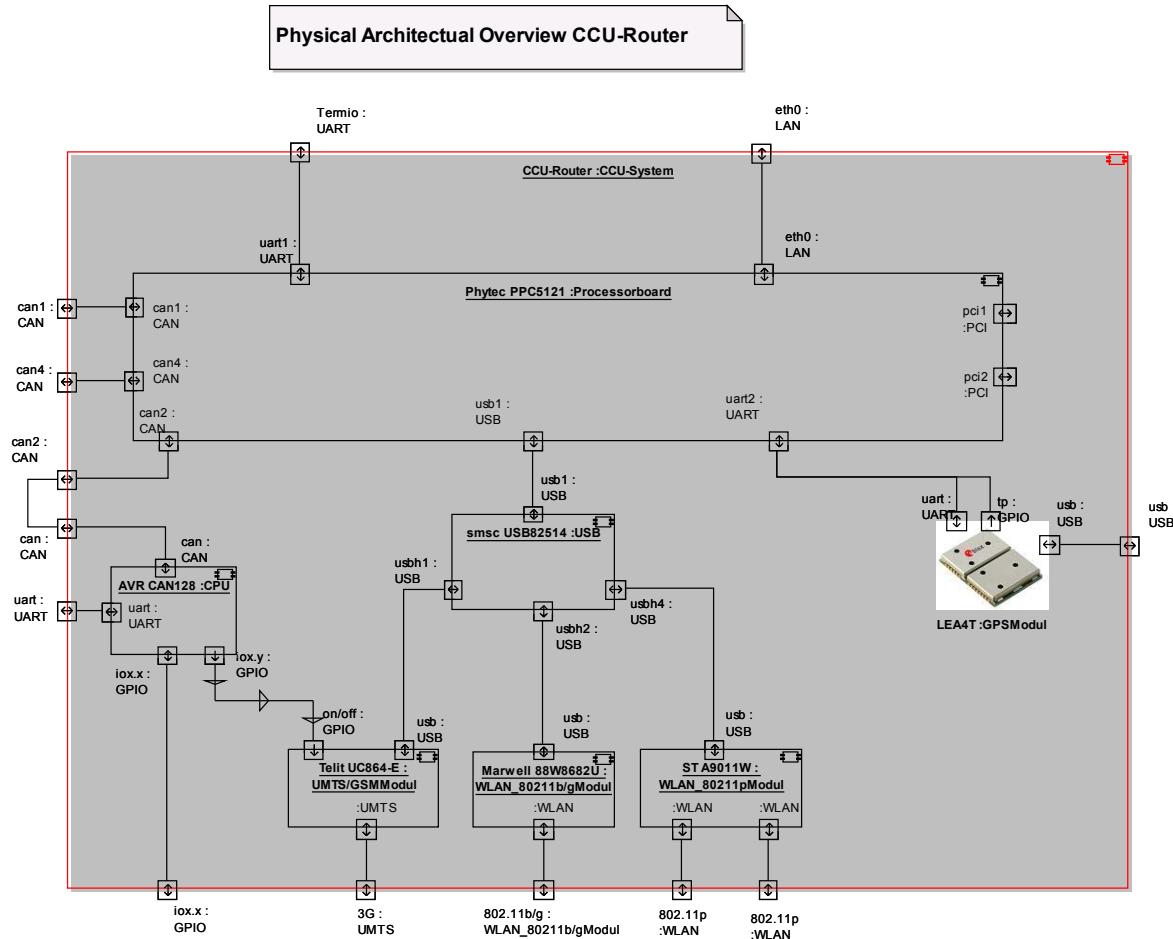


Abbildung 3-1: Hardwareüberblick IVS CCU

Die Kommunikationskomponenten und GPS werden jeweils an den zentralen Prozessor angeschlossen, der zur Entlastung von einfachen IO-Aufgaben durch einen Atmel Microcontroller ergänzt wird.

[Back to Viewpoints](#)

System Architectural Overview CCU-Router

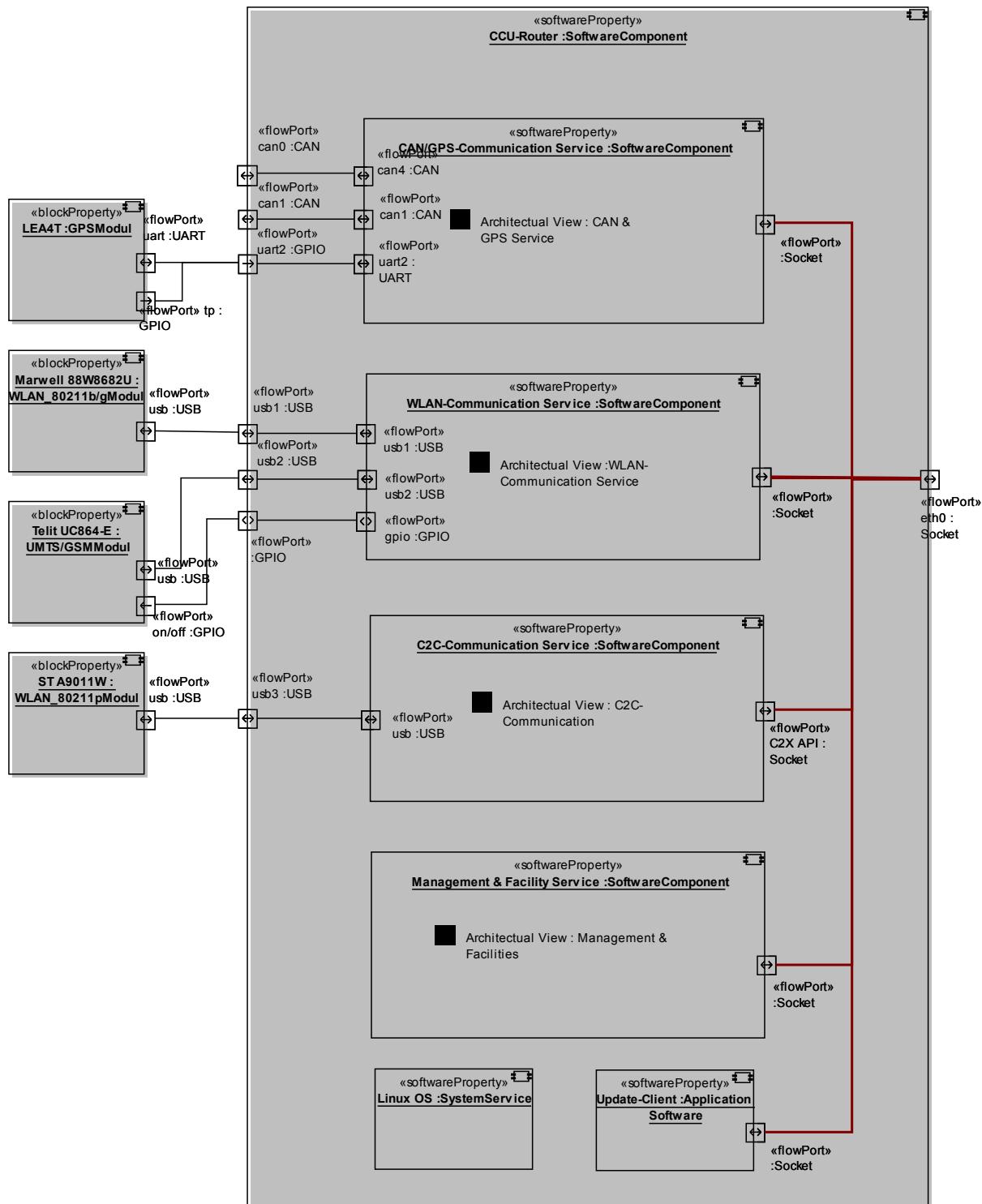


Abbildung 3-2: IVS CCU Systemarchitektur

Abbildung 3-2 stellt die oberste Ebene der Softwarearchitektur des CCU-Routers dar. Daneben werden die wesentlichen Verbindungen zu den Hardware-Modulen (wie WLAN etc.) aufgeführt.

Neben den drei großen Software-Komponenten CAN/GPS Communication Service (Abschnitt 3.1.1), WLAN Communication Service (Abschnitt 3.1.4), C2C Communication Service und Management & Facilities (Abschnitt 3.1.3) besteht diese Ebene aus folgenden weiteren Software-Komponenten:

- Linux OS: Neben den Software-Komponenten werden noch folgende (Echtzeit-) Applikationen auf dem Router zur Verfügung gestellt:
 - Update-Client

3.1.1 GPS

Folgendes UML-Diagramm zeigt den logischen Aufbau des GPS-Subsystems auf der CCU.

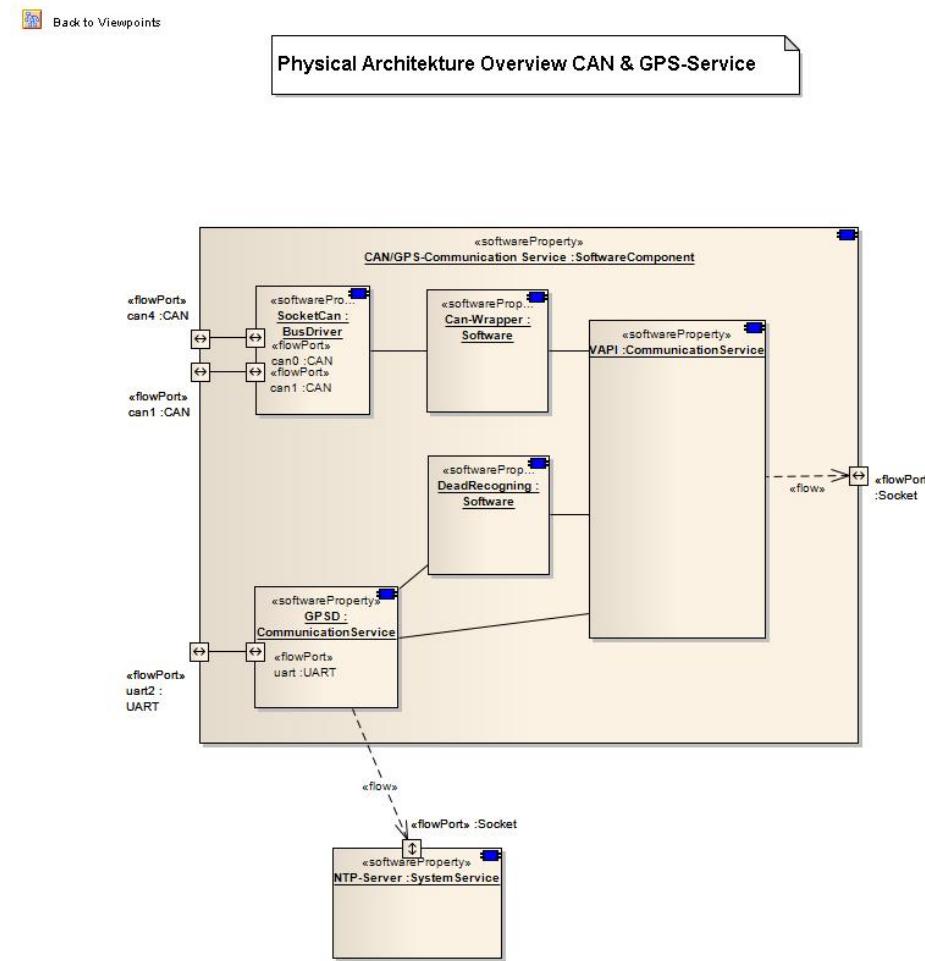


Abbildung 3-3: GPS-Subsystem

Als GPS-Modul wird ein LEA4T Modul der Firma µBlox verwendet, dieses Modul stellt mit über EGNOS empfangenen D-GPS-Daten korrigierte und validierte Positionsdaten mit einer

Updaterate von maximal 4 Hz zur Verfügung und kann GPS-Rohdaten mit einer Rate von 10 Hz liefern. Außerdem generiert das Modul einen Timing-Puls, der dazu genutzt wird, exakt festzulegen zu welchem Zeitpunkt eine Zeitinformation des Moduls gültig ist. Hierdurch kann eine zeitliche Synchronität aller internen CCU Zeiten gewährleistet werden, wobei die maximale Abweichung maximal einige μ s beträgt. Die Verarbeitung des Time-Pulses übernimmt der GPSD, der auch die Interpretation des Positionsdatenstroms übernimmt. Durch den GPSD wird ein NTP-Server synchronisiert, dessen Informationen über einen IP-Port verfügbar gemacht werden.

Um die Qualität der Positionsdaten zu verbessern und diese auch bereitzustellen, wenn keine ausreichenden GPS-Signale empfangen werden, wird das Modul „bessere Ortung“ verwendet. Als Input bekommt es die Daten des GPSD und Fahrdynamikinformationen aus der VAPI, d.h. vom Fahrzeug CAN-Bus. Die Ausgangsdaten des Moduls „bessere Ortung“ werden in eine Pseudo-CAN-Nachricht verpackt und über das VAPI-Framework an die AU und an SIM-NET geliefert.

3.1.2 Bessere Ortung

Die Komponente „Bessere Ortung“ wurde oben in Abschnitt 2.3 im Überblick dargestellt. Im Folgenden werden die Architektur- und vernetzungsrelevanten Aspekte des Teilsystems beschrieben.

3.1.2.1 Zeit und Positionsgenauigkeit

Die Komponente „Bessere Ortung“ liefert folgende Daten:

1. Aktuelle Position in Geokoordinaten als Länge und Breite
2. Zeitstempel, zu dem diese Position gültig war
3. Dynamische Genauigkeitsschätzung in Form einer Genauigkeitsellipse.

In der Genauigkeitsellipse wird die „bestmögliche Genauigkeit“, welche zeitlich in Abhängigkeit von den aktuellen Messbedingungen variiert, durch Kenngrößen beschrieben. Kenngrößen sind die maximale und die orthogonale minimale geschätzte Standardabweichung der Position als Halbachsen der Ellipse sowie die Richtung der maximalen geschätzten Standardabweichung, d.h. der größeren Halbachse relativ zur Nordrichtung). Wegen des Einflusses der Umgebungsbedingungen auf den GPS-Empfang ist eine garantierter Einhaltung von Schwellwerten für die Messgenauigkeit nicht möglich.

Inwieweit die aktuelle Positionsgenauigkeit für eine bestimmte Funktion ausreicht, muss von der Funktion selbst entschieden werden. Die für eine Funktion erforderliche „Mindestgenauigkeit“ kann von Situationsfaktoren abhängig sein. Beispielsweise erfordern Sicherheitsanwendungen wie die Kreuzungs-Querverkehrsassistenz typischerweise eine Mindestgenauigkeit von ca. 3,5 m, welche auf einspurigen Zufahrten für eine gut akzeptable Systemfunktion ausreicht. Auf mehrspurigen Zufahrten ermöglicht eine höhere Genauigkeit von ca. 1 m und eine damit mögliche Spurzuordnung und verbesserte Fahrerabsichtsmodellierung eine Reduktion von potentiellen Fehlwarnungen. Positionsgenauigkeiten von Egofahrzeug und relevanten Fremdfahrzeugen wirken sich gemeinsam auf die Funktionsgüte aus.

Generell wird empfohlen, dass Applikationen die für sie zweckmäßigen Güteschwellen selbst definieren und ggf. auf eine „zu schlechte“ Positionsgüte mit einer Reduktion des

Funktionsumfangs („graceful degradation“) oder durch kontrollierte temporäre Nichtverfügbarkeit reagieren. Es ist ferner zweckmäßig, dass sich verwandte Funktionen auf die gleichen Schwellwerte verständigen (z.B. Aktive Sicherheit).

3.1.2.2 Verwendung von DGPS-Korrekturdaten

Die Komponente „Bessere Ortung“ erhält vom GPS-Modul die Ausgangsdaten des GPS- Receivers (Position in Länge und Breite und Höhe zusammen mit der vom Receiver ermittelten Positionsgenauigkeit). Um möglichst gute Positionsdaten vom Receiver zu erhalten, ist dieser im Regelfall DGPS-fähig. Daher versorgt, sich die GPS-Komponente auch mit einem Korrektursignal welches an den DGPS-Receiver weitergeleitet wird. In Abwesenheit des Korrektursignals (z.B. bei Störungen) liefert das GPS-Modul weiterhin die gleichen Ausgangsdaten wie bei Verfügbarkeit des Korrektursignals, jedoch mit einer wesentlich geringeren Genauigkeit. Die aktuelle Genauigkeit wird laufend in den Ausgangsdaten an die Kunden weitergeben. Weitere Informationen über die technischen Aspekte, z.B. ob und wann ein Korrektursignal verfügbar war, werden nicht weitergegeben, sondern ggf. über die Loggingkomponente für Debugging-Zwecke komponentenintern protokolliert. Applikationen sollen sich bewusst unabhängig von technischen Aspekten nur an die geschätzte Positionsgenauigkeit adaptieren, zumal einzelne kurzzeitige Ausfälle durch die verwendete Positionsfilterung teilweise überbrückt und ausgeglichen werden können.

Die „Bessere Ortung“ benötigt eine Quelle für die Korrektursignale über einen Kommunikationskanal:

1. Entweder über den Zellularenfunk (UMTS, ggf. GSM/GPRS). Das ist die von externen Anbietern angebotene Standardlösung.
2. Über den sim^{TD} -eigenen 802.11p-basierten Kommunikationsanal von der IRS.

Architektonisch werden derzeit beide Möglichkeiten offen gehalten:

1. Die GPS-Komponente bezieht über Systemdienste („gpsd“) vom Mobilfunk-Interface des Routers Korrektursignale.
2. In der IRS wird eine derzeit noch leere „Korrektursignal-Copy-Komponente“ vorgesehen, welche an der IRS über eine beliebige Backbone-Anbindung eintreffende Korrektursignale über IEEE 802.11p weiterversprechen kann.

Da die Korrektursignale von keinem anderen Kunden als der „Besseren Ortung“ genutzt werden, ist neben der Bereitstellung der Kommunikations- und Transportmechanismen kein weiterer Architekturvorrhalt bzw. komponentenexterne Spezifikationen erforderlich. Die benötigten Datenraten sind vergleichsweise gering (ca. 500 Byte alle 5-10s). Als Nachrichtenprotokoll wird ein existierendes Standardprotokoll für Korrekturdaten verwendet (RTCMv2), im Nachrichtenkatalog von sim^{TD} muss lediglich ein Nachrichtentyp-Identifier für die Korrekturdaten vorgehalten werden.

Als Quelle für die Korrekturdaten kommen in Frage:

1. Externe kommerzielle Provider
2. Alternativ wenige sim^{TD} eigene Referenzstationen, die Korrektursignale generieren

Beispielsweise wird die Firma AXIO-Net GmbH (Betreiber des Ascot-Positionierungs-Service) den Dienst Ascot ED (http://www.ascos.de/uploads/tx_v3downloads/Produktblatt_ED_2009_01.pdf, nominelle Genauigkeit 30-50cm) für sim^{TD} in der gesamten sim^{TD}-Flotte bereitstellen (kostenfrei exklusive Datenübertragungskosten). Das

Datenvolumen von Ascot ED beträgt ca. 0,6 MB/h und Nutzer. Damit stehen für simTD die Korrekturdaten zu Verfügung, welche der designierte sim^{TD} GPS-Receiver (u-blox LEA-4T) noch verarbeiten kann.

Sollten auch manchmal (z.B. wegen Abriss der Datenverbindung zum Ascot-Dienst) keine Korrekturdaten verfügbar sein, entsteht hierdurch kein Architekturrisiko, da eine Nichtverfügbarkeit der Korrektursignale lediglich zu ungenauerer Positionen führt, aber keine Auswirkungen auf Interfaces der Architektur hat.

Zusätzlich ist vorgesehen, dass die Korrekturdaten auch mit den C2X-Nachrichten versendet werden (Details hierzu finden sich in Deliverable D21.4). Hiermit soll sichergestellt werden, dass möglichst viele Fahrzeuge über Korrekturdaten verfügen, auch wenn zeitweise nicht direkt die Korrekturdaten von den beiden genannten Quellen bezogen werden können. Eine Feinspezifikation der Schnittstellen sowie die Auswahl welche Korrekturdaten versendet werden sollen wird im Rahmen der Feinspezifikation definiert.

3.1.2.3 Positionsfilterung von Fahrzeugdaten über VAPI

Die Komponente „Bessere Ortung“ bezieht von der VAPI Fahrzeugdaten und speist diese mit der DGPS-Signal-Position in das interne Positionsfilter ein, um am Ausgangssignal wesentlich bessere, optimal gefilterte und gegenüber kurzzeitigen GPS-Ausfällen und Multi-Path-Effekten stabilisierte Positionsdaten zu bekommen.

3.1.2.4 Mapmatching und Digitale Karte

Während die Komponente „Bessere Ortung“ auf dem Router keinen Zugriff auf die Karte hat, findet in der AU-Komponente „Digitale Karte“ ein „Mapmatching“ der von der „Besseren Ortung“ gelieferten Position auf die digitale Karte statt. Dieses kann bei geringer Positionsgenauigkeit etwa infolge längerer Nichtverfügbarkeit des D-GPS-Signals (s.o.) zu einer verbesserten Position führen. Die „mapgematchte“ Position kann von anderen Komponenten und Funktionen als Ergebnis der Komponente „Digitale Karte“ ebenfalls bezogen werden. Welche Positionsinformation für eine bestimmte Aufgabe besser geeignet ist, hängt von der Problemstellung und der aktuellen Positionsgenauigkeit ab. Es bleibt daher der jeweiligen Kundenkomponente überlassen, welche Position sie wie in ihrer Implementierung verwendet.

3.1.2.5 Mapmatching und Funk-Topologiedaten

Neben der digitalen Karte als Informationsquelle zur Fahrumgebung können vor allem vor Kreuzungen Topologie- und Geometriedaten³ auch per Funk übertragen werden (vgl. z.B. SAE J 2735). Verfügbare Funk-Topologiedaten werden ebenfalls über die Komponente digitale Karte an ihre Kunden weitergereicht, um alle Karteninformationen zentral in einer liefernden Komponente zu bündeln. Bei der derzeit laufenden Ausschreibung (Stand Juli 2009) für die Teilkomponenten der digitalen Karte wird eine möglichst integrierte

³ Hinweis: In diesem Dokument wird nicht streng zwischen Topologie (Verbindungen im Raum) und Geometrie (Ausformung von Raumbereichen) unterschieden. Die im Projekt realisierten verwendete Datenrepräsentation ist eine reduzierte und abstrahierte Beschreibung der topologischen und geometrischen Eigenschaften der Fahrumgebung, welche für die realisierten Funktionen benötigt werden.

Applikationsschnittstelle zu kartenbasierten und funkbasiereten Geometrie- bzw. Topologieinformationen angestrebt.

3.1.2.6 Rückführung von Mapmatching-Daten zur Positionsverbesserung

Zur weiteren Positionsverbesserung ist es grundsätzlich möglich, die mittels einer Karte mapgemachte, verbesserte Position wieder an die Komponente „Bessere Ortung“ zurückzuführen. Dazu wäre es erforderlich, die Möglichkeit eines Bezugs von Ausgangsdaten von AU-Komponenten als Eingangsdaten der Router-Komponenten, also entgegen des Hauptdatenflusses, in der Architektur vorzuhalten.

Derzeit ist die rückgeführte Positionsverbesserung nicht erforderlich und nicht geplant. Die regelungstechnische Kontrolle der Rückwirkungsschleife erscheint in der jetzigen sim^{TD}-Architektur mangels garantiertem Zeitverhalten und in Anbetracht variabler Latenzen als risikobehaftet und aufwändig. Ferner kann eine genaue Position durch das Matchen mit einer vergleichsweise ungenau georeferenzierten Karte auch deutlich verschlechtert werden. Eine kartenunabhängig über DGPS gewonnene Position ist daher von weniger Einflussfaktoren abhängig und besser nachvollziehbar.

Später – in oder nach sim^{TD} – könnte bei Bedarf eine rückgeführte Positionsverbesserung zusätzlich implementiert werden und so ein iterativ mit Kartenzugang optimiertes zusätzliches Positionssignal ermöglichen. Für ein gutes Ergebnis muss dafür die Genauigkeit der Georeferenzierung der Karte an der aktuellen Position zur Verfügung stehen oder abgeschätzt werden können. Aus aktueller Sicht erscheint eine Rückkoppelung des Positionssignals nicht sinnvoll (u.a. wegen regelungstechnischer Folgeprobleme der Rückführung, Verkoppelung bislang unabhängiger Teilsysteme, negativer Einfluß nicht hochgenauen Kartenmaterials, notwendiger Kenntnis der Kartengenauigkeit,...)

3.1.2.7 Positions-Stempelung von C2X-Nachrichten

Die Positionsdaten, welche aus dem eigenen Fahrzeug über C2X-Nachrichten an andere Verkehrsteilnehmer versendet werden (insbesondere die CAM-Nachrichten), werden zusammen mit dem Zeitpunkt, für den diese Position gültig ist, als Positionstempel bezeichnet. C2X-Nachrichten werden mit der aus der „Besseren Ortung“ stammenden, nicht mapgemachten Position gestempelt, da ein Matchen mit der Karte des Senders für den Empfänger nicht zu kontrollieren und eindeutig nachzuvollziehen ist. Bei Bedarf muss also ein Empfänger empfangene Positionsdaten über die eigene Karte matchen. Auch wenn im sim^{TD}-Testfeld selbst alle Fahrzeuge voraussichtlich die gleiche Karte verwenden, soll eine grundsätzliche Bindung der Systemarchitektur an eine spezifische Karte bzw. einen Kartenhersteller bewusst vermieden werden.

3.1.2.8 Zentrale Auswertung von Positionsdaten

Eine generelle Veredelung aus dem Feld gelieferter Positionsdaten in der Versuchszentrale erscheint in Anbetracht der Verwendung von DGPS nicht erforderlich. Im Bedarfsfall existieren Verfahren, um offline unter Ausnutzung der a-posteriori Kenntnis der Satelliten-Korrektursignale durch Nachfilterung zu verbesserten Genauigkeiten zu kommen. Dies setzt allerdings das Mitloggen der „rohen“ Signale des GPS-Receiver voraus, welche voraussichtlich im vollen Receivertakt nur für Debuggingzwecke der Komponenten durchgeführt werden wird. Eine Nachfilterung würde am besten unter Verwendung aller relevanten onboard protokollierten Daten erfolgen. Bei Bedarf muss auch eine Empfängeranwendung in der Versuchszentrale über die eigene Karte ein Mapmatching

vollziehen. Wird die gleiche Kartenbasis verwendet wie in der sim^{TD}-Versuchsflotte, können ggf. die gleichen Basisalgorithmen, welche onboard verwendet werden, auch für ein offline Mapmatching in der Versuchszentrale genutzt werden. Die Umsetzung entsprechender Funktionen in der Versuchszentrale ist dabei nicht mehr Bestandteil der Onboard-Komponenten „Bessere Ortung“ und „Digitale Karte“.

3.1.2.9 OEM- und fahrzeugspezifische Parametrierung

Das Positionsfilter muss für jedes in sim^{TD} verwendete Fahrzeugmodell spezifisch parametriert werden, um optimale Genauigkeit zu erzielen. Die „Bessere Ortung“ stellt die Parametrisierungsmechanismen bereit sowie eine Beispiel-Parametrisierung über die VAPI. Eine OEM- bzw. fahrzeugmodellspezifische Parametrisierung ist nicht Inhalt des Moduls bessere Ortung und muss vom jeweiligen Hersteller selbst geliefert werden. Der dazu notwendige Mechanismus zur Konfiguration fahrzeugspezifischer Parameter ist nicht Bestandteil der Komponente „Bessere Ortung“, sondern es wird ein funktionsübergreifender Mechanismus genutzt.

3.1.2.10 Ausgangsdaten und deren Nutzung

Die Ausgangsdaten der „Besseren Ortung“ können von allen Komponenten und Funktionen bezogen werden, die eine Eigenposition in Geokoordinaten (Länge und Breite, Höhe) benötigen. Dies betrifft auch die Positionsstempelung der Car2X-Protokolle (siehe Abschnitt 3.1.2.7). Sie ersetzen damit funktionell die Ausgangsdaten des GPS-Receiver und des zugehörigen GPS-Moduls, dessen Ausgangsdaten ausschließlich von der „Besseren Ortung“ als Eingangsdaten zur Positionsverbesserung genutzt werden. Die Höhenkoordinate wird wie vom GPS-Receiver bezogen ohne Filterung weitergereicht, bei Ausfall des GPS-Signals wird die letzte erhaltene Höhenkoordinate beibehalten und weitergereicht.

Die Ausgangsdaten werden insbesondere auch von der Komponente „Navi/Karte“ bezogen und dort für das Mapmatching der aktuellen Position auf das Straßennetz verwendet (siehe Abschnitt 3.1.2.4). Durch das Mapmatching auf eine aktuell befahrene Straße kann vor allem im Falle vergleichsweise geringerer Positionsgenauigkeit der „Besseren Ortung“ (z.B. bei längerer Satellitenverdeckung) auch eine weiter verbesserte Position entstehen. Diese gematchte Position wird von „Navi/Karte“ ebenfalls als Ausgangsdaten angeboten.

Funktionen und Komponenten können auch parallel beide Positionen beziehen, da die Entscheidung, welches Signal für eine bestimmte Aufgabe unter den gegebenen Umständen zu besseren Ergebnissen führt, applikations- und situationsabhängig sein kann.

3.1.3 C2C-Module

PHY und MAC Layer des Dual Receiver IEEE-802.11p-Moduls sind ausführlich in Deliverable D21.3 beschrieben. Hier wird nur auf die Verarbeitung der von dem IEEE-802.11p-Modul stammenden Nachrichten bzw. der dafür bestimmten Nachrichten beschrieben. Die detaillierte Spezifikation der Netzwerk und Facility Layer Module sind in den Dokumenten „Deliverable NEC-M1 Functional Description and Basic SIM-NET Specification“ und „Spezifikation des C2X – Software-Stacks (von Cirquent bearbeitete Komponenten)“ zu finden. An dieser Stelle wird nur auf architektonische Aspekte eingegangen. Folgende UML-Diagramme geben einen Überblick über dieses Subsystem.

3.1.3.1 Management & Facilities auf der CCU

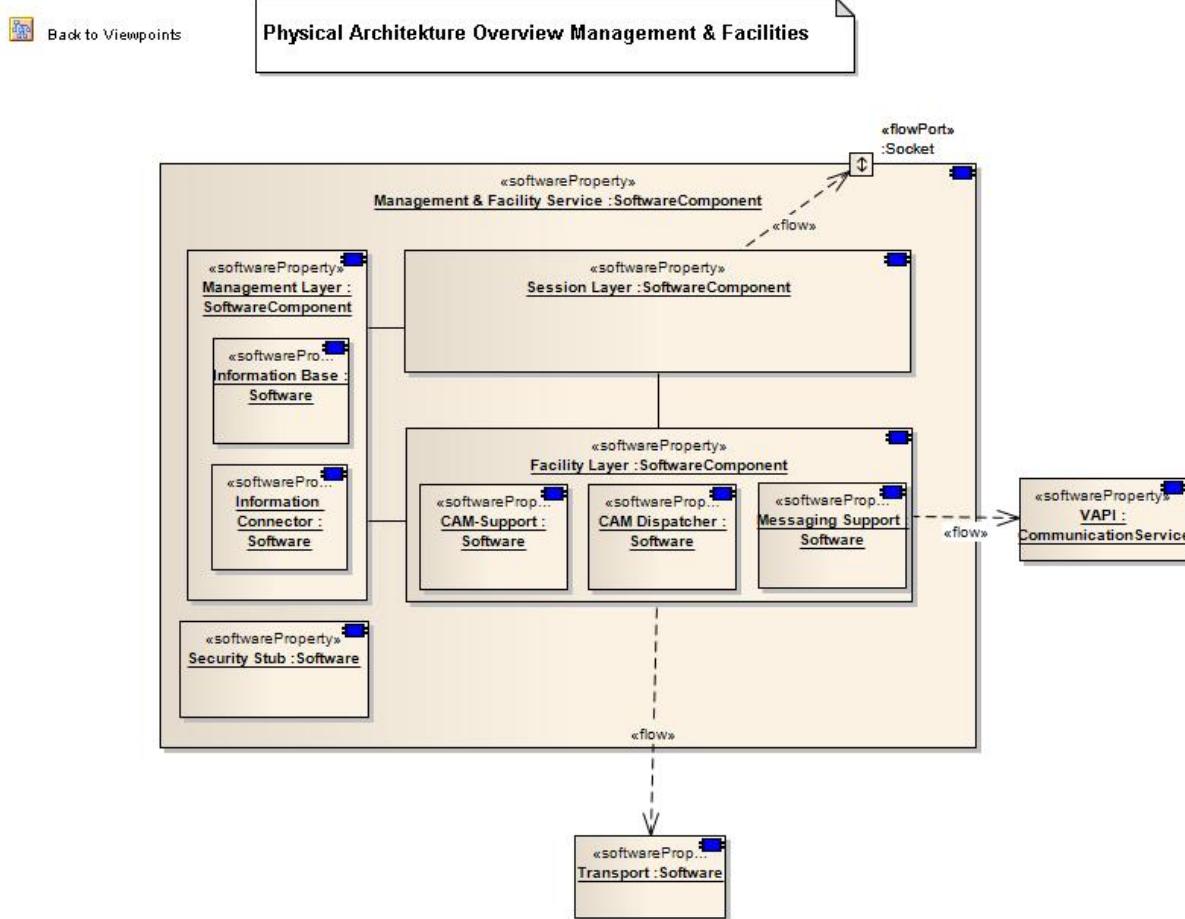


Abbildung 3-4: Management und Facilities

Session Layer

Aufgabe des Session-Layers ist es, die Kommunikation des Kommunikationsstack-Prozesses auf der CCU mit anderen Prozessen abzuwickeln. Die externen Prozesse können auf der CCU oder auf anderen Rechnern (hier die AU) laufen.

Facility Layer

Der Facility-Layer fasst unterstützende Funktionalitäten zusammen, die auf der CCU auf dem Weg vom Session-Layer zum Transport/Network-Layer liegen sollen.

Hier sind als Komponenten angesiedelt:

- CAM-Support: Der CAM-Support ist die Komponente, welche für die Erstellung von Cooperative Awareness Messages an einer zentralen Stelle im Gesamtsystem verantwortlich ist. Zu der Erstellung gehört sowohl die Codierung als auch die Decodierung der Nachrichten in das definierte Netzwerkformat. Durch dieses zentrale Modul soll zum einen gewährleistet werden, dass dieser häufig genutzte Nachrichtentyp immer in einem einheitlichen Format versendet wird. Zum anderen sollen Anwendungen entlastet werden, indem sie nicht in redundanter Weise für die Codierung der Nachrichten verantwortlich sind.

- CAM-Dispatcher: Der CAM-Dispatcher übernimmt einen Teilaspekt bei der Congestion Control. Hierbei werden einzelne CAM-Nachrichten mit unter Umständen unterschiedlichen Signalbelegungen versendet. Die Regelung der Sendefrequenz kann im Zusammenspiel mit dem Information Connector bestimmt werden. Werden keine weiteren Parameter übergeben, so wird eine Standard-CAM-Nachricht versendet, wobei nur ausgewählte Felder mit Werten belegt werden. Hierbei werden die Signalwerte aus der VAPI-Schnittstelle bezogen und die Nachricht damit angereichert.
- Messaging-Support: Durch die Komponente Messaging-Support wird jede Nachricht, die empfangen wurde, und jede Nachricht, die versendet werden soll, durchgereicht. Es können durch spezialisierte Implementierungen dieser Komponente zusätzliche Verarbeitungsschritte eingefügt werden, die dann an den vorbeikommenden Nachrichten ausgeführt werden können.

Management-Layer

Der Management-Layer versammelt diejenigen Funktionalitäten in sich, die nicht auf dem Weg der empfangenen oder zu versendenden C2X-Nachrichten liegen und die allen Layern des Kommunikationsstacks zur Verfügung stehen sollen. Er besteht aus folgenden Komponenten:

- Information-Base: Die Information-Base ist ein hierarchisch organisierter Datenspeicher für Konfigurations- und Betriebsdaten. Sie speichert die Konfigurationsdaten für die Initialisierung des gesamten CCU-seitigen Kommunikationsstacks und für einzelne Komponenten.
- Information Connector: Der Information Connector ist zuständig für den Austausch von Steuerungsinformationen innerhalb des Fahrzeugs. Das betrifft sowohl den Informationsaustausch zwischen Modulen des Kommunikationssystems, als auch zwischen Kommunikationssystem und Anwendungen über das Client-API. Hierbei übernimmt der Information Connector die Aufgabe, so genannte Event-Messages zu verarbeiten und diese an andere Module oder Protokollinstanzen zu verteilen.
- Security-Stub: In dieser Komponente erfolgt die En/Decryption sowie die Signierung bzw. Validierung der C2X-Network-Nachrichten.

3.1.3.2 C2X Communication Component – SIM-NET (in English)

 Back to Viewpoints

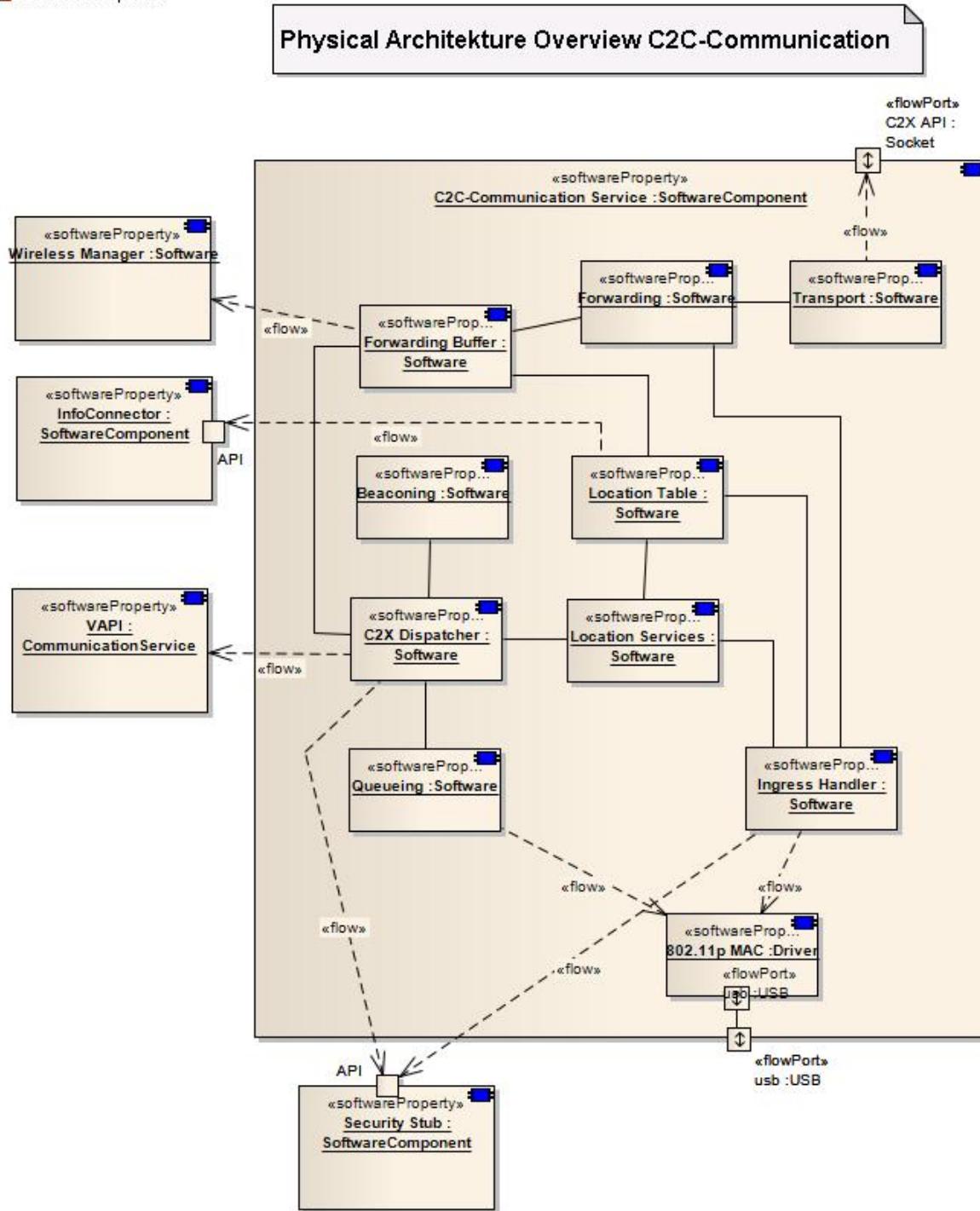


Abbildung 3-5: C2X-Communication Subsystem – SIM-NET

The SIM-NET component provides the system with the safety communication framework. It is composed by the following sub-components:

- Transport: Transport handles the multiplexing and de-multiplexing of communication of applications. Besides handling “application IDs” (or *ports*) and transport-level network header framing, it also interfaces with the local policy store to apply traffic class based constraints and guarantees to application generated messages, including messages not individually marked by applications. It exposes a “C2X API” interface, which is made available to applications running in the Application Unit by the facility layer.
- Forwarding: This component encompasses the multi-hop and geo-broadcasting forwarding protocols and mechanisms. It determines how to dispatch a message; for example: if it should be broadcast locally by the CCU or if this message should be forwarded to an intermediate node towards the reception area. It is also responsible to determine if messages that are received locally and will be handled locally (i.e. will be passed to Transport) should be re-broadcast, either for reliability or for safety purposes. Vice versa, if a message that was forwarded should be handled and buffered locally. Forwarding decisions also take traffic class based constraints into consideration.
 - Forwarding Buffer: The Forwarding Buffer handles the buffering of packets for future transmission. Buffering is triggered by forwarding and its policies: time-based constraints (and congestion-based), no neighbour-based constraints, etc. Additionally, the Forwarding Buffer takes the messages’ properties, including their length, requested traffic class, and others, to optimize the transmission of messages in an efficient way.
 - Location Table: This component provides the system with the node table: a comprehension of information about both nearby nodes and nodes with no direct communication but are involved in the current communication otherwise. Information maintained by the node table is updated by incoming messages from other nodes, which include their identifiers (who they are) and their location and heading (where they are, where they will be). This information is then used by geographical-based forwarding protocols to provide both multi-hop unicast communication and geo-casting (broadcast to a specified area).
 - Beacons: The Beacons component monitors current communication to establish if enough messages are being dispatched to allow neighbouring nodes to determine the node’s location and presence accurately. If the message rate drops below the pre-established minimum, and based on whether the node estimates the medium to be congested or not, dummy payload messages are dispatched (i.e. “beacon” messages) to broadcast the node’s presence.
 - Location Services: Location Services implement a location discovery mechanism used for unicast communication. When a node requests a message to be dispatched to another node via unicast transmission, the Forwarding component must first determine where this destination node is in order to be able to dispatch the message to it. This service is provided by this component.
 - C2X Dispatcher: This component provides C2X communication protocol framing, including the support for “extension” headers such as those supplied by the security capability. Before being dispatched, messages within the system are composed by the application payload and additional options supplied by both the application and the Transport component. As messages

dispatched by the system have temporal information, they are only assembled when they are determined to be sent.

- Queuing: A low level queuing component that provides simple priority based queuing and rate control.
- Ingress Handler: The Ingress Handler provides the communication system with an early check on incoming messages, to discard both badly formed and forged ones as soon as possible. Additionally, it interfaces with the security capability of the system to run messages through plausibility checks to further mitigate possible attacks. After messages are cleared, they are passed to either the Forwarding module or one of the local services, i.e. Location Services, to be handled.

3.1.4 2G/3G/Consumer-WLAN

Das detaillierte Kommunikationskonzept der 2G/3G-Nutzung ist in Deliverable D21.3 beschrieben. Ebenso sind dort die Details der Consumer-WLAN-Funktion zu finden. Allgemein werden beide Technologien dazu benutzt, um über TCP/IP bzw. UDP/IP Daten mit anderen Fahrzeugen und/oder Infrastruktursystemen auszutauschen. Infrastruktursysteme sind hierbei sowohl IRS als auch Server-Systeme. Die eigentlichen Kommunikationssysteme werden bis in den Netzwerk-Layer hinein nicht verändert, Die Nutzdaten hingegen können entweder C2X-Nachrichten sein oder direkt Applikationsdaten, je nach Anwendung. C2X-Nachrichten werden wie C2X-Nachrichten, die über einen IST-G5A-Kanal empfangen wurden, über die SIM-NET Facility-Layer-Kommunikationskomponenten an die Funktionen geliefert. Applikationsdaten werden direkt an die jeweilige Applikation geroutet. Folgendes UML-Diagramm zeigt die mit der Verarbeitung der 2G/3G- und Consumer-WLAN-Daten betrauten Komponenten.

[Back to Viewpoints](#)

Physical Architecture Overview WLAN-Communication Service

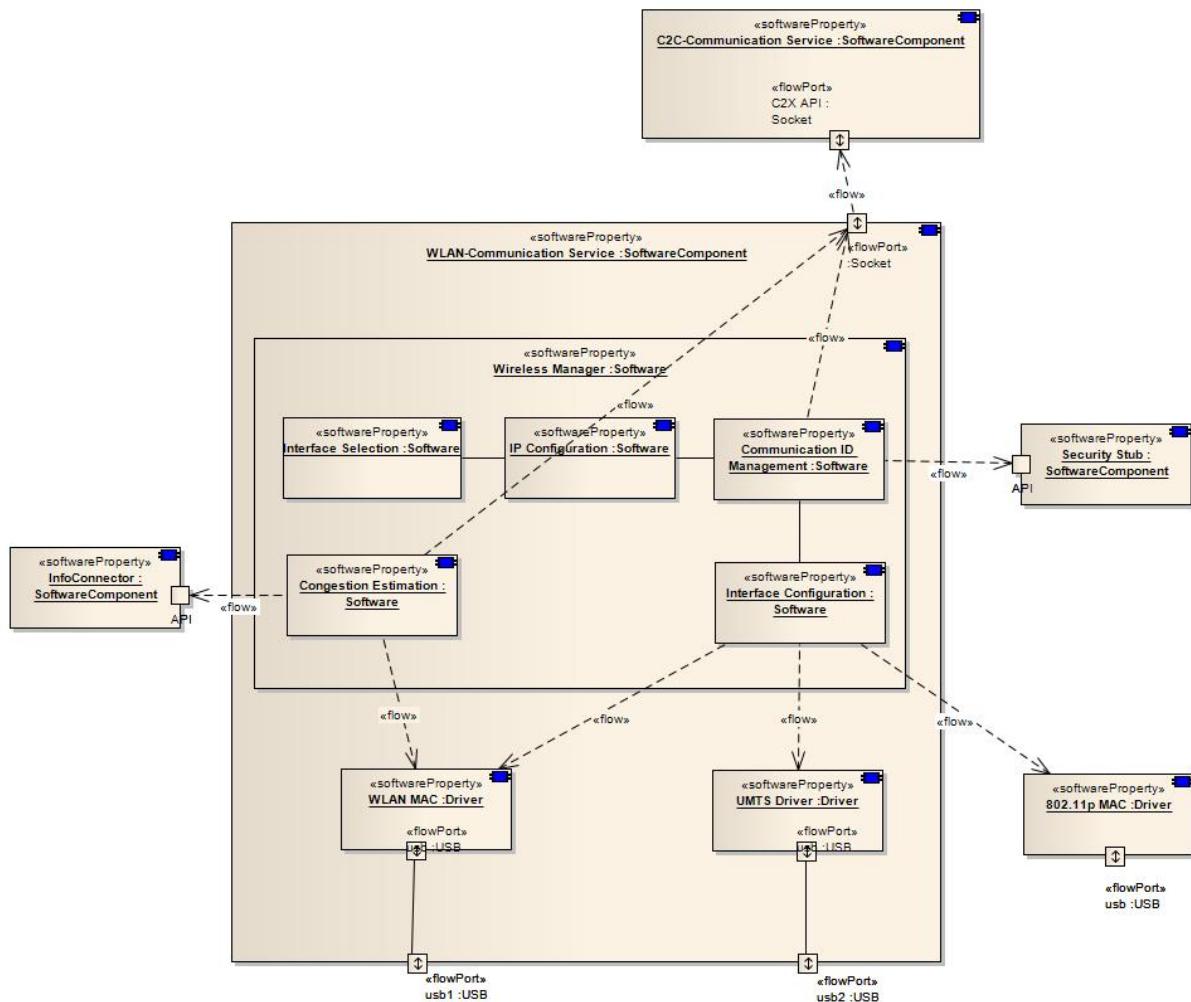


Abbildung 3-6: CCU – UMTS- und Consumer-WLAN-Subsystem

3.1.4.1 Wireless Manager (in English)

The Wireless Manager provides the system with the vertical management facilities required by other parts: IP configuration, interface selection, congestion estimation, etc. It is composed by the following sub-components:

- Communication ID management: Interfaces with the security capability to re-configure the identifiers used by the system when a pseudonym change is triggered.
- IP configuration: This component provides IP configuration support to the system. It will both support local IP configuration at the router, including automatic configuration of IP addresses and prefixes, and interfacing with other control daemons, including mobility based ones. It will also support the automatic configuration of IP at the Application Unit, in order to support the specified IP deployment solution.

- Congestion Estimation: Congestion estimation takes input from several other components, such as the C2C communication stack (SIM-NET) itself, lower layer information made available by the MAC, to build estimations on the level of congestion visible in the communication medium. This information is then made available to other parts of the system by the cross-layer information exchange: the Information Connector.
- Interface Selection: A local policy enforcement point: It bases its decisions on local information interfaces with IP Configuration to instruct the usage of a specific technology.
- Interface Configuration: Low level interface configuration and activation, including the configuration of MAC layer identifiers, enabling and disabling of network interfaces, power control and channel switching, etc.

3.1.5 Fahrzeug-Datenzugriff (VAPI)

Bei der Entwicklung prototypischer Anwendungen für Flotten verschiedenartiger Fahrzeuge ist es notwendig, einheitlich auf Sensordaten des Fahrzeug-Bordnetzes (Geschwindigkeit, Querbeschleunigung etc.) zuzugreifen.

Zwar hat sich der CAN-Bus im Automobilbereich als Standard durchgesetzt, jedoch nur auf physikalischer Ebene, da die CAN-Botschaften nicht herstellerübergreifend standardisiert sind. Zudem werden die Zuordnungen ständig an Bedürfnisse neuer Modelle angepasst und dabei ergänzt oder komplett neu definiert. Der Umfang der zugreifbaren Botschaften variiert dabei ausstattungs- und modellspezifisch.

3.1.5.1 VAPI Server

Die Vehicle API (kurz: VAPI) ist eine Anwendung, die Fahrzeugdaten beliebigen Anwendungen zur Verfügung stellt, und dabei die Datenquelle (Bussystem) und die herstellerspezifischen Protokolle abstrahiert. Den Anwendungen wird so ein generischer Zugriff auf Fahrzeugdaten ermöglicht wobei die fahrzeugspezifische Architektur (Verwendung von CAN-Bus, Datenrepräsentation etc.) verborgen wird. Beispiele für Fahrzeugdaten sind z.B. die Fahrzeuggeschwindigkeit und Fahrgestellnummer.

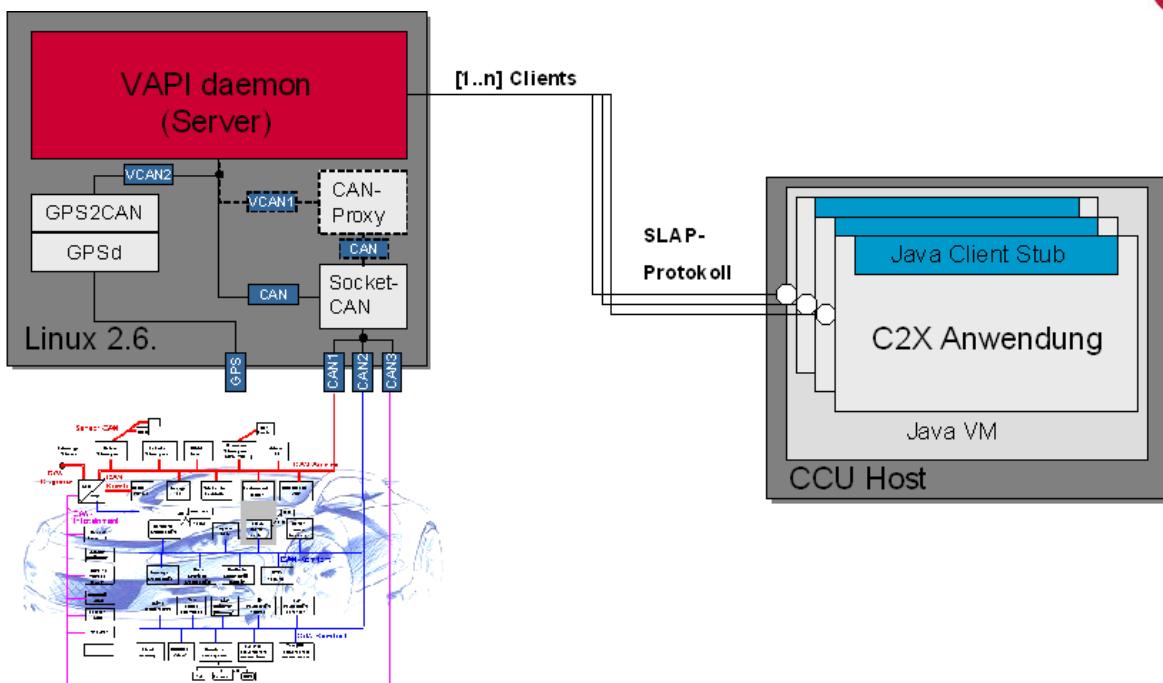


Abbildung 3-7: Konzept des Fahrzeugzugriffs

Abbildung 3-7 zeigt einen allgemeinen Überblick über die Funktionsweise des Fahrzeugzugriffs. Neben Fahrzeugdaten werden in sim^{TD} GPS-Daten über die VAPI an die Anwendungen (Clients) verteilt, um eine synchrone Aufzeichnung von Fahrzeugdaten und Positionsdaten zu ermöglichen.

Der VAPI-Server ermöglicht mehreren Anwendungen den parallelen Zugriff auf Fahrzeugdaten und GPS-Daten. Dazu stellt diese Funktionen zur Verfügung, um Datenobjektänderungen zu abonnieren (Publish/Subscribe) oder diese abrufen (Request/Response) zu können.

