Institut für Parallele und Verteilte Systeme

Universität Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Masterarbeit

Verteilte Datenpipelines in Software-Defined-Car-Anwendungen

Martin Kenzler

Studiengang:	Softwaretechnik
Prüfer/in:	Prof. Dr.-Ing. Bernhard Mitschang
Betreuer/in:	Dr. rer. nat. Pascal Hirmer
Beginn am:	3. Mai 2021
Beendet am:	3. November 2021

Kurzfassung

Mit einer stetig fortschreitenden Technisierung in der Automobilindustrie erhält das Konzept des autonomen Fahrens immer mehr auf Aufmerksamkeit. Dabei handelt es sich um Fahrzeuge, welche die Fähigkeit besitzen, eigenständige Entscheidungen zu treffen und umzusetzen. Dies ermöglicht einen effizienteren und komfortableren Transport. Als Wissensgrundlage für die Entscheidungsfindung dienen Sensordaten der Umgebung. Eigene Sensoren, wozu z.B. Außenkameras gehören, reichen aufgrund ihrer eingeschränkten Wahrnehmung nicht aus. Mit dem Einsatz von sogenannten *Connected Cars* kann die Datensammlung über die Sensoren eines einzelnen Fahrzeuges hinaus erweitert werden. Bei diesen handelt es sich um Fahrzeuge, die in der Lage sind, sich untereinander und mit dem Internet zu verbinden. Auf diese Weise können neben Sensordaten auch die Fahrabsichten kommuniziert werden. Somit gibt es zur Steuerung und Überwachung eines Fahrzeuges eine große Menge interner und externer Datenquellen. Zur Gewinnung von Wissen werden viele Verarbeitungsschritte, z.B. zur Reinigung der Daten, benötigt. Daraus resultieren äußerst komplexe Datenflüsse. In dieser Arbeit werden Konzepte entwickelt, um eine Modellierung dieser Datenflüsse durch Datenpipelines zu ermöglichen. Dabei werden Anforderungen, welche für Connected Cars und Datenpipelines gelten, herausgestellt und berücksichtigt. Dazu zählen z.B. die Eigenschaften der verteilten Rechenknoten von Connected Cars. Zusätzlich werden die entwickelten Konzepte in einem Modellierungswerkzeug implementiert. Dieses ermöglicht ein leichtes Anlegen von Modellen unter Berücksichtigung der Anforderungen. Die Modelle können schließlich für Deployments in Connected Cars eingesetzt werden. Wie so ein Deployment aussehen kann, wird anhand einer Demonstrations-Umgebung simuliert.

Inhaltsverzeichnis

Abbildungsverzeichnis

Verzeichnis der Listings

Abkürzungsverzeichnis

3GPP 3rd Generation Partnership Project. 19

BSM Basic Safety Message. 31

CMOS Complementary Metal-Oxide-Semiconductor. 32

C-V2X Cellular Vehicle-to-Everything. 19

CAN Controller Area Network. 17

D&D Drag and Drop. 52

DDoS Distributed Denial of Service. 31

DoS Denial of Service. 31

DSRC Dedicated Short Range Communication. 18

ECU Electronic Control Unit. 17

EU Europäische Union. 25

GPS Global Positioning System. 13

HTTP Hypertext Transfer Protocol. 59

IoT Internet of Things. 67

ITS Intelligent Transportation System. 15

JSON JavaScript Object Notation. 63

LiDAR Light Detection and Ranging. 16

MAC Media Access Control. 32

MANET Mobile Ad hoc Network. 17

MBP Multi-purpose Binding and Provisioning Platform. 67

MQTT Message Queuing Telemetry Transport. 24

OBU On-Board Unit. 15

OSI Open Systems Interconnection. 18

QoS Quality of Service. 9

Radar Radio Detection and Ranging. 13

RSU Roadside Unit. 15

RTP Real-Time Transport Protocol. 42

SAE Society of Automotive Engineers. 16

SIM Subscriber Identity Module. 26

SOS System of Systems. 24

V2I Vehicle-to-Infrastructure. 19

V2S Vehicle-to-Sensor. 17

V2V Vehicle-to-Vehicle. 17

VANET Vehicular Ad hoc Network. 17

VM Virtual Machine. 67

VRU Vulnerable Road User. 15

WAN Wide Area Network. 20

WAVE Wireless Access in Vehicular Environments. 18

1. Einleitung

Autonomes Fahren ist die Vision, in der Fahrzeuge eigenständig, d.h. ohne menschliches Beisteuern, fahren können [PKÅ+20]. Zur Realisierung müssen Fahrzeuge in der Lage sein, eigenständige Entscheidungen auf Basis gesammelter Informationen über ihre Umgebung zu treffen. Mit Hilfe einer Vielzahl an Sensoren, wie Kameras, Global Positioning System (GPS) und Radio Detection and Ranging (Radar), können Fahrzeuge bereits eigene Informationen sammeln. Durch fortgeschrittene Technisierung der Fahrzeuge kommt eine weitere Möglichkeit zur Informationsbeschaffung hinzu. Naheliegende Fahrzeuge können gesammelte Daten untereinander austauschen, um ein noch präziseres Abbild der Umgebung zu erzeugen. Eine Verbindung zum Internet stellt eine weitere Quelle dar, um sich über die Umgebung zu informieren. Aufgrund dieser Vernetzung werden diese als *Connected Cars* bezeichnet [CM16].

Die große Menge interner und externer Daten muss verarbeitet und interpretiert werden, um für die Entscheidungsfindung zu nützen. Somit sind eine Reihe von Verarbeitungsschritten nötig. Der entstehende Datenverkehr ist äußerst komplex. Dazu kommt, dass ein Connected Car nur eine begrenzte Kapazität an Ressourcen besitzt. Auf mehreren Rechenknoten, mit unterschiedlichen Eigenschaften, können Anwendungen zur Verarbeitung bereitgestellt werden. Dabei muss die genaue Aufteilung der Anwendungen auf die Knoten sorgfältig gewählt sein. Jede Anwendung hat individuelle Ansprüche an Ressourcen. Zwar bietet die Cloud große Mengen an Ressourcen, jedoch kann es Einschränkungen geben, welche dessen Nutzung verbieten. Zum Beispiel dürfen, zum Schutz der Privatsphäre, keine sensiblen Daten in die Cloud hochgeladen werden. Stattdessen muss die Verarbeitung lokal stattfinden. Dazu kommt, dass jeder Anwendungsfall individuelle Anforderungen an die Verarbeitung hat. Für zeitkritische Anwendungsfälle, wie z.B. dem Notbremsassistenten, muss eine geringe Latenz garantiert sein, wodurch die Verarbeitung in der Cloud nicht in Frage kommt.

Eine Umsetzung des Datenflusses ist aufgrund der vielfältigen Verarbeitungsschritte und deren Anforderungen komplex. Datenpipelines sind in der Lage diesen Datenfluss abzubilden [RSGJ13]. Ziel dieser Arbeit ist die Erstellung und prototypische Umsetzung eines Konzeptes zur Modellierung von verteilten Datenpipelines in Connected Cars. Dabei wird sich auf die Modellierung von Datenquellen und Verarbeitungsschritten fokussiert. Eine separate Modellierung von Datensenken liegt außerhalb des Rahmens dieser Arbeit. Als Voraussetzung des Konzeptes werden Anforderungen definiert. Diese behandeln Aspekte bezüglich Connected Cars und Datenpipelines. Das Konzept beschreibt, wie die Modellierung der Datenpipelines, unter Berücksichtigung der Anforderungen, aussehen kann. Zu dessen Demonstration wird ein Prototyp in Form eines Modellierungswerkzeuges entwickelt. Dieses beinhaltet einen graphischen Editor, welcher die Datenpipeline veranschaulicht. Mit dem Modellierungswerkzeug können Modelle entwickelt werden, welche zum Deployment in Connected Cars dienen. Zusätzlich wird auf einer Demonstrations-Umgebung ein potenzielles Deployment beispielhaft vorgeführt.

Gliederung

In Kapitel 2 werden die notwendigen **Grundlagen** dieser Arbeit erklärt. Dazu gehören der Aufbau und die Kommunikationswege von Connected Cars. Zusätzlich wird auf Datenpipelines eingegangen.

Kapitel 3 gibt einen Überblick über **Verwandte Arbeiten**.