Viele Sensor-Informationen werden mit einer hohen Frequenz (z.B. alle 10ms) auf dem CAN-Bus ausgetauscht. Da die Mehrzahl der Anwendungen die auf dem CAN zur Verfügung gestellten Informationen nicht mit dieser hohen zeitlichen Auflösung benötigen, besteht bei der Abonnement von Fahrzeugdaten die Möglichkeit, die Update-Rate für ankommende Daten zu drosseln. Es stehen zwei Möglichkeiten der Abonnement zur Verfügung:

- Anwendungen können ein maximales Intervall definieren, um die Anzahl der Aktualisierungen zu reduzieren.
- Anwendungen können eine Auflösung definieren. Somit wird ein Datenobjekt erst dann an die Anwendung weitergereicht, wenn sich dieses um einen definierten Wert von dem übermittelten Vorgängerwert unterscheidet.

Die folgenden beiden Beispiele sollen einen sinnvollen Umgang mit der Funktion erläutern.

Beispiel 1: Die aktuelle Fahrzeuggeschwindigkeit wird beispielsweise bei Volkswagen in einem Abstand von 10 ms auf dem CAN übertragen. Diese hohe Aktualisierungsrate ist für Anwendungen, die aktuelle Informationen über den Fahrzeugzustand benötigen, typischerweise nicht notwendig da sich die Fahrzeuggeschwindigkeit lediglich in gewissen Grenzen ändern kann. Zudem müsste eine Anwendung in der Lage sein, die entsprechenden Informationen alle 10 ms aus dem XML-Datenstrom zu dekodieren und zu Verarbeiten, was eine entsprechende leistungsfähige CPU voraussetzt. Eine Datendrosselung

auf 200 ms oder 500 ms entlastet somit die Anwendung (z.B. bei der Visualisierung der Geschwindigkeit) ohne wichtige Informationen zu verlieren.

Beispiel 2: Informationen über das Eingreifen des ESP oder das Setzen des Blinkers befinden sich nur zum Zeitpunkt der Aktion auf dem Fahrzeugbus. Um keine Informationen zu verlieren, sollte daher für solche Signale keine Drosselung vorgenommen werden. Die Anwendung wird so sicher beim Eingreifen des ESP informiert. In der übrigen Zeit erfolgt keine Benachrichtigung der Anwendung da sich der Zustand der Systeme nicht ändert und nur inhaltliche Updates von der VAPI übertragen werden.

Profiles

Um die verschiedenen Buskonfigurationen (CAN-IDs, Busgeschwindigkeit usw.) der OEMs verarbeiten zu können, ist es notwendig, für jedes Fahrzeug ein Profil zu definieren.

Das Profil beinhaltet Regeln, die zur Abbildung des Objektkatalogs zu physikalischen Fahrzeugdaten benötigt werden. Zur Abbildung der Daten wird zum einen die Beschreibung der Datenrepräsentation auf dem CAN-Bus (z.B. aus Vector Datenbasen *.dbc) benötigt und zum anderen ein Objektkatalog, der definiert, welche Daten (Objekte) von der abstrakten Schnittstelle angeboten werden. Die Erzeugung des VAPI-Servers ist in Abbildung 3-8 dargestellt.

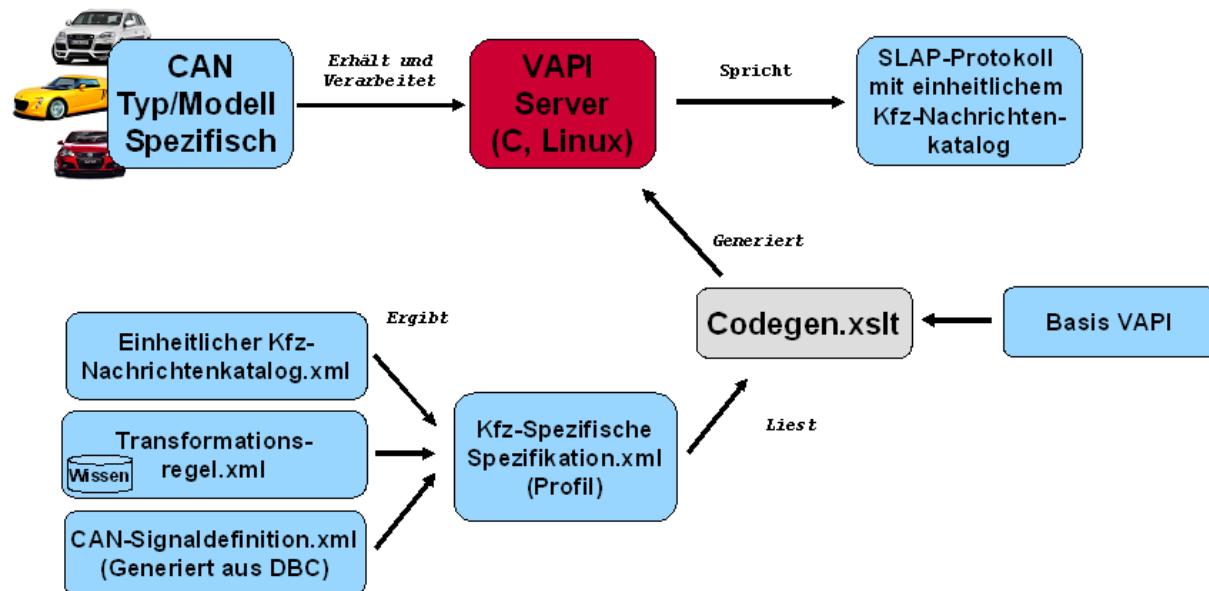


Abbildung 3-8: Generierung der VAPI

Die Erzeugung eines Profils wird durch einen Profilgenerator durchgeführt, der für die OEM Partner in sim^{TD} bereitgestellt wird. Dieser erzeugt mit Hilfe von Vector Datenbasen und dem durch Volkswagen bereitgestellten Objektkatalog ein fahrzeugspezifisches Profil. Um den Schutz der Profildaten auch im Feldtest zu gewährleisten, wird von der VAPI auf Grundlage des Profils Quellcode generiert, so dass die Software die Informationen der Datenrepräsentation im übersetzten Quellcode enthält. Da die Generierung der VAPI den Quellcode erfordert, kann die VAPI ausschließlich durch Volkswagen oder einem gemeinsamen Unterauftragnehmer vorgenommen werden. Dieses Vorgehen ist für einige Projektpartner problematisch, da keine Informationen über die Datenrepräsentation des CANs herausgegeben werden können und somit die Erzeugung der VAPI nicht durchgeführt werden kann.

3.1.5.2 CAN-Proxy

Um die VAPI dennoch für alle Projektpartner zur Verfügung zu stellen, wird eine Option ermöglicht, einen CAN-Proxy zu verwenden. Der CAN-Proxy kapselt die Information der Datenrepräsentation des CANs, indem die fahrzeug- und herstellerspezifischen CAN-Nachrichten gelesen und in ein für sim^{TD} vereinheitlichtes CAN-Nachrichten-Format überführt werden. Das sim^{TD}-CAN-Nachrichtenformat wird speziell für die Erfordernisse von sim^{TD} entworfen und erlaubt somit keine Rückschlüsse auf die Datenrepräsentation auf dem realen Fahrzeugbus. Weiterhin werden der VAPI mithilfe des CAN-Proxys statische Fahrzeugdaten und Fahrzeugparameter bereitgestellt, die möglicherweise nicht auf dem Fahrzeugbus verfügbar sind (z.B. Spurbreite, FahrzeugID, Fahrzeulgänge, Status Einsatzfahrzeug usw.).

Abbildung 3-7 zeigt in gestrichelter Umrandung die optionale CAN-Proxy-Komponente. Die in das sim^{TD}-Format überführten Nachrichten werden vom CAN-Proxy auf ein virtuelles CAN-Interface geschrieben und von der VAPI verarbeitet. Die VAPI ist für Fahrzeuge mit einer Zwischenschicht immer identisch und wird als einheitliche Schnittstelle für alle Funktionen der Vehicle AU beibehalten.

Eine Beispielimplementierung des CAN-Proxys, die CAN-Datenbase (Vector Datenbasen *.dbc) sowie eine für das sim^{TD} Format angepasste VAPI ist auf dem SVN-Server unter <https://svn.simtd.de/svn/simTD/TP2/AP22/VCCU/Lieferungen/System/VAPI> zur Verfügung gestellt.

3.1.5.3 LLCF

Die Funktionen des Low Level CAN Frameworks (LLCF) werden von der VAPI verwendet und erlauben Anwendungen den einfachen Zugriff auf die Kommunikationsschichten des CAN-Busses. Wesentliche Komponenten des LLCF sind die Netzwerk-Treiber für verschiedenen CAN-Controller und die darüberliegenden Protokolle wie TP1.6, TP2.0, MCNet, ISO-TP, etc. Diese Komponenten sind im Linux-Kernel implementiert und werden über eine definierte Socket-Schnittstelle angesprochen. Durch die Realisierung der verschiedenen höheren CAN-Protokolle als Kernelmodule können zeitliche Randbedingungen im Kernel-Kontext eingehalten werden, die auf der Anwenderschicht in dieser Form nicht realisierbar wären. Dieses ist beispielsweise für CAN-Transportprotokolle relevant, die Antwortzeiten im Bereich von Millisekunden definieren. Sollen auf einem System parallel mehrere CAN-Anwendungen zur Ausführung kommen, kann jede Anwendung unabhängig beliebig viele CAN-Sockets (auch verschiedener Socket-Typen auf verschiedenen CAN-Bussen) öffnen.

3.1.5.4 CAN-Schnittstelle

Die CCU stellt drei CAN-Ports nach dem Standard CAN2.0a/b zur Verfügung, davon sind Port 1 und 2 als reine Readonly-Ports ausgeführt, um sicherzustellen das selbst bei einer Fehlfunktion der CCU keine fehlerhaften Daten auf den Fahrzeug-CAN geschrieben werden. Diese Sicherung wird durch eine entsprechende Ausführung der Hardware erzielt, die Sicherheit der Versuchsfahrzeuge ist entsprechend gewährleistet.

CAN-Port 3 ist ein CAN2.0a/b-kompatibler CAN-Bus, der bidirektional ausgeführt ist. Dieser Bus ist dafür gedacht, die Verbindung zu zusätzlichen automotive Steuergeräten, wie z.B. einem LIDAR Scanner, aufzubauen. Dieser Bus muss aus Sicherheitsgründen komplett von allen anderen Fahrzeugbussen getrennt sein.

Port 1 und 3 sind als highspeed CAN-Busse ausgeführt, Port 2 ist als single-wire lowspeed CAN-Bus ausgeführt.

Die Daten aller CAN-Busse werden über das CAN-Socket-Interface (LLCF) an die VAPI weitergeleitet.

3.1.6 CCU Security

Mobilitätsdaten werden durch SIM-NET auf Vermittlungsebene in das Kommunikationsprotokoll integriert. Da die Mobilitätsdaten wie auch die Anwendungsdaten geschützt werden müssen, ist die Platzierung der Kommunikationsabsicherung durch den Security Daemon auf der CCU notwendig.

Alle eingehenden Nachrichten werden nach dem Empfang als erstes an den Security Daemon geleitet, um die Nachricht zu verifizieren oder zu entschlüsseln. In sim^{TD} muss je nach vorhandener CPU-Leistung und eingesetztem kryptografischen Algorithmus eine Strategie angewendet werden, die die Datenintegrität schützt. Für den Fall, dass Ressourcen für eine Prüfung aller eingehenden Nachrichten nicht vorhanden sind, können probabilistische Verifikationen durchgeführt werden oder es werden nur Zertifikate von unbekannten Knoten validiert. Diese Maßnahmen sind jedoch nur im absoluten Notfall anzuwenden, da somit gefälschte Nachrichten unbemerkt in das sim^{TD}-Netz eingespielt werden können.

Abbildung 3-9 zeigt die Positionierung der Kommunikationsabsicherung auf der Vehicle CCU. Die Nachrichten werden nun von der Komponente „C2C Communication“ und von der WLAN-Communication-Komponente auf der Vehicle CCU empfangen und zur Prüfung an die Kommunikationsabsicherung gegeben. Das Verifikationsergebnis bzw. der entschlüsselte Payload wird anschließend zur Vehicle AU zum „Communication Client“ geleitet.

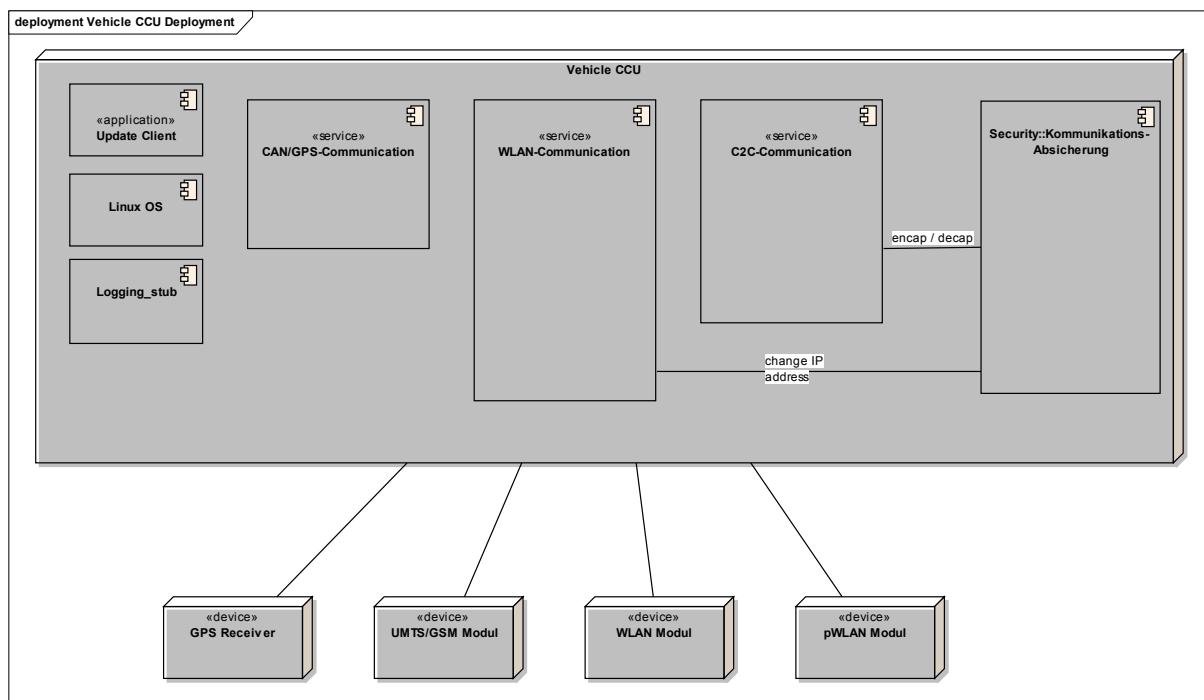


Abbildung 3-9: Positionierung der Kommunikationsabsicherung durch den Security Daemon auf der CCU

Beim Versand werden C2X-Nachrichten auf der Vehicle CCU von der C2C-Communication-Komponente an den Security Daemon gegeben, damit an dieser Stelle die gewünschte Absicherungsmaßnahme vorgenommen werden kann. Bis auf die Absicherung anwendungsspezifischer Nachrichten, beschrieben in Abschnitt 2.5.4.2, werden alle

Nachrichten und Kommunikationsverbindungen durch IT-Sicherheitskomponenten und Sicherheitsmaßnahmen direkt auf der CCU adressiert. Die Signierung und Verifikation von anwendungsspezifischen Nachrichten wird auch durch den Security Daemon auf der CCU durchgeführt. Dieser wird jedoch vom Security Daemon Client der AU angesprochen.

Eine detaillierte Beschreibung des Sicherheitsdienstes für die Ad hoc ITS Kommunikation wird in Deliverable D21.5 in Abschnitt 5.3 zur Verfügung gestellt.

Die Aufgaben des Security Daemons der CCU können wie folgt zusammengefasst werden:

- Signieren von Nachrichten
- Verifizieren von Nachrichtensignaturen und den zugehörigen Zertifikaten
- Verschlüsselung von Nachrichten
- Entschlüsselung von Nachrichten
- Bereitstellen und Verwalten der eigenen Basisidentität und Pseudonyme
- Bereitstellen und Verwalten der Pseudonyme (öffentliche Schlüssel) benachbarter Teilnehmer, die für eine Verschlüsselung benötigt werden.
- Pseudonyme wechseln

3.1.7 Router-API

C2X-Nachrichten werden über den Facility Layer versendet und empfangen. Internetverbindungen werden über Standard-Socket-Interface aufgebaut. Fahrzeug- und Positionsdaten werden über die VAPI an die AU geliefert.

Weitere APIs sind nicht vorgesehen.

3.2 Vehicle Application Unit

Im folgenden Deployment-Diagramm sind alle vorgesehenen Abhängigkeiten zwischen AU und anderen Hardware-Komponenten dargestellt. Die Kommunikation erfolgt über TCP/IP-Verbindungen, die jeweils direkt von Komponenten auf Router, Host bzw. HMI-Device geöffnet werden können. Bestimmten Komponenten und dem zugehörigen Service werden dazu bestimmte Ports zugewiesen. In der Regel greifen Applikationen nicht direkt auf die Sockets zu, sondern über die zugehörigen Systemkomponenten. Optional kann OEM-spezifische Hardware ebenfalls über TCP/IP angebunden werden.

Diese Verbindungen werden ausschließlich verwendet, um die dargestellten Zugriffe zu ermöglichen: C2X-Kommunikation, Abfrage von Fahrzeugdaten, Entgegennehmen von Log-Messages des Routers, Nutzen von TCP/IP-Verbindungen zu externen Servern sowie Kommunikation mit dem PDA zur Darstellung des User Interface auf dem PDA und ggf. Datenaustausch mit einem OEM-spezifischen Steuergerät.

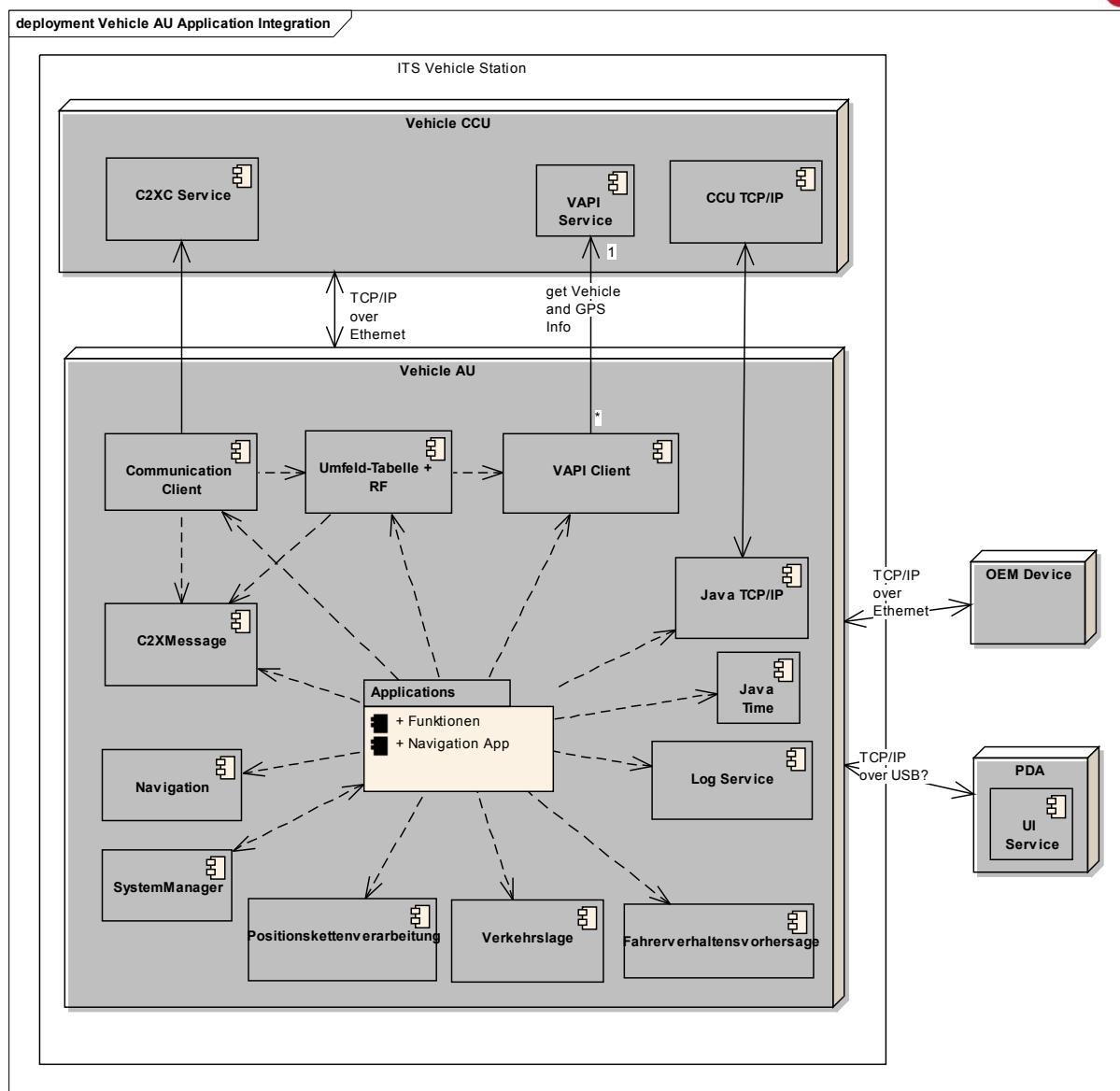


Abbildung 3-10: sim^{TD} Deployment

Das Diagramm zeigt außerdem alle Zugriffsmöglichkeiten der Applikationen, die auf der AU betrieben werden, auf Systemkomponenten. Die Daten der Umfeldtabelle, Fahrzeugdaten und Ortungsinformationen können ersatzweise von einem Trace Player in das System eingebracht werden, diese Unterscheidung ist aber für die Applikationen nicht sichtbar und hier nicht dargestellt. Neben dem OSGi-Log-Service ist noch ein spezielles Logging für die Versuchsauswertung vorgesehen, das hier ebenfalls nicht dargestellt ist. Diese Komponenten sind in Abschnitt 2.4 beschrieben. Außerdem sind Sicherheitsfunktionen vorgesehen, die für die Applikationen aber weitestgehend transparent realisiert werden und somit ebenfalls hier nicht beschrieben sind (vgl. Abschnitt 2.5).

Die Komponente SystemManager dient dazu, Konflikte beim Zugriff auf Systemressourcen aufzulösen. SystemManager informiert über die Auslastung kritischer Ressourcen, gibt bei Überlastsituationen Warnungen an die Funktionen aus und nimmt notfalls einzelne Funktionen außer Betrieb. Dabei greift SystemManager auf statische Informationen über die vordefinierten Prioritäten der Funktionen zurück, die entweder in der Komponente selbst definiert sind oder über den OSGi Config Admin Service für die Versuchssteuerung parametrierbar gemacht werden.

Eine komplexere Sicht auf das Zusammenspiel der Systemkomponenten ist im erweiterten Deploymentdiagramm nachfolgend dargestellt.

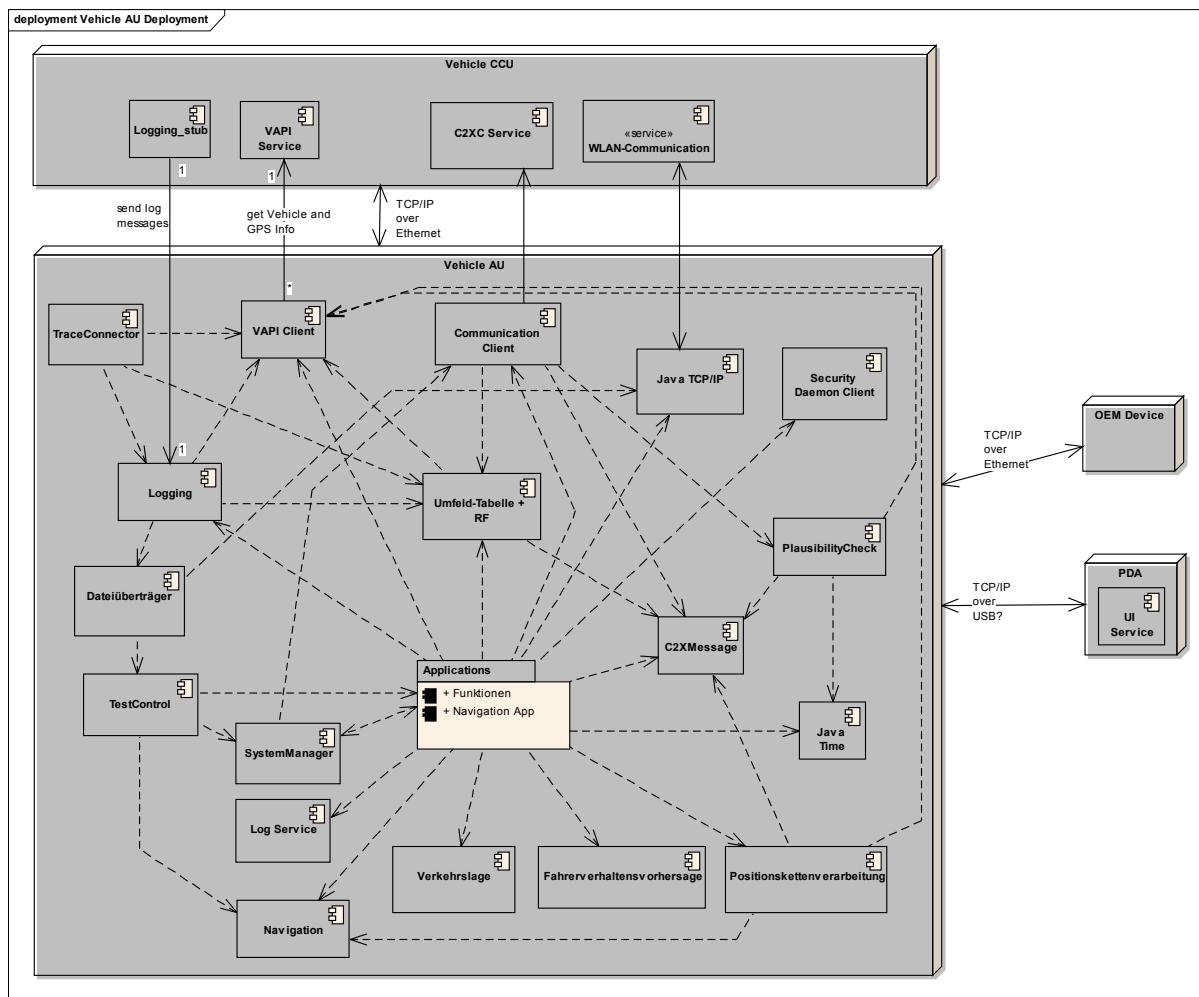


Abbildung 3-11: sim^{TD} Deployment (mit Zusammenhängen zwischen den Komponenten)

3.2.1 Application Framework

3.2.1.1 Plattform und Framework

Der Funktionshost basiert auf einem Car-PC (der noch ausgewählt wird) und einem Standard SW-Framework, bestehend aus

- Windows XP embedded
- Java-Laufzeitumgebung (voraussichtlich Sun JDK 1.6)
- OSGi-Framework von ProSyst (mBedded Server Professional Edition)
- Ergänzend zum OSGi Framework werden voraussichtlich zusätzliche "Telematik Erweiterungen" integriert.

- OSGi erleichtert es, Anwendungen und ihre Dienste über ein Komponentenmodell zu verwalten: Komponenten bestehen aus einem oder mehreren OSGi-Bundles. Ein Bundle veröffentlicht seine Schnittstellen (= "Dienste bzw. Services") per Registry (= "Service-Registry") und ist dann von anderen Bundles verwendbar. Bundles können zur Laufzeit eingespielt, aktualisiert und wieder entfernt werden. Ein OSGi-Bundle besteht in der Regel aus mehreren Klassen in mehreren Packages.

Die Implementierung aller Software-Komponenten auf dem Funktionshost erfolgt ausschließlich in Form von OSGi-Bundles, um die Robustheit, Wartbarkeit und Integrierbarkeit in sim^{TD} beherrschbar zu halten. Eine Ausnahme bilden Navigation und digitale Karte, die als kommerzielle Software nur in einer plattformspezifischen Implementierung verfügbar sind. Der Zugriff darauf erfolgt aber ebenfalls über ein OSGi-Bundle.

Eine Verwendung von JNI erfolgt nicht.

3.2.1.2 Komponentenklassifizierung

Die AU sieht zwei wesentliche Arten von Komponenten vor:

1. Hauptfunktionen

Hauptfunktionen enthalten die in TP1 festgelegten Funktionen. In der AU ist darunter genau genommen ein Teilsystem auf dem Funktionshost zu verstehen. Die enthaltenen Komponenten werden als Funktionskomponenten bezeichnet und implementieren die in TP1 spezifizierte Funktionalität. Sie können aus einem oder mehreren OSGi-Bundles bestehen. Pro Hauptfunktion wird ein geschlossenes, getestetes Softwarepaket, bestehend aus den OSGi-Bundles, vom jeweiligen Hauptfunktionsleiter an den Funktionshost-Integrator übergeben (s. TP2/AP22 Entwicklungshandbuch).

Hinweis: Der Funktionshost stellt die Plattform für die überwiegende Zahl von Hauptfunktionen dar. Funktionen mit zeitkritischen Anforderungen müssen auf dem Router umgesetzt werden!

2. Systemkomponenten

Hauptfunktionen greifen auf die Systemkomponenten zu. Diese stellen die für die Hauptfunktionen notwendige Funktionalität in Form von OSGi-Diensten bereit. Die Systemkomponenten können aus einem oder mehreren OSGi-Bundles bestehen.

Sie sind Projektpartnern zugeordnet und werden in Abbildung 3-12 dargestellt.

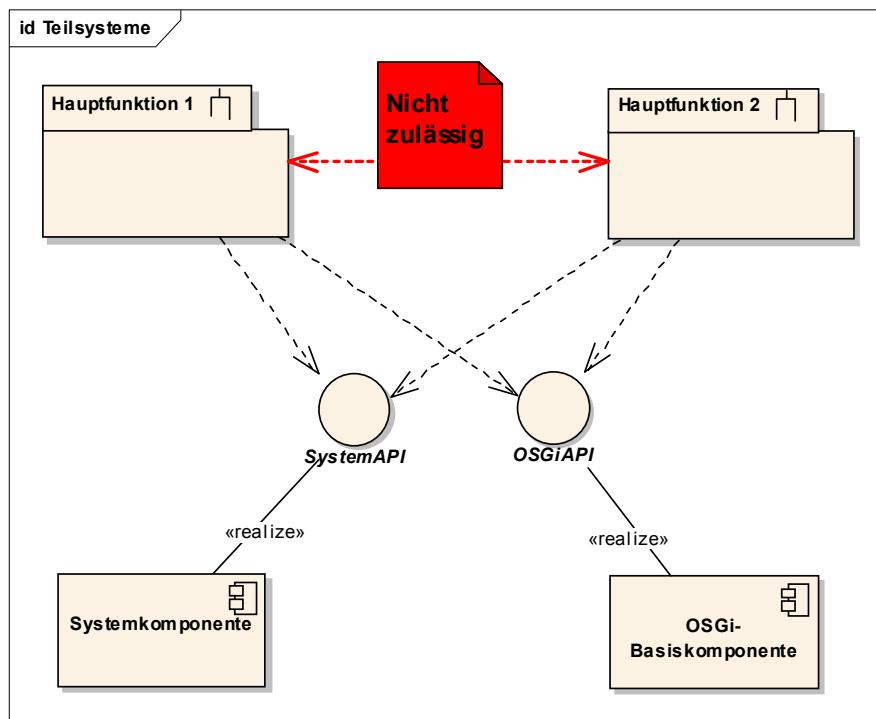


Abbildung 3-12: ID Teilsystem

Darüber hinaus werden über das OSGi-Framework von ProSyst weitere Dienste bereitgestellt (OSGi-Basiskomponenten). Dabei handelt es sich um

- OSGi-Dienste, die in der OSGi-Spezifikation definiert sind
- zusätzliche Erweiterungen von Dritten (z.B. von ProSyst)

Beispiel dafür sind: XML Service, Software Update Manager, Basisdienst Logging, Shutdown und Recovery, u.a. Die Umsetzung erfolgt durch ProSyst.

3.2.1.3 Zugriffsregeln bezüglich der Komponenten

- Funktionskomponenten dürfen auf alle Systemkomponenten und OSGi-Komponenten zugreifen.
- Funktionskomponenten dürfen innerhalb derselben Hauptfunktion auf andere Funktionskomponenten zugreifen.
- Funktionskomponenten dürfen nicht auf Funktionskomponenten anderer Hauptfunktionen zugreifen (in Abbildung 3-12 mit "nicht zulässig" markiert)!
- Funktionen mit zeitkritischen Anforderungen müssen auf dem Router umgesetzt werden!

3.2.2 System Manager

Die Komponente SystemManager ist dafür zuständig, das OSGi-System stabil zu betreiben. Dazu gehören zum einen das Verwalten aller Komponenten, die Überwachung der Systemauslastung und das Einleiten von Maßnahmen bei Überlast, zum anderen die

Verwaltung von allgemein verwendeten Konfigurationsparametern und Daten. Sie stellt auch die Systemzeit entsprechend der GPS-Zeit, die sie von der CCU über NTP erhält.

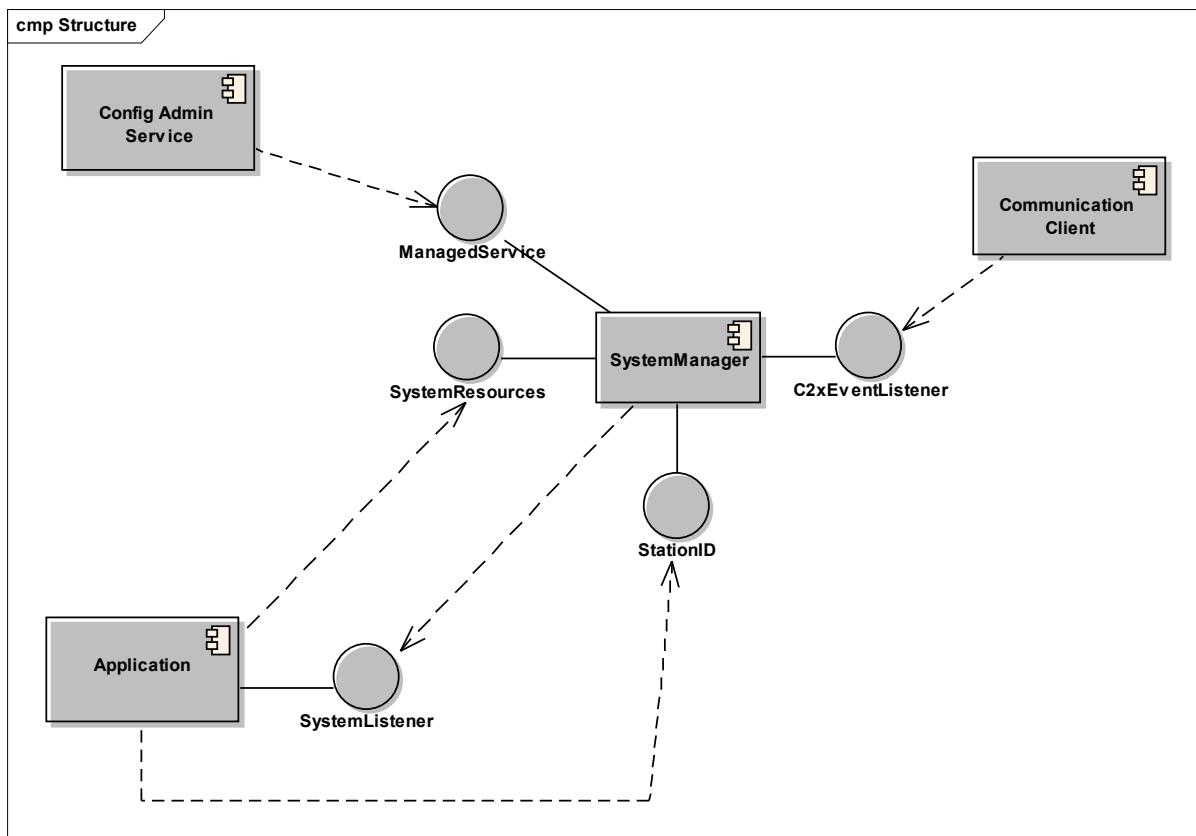


Abbildung 3-13: Einordnung des SystemManager

Den Funktionen stellt die Komponente SystemManager eine Schnittstelle SystemResources zur Verfügung, über die die Funktionen abfragen können, wieviel CPU, Speicher und Filesystem gerade ausgelastet sind, ob das System gerade online ist und welche Datenrate zur Verfügung steht. Auch die aktuelle Auslastung des Ad-Hoc-Netzes können die Funktionen über diese Schnittstelle abfragen. Darüber hinaus haben die Funktionen die Möglichkeit, über die Schnittstelle StationID die ID der IVS abzufragen, die als String zurückgegeben wird. SystemManager greift selbst auf CommunicationClient zurück, um Informationen über die Systemauslastung zu erhalten. Über den OSGi Config Admin Service bzw. aus einer vordefinierten Liste erhält sie Informationen über die Prioritäten der Funktionen, die bei Eingriffen zur Stabilisierung des Systems zu beachten sind. Ebenfalls über den Config Admin Service erhält SystemManager von der Versuchssteuerung Listen von Funktionen, die bei dem aktuellen Versuch in jedem Fall aktiv sein müssen bzw. die nicht aktiv sein dürfen.

Die Funktionen selbst sollen eine Schnittstelle SystemListener implementieren, über die sie selbst vom System benachrichtigt werden, wenn ein kritischer Systemzustand erreicht wird oder behoben wurde. Wenn die verwalteten Systemressourcen knapp werden und einen kritischen Zustand erreichen, werden die Funktionen entsprechend ihrer Priorität über diese Schnittstelle benachrichtigt, dass Sie ihren Ressourcenbedarf einschränken sollen. Wenn auf diesem Weg die Überlast nicht beseitigt werden kann, dann werden Funktionskomponenten entsprechend ihrer Priorität heruntergefahren.

Die für die Funktionen relevanten Abläufe sind nachfolgend dargestellt. Wenn z.B. die CPU-Last zu hoch wird, werden zunächst alle Funktionen über diesen Zustand informiert. Die

Funktionen werden damit aufgefordert, ihre CPU-Nutzung auf das Notwendige einzuschränken. Parallel dazu besteht die Möglichkeit, dass die Funktionen die genaue CPU-Last in Prozent von SystemManager abfragen und spezieller auf den kritischen Zustand reagieren.

Falls nach einer vorgegebenen Zeit die CPU-Last immer noch zu groß ist, fragt SystemManager bei Funktionskoordination für alle Funktionen ab, ob diese abgeschaltet werden können und welche Priorität sie haben. Danach werden so lange Funktionen abgeschaltet, bis die CPU-Last einen vertretbaren Wert annimmt. Es werden zuerst die Funktionen in der Reihenfolge ihrer Priorität abgeschaltet, für die die Versuchssteuerung angibt, dass sie abgeschaltet werden können. Die Vorgehensweise bei Speicherknappheit ist die gleiche.

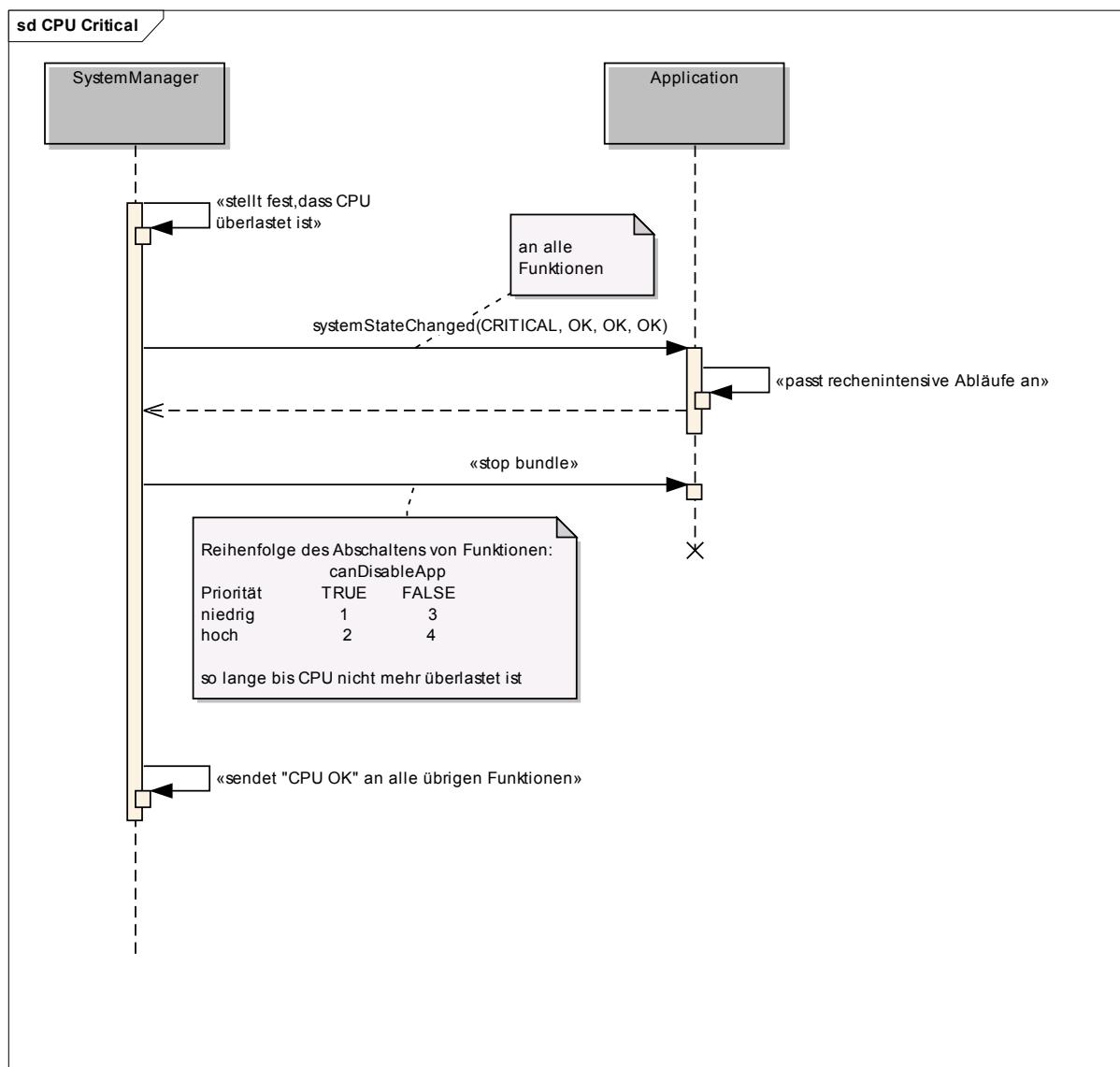


Abbildung 3-14: Ablaufdiagramm bei ausgelasteter CPU

Bei einer Knaptheit der C2X-Übertragungskapazität werden die Funktionen in gleicher Weise informiert und können reagieren. Hier werden die Funktionen aber nicht abgeschaltet, sondern der kritische Zustand wird dadurch aufgehoben, dass Communication Client C2X-Nachrichten der Funktionen verzögert oder gar nicht sendet. Dies geschieht wieder unter Berücksichtigung der vordefinierten Prioritäten der Funktionen.

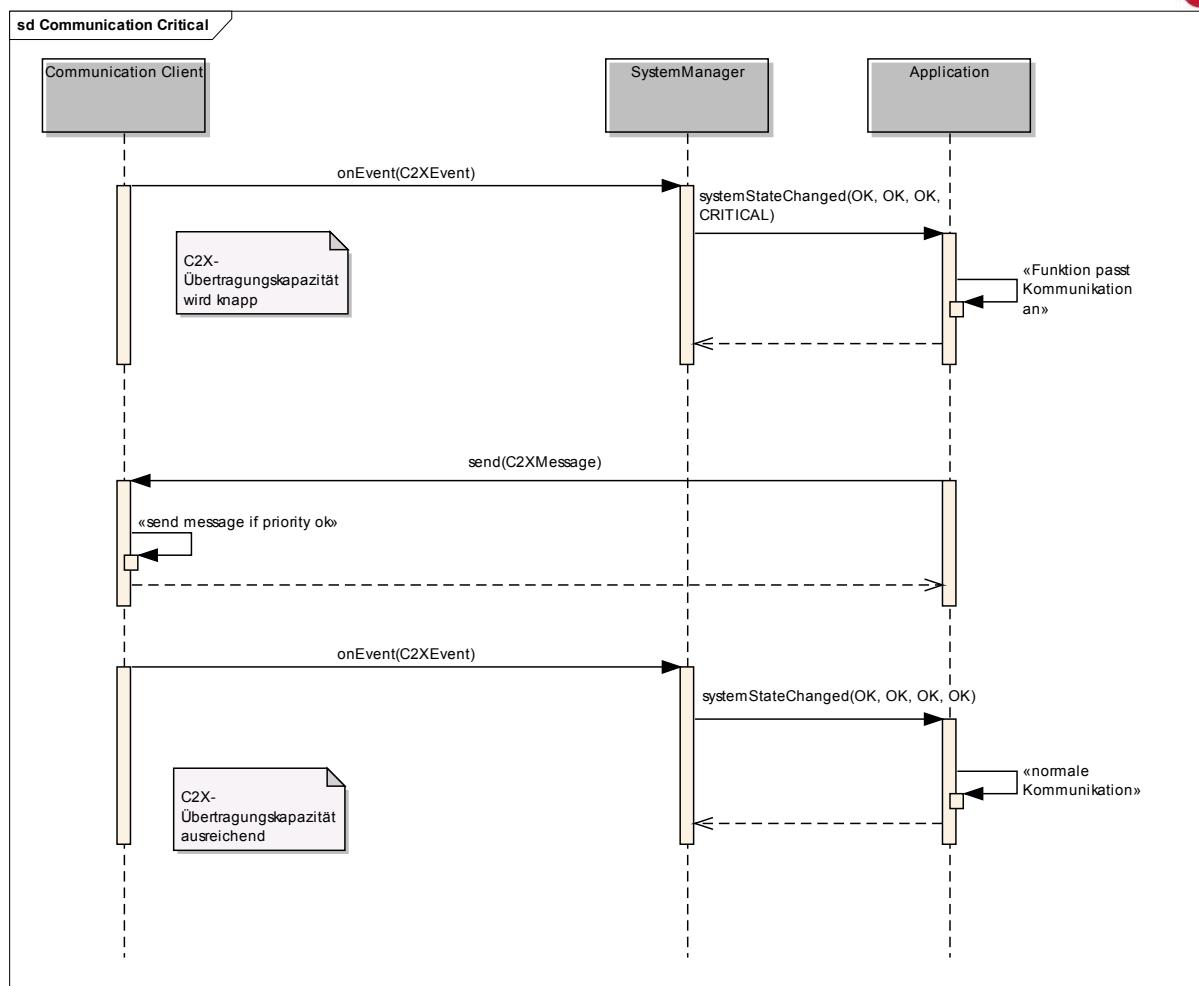


Abbildung 3-15: Ablaufdiagramm bei ausgelasteter Bandbreite

3.2.3 Kommunikations-API

C2XC

Die Nachrichten, die der Router empfängt, werden auf dem Host von der Komponente Communication Client entgegen genommen und (ggf. nach einer Plausibilitätsprüfung) in die Umfeldtabelle eingetragen. Die Nachrichten selbst werden von einer weiteren Komponente C2XCMessages definiert, in ein Binärformat codiert und von diesem decodiert. C2XCMessages wird als Library-Bundle implementiert. Die Umfeldtabelle implementiert ein Listener-Interface, um die C2XC-Nachrichten entgegen zu nehmen. Die Funktionen können die C2XC-Nachrichten aus der Umfeldtabelle über ein Relevanzfilter erhalten. Die Umfeldtabelle wird weiter unten in einem separaten Kapitel dargestellt.

Um C2XC-Nachrichten zu senden, können die Funktionen direkt auf die Komponente Communication Client zugreifen, die die Nachrichten dann an den Router weitergibt.

Consumer-WLAN / UMTS

Funktionen können Consumer-WLAN nach IEEE 802.11b/g und UMTS zum Zugriff auf Internetdienste verwenden. Dazu greifen die Funktionen direkt auf das von Java bereitgestellte TCP zu.

Die Übertragung von C2X-Nachrichten über Consumer-WLAN bzw. UMTS läuft wie folgt ab: Die Funktionen erzeugen ihre Nachrichten unabhängig vom Übertragungsmedium in Form von vordefinierten Message-Objekten, die sie an CommunicationClient übergeben. In diesen Nachrichten können sie spezifizieren, welcher Übertragungsweg für diese Nachricht zu wählen ist. Die Umsetzung dieser Vorgabe erfolgt auf der CCU. Eingehende Nachrichten werden unabhängig vom Übertragungsweg in die Umfeldtabelle eingestellt und haben ebenfalls die Form eines Message-Objekts. In den Message-Objekten ist eine Kennzeichnung des Übertragungsweges enthalten.

Dieser Mechanismus setzt voraus, dass C2X-Nachrichten unabhängig von einer spezifischen Hauptfunktion und in einem Broadcast-ähnlichen Verfahren gesendet und in das sim^{TD} Message Framework integriert werden. Um die Nachrichtenverarbeitung möglichst einheitlich handzuhaben, werden die gleichen Mechanismen für die Verarbeitung von IEEE 802.11p, Consumer-WLAN und UMTS-Nachrichten verwendet. Da es sich nicht um eine applikationsspezifische Nachricht handelt, erfolgt die Verarbeitung über Systemkomponenten (nicht über eine spezielle Hauptfunktion).

Auf dem Host muss dazu lediglich das Medienflag (Übertragungsflag) gesetzt werden. Die weitere Anpassung an den Übertragungsweg (inkl. des 'c2xueberUMTS' Protokolls) erfolgt auf dem Router bzw. im entsprechenden (Geo-)Server. Für die Übertragung über Consumer-WLAN sind keine spezifischen Protokollanpassungen notwendig.

3.2.4 Datenbasis

Die Datenbasis stellt auf der Application Unit (AU) insbesondere den Funktions-Komponenten Daten zur Verfügung, die als Informationspush vorliegen. Die Datenbasis ist keine eigene Komponente sondern besteht aus mehreren Komponenten. Die Daten kommen entweder aus der Fahrzeugsensorik über die VAPI oder über die Kommunikationsschnittstellen. Im Gegensatz zu einer „Local Dynamic Map“, wie sie in dem Projekt Safespot definiert wird, ist die Datenbasis nur eine Schnittstelle für Eingangsdaten, keine umfassende Datenbasis, auf der das gesamte Systemwissen abgelegt wird oder auf dem die Funktionen Daten manipulieren können. An dieser Schicht können alle Daten vom Logging aufgezeichnet und wieder über einen TracePlayer zu Testzwecken eingespielt werden.

Es sei aber darauf hingewiesen, dass ausschließlich die verbindungslos übertragenen Daten in den definierten C2X-Paketen (siehe Deliverable D21.4) in die Umfeldtabelle geschrieben werden, d.h. im Rahmen der Datenbasis vom Logging erfasst werden. Die Pakete sind nicht fragmentiert und bilden jeweils abgeschlossene Informationseinheiten. Daten, die im Rahmen von dialogorientierten Protokollen in das Fahrzeug kommen (z.B. SOAP) werden direkt von den Funktionen empfangen und werden nicht über die Datenbasis zur Verfügung gestellt. Das gleiche gilt über Statusdaten der CCU, die Dialogbasiert abgefragt werden.

Abbildung 3-16 stellt die Komponenten der Datenbasis im Zusammenhang mit den anderen relevanten Komponenten der AU dar. Zumindest die Komponenten Umfeldtabelle sowie C2XMessages werden nicht nur in den ITS Vehicle Stations, sondern auch in den ITS Roadside Stations integriert.

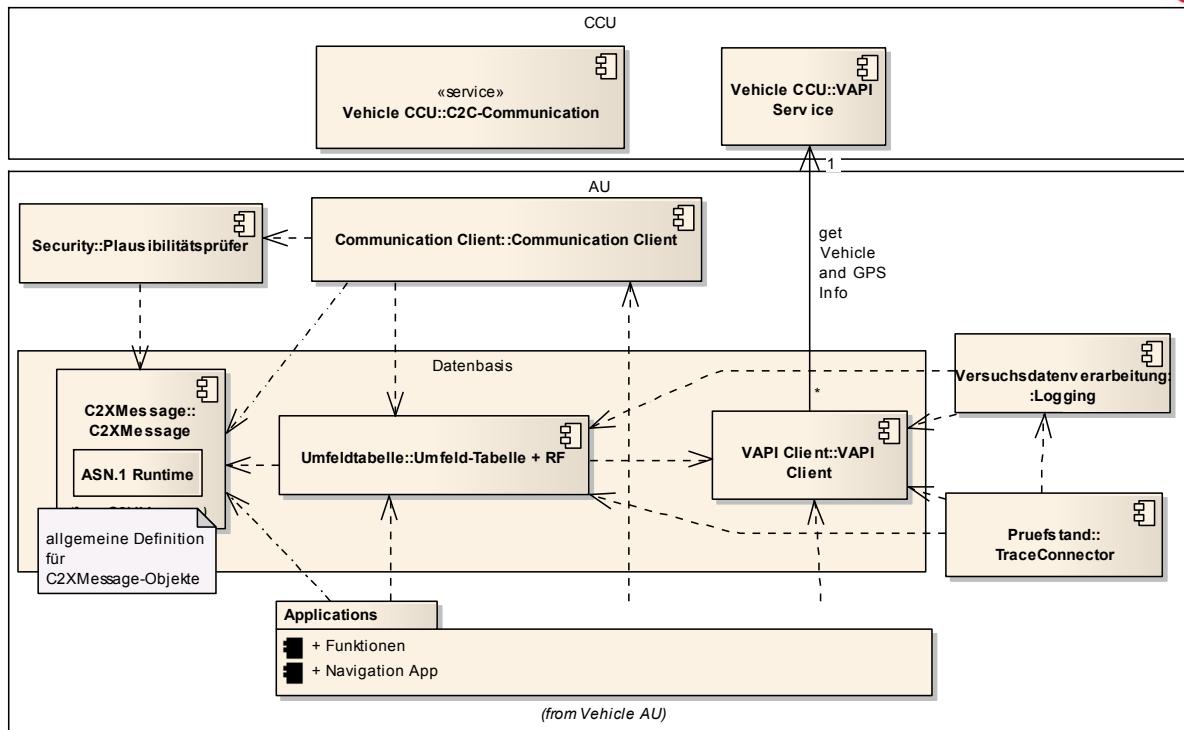


Abbildung 3-16: Komponenten der Datenbasis und Einbindung in die AU-Architektur

3.2.4.1 VAPI-Client

Das Konzept der VAPI sieht vor, dass mehrere Anwendungen auf den VAPI-Server zugreifen und Daten abonnieren können. Dies ermöglicht es jeder Anwendung, eine individuelle Drosselung der Daten vorzunehmen. Weiterhin wird die CPU-Last auf der Vehicle Communication Control Unit reduziert, da nicht alle im Profil vorhanden Objekte dauerhaft beobachtet werden müssen, solange keine Abonnement des Objekts vorliegt.