Die **Anforderungen** für den Hauptteil dieser Arbeit sind in Kapitel 4 aufgeführt. Diese sind in die sechs Kategorien Privatsphäre, Netzwerk, Sicherheit, Komplexität, Interoperabilität und Datenverarbeitung unterteilt.

Kapitel 5 behandelt die **Konzeptionierung**. Es wird zuerst erläutert, wie Datenpipelines für Connected Cars modelliert werden können. Daraufhin wird speziell auf verschiedene Konzepte eingegangen, welche helfen die zuvor aufgestellten Anforderungen zu bewältigen. Zusätzlich wird beschrieben, wie diese in die Modellierung integriert werden können. Das Kapitel schließt mit einer Beschreibung, wie das Deployment und die Wartung erzeugter Modelle für Connected Cars aussehen kann.

Die **Implementierung** wird in Kapitel 6 vorgestellt. Anhand des Prototypen eines Modellierungswerkzeuges wird gezeigt, wie das Konzept umgesetzt werden kann. Dabei wird auf den technischen Aufbau und detailliert auf die Komponenten des Werkzeuges eingegangen. Außerdem wird das Datenmodell dargestellt und ein umgesetztes Deployment in einer Demonstrations-Umgebung erläutert.

Zusammenfassung und Ausblick sind abschließend in Kapitel 7 beschrieben.

2. Grundlagen

Im Folgenden Kapitel werden für das Verständnis des Hauptteils dieser Arbeit wichtige Grundlagen erläutert. Dabei wird zuerst in Abschnitt 2.1 auf Connected Cars bezüglich ihres Aufbaus und der Kommunikationswege eingegangen. Daraufhin werden in Abschnitt 2.2 der Aufbau von Datenpipelines und deren Eigenschaften erklärt.

2.1. Connected Cars

Die Zukunft der Transportation liegt im Intelligent Transportation System (ITS). Ein ITS ist ein System, das Autonomität der Fahrzeuge ermöglicht. Daten, die von Sensoren an den Fahrzeugen und der Umgebung gesammelt werden, ermöglichen eine digitale Sicht auf den Zustand der Umgebung. Auf Grundlage dieser Daten können künstliche Intelligenzen Entscheidungen treffen, um einen optimalen Verkehrsfluss zu bestimmen. Ein ITS umfasst folgende Bestandteile, welche alle miteinander kommunizieren können [SLY+16]:

- On-Board Units (OBUs)
- Roadside Units (RSUs)
- Vulnerable Road Users (VRUs)
- ITS-Server

OBUs sind in Fahrzeugen eingebaute Komponenten. Neben Steuerfunktionen ermöglichen diese die Kommunikation mit den anderen Bestandteilen des ITS. Aufgrund dieser Fähigkeit werden diese Art Fahrzeuge auch als Connected Cars bezeichnet. Bei RSUs handelt es sich um in die Infrastruktur eingebettete Komponenten. Am meisten verbreitet sind statische RSUs, wie z.B. Ampeln, Straßenschilder, Gebäude und Parkplätze [KPD+09; PKÅ+20]. Doch auch an mobilen RSUs wird bereits geforscht, um eine größere und effizientere Netzabdeckung sicherzustellen [EAM18]. Aufgabe der RSUs ist es, nahegelegene Fahrzeuge mit Informationen über den Verkehr zu versorgen. Häufig sind diese mit Sensoren ausgestattet und werten die gesammelten Daten aus. Bei VRUs handelt es sich um weitere Verkehrsteilnehmer, wie Fußgänger und Fahrradfahrer. Diese unterscheiden sich grundlegend von Connected Cars durch fehlende Sicherheitsmaßnahmen, Autonomität und Verbundenheit zum ITS. Zwar können VRUs über mobile Geräte wie Smartphones an das ITS angebunden werden, jedoch muss davon ausgegangen werden, dass dies nicht immer der Fall ist. Daher müssen sie zusätzlich über Sensoren an den Fahrzeugen wahrgenommen werden, um die Sicherheit zu gewährleisten [PKÅ+20]. ITS-Server sind zentralisierte Einheiten, die auf Basis der ausgetauschten Daten den Verkehr kontrollieren und Informationen bereitstellen können [SLY+16]. Zu den Anwendungsgebieten des ITS zählen unter anderem automatische Mauterhebung, Kreuzungsmanagement, Unterstützung bei der Parkplatzsuche und Gefahrenwarnung [GZC18].

Connected Cars haben eine Vielzahl verschiedener Sensoren verbaut, um ihren Zustand und ihre Umgebung wahrzunehmen und relevante Daten mit anderen Fahrzeugen zu teilen. Für die Messung des eigenen Zustands sind bereits in klassischen, unvernetzten Fahrzeugen Sensoren, wie z.B. GPS, Einparkkameras und Füllstandsmesser für Öl und Treibstoff, verbaut. Um den Zustand der Umgebung wahrnehmen zu können sind Connected Cars mit fortgeschrittenen Sensoren wie Radar, Ultraschallsensoren, Light Detection and Ranging (LiDAR) [DNK+18] und Außenkameras ausgestattet [GZC18]. Insgesamt besitzt ein Connected Car bis zu 100 Sensoren [IV; ZNM+16]. Die gesammelten Daten müssen verarbeitet werden, um daraus einen Nutzen zu ziehen. Jedoch ist es nicht ratsam, die Fahrer mit der Gesamtmenge an Informationen zu versorgen. Werden Fahrer zu vielen Informationen ausgesetzt, kann es zu einer Reizüberflutung kommen. Diese kann negative Auswirkungen auf die Fahrer und somit die Fahrsicherheit haben [WL07]. Stattdessen sollten diese Daten genutzt werden, um autonome Aktionen des Fahrzeuges zu ermöglichen, wie z.B. automatische Spurenhaltung und Bremsvorgänge.

In Norm SAE J3016 (Juni 2018) [com18] der Society of Automotive Engineers (SAE) sind die folgenden sechs Level des autonomen Fahrens spezifiziert:

Level 0 Keine Automatisierung

Level 1 Fahrassistenzsysteme

Level 2 Teilweise Automatisierung

Level 3 Bedingte Automatisierung

Level 4 Hochautomatisierung

Level 5 Vollständige Automatisierung

Das Ermöglichen höherer Level der Automatisierung bringt eine Reihe an Vorteilen mit sich. Big-Data-Analyseverfahren können, auf Basis der Daten, die ein Fahrzeug während der Fahrt sammelt, Verkehrsunfälle vorhersagen [ZYW+18]. Mit umfassender Automatisierung können diese Informationen genutzt werden, um sofortige Gegenmaßnahmen einzuleiten. Häufige Unfallursachen, wie z.B. Ablenkungen während der Fahrt, zu schnelles oder betrunkenes Fahren und langsame Reaktionszeiten, sind beim autonomen Fahren nicht möglich. Deshalb kann laut Yaqoob et al. [YKK+19] die Anzahl der, durch menschliches Versagen verursachten, Verkehrsunfälle durch autonomes Fahren bis zu 90% reduziert werden. Die Effizienz des Fahrens kann ebenfalls gesteigert werden. Diesbezüglich zeigen die Autoren auf, dass autonomes Fahren die Fahrtdauer bis zu einer Stunde pro Tag reduziert. Gonder et al. [GES12] zufolge wird zusätzlich ermöglicht, über 30% Treibstoff einzusparen.

Im Folgenden wird zuerst in Abschnitt 2.1.1 auf den technischen Aufbau eines Connected Cars eingegangen. Danach werden die Kommunikationswege erläutert, über die Connected Cars sich austauschen können. Dabei wird in Abschnitt 2.1.2 die Kommunikation zwischen den Fahrzeugen behandelt. Darauf folgt in Abschnitt 2.1.3 die Kommunikation zwischen Fahrzeug und Infrastruktur. Abschließend wird in Abschnitt 2.1.4 der Einsatz von moderner Mobilfunktechnologie in Connected Cars erläutert.

2.1.1. Fahrzeuginternes Netzwerk

Im Inneren der Fahrzeuge müssen technische Komponenten miteinander kommunizieren. Für diese Aufgabe haben sich das Controller Area Network (CAN) [Int15] und FlexRay [Fle05] durchgesetzt, um den Anforderungen an Performanz und Zuverlässigkeit nachzukommen [HKGF09]. Diese Bussysteme werden genutzt, um Sensoren und Electronic Control Units (ECUs) miteinander zu verbinden. Auf dem CAN-Bus definiert der Standard SAE J1939 [SAE] die genaue Kommunikation speziell für Fahrzeuge. Das CAN ist günstig, robust gegen Interferenz und bietet eine Datenrate bis zu 1 MBps [HKGF09]. FlexRay wird mit dem Ziel entwickelt den wachsenden Ansprüchen, die das CAN nicht mehr erfüllen kann, gerecht zu werden. Es ist ein in ISO 17458-1 bis 17458-5 festgehaltener Standard, der eine Datenrate von 10 MBps ermöglicht [SJ08].

Die Datenverarbeitung und Steuerung der Connected Cars geschieht durch die ECUs. Bei diesen handelt es sich um kleine Computer mit, je nach Aufgabenbereich, variierender Größe und Mächtigkeit. Bis zu 100 ECUs befinden sich in einem modernen Fahrzeug [HKGF09]. Selbst Einstiegsmodelle besitzen 30-50 ECUs, um elektronische Komponenten wie Türen und Kofferraum zu steuern [Cha09]. Auch intelligente Systeme wie Fahrassistenz, Toter-Winkel-Monitor und Kollisionswarnsysteme arbeiten auf ECUs [Sin19]. Von Sensoren gesammelte Daten werden über das fahrzeuginterne Netzwerk an ECUs gesendet und weiterverarbeitet [LCZ+14]. Mit der Entwicklung der Connected Cars und zunehmender Anzahl an Sensoren steigt die Menge der erfassten Daten des eigenen und der vernetzten Fahrzeuge stetig an. Aufgrund komplexer Berechnungen, z.B. Umgebungsanalyse über die Außenkameras, bedarf es immer größerer Rechenressourcen. Die Menge an Daten ist bereits so groß, dass alle 10 ms Hunderte von Inputs gesendet und analysiert werden müssen [Cha09]. Zum Beispiel produzieren Googles autonome Fahrzeuge 750 MB Daten pro Sekunde ausschließlich aus eigenen Sensoren [Ang13]. Trotz dessen muss sichergestellt sein, dass zeitkritische Anwendung, wie Airbags, ohne Verzögerung gesteuert werden können.

Mit steigender Anzahl an Sensoren und ECUs steigt auch die Komplexität der Verbindungen. Unter der Bezeichnung Vehicle-to-Sensor (V2S) werden die immer populärer werdenden kabellosen Verbindungen zusammengefasst [Sin19]. Bei diesen sind Sensoren nicht über das interne Bussystem an die ECUs angeschlossen. Stattdessen werden kabellose Technologien, wie z.B. Bluetooth [Blu21] und ZigBee [Zig15], eingesetzt. Diese sind jedoch nur sinnvoll, wenn sie die Ansprüche erfüllen können, die auch für verkabelte Lösungen gelten [BE12]. Dazu zählen eine geringe Latenz und hohe Zuverlässigkeit, sodass Echtzeitübertragungen stattfinden können [LCZ+14]. Das V2S Netzwerk hat aufgrund der unbeweglichen Sensoren eine immer gleichbleibende Topologie. Innerhalb dieser gibt es im Normalfall ausschließlich Single-Hop-Verbindungen zwischen den Sensoren und den empfangenden ECUs [LCZ+14].

2.1.2. Vehicle-to-Vehicle

Der direkte Austausch zwischen Connected Cars wird als Vehicle-to-Vehicle (V2V)-Kommunikation bezeichnet. Dabei formen diese ein Vehicular Ad hoc Network (VANET), welches eine Unterklasse des Mobile Ad hoc Network (MANET) ist [AG14]. In diesem sind Fahrzeuge mobile Knoten und RSUs sind, als statische Knoten, ebenfalls Teil des Netzwerkes [CFR+16; KPD+09]. Connected Cars können Nachrichten an andere Connected Cars aussenden und von ihnen empfangen. Zwar sind sie in der Regel mit eigenen Sensoren ausgestattet, jedoch kann durch das Empfangen externer Nachrichten ein noch umfassenderes Bild der Umgebung erzeugt werden. Vor allem in Gebieten

mit schwacher Infrastruktur können Connected Cars auf diesem Weg an viele neue Informationen gelangen und die Sicherheit der Fahrt erhöhen. Gleichzeitig kann jedes Connected Car das Routing anderer Nachrichten übernehmen, um diese über Multiple-Hop-Verbindungen an weiter entfernte Fahrzeuge weiterzuleiten [AG14; KT19; LCZ+14; NHCB18].

Aufgrund dieser Eigenschaft ist es im VANET beispielsweise möglich, dass Einsatzfahrzeuge ihr Erscheinen ankündigen, selbst wenn diese noch nicht in Hör- oder Sichtweite sind [CFR+16]. Vorausfahrende Fahrzeuge können frühzeitig den Weg räumen und Ampelschaltungen so gestellt werden, dass die Fahrbahn freigehalten wird. Weiterer Nutzen der V2V-Kommunikation liegt in der Mitteilung der Fahrabsichten an umliegende Fahrzeuge. Bremsmanöver und Spurenwechsel können ohne menschliche Reaktionszeit kommuniziert und darauf reagiert werden [LCZ+14]. Besitzt ein Fahrzeug eine solche Autonomität ist auch Platooning möglich.

Platooning ist ein Konzept, in dem Fahrzeuge dicht aneinander auffahren können, ohne ein Unfallrisiko einzugehen. Dies ist möglich, indem die Fahrzeuge Informationen, wie Geschwindigkeit, miteinander teilen und sich synchronisieren. Auf diese Weise ist sichergestellt, dass der Abstand immer eingehalten wird. Macht ein Fahrzeug der Kolonne dennoch ein anderes Manöver, z.B. einen plötzlichen Bremsvorgang, teilt es diese Information der Kolonne mit. Diese kann durch den Empfang der Information gleichzeitig mit demselben Manöver reagieren und weiterhin den Abstand einhalten. Daraus resultiert ein System, das effizienten und sicheren Transport bei hohen Geschwindigkeiten ermöglicht. Zusätzlich ermöglicht das dichte Auffahren einen reduzierten CO_2-Ausstoß und Treibstoffverbrauch [BSC+12; PKÅ+20].

Aufgrund der Mobilität ist das VANET sehr dynamisch. Unterschiedliche Geschwindigkeiten, Fahrtrichtungen und Fahrtverläufe sorgen für ein regelmäßig wachsendes und schrumpfendes Netzwerk. Jedoch ist es selbst bei einer geringen Fahrzeugdichte möglich VANETs zu bilden [KPD+09]. Ein Vorteil des VANET ist, dass es bei Connected Cars festgeschriebene Straßenverläufe gibt. Dadurch lässt sich trotz dynamischer Topologie Vorhersagen über die zukünftige Zusammensetzung des Netzes treffen [HFB09]. Auf dieser Grundlage gibt es eine Vielzahl von Algorithmen mit dem Ziel, ein optimales Cluster aus VANET-Knoten zu finden [CFR+16].

Aufbauend auf WiFi, ist Dedicated Short Range Communication (DSRC) eine Technologie, um die Kommunikation zwischen Fahrzeugen zu ermöglichen. DSRC basiert auf dem Standard IEEE 802.11p [IEE10]. In diesem sind der Physical- und Data Link Layer des Open Systems Interconnection (OSI)-Modells definiert [LCZ+14]. IEEE 802.11p ist ergänzt um Wireless Access in Vehicular Environments (WAVE), welches in IEEE 1609 festgehalten ist [IEE19]. Dieses ist speziell für den Automobilgebrauch entwickelt und definiert die oberen Layer [AG14]. DSRC besitzt eine maximale Reichweite von 1 km und kann mit bis zu 27 MBps auf dem Frequenzbereich um 5,9 GHz übertragen [AG14; Sin19]. In einem Vergleich möglicher VANET-Technologien stellten Anwer und Guy [AG14] heraus, dass DSRC aufgrund der Zuverlässigkeit, sicheren Übertragung und geringen Latenz am vielversprechendsten ist und erklären so die weite Verbreitung. In Abschnitt 2.1.4 wird DSRC erneut aufgegriffen und mit einer moderneren Technologie, die in direkter Konkurrenz zu DSRC steht, verglichen. Gleichzeitig wird festgehalten, welche Technologie zukunftsweisender ist.