Da somit jede Anwendung eine Verbindung zur VAPI benötigt, initiiert die VAPI-Client-Systemkomponente auf der Vehicle Application Unit (VAU) diese Verbindung zum VAPI-Server auf der Vehicle Communication Unit (VCCU) und stellt der Anwendung ein Service-Interface (Dienst) zur Verfügung, welches Methoden zum Abonnieren und Abrufen von Fahrzeugdaten bereitstellt. Abbildung 3-17 zeigt die Schnittstelle des Service-Interfaces, das den Funktionen zur Verfügung gestellt wird.

Ein Objektpool auf Seiten der VAU ist derzeit nicht vorgesehen.

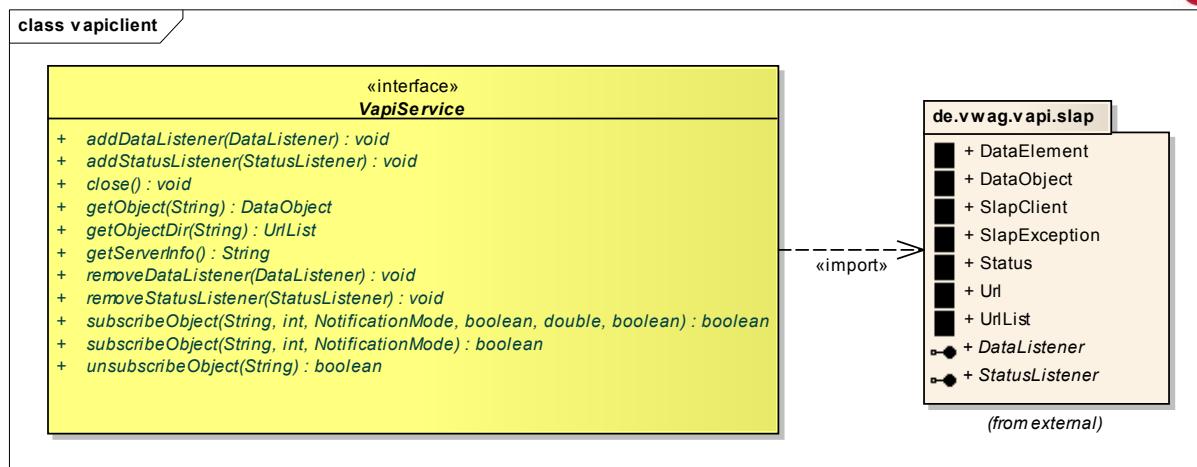


Abbildung 3-17: VapiClient-Bundle Interface

Weiterhin wird die Komponente VAPIClient einige zusätzliche Klassen exportieren, die benötigt werden, um Datentypen, Listener und Exceptions der Java SLAP API verwenden zu können.

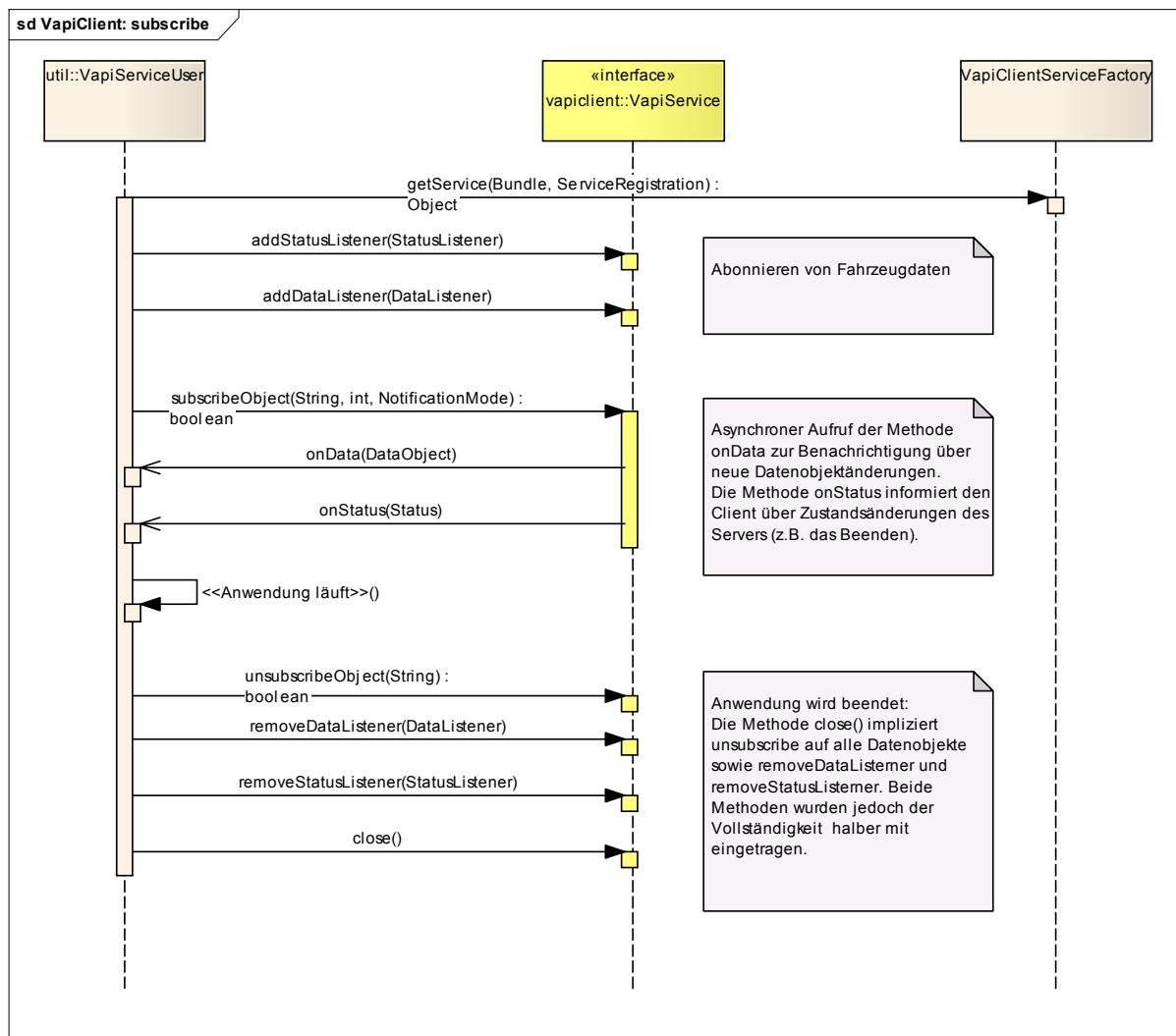


Abbildung 3-18: Sequenzdiagramm: Subscribe

Abbildung 3-18 zeigt ein Sequenzdiagramm zur Beschreibung eines Subscribe-Vorgangs. Die Komponente „VapiServiceUser“ ist eine fiktive Client-Anwendung, die Fahrzeugdaten abonnieren möchte. Diese Anwendung muss das Interface DataListener implementieren, um über neue Datenobjektänderungen informiert zu werden.

Beim Beenden der Anwendung werden die Listener entfernt und die Verbindung zur VAPI terminiert.

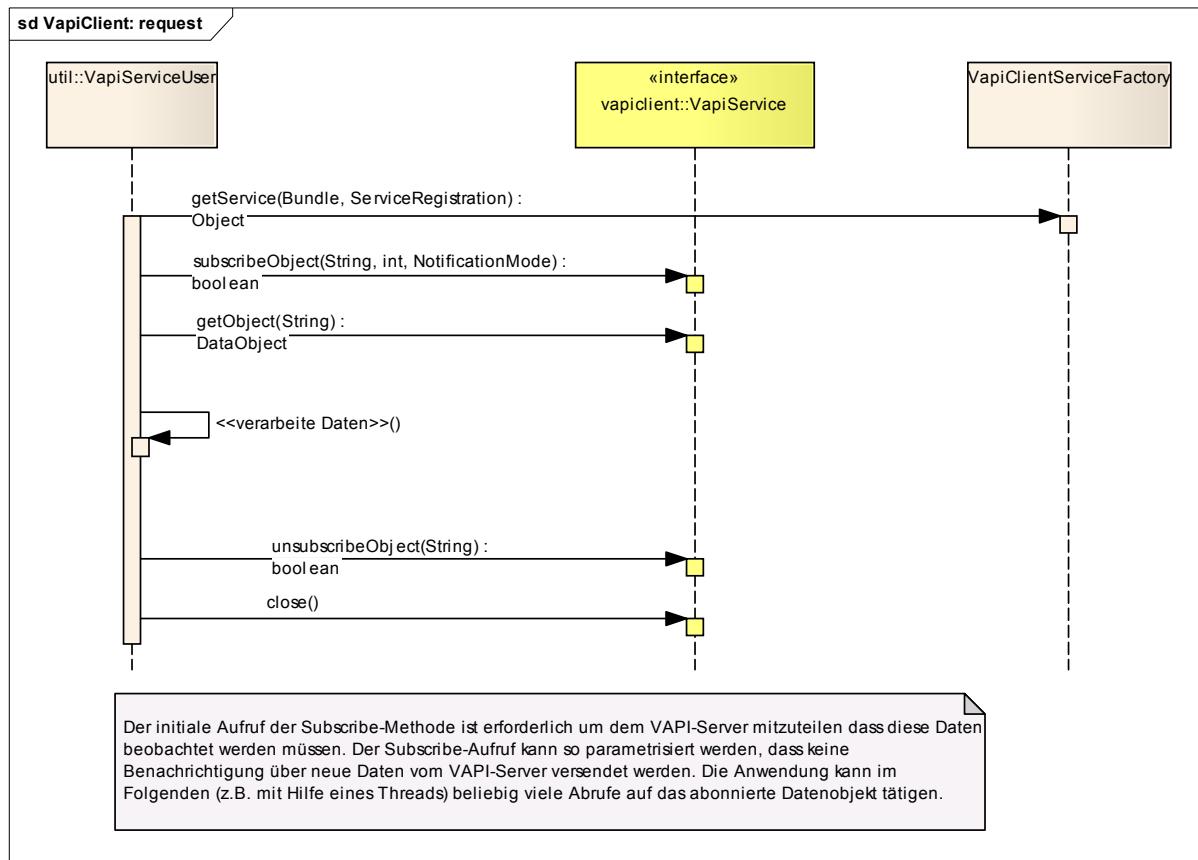


Abbildung 3-19: Sequenzdiagramm: Request

Abbildung 3-19 visualisiert den einfachen Abruf von Fahrzeugdaten mit der Request-Methode getObject(). Der initiale Aufruf der Subscribe-Methode ist erforderlich, um dem VAPI-Server mitzuteilen, dass diese Daten beobachtet werden müssen. Der Subscribe-Aufruf kann so parametrisiert werden, dass keine Benachrichtigungen über neue Daten vom VAPI-Server versendet werden. Die Anwendung kann im Folgenden z.B. mit Hilfe eines Threads beliebig viele Abrufe auf das abonnierte Datenobjekt tätigen.

Datenobjektanforderungen der sim^{TD}-Funktionen

Die VAPI-Objekte sind in Deliverable D11.4 beschrieben. Zusätzlich wird die VAPI aber noch weitere Datenobjekte zur Verfügung stellen, die von anderen Teilprojekten benötigt werden. Es ist zu berücksichtigen, dass die Verfügbarkeit, Aktualisierungsfrequenz und Standardabweichung der gelisteten CAN-Signale bzw. Messwerte sowohl fahrzeug-, hersteller- als auch modellspezifisch variieren!

3.2.4.2 C2X-Nachrichten und ASN.1-Decoder

Die Standardnachrichten des C2X-Netzwerkes werden in ASN.1 spezifiziert. Aus dieser Spezifikation wird mit dem Tool „ASN1C“ von Objective Systems Inc. Java Code erstellt, der zu den einzelnen Nachrichtentypen Klassen definiert. Diese Klassendefinitionen werden in der Komponente C2XMessages zur Verfügung gestellt, sodass entsprechende Nachrichtenobjekte zwischen den Komponenten Communication Client, Plausibilitätsprüfer, Umfeldtabelle, Relevanzfilter und den Funktionen ausgetauscht werden. Zusätzlich werden in den Nachrichtenobjekten noch Informationen der Netzwerkschicht (z.B. Adressierung) und der Sicherheitskomponenten abgelegt.

Die Komponente enthält auch die Runtime Libraries zum de- und encodieren des Applicationpayloads nach den Packed Encoding Rules (PER). Die Nachrichtenklassen haben entsprechende Methoden, um die Decodierung durchzuführen und die Datenelemente des Objektes auf Basis eines empfangenen Bytestrings zu befüllen, bzw. die Encodierung durchzuführen und aus den Datenobjekten einen entsprechenden Bytestring für das Senden zu erstellen. Details hierzu finden sich in Deliverable D21.4.

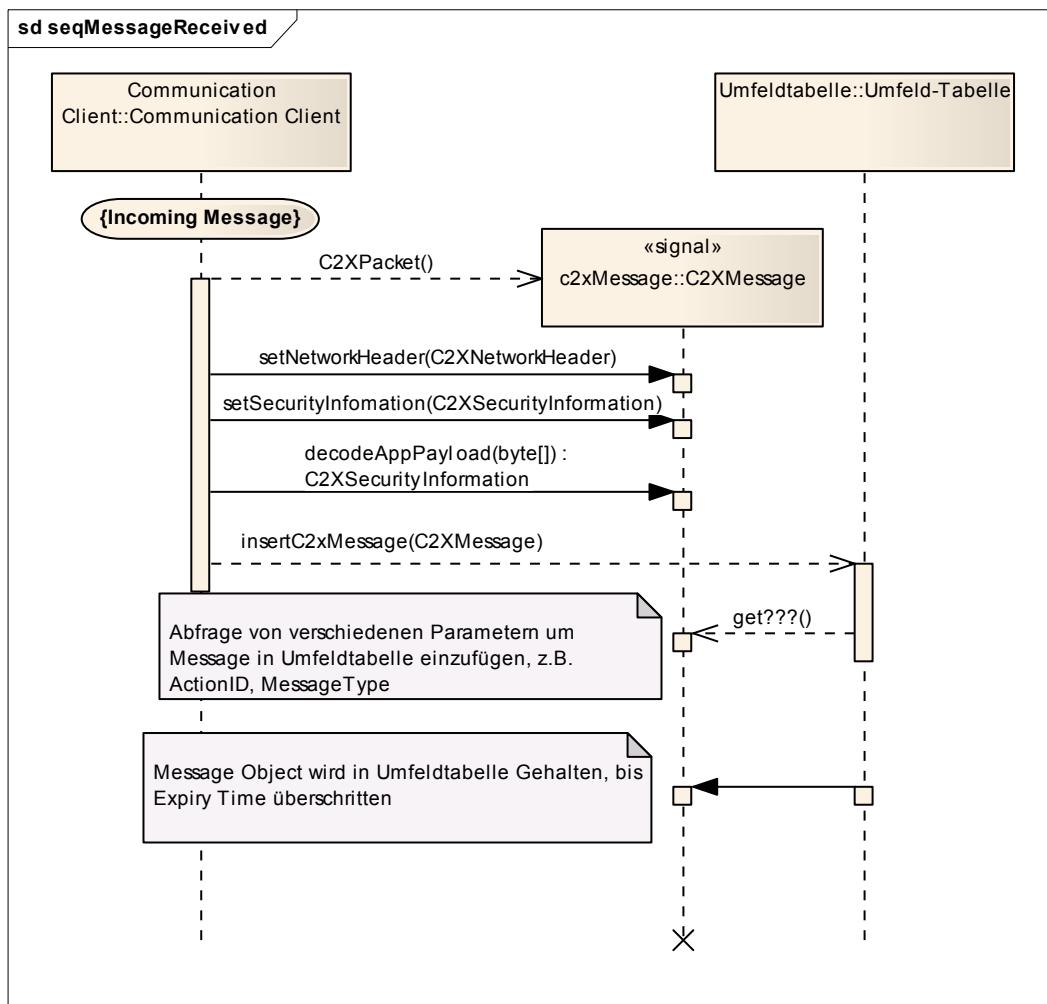


Abbildung 3-20: Sequenzdiagramm zum Erstellen eines Java-Objektes für eine C2X-Nachricht und das Ablegen in die Umfeldtabelle

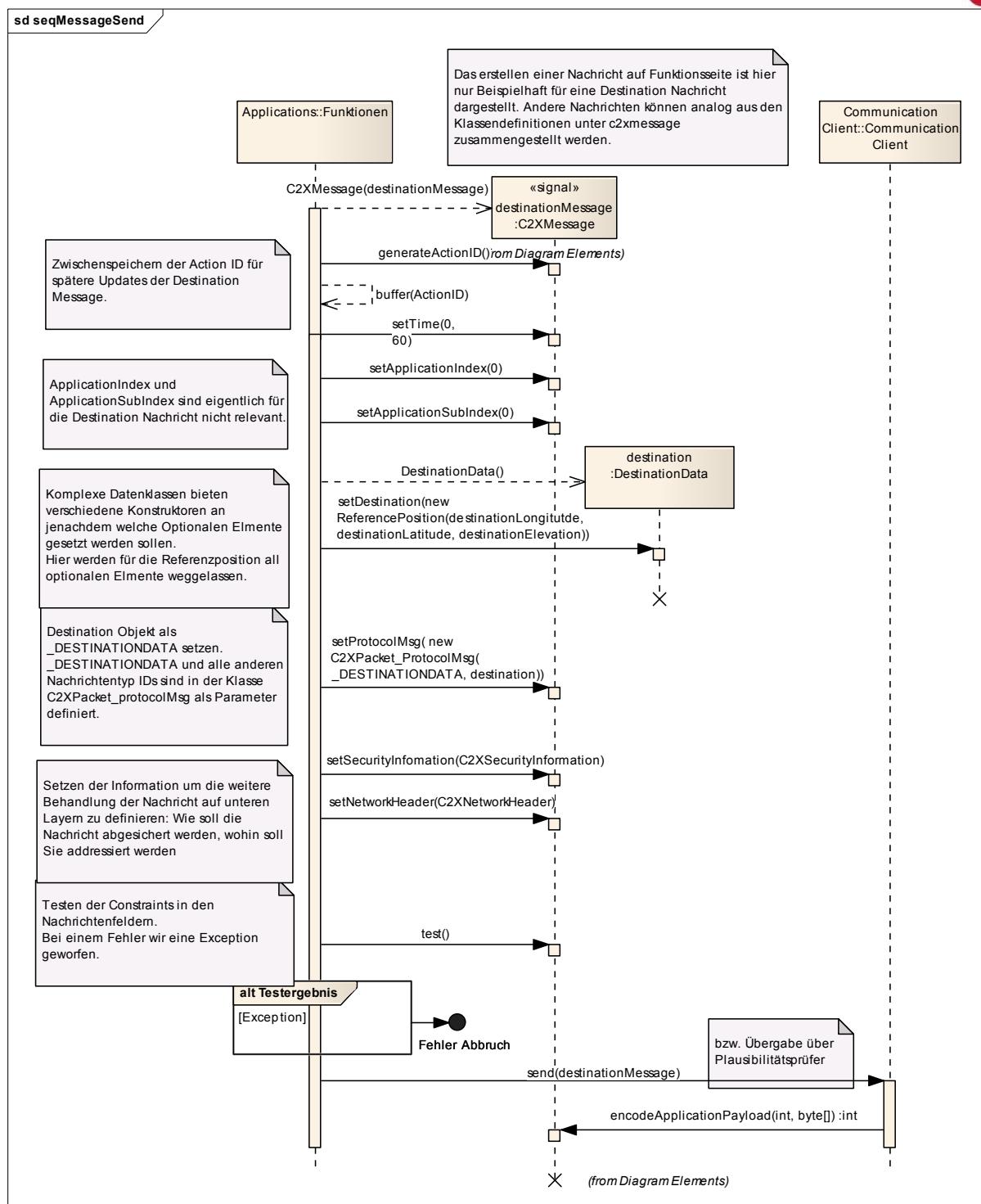


Abbildung 3-21: Sequenzdiagramm zum Erstellen einer C2X-Nachricht in einer Funktion; beispielhaft für eine Destination-Nachricht

3.2.4.3 Umfeldtabelle

Die Umfeldtabelle stellt Information aus standardisierten C2X-Nachrichten zentral auf dem Host zur Verfügung. Dabei werden im Wesentlichen zwei Datentypen unterschieden:

- CAM (Cooperative Awareness Messages), die zu einer Nachbarschaftstabelle der kommunizierenden Objekte zusammengestellt werden. Die Nachbarschaftstabelle ist Teil der Umfeldtabelle.
- Andere Nachrichten (z.B. DEN: Decentralized Environmental Notification) die über Standardinformationen (z.B. Typ, Nachrichten ID) zugreifbar sind.

Soweit Nachrichten allgemein in sim^{TD} spezifiziert sind, werden die Inhalte decodiert als JAVA-Objekte zur Verfügung gestellt. Andere Daten werden als dekodierte Nachricht in einem Container-Objekt zur Verfügung gestellt.

Daten werden in den Umfeldtabellen nach Gültigkeitsparametern (zeitlich) gehalten und automatisch entfernt.

Bereithalten von CAM-Nachrichten

CAM-Nachrichten werden von allen Kommunikationspartnern über das C2X-Netzwerk (IEEE 802.11p) periodisch ausgesandt. Sie zeigen die Präsenz, grundlegende Eigenschaften und fundamentale Zustandsdaten an. Das Modul Umfeldtabelle bekommt diese Nachrichten über den CommunicationClient als decodierte C2X-Nachricht (C2XMessage). Alle Informationen werden in ein JAVA-Datenobjekt geschrieben. Dabei werden evtl. auch Information aus dem Netzwerkheader mit verwendet (z.B. ID und Positionsdaten). Die Information der empfangenen CAM-Nachrichten bildet eine Nachbarschaftstabelle, die ein Teil der Umfeldtabelle ist.

Zu Testzwecken können die Nachrichten statt vom CommunicationClient auch vom TraceConnector eingespielt werden. Dies können synthetische Testszenarien sein oder aufgezeichnete Daten. Es wird von der Umfeldtabelle jedoch nur ein Interface zur Verfügung gestellt, das im Prüfstand vom TraceConnector und im Testgelände und Versuchsgebiet von der Komponente CommunicationClient bedient wird. Der Plausibilitätscheck wird direkt vom CommunicationClient abgefragt und ist somit kein direkter Kommunikationspartner der Umfeldtabelle mehr.

Die Information wird in die Nachbarschaftstabelle eingepflegt. D.h. falls es sich um einen Eintrag mit bekannter Fahrzeug-ID handelt wird dieser ersetzt. Für eine neue ID wird ein neuer Eintrag angelegt.

Einträge werden in der Nachbarschaftstabelle nach einer Zeit CAM_TIMEOUT aus der Nachbarschaftstabelle gelöscht, wenn keine Aktualisierung erfolgte. CAM_TIMEOUT sollte mindestens doppelt so groß sein wie die maximale Periode für das Aussenden von CAM, um einzelne Kommunikationsausfälle zu überbrücken. Dabei orientiert sich die Umfeldtabelle an den im C2XMessage-Package definierten CAM-Frequenzen.

Es wird davon ausgegangen, dass im normalen Feldtest nur eine begrenzte Anzahl CAM_N_MAX Fahrzeuge auf einmal in Reichweite sind. Diese Zahl kann aber in Extremfällen überschritten werden, z.B. beim Startup der Testflotte auf einem Parkplatz oder einem speziellen Belastungstest. Wird jedoch doch eine Begrenzung der CAM-Einträge notwendig, erfolgt diese nach definierten Kriterien (z.B. Entfernung, Plausibilitätswert, letztes Update, etc). Der Wert CAM_N_MAX kann im späteren Versuchsbetrieb angepasst werden.

API-Methoden

- Registrieren auf Änderungen von CAM
- Registrieren auf neue CAM-Einträge
- Abfrage der Nachbarschaftstabelle (Alle, Einsatzfahrzeuge, ÖPNV)
- Ausgeben von allen empfangen Nachrichten an das Logging zum Erstellen von Traces

Weitere spezieller Zugriffsmechanismen werden in dem Kapitel Relevanzfilter beschrieben.

Bereithalten anderer Nachrichten

Andere Nachrichten aus dem C2X-Netzwerk werden parallel zu den CAM gespeichert. Diese Nachrichten sind z.B. Geometrie- bzw. Topologie-Daten, Ampelstatusinformation oder Decentralized Environmental Notification (DEN) mit ereignisbasierten Daten. Nachrichten werden evtl. von mehreren Hauptfunktionen aus den Umfeldtabellen abgerufen, z.B. DEN von den Funktionen Lokalen Gefahrenwarnung, Straßenvorausschau und Erweiterte Navigation.

Zu Testzwecken können die Nachrichten, statt vom CommunicationClient auch von einem TraceConnector eingespielt werden. Dies können synthetische Testszenarien sein oder aufgezeichnete Daten. Zum Erstellen dieser Traces registriert das Logging einen C2XMessageHandler mit den Properties: RequestType = RequestType.RECORDING an der Umfeldtabelle. Die Umfeldtabelle bedient ab diesem Zeitpunkt den C2XMessageHandler des Loggings. Alle empfangenen Nachrichten werden dabei unverändert an das Logging weitergereicht. Die zentrale Komponente TracePlayer kann dann bestehende Traces abspielen und auf die einzelnen Plattformen verteilen.

Die Nachrichten werden dekodiert und in JAVA-Objekte abgelegt. Hierbei sollten zunächst allgemeine Daten abgelegt werden, z.B. ein Typ-Identifier, Zeitstempel (Zeitpunkt der Generierung und ggf. Verfallszeitpunkt), Referenzposition, SenderID. Außer dem Typ-Identifier sind die Daten optional.

Daten zur Vertrauenswürdigkeit einer Nachricht, werden von der vorgeschalteten Plausibilitätsprüfung in die C2X-Nachricht eingearbeitet und an die Umfeldtabelle weitergeleitet. Die Funktion kann dann entscheiden, ob sie eine ggf. nicht vertrauenswürdige Nachricht verarbeitet oder nicht.

Nachrichten können über die Ausbreitungswege ITS G5A, Consumer-WLAN und UMTS empfangen werden. Sie werden auch als unterschiedliche Nachrichten eingestuft, wenn die gleiche Nachricht über zwei unterschiedliche Übertragungswege empfangen wurde. Dies ist notwendig, da die Funktionen abhängig vom Übertragungsweg getestet werden sollen. Nachrichten werden daher nur als Duplikate eingestuft, wenn sie einen identischen Inhalt haben und von dem gleichen Originator (gleicher Zeitstempel und Message ID) kommen. Es wird ein Empfangszähler inkrementiert und die neue Empfangszeit im entsprechenden Objekt abgelegt. Dies kann hilfreich sein, um das Weiterleiten der Nachrichten auf Applikationsebene zu steuern. Die Nachricht selber ist in jedem Fall nur einfach abzulegen.

Nachrichten werden für eine definierte Gültigkeitsdauer in der Umfeldtabelle gehalten. Diese Gültigkeit ist Teil der Nachricht. Nach Ablauf der Gültigkeit werden die Nachrichten gelöscht.

API-Methoden

- Registrieren auf jede Nachricht eines bestimmten Typs
- Abfrage aller Nachrichten eines bestimmten Typs
- Empfangen einer neuen Nachricht (evtl. Teil der C2X-API)⁴
- Ausgeben von allen empfangen Nachrichten an das Logging zum Erstellen von Traces (Registrierung mit RequestType TRACE_RECORDING erforderlich)

Weitere spezieller Zugriffsmechanismen werden in dem Kapitel Relevanzfilter beschrieben. In der Implementierung werden die beiden Funktionalitäten Umfeldtabelle und Relevanzfilter in einer Komponente implementiert. Den Hauptfunktionen stehen beide Zugriffsmechanismen zur Verfügung. Es ist zu klären ob Informationen aus dem Relevanzfilter für einen späteren Zugriff in der Umfeldtabelle abgelegt wird, z.B. die Zuordnung auf Streckenabschnitte.

Die Funktionen können auf Daten der Umfeldtabelle über ein einheitliches Interface zugreifen. Die Funktion implementiert das von der Umfeldtabelle definierte Interface C2XMessageHandler.

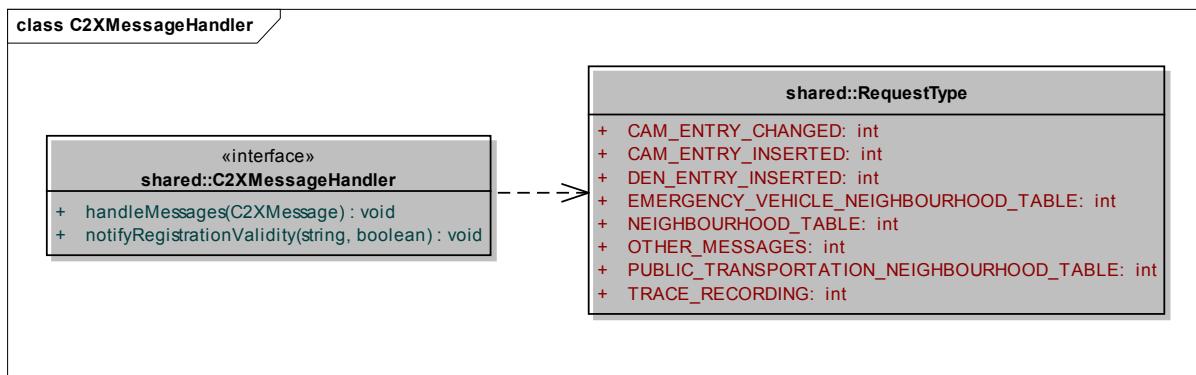


Abbildung 3-22: Interface C2XMessageHandler

Diese Implementierung registriert die Funktion als neuen Service im OSGi-Framework. Bei der Registrierung müssen verschiedene Properties mitgegeben werden. Die Funktion beschreibt mit diesen Properties die gewünschten Daten, die sie empfangen möchte. Sind die Properties gültig und beschreiben einen validen Filter, so bekommt die Funktion von der Umfeldtabelle darüber Rückmeldung. Die Rückmeldung geschieht seitens der Umfeldtabelle durch den Aufruf der Methode `notifyRegistrationValidity ()`. Anhand der beiden Parameter `ErrorString` und `isValid` kann die Funktion zum einen erkennen ob die bei den Properties angegebene Werte korrekt sind, und zum anderen im Falle der nicht validen Filterdefinition erhält sie über den `ErrorString` genauere Angaben zur Lokalisierung des Fehlers. Die Funktion kann somit feststellen, dass sie zum einen von der Umfeldtabelle bedient wird und zum anderen, ob ihr bei der Registrierung angegebener Filter gültig ist.

Im folgenden Komponenten-Diagramm sind die an der Umfeldtabelle beteiligten Komponenten aufgezeigt.

⁴ Gemeinsame Methoden für CAM-Nachrichten und andere C2X-Nachrichten

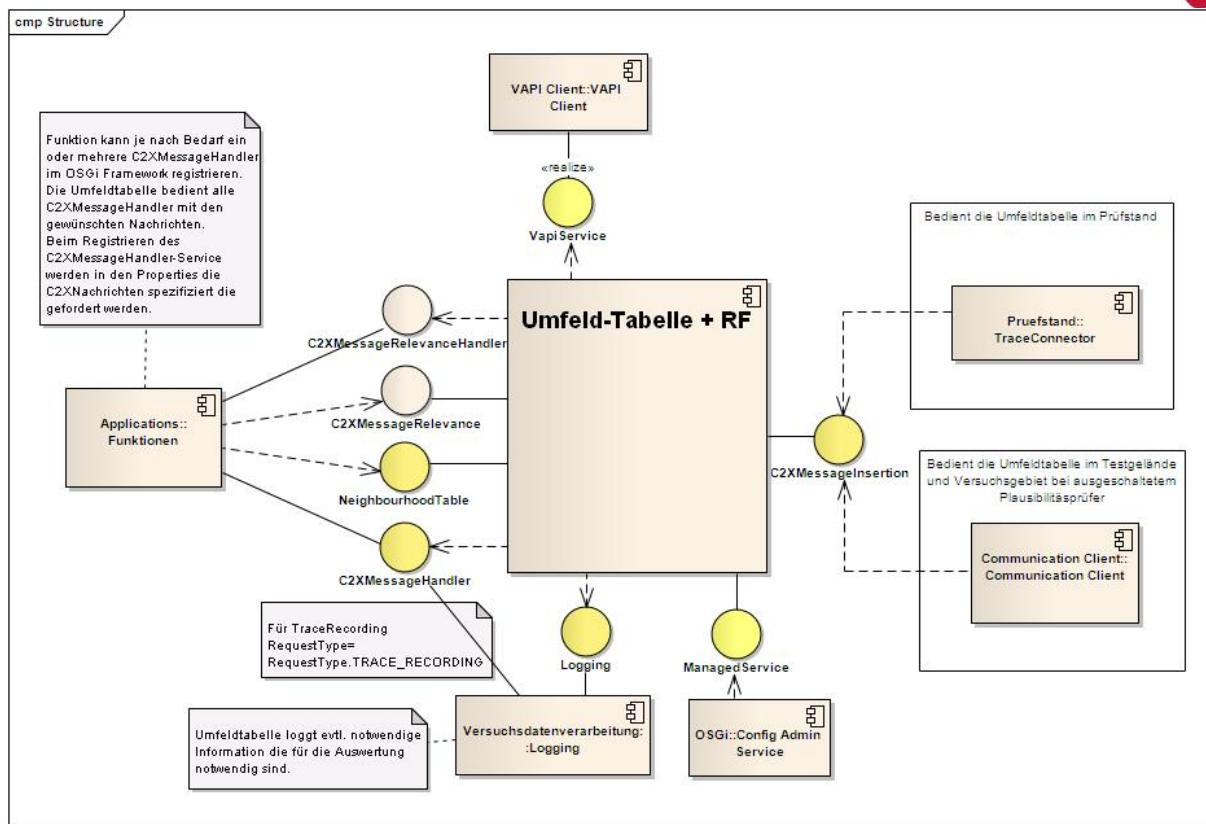


Abbildung 3-23: ComponentModel Umfeldtabelle

Wie in Abbildung 3-23 zu erkennen ist, ist als zwingendes Property der RequestType mit anzugeben. Im Folgenden wird zu jedem RequestType ein Sequenzdiagramm aufgezeigt, um den zeitlichen Ablauf und den Zugriff auf die Umfeldtabelle zu verdeutlichen.

Registrieren auf sich ändernde CAM-Nachrichten

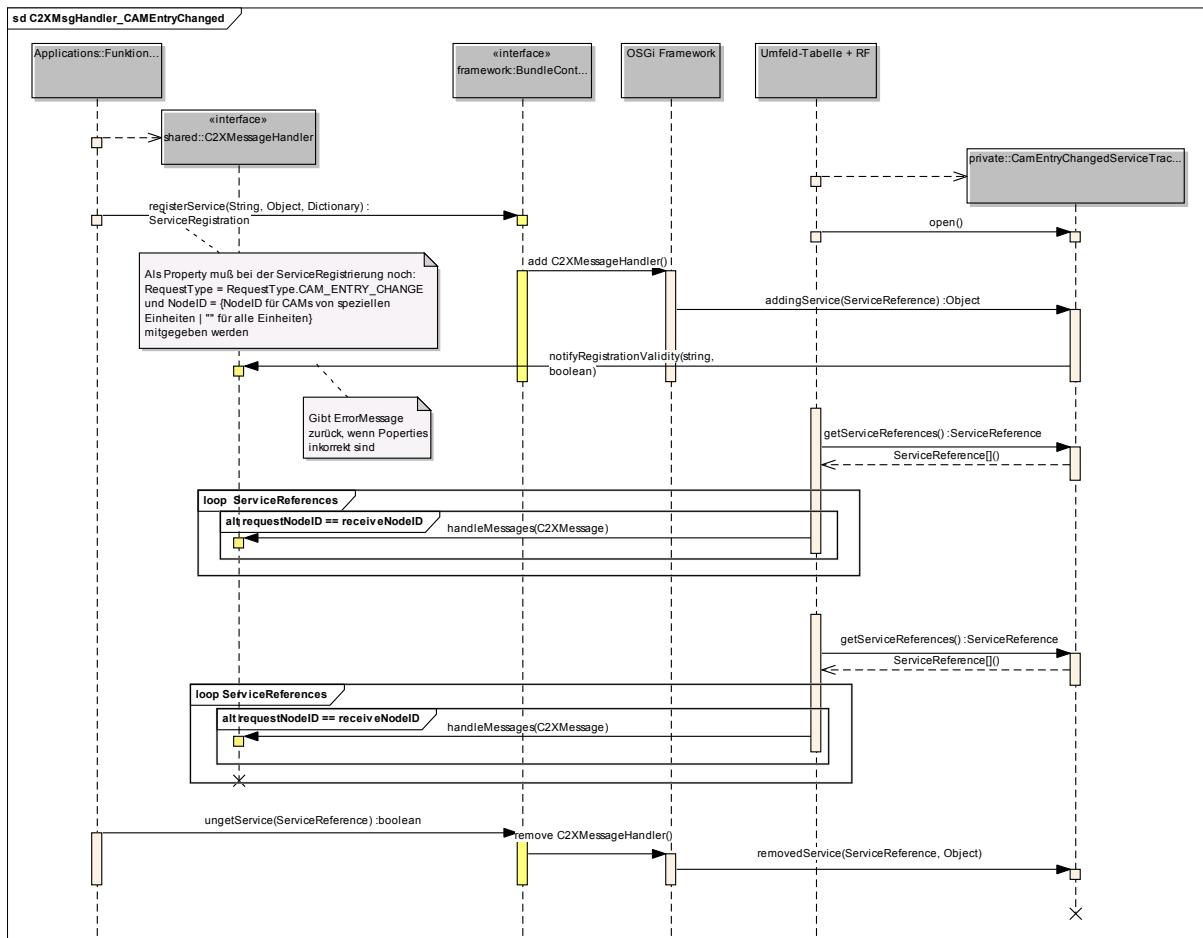


Abbildung 3-24: Ablauf Registrierung geänderte CAM

Registrieren auf neu empfangene CAM-Nachrichten

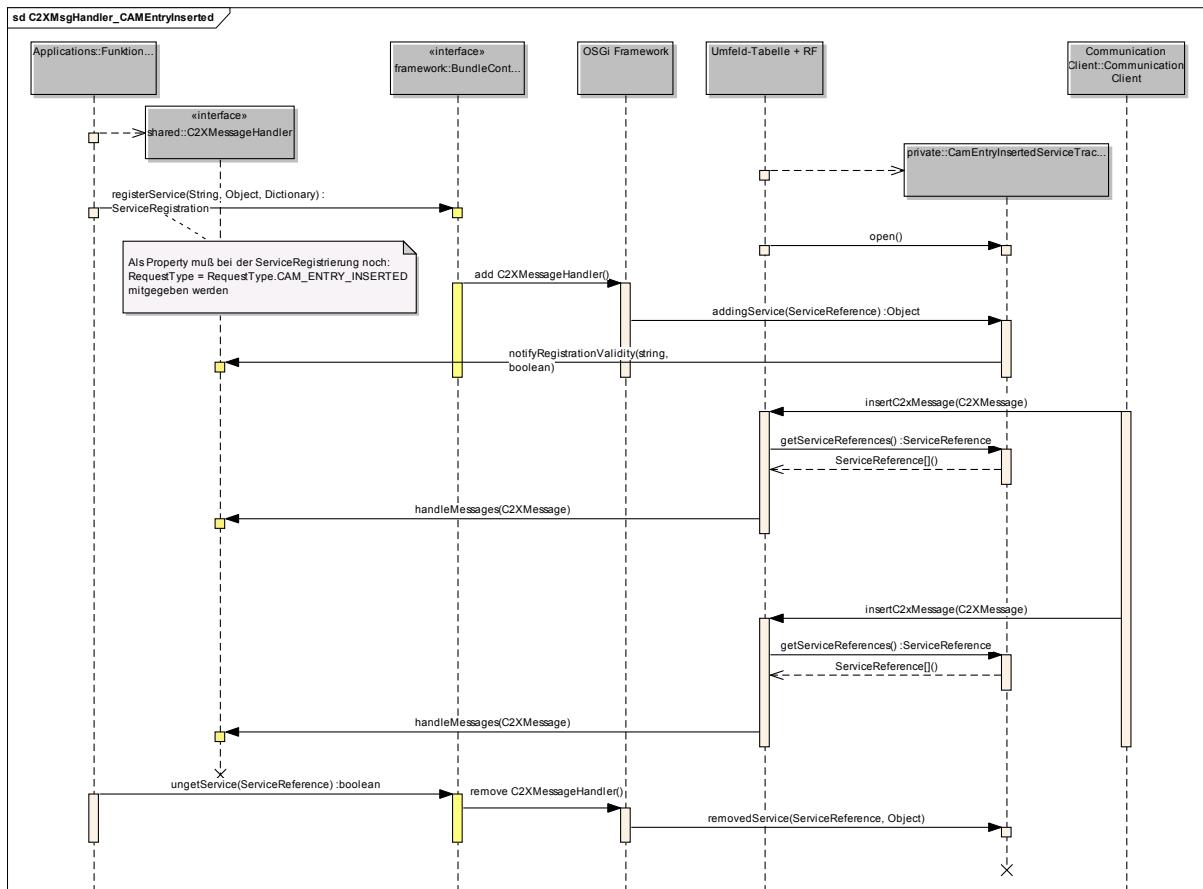


Abbildung 3-25: Ablauf Registrierung neue CAM

Registrieren auf DEN-Nachrichten

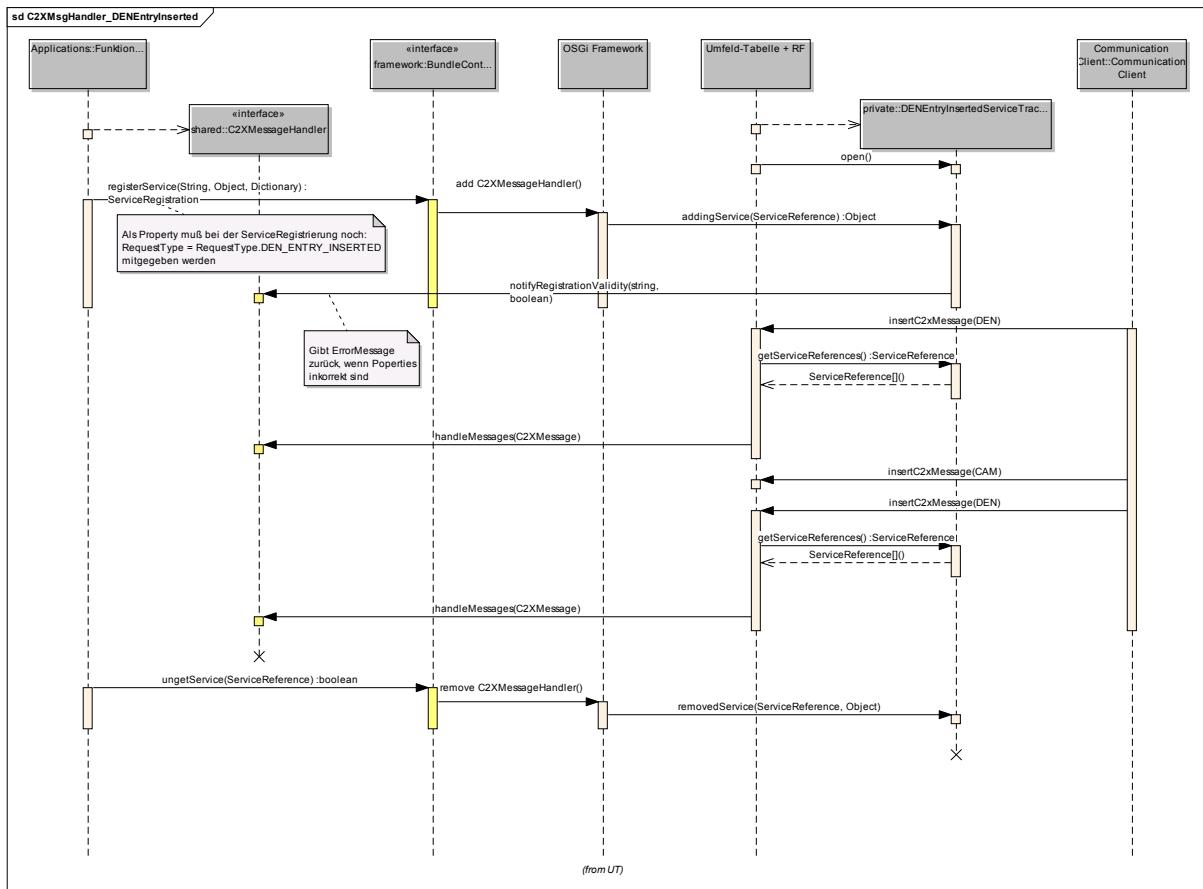


Abbildung 3-26: Ablauf Registrierung DEN

Registrieren für Nachbarschaftstabelle (Periodische Lieferung)

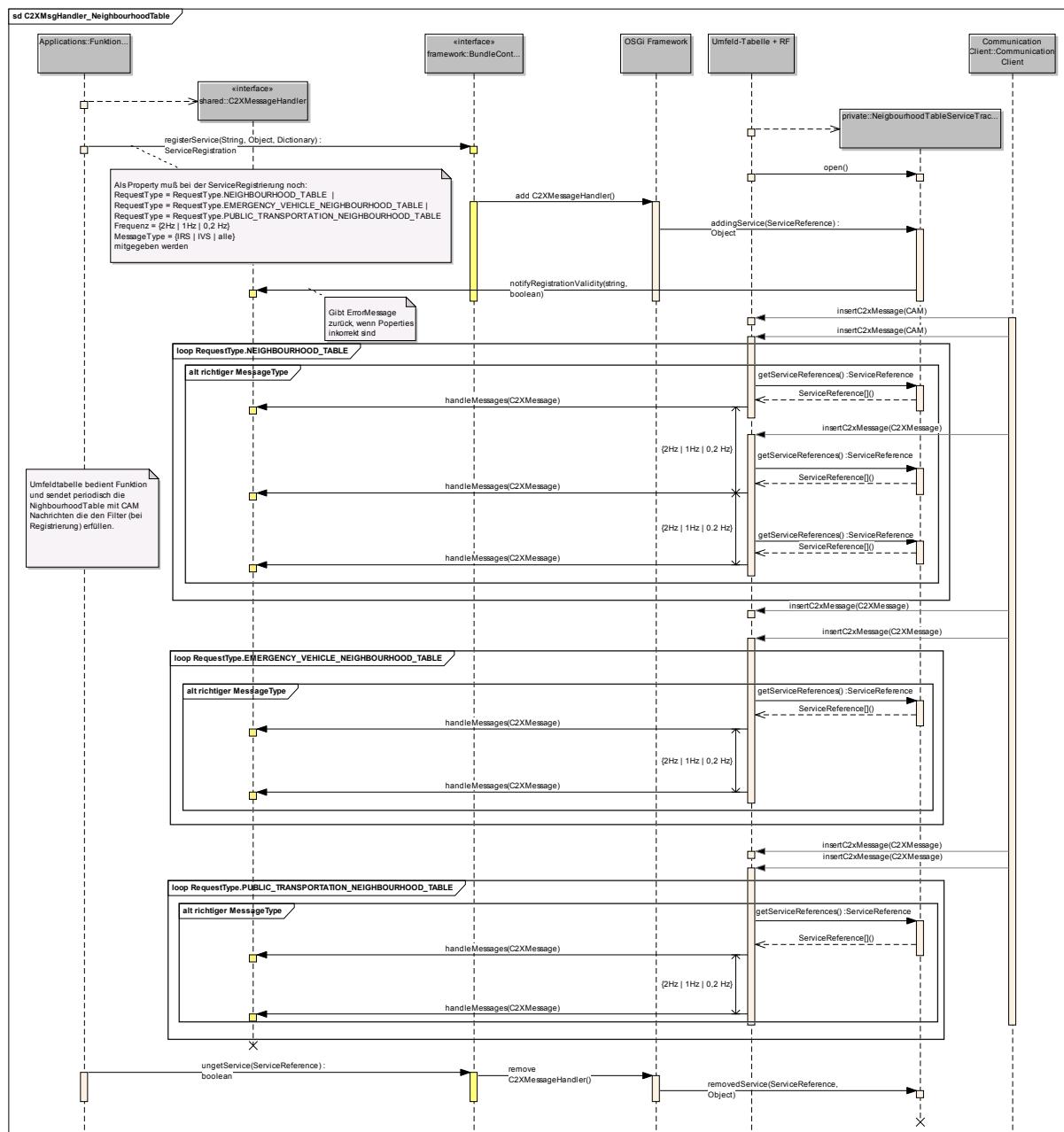


Abbildung 3-27: Ablauf Registrierung Nachbarschaftstabelle

Analog zu dem Parameter
`RequestType.EMERGENCY_VEHICLE_NEIGHBOURHOOD_TABLE`
 können auch entsprechende Parameter für ÖV-Fahrzeuge oder Einsatzfahrzeuge gesetzt werden.

Registrieren auf andere Nachrichten

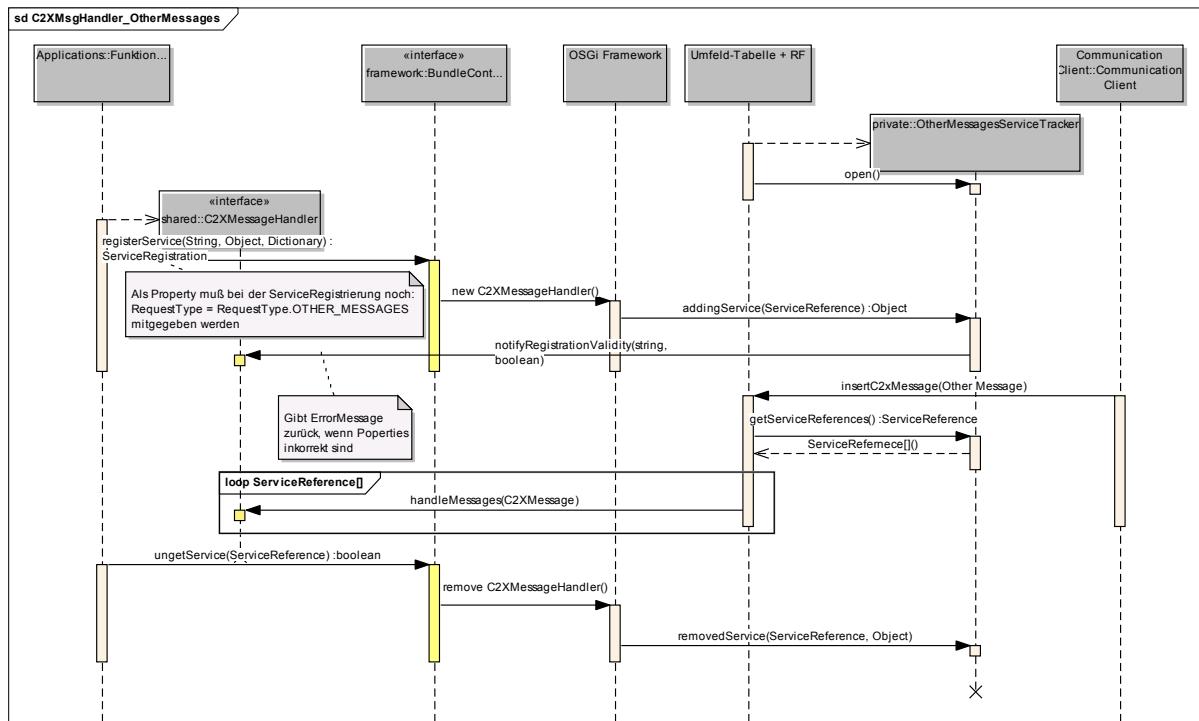


Abbildung 3-28: Ablauf Registrierung andere Nachrichten

Neben den asynchronen Schnittstellen wird es auch eine synchrone Schnittstelle geben, um die Nachbarschaftstabelle direkt abzufragen. Diese Schnittstelle wird durch den Service NeighbourhoodTable repräsentiert.

3.2.4.4 Relevanzfilter

Der Relevanzfilter wird auf die in der Umfeldtabelle einer IVS/IRS abgelegten empfangenen Nachrichten angewendet. Er ermöglicht die Filterung von Nachrichten zum einen bezüglich ihrer Attribute und zum anderen bezüglich bestimmt vordefinierter Filterfunktionen. Dafür werden Filter durch die Komponente spezifiziert. Der Relevanzfilter ist ein Teil der Umfeldtabelle und bietet zusätzliche Dienste für den Zugriff auf die Umfeldtabelle. In der Realisierung verschmelzen die Komponenten Umfeldtabelle und Relevanzfilter. Durch das Zusammenlegen der beiden Komponenten wird ein effizienter Datenzugriff ermöglicht, der den Ressourcenverbrauch für Filterung von Nachrichten für mehrere Hauptfunktionen minimiert. Die genaue Feinspezifikation samt der UML-Diagramme findet sich in der Spezifikation „Spezifikation des C2X – Software-Stacks (von Cirquent bearbeitete Komponenten)“.

Ein Relevanzfilter wird aus Effizienzgründen bei den synchronen Requests als Objekt definiert, der einen logischen Ausdruck in disjunktiver Normalform (DNF) widerspiegelt. Ein Filter wird in XML spezifiziert. Das XML-Schema hierfür ist der Detailspezifikation der Systemkomponente Relevanzfilter zu entnehmen. Um der Forderung nach Verwendung von XML-Dokumenten nachkommen zu können, wird eine Methode zum Parsen angeboten. Diese Methode liefert aus einem String, der einen in XML definierten Filter repräsentiert, ein RelevanzFilter-Objekt.

Schnittstelle

Die Schnittstelle zur Komponente „Relevanzfilter“ sieht wie folgt aus:

- C2XMessageRelevance: Die Komponente erhält durch die Verwendung der Schnittstelle eine Trefferliste aller Nachrichtenobjekte, die den Kriterien des übergebenen Relevanzfilter Objektes genügen. Im Fehlerfall, z.B. bei Übergabe eines ungültigen Filterobjektes wird die Ausnahme FilterException geworfen.
- C2XMessageRelevanceHandler: Die Komponente erhält durch das Registrieren der Schnittstelle als Service asynchron eintreffende Nachrichten, die den Filterkriterien entsprechen. Bei der Registrierung des Services müssen Filterkriterien als Service-Properties übergeben werden. Das Property hat den Schlüssel "RelevanzFilter" und besitzt als Wert den RelevanzFilter XML-String, der den Filter beschreibt und der RelevanzFilter.xsd genügen muss. Die Schnittstelle definiert eine notifyFilterValidity-Methode, die bei der Registrierung des Services positiv bzw. negativ aufgerufen wird. Im Fehlerfall kann durch die entkoppelten Services keine Exception geworfen werden. Stattdessen wird die Exception an eine notifyFilterError-Methode in der Schnittstelle zugestellt.

Aufgrund der Anforderungen kann der Relevanzfilter auch nach folgenden teilweise relativen geometrischen Kriterien filtern:

- "umkreis": Liefert alle Nachrichten im Umkreis von x Metern um die aktuelle Ego-Position (kartesische Berechnung, keine Sphärentegeometrie). Als Parameterwert wird der Radius in Metern erwartet. Die Filterung bezieht sich auf die Daten Longitude und Latitude aus dem Objekt ReferencePosition des C2XPackets. Hierbei ist es notwendig, dass jedeFunktion diese Felder befüllt.
- "rechteck": Liefert alle Nachrichten innerhalb des durch Koordinate 1 (links oben) und Koordinate 2 (rechts unten) aufgespannten Rechtecks (kartesische Berechnung, keine Sphärentegeometrie). Als Parameterwert wird Latitude1|Longitude1|Latitude2|Longitude2|Size (Höhe der Seite)|clockwise (im oder gegen Uhrzeigersinn). Positionen werden als Integer-Werte Einheiten von 1/8 Mikrograd angegeben. Die Höhe wird als Integer in Metern und clockwise als Boolean (true/false) angegeben. Die Filterung bezieht sich auf die Daten Longitude und Latitude aus dem Objekt ReferencePosition des C2XPackets. Hierbei ist es notwendig, dass jedeFunktion diese Felder befüllt.
- "trace": Liefert alle Nachrichten, die bezüglich einer definierten DEN eine um x% größere Trace-Chain-Match-Konfidenz besitzen. Als Parameterwert wird MessageID|Konfidenz erwartet.
- "richtung": Liefert alle Nachrichten, die bezüglich der Fahrtrichtung (Heading) relativ zum Ego-Fahrzeug in einem bestimmten Sektor liegen. Als Parameterwert wird Grad1|Grad2 erwartet. Die Filterung ob eine bestimmte Nachricht in dem aufgespannten Sektor liegt wird anhand der Daten Longitude und Latitude des Objekts ReferencePosition des C2XPackets vorgenommen. Hierbei ist es notwendig, dass jedeFunktion diese Felder befüllt.
- "hoehe": Liefert alle Nachrichten, deren Höhenunterschied (Elevation) relativ zum Egofahrzeug kleiner als x Meter beträgt (nur möglich, falls in den DEN/CAM-Höheninformationen enthalten sind).

Der Relevanzfilter ermöglicht keine weiterführende, funktionsspezifische Vorfilterung von Nachrichten oder Informationen, wie beispielsweise die Distanz zum nächsten Kreuzungsmittelpunkt. Sofern unterschiedliche Funktionen diese Funktionalität benötigen, müssten diese in einer neuen Systemkomponente implementiert werden..

Ein Relevanzfilter setzt sich aus einer Liste (DNF) von Statement-Objekten zusammen, die durch ODER Verknüpfungen zusammengesetzt werden. Die Statements ihrerseits bestehen jeweils aus einer Liste von Filterdefinition-Objekten, die durch UND-Verknüpfungen zusammengesetzt werden. Da sich jeder logische Ausdruck in eine disjunktive Normalform (DNF) umwandeln lässt, können alle Kombinationen von Filtern abgebildet werden. Eine Filterdefinition besteht aus einem Attribut, einer Relation ("lt" kleiner, "gt" größer, "eq" gleich, "ne" nicht, "le" kleiner gleich, "ge" größer gleich) und einem Wert. Ein Attribut kann über den Namen ein Attribut des Nachrichtenobjektes bezeichnen oder eine vordefinierte Filterfunktion. Der Wert enthält im Normalfall bei Attributen einen Bezeichner, bei Filterfunktionen können aber auch mehrere Bezeichner übergeben werden, die durch ein Trennzeichen "|" separiert werden.

Da der Relevanzfilter von mehreren Komponenten/Funktionen verwendet werden kann, ist die Filterfunktion selbst mehrfach instanzierbar. Bei der Implementierung sollte darauf geachtet werden, dass der Relevanzfilter nur einmalig auf eine empfangene Nachricht zugreift, um darauf sämtliche Filterdefinitionen anwenden zu können.

Funktionsweise/Ablauf

1. Subscribe: Eine Komponente erzeugt einen Relevanzfilter mit dessen Statements und Filterdefinitionen und registriert diesen zusammen mit dem C2XMessageRelevanceHandler-Service an der Service-Registry. Durch das Registrieren eines C2XMessageRelevanceHandler-Service an der Service-Registry wird die Methode addingService im Relevanzfilter aufgerufen. Anhand der übergebenen ServiceReference kann nun der C2XMessageRelevanceHandler der Komponente und die in den Service-Properties verpackten Filterkriterien bezüglich eingehender Nachrichten verwaltet werden. Zuerst prüft der Relevanzfilter den Filter und ruft die Methoden notifyFilterValidity und, im Fehlerfall, notifyFilterError auf dem Listener-Service auf. Dadurch erhält die Hauptfunktion eine asynchrone Rückmeldung über die Filterregistrierung. Werden nun Nachrichten vom Communication Client in die Umfeldtabelle durch den Aufruf von insertC2XMessage eingetragen, werden intern im Relevanzfilter die Filter der registrierten Hauptfunktionen auf Übereinstimmung geprüft und den Hauptfunktionen durch Aufruf der handleRelevanceMessage-Methode zugestellt.
2. Request: Im Programmablauf der Hauptfunktion kann ein Relevanzfilter mit dessen Statements und Filterdefinitionen erzeugt werden und dem C2XMessageRelevance-Service übergeben werden. Der C2XMessageRelevance-Service überprüft und analysiert den Filter und stellt alle gespeicherten Nachrichten, die auf die Filterkriterien passen, in einer Liste zusammen und liefert diese dem Aufrufer zurück.

Der Relevanzfilter stellt eine Hilfsfunktion zur Verfügung, mit der eine XML-Definition der Filterkriterien in ein Java-Object RelevanzFilterObjekt umgewandelt werden kann. Dieses Java-Objekt wird dann von der Applikation bei der Registrierung eines Filters an den Relevanzfilter übergeben. Die Definition der Filterkriterien in XML ermöglicht den Funktionsentwicklern eine einfache, wartbare Definition Ihrer Filter. Wegen des Rechenaufwand für das parsen der XML-Daten sollte jedoch von der Umwandlung der Filterkriterien sehr sparsam Gebrauch gemacht werden und die JavaObjekte "RelevanzFilterObjekt" durch die Funktionen gepuffert werden.

Das XML Schema, dem der XML Filter-String genügen muss, ist wie folgt definiert:

```
<?xml version="1.0" encoding="windows-1252" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns="http://www.simtd.org"
  targetNamespace="http://www.simtd.org"
  elementFormDefault="qualified">

  <xsd:element name="RelevanzFilter" type="RelevanzFilter">
    <xsd:annotation>
      <xsd:documentation>
        Relevanz Filter
      </xsd:documentation>
    </xsd:annotation>
  </xsd:element>

  <xsd:complexType name="RelevanzFilter">
    <xsd:sequence maxOccurs="unbounded">
      <xsd:element name="Statement" type="Statement"/>
    </xsd:sequence>
    <xsd:attribute name="id" use="optional" type="xsd:string"/>
  </xsd:complexType>

  <xsd:complexType name="Statement">
    <xsd:sequence maxOccurs="unbounded" minOccurs="1">
      <xsd:element name="FilterDefinition" type="FilterDefinition"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="FilterDefinition">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="attribut" type="xsd:string"/>
      <xsd:element name="relation">
        <xsd:simpleType>
          <xsd:restriction base="xsd:string">
```

```
<xsd:enumeration value="lt"/>
<xsd:enumeration value="gt"/>
<xsd:enumeration value="le"/>
<xsd:enumeration value="ge"/>
<xsd:enumeration value="eq"/>
<xsd:enumeration value="ne"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="wert" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

3.2.5 Navi/Karte

Diese Komponente stellt die Navigationsfunktionalität sowie das Kartenmaterial bereit. Diese Funktionalität wird hier vorgestellt und beschrieben.

3.2.5.1 Zieleingabe

Die Navigation stellt die Möglichkeit zur Verfügung, ein Ziel einzugeben. Die Zieleingabe erfolgt via Adressen und wird, wenn nötig, von der Navigationskomponente durch eine Spelling-Funktionalität unterstützt. Zusätzlich kann die Zieleingabe via Geo-Koordinaten erfolgen. Die Geo-Koordinaten werden entweder direkt eingegeben oder via Auswahl eines POIs (Point of Interest) als Ziel. Ein POI kann ein beliebiges Objekt (statisch oder bewegt) auf einer Karte sein.

Die Daten der Zieleingabe (Adresse oder Geo-Koordinaten) werden dann verwendet, um ein Ziel zu definieren. Weil die Zieleingabe durch Geo-Koordinaten möglicherweise zu einem nicht navigierbaren Ziel führen kann, überprüft die Navigationskomponente immer die eingegebenen Daten. Nach einer erfolgreichen Zieleingabe wird jede interessierte Komponente benachrichtigt. Das bedeutet, jede Komponente soll sich bei der Navigation für das relevante Ereignis registrieren. Zusätzlich ist es immer möglich das aktuelle Ziel abzufragen.

3.2.5.2 Routenplanung

Normalerweise wird als Startpunkt die aktuelle Position des Fahrzeugs verwendet. Es ist aber auch möglich, den Startpunkt einer Route via Adressen oder Geo-Koordinaten zu definieren. Zusätzlich kann man bei der Routenplanung via Adressen oder Geo-Koordinaten Zwischenziele definieren.

Die Navigationskomponente berechnet eine Route anhand von Kriterien wie schnellste/kürzeste Route, Vermeidung von Autobahnen, Mautstraßen und kleinen Straßen. Außerdem ist es immer möglich, den aktuellen Startpunkt, die aktuelle Zwischenziele und die aktuelle Konfiguration der Route abzufragen.

3.2.5.3 Routenberechnung

Nach der Routenplanung folgt die Routenberechnung. Die Navigationskomponente kann prinzipiell eine oder mehrere Routen berechnen und zurückliefern.

Ist die Routenführung aktiv und weicht der Nutzer von der empfohlenen Route ab, findet eine automatische Neuberechnung der Route statt. Eine Neuberechnung ist bei Eingang von zusätzlichen Informationen wie Wetterereignissen oder Verkehrswarnungen, oder bei manuellen Änderungen von Gewichten (Geschwindigkeiten und Reisezeiten auf einzelnen Straßensegmenten) auch möglich und basiert auf der alten Route.

Nach einer erfolgreichen Routenberechnung wird jede interessierte Komponente benachrichtigt. Das bedeutet, jede Komponente soll sich bei der Navigation für das relevante Ereignis registrieren.

Zusätzlich ist es immer möglich, die aktuelle Route abzufragen.

3.2.5.4 Routenführung

Die Navigationskomponente wird zusätzlich verwendet, um den Nutzer auf einer Route zu führen.

Abstrakte Beschreibungen von Fahranweisungen werden generiert, die die nächsten Manöver inklusive Richtung und Abstand bis zum nächsten Entscheidungspunkt beinhalten. Die Navigation kann auch Unterstützung für verschiedene Arten von Fahranweisungen leisten, wie z.B. Early Instruction, Prepare Instruction und Approach Instruction.

Nach der Ausgabe einer Anweisung wird jede interessierte Komponente benachrichtigt. Das bedeutet, jede Komponente soll sich bei der Navigation für das relevante Ereignis registrieren.

Zusätzlich ist es immer möglich, die aktuelle Anweisung abzufragen.

3.2.5.5 Zugriff auf die Route

Die Navigationskomponente erlaubt Zugriff auf die gesamte Route und auch auf Teilstrecken (z.B. wenn man Zwischenziele definiert hat). Generell, wird eine Route als Liste von Segmenten beschrieben.

Für die gesamte Route werden folgende Daten zur Verfügung gestellt und können abgefragt werden:

- Dauer der Route
- Länge der Route
- Liste von Zwischenzielen
- Segmentliste der Route (bzw. Teilstrecken) mit vollständigem Zugriff auf sämtliche Segment-Attribute in der Karte anhand von LinkIDs

- Startzeitpunkt als Zeitstempel
- Nächstes Ziel bei Zwischenzielen
- Aktuelles Ziel

3.2.5.6 Elektronischer Horizont

Die Navigationskomponente berechnet einen elektronischen Horizont und stellt eine geführte bzw. nicht geführte Route zur Verfügung. Der Elektronische Horizont basiert auf digitalen Kartendaten und ermittelt was den Fahrer auf der vorausliegenden Strecke erwartet. Wenn keine Route vorliegt, wird der Zugriff auf den Most Probable Path (MPP) angeboten. Dadurch kann man auf die Geometrie und auf Kartenkantenattributen auf der vorausliegenden Strecke zugreifen.

Die Länge des elektronischen Horizonts ist konfigurierbar.

Für jedes Segment werden sämtliche Attribute (die werden später beschrieben, siehe Tabelle 10) mitgeliefert.

Notwendig ist auch eine Schnittstelle der Navigation, die prüft, ob ein bekannter Punkt auf der aktuellen Route bzw. MPP liegt. Es wird zusätzlich die reale Entfernung bis zum eingegebenen Punkt berechnet.

3.2.5.7 Zugriff auf Attribute von Segmenten/Kanten

Für jedes Kartensegment kann man die Geometrie (Shapepoints) und folgende Attribute abfragen:

- Länge
- Ausrichtung (Primary Heading)
- Straßenklasse, Straßename, Straßensummer, Straßentyp (z.B. Highway)
- Lage (z.B. in City)
- Geschwindigkeitsbeschränkungen (zeitlich, wetterbedingt)
- Geschwindigkeiten (empfohlen, erwartet)
- Brücken-Flag, Tunnel-Flag, Ramp-Flag, Complex-Intersection-Flag
- TMC-Referenzierung
- Krümmung
- Anzahl der Spuren

Außerdem gibt es die Möglichkeit, auf die Geometrie zuzugreifen, die zwischen einem Start- und einem Endpunkt liegt.

Es gibt zusätzlich eine Schnittstelle, mit der die Gewichte auf Kanten verwaltet werden können (d.h. ändern, zurücksetzen) und dadurch die Routenberechnung beeinflusst werden kann.

3.2.5.8 Fahrzeugpositionierung

Die Position des Fahrzeugs wird zyklisch von der Komponente „Bessere Ortung“ geliefert und gibt es deswegen keinen direkten Zugriff auf die GPS-Position. Die Komponente „Bessere Ortung“ kann indirekt über die VAPI zugegriffen werden (siehe Abbildung 3-35). Input für die Positionierung sind Koordinaten oder NMEA-Strings.

Das eigene Fahrzeug bzw. eine Position wird auf der Karte gematcht und die Navigationskomponente bietet Zugriff auf die gematchte Position. Diese gematchte Position beinhaltet folgende Attribute:

- Geo-Koordinaten (Latitude, Longitude)
- Kanten-ID und Offset

Alle Geo-Koordinaten werden in einer Form von WGS84 Koordinaten geliefert. Eine Reverse-Geokodierung wird nicht benötigt.

Es ist keine fahrstreifengenaue Positionierung möglich.

3.2.5.9 Grafische Kartenschnittstelle

Die Navigationskomponente stellt auch eine Karte als Bild (im BMP-Format) zur Verfügung. Die aktuelle Position des Fahrzeugs und auch die aktuelle Route werden immer auf der Karte angezeigt.

Es gibt auch die Möglichkeit, über eine Schnittstelle die Karte zu verwalten und Attribute (siehe folgende Liste) zu modifizieren. Folgende Funktionalität ist möglich, um die Karte zu verwalten:

- Angabe von Breite und Höhe der Karte in Pixeln und Änderung während der Laufzeit.
- Änderung des Mittelpunkts des Kartenausschnitts über Geo-Koordinaten veranlasst durch den Benutzer oder die Funktionen.
- Änderung der Zoomstufe (Zoom-In, Zoom-Out) veranlasst durch den Benutzer oder die Funktionen. Die in der Karte anzuzeigende Information wird automatisch und abhängig von der Zoomstufe geändert.
- Änderung der Orientierung der Kartenansicht veranlasst durch den Benutzer oder die Funktionen.

Auf der Karte werden auch zusätzliche Informationen wie POIs (Warnungen und andere sich bewegende Fahrzeuge) angezeigt. Die in der Karte angezeigten Informationen werden über eine Schnittstelle dynamisch geändert.

Die Verwendung einer Tag- und Nachtversion wird nur als Option beim Starten der Navigation angeboten und ist während der Laufzeit nicht änderbar.

Es ist auch möglich, Streckenabschnitte bzw. ganze Bereiche zu markieren bzw. einzufärben. Die Einfärbung ist konfigurierbar und kann wieder entfernt werden.

Es wird zusätzlich benötigt, dass man eine Auswahl bezüglich der anzuzeigenden Informationen auf der Karte bzw. auf der Route trifft. Die Informationen, die zur Laufzeit geändert werden sollen, sind die Folgenden:

- Benutzerdefinierte POIs (nach Kategorien) alternativ ein- und ausblenden
- Kanteneinfärbung ein- bzw. ausschalten

Die benutzerdefinierte Kategorien von POIs werden später in der Projektlaufzeit definiert. Hier kann man eine vorläufige Liste finden:

- Positionen/Richtungen von Ego-Fahrzeug und Einsatzfahrzeug
- Nachbarschaftstabelle, Positionen anderer Fahrzeuge in Funkreichweite
- Baustellenanfang und Baustellenende
- Verkehrereignisse
- Gefahrenhinweis
- Wetterereignisse
- Position und Art von Veranstaltungen und Kommunalereignissen
- Von Veranstaltungen beeinflusste (z.B. gesperrte) Gebiete
- Parkraum (bewirtschaftet und mit Informationen über die Verfügbarkeit/Auslastung)

3.2.5.10 Verwaltung von mehreren Karten

Die Navigationskomponente in sim^{TD} hat keine eigene HMI. Alle grafischen Operationen werden durch die HMI des gesamten Systems laufen. Es ist trotzdem notwendig, dass die Navigation nicht nur die letzte Karte speichert, die zur Anzeige geschickt wurde; es wird eine Reihe von Karten und Layern verwaltet. Jede Karte wird mit einer ID versehen. Die Anzahl der zu verwalteten Karten ist konfigurierbar sein. Hintergrund ist, dass nur die HMI weiß, welche Karte im Moment angezeigt wird und manche Karten, die die Navigationskomponente erzeugt, können aufgrund zu geringer Priorisierung nicht angezeigt werden.

Es wird auch ein Layer-Konzept geben. Das bedeutet, dass es eine Grundkarte gibt und die POIs und Kanteneinfärbungen werden als separate Layers geliefert. Alle Einfärbungen werden in einem Layer zusammengefasst, aber die POIs werden in mehrere Layers aufgeteilt. Alle Karten (Grundkarte und Layers) werden durch eine ID identifiziert.

Damit das oben genannte Konzept funktioniert, braucht man Umrechnungsfunktionen von Bildschirm-Koordinaten zu Geo-Koordinaten und zurück. Dazu wird folgende Funktionalität benötigt:

- Berechnung von Geo-Koordinaten anhand relativer Pixel-Koordinaten
- Berechnung von relativen Pixel-Koordinaten anhand von Weltkoordinaten

3.2.5.11 Verwaltung von POIs

Es ist möglich, benutzerdefinierte POIs auf der Karte bzw. entlang der Route ein- und ausblenden. Diese POIs sollen erst erzeugt und dann verwaltet werden.

Die Navigationskomponente bietet eine Verwaltung von benutzerdefinierten POIs und entsprechenden Kategorien. Folgendes ist möglich:

- Benutzerdefinierten POIs (z.B. Warnungen) auf der Karte platzieren
- POIs hinzufügen, Unterstützung benutzerdefinierter Piktogramme, POI-ID erzeugen
- POIs aktualisieren (Position, Piktogramm)
- POIs löschen
- POI-Kategorien verwalten (wahrscheinlich vordefiniert).
- Bei Drücken auf den Bildschirm: POI finden und ID zurückliefern.
- Ein- und ausschalten von POI-Kategorien auf der Karte.

3.2.5.12 Sonderfall Testgelände und Versuchsplanung

Es besteht die Anforderung, dass eine navigationsfähige Geometrie des Testgeländes und insbesondere des künstlich angelegten Knotenpunktes in den Navigationsgeräten der Testfahrer verfügbar ist. Deswegen wird das Testgelände digitalisiert und wird navigierbar sein, d.h. es werden mehrere (mindestens vier) Punkte definiert, die die Testfahrer als Start und Zielpunkte für die Navigation eingeben können (idealerweise sollte auch eine längere Teststrecke programmierbar sein (nicht zwingend während der Fahrt oder durch den Fahrer), die mehrfach den Testknotenpunkt durchläuft. Das wird durch das Setzen von Zwischenzielen ermöglicht.

Die entwickelte Funktionalität wie Routenplanung, grafische Darstellung und Zugriff auf Kartenmaterial werden auch für die Versuchsplanung zur Verfügung gestellt. Das bedeutet, dass es in der Versuchszentrale möglich ist, Routen festzulegen und anschließend in jedem Fahrzeug zuladen.

3.2.5.13 Logging

Die Navigation wird während der Fahrt und durch die verfügbare sim^{TD} Logging Funktion bestimmte Daten loggen. Es werden mindestens zwei Logging-Einträge erfolgen: wenn eine Navigation gestartet oder gestoppt wird.

Die Navigationslösung liefert Informationen darüber, ob sie aktiv ist und fehlerfrei läuft (Logging & Monitoring).

3.2.5.14 Architektur

Die Navigationskomponente liegt auf der Application Unit (AU) im Fahrzeug und wird eine eingekaufte Lösung sein. Das bedeutet, dass die Funktionalität als nativer Code vorliegen wird. Deswegen ist eine Komponente erforderlich, die die Navigationsfunktionalität in OSGi zur Verfügung stellt. Die Navigation stellt eine Schnittstelle für andere Komponenten und Funktionen zur Verfügung.

Die folgenden Abbildungen geben eine Übersicht über die Navigationskomponente.

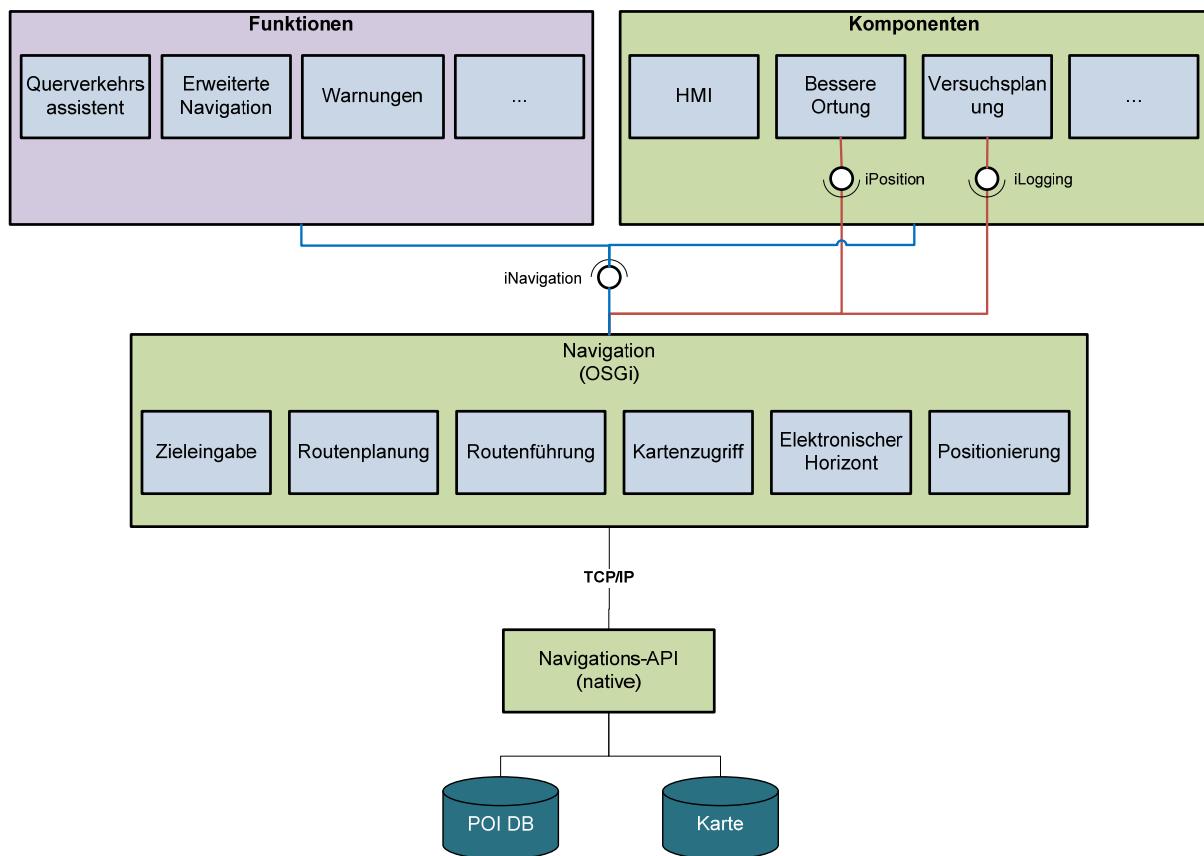


Abbildung 3-29: Übersicht Navigationskomponente

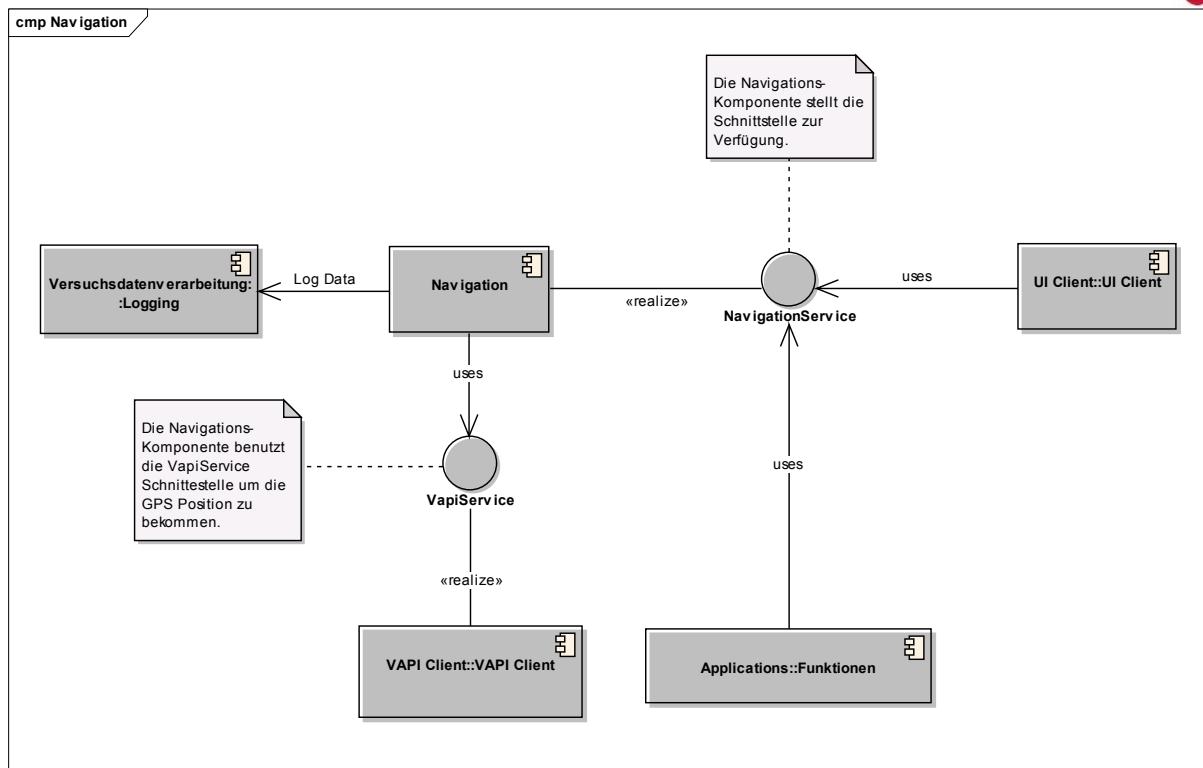


Abbildung 3-30: Übersicht Komponenten-Diagramm

3.2.5.15 Datenspezifikation

Die folgenden Abbildungen beschreiben kurz alle notwendigen Daten und bekannten Datenflüsse. Für eine detaillierte Spezifikation siehe Annex 2.

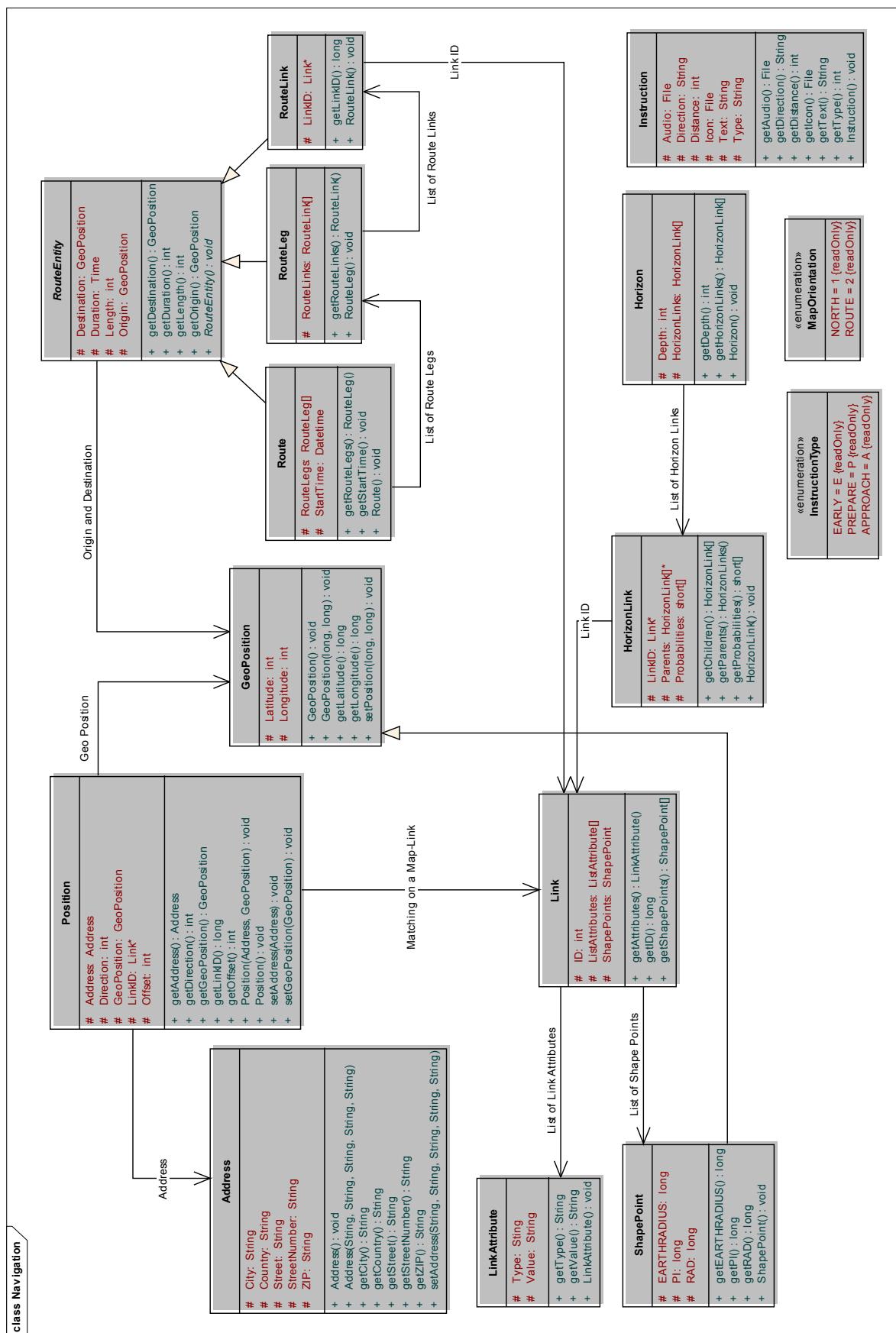


Abbildung 3-31: Übersicht Klassendiagramm

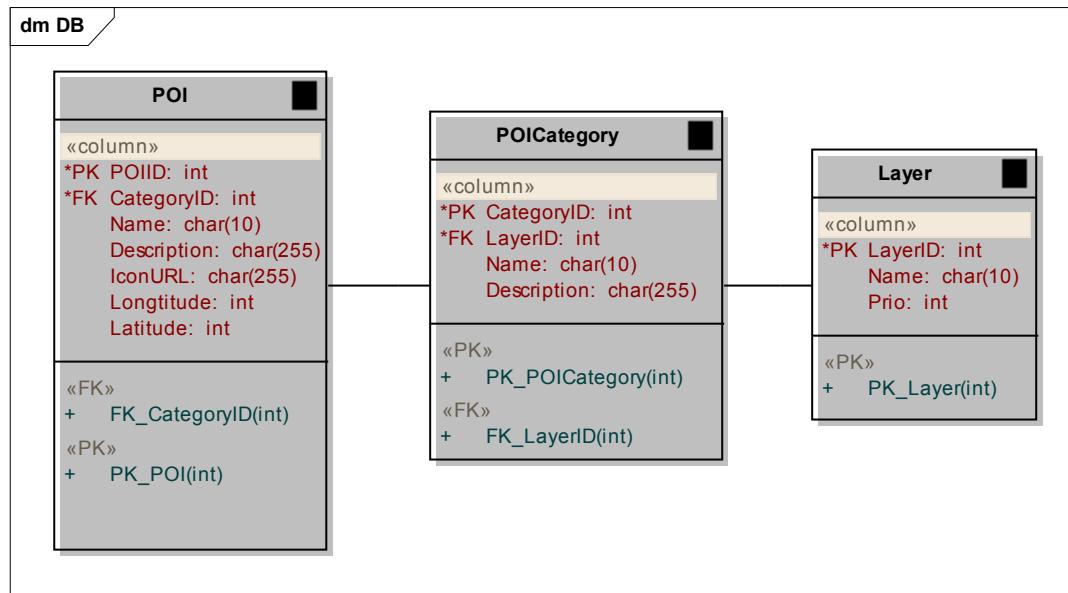


Abbildung 3-32: Übersicht DB-Diagramm

Datenflüsse zwischen Navigation und anderen Komponenten bzw. Funktionen

Die Navigationskomponente stellt folgende Daten für Komponenten und Funktionen zur Verfügung:

- Gematchte Position des Fahrzeuges
- Route
- Fahranweisungen
- Daten an Kartenkanten
- Elektronischer Horizont
- Grundkarte und POIs/Segmenten-Einfärbung als Layers (BMP-Bilder)

Die Navigationskomponente nimmt die Rohdaten der aktuellen Position von der Komponente „Bessere Ortung“ und matched diese auf die Karte.

Die Navigationskomponente erzeugt Logging-Einträge für bestimmte Ereignisse.

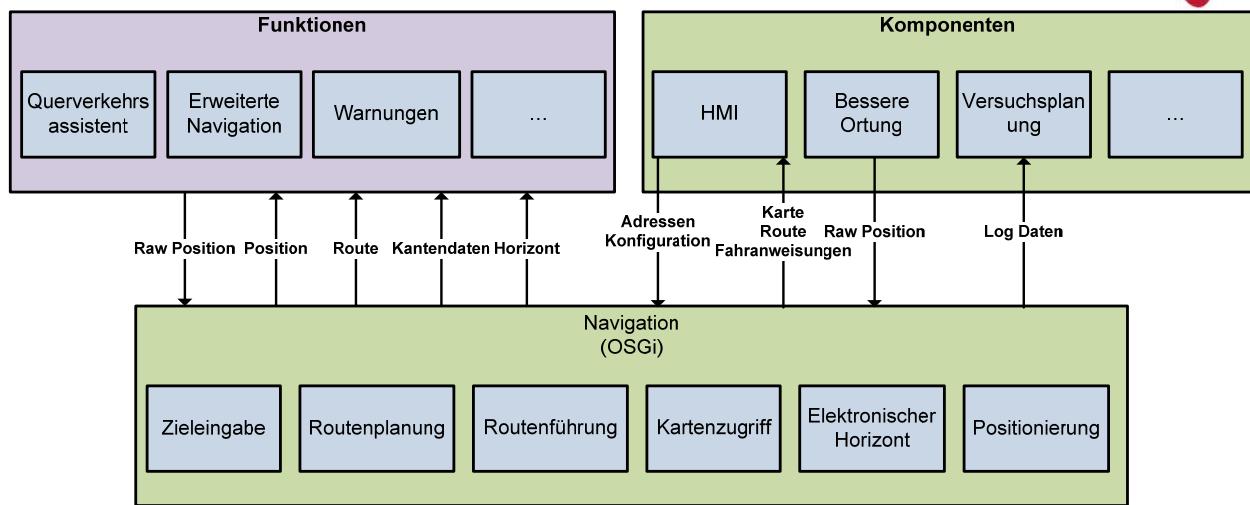


Abbildung 3-33: Übersicht Datenflüsse

3.2.5.16 Schnittstellen

Die Navigationskomponente kommuniziert mit der HMI des Fahrzeugs. Diese Kommunikation ist bilateral. Die HMI nimmt Navigationsfunktionen direkt in Anspruch (z.B. Routenberechnung, Routenführung). Auf der anderen Seite schickt die Navigation an die HMI beliebige Informationen zur Anzeige.

Zusätzlich werden diverse Hauptfunktionen und Komponenten voraussichtlich auf die Navigationsfunktionen und Kartenmaterial zurückgreifen.

Die Navigationskomponente wird die Schnittstelle von der Komponente „Bessere Ortung“ benutzen, um die eigene Position zu bestimmen.

Bereitgestellte Schnittstellen

Die folgende Abbildung beschreibt kurz alle Methoden der Navigationsschnittstelle. Für eine detaillierte Spezifikation siehe Anhang 2.

class Structure