2.1.3. Vehicle-to-Infrastructure

Mit V2V existiert eine Technologie, die Autonomität in Connected Cars ermöglicht. Diese reicht jedoch nicht aus, um umfassenden Schutz zu garantieren. Bei undeutlichen Wetterverhältnissen, wie Nebel, können Straßenmarkierungen und Schilder gegebenenfalls nicht von Sensoren erkannt werden. Die Fähigkeit, Informationen von anderen Fahrzeugen zu erhalten, hat in diesem Fall keinen Nutzen mehr. Ohnehin kann nie garantiert werden, dass andere Connected Cars in der Umgebung sind. Aus diesem Grund ist ein hybrides Modell aus V2V und Vehicle-to-Infrastructure (V2I) notwendig [NHCB18].

Bei V2I kommunizieren Connected Cars mit, in die Infrastruktur eingebetteten, RSUs. Selbst bei geringer Verbreitung von Connected Cars können diese dennoch durch V2I Informationen über die Umgebung erhalten [SLY+16]. Die Infrastruktur ist Teil des VANETs, weshalb für die Kommunikation, wie bei V2V, DSRC verwendet wird [AG14; LCZ+14]. Als RSUs zählen Objekte, mit denen Connected Cars sich austauschen können. Dazu gehören unter anderem freie Parkplätze, die ihren Standort mitteilen [PKÅ+20] und Ampeln, die ihren aktuellen Zustand aussenden, sodass Fahrzeuge sich nicht auf die eigenen Sensoren zur Erkennung des Lichtsignals verlassen müssen [MLKS18]. Mobile RSUs sind eine Möglichkeit eine größere Flächenabdeckung für V2I zu ermöglichen. Ercan et al. [EAM18] konnten zeigen, dass bereits ein Anteil von 5% mobiler RSUs genügt, um die Wahrscheinlichkeit eines Verbindungsaufbaus mit der Infrastruktur zu verdoppeln.

Die Infrastruktur kann auch die Verkehrssteuerung übernehmen. Dafür können, neben den Daten der Fahrzeuge, zusätzlich von der Infrastruktur selbst gesammelte Daten herangezogen werden. Mit Hilfe von Sensoren, welche in die Infrastruktur eingebettet sind, kann sich ein Bild des Zustands der Umgebung gemacht werden [GZC18]. Ein zentralisierter ITS-Server kann auf Basis der gesammelten Informationen einen optimalen Verkehrsfluss berechnen und den Fahrzeugen mitteilen [SLY+16].

RSUs können auch dazu dienen Connected Cars nahen Datenspeicher, Rechenleistung und eine Verbindungsmöglichkeit ins Internet zur Verfügung zu stellen. Mit umfangreicher Verteilung der RSUs ist auch Edge-Caching für die Übertragung großer Daten mit geringer Latenz möglich [GSS+18]. Studien haben ergeben, dass bei autonomen Fahrzeugen bereits 0,75 GB Sensordaten pro Sekunde anfallen [YKK+19]. Mit Hilfe von Edge Computing können auch solch große Datenmengen schnell gespeichert und verarbeitet werden [YKK+19]. Zusätzlich können Services, wie Infotainment, zur Verfügung gestellt werden. Für die effiziente Nutzung der Edge Nodes ist es wichtig, die benötigten Daten im richtigen Moment an den bestmöglichen Nodes bereitzustellen. Um dies zu ermöglichen existieren Algorithmen, die dies auf Basis der Fahrtrichtung probabilistisch bestimmen [MCC+16].

2.1.4. Cellular Vehicle-to-Everything

Unter Cellular Vehicle-to-Everything (C-V2X) sind verschiedene Technologien zusammengefasst. Namensgebend, ist C-V2X in der Lage, sich mit *allem* zu verbinden und umfasst somit sowohl V2V, als auch V2I. Es basiert auf Mobilfunktechnologie und hat 2018 mit Release 15 [3GPb] von 3rd Generation Partnership Project (3GPP) den bisherigen 4G-Standard mit dem neueren 5G-Standard abgelöst [Sin19]. Bereits seit Release 14 [3GPa] hat 3GPP Gerät-zu-Gerät-Kommunikation, d.h. Kommunikation, ohne auf Basisstationen angewiesen zu sein, standardisiert [PKÅ+20]. Um dies

zu ermöglichen, definiert der Standard zwei Interfaces: Ein Wide Area Network (WAN)-Interface, um die Geräte mit Basisstationen zu verbinden und ein PC5-Interface, welches kein Mobilfunknetz benötigt, um eine Verbindung mit anderen Geräten aufzubauen [5G 17].

Durch die Verwendung des Mobilfunknetzes ist es möglich, die C-V2X-Verbindung auf zwei Wege einzurichten. Zum einen kann die Kommunikationsschnittstelle, ähnlich wie bei DSRC, fest in das Fahrzeug integriert sein. In diesem Fall hätte das Fahrzeug ein eigenes Mobilfunkmodul verbaut. Zum anderen kann das eigene Smartphone verwendet werden. Über Systeme, wie z.B. MirrorLink [Car], kann das Fahrzeug das Mobilfunkmodul eines angeschlossen Smartphones für die Kommunikation nutzen [LCZ+14].

Im direkten Vergleich schneidet die Kommunikation über PC5 besser ab als über DSRC. In einer Studie haben 5G Automotive Association [5G 17] gezeigt, dass der erfolgreiche Empfang von Warnnachrichten bei PC5 größer ist als bei DSRC. Die Untersuchung wurde durchgeführt für Autobahnen, städtische- und ländliche Umgebungen, jeweils mit und ohne Kreuzung. Aus den Ergebnissen wurden Statistiken für die Anzahl an resultierenden Unfällen berechnet. Diese zeigen, dass es in der Zeitspanne von 2018 bis 2040 33% weniger Todesfälle beim Einsatz von PC5 geben würde. Die Empfangsquote ist zurückzuführen auf eine geringere Sensitivität des PC5-Empfängers, als bei DSRC, wodurch Nachrichten bei schwachem Signal besser erkannt werden [SLY+16]. Trotz der jungen Technologie findet C-V2X aufgrund des Einsatzes von 5G bereits jetzt viele Befürworter über DSRC [5G 17; HER19; Sin19; TKR21].

2.2. Datenpipelines

Der Datenfluss in Connected Cars ist äußerst komplex. Die Sensoren produzieren große Mengen an Daten. Diese müssen in vielen Verarbeitungsschritten aufbereitet werden, um aus ihnen Informationen zu gewinnen. Zum Beispiel werden bei der Analyse von Bildmaterial Vorverarbeitungsschritte, wie Parsen und Feature Extraction, benötigt [WZX+16]. Das Ergebnis dieser Verarbeitung kann in Aktuatoren, z.B. zur Steuerung des Fahrzeuges, verwendet werden. Um die Komplexität der Verarbeitung aufzulösen, eignen sich Datenpipelines [RSGJ13]. Die Sensoren und externen Informationsquellen eines Connected Cars bilden die Startpunkte der Pipelines und werden als Datenquellen bezeichnet. Datensenken sind die Zielpunkte einer Pipeline, an denen die Informationen eingesetzt werden. Dies können z.B. Aktuatoren oder Datenbanken sein. Datenquellen und Datensenken sind über eine Reihe von Prozessen miteinander verbunden. Dabei wird der Ausgang eines Prozesses zum Eingang des Nächsten [RBOW20]. Die Aufteilung komplexer Prozesse in Teilschritte macht die Verarbeitung der Daten transparenter. Munappy et al. [MBO20] empfehlen, dass jeder Prozess nur eine einzige Funktionalität implementieren soll. Dies ermöglicht eine leichtere Identifizierung fehlerhafter Verarbeitungen, erhöht aber gleichzeitig die Anzahl der Prozesse und Verbindungen. Zusätzlich wird ermöglicht, dass Teilschritte wiederverwendet werden können, anstatt die Funktionalität mehrfach zu implementieren [RSGJ13]. Der Datenfluss und die Verarbeitung sind somit leichter nachzuvollziehen. Dies sorgt für ein besseres Verständnis der Gesamtarchitektur. Damit einhergehend wird in Unternehmen eine bessere Kommunikation zwischen zusammenarbeitenden Teams ermöglicht [MBO20].