```

    «interface»
navigation::NavigationService

+ addLayer(String, int) : int
+ addMapPolygon(GeoPosition[], int, int, int, int, boolean) : int
+ addMapPolyline(GeoPosition[], int, int, int, int) : int
+ addMapPolyline(long, int, int, int, int) : int
+ addPOI(long, long, String, String, int) : int
+ addPOICategory(String, int) : int
+ calculateRoute(boolean) : Route
+ clearRoute() : void
+ clearRouteCost(long, long, long, long) : void
+ clearRouteCost(long) : void
+ deleteMapPolygon(int) : void
+ deleteMapPolyline(int) : void
+ findPOI(long, long) : int[]
+ getCity(String, String) : String[]
+ getCounty(String) : String[]
+ getCurrentInstruction() : Instruction
+ getCurrentPosition() : Position
+ getDestination() : Position
+ getDistanceInMPP(long, long) : int
+ getDistanceInRoute(long, long) : int
+ getHorizon() : Horizon
+ getLinkAttributes(int) : LinkAttribute[]
+ getLinkIdsFromTo(long, long, long, long) : Long[]
+ getLinkShapePoints(int) : ShapePoint[]
+ getMapPosition(String) : Position
+ getMapPosition(long, long) : Position
+ getMaps() : File[]
+ getMPP() : HorizonLink[]
+ getNextInstruction() : Instruction
+ getNextVia() : Position
+ getRoute() : Route
+ getRouteConfig() : String
+ getStart() : Position
+ getVias() : Position[]
+ isInMPP(long, long) : boolean
+ isInRoute(long, long) : boolean
+ loadRoute(String) : void
+ removeLayer(int) : void
+ removePOI(int) : void
+ removePOICategory(int) : void
+ saveRoute(Position[], String)
+ screen2world(int, int, int) : GeoPosition
+ setDestination(String, String, String, String, String) : void
+ setDestination(long, long) : void
+ setMapCenter(long, long) : void
+ setMapOrientation(int) : void
+ setRouteConfig(String) : void
+ setRouteCost(long, long, long, long, int) : void
+ setRouteCost(long, int) : void
+ setStart(long, long) : void
+ setStart(String, String, String, String, String) : void
+ setVias(Position[]) : void
+ setZoomLevel(int) : void
+ showAllPOIs() : void
+ showPOICategory(boolean, boolean, int) : void
+ startGuidance() : void
+ stopGuidance() : void
+ updatePOI(int, long, long, String, int)
+ world2screen(long, long, int) : int[2]

```

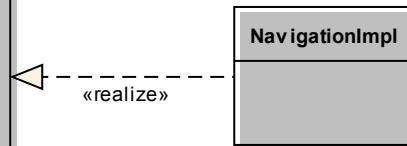


Abbildung 3-34: Übersicht Navigationsschnittstelle

Genutzte Schnittstellen

Die Navigationskomponente benutzt die Schnittstelle der Komponente „Bessere Ortung“. Diese liefert die aktuelle Raw-Position, einen Geschwindigkeitsvektor und die Gierrate. Die Navigation wird sich für den Service anmelden (via VAPI) und die gelieferten Daten benutzen, um die Position des Fahrzeugs auf der Karte zu bestimmen (Map Matching).

Die Navigation benutzt zusätzlich ein Logging-API, um alle notwendigen Inhalte den „Auswertern“ zur Verfügung zu stellen. Es sollten bestimmte Logging-Einträge erfolgen. Für eine detaillierte Beschreibung siehe Abschnitt 3.2.5.15.

Abläufe

Bessere Ortung

Die Navigationskomponente hat keine direkte Verbindung zu dem GPS-Signal und benutzt deswegen die Position, die die Komponente „Bessere Ortung“ liefert. Da die „Bessere Ortung“ nicht auf der Application Unit liegt, wird die Position an die Navigation via die Komponente „VAPI Client“ geliefert. Die Navigation matched dann die Position auf der Karte und stellt die gematchte Position den Funktionen und Komponenten zur Verfügung. Die folgende Abbildung beschreibt diesen Prozess.

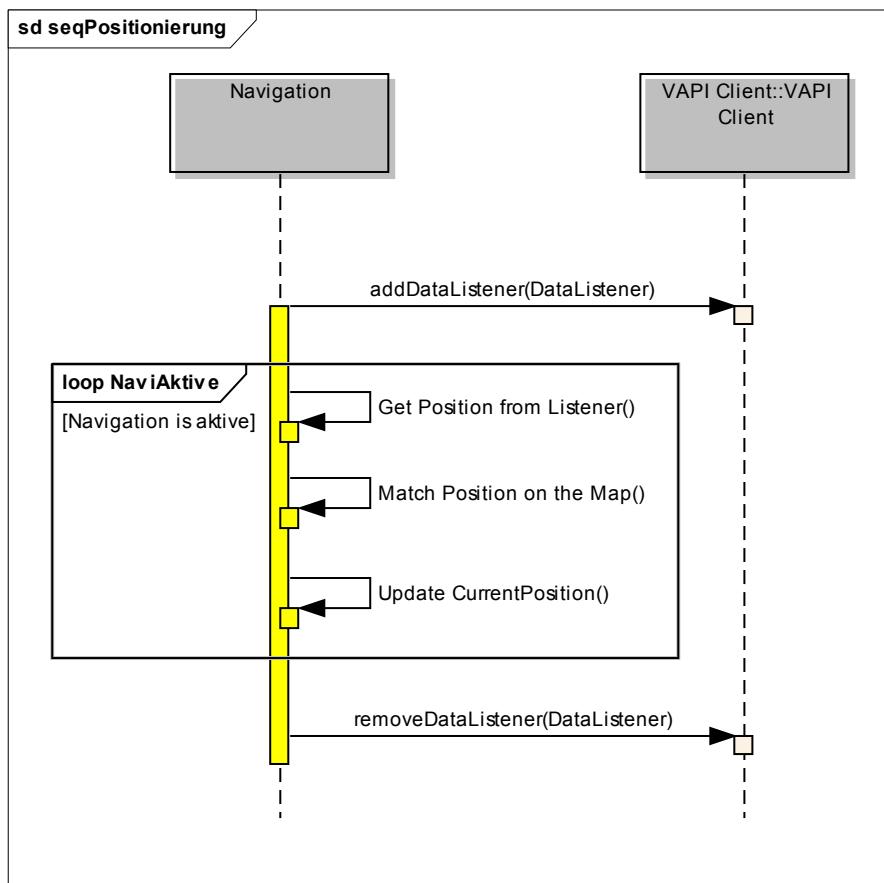


Abbildung 3-35: Ablauf „Aktuelle Position aktualisieren“

Logging

Die Navigationskomponente soll sämtliche Logging-Einträge schreiben. Log-Files werden beim Hochfahren, Runterfahren und diversen Fehlern produziert. Ein Logging-Eintrag wird an die Logging-Komponente in folgenden Fällen geschickt.

- Erzeugung eines POIs
- Routenberechnung
- Routenführung (Aktivierung und Deaktivierung)

Die folgende Abbildung beschreibt diesen Prozess.

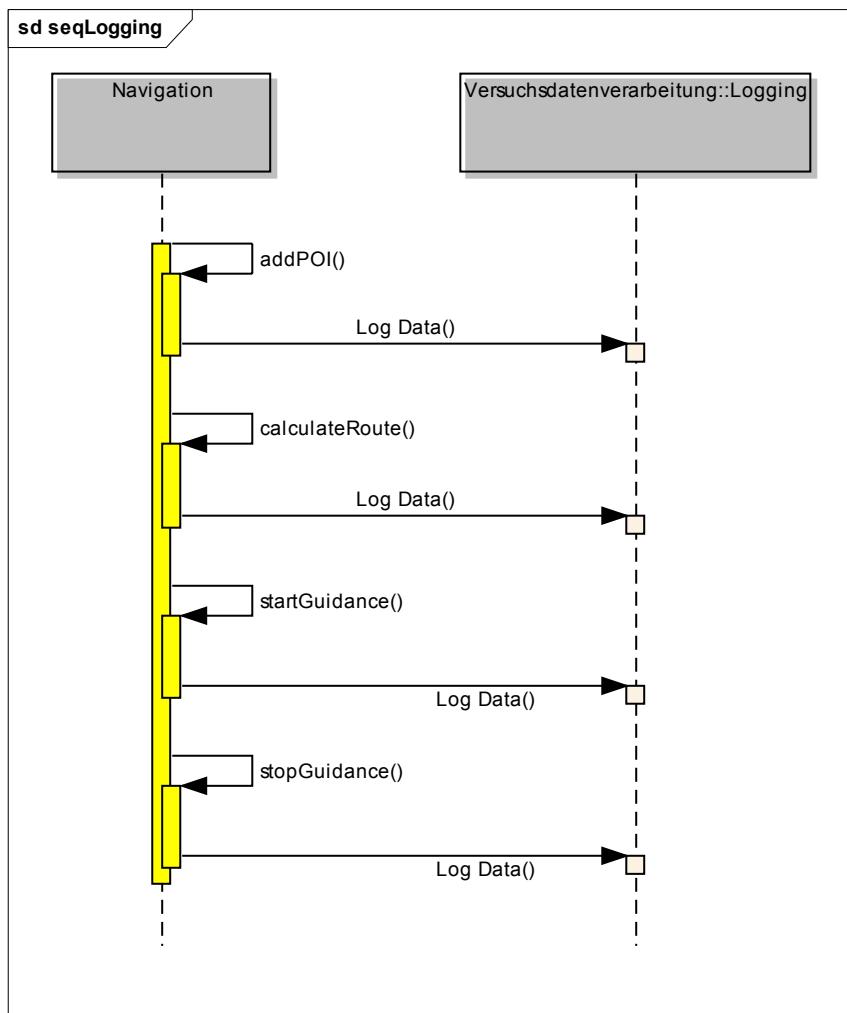


Abbildung 3-36: Ablauf „Logging-Einträge“

HMI

Spelling Funktionalität/Zieleingabe

Die Navigationskomponente stellt die Spelling-Funktionalität zur Verfügung. Die folgende Abbildung beschreibt diesen Prozess.

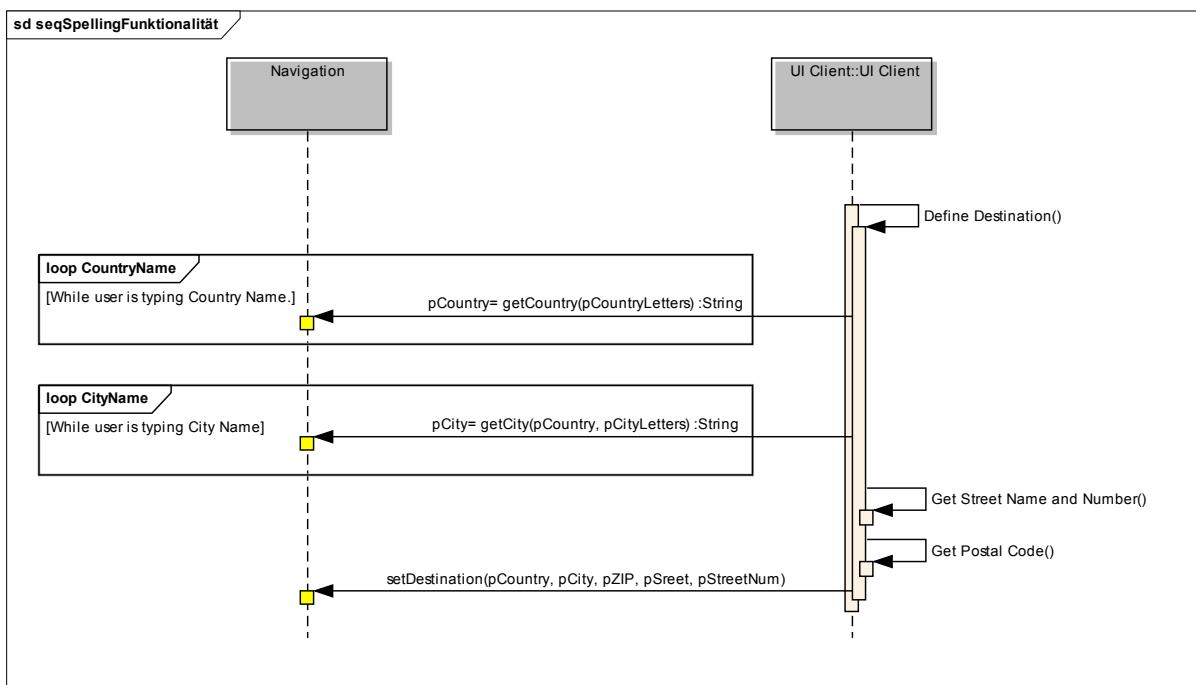


Abbildung 3-37: Ablauf „Zieleingabe“

Routenplanung

Die Navigationskomponente stellt die Routenplanungsfunktionalität zur Verfügung. Die folgende Abbildung beschreibt diesen Prozess.

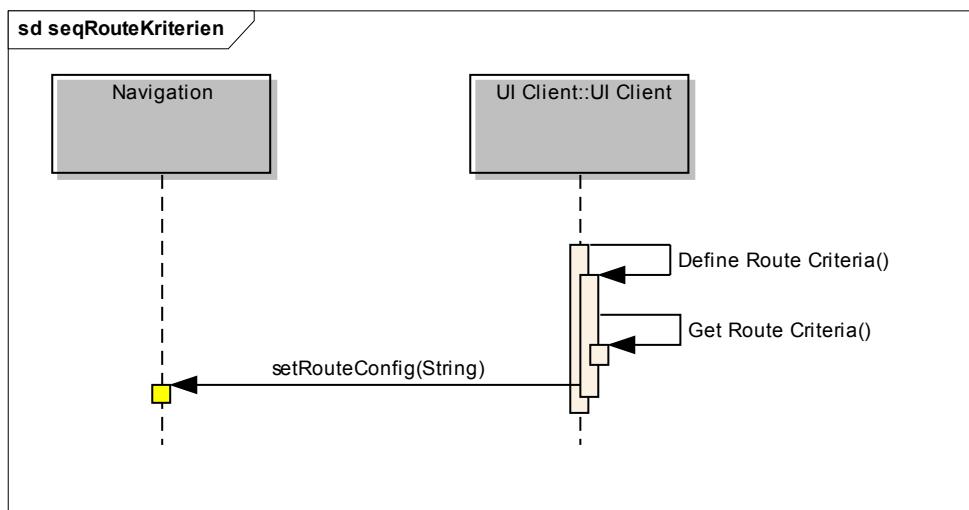


Abbildung 3-38: Ablauf „Routenkriterien“

Routenberechnung

Die Navigationskomponente stellt die Routenberechnungsfunktionalität zur Verfügung. Die folgende Abbildung beschreibt diesen Prozess.

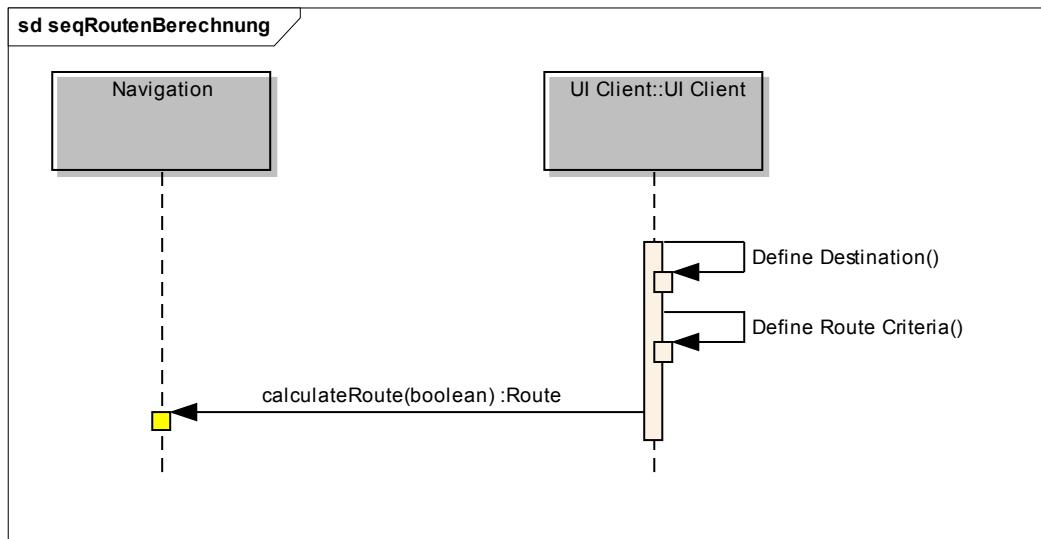


Abbildung 3-39: Ablauf „Routenberechnung“

Routenführung

Die Navigationskomponente stellt die Routenführungsfunktionalität zur Verfügung. Die folgende Abbildung beschreibt diesen Prozess.

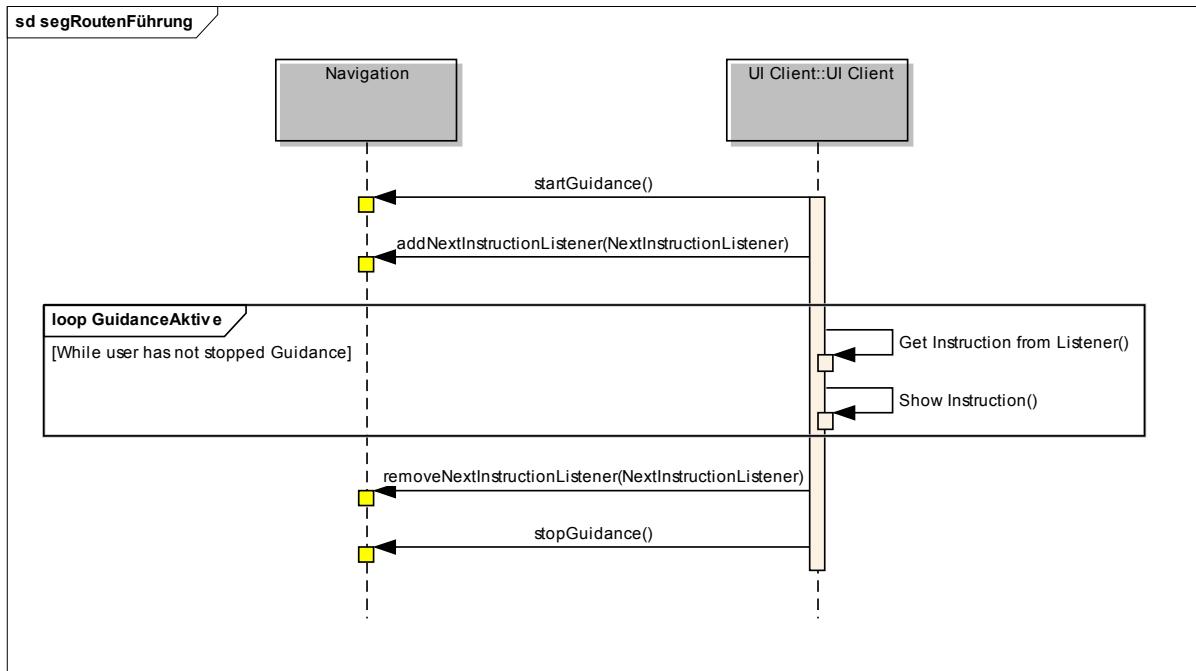


Abbildung 3-40: Ablauf „Routenführung“

Funktionen

Diverse Funktionen nehmen Navigationsfunktionalitäten (wie POIs platzieren, Karten-Segmente einfärben, Kartenattribute ändern, Kantenattribute abrufen, Horizont abrufen, etc.) direkt in Anspruch.

Diese Abläufe werden von den Funktionen beschrieben.

3.2.5.17 Offene Punkte

Die Navigations-Schnittstelle ist wegen der bevorstehenden Beauftragung noch nicht mit dem Unterauftragnehmer abgeklärt. Es gibt deswegen einige Punkte die noch diskutiert und die Ergebnisse festgelegt werden sollen. Die ersten Abstimmungen sind schon gelaufen.

- Transparenter Zugriff auf normale Straßenkarte und genauere Kreuzungskarte.
- POI Funktionalität:
 - Mechanismus zum Eintragen neuer POIs / senden von POIs an die Karte über HMI
 - Flag, um einzelne POIs auf sichtbar / unsichtbar zu setzen
 - Mechanismus zur Notifikation von Funktionen bei Klick auf einen POI und Generierung eines Präsentationsauftrags
 - Anfrage: welche POIs definiert sind.
 - POI Filterung: Berechnung der Relevanz/des Abstandes von POIs mit GPS Positionen bezogen auf den Standort, die Zielposition und die berechnete Route zum Ziel benötigt. Die Relevanz kann z.B. die Entfernung zum Bezugspunkt oder der Bezugsroute sein.
- Die Navigation bietet eine Listener-Schnittstelle, die einer anderer Komponente Trigger gibt, wenn ein bestimmter Abstand zu einem Knoten (z.B. BAB AS) oder einem Abbiegevorgang erreicht ist.
- Methoden und Parameter abstimmen.
- Thema Ressourcenauslastung
 - Welche Funktionen greifen wie häufig auf die Navigation zu? Wie soll auf die Navigation zugegriffen werden?
 - Ressourcenauslastung der Kantenfärbung muss geklärt werden.
 - Performance des E-Horizon muss getestet werden.
- Die Zeitabstände, in den die Navigation die Daten zur Verfügung stellt, sollen definiert werden.
- Thema Zwischenspeichern von Ergebnissen bei Kartenabfragen.
- Aufteilung von der Navigations-Schnittstelle in mehrere logische Schnittstellen.

3.2.6 HMI

Siehe Deliverable D22.2 (Formale HMI-Spezifikation)

3.2.7 Security

Auf der AU des Fahrzeugs und auf der AU der ITS Roadside Station sind zwei Komponenten der IT-Sicherheit untergebracht.

3.2.7.1 Kommunikationsabsicherung Client

Neben der Absicherung von Daten auf Vermittlungsschicht gibt es für Anwendungen auch die Möglichkeit, die Kommunikationsabsicherung von der AU aus zu nutzen. Die Absicherung von anwendungsspezifischen Nachrichten erfolgt über die Nutzung der Security-Nachrichten (in Anlehnung an IEEE 1609.2) wie sie in Kapitel 2.5.4.2 beschrieben wird.

Bei jeder Signierung, Verifikation oder Verschlüsselung auf Anwendungsschicht baut der Security Daemon Client zum Security Daemon der CCU eine Verbindung auf. Im Gegensatz zur Absicherung der C2X-Kommunikation wird bei der Nutzung auf Anwendungsebene das sichere Nachrichtenformat in Anlehnung an IEEE 1609.2 genutzt. Da jede Nachricht in diesem Format übertragen wird, muss zwingend beim Empfang der Nachricht der Security Daemon Client aufgerufen werden.

3.2.7.2 Plausibilitätsprüfer

Die Plausibilitätsprüfung der ITS Ad-hoc-Kommunikation wird auf der AU mit der Komponente Plausibilitätsprüfer durchgeführt. Um zusätzliche Informationen aus den C2X-Nachrichten nutzen zu können, ist diese Komponente auf der AU platziert. Die grundlegende Aufgabe der Plausibilitätsprüfung ist die Beurteilung der Mobilitätsdaten eingehender C2X-Nachrichten wie in Abschnitt 2.5.3.2 beschrieben.

Wie in Abbildung 3-41 dargestellt, wird der Plausibilitätsprüfer von dem Communication Client angesprochen. Alle eingehenden Nachrichten werden beim Erhalt durch den Communication Client als C2X-Nachrichtenobjekt an den Plausibilitätsprüfer gesendet und dort geprüft. Das Ergebnis wird in das Nachrichtenobjekt geschrieben und zurückgegeben, sodass es anschließend in die Umfeldtabelle eingespeist werden kann. Für eine erweiterte Prüfung hat die IT-Sicherheitskomponente die Möglichkeit, auf Kontextinformationen der VAPI zurückzugreifen.

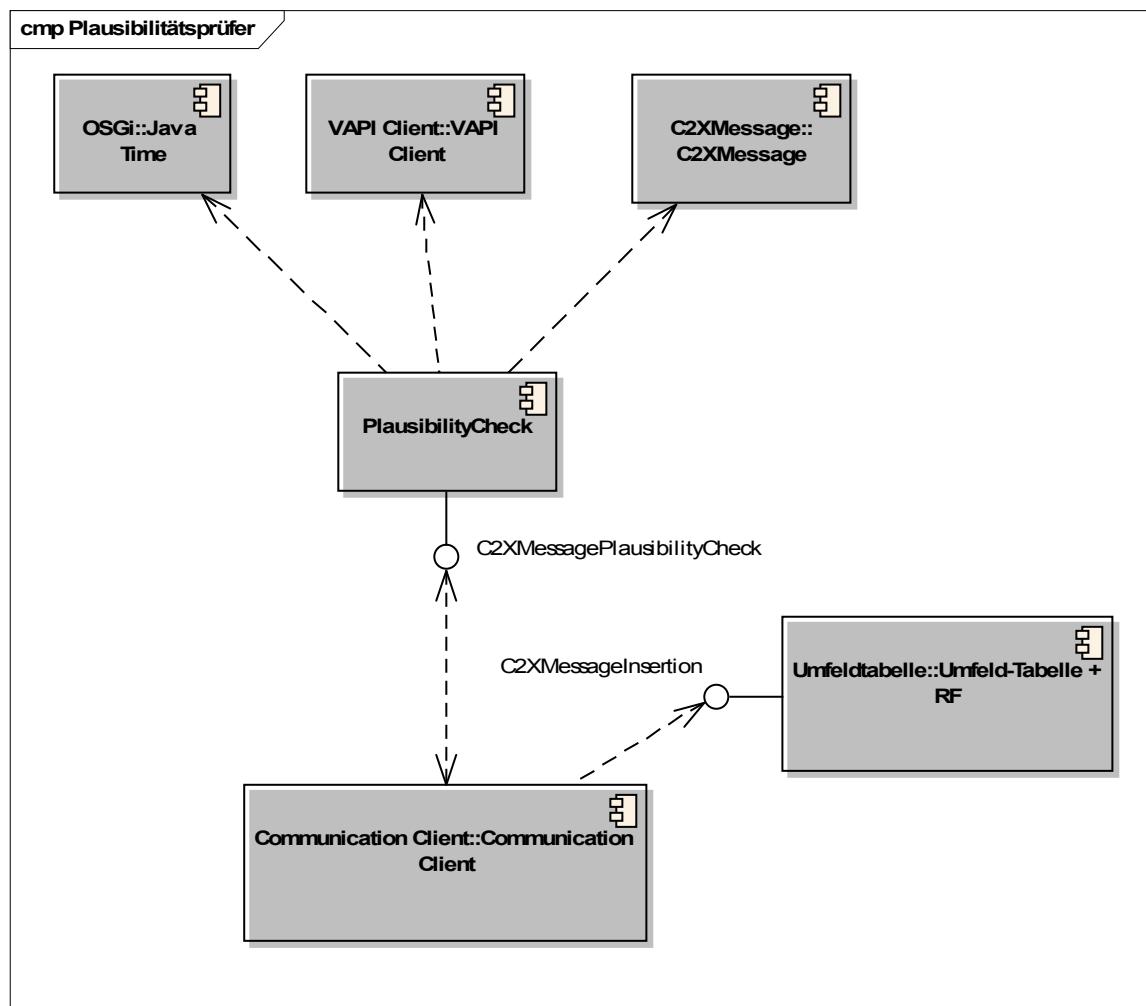


Abbildung 3-41: Plausibilitätsprüfer auf der Vehicle AU

Weitere Details zur Plausibilitätsprüfung können in Deliverable D21.5 in Kapitel 5.3.4 nachgelesen werden.

3.2.7.3 Verteildienste für individuelle und allgemeine Sicherheitsparameter

Für die Verteilung von Basisidentitäten und Pseudonymen wird ein Verteildienst für individuelle Sicherheitsparameter als Funktion 5.1.1 realisiert. Bei der Initialisierung der IVS bzw. IRS baut der Dienst eine Verbindung zur Versuchszentrale auf, um ein Basiszertifikat anzufordern. Ebenso wie in regelmäßigen Abständen eine Verbindung zur Versuchszentrale über diese Funktion aufgebaut um Pseudonyme zur aktiven Kommunikation abzurufen. Da die Abfrage dieser Parameter nur für berechtigte Fahrzeuge erfolgen darf, wird eine Konfiguration des Fahrzeugsystems bei jeder Anfrage mit übertragen. Im sim^{TD}-Konzept werden das die Versionen der eingesetzten OSGi-Bundles sein.

Die Funktion 5.1.2 ist für das Abrufen allgemeiner Sicherheitsparameter zuständig, die nicht einer besonderen Authentifizierung unterliegen. Als Anwendungsfall wird die Übertragung von Revokationslisten umgesetzt.

3.2.8 Positionskettenverarbeitung (TraceHandling)

Die Positionskettenverarbeitung stellt im Wesentlichen ein Bundle zur Verfügung, um mit Positionsketten zur Ortsreferenzierung (siehe Datenelement Traces in D21.4) umzugehen. Diese Positionsketten werden in Gefahrenmeldungen und bei Verkehrsregeln verwendet.

Um beim wiederholten Matchen auf die Positionsgröße nicht immer wieder entsprechende Koordinatentransformationen durchführen zu müssen und so erheblich Rechenzeit zu sparen, wird zunächst der Trace in ein geeignetes Format `ExtTrace` konvertiert und sollte in der aufrufenden Komponente zu weiteren Verwendung zwischengespeichert werden.

Es ist anzumerken, dass die Behandlung der Positionsketten in dem Bundle unabhängig von der Karte stattfindet. Ein Matchen der Positionsketten auf die Karte findet nicht statt. Außerdem ist anzumerken, dass es neben den hier beschriebenen Positionsketten auch noch andere ähnliche Daten gibt, die zur Beschreibung der Geschwindigkeitsprofile entlang einer Strecke genutzt werden. Hier kommen andere Formate zum Einsatz und die Ketten werden nach anderen Kriterien erstellt. Daher sind diese Daten getrennt in den Funktionen zu behandeln.

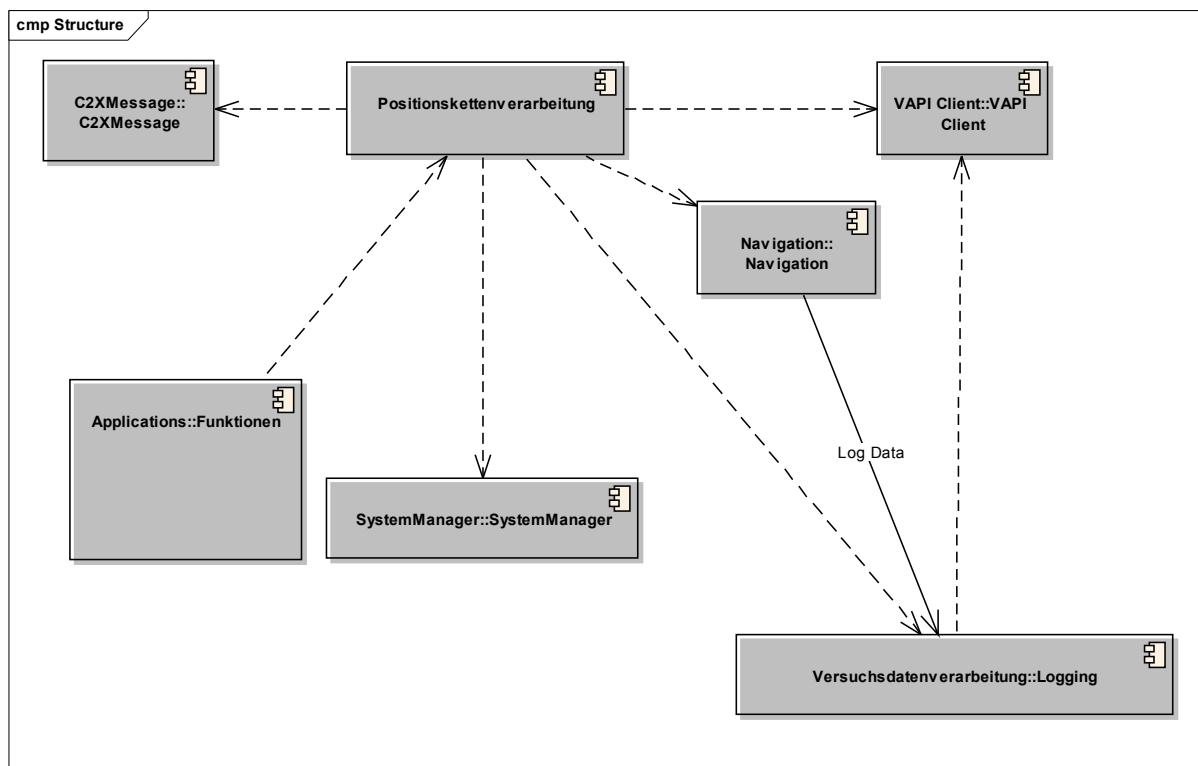


Abbildung 3-42: Einordnung der Komponente „Positionskettenverarbeitung“

Importierte Schnittstelle

Die Funktion ruft über die VAPI die Position der verbesserten Ortung ab. Zusätzlich wird noch der aktuelle Straßenname von der Navigation abgerufen.

Außerdem importiert die Schnittstelle die Objektdefinitionen für den Trace aus der Komponente „C2X Message“ nutzt die Komponente die üblichen Schnittstellen zum Logging, System-Management und zur Parametrierung.

Exportierte Schnittstellen

Die Komponente exportiert eine Schnittstelle mit den folgenden Methoden. Diese Schnittstelle wird von den Hauptfunktionen und von dem Relevanzfilter verwendet.

```
asn1.CommonElements.ReferencePosition    getCurrentReferencePosition(boolean
setStreetName);
```

Die Funktion gibt die Referenzposition als Objekt des ASN.1-Bundles zu der aktuellen Fahrzeugposition zurück. Ist setStreetName true, wird auch der Straßename eingetragen. Da der Name viel Bandbreite benötigt, sollte von der Option vorsichtig Gebrauch gemacht werden. Die Referenzposition enthält die Fahrtrichtung und die horizontale Genauigkeit.

```
asn1.CommonElements.Trace getCurrentTrace(
asn1.CommonElements.ReferencePosition refMilage)
asn1.CommonElements.Trace getCurrentTrace(
asn1.CommonElements.ReferencePosition refMilage,
asn1.CommonElements.ReferencePosition extensionEnd)
```

Gibt ein Trace-Objekt als Objekt des ASN.1-Bundles zurück. Dabei muss die Referenzposition nicht an der aktuellen Fahrzeugposition liegen sondern kann eine beliebige Position sein, die das Fahrzeug kürzlich passiert hat. Gleiches gilt für das Ende der Extension, das heißt die Positionsfolge, die von der Referenzposition entlang der Fahrspur bis zu der gegebenen Position extensionEnd führt. Sollte die Positionsfolge der Zufahrt zu der Referenzposition oder die extensionEnd nicht aus den abgelegten Positionsdaten gebildet werden können, so wird die entsprechende Positionsfolge im Trace nicht codiert und nur die Referenzposition oder das extensionEnd als einzelner Punkt kodiert.

```
ExtTrace convertTrace(asn1.CommonElements.Trace asn1Trace)
```

Konvertiert das Objekt eines Traces aus dem ASN.1-Bundle in ein Objekt für weiteres Matching. Dieses Objekt sollte auch für weitere Aufrufe in der aufrufenden Funktion gespeichert werden.

```
TraceMatch matchEgo2Trace(extTrace trace)
TraceMatch matchEgo2Trace(extTrace trace, TraceMatch oldMatch)
```

Matcht die aktuelle Fahrzeugposition auf einen Trace und gibt das Ergebnis zurück mit einer Matchqualität, Matchposition und Entfernung bis zur Referenzposition. Für weitere folgende Aufrufe im Sinne eines kontinuierlichen Matchens der Fahrzeugposition sollte das Ergebnis gespeichert und wieder übergeben werden. Wird kein vorangegangenes Match-Ergebnis beim Aufruf mit übergeben, wird nicht nur die aktuelle Position gematched sondern auch die Historie aus eigenen gepufferten Positions punkten. Hierdurch entsteht ein erhöhter Rechenaufwand.

```
TraceMatch matchTrace2Trace(extTrace trace1, extTrace trace2)
```

Matcht zwei Traces aufeinander. Auch die Funktion gibt ein Objekt mit der Matchqualität und den Abstand der Referenzpositionen zurück.

3.2.9 Funktionskoordination

In der VAU werden mehrere Hauptfunktionen, die jeweils mehrere Funktionen beinhalten, parallel betrieben. Diese fordern i.A. Ausgaben durch das HMI oder durch die Kommunikation (Communication Client) an. Dem Grundsatz folgend, dass Zugriffe koordiniert zu erfolgen haben, wenn gemeinsame Ressourcen von mehreren Nutzern angefordert werden, sollten entsprechende Koordinationsmechanismen vorgesehen werden. Weiterhin fordern die Funktionen als Ressource CPU-Leistung auf der VAU an. Kommt die

CPU-Leistung der VAU an ihre Grenzen, muss die Abarbeitung der Funktionen so gesteuert werden, dass die CPU-Last soweit verringert wird, damit hochpriore Funktionen zufriedenstellend weiterlaufen können.

Die Zielsetzungen der Funktionskoordination sind damit:

- Vermeidung konkurrierender Information für den Fahrer.
- Vermeidung nicht situationsgerechter Information für den Fahrer.
- Vermeidung eines Kommunikationsengpasses durch situationsgerechte Priorisierung zu versendender Daten.
- Situationsgerechte Steuerung der CPU-Auslastung.

3.2.9.1 Koordination der Ausgaben

Zunächst wird das Vorgehen bei der Nutzung von HMI und Communication Client beschrieben. Eine einfache Art einer Koordination ist die Vergabe von (statischen oder dynamischen) Prioritäten an die „Nutzer“ der Komponenten HMI und Communication Client, anhand derer die Abarbeitung der Anforderungen gesteuert wird. Dazu ist in diesen Komponenten eine Prioritätentabelle implementiert, die eine a priori Festlegung von Prioritätsbereichen enthält, in denen sich die Funktionen bewegen können. Diese Festlegung wird zusammen mit den Funktionsentwicklern getroffen. Sie stellt ein den Funktionen übergeordnetes „Wissen“ dar, wie diese relativ zueinander zu priorisieren sind. Innerhalb dieser Prioritätsbereiche können die Prioritäten situationsabhängig angepasst werden. Dies ermöglicht eine dynamische, (fahr-)situationsabhängige Priorisierung, um das Gesamtsystemverhalten aus Nutzersicht konsistent zu gestalten.

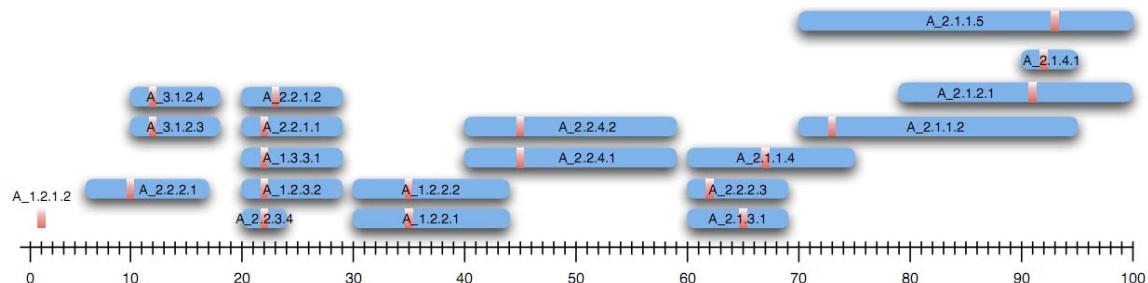


Abbildung 3-43: Schema einer Prioritätentabelle mit Prioritätsbereichen für Funktionen/Anwendungsfälle (Quelle: Deliverable D22.2)

Die Prioritätswerte werden durch die Funktionen selber generiert in einem Wertebereich von 1 - 100%, d.h. die Funktionen bewerten ihre Priorität aus ihrem internen „Wissen“. Um kein gegenseitiges „Wissen“ in die Funktionen untereinander einzubauen, wird ein normierter Wertebereich verwendet, der in die o.g. Prioritätsbereiche abgebildet wird.

Um situationsgerecht Funktionen relativ zueinander zu priorisieren, kann eine Anpassung der Prioritätswerte erforderlich sein. Dazu wird ein Koordinator implementiert, der übergeordnetes Situationswissen haben muss. Dieses bezieht er aus den Funktionen und ggf. aus dem Zugriff auf weitere verfügbare Daten (Fahrzeugdaten, Karte/Navi, Position, ...) generiert. Die Funktionen „fragen“ beim Koordinator nach, ob die vorgeschlagene Priorität in Ordnung ist oder ob vom Koordinator andere Prioritäten vorgegeben werden, die dann von den Funktionen umzusetzen sind.

Die situationsgerechte Anpassung der Prioritätswerte kann auch dazu verwendet werden, einzelne Ausgaben (Präsentationsaufträge und/oder Kommunikationsaufträge) nicht auszuführen, da sie z.B. in dieser Situation mit anderen konkurrieren. Die angepassten Prioritätswerte werden den Funktionen mitgeteilt, damit sich diese ggf. andere (Anzeige-) Strategien „überlegen“ können.

3.2.9.2 Koordination der Funktionsaktivierung

Bei einer hohen CPU-Last kann ebenfalls situationsabhängig eine Anpassung der Funktionspriorisierung erfolgen. Hierbei werden die absehbar (d.h. in der aktuellen Situation) nicht wichtigen Funktionen (z.B. Funktion 1.2.1 und 1.2.2) niedrig priorisiert, und lediglich absehbar wichtige Funktionen hoch priorisiert (z.B. Kreuzungsassistent vor einer Kreuzung und kritischem Querverkehr). Die niedrig priorisierten Funktionen sollten dabei ihre Last auf ein Minimum reduzieren. Hierbei wird die Priorisierung nicht durch OSGi oder Java Mechanismen durchgeführt, sondern es wird eine Meldung an die Funktionen gesandt, so dass diese selbst entsprechende Maßnahmen ergreifen. Vorstellbar sind hier z.B. ein Abschalten des Hauptalgorithmus, so dass nur noch solche Funktionsteile laufen, die Daten zur Situationsbewertung liefern oder eine Reduktion der Taktrate oder Warten. Die angepassten Prioritäten werden vom Koordinator an die Funktionen geliefert. Die Funktionen haben dann noch Zeit, z.B. kritische Situationen abzuschließen und ihre Funktion in einen geordneten (Ruhe-)Zustand überzuführen.

3.2.9.3 Anwendung der Funktionskoordination in den Hauptfunktionen

Grundsätzlich lässt sich die Funktionskoordination auf alle sim^{TD}-Funktionen anwenden. Da jeweils ein zusammenhängendes Set von den Funktionen zu Hauptfunktionen gebündelt sind, müsste die sich Koordinationsschicht in einer den Hauptfunktionen übergeordneten Systemfunktion befinden. Diese ist in sim^{TD} nicht vorgesehen, auch sind hierfür keine Ressourcen und Verantwortlichkeiten eingeplant. Daher wird die Funktionskoordination auf die Hauptfunktionen beschränkt und sollte dort als hauptfunktionsinterne Komponente vorgesehen werden. Diese

- priorisieren ihre HMI-Ausgaben intern vor und geben die normierten Prioritäten an das HMI weiter. Das HMI priorisiert die Aufträge der Funktionen nach einem vorab festgelegtem Schema, wie in Abbildung 3-43 beispielhaft dargestellt, ohne Verwendung von aktuellem, HF-übergreifendem Situationswissen.
- priorisieren analog die Aufträge an die Kommunikation. Diese arbeitet die Aufträge nach einem vorab festgelegten Prioritätenschema ab. Die Prioritätswerte unterscheiden sich i.A. von den für die HMI-Ausgabe gewählten Werten.
- sehen bei CPU-Überlast eigenständig ressourcensteuernde Maßnahmen vor. Der System-Manager verteilt Aufträge zur Reduktion von CPU-Last nach einem vorab festgelegten Schema an die Funktionen. Greifen diese Maßnahmen nicht, schaltet er die Funktionen (wie vorgesehen) ab.

Die Einbindung der Koordinationsschicht in eine Hauptfunktion zeigt Abbildung 3-44. Die Koordinationskomponente hat dabei vier Teilkomponenten:

- Berechnung der Gesamtsituation innerhalb der Hauptfunktion
- Berechnung der Prioritäten für Funktionsaktivierung
- Anpassung der Prioritäten für HMI

- Anpassung der Prioritäten für Communication Client

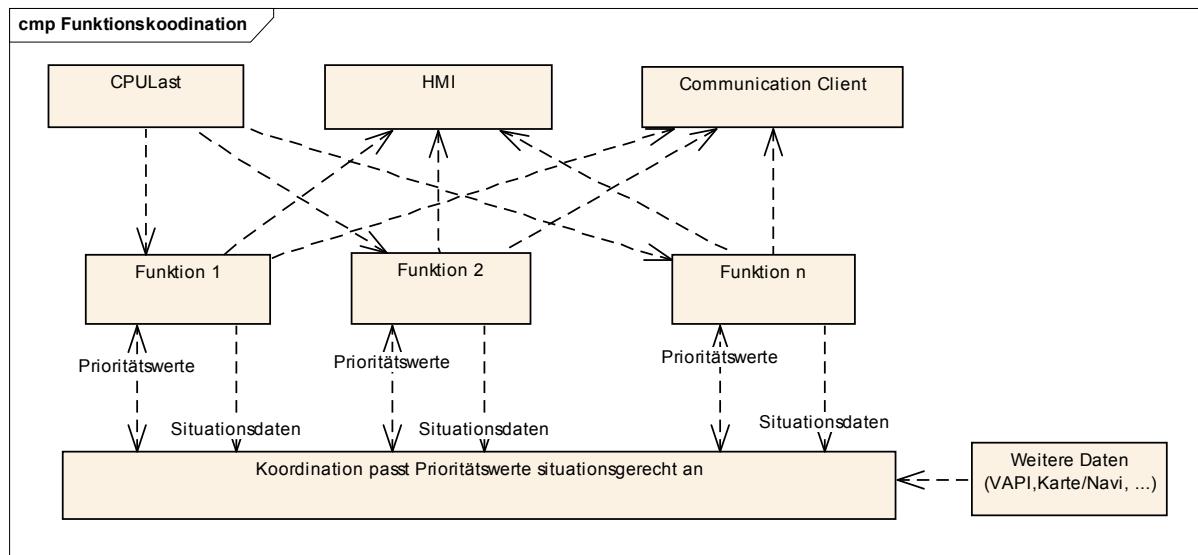


Abbildung 3-44: Struktur einer Funktionskoordination.

3.2.10 Verkehrslage

Die Systemkomponente Verkehrslage auf der VAU hat die Aufgabe, die aus mehreren Quellen über verschiedene Wege im Fahrzeug eingehenden Verkehrslageinformationen zentral zu speichern und verfügbar zu machen. Sie schlägt dabei eine Brücke zwischen dem fahrzeugeigenen Navigationssystem und Modellen, die der Verkehrslage zu Grunde liegen.

Weiterhin bietet sie die Möglichkeit, eine Kanteneinfärbung in der Navigation abhängig von der Herkunft der Verkehrslageinformationen durchzuführen.

Im Detail treffen im Fahrzeug zwei Arten von Verkehrslageinformationen ein:

- VL-Informationen der Funktion F_1.1.4, die in der VsZ generiert wurden. Diese Funktionen werden von der IRS in Form von DEN-Messages über IEEE 802.11p ausgestrahlt, von der CCU im Fahrzeug empfangen und gelangen über die Umfeldtabelle und die Funktion F_1.2.3 in die Verkehrslagedatenbank. Alternativ werden die Informationen aber auch von der Funktion F_3.1.1 über UMTS aus der VsZ abgerufen und in die Verkehrslagedatenbank eingespeist.

SOTIS-VL-Informationen, die im Fahrzeug generiert werden, treffen per IEEE 802.11p WLAN in der CCU im Fahrzeug ein, werden in die Umfeldtabelle geschrieben, dort von der Funktion F_1.2.1 ausgelesen und in die Verkehrslagedatenbank geschrieben.

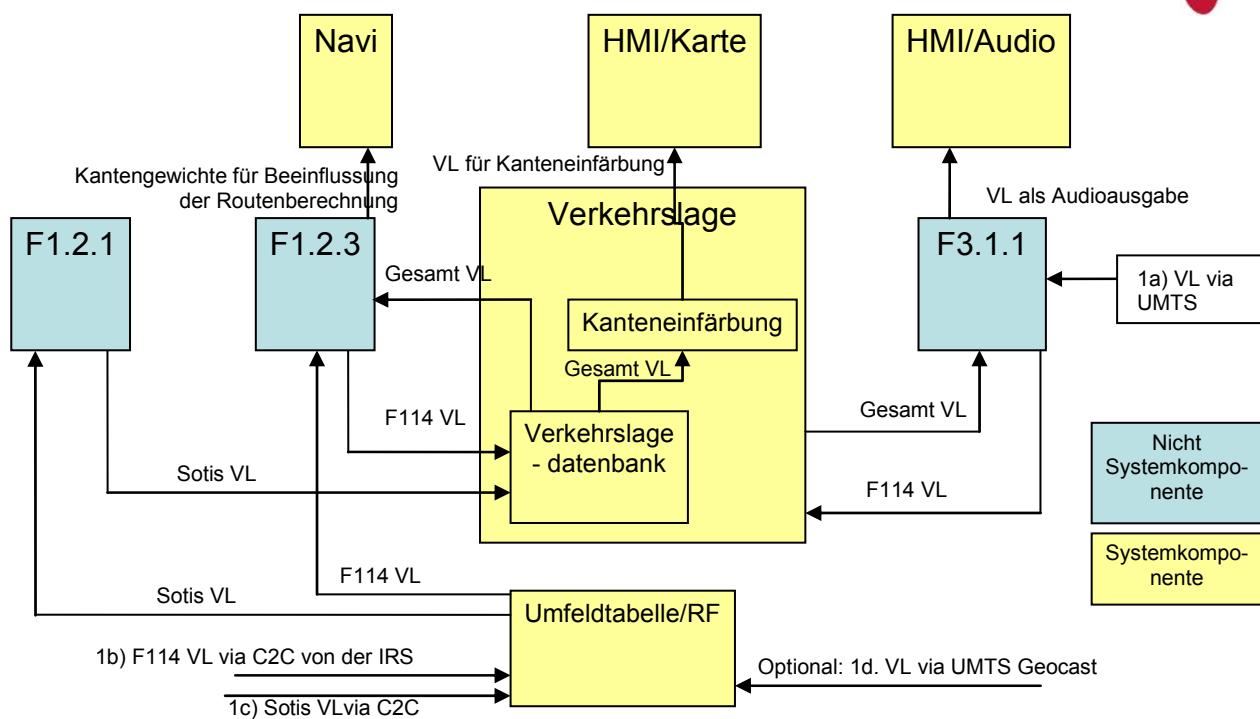


Abbildung 3-45: Einordnung der Komponente „Verkehrslage“

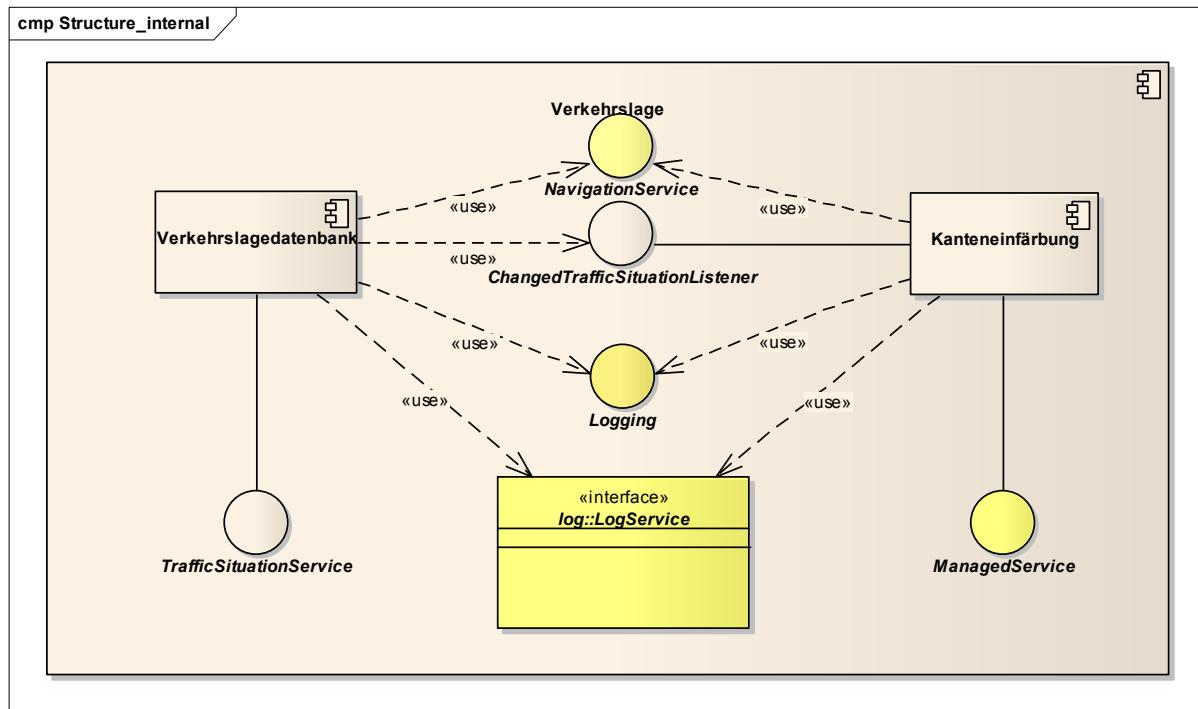


Abbildung 3-46: UML-Modell der Verkehrslagedatenbank

Teilfunktionalität Verkehrslagedatenbank

In der Verkehrslagedatenbank werden somit alle im Fahrzeug verfügbaren Informationen zur Verkehrslage zusammengefasst, wobei zu den einzelnen Elementen die Übertragungsart,

der Generierungszeitpunkt und der Empfangszeitpunkt gespeichert werden, um eine spätere Differenzierung und Auswertung zu ermöglichen.

Die Daten werden für drei unterschiedliche Ausgabearten von den Funktionen genutzt:

- F_1.2.1 aktualisiert die SOTIS-VL und sendet sie über IEEE802.11p an andere Fahrzeuge weiter.
- F_1.2.3 übergibt die Verkehrslage-Informationen an die Navigationskomponente, ändert dadurch für individuelle Links der digitalen Karte die Eigenschaften, die den Verkehrsfluss darstellen (Geschwindigkeit auf der Kante, Fahrzeit auf der Kante oder Level of Service, das muss noch nach Klärung des konkreten Datenformats und der Navigationsschnittstelle festgelegt werden).
- F_3.1.1 gibt die für den Fahrer relevanten Verkehrslage-Informationen nach einer Text-to-Speech-Konvertierung akustisch aus.
- Weiterhin Kommuniziert die Systemkomponente Verkehrslagedatenbank selbst die Verkehrslage-Informationen an die Navigationskomponente, um eine von der Verkehrslage abhängige Einfärbung der Straßen auf der Karte zu bewirken.

Innerhalb der Verkehrslagedatenbank erfolgt die Speicherung der Verkehrslage-Informationen auf Basis der LinkIDs der Navigation, den eindeutigen Bezeichnern der Straßensegmente, aus denen sich die definierten Verkehrswege der Navigation zusammensetzen. Die eingehenden Verkehrslage-Informationen der Funktion F_1.1.4 beziehen sich auf Abschnitte, wobei im allgemeinen Fall davon ausgegangen wird, dass ein Abschnitt sich auf einen oder mehrere Links bezieht. Dieser Bezug muss zunächst ermittelt werden und die eingehenden Daten müssen entsprechend konvertiert werden. Da sowohl das Straßennetz der digitalen Karte – und damit die Links – als auch das Kantenmodell der VsZ-Verkehrslage als relativ statisch im Verlauf der Versuchsdurchführung angesehen werden, kann eine Zuordnung von Kanten auf Links in Tabellenform gespeichert werden.

Einem Link in der Verkehrslagedatenbank können mehrere Verkehrslage-Informationen zugeordnet sein: F_1.1.4 Verkehrslage, via F_1.2.3 empfangen, F_1.1.4 Verkehrslage über F_3.1.1 empfangen sowie der SOTIS Verkehrslage der Funktion F_1.2.1, wobei die Information selbst eine durchschnittliche Geschwindigkeit, ein Level of Service und ein Delay, eine Verzögerungszeit auf dem Segment sein können.

Die Schnittstellen der Verkehrslagedatenbank sind im folgenden Bild dargestellt.

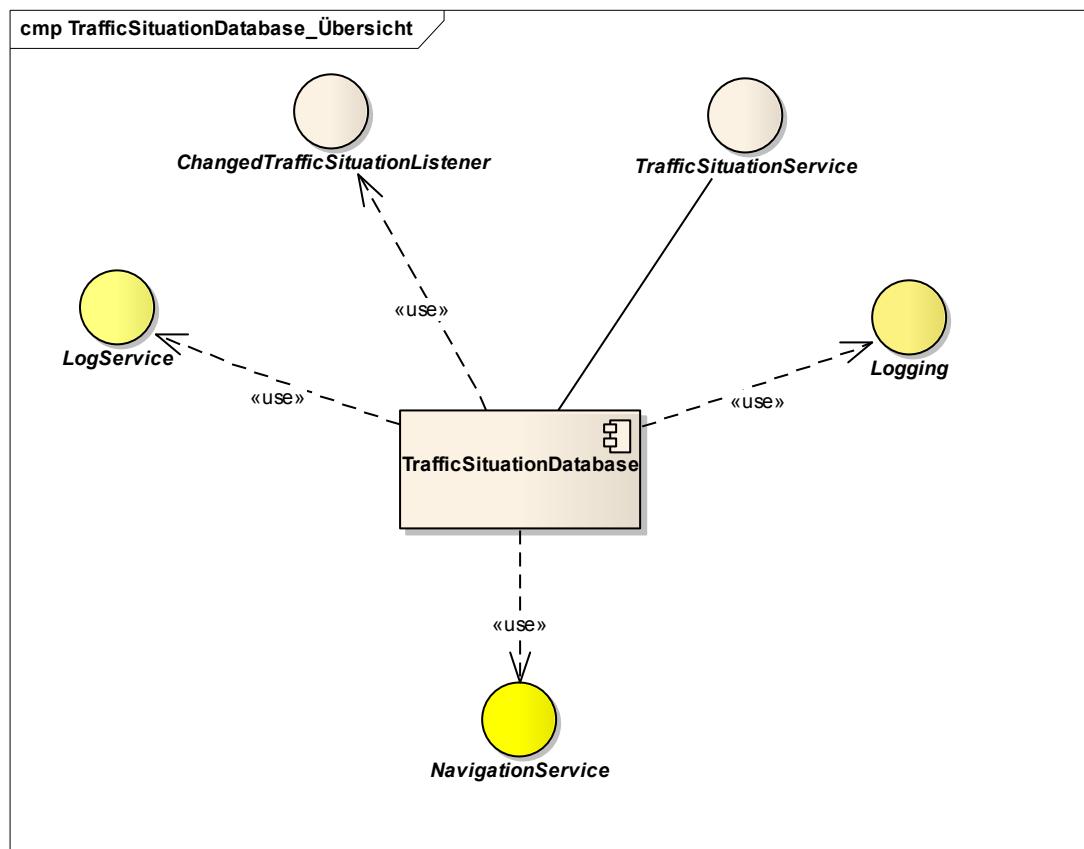


Abbildung 3-47: Schnittstellen der Verkehrslagedatenbank

Die Verkehrslagedatenbank bietet zwei Möglichkeiten der Interaktion: Zum einen registriert Sie den Service `TrafficSituationService`. Dieser Service bietet ein Interface an, über das neue oder geänderte Verkehrslagen in die Verkehrslagedatenbank eingepflegt oder entfernt werden können. Weiterhin bietet er die Möglichkeit, zu einem oder mehreren Links die dazugehörigen Verkehrslagen abzufragen.

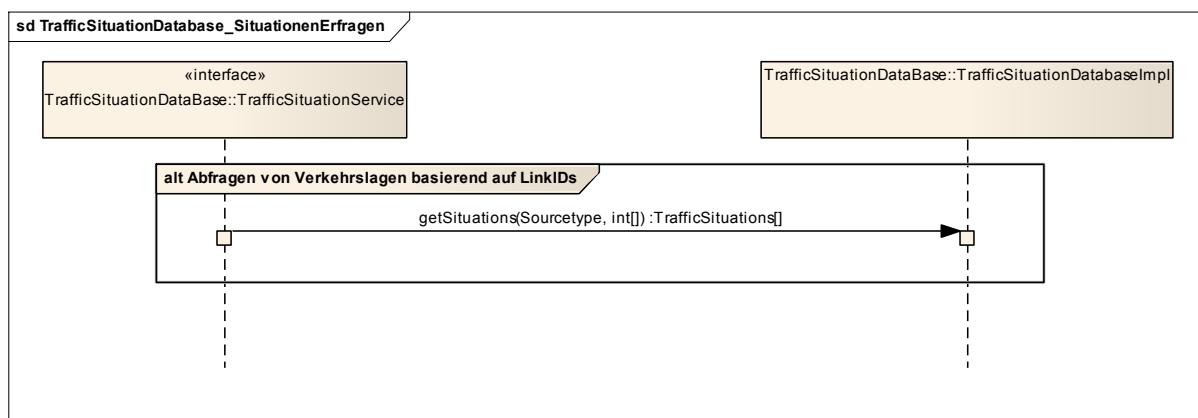


Abbildung 3-48: Ablaufdiagramm „SituationenErfragen“

Der `TrafficSituationService` wird genutzt von `F_1.2.1`, `F_1.2.3` und `F_3.1.1` um Verkehrslagedaten einzupflegen. `F_3.1.1` nutzt die Schnittstelle um gezielte Anfragen bezüglich der LinkIDs der geplanten Route zu stellen.

Zum anderen unterstützt die Verkehrslagedatenbank das Whiteboard-Pattern über das Interface `ChangedTrafficSituationListener`. Funktionen, die an geänderten Verkehrslagen

interessiert sind, können dieses Interface implementieren. Sie registrieren sich damit im OSGi-Framework. Die Verkehrslagedatenbank liefert dann an alle registrierten Listener neue, geänderte oder entfernte Verkehrslagen mit den dazugehörigen LinkIDs sowie der Quelle, aus der diese Daten stammen.

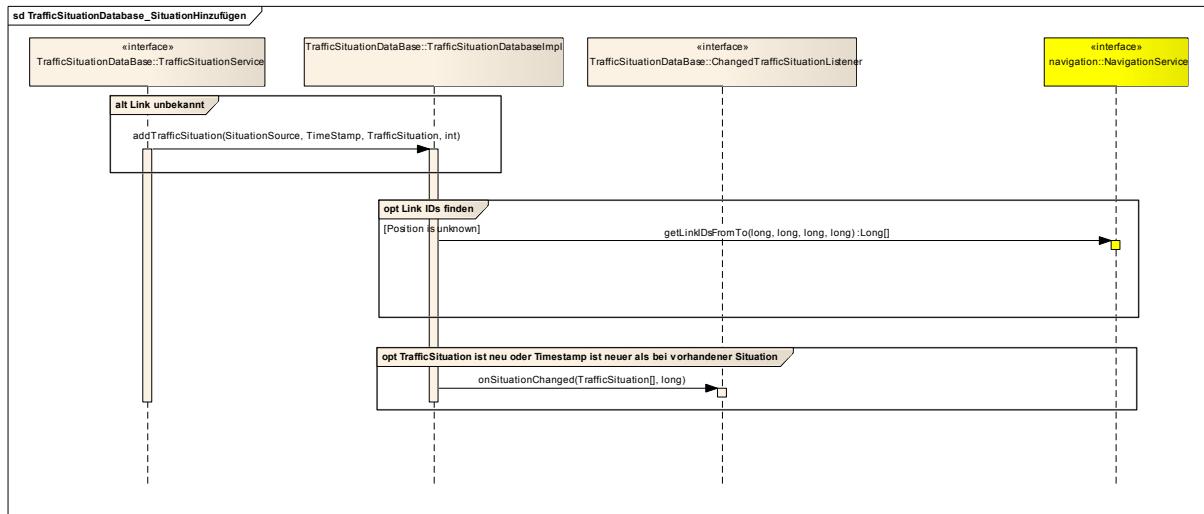


Abbildung 3-49: Ablaufdiagramm „SituationHinzufügen“

Das Interface `ChangedTrafficSituationListener` wird implementiert werden von der Funktion F_1.2.3 und der Teilkomponente Kanteneinfärbung.

Teilfunktionalität Kanteneinfärbung

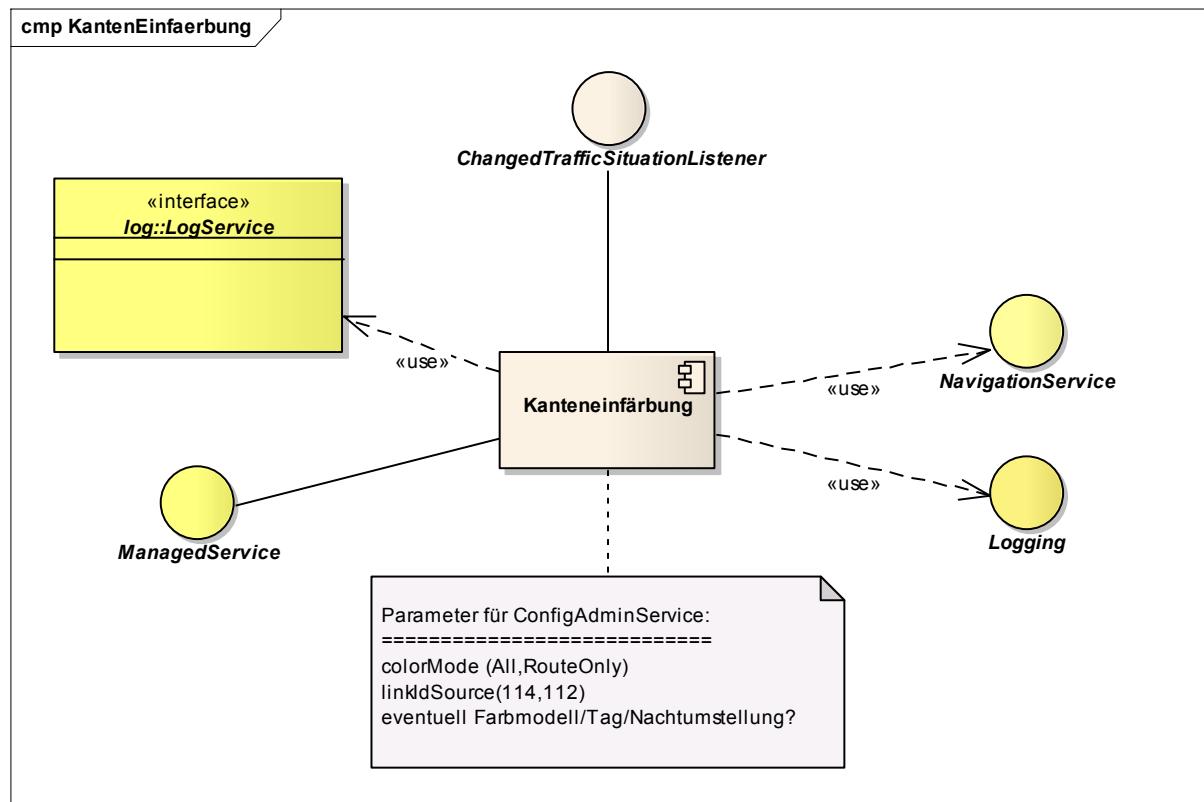


Abbildung 3-50: Einordnung der „KantenEinfärbung“

Die Kanteneinfärbung dient der Einfärbung von Navigationskanten abhängig von diesen zugeordneten Verkehrslagen und der Quelle, aus der diese Verkehrslage stammt.

Sie ist konfigurierbar hinsichtlich

- der Quelle der Verkehrslagedaten (Funktionen F_1.1.4, F_1.2.1),
- Einbeziehung einer geplanten Route, also ob nur Kanten eingefärbt werden, die sich auch auf der Route befinden, oder alle bekannten Verkehrslagen abgebildet werden,
- Abhängig von der Navigationsfunktionalität muss möglicherweise zwischen einem Farbprofil für den Tag und eines für die Nacht unterschieden werden.

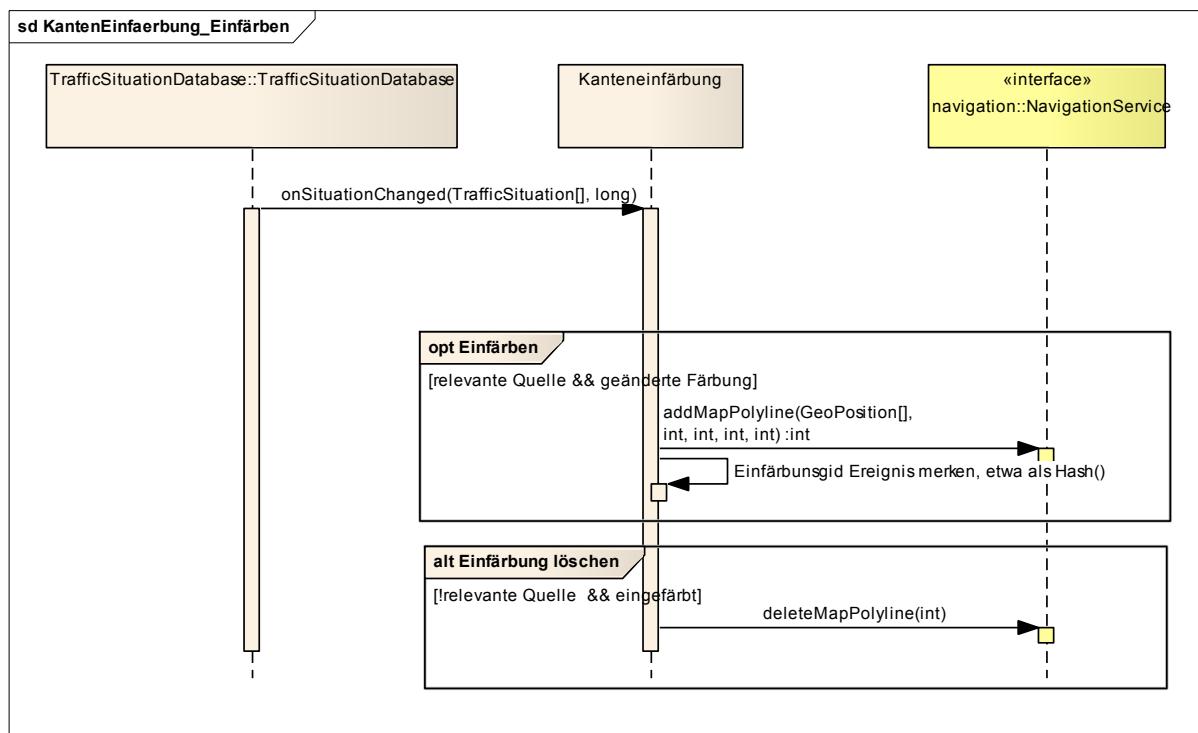


Abbildung 3-51: Ablaufdiagramm „KantenEinfärbung_Einfärben“

Konfigurierbarkeit wird über die Implementierung des Interfaces ManagedService erreicht. Die Kanteneinfärbung stellt keinen eigenen Service zur Verfügung. Sie registriert sich als ChangedTrafficSituationListener, um über für sie eventuell relevante Änderungen durch die Verkehrslagedatenbank informiert zu werden.

3.2.11 Fahrerverhaltensvorhersage

Die Fahrerverhaltensvorhersage stellt ein Bundle zur Verfügung. Dieses umfasst den Service „Abbiegevorhersage“. Die Angaben beziehen sich jeweils auf das Eigenfahrzeug und die nächste Kreuzung. Dieser Service wird in den Hauptfunktionen 1.3 und 2.2 verwendet.

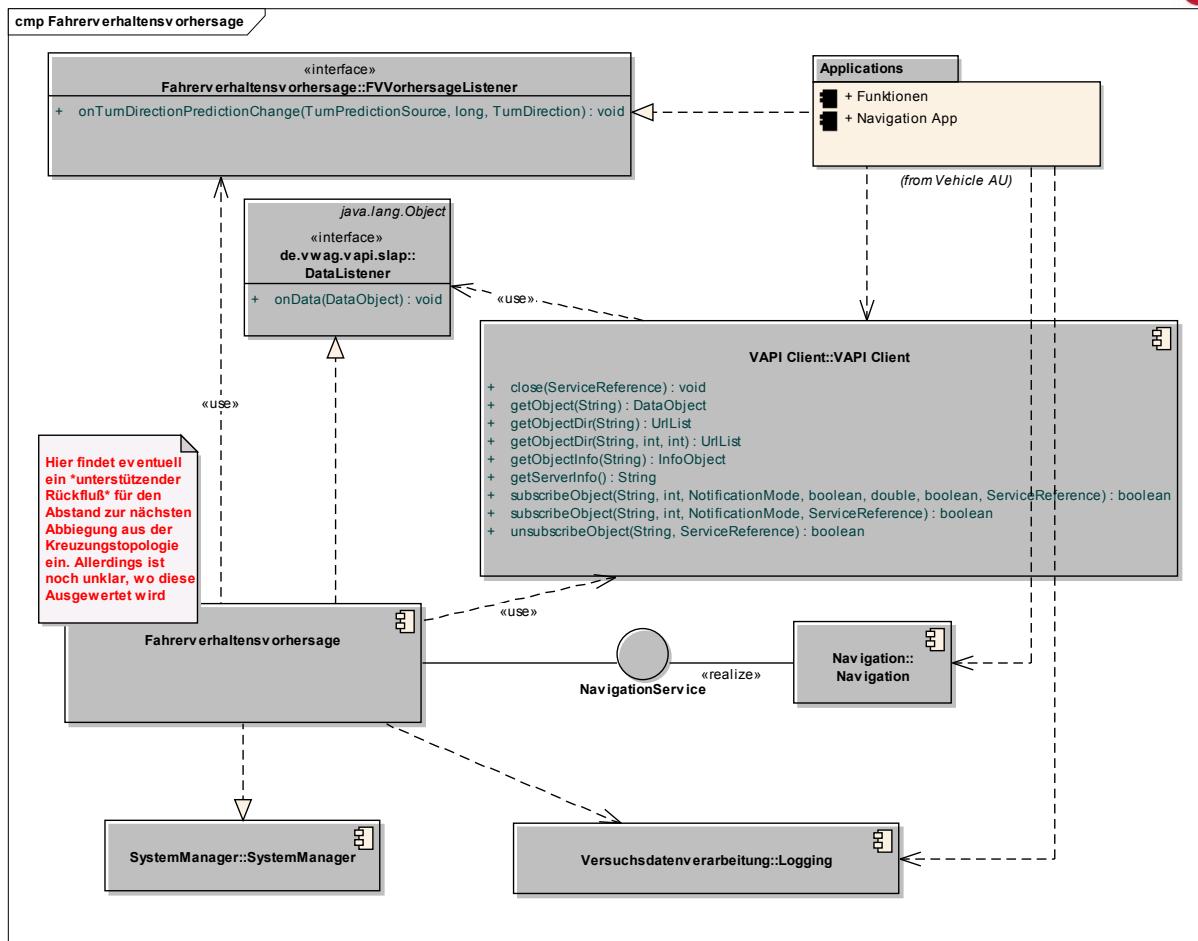


Abbildung 3-52: Einordnung und Schnittstellen der Komponente „Fahrverhaltensvorhersage“

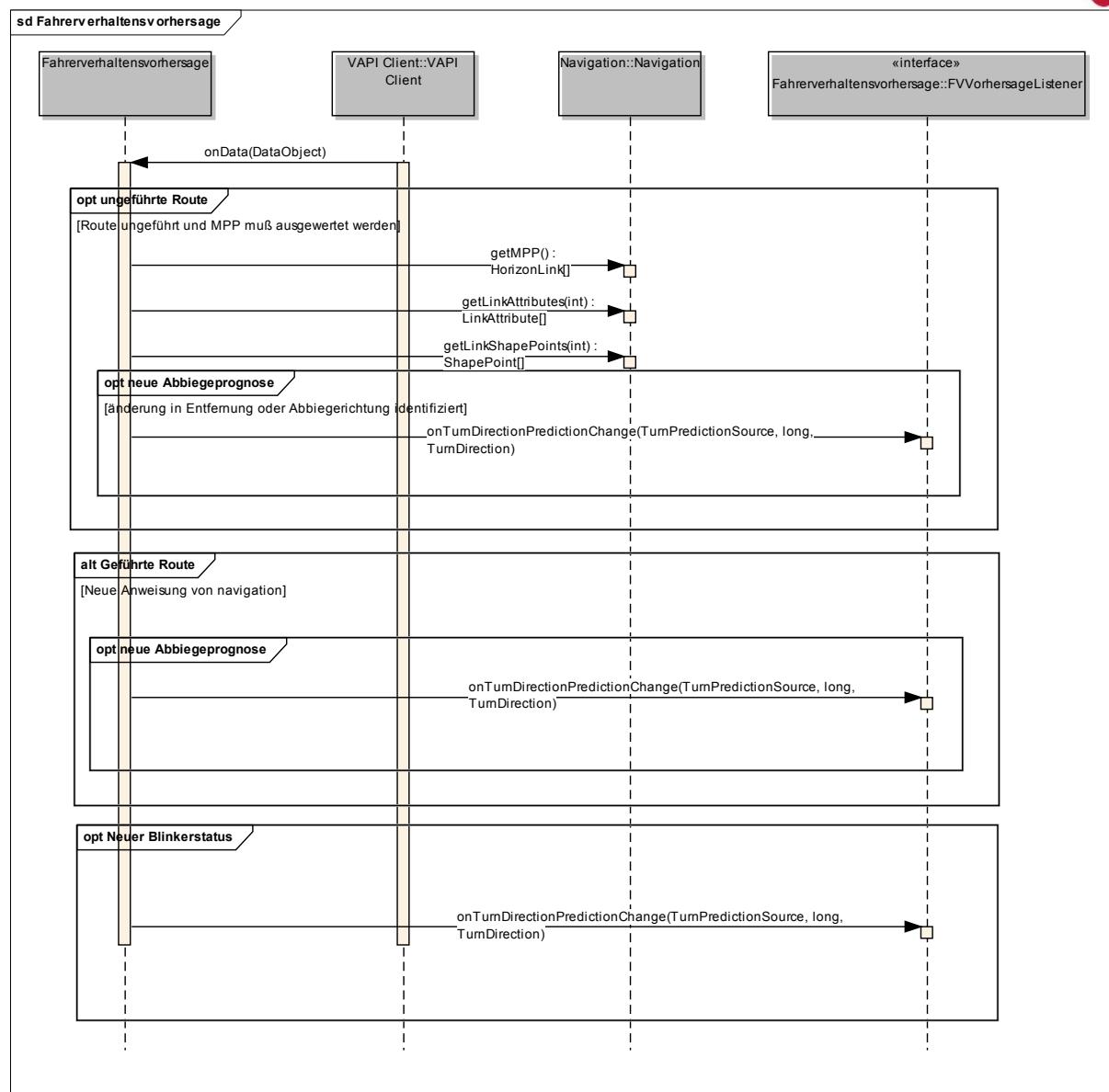


Abbildung 3-53: Ablaufdiagramm der Komponente „Fahrerverhaltensvorhersage“

3.2.11.1 Importierte Schnittstellen

Die Funktion ruft über die VAPI den Blinker Status ab. Optional werden Detailinformationen über die Kreuzungstopologie genutzt. Der Zugriff auf die Kreuzungstopologie ist noch nicht final geklärt. Es wird wahrscheinlich eine Unterstützung durch die Kreuzungstopologie geben, die die Genauigkeit der Komponente im Hinblick auf den Abstand zur Halteline/Abbiegestelle verbessern kann an mit entsprechenden IRS ausgestatteten Kreuzungen.

Von der Navigation werden u. a. Angaben bezüglich dem „Most Probable Path“ (MPP) oder der Navigationsvorgabe an der nächsten Kreuzung (z. B. links abbiegen) benötigt.

Außerdem nutzt die Komponente die üblichen Schnittstellen zum Logging, SystemManager und zur Parametrierung.

3.2.11.2 Exportierte Schnittstellen

Die Komponente exportiert eine Schnittstelle mit folgenden Angaben:

- Quelle, auf die sich die Berechnung stützt (MPP, Routenplanung oder Blinker)
- Entfernung des Eigenfahrzeugs zur Haltelinie/Sichtlinie an der nächsten Kreuzung, sofern dafür eine Abbiegeprognose vorliegt.
- Vorhersage, wohin das Eigenfahrzeug an der nächsten Kreuzung abbiegen wird.

Diese Angaben besitzen dann eine Gültigkeit, wenn sich das Eigenfahrzeug an eine Kreuzung annähert und eine definierte Entfernung zur Kreuzung unterschritten ist.

3.2.12 Hauptfunktionen

Die Feinspezifikation der Hauptfunktionen findet sich in Deliverable D11.3.

3.2.13 OEM-Sonderfunktionen

3.2.13.1 Sonderfunktionen und OEM-Devices

In sim^{TD} werden zusätzliche OEM-Versuchsträger aufgebaut, welche über die Basisfunktionen hinausgehende „Sonderfunktionen“ implementieren. Eine Sonderfunktion kann beispielsweise sein:

1. Eine alternative Ausprägung einer Basisfunktion (z.B. ein Kreuzungs-Querverkehrsassistent mit alternativer HMI-Lösung in ggf. alternativer Parametrierung).
2. Ein Anwendungsfall, welcher nicht in allen Fahrzeugen der Basisflotte implementiert ist.
3. Eine über die Basisfunktionen hinausgehende „neue“ kooperative Funktion.

Auch Funktionen aus anderen Projekten (z.B. AKTIV) sollen – auf eigene Rechnung – in das sim^{TD}-Testfeld integriert und evaluiert werden können. Ferner sollen in sim^{TD} Funktionen unter Verwendung einzelner OEM-Devices (z.B. PDAs) implementiert und integriert werden können.

sim^{TD} stellt für die Integration von OEM-Sonderfunktionen und OEM-Devices in das sim^{TD}-Funktionsnetz eine universell verwendbare Datenschnittstelle als generisches Interface bereit. Diese Schnittstelle wird im folgenden Überblick beschrieben. Eine detailliertere Beschreibung findet sich im Anhang „Beschreibung der Schnittstelle zu OEM-Sonderfunktionen“ zu diesem Dokument.

3.2.13.2 Anforderungen

Da Sonderfunktionen sehr unterschiedliche Hard- und Softwareplattformen nutzen können, wird das OEM-Interface auf einfache Weise einen plattformunabhängigen Datenaustausch ermöglichen.

OEM-Funktionen sollen Zugang zu allen relevanten Outputs aller Module, Systemkomponenten und Hauptfunktionen haben, damit sie – basierend auf äquivalentem

Input – gleichwertige oder verbesserte Funktionsumfänge implementieren können. Die Schnittstelle soll also möglichst nicht nur einen kleinen Subset von Signalen für OEM-Devices zugänglich machen, sondern soll performant die Signale bereitstellen, welche auch von Komponenten im OSGi-Framework abonniert werden können. Dies beinhaltet insbesondere einen effizienten und vollständigen Zugriff auf in der „Umfeldtabelle“ repräsentierte Fahrumgebung.

OEM-Funktionen liefern nach aktuellem Planungsstand keine Output-Signale, welche in das sim^{TD}-System zurückgespeist werden. Insbesondere verwenden OEM-Funktionen ihr eigenes OEM-spezifisches HMI-Interface. Damit findet im Regelfall keine Integration des sim^{TD}-HMI mit dem OEM-HMI statt. Wird eine solche Integration von einzelnen OEMs für einzelne sim^{TD}-Funktionen gewünscht, muss sie innerhalb der jeweiligen Funktion bzw. Hauptfunktion in eigener Verantwortung implementiert werden (Beispiel: Unterdrückung von Kreuzungswarnungen am sim^{TD}-HMI bei Vorhandensein einer entsprechenden OEM-Displays wie z.B. Headup-Display). Bei Bedarf können mit der Sonderfunktion interferierende sim^{TD}-Hauptfunktionen oder –Teilfunktionen deaktiviert werden.

3.2.13.3 Konzept

Die Daten des Funktionshost werden über ein zentrales OSGi-Bundle und ein Ethernet-Interface nach außen verfügbar gemacht.

Als Schnittstelle für OEM-Komponenten werden auf Grund der universellen Nutzbarkeit die Ethernet-Schnittstellen von Router und Application Unit verwendet werden. Alle relevanten Outputs von sim^{TD}-Komponenten auf der ITS Vehicle Station werden über diese Schnittstellen externer Hardware verfügbar gemacht. Weitere Schnittstellen (z.B. USB, CAM) auf der AU erscheinen aus Aufwandsgründen nicht erforderlich, da eine etwaige benötigte Umsetzung auch von dem externen System selbst durchgeführt werden kann.

Die Daten jeder relevanten sim^{TD}-Output-Komponente werden sowohl auf der Vehicle CCU und auf der Vehicle AU per Ethernet per Multicast auf den Komponenten zugeordnete Multicast-Adressen über UDP-Sockets publiziert. Die Daten werden damit einmal verbreitet und können nach Aktivierung des OEM-Interface von beliebig vielen Empfängern und ggf. Konvertern (z.B. zu CAN) mitgelesen werden. Ein kleingranulares Abonnieren/Subskribieren einzelner Daten entfällt. Sender haben keine Information darüber, wer ihre publizierten Daten mitliest.

Output-Datensätze werden möglichst schnell nach Verfügbarkeit in einem für die jeweiligen Komponenten spezifizierten Output-Takt unsynchronisiert gesendet (ähnlich z.B. einem CAN-Bus).

3.2.13.4 Integrationsverantwortung und Integrationsregeln

Für die Integration von OEM-Sonderfunktionen gelten die in Abschnitt 3.2.1 beschriebenen Integrationsregeln. Darüber hinaus gilt:

1. Die in Abschnitt 3.2.13.3 beschriebene Komponente wird ausschließlich auf den Versuchsträgern entsprechender Projektpartner aktiviert.
2. Eine Abschaltung ist möglich und es erfolgt keine Verwendung zur internen Kommunikation auf der IVS.
3. Es ist keine Erhöhung des Gesamt- bzw. Integrationsbudgets erforderlich.

Die externen Geräte werden nicht von TP2 bzw. TP3 getestet, sondern von den die Geräte benutzenden Partnern.

3.3 Roadside Application Unit

Die IRS wurde im Abschnitt 1.2.3 bereits beschrieben. Im Folgenden wird die Gesamtarchitektur der IRS aufgegriffen und detaillierter beschreiben, da somit auch deutlich wird, welche Verbindungen über welche Teilkomponente laufen.

3.3.1 Physikalischer Aufbau einer IRS

Die IRS besteht physikalisch aus mehreren Modulen bzw. funktionalen Einheiten. Die IRS lässt sich in die beiden Teilsysteme CCU und RAU aufteilen. Die CCU ist im Kapitel 3.1 genauer beschrieben. Sie ist, wie auch schon erwähnt, auf den beiden Plattformen IVS und IRS weitestgehend identisch.

Der Anschluss einer IRS an die Versuchszentrale ist abhängig von ihrem Standort. Das Land Hessen betreibt IRSs neben der Autobahn oder verschiedenen Bundesstraßen im Versuchsgebiet. Im folgenden Bild ist eine IRS des Landes Hessen mit allen Schnittstellen und Teilsystemen abgebildet.

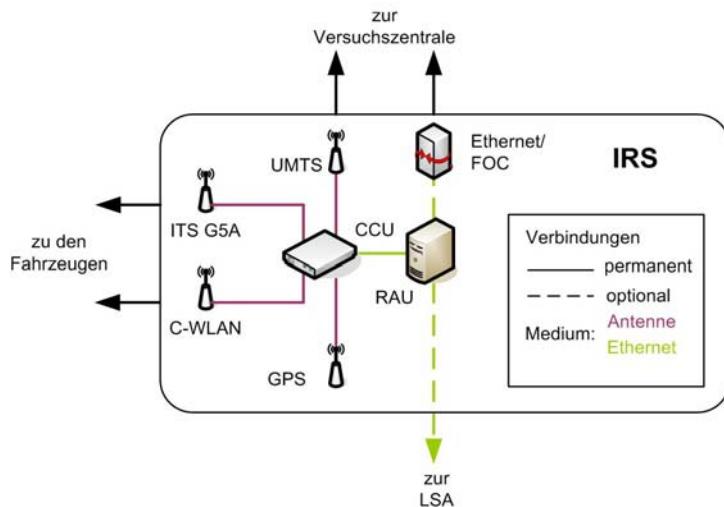


Abbildung 3-54: IRS Land Hessen mit Anbindung VsZ

Die für sim^{TD} ausgewählten IRS-Standorte neben der Autobahn sind alle mit Glasfaser erschlossen und mit der VZH verbunden. Von dort wird es eine direkte Verbindung in die Versuchszentrale geben. Neben den IRSs an der Autobahn gibt es auch IRSs, die an Bundesstraßen im Versuchsgebiet installiert werden sollen. Diese Standorte verfügen jedoch über keine kabelgebundene Kommunikationsinfrastruktur. Daher wird die Anbindung an die VsZ über Mobilfunk realisiert. Bei dieser Art von Anbindung wird die vorhandene UMTS Hardware in der CCU genutzt, um ein Kommunikationskanal in die VsZ aufzubauen.

Neben den IRSs vom Land Hessen werden in sim^{TD} noch IRSs der Stadt Frankfurt betrieben. Diese im städtischen Gebiet installierten IRS unterstützen verschiedene Kreuzungsszenarien. Der Einbau dieser IRS erfolgt in schon existierende Schaltschänke der Stadt Frankfurt. Diese befinden sich im direkten Umfeld einer Kreuzung oder LSA. Um die Anbindung dieser IRS an einem späteren Rollout zu orientieren, werden vorhandene

Kupferleitungen benutzt. Ein SDSL-Modem stellt dabei eine Ethernetverbindung zur Leitungsendstelle in der IGLZ bereit. Auf dieser Teilstrecke kommt wegen dem zu erwartenden synchronen Datenaufkommen SDSL (Symmetric Digital Subscriber Line) zum Einsatz. Da jede IRS in sim^{TD} eine Verbindung zur VsZ benötigt und IGLZ und VsZ zusammen eine Gesamtverkehrslage berechnen müssen, ist eine Verbindung zwischen IGLZ und VsZ zwingend erforderlich. Im Rahmen von sim^{TD} wird eine Glasfaserverbindung zwischen IGLZ und der VsZ geschaffen. Somit ist gewährleistet, dass Funktionsanteile auf der IRS auch mit der VsZ kommunizieren können und IGLZ-Funktionen auf der IRS direkt mit der IGLZ kommunizieren können. Der später beschriebene CommunicationManager (CoMa) stellt für diese beiden Kommunikationswege entsprechende Services zur Verfügung. Die Versuchszentrale abstrahiert in sim^{TD} die Produktivsysteme der VZH; zusätzlich ist das IRSMC (IRS Management Center) dort untergebracht.

Das folgende Bild zeigt den Aufbau und die Anbindung einer städtischen IRS an IGLZ und VsZ.

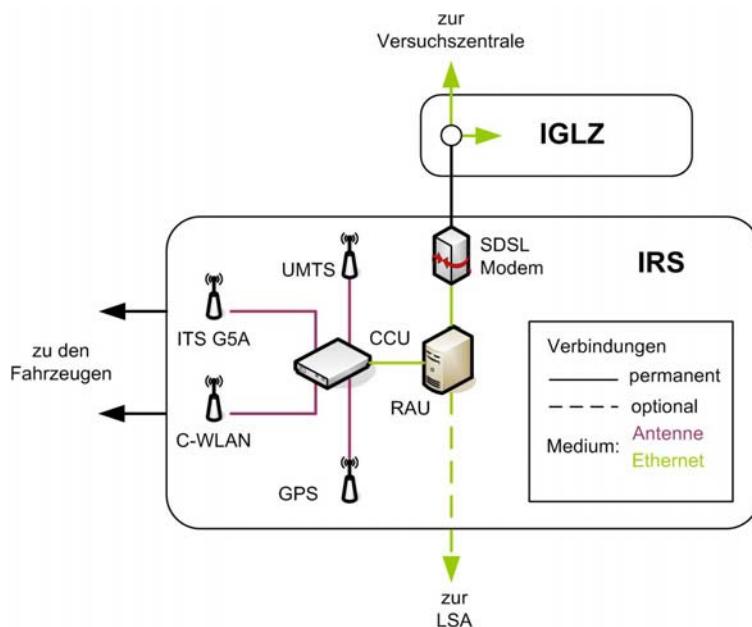


Abbildung 3-55: IRS Stadt Frankfurt mit Anbindung an IGLZ und VsZ

Die Verbindung über UMTS in die Versuchszentrale wird nur benötigt, wenn kein leitungsgebundener Anschluss vorhanden ist. Es ist aber noch zu klären, ob es sinnvoll ist, dass alle IRSs die UMTS-Verbindung als Backupleitung zu nutzen, die dann für eine ausgefallene leitungsgebundene Verbindung dienen kann. Damit könnte die Ausfallsicherheit einer IRS erhöht werden.

Jede CCU verfügt weiterhin über ein Consumer-WLAN-Modul (802.11b/g). Dieses Modul ermöglicht das Senden und Empfangen von C2X-Nachrichten über Consumer-WLAN. Eine permanente oder transparente Verbindung zwischen Fahrzeug und IRS bzw. Fahrzeug und Zentrale (VsZ, IGLZ) ist hierüber nicht möglich. Die einzige Betriebsart des Consumer-WLAN-Moduls ist der Ad-hoc-Modus.

Die IRS ist über die Schnittstelle OTS2 mit dem LSA-Steuergerät verbunden. Hierüber werden aktuelle LSA-Daten ausgetauscht. Da im Versuchsgebiet nicht alle LSA-Steuergeräte über eine Ethernet Schnittstelle verfügen, muss bei dieser Schnittstelle zwischen Testgelände und Versuchsgebiet unterschieden werden. Diese Schnittstelle basiert im Testgelände auf Ethernet und ist im Versuchsgelände über einen potentialfreien Kontakt

angeschlossen. Unter potentialfreien Kontakten sind digitale E/A Anschlüsse zu verstehen, über die alle Informationen über aktuell anstehende LSA Signalzeiten übertragen werden.

Die RAU (Roadside Application Unit) ist ebenfalls an das IRS-interne Netzwerk angeschlossen. Dies ist die zentrale Intelligenz der IRS und beinhaltet sowohl die IRS-seitigen Funktionsanteile als auch die IRS-Management-Module.

3.3.2 Logischer Aufbau IRS

Im folgenden Bild wird eine Übersicht über den logischen Aufbau der IRS gegeben. Die dort enthaltenden Komponenten und Schnittstellen werden in den folgenden Kapiteln näher beschrieben.

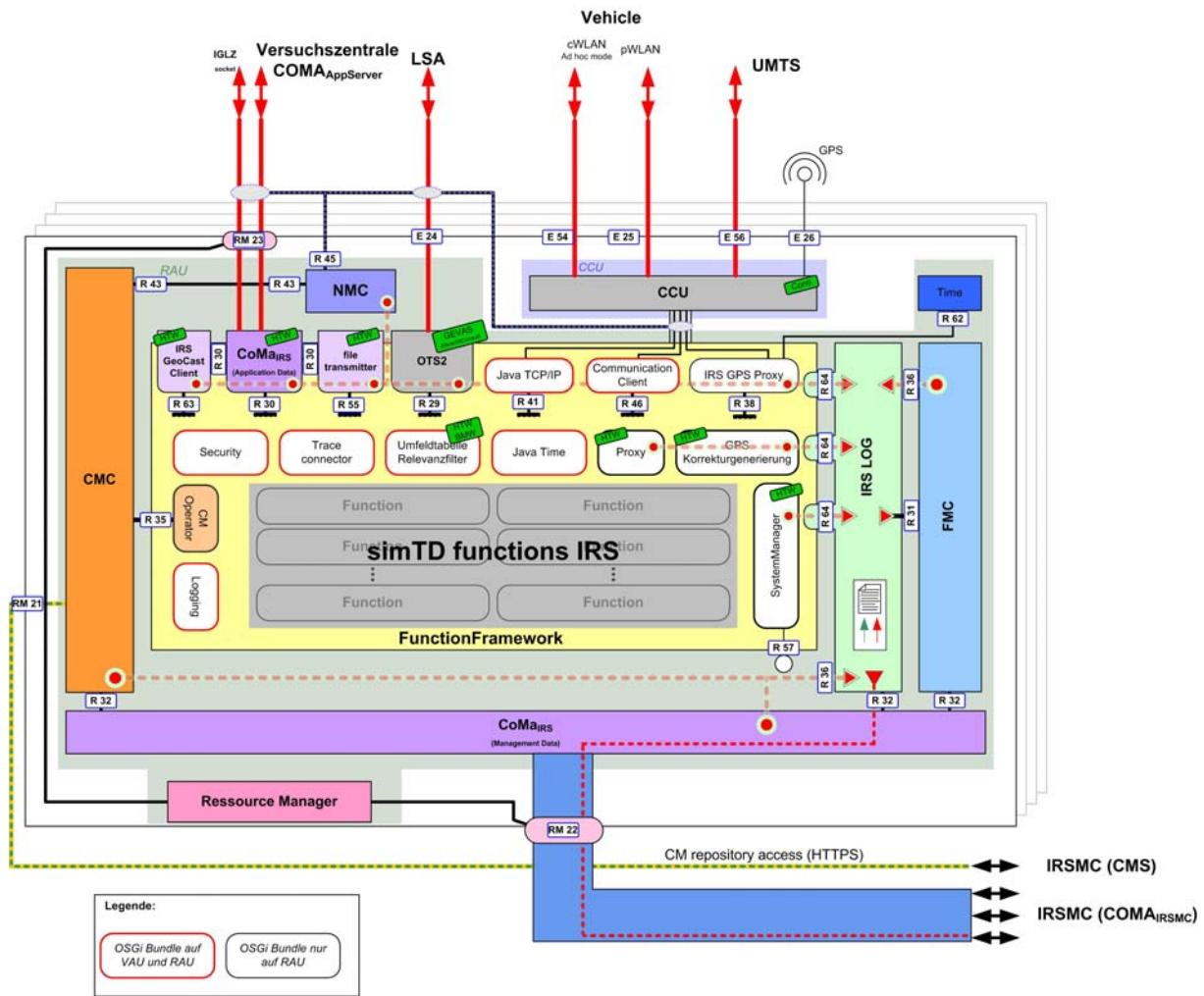


Abbildung 3-56: Logischer Aufbau einer IRS

Alle nummerierten Schnittstellen sind im Abschnitt 3.3.7 genauer definiert. Dem FunctionFramework liegt ein OSGi-Framework zugrunde. Alle rot umrandeten Bundles werden sowohl auf der VAU als auch auf der RAU zum Einsatz kommen. Diese werden, sofern nicht anders angegeben, in Kapitel über die IVS beschrieben. Die Beschreibung der übrigen Bundles und Module wird in den folgenden Abschnitten gegeben.

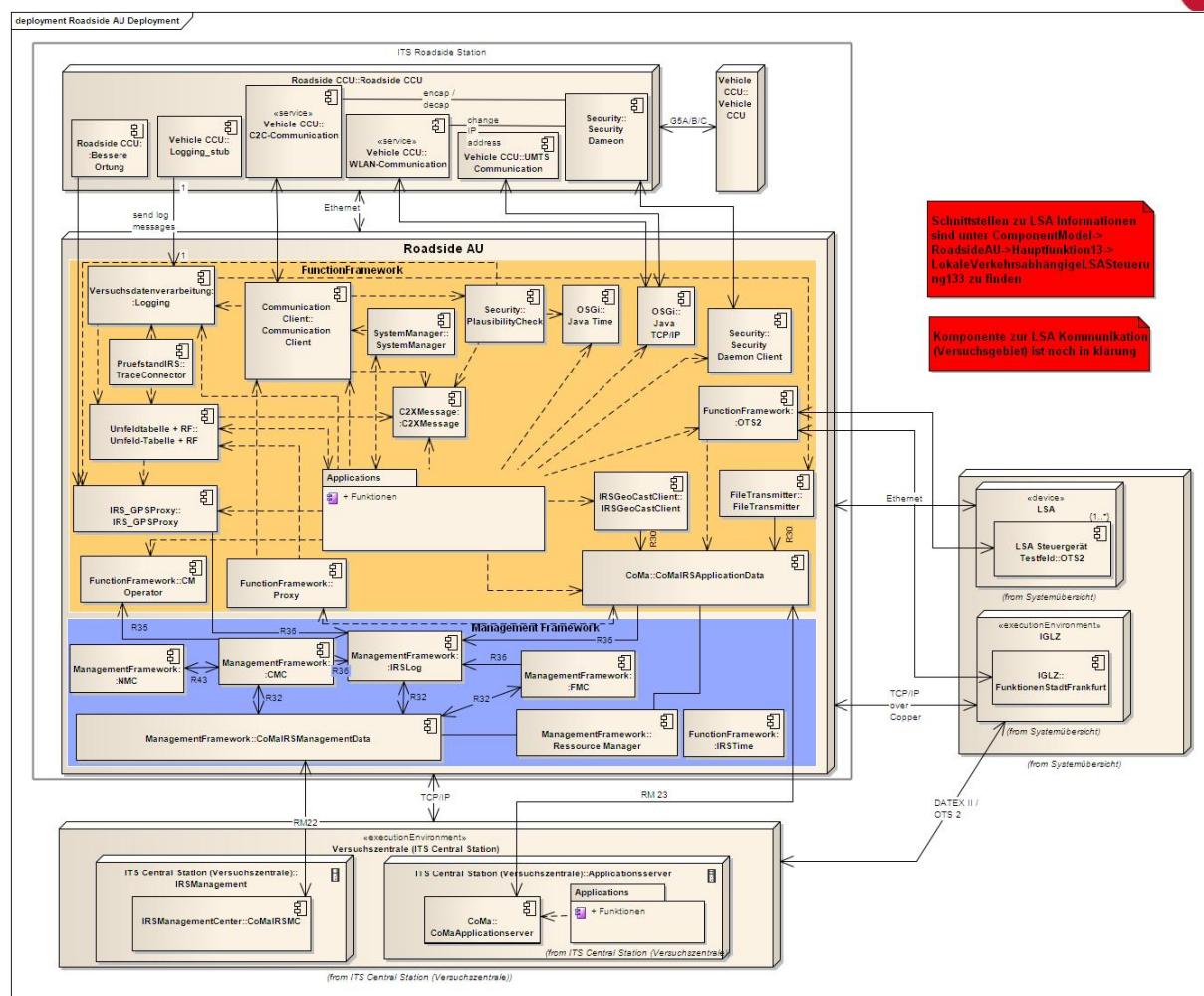


Abbildung 3-57: Roadside AU Deployment-Diagramm

In Abbildung 3-57 ist die IRS mit ihren angrenzenden Systemen als Deployment-Diagramm abgebildet. Die Roadside AU beinhaltet alle für den Betrieb der Funktionen notwendigen Systemkomponenten. Diese Systemkomponenten bilden zusammen mit den Funktionen das FunctionFramework. Dieses Framework bietet im Vergleich zur VAU zusätzlichen Schnittstellen (LSA, IGLZ, VsZ). Alle Systemkomponenten, die das FunctionFramework beinhaltet, werden im Kapitel 3.3.3 beschrieben.

Alle zum Betrieb der IRS Plattform und zur Sicherung des FunctionFrameworks notwendigen Komponenten werden in einem anderen OSGi-Framework installiert und ausgeführt. Diese Kapselung von Funktionsanteilen ist notwendig, um sicher zu stellen, dass etwaige auftretende Fehlverhalten der Funktionen sich nicht negativ auf den Gesamtbetrieb der IRS auswirken. Kommt es zu einem Fehler im FunctionFramework, kann das Fault Management auf der RAU des FunctionFramework neu starten oder auf einen bekannten Fehler mit einer bereits definierten Strategie reagieren. Alle Komponenten dieses Management Frameworks werden im Kapitel 3.3.3.6 beschrieben.

3.3.3 FunctionFramework

In diesem Abschnitt werden alle Module der IRS beschrieben, die als OSGi-Bundles realisiert werden und im direkten Kontakt zu den IRS-seitigen Funktionsanteilen stehen. Sie bilden zusammen ein OSGi-Framework, das so genannte FunctionFramework.

Ein weiterer Punkt, der beim Design einer Anwendungsplattform berücksichtigt werden muss, ist die Kontrollmöglichkeit aller externen Schnittstellen. Es muss gewährleistet werden, dass alle Funktionen Zugriff auf diese externen Schnittstellen haben, abhängig von ihrer Priorität.

3.3.3.1 Bundle Umfeldtabelle

Die Umfeldtabelle wird auf den beiden Plattformen VAU und RAU verfügbar sein. Die Umfeldtabelle wurde bereits im Kapitel 3.2.4 beschrieben. Im Gegensatz zur VAU bezieht die Umfeldtabelle auf der RAU ihre Positionsinformation über den IRS_GPSProxy. Die VAPI wird auf der IRS CCU nicht installiert, da keine Fahrzeugdaten über CAN abgefragt werden können. Die Positionsinformation wird im Relevanzfilter benötigt, der Teil der Umfeldtabelle ist.

3.3.3.2 Bundle Proxy

Zentraleitige Funktionen haben die Möglichkeit, auch ohne Funktionsanteil auf der IRS C2X-Messages zu empfangen. Dazu nutzen sie die IRSProxy Komponente auf dem Application Server in der VsZ. Dieser IRSProxy nimmt Anfragen entgegen und registriert diese über das Proxy Bundle an einer Umfeldtabelle auf einer bestimmten IRS. Der genaue Ablauf wird im Kapitel IRSProxy erklärt.

3.3.3.3 Zugriff auf Zeit: Java Time

Die Funktionen, die die aktuelle Uhrzeit benötigen, können diese durch den einfachen Zugriff auf die Systemzeit erlangen. Die Systemzeit wird auf der IRS von der Komponente IRSTime gesetzt. IRSTime nutzt dafür als Zeitgeber den Zeitstempel des GPS-Empfängers. Diese Funktionalität ist gleich der Zugriffsweise auf der VAU. Da es sich um Java-API Funktionsaufrufe handelt werden hierfür keine UML-Sequenzdiagramme bereitgestellt. Als Basiszeitzone wird UTC angenommen.

3.3.3.4 Bundle IRS_GPSProxy

Der IRS_GPSProxy stellt den Funktionen die aktuellen GPS-Daten zur Verfügung. Den Funktionen wird ein Service zur Verfügung gestellt, den sie benutzen können, um die aktuellen GPS-Daten abzufragen. Diese Komponente stellt nur die Positionsinformation zur Verfügung. Wird die genaue Zeit benötigt: siehe Kapitel Zugriff auf Zeit: Java Time.

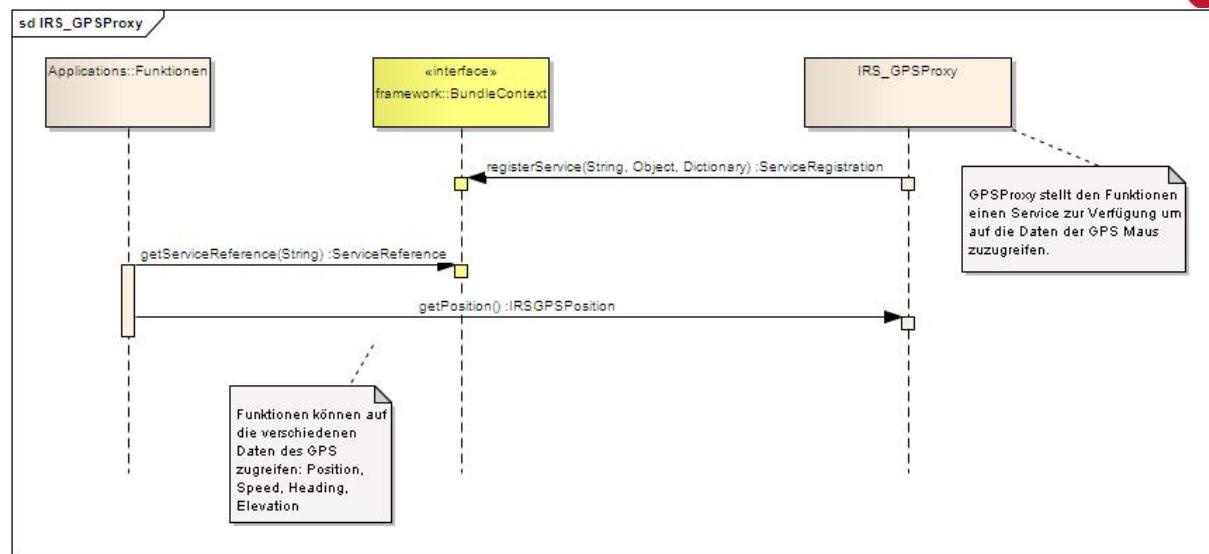


Abbildung 3-58: Sequenzdiagramm IRS_GPSProxy auf IRS

3.3.3.5 Bundle CoMa

Der Communication Manager (CoMa) soll eine fehlerfreie, sichere und performante Verbindung zwischen ITS Roadside Station (IRS) und Versuchszentrale (VsZ) oder zwischen IRS und Integrierten Gesamtverkehrsleitzentrale (IGLZ) zur Verfügung stellen. Diese Verbindung wird zum einen von den Funktionsanteilen genutzt um, zwischen ihren Funktionsanteilen auf der IRS und in der VsZ zu kommunizieren. Zum anderen werden über diese Verbindung Managementinformationen zwischen der IRS und dem IRS Management Center ausgetauscht.

Durch den modularen Aufbau des CoMa und den zur Verfügung stehenden Schnittstellen kann dieser sowohl auf der IRS, den IRS Management Servern (IRSMS) als auch in der VsZ eingesetzt werden. Der CoMa priorisiert sowohl die Managementdaten als auch die Funktionsdaten. Die Funktionen bzw. Management-Bundles haben die Möglichkeit, Daten in Form von Byte-Arrays oder serialisierbare Java-Objekte zwischen den beiden Plattformen auszutauschen. Adressiert werden die IRSs über Position (longitude, latitude, (radius)), Location (Stadt, Land, Autobahn) oder Type (mobil, stationär). Es kommen zwei verschiedene OSGi-Mechanismen zum Einsatz, je nach dem, ob gesendet oder empfangen werden soll. Will eine Komponente (Funktion oder Management) auf der IRS Daten in die Versuchszentrale senden, so muss sie einen Service vom Framework anfordern, den sie dann zum Übertragen des Byte-Arrays oder des Java-Objektes benutzen muss. In Empfangsrichtung muss eine Komponente ein vordefiniertes Interface (DataReceiverIRS, DataReceiverCentral) implementieren und als Service im OSGi-Framework registrieren. Der CoMa bedient dann alle registrierten Services (DataReceiverIRS, DataReceiverCentral) mit den Daten, die von dem zugehörigen Funktionsanteil auf der IRS bzw. Versuchszentrale stammen. In der Versuchszentrale wird den Funktionen zusätzlich noch eine Schnittstelle (IRSInfolProvider) angeboten, über die verschiedene Informationen abgefragt werden können, die zur Adressierung einer IRS benötigt werden.

Vom CoMa wird ebenfalls ein Mechanismus angeboten, um Daten von einer IRS an eine oder mehrere IRSs in einem geografischen Gebiet zu senden. Dieser Mechanismus wird im Bundle IRSGeoCast näher erläutert. Es wird mit diesem Bundle möglich sein, von einem Funktionsanteil Daten (Byte-Array oder serialisierbare Java-Objekte) an Funktionsanteile auf (einer) anderen IRS in einem geografisches Gebiet (Longitude, Latitude) zu senden.

Die Adressierung von Funktionen auf den einzelnen Plattformen geschieht anhand der Funktionsnamen, die als Property gesetzt werden müssen. Sind diese Funktionsnamen auf beiden Plattformen identisch, können Daten über den CoMa ausgetauscht werden.

Datenaustausch: VsZ – IRS

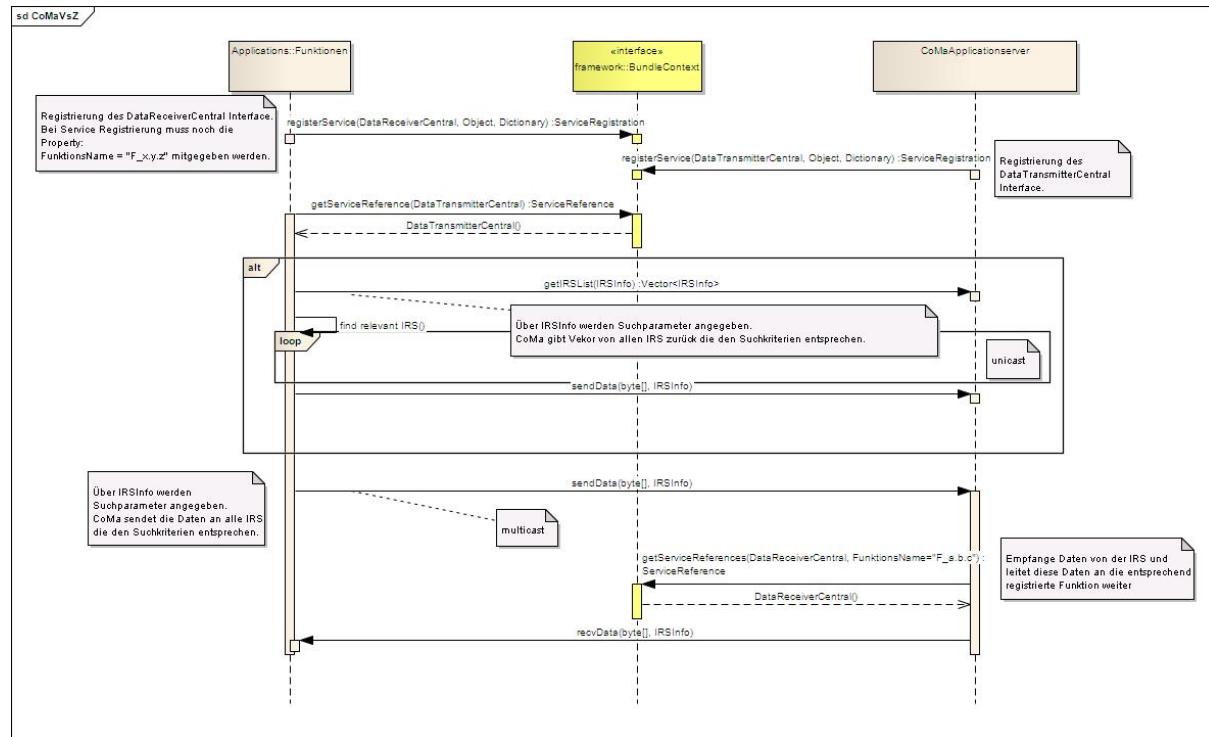


Abbildung 3-59: Datenaustausch: VsZ – IRS

Der Datenaustausch zwischen VsZ und IRS ist in Abbildung 3-59 gezeigt:

- Zunächst registrieren die Funktionen das DataReceiverCentral-Interface, um Daten von der IRS empfangen zu können. Bei der Registrierung muss der Funktionsname als Property mitgegeben werden.
- Anschließend registriert der CoMa in der VsZ das DataTransmitterCentral Interface, in dem alle Methoden zum Senden bereitgestellt werden.
 - Nach der Registrierung können die Funktionen den Service anfordern und die gewünschte Sendefunktion auswählen. Auf der VsZ besteht auch die Möglichkeit, sich Informationen über die IRS in verschiedenen Regionen (Stadt, Land, Autobahn), Positionen (GPS-Koordinate mit oder ohne Radius), oder mit verschiedenen Typen (mobil, stationär) anzufordern. Dies wird über das IRSInfoProvider-Interface ermöglicht. Ein möglicher Aufruf für den IRSInfo-Parameter wäre z.B.:


```
IRSInfo irs = new IRSInfo.Builder().position(17.7, 15.6, 2000.0).location(IRSLocation.CITY).type(IRSType.STATIONARY).build();
```
- Um Daten von der IRS empfangen zu können, muss das DataReceiverCentral-Interface implementiert sein. Der CoMa ruft die entsprechende Implementierung für die passende Gegenfunktion auf.

Datenaustausch: IRS – VsZ

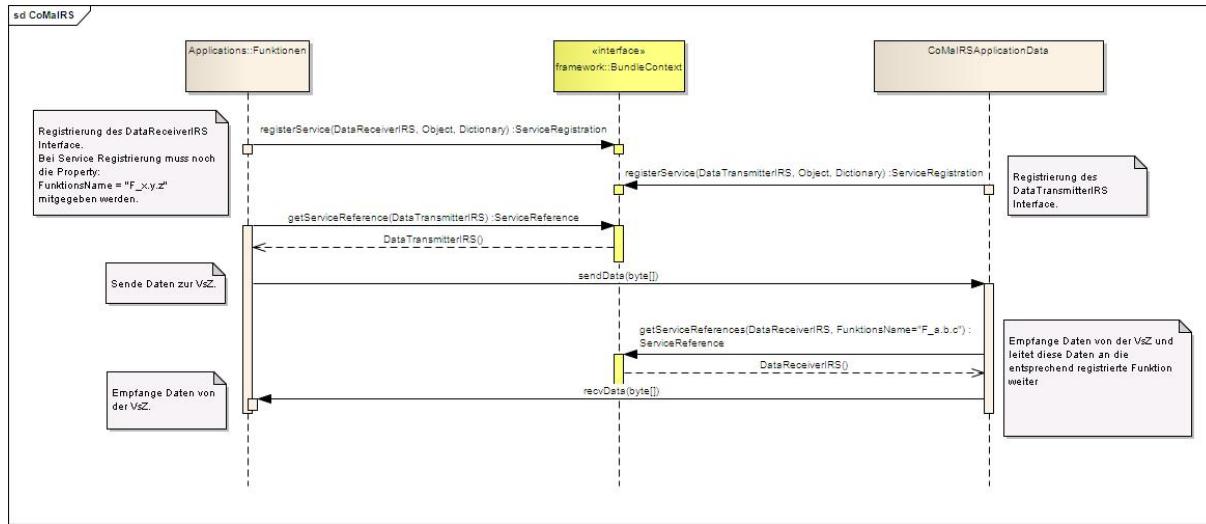


Abbildung 3-60: Datenaustausch: IRS – VsZ

Dieser Ablauf entspricht demselben wie beim Senden von Daten von VsZ zur IRS. Hier stehen jedoch andere Sendefunktionen zur Verfügung, bzw. eine Abfrage der IRS-Informationen ist nicht möglich, da IRS untereinander nur über das IRSGeoCast-Bundle kommunizieren können.

- Bei den Sendefunktionen kann alternativ die Zentral-ID mit angegeben werden, falls es in Zukunft mehrere VsZ gibt.
- Zum Senden wird das DataTransmitterIRS-Interface registriert. Zum Empfangen wird das DataReceiverIRS-Interface zur Verfügung gestellt. Dieses muss von den Anwendern implementiert werden.

Datenaustausch: IRS – IGLZ

Neben der Kommunikation IRS – VsZ wird ebenfalls eine direkte Kommunikation IRS – IGLZ gewünscht. Hierfür wird die bestehende Kupferinfrastruktur benutzt. Eine weitere Anforderung an den CoMa seitens der Stadt Frankfurt ist die Bereitstellung eines Sockets, der benutzt werden kann, um bestehende auf TCP/IP basierende Protokolle (OTS2) weiter zu verwenden.

Um diesen Mechanismus auf der IRS nutzen zu können, muss vom OSGi-Framework der Service SocketProvider angefordert werden. Durch Aufruf der Methode getSocket() wird ein Socket zur Verfügung gestellt, der in die Prioritätenkalkulation und Bandbreitenberechnung des CoMa mit eingeht. Der Socket ist mittels TLS/SSL verschlüsselt und benötigt auf der Gegenstelle einen SSL-Server. Da der Service lediglich einen Socket liefert, ist die Client/Server-Anwendung für die Handhabung der Kommunikation zwischen IRS und IGLZ verantwortlich. Auch wird keine Übertragung garantiert, sofern es von TCP/IP nicht geregelt ist. Nachdem eine Verbindung zur IGLZ aufgebaut wurde, können Daten bidirektional ausgetauscht werden. Im folgenden Diagramm ist der Ablauf verdeutlicht.

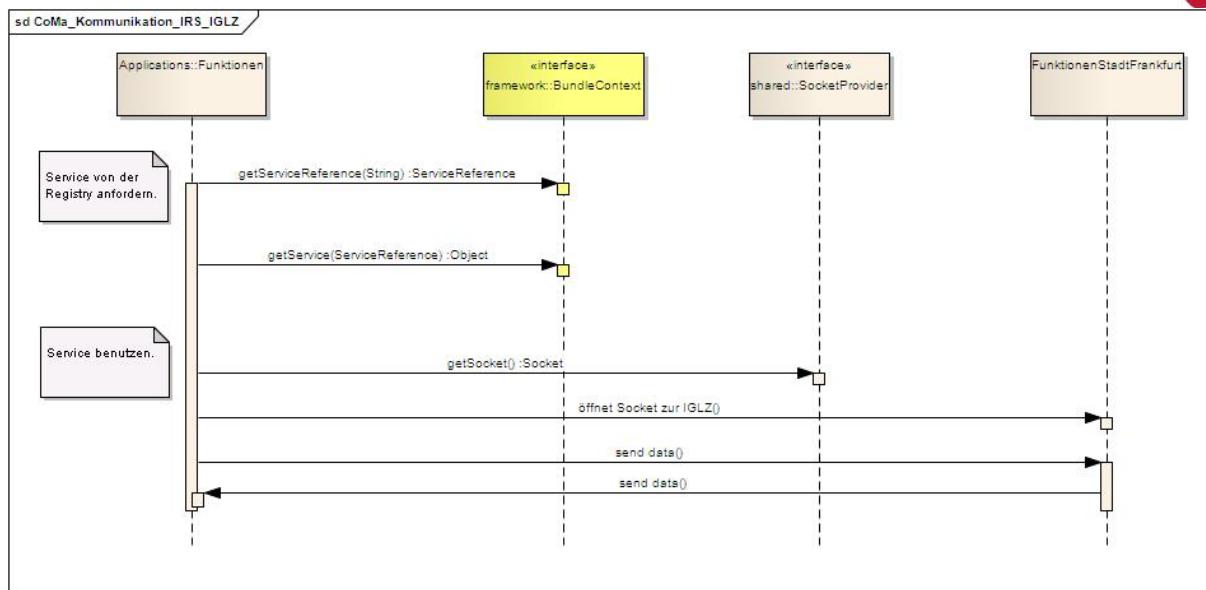


Abbildung 3-61: CoMa-Kommunikation IRS – IGLZ

3.3.3.6 Bundle IRSGeoCastClient

Beim IRS-Geocast können von einer IRS aus Daten an alle IRSs, die in einem angegebenen Gebiet liegen, gesendet werden. Diese Funktionalität wurde seitens der Funktionen gefordert. Dabei setzt diese Funktionalität auf dem CommunicationManager auf.

Hierzu verwenden die Funktionen den IRSGeoCastClient, welcher eine `sendGeoCast`-Funktion für Byte-Array sowie Java-Objekten zur Verfügung stellt. Die zu adressierenden IRS werden als Parameter `IRSInfo` (`IRSPosition`, `IRSLocation`, `IRSType`) dem Methodenaufruf mitgegeben. Dieses Bundle nutzt den CommunicationManager für die Übertragung eines GeoCast-Objektes (`IRSInfo`, byte-Daten, Funktionsname) an die Zentrale. Der Funktionsname wird über Properties beim Anfordern des Dienstes gesetzt. In der Zentrale wiederum existiert der IRS-GeoCastDistributor, welches den CoMa nutzt, um die Daten an die gewünschten IRS weiterzuleiten.

Der GeoCastClient setzt als Property den übertragenen Funktionsnamen. So wird sichergestellt, dass die Adressierung der Funktionen auf den verschiedenen IRS garantiert wird. Dabei bekommen die adressierten IRSs keine Informationen darüber, wer der ursprüngliche Sender der Information war. Diese Information muss die Funktion auf der Senderseite mit in die Nutzdaten aufnehmen.

Um das Weiterleiten der Nachricht an die Absender-IRS (IRS, die den Geocast initiiert hat) zu verhindern, fordert der IRSGeoCastDistributor in der Zentrale alle gewünschten IRS in dem spezifizierten Gebiet über die `getIRSLIST`-Funktion an. Anschließend wird die Absender-IRS aus der Liste herausgefiltert und für jede IRS aus der Liste die `Send`-Funktion mit der entsprechenden IRS-ID (vom IRSGeoCast-Bundle im `IRSInfo`-Objekt gesetzt) aufgerufen.

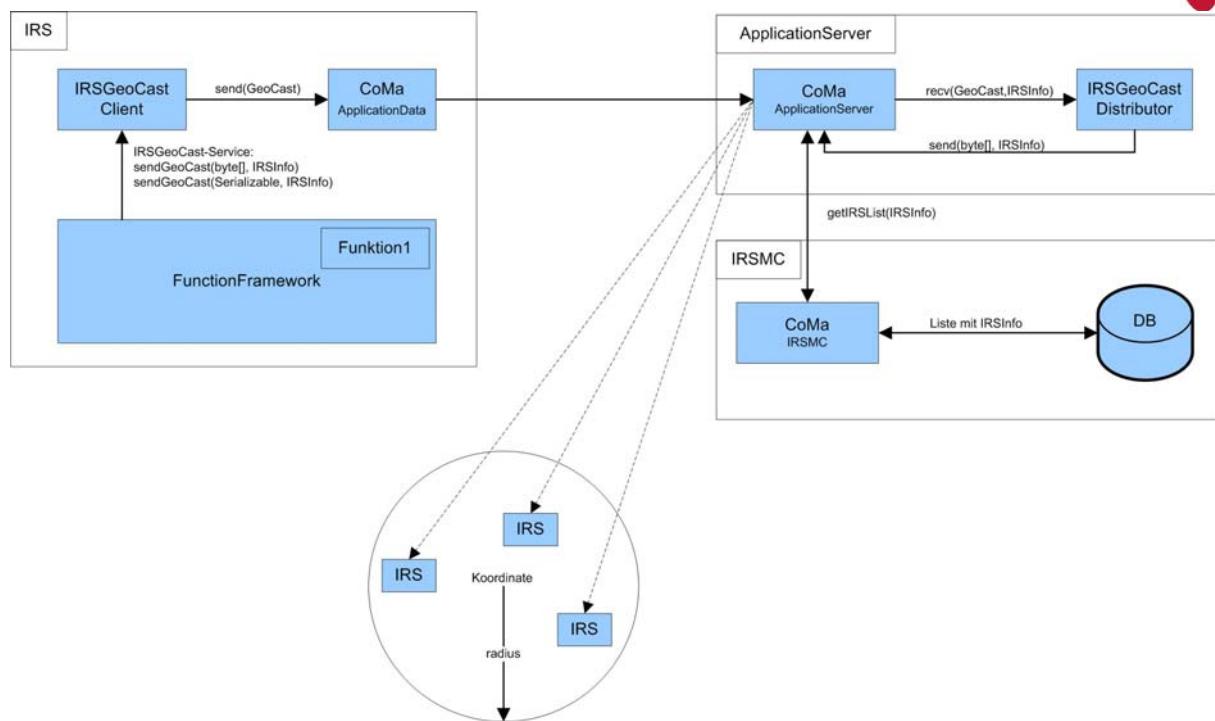


Abbildung 3-62: IRSGeoCast

3.3.4 IRS Management Framework

Die Bundles und Services im IRS Management Framework dienen der Verwaltung der IRS, im besonderen des FunctionFrameworks, und zur Kommunikation mit dem IRS Management Center. Alle IRS-Management-Bundles auf der IRS werden in einem eigenen Framework installiert und ausgeführt. Damit haben die Management-Bundles die Möglichkeit auf Fehlverhalten des FunctionFrameworks zu reagieren.

3.3.4.1 CMC

Der Configuration Management Client erhält Management Anweisungen vom Configuration Management Server. Diese Anweisungen umfassen zum einen die Verwaltung der Plattform selbst sowie zum anderen die Verwaltung der IRS-seitigen Funktionsanteile.

In Bezug auf die Plattform wird so sichergestellt, dass Aktualisierungen des Betriebssystems und des FunctionFramework durchgeführt werden können. Funktionsanteile können aus einem zentralen Repository installiert werden und anschließend durch die Zentrale gestartet bzw. gestoppt werden.

3.3.4.2 FMC

Das Fehlermanagement der IRS ist ein Prozess zur Fehlererkennung und Überwachung. Mit der Inbetriebnahme des Systems sind bereits Teile des Fehlermanagements aktiv und überwachen den korrekten Systemstart. Diese Komponente überwacht ebenfalls die korrekte Funktionalität des FunctionFramework.

3.3.4.3 NMC

Der Network Management Client ist für die Überwachung der Netzwerkschnittstellen auf der IRS und der Verbindungen zu den angeschlossenen Komponenten verantwortlich. Er erhält seine Konfiguration durch den CMC und leitet auftretende Fehler an den FMC bzw. das Logging weiter.

3.3.4.4 IRS-LOG

Das IRS Logging Bundle stellt den Management Funktionalitäten im FunctionFramework und allen Bundles im IRS Management Framework auf der IRS Schnittstelle zur Verfügung, über die Statusinformationen und auftretende Fehler zur Zentrale versendet werden können. Diese Informationen werden von FMS dann zentralseitig weiter verarbeitet. Kommt es zu einem Abbruch der Verbindung zur Zentrale so werden alle Log-Nachrichten automatisch in eine Datei gespeichert. Die Komponenten, die den IRS-LOG Service benutzen, können nach wie vor loggen. Mögliche Fehler, die in dieser Zeit auftreten, versucht das FMC zu lösen. Besteht die Verbindung wieder, werden die in der Datei gespeicherten Log-Nachrichten an die Zentrale geleitet. Über IRS-LOG werden keine funktionsbezogenen Daten geloggt. Hierfür wird das in Arbeitspaket AP24 entwickelte Projekttestsysteem benutzt.

3.3.4.5 CoMa

Der CoMa hat zwei Instanzen auf jeder RAU. Die Instanz im FunctionFramework übermittelt alle funktionsbezogenen Daten zum in der VsZ stehenden Application Server und umgekehrt. Im IRS Management Framework stellt der CoMa einen Kommunikationskanal bereit für alle in diesem Framework laufenden Bundles. Der Endpunkt dieses Kommunikationskanals ist jedoch das IRSMC. Weitere Informationen zum Ablauf sind im Kapitel 3.3.3.5 erläutert.

3.3.4.6 RessourceManager

Dieser Manager soll sowohl auf den einzelnen IRS als auch in der VsZ eine priorisierte Vergabe der zur Verfügung stehenden Bandbreite gewährleisten. Um den Betrieb einer IRS zu garantieren, muss es zu jeder Zeit möglich sein, auf die Management Funktionalitäten einer IRS zugreifen zu können. Der Ressource-Manager sorgt dafür, dass für das Management einer IRS immer genügend Bandbreite bereit steht. Dieser Wert ist jedoch auch nach oben begrenzt. Die Funktionen haben daher zu jeder Zeit ausreichend Bandbreite, um ihren Betrieb aufrecht zu erhalten.

Um dies zu gewährleisten, muss seitens des Ressource-Managers ständig eine Kanalschätzung durchgeführt werden, die die zuvor festgelegten Regeln und Zuweisungen der Bandbreite kontrolliert. Die Regeln agieren auf Betriebssystemebene und können daher von einem OSGi-Bundle nicht umgangen werden. Der Ressource-Manager kontrolliert den Kommunikationskanal IRS – IRSMC und IRS – ApplicationServer.

3.3.4.7 IRS-Funktionen

Die Intelligenz der IRS wird maßgeblich durch die darauf installierten Funktionen bestimmt. Alle Funktionen sind in das sog. FunctionFramework eingebettet, das den Funktionen einheitliche Schnittstellen zum Zugriff auf die Ressourcen der IRS bietet. Dem FunctionFramework unterliegen sämtliche Aufgaben und Bereiche, welche zum

reibungslosen Ablauf der Funktionen von Nöten sind. Das FunctionFramework ist somit Dienstleister und Plattform für alle IRS-seitigen Funktionen.

Die IRS Plattform installiert und konfiguriert die Funktionen selbsttätig. Dazu liegen die Funktionen und deren Konfiguration in einem Funktionsrepository in der Versuchszentrale vor.

Das FunctionFramework verwaltet und garantiert alle Zugriffe der IRS-Funktionen auf die Ressourcen der IRS, wie beispielsweise Rechenzeit, Arbeitsspeicher, Festspeicher und Bandbreite. Die Ressourcen jeder Funktion sind begrenzt und werden überwacht.

Basisfunktionen

Durch Basisfunktionen werden grundlegende Dienste zur Verfügung gestellt, die von mehreren Funktionen benötigt werden. Beispiele für solche Basisfunktionen sind die Application Support Facilities (CAM/Neighbourhood Table, DEN) oder eine Proxyfunktion, die Nachrichten zwischen Zentrale und Fahrzeug weiterleitet.

Funktionen sind in Hauptfunktionen eingeteilt, um den Austausch von Daten zwischen Hauptfunktionen zu vermeiden. Ist es dennoch erforderlich Daten zwischen Hauptfunktionen auszutauschen, muss dies über einen funktionseigenen OSGi-Service geschehen. Einer der Kommunikationspartner registriert ein Service im OSGi-Framework, den der andere Kommunikationspartner nutzen kann. Somit lassen sich Daten zwischen Hauptfunktionen austauschen.

IRS-seitige Funktionsanteile

Zur Umsetzung eines Anwendungsfalls, der Fahrzeug, IRS und Zentrale beinhaltet, muss jeder dieser Akteure einen bestimmten Funktionsanteil implementieren. Dieser wird als OSGi-Bundle entwickelt und kann nach einer erfolgreichen Testphase vom IRSMC auf jede IRS (FunctionFramework) eingespielt werden. Dies soll im Folgenden beispielhaft anhand von fahrzeugseitig generierten Gefahrenmeldungen verdeutlicht werden, die an die VsZ zu übermitteln sind: Ein Fahrzeug detektiert eine Gefahr, woraufhin der fahrzeugseitige Funktionsanteil eine Meldung generiert. Diese Meldung wird mittels DEN Nachricht an eine IRS übertragen.

Der IRS-Funktionsanteil hat sich für alle DEN-Nachrichten an der Umfeldtabelle registriert. Die ausgesendete DEN-Nachricht gelangt über die CCU und den CommunicationClient in die Umfeldtabelle. Diese bedient den C2XMessageHandler des IRS-Funktionsanteils. Diese aggregiert sie gegebenenfalls mit anderen gleichartigen Nachrichten. Die aggregierten Nachrichten werden dann an die Testzentrale weitergeleitet. Gleichzeitig kann die IRS auch Store and Forward für das Fahrzeugnetz abbilden und so die Nachrichten sich annähernden Fahrzeugen zur Verfügung zu stellen.

Ein zentralseitiger Funktionsanteil empfängt die aggregierten DEN-Nachrichten und wertet diese aus. Anschließend können dann entsprechende Maßnahmen veranlasst werden, wie beispielsweise der Vorschlag zur Begrenzung der Geschwindigkeit, Einleiten von Umleitungen, Verteilen der Warnmeldung über weitere IRS, Benachrichtigen eines Operators oder ähnliches.

3.3.5 IRS Management Center

3.3.5.1 Allgemeine Aufgaben eines IRS-Management

IRSS werden an strategischen Punkten platziert und dort fest installiert. Ein typischer Standort für eine IRS wäre zum Beispiel an einer Autobahnschilderbrücke oder eine Lichtsignalanlage (LSA). Weiterhin sollen IRS mehrere Jahre dort stationiert sein. Während dieser Zeit wird sich die Software für diese IRS mehrmals ändern, daher muss es möglich sein Funktionsaktualisierungen oder gänzlich neue Funktionen auf den IRS zu installieren.

Die benötigte Flexibilität verbunden mit dem erschwerten Zugang für Wartungspersonal macht es erforderlich, dass alle IRS „ferngesteuert“ gewartet werden können. Wäre dies nicht möglich, müsste in einem Fehlerfall oder bei Änderungen an der Software eine Wartung vor Ort erfolgen. Dies ist bei der hohen Anzahl der IRS ein nicht zu vertretender zeitlicher und finanzieller Aufwand.

Im Gegensatz zur IVS agiert kein Benutzer direkt mit der IRS. Tritt ein Fehler in einem Funktionsanteil oder sogar an der Plattform selbst auf, ist die Kommunikation zwischen Fahrzeugen und Zentrale gegebenenfalls beeinträchtigt. Jedoch ist weder für die Fahrzeuge noch für die Zentrale dieser Fehler ersichtlich, da sie nicht wissen können, ob keine Nachrichten vorhanden ist oder ob die Kommunikation gestört ist. Daher ist es essentiell, dass die Funktionalität der IRS ständig überwacht wird und gegebenenfalls Maßnahmen zur Fehlerbehebung ergriffen werden.

Diese Anforderungen an die IRS machen eine zentrale Instanz notwendig, von der aus alle IRS überwacht und gesteuert werden können. Dieses IRS Management Center (IRSMC) enthält das Funktionsrepository, die Fehlerüberwachung, Strategien zur Fehlerbehebung und stellt dem Administrator eine Konfigurationsoberfläche zur Verfügung. Außerdem sind im IRSMC alle Informationen zu den IRS gespeichert, so dass diese vollständig konfiguriert werden können. Die Aufgaben des IRS Management Systems sind in verschiedene Teilsysteme aufgeteilt. Diese sind in Abbildung 3-63 zu sehen.

Konfigurationsdaten werden zusammen mit den anfallenden Daten der einzelnen Systeme in der zentralen Datenbank gespeichert und können so von allen Teilsystemen genutzt werden. Dies betrifft allerdings nur Daten, die für die Verwaltung der IRS benötigt werden. Daten zur Versuchsdurchführung oder Daten der Funktionen werden hier nicht gespeichert. Die Software auf den einzelnen IRSS wird durch den Configuration Management Server (CMS) verwaltet. Der Fault Management Server (FMS) überwacht alle IRS und Teilsysteme und veranlasst im Fehlerfall Maßnahmen zur Fehlerbehebung. Sowohl CMS als auch FMS kommunizieren nur verschlüsselt mit den IRS. Den verschlüsselten Kommunikationskanal zwischen IRS und Versuchszentrale stellt der Communication Manager (CoMa) zur Verfügung. Der CoMa wird sowohl vom IRS-Management als auch von den Funktionsanteilen benutzt.

Der Administration and Monitoring Interface Server (AMIS) bietet den zentralen Funktionsanteilen eine Schnittstelle zum IRS Management System, um beispielsweise den Status aller IRS abrufen zu können. Gleichzeitig erlaubt er den Administratoren der IRS einen Vollzugriff auf das Verwaltungssystem.

In Abbildung 3-63 ist der schematische Aufbau des IRS Management Centers dargestellt. Hier sind alle vorhandenen Schnittstellen zwischen den Komponenten und Subsystemen zu erkennen. Alle Schnittstellen, die in der Abbildung bei den jeweiligen Zugangspunkten der Komponenten bezeichnet sind (Blaue Rechtecke), werden unter Abschnitt 3.3.7 genauer

definiert. Im Folgenden werden alle Komponenten des IRS Managements in der Versuchszentrale beschrieben.

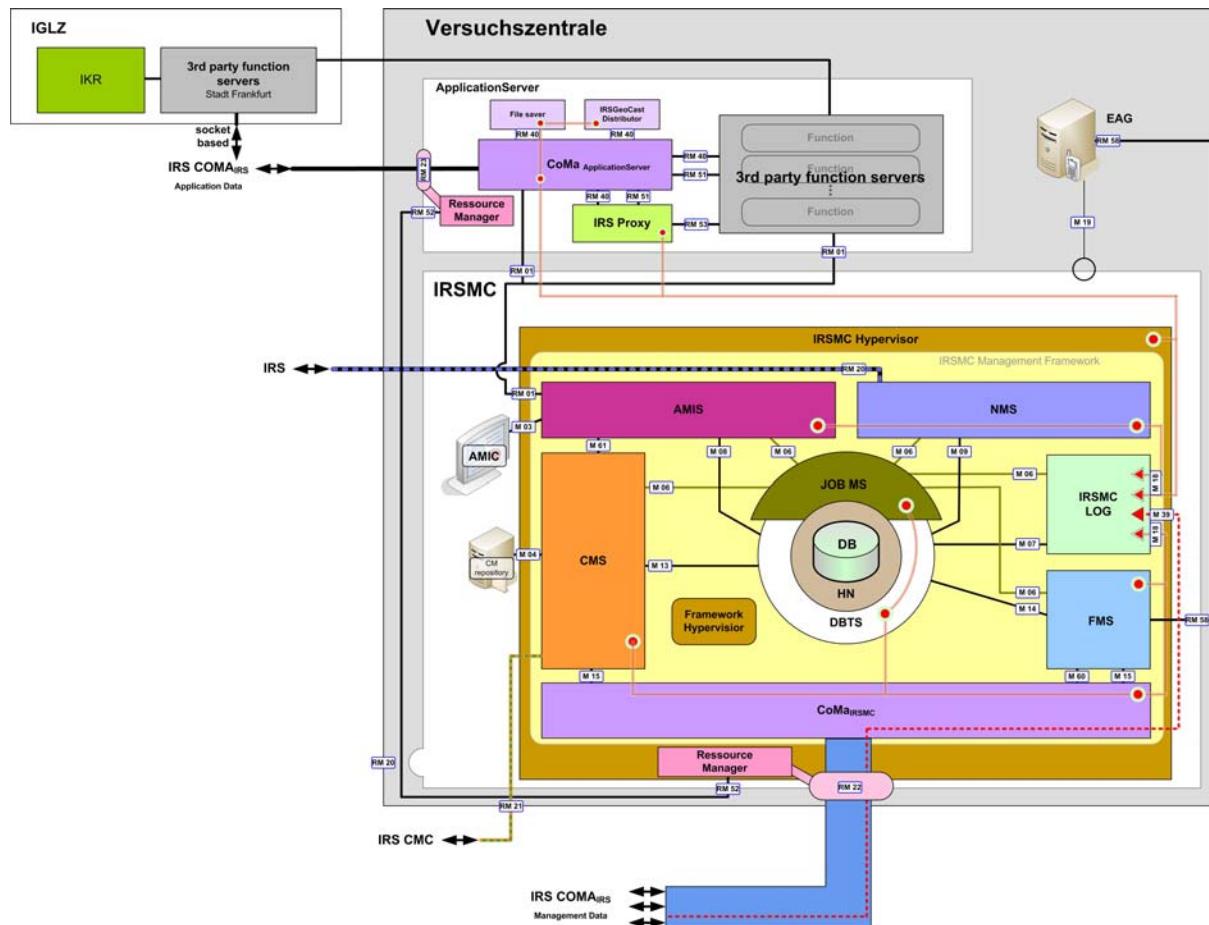


Abbildung 3-63: IRS Management Center in der VsZ

3.3.5.2 Physikalischer Aufbau des IRS Management Centers (IRSMC)

Der zentrale Verwaltungs- und Steuerungspunkt des IRS Managements ist das IRSMC. Hier laufen die Verbindungen aller IRS in sim^{TD} zusammen. Aufgabe dieses Backends ist nicht nur die Verarbeitung von IRS-seitigen Managementanfragen, sondern auch die Verwaltung und Überwachung des gesamten Systems.

Bei der Konzeption des IRSMCs spielt vor allem die Skalierbarkeit des Systems eine entscheidende Rolle; es soll flexibel erweiterbar, hochverfügbar und auf neue organisatorische Bedingungen anpassbar sein (siehe Abbildung unten).

Im Zentrum des IRSMC steht der IRS-Management-Server (IRSMS). Er bearbeitet auf mehreren Instanzen lastverteilt alle Anfragen und notwendigen Prozesse. Die Verteilung der Anfragen findet vor den RMS statt. Hier kommen die Anfragen der IRS an und werden von den so genannten Load-Balancern gleichmäßig auf die vorhandenen RMS verteilt.

Die eingesetzte Struktur der vorgesetzten Load-Balancer ist eine Master-Worker-Topologie. Fällt der Master aus oder ist nicht mehr erreichbar, so übernimmt der Worker die Funktion der Lastverteilung.

Alle im IRSMS anfallenden Daten werden in einer Datenbank gespeichert. Dabei nimmt das auf dem IRSMS laufende Modul: Database Transaction Server (DBTS) alle Anfragen entgegen und leitet diese an einen MySQL-Cluster weiter.

Um das spätere System besser skalieren zu können, wurden die drei Servervarianten: LoadBalancer, IRSMS und Datenbankserver virtualisiert auf einen Server untergebracht. Somit wird die Hardware besser ausgenutzt und das System lässt sich durch zusätzliche VMWare ESX Server erweitern.

Alle Daten, die in irgendeiner Form von den VMWare ESX Servern benötigt werden, werden auf einem IRSMC-eigenen SAN abgespeichert. Hier sind alle VMs und die MySQL Datenbank zu finden. Bei zusätzlichem benötigtem Speicherplatz kann somit an zentraler Stelle Speicherplatz angeschafft werden, der dann von allen VMWare ESX Servern benutzt werden kann.

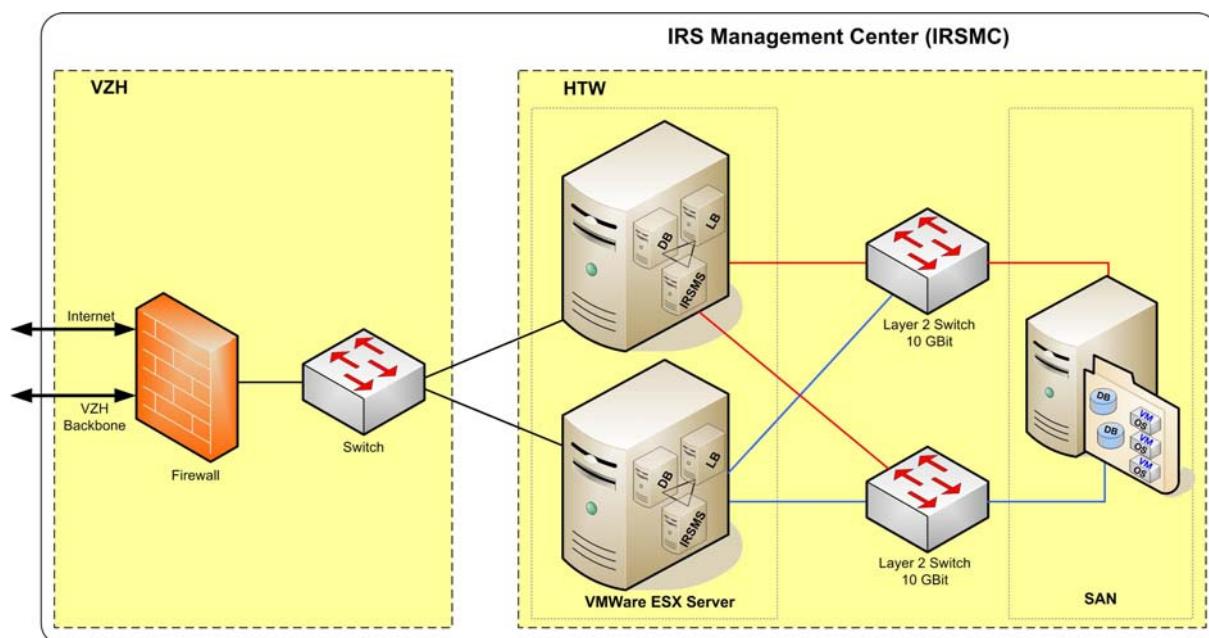


Abbildung 3-64: Physikalische Übersicht des IRSMC

3.3.5.3 Subsysteme des IRSMC

CMS

Der Configuration Management Server erstellt anhand der Informationen aus der zentralen Datenbank sowie vom Administrationsinterface Managementanweisungen für die verschiedenen CMC.

FMS

Das Fehlermanagement im IRSMC ist ähnlich aufgebaut wie das Fehlermanagement der IRS. Seine Aufgaben teilen sich in zwei Themenbereiche auf. Die Überwachung der serverseitigen Prozesse und die Überwachung der IRS. Das FMS registriert sich bei der Komponente IRSMC LOG auf alle im IRS-Gesamtsystem anfallende Log-Nachrichten. Somit hat der FMS über die gesamte Zeit den Überblick über den Status aller Module auf der IRS sowie derer im IRSMC.

AMIS

Der Administrative Monitoring Interface Server (AMIS) stellt die Schnittstelle zur Administration dar. Durch ihn können Daten zwischen Datenbank und der Webanwendung Administrative and Monitoring Interface Client (AMIC) ausgetauscht werden. Geplant ist hier ein Monitoring mit Webschnittstelle zum Benutzer. Eine weitere Funktionalität des AMIS ist die Bereitstellung einer API, die von Funktionsentwicklern benutzt werden kann. Daten die normalerweise über die Web-Schnittstelle ausgetauscht werden, können damit direkt in einen Programmablauf zu integrieren werden. Welche Daten diese API bereitstellt, muss noch festgelegt werden. Hier wird sich sehr stark an Anforderungen orientiert, die seitens der Funktionsentwickler festgelegt wurden.

NMS

Der Network Management Server überwacht die im IRSMC vorhandenen Netzwerkschnittstellen und netzwerktechnische Hardware. Das bedeutet, dass die Server, die im IRSMC installiert sind, sowie alle IRSs überwacht werden. Das IRSMC hat eine direkte Verbindung zur RAU, die Teil der IRS ist. Somit lässt sich nur der Kommunikationskanal IRSMC – RAU entsprechend überwachen. Die einzelnen Teilkomponenten der IRS sind alle an die RAU angeschlossen. Durch das auf der RAU installierte NMC können auch diese Teilkomponenten überwacht werden. Im Fehlerfall meldet der NMC über das IRS Loggingsystem den Fehler, der dann vom FMC erkannt wird.

CoMa

Dieser CoMa ist im IRSMC für den Kommunikationskanal IRS Management Framework – IRSMC zuständig. Nähere Information über den Ablauf ist unter Kapitel Bundle CoMa zu finden.

IRSMC LOG

Der IRSLOG enthält alle serverseitigen Log-Nachrichten und schreibt diese in die Datenbank. Darüber hinaus nimmt er auch alle von den IRS kommenden Log-Nachrichten entgegen und schreibt diese ebenfalls in die Datenbank. Beim erfolgreichen registrieren des FMS am IRSLOG werden alle Log-Nachrichten als Kopie auch noch an den FMS geleitet.

Ressource Manager

Diese Instanz des Ressource Managers übernimmt die Bandbreitenanpassung für den Kommunikationskanal IRS – IRSMC. Er tauscht sich mit der Instanz auf dem Application Server aus, um eine Bandbreitenberechnung für jede einzelne IRS durchzuführen und gegebenenfalls anzupassen. Weitere Informationen sind im Kapitel RessourceManager zu finden.

IH

Der IRSLOG Hypervisor überwacht die Komponenten der IRS Management Server und startet die Komponenten im Fehlerfall neu.

JOB MS

Der Job Management Server verwaltet alle aktuell anstehenden Aufgaben des Management Center. Über ihn ist es möglich, den aktuellen Stand aller ausstehen Aufgaben des

Management Centers zu erhalten. Um das IRSMC skalierbar zu halten, müssen alle Aufgaben als Jobs in der Datenbank abgelegt werden. Nur so kann sichergestellt werden, dass eine CMS-Instanz auf einem Server die Arbeit einer fehlerhaften CMS-Instanz übernehmen kann.

DB

Die Datenbank speichert alle im IRS-Gesamtsystem anfallenden Daten. Die Datenbank ist hochverfügbar ausgelegt. In der IRS findet jedoch eine Aggregation der Daten statt, bevor sie zum IRSMC übertragen werden. Werden diese Rohdaten benötigt, kann lokal auf der IRS auf diese Daten zurückgegriffen werden.

Die Datenbank besteht aus mehreren MySQL-Clustern. Die Anfragen werden bereits in der Datenbank-Zugriffsstufe auf alle verfügbaren MySQL-Cluster lastverteilt.

HN

Hibernate stellt einen Persistenzlayer für den Zugriff auf die Datenbank her. Über ihn laufen alle Anfragen zur Datenbank.

DBTS

Der Database Transaction Server regelt den Zugriff der einzelnen Management-Server-Komponenten auf die DB. Er kümmert sich um die Lastverteilung auf die einzelnen MySQL-Cluster.

EAG

Das Error Alert Gateway dient zur Eskalation von kritischen Systemzuständen an den verantwortlichen Administrator. Um unabhängig vom bereitgestellten Ethernet zu sein, wird hier ein GSM/GPRS-Gateway zum Einsatz kommen. Auftretende Fehlermeldungen können direkt per SMS an den verantwortlichen Administrator des Systems verschickt werden. Des Weiteren lassen sich verschiedene digitale Eingänge mit Signalen belegen, die dann vordefinierte Meldungen per SMS aussenden. Im Falle eines Netzwerkproblems besteht die Möglichkeit, sich per GSM über das Gateway einzuhüpfen und direkt Befehle am Server abzusetzen.

IRSGeoCastDistributor

Der IRSGeoCastDistributor ist die serverseitige Komponente des IRSGeoCast-Moduls. Dieses Bundle empfängt die Daten des IRSGeoCastClients und generiert aus diesen Daten den Geocast, welcher über den CoMa übertragen werden.

In Abbildung 3-65 zeigt das Deployment-Diagramm der Versuchszentrale. Die Darstellung beinhaltet nicht alle Komponenten, die in Abbildung 3-63 enthalten sind. Für das Verständnis der Versuchszentrale und für den besseren Überblick sind nur die wichtigsten Komponenten des IRS Managements dargestellt.

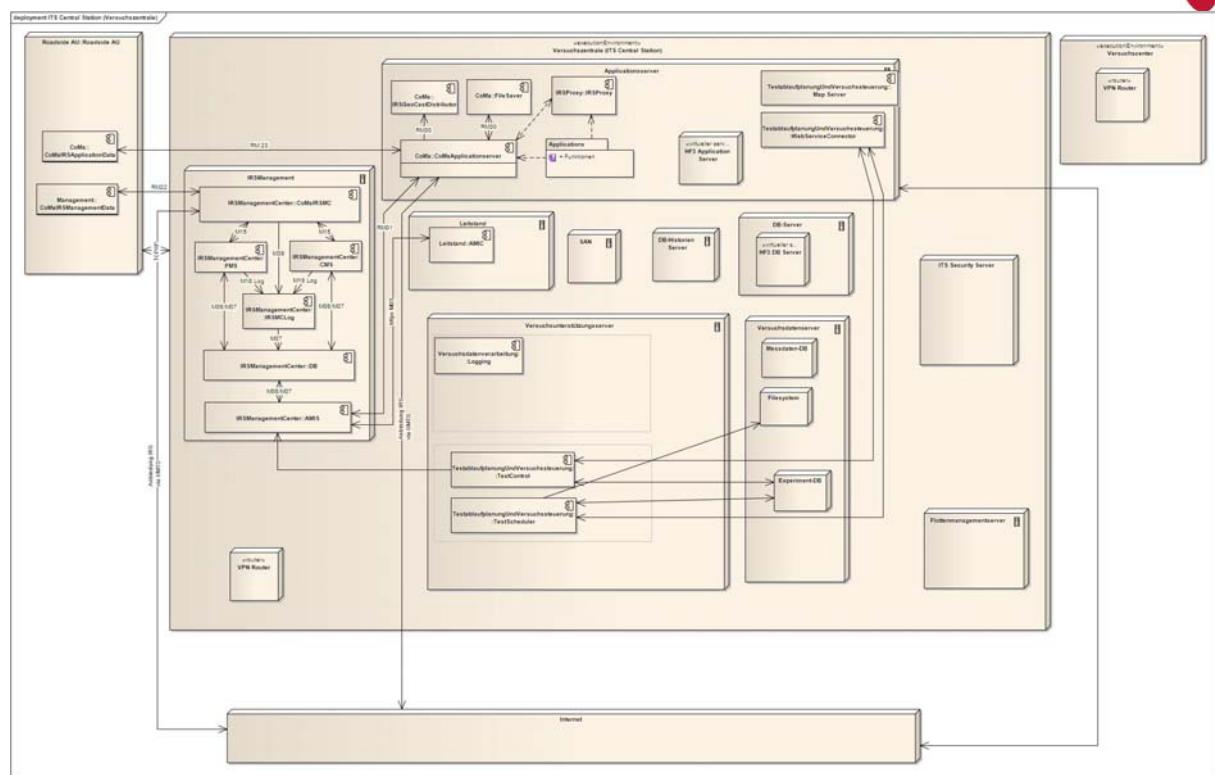


Abbildung 3-65: Deployment-Diagramm der Versuchszentrale

3.3.6 ApplicationServer

3.3.6.1 AMIC

Der Administrative and Monitoring Interface Client (AMIC) ist die Nutzerschnittstelle zum Administrative Monitoring Interface Server. AMIC ist eine web-basierte Anwendung, die den Administratoren die Möglichkeit bietet, mittels einer GUI verschiedene Administrationsaufgaben auszuführen. Des Weiteren können über ihn der Status der IRS und der darauf laufenden Funktionen abgefragt werden.

3.3.6.2 IRSProxy

Ein Teil der in sim^{TD} realisierten Funktionen haben einen sehr starken zentralen Bezug. Jedoch benötigen diese Funktionen auch Informationen aus dem Versuchsgebiet, die keine Aggregation auf einer IRS benötigen. Das bedeutet, dass verschiedene Warnmeldungen ohne Änderung durch den IRS-seitigen Funktionsanteil in die Zentrale gesendet werden. Das übliche Vorgehen hierbei wäre die Entwicklung eines IRS-seitigen Funktionsanteils. Dieser Funktionsanteil würde lediglich die Nachricht vom C2X-Netzwerk über die IRS aufnehmen und unverändert an die Zentrale weiterleiten. Es wurde erkannt, dass dieser Mechanismus von mehreren Funktionen benötigt wird. Um nicht unnötig Systemressourcen zu allozieren, wurde für dieses Problem das Proxy-Konzept entworfen. Zentralseitige Funktionen können über die Komponente IRSProxy auf den Proxy-Service einer IRS zugreifen. Die Funktionen in der Zentrale können sich dabei für ausgewählte C2X-Nachrichten über den IRSProxy registrieren oder bestimmte C2X-Nachrichten über eine IRS aussenden lassen.

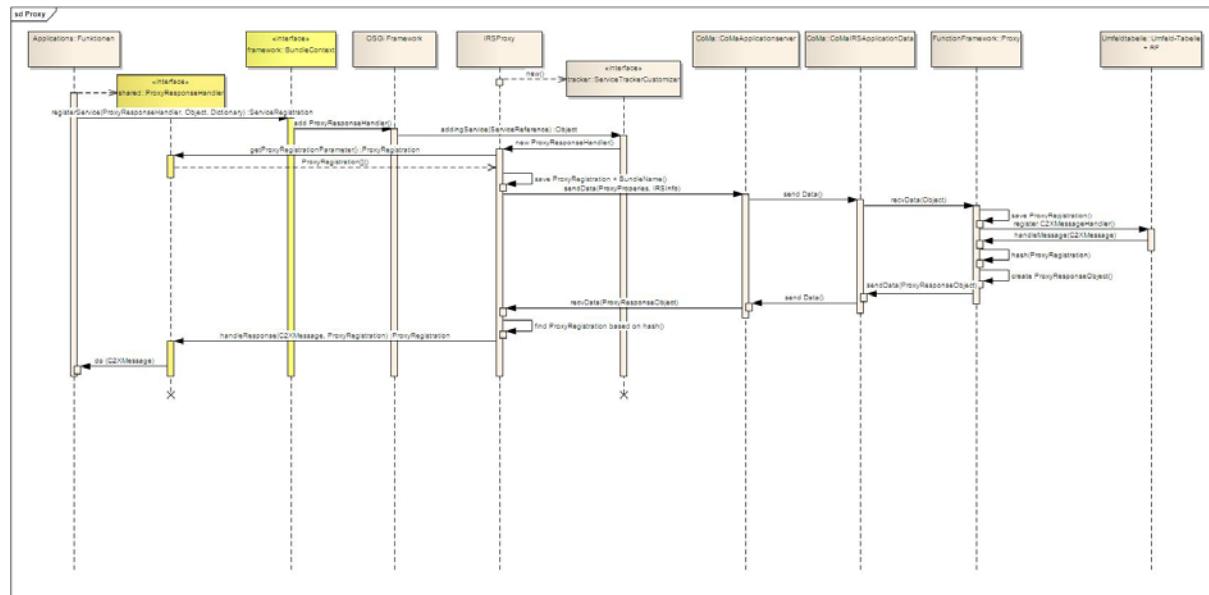


Abbildung 3-66: Empfang von C2XMessages über IRSProxy

3.3.7 Schnittstellen

3.3.7.1 Schnittstelle IRS – IRS

Eine direkte Kopplung von zwei oder mehreren IRS untereinander ist vom Konzept nicht vorgesehen. Die Anforderung an das IRS-Gesamtsystem wurde jedoch seitens der Funktionen formuliert. Um Ressourcen in der Versuchszentrale zu schonen, wird eine IRS-Geocast Funktionalität eingerichtet. Mit diesem Service ist es IRS-seitigen Funktionen erlaubt, Daten von anderen Instanzen ihrer Funktion auf anderen IRS auszutauschen. Die Funktionalität kann auf der IRS über die Komponente IRSGeoCastClient (siehe Bundle IRSGeoCastClient) abgerufen werden. Dieser Dienst abstrahiert das schon vom CoMa bereitgestellte Interface um die Angaben des geografischen Gebiets. Die Nachricht wird zusammen mit den Positionsinformationen über das zu adressierende Gebiet an die zentralseitige Komponente des IRSGeoCast, den IRSGeoCastDistributor in die VsZ gesendet (siehe IRSGeoCastDistributor im Kapitel 3.3.5.3).

Ist jedoch zentralseitig eine weitere Bearbeitung oder Aggregation der Daten notwendig, kann dies nicht über den GeoCast-Service abgedeckt werden. Hierfür muss ein eigener Funktionsanteil für die VsZ entwickelt werden.

3.4 IGLZ

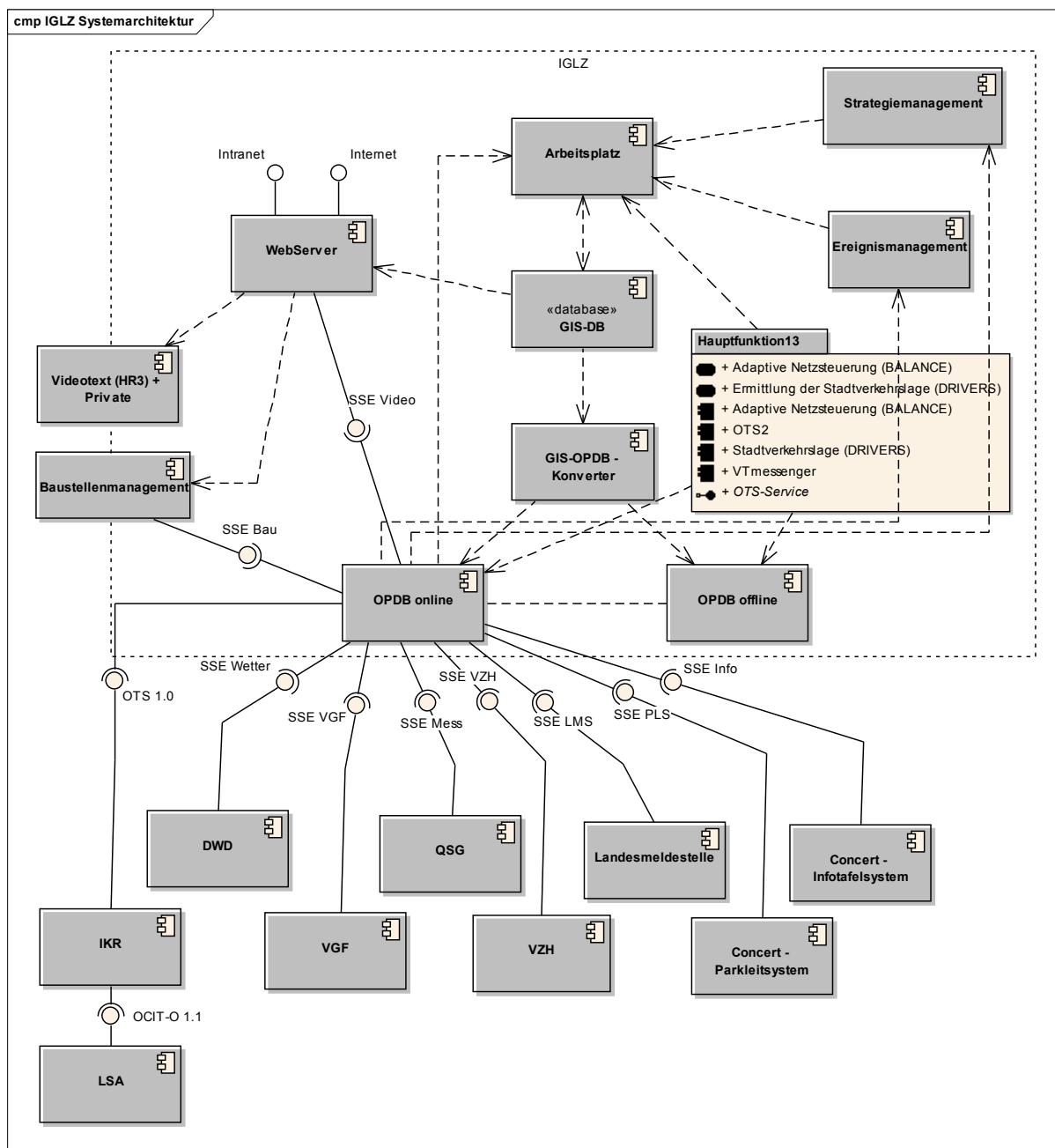


Abbildung 3-67: IGLZ-Systemarchitektur

Das Kernsystem der IGLZ besteht aus der operativen Datenbank (OPDB), in der sämtliche Verkehrsinformationen und Verkehrsdaten zusammengeführt werden. Intern sind weitere Komponenten angeschlossen:

- Strategiemanagementsystem
- Ereignismanagementsystem
- Webserver
- Arbeitsplätze zur Verwaltung, Kontrolle und Modifizierung der Verkehrsdaten und Verkehrsinformationen

Weitere Informationen bekommt / gibt die OPDB von / an folgenden externen Systemen:

- IKR (IGLZ Kommunikationsrechner, der die Verwaltung und Steuerung der Lichtsignalanlagen übernimmt)
- DWD (Deutscher Wetter Dienst)
- VGF (Verkehrsgesellschaft Frankfurt)
- QSG (Verkehrsdaten Strategischer Detektoren)
- VZH (Verkehrszentrale Hessen)
- Landesmeldestelle
- Parkleitsystem (Siemens)
- Infotafelsystem (Siemens)

Die neuen Komponenten der Funktion F_1.3.2 werden in der IGLZ realisiert und über externe Schnittstellen an die Infrastrukturkomponenten (Detektoren, ITS Roadside Station) und an die sim^{TD}-Versuchszentrale angeschlossen.

3.5 VZH

Die Verkehrszentrale Hessen (VZH) ist das amtliche Wirkssystem für die Steuerung des Straßenverkehrs auf hessischen Fernstraßen. Sie ist unabhängig von simTD. Konkret bedeutet diese Aussage, dass keine simTD-Funktionalität in der VZH implementiert wird. Damit erfolgen auch keinerlei Eingriffe in die Abläufe der VZH.

3.6 Versuchszentrale (VsZ)

Zwei Komponenten der Versuchszentrale sind die Testablaufplanung und Versuchssteuerung. Beide werden im Rahmen der Versuchsunterstützung realisiert und sind in Abschnitt 2.4 beschrieben.

Im Pflichtdokument D23.1 ist die simTD-Versuchszentrale detailliert beschrieben.

3.6.1 Organisatorisches

Im August wurde die Ausschreibung der simTD-Versuchszentrale (Stufe 4) veröffentlicht. Der Submissionstag ist der 24.9.2009. Nach Sichtung und Auswertung der eingegangenen Angebote gemäß einem den Bieter bekanntenen Katalog zur Bewertung der Leistungspunkte, sowie nach Durchführung der Bietergespräche wird der Auftragnehmer ermittelt. Er beginnt am 1.11.2009 mit den Arbeiten. Die simTD-Versuchszentrale soll am 31.10.2010 betriebsbereit sein.

3.6.2 Hardware

In der Ausschreibung der Versuchszentrale (Stufe 3) wurde die Hardware ausgeschrieben. Sie ist bereits beschafft, in der VsZ installiert und abgenommen. Sie besteht aus 5 SPARC Servern der Modellreihe T5140, auf denen als Betriebssystem Solaris 10 installiert ist. Ferner gibt es einige Windows-basierte Arbeitsplatzrechner. Eine genaue Auflistung der in der VsZ installierten wurde im Februar 2009 auf www.ProjectPlace.de (TP2 --> AP23) veröffentlicht.

3.6.3 Software

Die Ausschreibung der Versuchszentrale (Stufe 4) – siehe oben – bezieht sich hauptsächlich auf Anwendungssoftware und systemnahe Software (Middleware). In begründeten Fällen kann der Auftragnehmer für bestimmte Funktionen zusätzliche Hardware anbieten.

Die Anwendungssoftware deckt die simTD-Funktionalität ab, die in der VHB bereits grob skizziert ist; die Middleware ergänzt die softwareseitige Ausstattung der VsZ durch systemnahe Software wie z.B. Datenbanken. Die simTD-Versuchszentrale ist als schlüsselfertiges System zu liefern.

Die zu beschaffende Anwendungssoftware deckt alle verkehrlichen Anforderungen einer leistungsfähigen Verkehrszentrale ab. Sie ist untergliedert in Dienste (services), die an einer zeitgemäßen Architektur ausgerichtet sind. Zum Einsatz kommen SOA (Service Oriented Architecture) und ESB (Enterprise Service Bus). Diese grundlegenden Architekturanforderungen sind durch in der Praxis bewährte Produkte zu verwirklichen. Ausdrücklich sind Open Source Produkte zulässig.

Für die Implementierung von Funktionalität, die sowohl auf den IRS als auch in der ICS zu realisieren ist, sind OSGi-Bundles zweckmäßig. In der ICS kommt als OSGi-Rahmenwerk *Eclipse Equinox* zum Einsatz. Der Verzicht auf kommerzielle OSGi-Rahmenwerke mit zusätzlicher Funktionalität erfolgte im Hinblick auf Lizenzkosten in einem eventuellen späteren Wirkbetrieb.

3.6.4 Internet-Anbindung

Die ICS erhält eine Internet-Anbindung, über die simTD-Partner von außen über VPN-Tunnel auf die Rechner der ICS zugreifen können (im Rahmen zuvor erteilter Berechtigungen).

Die IGLZ (Integrierte Gesamtleitzentrale) der Stadt Frankfurt wird ebenfalls an die ICS abgebunden. Über diese Verbindung erhält die ICS die aktuelle städtische Verkehrslage sowie die Rohdaten der städtischen IRS.

Die simTD-Versuchszentrale benötigt Daten (hauptsächlich FG-Daten) aus der VZH. Sie ist daher an der VZH lesend angebunden.

Die stationären simTD-IRS des Landes Hessen sind über Glasfaser an die ICS angebunden, die mobilen IRS (z.B. auf Baustellenanhängern installiert) über UMTS.

Es versteht sich von selbst, dass alle Anbindungen über Firewalls abgesichert sind (Paketfilter und Stateful Inspection).

3.6.5 Logging-Daten (nur Hinweis)

Die VHB ging davon aus, dass das Datenvolumen der Logging-Daten (Meßdaten), mit denen die Versuche ausgewertet werden, klein genug ist, um auf dem bereits beschafften SAN in der ICS gespeichert werden zu können.

In der Zwischenzeit zeichnet sich nach vielen Gesprächen mit den beteiligten simTD-Partnern ein wirklichkeitsnahes Volumen von 30 TB (Terabyte) ab. Diese Datenmenge ist hinsichtlich Speicherbedarf und Pflege zu betreuen; das HLSV ist für diese Aufgaben weder personell noch logistisch gerüstet. AP23, AP24 und AP41 erarbeiten ein tragfähiges Konzept für die noch ungelöste Aufgabenstellung.

3.6.6 IPv6-Fähigkeit

Jeder der fünf SPARC-Server der ICS ist mit mindestens zwei Netzwerkadapters ausgerüstet. Die Netzwerkadapter lassen sich unabhängig voneinander entweder mit IPv4 oder IPv6 betreiben. Im Hinblick auf einen eventuellen späteren Wirkbetrieb ist es wichtig, dass das Forschungsprojekt simTD wichtige Aussagen über eine IPv6-gestützte Kommunikation liefert. Obwohl die Telekom bundesweit nur IPv4 unterstützt, können IPv6-Pakete in IPv4-Paketen verpackt (getunnelt) und auf der Empfangsseite entpackt werden: Sendende und empfangende Instanzen im simTD-Umfeld arbeiten auf Grundlage von IPv6, während die Nutz- und Protokolldaten über IPv4 verschickt werden.

3.6.7 Netzwerkmanagement für IRS

Die HTW betreut über ihr RMC (Remote Management Centre) die IRS. Es kommt ein proprietäres Protokoll zum Einsatz.

3.6.8 ICS-seitiges Systemmanagement

Das HLSV beschafft für die ICS und die angeschlossenen Netzknoten ein Netzwerkmanagement auf der Grundlage von SNMPv3. Es wird auch Aufgaben aus dem Bereich Systemmanagement übernehmen.

3.6.9 Security

Die IT-Sicherheitsarchitektur benötigt eine zentrale Instanz, welche die Verbreitung der Pseudonyme der ITS Ad-hoc-Kommunikation kontrolliert und verwaltet. Darüber hinaus kann diese Instanz die Absicherungsmaßnahmen für die internen und externen Schnittstellen der Versuchszentrale verwalten. In sim^{TD} werden dazu die Hauptfunktionen 5.1 als OSGi-Bundles umgesetzt und als zentrale Hauptkomponente ein sogenannter *Security Server*. Die Funktionen werden in enger Zusammenarbeit mit dem Security Server für die Verteilung der allgemeinen und individuellen Sicherheitsparameter sorgen.

3.6.9.1 Security Server

Der Security Server befindet sich in der Versuchszentrale und übernimmt die folgenden Aufgaben:

- Erstellung von Basisidentitäten, die auf IEEE 1609.2 basieren
- Ausstellung von Pseudonymen, die auf IEEE 1609.2 basieren
- Verwaltung und Auflösung der Pseudonyme für die Versuchsauswertung
- Revokation von Pseudonymen
- Hard- und Software Validierung der Fahrzeugsysteme
- Verwaltung von Serverzertifikate, die auf X.509v3 basieren

Eine detaillierte Beschreibung der Funktionen des Security Servers ist in Deliverable D21.5 in Abschnitt 5.3.2 zu finden.

3.6.9.2 Schnittstellen der Versuchszentrale

Die Schnittstellen der Versuchszentrale, wie sie in Abbildung 3-68 zu sehen sind, werden im Folgenden erläutert. Details zu den jeweiligen Maßnahmen sind in Deliverable D21.5 in Abschnitt 5.3 zu finden.

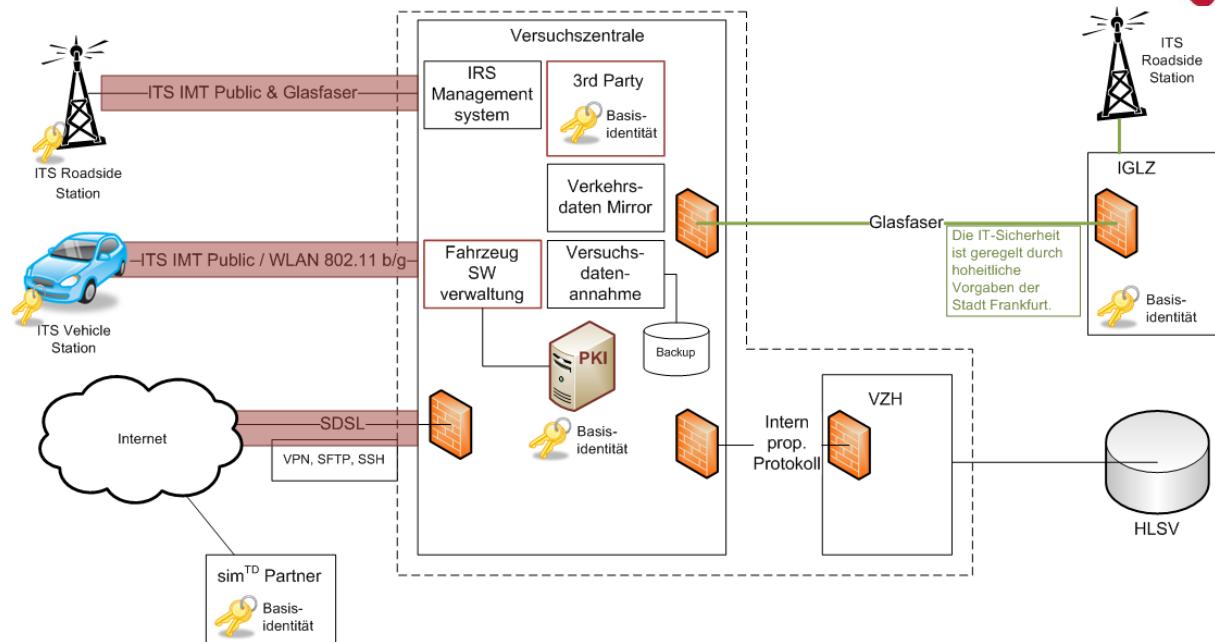


Abbildung 3-68: Absicherung der Versuchszentrale

Schnittstellen zwischen VsZ und VZH

Die Verbindung zwischen VZH und VsZ wird über eine Firewall abgesichert. Da die VZH Teil des Netzes des Landes Hessen ist, gelten die Vorgaben der HZD (Hessische Zentrale für Datenverarbeitung). Weitere Sicherungsmaßnahmen durch das IT-Sicherheitskonzept in sim^{TD} sind nicht notwendig.

Schnittstelle zwischen VsZ und IGLZ

Die Anbindung wird über eine Firewall abgesichert. Die IT-Sicherheit ist geregelt durch hoheitliche Vorgaben der Stadt Frankfurt am Main dadurch sind in diesem Konzept keine weiteren IT-Sicherheitsmaßnahmen notwendig.

Schnittstelle zwischen VsZ und Internet / sim^{TD}-Partner

Die sim^{TD}-Partner erhalten im Rahmen ihrer Aufgaben Zugang zu den Rechnern der VsZ. Die Schnittstelle ist ebenfalls durch eine Firewall abgesichert.

Die Partner benennen die Protokolle, die sie zur Verrichtung ihrer Arbeit in der VsZ (zum Zugriff auf dieselbe) benötigen. Zulässige Protokolle sind die Protokolle scp, ssh und die sicheren SNMP-Protokolle (SNMPv3). Des Weiteren kann zusätzlich für externe Dienste die Möglichkeit bestehen direkt auf die komplette VsZ per VPN zu zugreifen. Der Zugriff auf einzelne Dienste der Server per SSL ist jedoch gegenüber des generellen Zugriffs per VPN vorzuziehen.

Schnittstelle zwischen VsZ und ITS Roadside Stations

Es gibt landeseigene und städtische IRSs. Die landeseigenen IRSs unterteilen sich in zwei Kategorien: stationäre und mobile IRSs. Erstere sind über landeseigene Glasfaser-Kabel an die VsZ angebunden, dabei ist die Verbindung über eine Firewall abgesichert. Die Firewall wird so konfiguriert, dass die verantwortlichen Betreiber direkten Zugriff in die IRSs erhält.