Datenpipelines unterstützen heterogene Umgebungen. Somit können Daten in vielfältigen Formaten entstehen und dennoch weiterverarbeitet werden [MBO20; WZX+16]. Aufgrund dieser Eigenschaft fällt neben der funktionalen Komplexität auch technische Komplexität an. Bei der Implemen-

tierung der Pipeline muss gesichert sein, dass Schnittstellen komplexe Datentypen übertragen können [EQS17]. Zusätzlich kann es unterschiedliche Datenübertragungsarten geben. Daten können sowohl in Streams als auch in Batches übertragen werden. Eine sehr unregelmäßige Übertragung ist ebenfalls möglich [RBOW20]. Eine Herausforderung für Datenpipelines ist, wenn diese Änderungen benötigen. Vor allem komplexe Datenpipelines durchlaufen eine Evolution, welche die Struktur ändern kann. Dazu gehört, dass neue Datenquellen in das System integriert werden. Potenziell unterscheiden sich diese von den bisher existierenden Quellen und benötigen eine spezielle Behandlung. Gleiches gilt für das Format der produzierten Daten, welches inkompatibel mit den Formaten bestehender Verarbeitungsprozesse sein kann [MBO20]. Anforderungen können sich auch in anderen Bereichen der Pipeline ändern. Es muss davon ausgegangen werden, dass sich jeder Teilschritt verändert. Zum Beispiel kann sich die Funktionalität einer Komponente ändern, sodass andere Daten erwartet oder produziert werden. Dies könnte Auswirkungen auf den gesamten Datenfluss haben, wodurch ein hoher Aufwand für die Wartung von Datenpipelines benötigt wird.

Datenpipelines können Latenzen mit sich bringen. Durch das Aufteilen komplexer Verarbeitungen in Teilschritte werden ebenfalls die zu durchlaufenden Verbindungen vermehrt. Das erhöht den Datenverkehr und sorgt für hohe Last auf den einzelnen Verbindungen. Ist der Datenfluss höher als die Geschwindigkeit der Datenverarbeitung, kann es zu Rückstau und somit Latenz kommen [RBOW20]. Die Verarbeitungsschritte der Datenpipelines müssen fehlertolerant sein. Produziert eine Komponente falsche Ergebnisse, kann dies alle nachfolgenden Komponenten dazu bringen ebenfalls falsche Ergebnisse zu produzieren [RSGJ13]. Ein robustes Design, welches den Datenfluss überwacht und auf Korrektheit überprüft ist daher wichtig [RBOW20]. Dennoch muss eine Balance zwischen Robustheit und Komplexität der Datenpipelines bewahrt werden, um Wartbarkeit und Verständlichkeit nicht zu verlieren [MBO20].

3. Verwandte Arbeiten

In diesem Kapitel werden thematisch verwandte Arbeiten aus den Bereichen Datenpipelines und Connected Cars vorgestellt. In seiner Dissertation hat Hirmer [Hir18] sich mit der Modellierung und Ausführung von Datenflussmodellen beschäftigt. Besonderer Fokus wurde auf die Anwendbarkeit verschiedener Nutzergruppen gelegt. Auf diese Weise können Domänennutzer außerhalb der IT-Domäne mit den Modellen arbeiten. Für die Modellierung der Datenflüsse wurden Anforderungen bezüglich Flexibilität, Verständlichkeit, Abstraktionen und Visualisierbarkeit aufgestellt. Das Ergebnis ist ein graphisches Modellierungswerkzeug, welches in seinen Grundzügen dem dieser Arbeit (vgl. Kapitel 6) ähnelt. Über das Pipes-and-Filters-Konzept werden Datenquellen und Verarbeitungsschritte verbunden und formen Datenflüsse. Um Domänennutzern die Modellierung zu ermöglichen und somit gleichzeitig mehrere Domänen in einem Modell zu verbinden, setzt Hirmer *Modellierungsmuster* ein. Modellierungsmuster repräsentieren Reihen von Datenquellen und Verarbeitungsschritten. Sie sollen dazu genutzt werden häufig wiederkehrende Teilprozesse zusammenzufassen und wiederverwendbar zu machen. Technische Details werden durch die Beschreibung der Modellierungsmuster abstrahiert, sodass jeder Domänennutzer sich auf die Modellierung seiner Domäne fokussieren kann. Auf diese Weise entsteht eine Menge von ausformulierten Modellierungsmustern, welche von anderen Domänennutzern als einzelne Verarbeitungsschritte wiederverwendet werden können, ohne die Details kennen zu müssen. Die Implementierung hinter den Modellierungsmustern kann unterschiedlich ausfallen, um z.B. verschiedene nichtfunktionale Anforderungen zu erfüllen. Das resultierende Modell beinhaltet Datenflüsse, die sich über viele Domains verbreiten, aber dennoch zusammenhängend sind. Das Modellierungswerkzeug von Hirmer unterscheidet sich von dem dieser Arbeit durch den Einsatzbereich. Die Arbeit von Hirmer ist für generische Anwendung ausgelegt, während in dieser Arbeit der Fokus auf Connected Cars liegt. Dadurch ist die Anwendung domänenspezifischer Konzepte ermöglicht.

Wu et al. [WZX+16] haben ein Framework für das Erstellen von Datenpipelines für heterogene Umgebungen entwickelt. Das Ziel der Autoren ist es eine Lösung für die Integration heterogener Umgebungen zu schaffen, welche keinen *Glue Code* benötigt und die Wartbarkeit erleichtert. Zusätzlich soll die Dynamik der Datenpipelines, z.B. das Hinzufügen von Quellen und Ändern von Schnittstellen, nachvollziehbarer gemacht werden. Das entwickelte Framework *Pipeline61* verbindet die Erzeugung von Pipelines in Spark-, MapReduce- und Shell-Umgebungen. Unter einer gemeinsamen Syntax können Jobs dieser Technologien definiert und miteinander verbunden werden. Zur Unterstützung der Nachvollziehbarkeit der sich stetig ändernden Datenpipelines besitzt Pipeline61 eine integrierte Versionskontrolle. Diese historisiert alle Versionen des Datenflussmodells mit den Verbindungen der Jobs. Zusätzlich werden alle Versionen der einzelnen Jobs festgehalten, sodass bei Fehlern ein Rollback zu alten Versionen durchgeführt werden kann. Umso komplexer die Datenpipelines werden, desto wichtiger ist es die Wartbarkeit durch Funktionen dieser Art zu sichern. Während Wu et al. Datenpipelines auf einer programmatischen Ebene behandeln, wird sich in dieser Arbeit auf das Konzeptionelle konzentriert.

Dhall und Solanki [DS17] stellen ein Szenario vor, welches den Datenfluss von Connected Cars unter Einsatz verschiedener Technologien demonstriert. Die verbauten Sensoren können gesammelte Daten über das Internet an Backend-Services senden, welche den Zustand des Fahrzeuges bestimmen. Dazu zählt die Feststellung von Materialverschleiß oder ähnlichen Schäden. Diese Informationen können genutzt werden, um *Predictive Maintenance* umzusetzen. Bei diesem Verfahren wird berechnet, wann und in welcher Form Wartungsarbeiten am Fahrzeug vorgenommen werden müssen. Dies ist eine Verbesserung der verbreiteten *Periodic Maintenance*, bei welcher in regelmäßigen Zeitabständen eine Werkstatt für Kontrolluntersuchungen aufgesucht wird. Dieser Anwendungsfall demonstriert, wie Connected Cars durch ihre Vernetzung, Services im Internet nutzen können. Diese Services können Fahrzeugdaten analysieren und feststellen, welche Bauteile wie stark beschädigt sind. Wird festgestellt, dass Reparaturen nötig sind, senden die Services eine Benachrichtigung zurück an die Fahrer. Neben der Kosteneinsparung durch Vermeidung unnötiger Werkstattaufenthalte bringt die Vernetzung weitere Vorteile mit sich. Zum Beispiel kann automatisiert eine Terminanfrage an die Werkstatt ausgesendet werden, welche detaillierte Informationen über den Schaden enthält. Die Architektur des Systems ist nachrichtenbasiert. Die Connected Cars nutzen z.B. Mobilfunk, um ihre Daten an einen Message Queuing Telemetry Transport (MQTT)-Broker in der Cloud zu übermitteln. Von dort aus werden die Daten an Services für weitere Verarbeitungsschritte weitergeleitet. Die von Dhall und Solanki demonstrierte Datenverarbeitung findet größtenteils in der Cloud statt. Hingegen wird sich in dieser Arbeit auf die Verarbeitung innerhalb der Connected Cars und die Verbindung zur Cloud befasst.