Ferner wurde in sim^{TD} ein auf SNMPv3 basiertes Netzwerkmanagement für alle IRSs beschafft, so dass auch diese Art von Protokollen freigeschaltet wird.

Die mobilen IRSs kommen in Baustellenfahrzeugen zum Einsatz; sie sind über ITS IMT Public an die VsZ angebunden. Für den Zugang zu diesen IRSs von der VsZ gilt das gleiche wie für die stationären IRSs.

Schnittstelle zwischen VsZ und ITS Vehicle Stations

Die Absicherung zwischen der Versuchszentrale und den ITS Vehicle Stations wird im sim^{TD} über die definierte Tunnelvariante realisiert.

Schnittstelle zwischen VsZ und 3rd Party Webservices

Ein externer Zugriff auf die VsZ durch dritte ist in sim^{TD} nicht beabsichtig. Zum Simulieren eines 3rd-Party-Dienstes wird innerhalb der VsZ ein Server installiert, der Webservices anbietet. Eine gesonderte Absicherung ist in diesem Fall nicht vor zu sehen.

4 Zusammenfassung

Zusammen mit den vertiefenden Deliverables D21.3, D21.4, D21.5, D22.1 und D23.1 dokumentiert Deliverable D21.2 die konsolidierte Systemarchitektur für sim^{TD}. Mit der Funktionsspezifikation in den Deliverables D11.4 und D11.3 bilden die genannten Deliverables somit die zentralen Ergebnis für den Projektmeilenstein MS2 („Systemarchitektur erstellt“), der hiermit erreicht wurde. Der Erstellung der genannten Deliverables ging eine intensive (projektübergreifende) Zusammenarbeit in mehreren Iterationsschritten voraus, bei der die Systemkomponenten einerseits und die Anforderungen aus Teilprojekt TP1 andererseits bestmöglich aufeinander abgestimmt wurden.

Gleichzeitig wurde mit Projektmeilenstein MS2 die Grundlage für diverse anschließende Aktivitäten gelegt. Neben der Umsetzung des Gesamtsystem und der einzelnen Komponenten, die sich in den nachfolgenden Aktivitäten anschließt, bilden die Ergebnisse auch eine wichtige Grundlage für die Arbeiten in AP12 (Validierungs- & Optimierungsziele, -methoden und -metriken) sowie einen Input für die Aktivitäten in den Teilprojekten TP3 und TP4. Indirekt ist die Systemarchitektur auch für Teilprojekt TP5 interessant, da diese auch Einfluss auf diverse Rahmenbedingungen und Bewertungsaspekte hat.

Anhang 1 Die Schnittstelle zu OEM-Sonderfunktionen

Dieser Anhang beschreibt die im Abschnitt 3.2.13.3 im Überblick dargestellte Datenschnittstelle zur Einbindung von OEM-Sonderfunktionen und OEM-Devices

Die Schnittstellenkomponente sendet Output-Datensätze der relevanten sim^{TD}-Komponenten, welche für OEM-Sonderfunktionen oder OEM-Devices interessant sein können (z.B. Egoposition und Fahrzeugdaten, Umfeldtabelle, Route, ...), möglichst schnell nach Verfügbarkeit in einem für die jeweilige Komponenten spezifizierten Takt per Multicast über Ethernet als Broadcast aus, so dass sie von beliebigen vernetzten externen Komponenten zeitgleich empfangen werden können. Damit können OEM-Komponenten und Steuergeräte außerhalb der AU diese Daten mit eigenen Algorithmen auswerten und eigene Funktionen realisieren.

Die OEM-Schnittstelle kann nach Bedarf in OEM-Versuchsträgern aktiviert werden und ist in den Fahrzeugen der Basisflotte in der Regel deaktiviert. Wird sie in einem OEM-Versuchsträger aktiviert, können zum Ausgleich des zusätzlichen Ressourcenverbrauchs im OEM-Versuchsträger nicht benötigte Basisfunktionen deaktiviert werden.

Wesentliche Merkmale der Übertragung

Jeder relevanten sim^{TD}-Komponente wird zweckmäßigerweise eine eigene Multicast-Adresse zugeordnet. Damit brauchen Empfänger nur die Multicast-Adressen der Komponenten verfolgen, an deren Output sie interessiert sind.

Große Outputdatensätze (z.B. Umfeldtabelle) werden bei Bedarf entsprechend der MTU (maximum transmission unit) in mehrere Datenpakete aufgeteilt.

Publizierten Outputdaten können verloren gehen (z.B. bei Netzwerkproblemen/-kollisionen) oder nur zeitweise gesendet werden (z.B. bei Funktionsausfällen). Es liegt in der Verantwortung des Empfängers, Signalsausfälle zu entdecken und geeignet zu behandeln (Extrapolation alter Daten, Degradation oder Ausfall der Systemfunktion).

Weitere Diagnosedaten oder Diagnoseschnittstellen zu externen Komponenten werden nicht funktionsübergreifend implementiert (z.B. Abfragen von Statusinformationen der OSGi-Komponenten). Sie müssen bei Bedarf OEM-spezifisch vom OEM für die eigene OEM-Funktion implementiert werden.

Interface zur digitalen Karte

Ein Sonderfall ist die Komponente „Digitale Karte“, auf deren API externe OEM-Funktionen ebenso Zugriff benötigen wie innerhalb des OSGi-Frameworks realisierte Komponenten. Daher soll die Komponente „Digitale Karte“ so erweitert werden, dass ein Subset ihrer API als Service über ein Socketinterface sowohl für die internen OSGi-Funktionen der Vehicle Application Unit wie für die externe OEM-Funktionen gleichermaßen zu Verfügung steht.

Weitere Informationen

Die weiteren technische Einzelheiten werden nach Abschluss der Spezifikation der Outputdaten der relevanten sim^{TD}-Komponenten in einer separaten technischen Beschreibung dokumentiert und dort fortlaufend aktualisiert (z.B.: Multicast-Adressen und Datenoffsets der versendeten Output-Daten).

Anhang 2 Schnittstellenspezifikation Komponente „Navi/Karte“

Datenspezifikation

Hier werden alle notwendigen Daten spezifiziert und alle bekannten Datenflüsse beschrieben.

Spezifikation einer Position

Position	
<i>Address</i>	Adresse (siehe Tabelle 3)
<i>GeoPosition</i>	Raw Geo-Position (siehe Tabelle 4)
<i>Direction</i>	Richtung
<i>LinkID</i>	ID des Segments in der Karte - Map Matching
<i>Offset</i>	Offset der Position auf dem Kartensegment - Map Matching

Tabelle 2: Definition Position

Address	
<i>Country</i>	Land
<i>City</i>	Stadt
<i>ZIP</i>	Postleitzahl
<i>Street</i>	Straße
<i>StreetNumber</i>	Hausnummer

Tabelle 3: Definition Address

GeoPosition	
<i>Longitude</i>	Längengrad in 10^-7 degrees
<i>Latitude</i>	Breitengrad in 10^-7 degrees

Tabelle 4: Definition GeoPosition

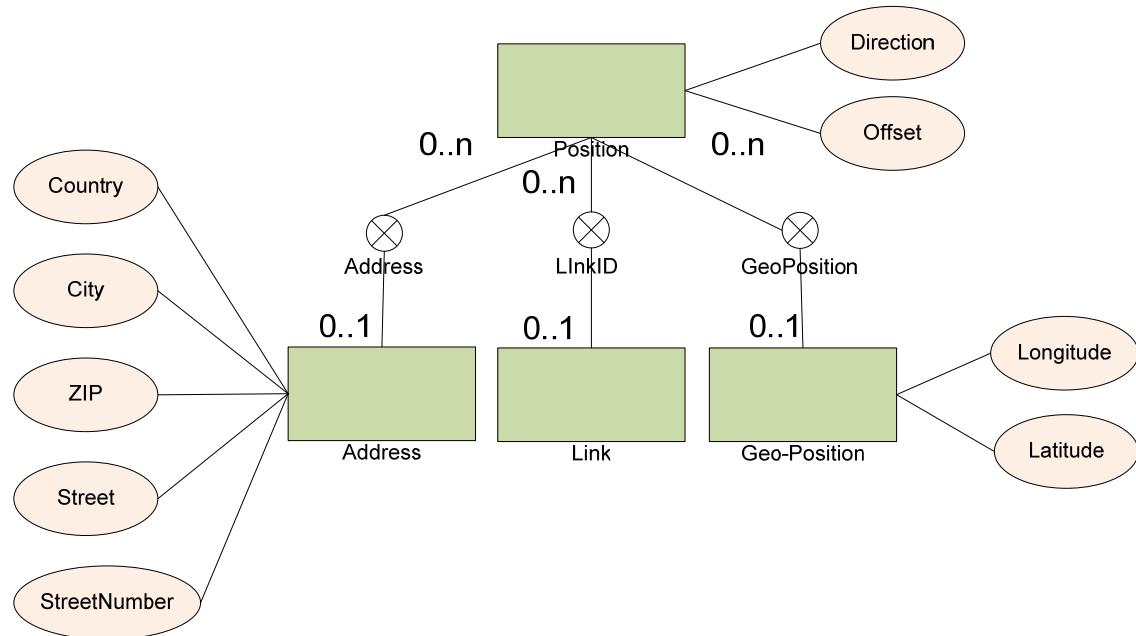


Abbildung 4-1: Übersicht Position

Spezifikation eines POIs

POI	
<i>ID</i>	Eindeutiger Bezeichner
<i>Name</i>	Name
<i>Description</i>	Kurze Beschreibung
<i>IconURL</i>	URL des POI-Piktogramms
<i>CategoryID</i>	Bezeichner der POI-Kategorie
<i>Position</i>	Position des POIs (siehe Tabelle 4)

Tabelle 5: Definition POI

POICategory	
<i>ID</i>	Eindeutiger Bezeichner
<i>Name</i>	Name
<i>Description</i>	Kurze Beschreibung
<i>LayerID</i>	Bezeichner der Kategorie-Layer

Tabelle 6: Definition POICategory

Layer	
<i>ID</i>	Eindeutiger Bezeichner
<i>Name</i>	Name
<i>Prio</i>	Priorität des Layers

Tabelle 7: Definition Layer

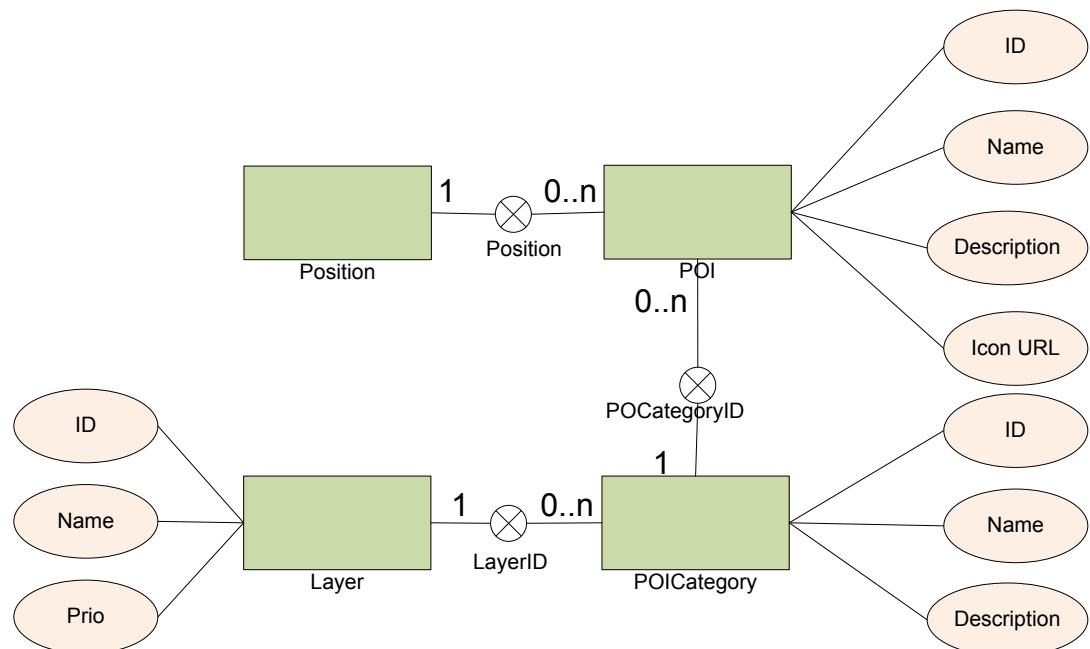


Abbildung 4-2: Übersicht POI

Spezifikation eines Karten-Segments

Link	
<i>ID</i>	Eindeutiger Bezeichner
<i>LinkAttributes</i>	Liste von Link-Attributen (siehe Tabelle 10 für die Attributen, die für sim ^{TD} relevant sind)
<i>ShapePoints</i>	Liste von ShapePoints (Geometrie)

Tabelle 8: Definition Link

ShapePoint	
<i>PI</i>	PI
<i>RAD</i>	Konvertierungsfaktor zwischen DMD (Decimal Degrees) und Radians.
<i>EARTH_RADIUS</i>	Radius in Meter
<i>GeoPosition</i>	GeoPosition des ShapePoints (siehe Tabelle 4)

Tabelle 9: Definition ShapePoint

Type	Description
<i>Length</i>	Länge in CM
<i>PrimaryHeading0</i>	Primary heading am Knoten 0 in Degrees, N=0, clockwise
<i>PrimaryHeading1</i>	Primary heading am Knoten 1 in Degrees, N=0, clockwise

Type	Description
<i>Lanes0</i>	Anzahl von Spuren am Knoten 0
<i>Lanes1</i>	Anzahl von Spuren am Knoten 1
<i>FunctionalClass</i>	Functional Class für den Link
<i>Country</i>	Land
<i>StreetName</i>	Straßennamen für den Link
<i>StreetNumber</i>	Straßennummer
<i>SpecialSpeedLimitKMH</i>	Special Speed Limit in km/h
<i>SpeedLimitRainKMH</i>	Special Rain Speed Limit in km/h
<i>SpeedLimitSnowKMH</i>	Special Snow Speed Limit in km/h
<i>SpeedLimitDayKMH</i>	Special Day Speed Limit in km/h
<i>SpeedLimitNightKMH</i>	Special Night Speed Limit in km/h
<i>SpeedLimitTimeKMH</i>	Special Time Speed Limit in km/h
<i>AdvisorySpeedLimitKMH</i>	Vorgeschlagene Geschwindigkeit in km/h
<i>ExpectedSpeedKMH</i>	Erwartete Geschwindigkeit in km/h
<i>ADASSpeedFrom</i>	ADAS Geschwindigkeit Von in km/h
<i>ADASSpeedTo</i>	ADAS Geschwindigkeit Bis in km/h
<i>ADASNumberOfLanesFrom</i>	ADAS Anzahl von Spuren Von
<i>ADASNumberOfLanesFrom</i>	ADAS Anzahl von Spuren Von
<i>isAccessRoad</i>	Flag (true/false) - Access Road
<i>isArterialRoad</i>	Flag (true/false) – Arterial Road
<i>IsHighway</i>	Flag (true/false) – Highway
<i>IsMotorway</i>	Flag (true/false) – Motorway
<i>IsOrdinaryRoad</i>	Flag (true/false) – normale Straße
<i>IsInCity</i>	Flag (true/false) – Liegt der Link in einer Stadt
<i>IsBridge</i>	Flag (true/false) - Existenz einer Brücke
<i>IsTunnel</i>	Flag (true/false) - Existenz eines Tunnels
<i>IsRamp</i>	Flag (true/false) - Existenz einer Rampe
<i>IsComplex-Intersection</i>	Flag (true/false) - Existenz einer komplexen Kreuzung

Tabelle 10: Liste von Link-Attributen

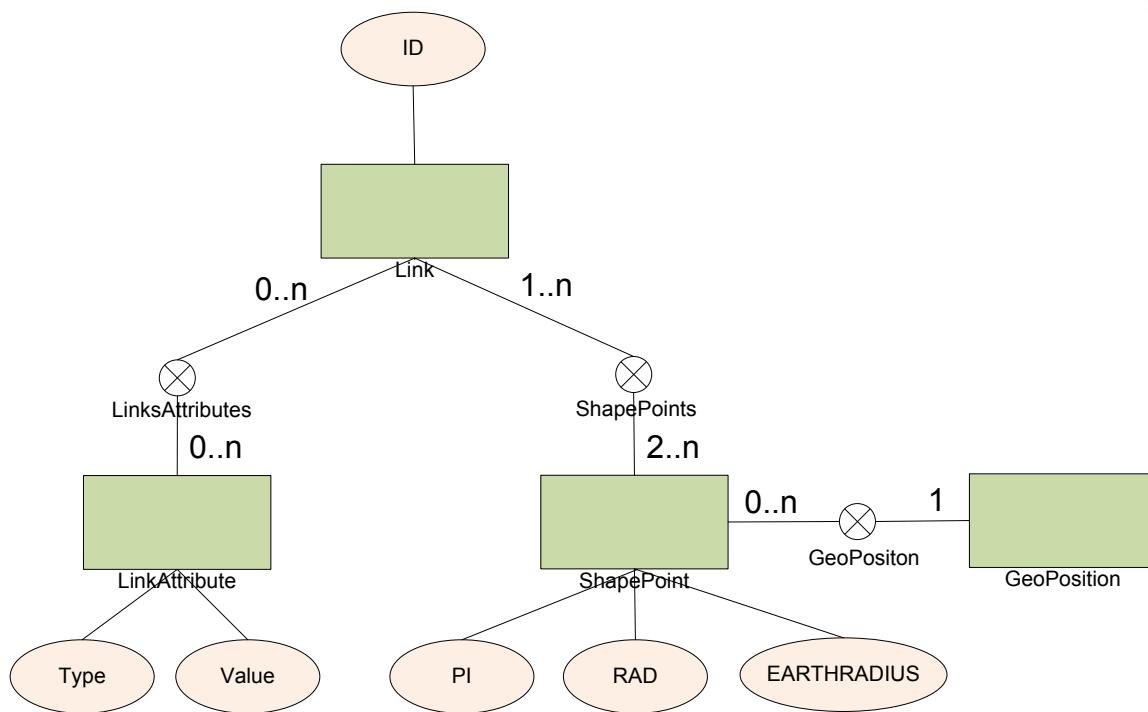


Abbildung 4-3: Übersicht Karten-Segment

Spezifikation einer Route

Route	
<i>Duration</i>	Dauer der Route in sec
<i>Length</i>	Länge der Route in cm
<i>StartTime</i>	Start Timestamp (wird von einer zentralen Komponente geliefert)
<i>Origin</i>	Start Punkt der Route (type GeoPosition)
<i>Destination</i>	Ziel der Route (type GeoPosition)
<i>RouteLegs</i>	Liste von Routen-Segmenten (Teilrouten)

Tabelle 11: Definition Route

RouteLeg	
<i>Duration</i>	Dauer des RouteLegs in sec
<i>Length</i>	Länge des RouteLegs in cm
<i>Origin</i>	Start Punkt des RouteLegs (type GeoPosition)
<i>Destination</i>	End Punkt des RouteLegs (type GeoPosition)
<i>RouteLinks</i>	Liste von Routen-Links

Tabelle 12: Definition RouteLeg

RouteLink	
<i>Duration</i>	Dauer des RouteLinks in sec
<i>Length</i>	Länge des RouteLinks in cm
<i>Origin</i>	Startpunkt des RouteLinks (type GeoPosition)
<i>Destination</i>	Endpunkt des RouteLinks (type GeoPosition)
<i>LinkID</i>	ID des relevanten Karten-Segmenten (siehe Tabelle 8)

Tabelle 13: Definition RouteLink

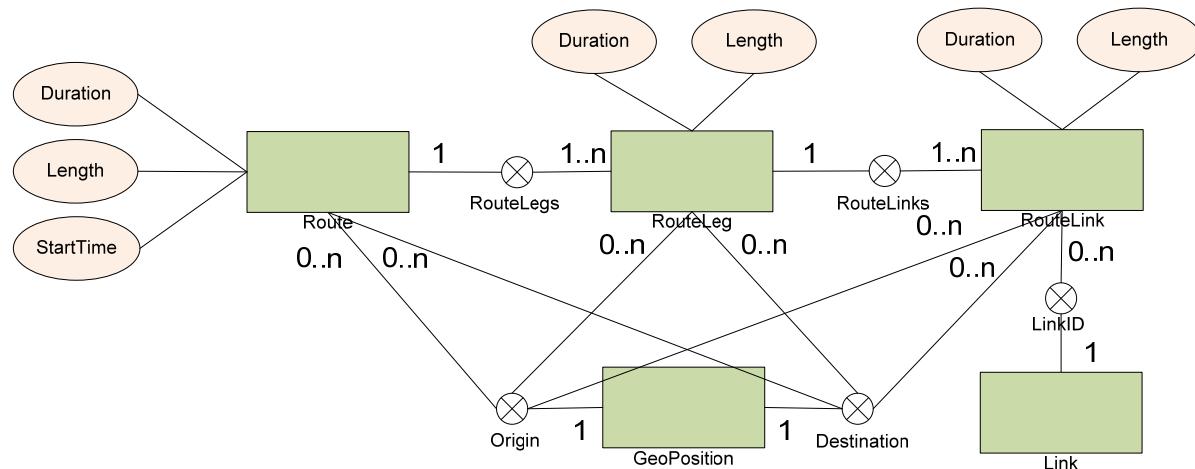


Abbildung 4-4: Übersicht Routendaten

Spezifikation einer Fahranweisung

Instruction	
<i>Type</i>	Typ: Early Instruction, Prepare Instruction und Approach Instruction
<i>Text</i>	Beschreibung
<i>Distance</i>	Abstand zum nächsten Entscheidungspunkt
<i>Direction</i>	Richtung am nächsten Entscheidungspunkt
<i>Icon</i>	Piktogramm für den Pfeil
<i>Audio</i>	Audio Datei

Tabelle 14: Definition Fahranweisung

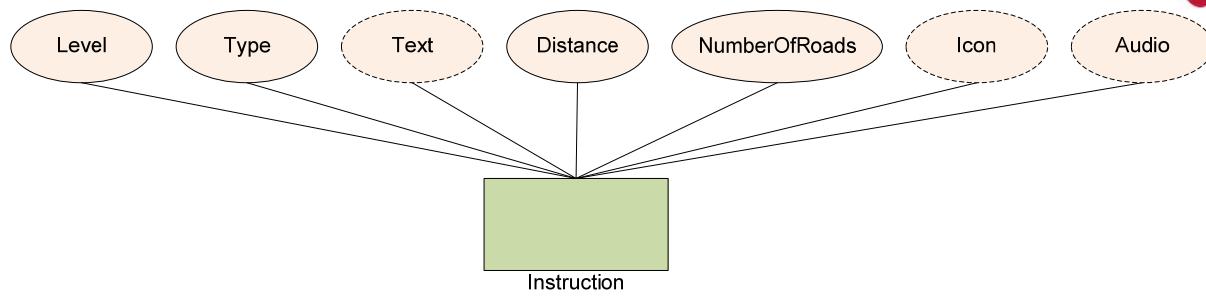


Abbildung 4-5: Übersicht Instruction

Spezifikation des ADAS-Horizonts

Horizon	
<i>Depth</i>	Länge des Horizonts in cm
<i>HorizonLinks</i>	Liste von HorizonLinks

Tabelle 15: Definition Horizon

HorizonLink	
<i>LinkID</i>	ID des relevanten Karten-Segmenten (siehe Tabelle 8)
<i>Parents</i>	Liste von Parent HorizonLinks
<i>Probabilities</i>	Liste von Wahrscheinlichkeit, dass der HorizonLink von jedem Parent gefahren wird.

Tabelle 16: Definition HorizonLink

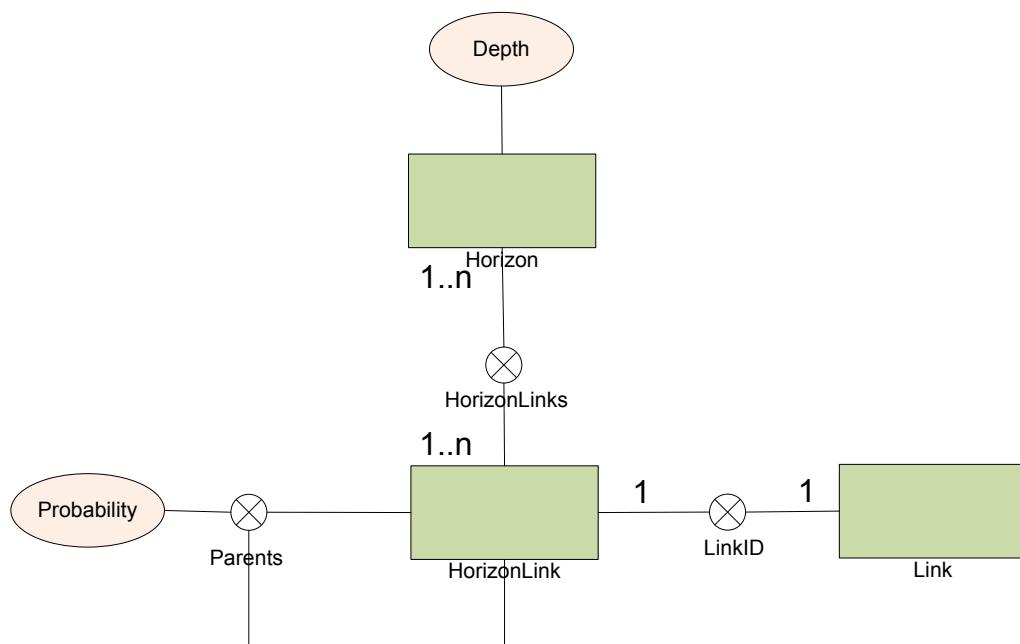


Abbildung 4-6: Übersicht ADAS Horizon

Spezifikation des Most Probable Path (MPP)

Der Most Probable Path ist als eine Liste von HorizonLinks definiert. Diese HorizonLinks haben die höchste Wahrscheinlichkeit.

Spezifikation von Daten, die die Navigation zur Laufzeit speichert

- Aktuelle Route (wird nach jeder Berechnung zur Verfügung gestellt)
 - Aktueller Startpunkt (für manuelles Setzen der Position)
 - Aktuelles Ziel
 - Aktuelle Zwischenziele
- Aktuelle Routenkriterien: schnellste/kürzeste Route, Autobahn vermeiden, Mautstraßen vermeiden (werden nach jeder Berechnung zur Verfügung gestellt)
- Routenführung aktiviert (ja/nein)
- Aktuelle Position des Fahrzeugs
- Aktuelle Anweisung
- Aktueller Horizont (EH)
- Aktuelle Karte und Layers (n letzten Karten speichern, konfigurierbar)
- POI-Liste: Alle benutzerdefinierten POIs nach Kategorien
- POI ID + Layer ID Liste: Alle POIs, die auf einem Layer liegen.
- Konfigurationsdaten
 - Initiale Dimensionen der Karte (Höhe, Breite)

- Initiale Zoomstufe (Max Zoom In, Max Zoom Out)
- Tag- oder Nachtversion der Karte bei Initialisierung der Navigation
- Initiale Länge des elektronischen Horizonts

Die Layers, die POI Kategorien und eine Menge von POIs werden permanent in einer Datenbank gespeichert und können bei Bedarf abgefragt werden.

Die Konfigurationsdateien werden in einer Konfigurationsdatei gespeichert und werden beim Initialisieren der Navigationskomponente geladen.

Alle anderen Daten sind nur während der Laufzeit gültig und werden nicht permanent gespeichert.

Spezifikation von Logging-Einträgen

Die Navigation wird folgende Ereignisse loggen:

- Aufgetretene Fehler
- Erzeugung eines POIs
- Routenberechnung
- Routenführung (Aktivierung und Deaktivierung)
- Starten und Stoppen der Navigation

Die aufgetretenen Fehler werden in einem Log-File gespeichert.

Der Rest wird zu der Logging-Komponente geschickt.

Log POI	
<i>Typ</i>	POI
<i>Zeitstempel</i>	Zeit wird von einer zentralen Komponente geliefert
<i>FzgPosition</i>	Aktuelle GeoPosition des Fahrzeugs
<i>POIType</i>	Typ des POIs
<i>POIPosition</i>	GeoPosition des POIs

Tabelle 17: Definition Logging-Eintrag bei POIs

Log Routenberechnung	
<i>Typ</i>	Routenberechnung
<i>Zeitstempel</i>	Zeit wird von einer zentralen Komponente geliefert
<i>StartPosition</i>	GeoPosition des Startpunktes
<i>EndPosition</i>	GeoPosition des Endpunktes
<i>Dauer</i>	Berechneter Dauer der Route
<i>Länge</i>	Berechnete Länge der Route

Tabelle 18: Definition Logging-Eintrag bei einer Routenberechnung

Log Routenführung Aktiv/Inaktiv	
<i>Typ</i>	Routenführung Aktiv/Inaktiv
<i>Zeitstempel</i>	Zeit wird von einer zentralen Komponente geliefert
<i>FzgPosition</i>	Aktuelle GeoPosition des Fahrzeugs

Tabelle 19: Definition Logging-Eintrag beim Aktivieren/Deaktivieren der Routenführung

Log Start/Stop Navigation	
<i>Typ</i>	Start/Stop Navigation
<i>Zeitstempel</i>	Zeit wird von einer zentralen Komponente geliefert
<i>FzgPosition</i>	Aktuelle GeoPosition des Fahrzeugs

Tabelle 20: Definition Logging-Eintrag beim Starten/Stoppen der Navigation

Bereitgestellte Schnittstellen

Die folgenden Tabellen listen die Methoden der Navigationsschnittstelle in Gruppen anhand der Funktionalität, die diese Methoden anbieten. Alle Methoden und Parametern sind kurz beschrieben.

Zieleingabe (mit Spelling-Funktionalität)

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>getCountry</i> (Spelling-Funktionalität für Ländern)	pCountry	String	Input	Erste Buchstabe oder Teil der Ländernamen.
		String[]	Output	Liste mit verfügbaren Ländernamen.
<i>getCity</i> (Spelling-Funktionalität für Städte)	pCountry	String	Input	Name des Landes in dem eine Stadt gesucht wird.
	pCity	String	Input	Erste Buchstabe oder Teil des Städtenamens.
		String[]	Output	Liste mit verfügbaren Städtenamen.
<i>setStart</i> (manuelles Setzen des Start-Punktes anhand einer GeoPosition)	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>setStart (manuelles Setzen des Start-Punktes anhand einer Adresse)</i>	<i>pCountry</i>	String	Input	Name des Landes
	<i>pCity</i>	String	Input	Name der Stadt
	<i>pZIP</i>	String	Input	Postleitzahl
	<i>pStreet</i>	String	Input	Name der Straße
	<i>pStreetNum</i>	String	Input	Straßennummer
<i>getStart (liefert den aktuelle Start-Punkt)</i>		Position	Output	Gematchte Position
<i>setDestination (setzt das Ziel anhand von Geo-Koordinaten)</i>	<i>pLongitude</i>	Long	Input	Längengrad der Geo-Position
	<i>pLatitude</i>	Long	Input	Breitengrad der Geo-Position
<i>setDestination (setzt das Ziel anhand von Adressen)</i>	<i>pCountry</i>	String	Input	Name des Landes
	<i>pCity</i>	String	Input	Name der Stadt
	<i>pZIP</i>	String	Input	Postleitzahl
	<i>pStreet</i>	String	Input	Name der Straße
	<i>pStreetNum</i>	String	Input	Straßennummer
<i>getDestination (liefert das aktuelle Ziel)</i>		Position	Output	Gematchte Position

Tabelle 21: Navigationsschnittstelle: Zieleingabe

Routenplanung, Routenberechnung, Routenführung

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
setVias <i>(setzt die Zwischenziele)</i>	pViaList	Position[]	Input	Liste von Adressen oder Geo-Positionen.
getVias <i>(liefert die Zwischenziele)</i>		Position[]	Input	Liste von Zwischenzielen (gematchete Positionen)
getNextVia <i>(liefert den nächste Zwischenziel auf der Route)</i>		Position	Output	Gematchete Position
setRouteConfig <i>(die Routenberechnung wird durch Optionen konfiguriert)</i>	pConfig	String	Input	Optionen für die Konfiguration (kürzeste/schnellste, Autobahn vermeiden, Mautstraßen vermeiden und kleinen Straßen vermeiden). z. B. „S;0;0;0“ bedeutet kürzeste Route; Autobahn vermeiden; Mautstraßen vermeiden; kleinen Straßen vermeiden
getRouteConfig <i>(liefert die Optionen für die Routenberechnung)</i>		String	Output	Optionen für die Konfiguration (kürzeste/schnellste, Autobahn vermeiden, Mautstraßen vermeiden und kleinen Straßen vermeiden)
calculateRoute <i>(berechnet eine Route)</i>	pFirst	Boolean	Input	Wenn Ja, ist eine erste Berechnung einer Route Wenn Nein, dann ist es eine neue Berechnung auf Basis einer alten Route

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>getRoute</i> (liefert die aktuelle Route)		Route	Output	Die aktuelle Route
<i>clearRoute</i> (löscht die aktuelle Route)				
<i>startGuidance</i> (Routenführung wird aktiviert)				
<i>stopGuidance</i> (Routenführung wird deaktiviert)				
<i>getCurrentInstruction</i> (liefert die aktuelle Fahranweisung)		Instruction	Output	Eine Fahranweisung
<i>getNextInstruction</i> (liefert die nächste Fahranweisung)		Instruction	Output	Eine Fahranweisung

Tabelle 22: Navigationsschnittstelle: Routenplanung, Routenberechnung, Routenführung

Positionierung

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>getCurrentPosition</i> (liefert die aktuelle Position des Fahrzeuges)		Position	Output	Gematchte Position
<i>getMapPosition</i>	pLongitude	Long	Input	Längengrad der Geo-Position

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>(Matched eine Geo-Position auf der Karte)</i>	pLatitude	Long	Input	Breitengrad der Geo-Position
		Position	Output	Gematchte Position
<i>getMapPosition</i> <i>(Matched eine Geo-Position auf der Karte anhand von NMEA Strings)</i>	pPos	String	Input	Geo-Koordinaten in NMEA-String
		Position	Output	Gematchte Position

Tabelle 23: Navigationsschnittstelle: Positionierung

Kartenschnittstelle und POIs

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>setZoomLevel</i> <i>(setzt die Zoomstufe der Karte)</i>	pLevel	Integer	Input	Zoomstufe 1 bis 5
<i>setMapCenter</i> <i>(setzt den Mittelpunkt des Kartenausschnitts über Geo-Koordinaten)</i>	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position
<i>setMapOrientation</i> <i>(setzt die Orientierung der Kartenansicht)</i>	pMapOrientation	Integer	Input	Enumeration: Nord 1, Entlang der Route 2
<i>getMaps</i> <i>(liefert die aktuelle Karte und die Layers dazu)</i>		File[]	Output	Bitmap Files
<i>setRouteCost</i> <i>(setz die Gewichte für alle Segmenten in der Karte zwischen einem Start-und einem End-Punkt, wenn der End-Punkt NULL ist dann wird nur ein Link</i>	pStartLongitude	Long	Input	Längengrad des Start-Punktes
	pStartLatitude	Long	Input	Breitengrad des Start-Punktes
	pEndLongitude	Long	Input	Längengrad des End-

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>berücksichtigt)</i>	de			Punktes
	pEndLatitude	Long	Input	Breitengrad des End-Punktes
	pCost	Integer	Input	Neues Gewicht
<i>setRouteCost (setz den Gewicht für einen Segmenten in der Karte anhand von LinkIDs)</i>	pLinkID	Long	Input	ID (eindeutiger Bezeichner) der Kartenkante
	pCost	Integer	Input	Neues Gewicht
<i>clearRouteCost (setz die Gewichte für alle Segmenten in der Karte zwischen einem Start-und einem End-Punkt zum Default zurück)</i>	pStartLongitude	Long	Input	Längengrad des Start-Punktes
	pStartLatitude	Long	Input	Breitengrad des Start-Punktes
	pEndLongitude	Long	Input	Längengrad des End-Punktes
	pEndLatitude	Long	Input	Breitengrad des End-Punktes
<i>clearRouteCost (setz den Gewicht für einen Segmenten in der Karte anhand von LinkIDs zum Default zurück)</i>	pLinkID	Long	Input	ID (eindeutiger Bezeichner) der Kartenkante
<i>addMapPolyline (setzt die Einfärbung für alle Segmenten in der Karte zwischen einem Start-und einem End-Punkt)</i>	pPositions	GeoPosition[]	Input	Geo-Positionen, die eine Linie definieren
	pColorR	Integer	Input	Neue Farbe Red Value
	pColorG	Integer	Input	Neue Farbe Green Value
	pColorB	Integer	Input	Neue Farbe Blue Value
	pWidth	Integer	Input	Neue Dichte in Pixeln
		Integer	Output	ID der Polyline

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>addMapPolyline (setzt die Einfärbung für einen Segmenten in der Karte anhand von LinkIDs)</i>	pLinkID	Long	Input	ID (eindeutiger Bezeichner) der Kartenkante
	pColorR	Integer	Input	Neue Farbe Red Value
	pColorG	Integer	Input	Neue Farbe Green Value
	pColorB	Integer	Input	Neue Farbe Blue Value
	pWidth	Integer	Input	Neue Dichte in Pixeln
		Integer	Output	ID der Polyline
<i>deleteMapPolyline (löscht die Polyline anhand von IDs)</i>	pID	Integer	Input	ID der Polyline
<i>addMapPolygon (setzt die Einfärbung für alle Segmenten in der Karte in einem Bereich)</i>	pPositions	GeoPosition[]	Input	Geo-Positionen, die einen Bereich definieren
	pColorR	Integer	Input	Neue Farbe Red Value
	pColorG	Integer	Input	Neue Farbe Green Value
	pColorB	Integer	Input	Neue Farbe Blue Value
	pWidth	Integer	Input	Neue Dichte in Pixeln
	pFill	Boolean	Input	Bereich wird gefüllt ja/nein
		Integer	Output	ID des Polygons
<i>deleteMapPolygon (löscht den Polygon anhand von IDs)</i>	pID	Integer	Input	ID des Polygons
<i>addPOI (erzeugt ein neues POI und liefert ein ID zurück)</i>	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>insertPOI</i> <i>(erstellt einen POI anhand von den angegebenen Parametern)</i>	pName	String	Input	Name des POIs
	plconURL	String	Input	URL des Piktogramms für den POI
	pCategoryID	Integer	Input	Kategorie des POIs
		Integer	Output	ID (eindeutiger Bezeichner) des POIs
<i>updatePOI</i> <i>(aktualisiert ein POI anhand von einem ID)</i>	pPoID	Integer	Input	ID (eindeutiger Bezeichner) des POIs
	pLongitude	Long	Input	Neuer Längengrad der Geo-Position des POIs
	pLatitude	Long	Input	Neuer Breitengrad der Geo-Position des POIs
	plconURL	String	Input	Neue URL des Piktogramms für den POI
	pCategoryID	Integer	Input	Neue Kategorie des POIs
<i>removePOI</i> <i>(löscht ein POI anhand von einem ID)</i>	pPoID	Integer	Input	ID (eindeutiger Bezeichner) des POIs
<i>addPOICategory</i> <i>(erzeugt eine neue POI Kategorie und liefert eine ID zurück)</i>	pName	String	Input	Name der POI-Kategorie
	pLayerId	Integer	Input	ID des Layers
		Integer	Output	ID (eindeutiger Bezeichner) der POI Kategorie
<i>removePOICategory</i> <i>(löscht eine POI Kategorie anhand von einem ID)</i>	pCategoryID	Integer	Input	ID (eindeutiger Bezeichner) der POI Kategorie

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>addLayer (erzeugt ein neues Layer und liefert ein ID zurück)</i>	pName	String	Input	Name des Layers
	pPrio	Integer	Input	Priorität des Layers
		Integer	Output	ID (eindeutiger Bezeichner) des Layers
<i>removeLayer (löscht ein Layer anhand von einem ID)</i>	pLayerID	Integer	Input	ID (eindeutiger Bezeichner) des Layers
<i>screen2world (rechnet Pixel-Koordinaten zu Geo-Koordinaten um)</i>	pLongitude	Integer	Input	Pixel-Koordinaten
	pLatitude	Integer	Input	Pixel-Koordinaten
	pMapID	Integer	Input	ID der Karte
		GeoPosition	Output	Geo-Koordinaten
<i>world2screen (rechnet Geo-Koordinaten zu Pixel-Koordinaten um)</i>	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position
	pMapID	Integer	Input	ID der Karte
		Integer[2]	Output	Pixel-Koordinaten
<i>showPOICategory (blendet benutzerdefinierte POI Kategorien in der Karte oder entlang der Route ein und aus)</i>	pShow	Boolean	Input	Wird die POI-Kategorie angezeigt: JA/NEIN
	pRoute	Boolean	Input	Wird die POI-Kategorie nur entlang der Route oder generell in der Karte angezeigt oder nicht angezeigt: JA/NEIN
	pPOICategoryID	Integer	Input	ID der POI-Kategorie
<i>showAllPOIs (blendet alle benutzerdefinierte POI</i>				

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>Kategorien in der Karte ein)</i>				
<i>findPOI (findet POI(s) in einer Geo-Position)</i>	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position
		Integer[]	Output	ID(s) von POIs

Tabelle 24: Navigationsschnittstelle: Kartenschnittstelle und POIs

Elektronischer Horizont und Kartenzugriff

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>getLinkAttributes (liefert alle Attribute einer Kartenkante anhand von einem ID)</i>	pLinkID	Integer	Input	ID (einheitlicher Bezeichner) der Kartenkante
		LinkAttribute[]	Output	Liste von Attributen einer Kartenkante
<i>getLinkShapePoints (liefert die Geometrie einer Kartenkante anhand von einem ID)</i>	pLinkID	Integer	Input	ID (einheitlicher Bezeichner) der Kartenkante
		ShapePoint[]	Output	Liste von ShapePoints (Geometrie) einer Kartenkante
<i>getLinkIDsFromTo (liefert alle LinkIDs in einem Bereich anhand von Positionen)</i>	pStartLongitude	Long	Input	Längengrad des Start-Punktes
	pStartLatitude	Long	Input	Breitengrad des Start-Punktes
	pEndLongitude	Long	Input	Längengrad des End-Punktes
	pEndLatitude	Long	Input	Breitengrad des End-Punktes
		Long[]	Output	Liste von Links-IDs (keine Geometrie, keine

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
				Attribute)
<i>getHorizon</i> (liefert den Horizont in den nächsten n Metern, wo n im einer Konfigurationsdatei definiert ist)		Horizon	Output	Liste von Links (Attribute und Geometrie)
<i>getMPP</i> (liefert das Most Probable Path in den nächsten n Metern, wo n im einer Konfigurationsdatei definiert ist)		HorizonLin k[]	Output	Liste von Links (Attribute und Geometrie)
<i>isInRoute</i> (überprüft ob eine Adresse auf der Route liegt)	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position
		Boolean	Output	Ja oder Nein
<i>getDistanceInRoute</i> (berechnet den realen Abstand bis zu einem Punkt auf der Route)	pLongitude	Long	Input	Längengrad der Geo-Position
<i>isInMPP</i> (überprüft ob eine Adresse auf dem Most Probable Path liegt)	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-Position
		Boolean	Output	Ja oder Nein
<i>getDistanceInMPP</i> (berechnet den realen Abstand bis zu einem Punkt auf dem Most	pLongitude	Long	Input	Längengrad der Geo-Position
	pLatitude	Long	Input	Breitengrad der Geo-

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>Probable Path)</i>				Position
		Integer	Output	Abstand in Meter

Tabelle 25: Navigationsschnittstelle: Elektronischer Horizont und Kartenzugriff

Versuchsplanung

Operationsname	Parameter	Datentyp	In/Out	Beschreibung
<i>saveRoute</i> <i>(Speicherung einer geplanten Route)</i>	pPositions	Position[]	Input	Liste von Positionen (Start, Zwischenziele, Ziel)
	pRouteConfig	String	Input	Optionen für die Konfiguration der Route (kürzeste/schnellste, Autobahn vermeiden, Mautstraßen vermeiden und kleinen Straßen vermeiden)
<i>loadRoute</i> <i>(Ladung einer vordefinierten Route)</i>	pRoute	String	Input	Name der Datei einer vordefinierten Route

Tabelle 26: Navigationsschnittstelle: Versuchsplanung

Events und Listeners

Folgende Events und Listeners sind in der Schnittstelle definiert:

Event Name	Listener	Beschreibung
<i>DestinationComplete</i>	<i>DestinationCompleteListener</i>	Zieleingabe ist abgeschlossen.
<i>RouteComplete</i>	<i>RouteCompleteListener</i>	Routen-(neu)Berechnung ist abgeschlossen.
<i>NextInstruction</i>	<i>NextInstructionListener</i>	Die nächste Fahranweisung ist generiert worden.
<i>DestinationReached</i>	<i>DestinationReachedListener</i>	Das Ziel ist erreicht worden.
<i>CurrentPosition</i>	<i>CurrentPositionListener</i>	Die aktuelle Position ist aktualisiert worden.

Event Name	Listener	Beschreibung
<i>MapChanged</i>	<i>MapChangedEventArgs</i>	Die Kartenansicht hat sich verändert.
<i>HorizonUpdated</i>	<i>HorizonUpdatedListener</i>	Der Horizont ist aktualisiert worden.

Tabelle 27 Navigationsschnittstelle: Events und Listener

Anhang 3 Abkürzungen

3G	Third Generation Mobile
3GPP	3rd Generation Partnership Project
ABS	Anti-blockier System
ACC	Adaptive Cruise Control
ACC	Application Communication Channel
ACCC	Application Communication Channel Client
ACCS	Application Communication Channel Server
AD	Autobahndreieck
ADAS	Advanced Driver Assistant Systems
AHS	Advanced Cruise-Assist Highway System
AK	Autobahnkreuz
AMIC	Administration and Monitoring Interface Client
AMIS	Administration and Monitoring Interface Server
AP	Arbeitspaket
APA	Ampel-Phasen-Assistent (Anwendungsfall in sim ^{TD})
API	Application Programming Interface
AS	Anschlussstelle
ASV	Advanced Safety Systems mounted on vehicles
AU	Application Unit
AUSA	Autobahn-Selbstwähl-Anlage
BMBF	Bundesministerium für Bildung und Forschung
BMVBS	Bundesministerium für Verkehr, Bau und Stadtentwicklung
BMWI	Bundesministerium für Wirtschaft und Technologie

C2C	Fahrzeug-zu-Fahrzeug Kommunikation (Car-to-Car)
C2C-CC	Car-2-Car-Communication Consortium
C2I	Bidirektionale Fahrzeug-zu-Infrastruktur Kommunikation(Car-2-Infrastructure)
C2X	Fahrzeug-zu-Fahrzeug (C2C) und Fahrzeug-zu-Infrastruktur (C2I) Kommunikation (Car-2-X)
CAN	Controller Area Network
CAM	Cooperative Awareness Message
CCU	Communication & Control Unit
CEPT	Europäische Konferenz der Verwaltung für Post und Telekommunikation (Conférence Européenne des Administrations des Postes et des Télécommunications)
CM	Configuration Management
CMC	Configuration Management Client
CMS	Configuration Management Server
COOPERS	Cooperative Systems for Intelligent Road Safety
CoMa	Communication Manager
CoP	Code of Practice
CSMA	Carrier Sense Multiple Access
CVIS	Cooperative Vehicle Infrastructure System
DAB	Digital Audio Broadcasting
DATEX II	Data Exchange Format
DB	Database
DBTS	Database Transaction Server
DBS	Database Server
DEN	Decentralized Environmental Notification
dIRA	Dynamische Informationstafel für Reisezeitanzeigen
DMB	Digital Multimedia Broadcasting
DoS	Denial of Service
DSRC	Dedicated Short Range Communication

DWD	Deutscher Wetter Dienst
dWiSta	Dynamischer Wegweiser mit integrierter Stauanzeige
EAG	Error Alert Gateway (SMS/Mail)
EC	European Commission
ECC	Electronic Communications Committee
ECU	Electronic Control Unit, Steuergerät
EDGE	Enhanced Data Rates for GSM Evolution
EH	Electronic Horizon
EOC	Embedded Operation Channel
EOCC	Embedded Operation Channel Client
EOCS	Embedded Operation Channel Server
ESC	Electronic Stability Control
ESoP	European Statement of Principles
ESP	Elektronisches Stabilitätsprogramm
ETC	Electronic Toll Collection
ETSI	Europäische Institut für Telekommunikationsnormen
EWS	Empfehlungen für die Wirtschaftlichkeitsuntersuchungen an Straßen
FCD	Floating Car Data
FHWA	Federal Highway Administration
FM	Fault Management
FMC	Fault Management Client
FMS	Fault Management Server
FOC	Fiber Optical Cable
FOT	Field Operational Tests
FTPS	File Transfer Protocol Secure
GERAN	GSM EDGE Radio Access Network
GGSN	Gateway GPRS Support Node
GPRS	General Packet Radio Service
GPS	Global Positioning System

GSM	Globales System für mobile Kommunikation (Global System for Mobile Communications)
GST	Global System for Telematics
GUI	Graphical User Interface
HLSV	Hessisches Landesamt für Straßen- und Verkehrswesen
HMI	Mensch-Maschine-Schnittstelle (Human Machine Interface)
HSDPA	High Speed Downlink Packet Access
HSPA	High Speed Packet Access
HSUPA	High Speed Uplink Packet Access
HTTPS	Hypertext Transfer Protocol Secure
ICS	ITS Central Station
ICT	Information and Communications Technology
IEEE	Institute of Electrical and Electronics Engineers
IGLZ	Integrierte Gesamtverkehrsleitzentrale der Stadt Frankfurt am Main (Verkehrsmanagementsystem)
IH	IRSMC Hypervisor
IKR	IGLZ Kommunikationsrechner der Stadt Frankfurt am Main
IMT	International Mobile Telecommunications
IPv6	Internet Protocol Version 6 (Internet Protocol Version 6)
IRS	ITS Roadside Station
IRSMC	IRS-Management-Center
IRSMS	IRS-Management-Server
IS	ITS Station
ISM	Industrial, Scientific, and Medical Band
ITS	Intelligent Transportation Systems
ITU	International Telecommunication Union
IV	Individualverkehr (siehe auch IV-Fahrzeug)
IVS	ITS Vehicle Station
KMU	Kleine und mittlere Unternehmen

KNA	Kosten-Nutzen-Analysen
KQA	Kreuzungs-/ Querverkehrs-Assistent (Anwendungsfall in sim ^{TD})
LIDAR	Light Detection and Ranging
LLCF	Low Level CAN Framework (siehe auch SocketCAN)
LSA	Lichtsignalanlage
LWL	Lichtwellenleiter (Glasfaserkabel)
MAC	Media Access Control
MARZ	Merkblatt für die Ausstattung von Verkehrsrechnerzentralen und Unterzentralen (VkB1. 2000 S 179)
MLIT	Ministry of Land, Infrastructure and Transport
MPP	Most Probable Path
MTU	Maximum Transmission Unit
NHTSA	National Highway Traffic Safety Administration
NMC	Network Management Client
NMEA	National Marine Electronics Association
NMS	Network Management Server
NoW	Network on Wheels
OCIT	Offene Schnittstellen für die Straßenverkehrstechnik (Open Communication Interface for Road Traffic Control Systems)
OEM	Originalgerätehersteller (Original Equipment Manufacturer)
OMA-DM	Open Mobile Alliance Device Management
OSI	Open System Interconnection (Reference Model)
OSGi	Open Services Gateway initiative
OTS	Open Traffic Systems
ÖV	Öffentlicher Verkehr (siehe auch ÖV-Fahrzeug)
PCM	Puls–coderite Modulation (Puls-Code-Modulation)
PDP	Packet Data Protocol

PDA	Persönlicher digitaler Assistent (Personal Digital Assistant)
PER	Pecked Encoding Rules
PHY	Physical Layer oft he OSI model
PKI	Public Key Infrastruktur
PND	Personal Navigation Device
POI	Point Of Interest
QoS	Quality of Service
RAU	Roadside Application Unit
RDS	Radio Data System
RSCOM	Radio Spectrum Committee
RSU	Straßenseitige Kommunikationseinheit (Road Side Unit)
SAE	Society of Automotive Engineers
SBA	Streckenbeeinflussungsanlage
SDR	Software Defined Radio
SDSL	Symmetric Digital Subscriber Line
SEVECOM	Secure Vehicle Communication
SLAP	Serial Line ASCII Protocoll
SMS	Short Message Service
SRD/MG	Short Range Devices/Maintenance Group
SRDoc	System Reference Document
SSL	Secure Sockets Layer
SWIS	Strassenzustand- und Strassenwetter-Informations-System
TCS	Antriebsschlupfregelung (traction control system)
TCP	Transmission Control Protocol
TG	Technical Group
TLS	Technische Lieferbedingungen für Streckenstationen
TMC	Traffic Message Channel
TP	Teilprojekt

TPEG	Transport Protocol Experts Group
TPM	Trusted Platform Modul
TTC	Time to Collision
UDP	User Datagram Protocol
UMTS	Universelles mobiles Telekommunikationssystem (Universal Mobile Telecommunications System)
UML	Unified Modeling Language
US DOT	US Department of Transportation
UTRAN	UMTS Terrestrial Radio Access Network
VAPI	Vehicle API
VANET	Vehicular Ad hoc Networks
VAU	Vehicle Application Unit
VBA	Verkehrsbeeinflussungsanlage
VDA	Verband der Automobilindustrie e.V.
VGF	Verkehrsgesellschaft Frankfurt
VICS	Vehicle Infrastructure Communication System
VII	Vehicle Infrastructure Integration
VL	Verkehrslage
VPN	Virtual Private Network
VMZ	
VsZ	sim ^{TD} Versuchszentrale
VZH	Verkehrszentrale Hessen
WG FM	Working Group Frequency Management
WGS 84	World Geodetic System (dating from 1984)
WG SE	Working Group Spectrum Engineering
WLAN	Kabelloses lokales Netzwerk (Wireless Local Area Network)
WVZ	Wechselverkehrszeichen
WWW	Wechselwegweiser

XML

Auszeichnungssprache (Extensible Markup Language)