Pelliccione et al. [PKÅ+20] betrachten Connected Cars als *System of Systems (SOS)*. Das bedeutet, Connected Cars können als alleinstehende Systeme funktionieren, aber haben als ein zusammenarbeitendes System mehr Möglichkeiten. Mit dieser Auffassung stellen die Autoren eine Referenzarchitektur eines Connected Cars als Bestandteil des SOS vor. Dazu wurden Stakeholder aus verschiedenen Gruppen, wie End-User, Entwickler und Betreiber identifiziert. Mit diesen wurden Anliegen erarbeitet, z.B. bezüglich Sicherheit und Datensynchronisation, die es zu beachten gilt. Die vorgestellte Referenzarchitektur bezieht sich auf die Wahrnehmung des internen Zustandes des Fahrzeuges und dessen externe Umgebung. Deshalb ist diese in drei zusammenarbeitende Sichten unterteilt: Die interne und externe Sicht, welche von eigenen Sensoren wahrgenommen werden, und eine weitere externe Sicht, wie sie von anderen Connected Cars wahrgenommen wird. Für die interne Sicht werden die Daten aller internen Sensoren bereinigt und analysiert. Schließlich wird das *World Model* bereichert, welches den Zustand des Fahrzeuges und der Umgebung widerspiegelt. Die externe Sicht, welche von anderen Connected Cars erzeugt wird, muss gesondert behandelt werden. Da diese Daten von externen Quellen kommen, müssen sie erst validiert werden. Dazu gehört eine Firewall, welche Angriffe auf das Fahrzeug verhindert und eine anschließende Prüfung der Vertrauenswürdigkeit. Daten, welche diese Überprüfungen bestehen, werden zusammen mit den eigenen Daten der externen Sicht weiterverarbeitet. Nach einer gemeinsamen Interpretation der Daten, reichern diese das World Model an. Zusätzlich werden sie für die Kontrolle des Fahrzeuges herangezogen, um autonomes Fahren zu verwirklichen. Pelliccione et al. geben mit ihrer Referenzarchitektur eine Übersicht über die Verarbeitungsschritte in Connected Cars. Die Modellierung der Datenpipelines in dieser Arbeit ermöglichen eine detaillierte Umsetzung dieser Vorstellung.

4. Anforderungen

Ein sicheres Connected-Cars-System kann nur funktionieren, wenn die Technologie flächendeckend auf den Straßen im Einsatz ist [HCL04]. Connected Cars können sich mithilfe genauer Standortdaten gegenseitig über ihre Position informieren und profitieren zusätzlich von eingebauter Sensorik. Die Sensoren können die Anwesenheit von Verkehrsteilnehmern oder Hindernissen wahrnehmen, welche nicht mit dem System verbunden sind. Dadurch haben sie einen besseren Überblick über den Verkehr und können Gefahren besser einschätzen. Zusätzlich besteht die Möglichkeit sich gegenseitig über diese Hindernisse zu informieren. Durch diese Vernetzung können automatisch Sicherheitsmaßnahmen, wie Umfahren, eingeleitet werden, selbst wenn weder die Fahrer noch die eigenen Sensoren das Hindernis wahrgenommen haben.

Je verbreiteter der Einsatz von Connected Cars ist, desto größer ist der Nutzen, den sie als Ganzes erzeugen. Allerdings genügt es nicht die Fahrzeuge mit der nötigen Technologie auszustatten, da die Nutzer sich erst dazu bereit erklären müssen diese einzusetzen. Aktuellen Umfragen [Del21] zufolge gehen die Meinungen über die Nützlichkeit von Connected Cars weit auseinander. So sehen in China und Indien über 80% der Befragten einen Vorteil in der Vernetzung der Fahrzeuge, wohingegen dieser Wert in Deutschland bei nur 33% liegt. Ein Grund hierfür ist, dass die Technologie noch unerprobt ist und viele Anforderungen erfüllen muss, um von der Masse akzeptiert zu werden. In diesem Kapitel werden die Anforderungen an Connected Cars vorgestellt. Da eine Modellierung über Datenpipelines vorgesehen ist, werden zusätzlich die Anforderungen an die Modellierung aufgezeigt. Zuerst wird in Abschnitt 4.1 auf den Schutz der Privatsphäre eingegangen. Danach folgen in Abschnitt 4.2 Anforderungen an das Netzwerk. Danach werden in Abschnitt 4.3 allgemeine Sicherheitsaspekte behandelt. In Abschnitt 4.4 geht es um die Komplexität der Modellierung von Datenpipelines. In Abschnitt 4.5 folgen Anforderungen an die Interoperabilität von Connected Cars. Schließlich wird in Abschnitt 4.6 auf die Datenverarbeitung eingegangen.

4.1. Privatsphäre

Um die Funktionstüchtigkeit von Connected Cars zu gewährleisten, müssen diese große Mengen an Sensordaten verarbeiten [MLKS18]. Selbst heute gängige Fahrzeuge produzieren bereits viele Daten aus eigenen Sensoren (vgl. Abschnitt 2.1.1). Mit diesen Daten werden z.B. Sicherheitsfunktionen gewährleistet, wie die Kontrolle des Motors, aber auch das Fahren erleichtert. Technologien wie Einparkkamera, Spurenhalteassistent und GPS [HCL04] gehören bereits zum Standard. Mit jedem dieser Sensoren werden Daten aufgezeichnet und verarbeitet. Diese Daten beinhalten aber auch Eingriffe in die Privatsphäre. Zum Beispiel zeichnet eine Einparkkamera die Umgebung auf und könnte somit fremde Gärten filmen, ein Spurenhalteassistent sammelt Daten zum Fahrverhalten und das GPS arbeitet mit hochsensiblen Standortdaten. Die Europäische Union (EU) hat 2018 das eCall-System [ADA19] eingeführt, welches in jedes neu produzierte Fahrzeug eingebaut werden

muss. Dieses System sendet bei Verkehrsunfällen automatisch einen Notruf an die Rettungsleitstelle [HCL04; LML15]. Der Notruf beinhaltet für die Rettung wichtige Informationen wie Zeitpunkt, Standort und Anzahl der Insassen. Trotz der Nützlichkeit steht eCall für Eingriffe in die Privatsphäre und der Ermöglichung von Überwachung in der Kritik. Wisman [Wis16] hebt die technischen und rechtlichen Probleme hervor und argumentiert das System schaffe die Basis für eine Nutzung, die über das angedachte Ziel hinausgehe. Widersprüchliche und unzureichende Bestimmungen würden eine Nachverfolgung, selbst ohne Eintreten eines Unfalls, erlauben. Zudem führt das System durch Nutzung einer Subscriber Identity Module (SIM)-Karte ein weiteres Angriffsziel für Hacker ein. Wisman [Wis16] beschreibt ein Szenario, in dem ein EU-Land das System zur Überwachung einsetzen könnte und unterstreicht, dass es von der EU keinen rechtlichen Schutz dagegen gibt. Final wirft er dem System vor, es sei von kommerziellem Interesse inspiriert worden.

Unsicherheiten über den Schutz der Privatsphäre gibt es auch bei den Fahrern. Umfragen zufolge sind 51% der Deutschen abgeneigt Connected-Car-Services zu nutzen, da sie ihre Privatsphäre nicht gefährden möchten. Nawrath et al. [NFM16] präsentieren die verschiedenen Kategorien von privaten Daten, die im Fahrzeug gesammelt werden, wie folgt:

- Identifikation
- Kundenaccount
- Fahrzeugzustand
- Fahrverhalten
- Biometrie und Gesundheitszustand
- Position
- Infotainment
- Persönliche Kommunikation

Besonders Infotainment und die persönliche Kommunikation beinhalten viele private Daten, wie etwa Sprach- und Textnachrichten, Aktivitäten auf sozialen Medien, persönliche Kontakte und Musikgeschmack [LML15]. All diese Daten werden geteilt und durchlaufen dabei unterschiedliche Stationen. Manche Informationen werden mit Fahrzeugen im direkten Umfeld und manche mit zahlreichen Empfängern in der Cloud geteilt. Auch smarte Ampeln, Parkplätze und Gebäude können zu den Empfängern zählen [PKÅ+20]. Diese können sogar als Broker zur Weiterverteilung der Daten eingesetzt werden [GS18]. Je weiter die Verbreitung der Daten, desto größer ist die Chance, dass es an einer Stelle zu Angriffen und somit zum Verlust privater Daten kommt.

Schutz der Privatsphäre ist wichtig, um das Vertrauen in Connected Cars und somit deren Verbreitung zu fördern [OEB18; ZNY+17]. Dafür muss verstanden werden, wer die Stakeholder privater Daten sind und wofür sie die Daten verwenden. Dazu zählen zum Beispiel Hersteller, Zulieferer oder auch Versicherungsfirmen [LML15]. Abhängig vom Verwendungszweck profitieren die Fahrer dabei unterschiedlich stark von dem Teilen ihrer Daten. Durch das Teilen der Daten mit Versicherungsfirmen können diese Risikoanalysen auf Basis der Geschwindigkeit und Position des Fahrzeuges durchführen [LML15]. Für manche Fahrer könnte dies zu Nachteilen führen und sie dazu verleiten das Connected-Car-System nicht zu akzeptieren, was den Nutzen auch für andere Straßenteilnehmer mindert. Werden dieselben Daten allerdings für einen anderen Nutzen

herangezogen, kann die Akzeptanz wiederum anders ausfallen. Ostern et al. [OEB18] zeigen, dass die Erwartungshaltung dabei eine zentrale Rolle spielt und nennen vier Kontexte, welche diese beeinflussen:

- Art der Information
- Verwendungszweck
- Politischer Kontext (Gesetze)
- Technologische Anwendung

Der Verwendungszweck der Daten lässt sich aufschlüsseln in zwei wichtige Faktoren: Die Angebrachtheit und die Verbreitung der Daten [OEB18]. Zur Veranschaulichung der Wichtigkeit von Angebrachtheit und Verbreitung, eignet sich ein Beispiel mit Positionsdaten. Diese sind notwendig, um die Sicherheit der Fahrer zu gewährleisten und stoßen daher auf Akzeptanz in der Verbreitung, obwohl diese hochsensibel sind. Sollte allerdings die Angebrachtheit nicht gewährleistet sein und die Daten werden für kommerzielle Zwecke verwendet, ist der Eingriff in die Privatsphäre nicht gerechtfertigt und die Akzeptanz der Technologie sinkt wiederum. Auch der politische Kontext hat einen Einfluss. Gibt es rechtliche Vorgaben, neigen die Nutzer dazu über den Eingriff in die Privatsphäre hinwegzusehen [OEB18].

Rechtliche Vorgaben reichen allerdings nicht aus, um einen tatsächlichen Schutz zu gewährleisten. Durch das Abgeben von Einwilligungserklärungen kann dieser Schutz umgangen werden. Beim Kauf eines Fahrzeuges wird der Käufer mit vielen Dokumenten konfrontiert und setzt Unterschriften ohne die legalen Hintergründe zu kennen oder die Möglichkeit zu haben, manche Bedingungen der Verträge zu verhandeln. Moderne Fahrzeuge besitzen Features, die vielleicht nicht einmal verwendet werden, aber trotzdem eine Einwilligung für die Nutzung der Daten benötigen. Letztendlich sind Kunden überwältigt von den rechtlichen Bestimmungen und willigen in Datennutzungszwecke ein, die ihre Privatsphäre offenlegen, obwohl sie nur das Fahren interessiert, wie von Lawson et al. [LML15] ausführlich argumentiert. Bestätigt wird dies von Hubaux et al. [HCL04], welche einen Vergleich zwischen Connected Cars und den bereits etablierten Systemen des Mobilfunknetzes und elektronischen Zahlungsmethoden anstellen. Sie kommen dabei zu dem Schluss, dass der Nutzen so groß war, dass der Eingriff in die Privatsphäre die Verbreitung nicht abhalten konnte. Es stelle sich daher nicht die Frage, ob der Eingriff gerechtfertigt ist, sondern wie er am erstrebenswertesten gestaltet werden kann.

Ein großes Problem, das den Schutz von Daten erschwert, ist Data Mining [BE99]. Data Mining ist in der Lage Verbindungen zwischen Datenmengen herzustellen und an Informationen zu gelangen, die für einen anderen Zweck vorgesehen waren. Aus diesem Grunde ist Anonymisierung von Daten meist nicht ausreichend [Cot09; LML15]. Wenn z.B. ein Third-Party-Service ein Datenleck hat und Standortdaten mit Namen der Personen veröffentlicht werden, sind auch Daten dieser Kunden aus anderen Services, welche Anonymisierung anwenden, in Gefahr. Sind bei diesen Services ebenfalls Standortdaten hinterlegt, können diese erkannt und auf die Personen zurückgeführt werden. Selbst bei voller Anonymität und ohne diese mit anderen Datensets zu verbinden, lässt es sich nicht vermeiden, dass Schlüsse über eine Person gezogen werden können. Durch Standortdaten kann herausgefunden werden, wo sich eine Person häufig aufhält. Das reicht meist schon aus, um Heimat, Arbeitsplatz, besuchte Restaurants und Schönheitschirurgen zu identifizieren [EL14]. Mit diesen Informationen lassen sich bereits Schlüsse über Essvorlieben und Wohlstand ziehen. Dies zeigt, wie

viele Informationen sich bereits aus wenigen Daten ableiten lassen. Daher ist es wichtig, diese so sorgsam wie möglich zu behandeln. Vor allem, da im Kontext der Connected Cars mit sehr großen Mengen an sensiblen Daten gearbeitet wird.

In manchen Fällen können Daten geschützt werden, indem diese aggregiert oder verzerrt [OEB18] werden. Bei einer Aggregation würden die Daten unterschiedlicher Personen gesammelt und miteinander verrechnet werden. Auf diese Weise können keine einzelnen Werte erkannt, aber immer noch Schlüsse aus dem aggregierten Wert gezogen werden. Bei der Verzerrung werden die Daten vor Veröffentlichung leicht verfälscht, sodass zwar einzelne Daten erkannt werden können, allerdings sind diese nie korrekt oder spiegeln nur einen Wertebereich wider. Der Einsatz dieser Methoden ist jedoch nicht immer möglich. Für die Erkennung von Staus wären diese einsetzbar, da erkennbar ist wie viele Fahrzeuge sich auf einer Fläche befinden, ohne die exakten Standorte zu kennen. Für die Aggregation reicht die Information, wie viele Fahrzeuge Teil der Aggregation waren. Jedoch bedarf es bei der Verzerrung mehr Achtsamkeit, um noch in der Lage zu sein, die Fahrzeuge einer bestimmten Spur oder Fahrtrichtung zuordnen zu können. So müsste auf Autobahnen eine Verzerrungsfunktion gewählt werden, welche die Positionen nur nach vorne oder hinten verzerrt. Eine Verzerrung zur Seite wäre problematisch, da ansonsten fälschlicherweise ein Stau für den Gegenverkehr erkannt werden könnte. Bei der Kollisionsverhinderung wären diese Verfahren nicht geeignet, da die Positionsdaten sehr präzise sein müssen.

Etablierte Lösungen für den Schutz privater Daten sind Anonymisierung und Verschlüsselung. Die Gründe, wegen welchen Anonymisierung nur begrenzt Schutz gewährt, wurden bereits aufgezeigt. Im Bereich der Verschlüsselung hingegen wurden zuletzt Fortschritte gemacht. Homomorphe Verschlüsselung macht es theoretisch möglich Operationen auf verschlüsselten Daten durchzuführen, ohne diese entschlüsseln zu müssen [Gen09; ZNY+17]. Die 2009 von Gentry [Gen09] vorgestellte voll-homomorphe Verschlüsselung ist jedoch zu ineffizient, um in der Praxis angewendet zu werden und wird seitdem ständig weiterentwickelt, um Sicherheit und Effizienz zu verbessern [YLSS21]. Letztendlich ist der Schutz von Daten ein äußerst komplexes Thema, das die digitalisierte Welt seit langem beschäftigt. Das Ziel des Konzepts dieser Arbeit ist es, diesen Schutz so umfangreich und früh wie möglich sicherzustellen. Noch vor Aussenden der Daten soll dies, durch Modellierung der Datenpipelines, gewährleistet sein.

4.2. Netzwerk

Für die namensgebende Funktionsweise von Connected Cars ist ein Netzwerk zwischen den Fahrzeugen für den Austausch von Informationen notwendig. Zusätzlich bilden Sensoren und ECUs ein Netzwerk innerhalb des Fahrzeuges. Die besondere Beschaffenheit eines Connected-Car-Systems stellt dabei ungewöhnliche Anforderungen. Unter anderem gibt es neben den Fahrzeugen weitere Verkehrsteilnehmer, die Teil des Systems sind, wie etwa Fußgänger oder Fahrradfahrer. Mit Hilfe von Smartphones können diese Teil des Netzwerkes werden. Im Vergleich zu Fahrzeugen haben Smartphones andere technische Eigenschaften, vorrangig leistungsschwächere Hardware. Eine stark begrenzte Batterie hat einen großen Einfluss auf die Leistungsfähigkeit [SLY+16]. Daher muss bei diesen Knoten mit einem vollständigen Ausfall gerechnet werden. Zusätzlich kann es Fußgänger geben, die kein Smartphone bei sich tragen. Insgesamt betrachtet, kann sich nicht darauf verlassen werden, dass die Umgebung, welche die Fahrzeuge über die Netzwerkknoten wahrnehmen, der Realität entspricht.

Wie zuvor erwähnt, ist eine weite Verbreitung von Connected Cars wichtig, um das gesamte System funktionsfähig zu machen. Dieses Ziel kann nur erreicht werden, wenn ein Fahrzeug über die gesamte Lebensdauer Teil des Connected-Car-Systems sein kann. Dazu muss sichergestellt werden, dass ältere Fahrzeuge, trotz sich ständig weiterentwickelnder Technologie, immer noch unterstützt werden [ABG+17]. Ohne Abwärtskompatibilität ist dies schwer vorstellbar, da die durchschnittliche Lebensdauer eines Fahrzeuges in Deutschland 18 Jahre beträgt [Sta14].

Connected Cars arbeiten mit zwei Arten von Netzwerk: Dem fahrzeuginternen und dem fahrzeugexternen Netzwerk. In Abschnitt 2.1 wurde bereits deren Aufbau und Funktionsweise erläutert. Im Folgenden werden die Anforderungen des fahrzeuginternen Netzwerkes in Abschnitt 4.2.1 erklärt. Im Anschluss werden in Abschnitt 4.2.2, zusammenfassend für V2V und V2I, die Anforderungen des fahrzeugexternen Netzwerkes behandelt.

4.2.1. Fahrzeuginternes Netzwerk

Moderne Fahrzeuge weisen immer mehr Funktionalitäten auf. Grundlegende Mechanismen, z.B. Antrieb und Bremse, aber auch Accessoires, wie Infotainment und viele weitere, werden von ECUs gesteuert und überwacht [CMK+11]. Dabei sind diese über Bussysteme wie CAN oder FlexRay miteinander verbunden. Ein Problem bei einer solchen Verbindung liegt darin, dass es bei einem Angriff möglich ist, Zugriff auf das gesamte interne Netz und den verbundenen ECUs zu bekommen. Dafür reicht es aus, eine einzelne ECU, mit der sich Nachrichten versenden lassen, zu kompromittieren [CMK+11; KSR+10]. Diese Referenzarchitektur stammt aus einer Zeit, in der Fahrzeuge noch nicht miteinander kommunizieren konnten. Trotz Ausstattung mit Kommunikationstechnologien ist die Architektur in modernen Fahrzeugen unverändert und somit nicht mehr zeitgemäß [Mcc17]. Je mehr Schnittstellen ein Fahrzeug anbietet, um sich mit der Außenwelt zu verbinden, desto leichter ist es eine Schwachstelle zu finden. Sobald diese aufgedeckt ist, ermöglicht die unangemessene Architektur, dass bereits ein simpler Angriff das gesamte CAN befällt. Weitere Herausforderungen bei der Sicherheit von Connected Cars sind in Abschnitt 4.3 formuliert.

Der geringe verfügbare räumliche Platz innerhalb des Fahrzeuges stellt das interne Netzwerk vor weiteren Herausforderungen. Dieser macht es schwierig für die stetig wachsende Anzahl an Sensoren Platz zu finden. Die benötigten Kabelverbindungen schränken diesen Platz weiter ein und belasten das Fahrzeug mit bis zu 50 kg [QWY10]. Aus diesem Grund wird an kabellosen Verbindungen geforscht (V2S) [LCZ+14]. Der Einsatz dieser Technologie ist jedoch erst realistisch, sobald ein kabelloses Netzwerk dieselbe Performanz und Zuverlässigkeit bietet, wie Kabel [BE12]. Der Verzicht auf Kabel wirft jedoch weitere Probleme auf. Die Interferenz benachbarter Fahrzeuge, vor allem an mehrspurigen Straßen mit langsam fließendem Verkehr, kann das interne Netz stören und somit die Zuverlässigkeit wichtiger Funktionen verringern [LCZ+14].

4.2.2. Fahrzeugübergreifendes Netzwerk

Connected Cars unterstützen verschiedene Arten von Netzwerken, um mit ihrer Umgebung zu kommunizieren. In Abschnitt 2.1 wurden bereits die Technologien hinter V2V, V2I und C-V2X erläutert. Grundlegend für die Funktionalität ist die Kommunikation mit anderen Fahrzeugen und der Infrastruktur wie Ampeln, Laternen, Gebäuden, Parkplätzen, Basisstationen und weiteren Verbindungspunkten [NFM16; PKÅ+20]. Unabhängig von der Umgebung muss sichergestellt

werden, dass Connected Cars sich mit dieser verbinden können. Aufgrund der Mobilität kann die Umgebung sehr vielfältig ausfallen. Im Stadtinneren gibt es viele Fahrzeuge und Verbindungspunkte in der Infrastruktur, mit denen parallel Verbindungen gehalten werden müssen. Gleichzeitig gibt es viele Hindernisse wie Gebäude, Tunnel und große Fahrzeuge wie Vans oder Busse, die das Signal stören können [BVF+10]. In ländlichen Gebieten gibt es weniger Infrastruktur, um sich über die Umgebung zu informieren und das Signal von Basisstationen kann sehr schwach sein. Auf Autobahnen müssen die Verbindungen trotz hoher Geschwindigkeiten gehalten werden.

Die Mobilität der Fahrzeuge bringt Probleme mit sich, selbst wenn diese sich nur langsam bewegen. Beim VANET gibt es kontinuierliche Änderungen in der Netzwerktopologie. Durch umgebende Fahrzeuge, die sich entfernen oder nähern, kommt es zu vielen Beitritten und Trennungen mit dem Netzwerk. Die daraus resultierenden Partitionierungen des Netzwerkes führen zu regelmäßigen Unterbrechungen des Datenflusses [LCZ+14]. Im innerstädtischen Verkehr kann die Last auf das Netzwerk innerhalb von Minuten stark variieren. Zum Beispiel können sich, bei einer langen Rotphase, an einer großen Kreuzung, viele Fahrzeuge auf engem Raum sammeln und so ein Datenstau verursachen [ABG+17]. Während der Fahrt ist die Kommunikation, vor allem mit entgegenkommenden Fahrzeugen, aufgrund der kurzen Verbindungsdauer erschwert [SLY+16]. Je höher dabei die Geschwindigkeiten, desto kürzer dauert die Verbindung an. Gravierend ist diese Einschränkung für Verbindungen, die viel Overhead mit sich bringen. Dazu zählt z.B. der Verbindungsaufbau oder das Sicherstellen von *exactly-once*-Kommunikation. Dieser notwendige Overhead reduziert die Zeit für die Übertragung der Nutzdaten weiter.

Bei der Ausbreitung des Signals kann es aufgrund der Mobilität zu weiteren technischen Problemen kommen. Vor allem bei hohen Geschwindigkeiten kann der Dopplereffekt auftreten. Bewegen sich zwei miteinander kommunizierende Fahrzeuge in entgegengesetzte Fahrtrichtungen, wird dieser noch verstärkt [CHS+07; SLY+16]. Innerorts kann es durch viele Hindernisse, wie etwa Gebäude auch zu Mehrwegempfang und allgemeiner Abschwächung, sowie Unterbrechung des Signals kommen [CHS+07; LCZ+14]. Connected Cars können mit ihren ausgesendeten Signalen zusätzlich für Interferenz sorgen, selbst wenn diese sich nicht bewegen [LCZ+14].

Für funktionierende V2I-Kommunikation existieren ebenfalls hohe Ansprüche. Eine geringe Latenz, zuverlässige Verbindung und hohe Bandbreite sind verlangt, um den Anforderungen nach Sicherheit und Infotainment nachzukommen [GSS+18]. Jedoch erschwert die hohe Mobilität der Fahrzeuge, ähnlich wie im VANET, den Datenaustausch. Fahrzeuge legen große Distanzen zurück, wobei sie regelmäßig den Funkzellen von Basisstationen beitreten und sie verlassen. Findet während des Verlassens eine Datenübertragung statt, muss der Download von einer anderen Basisstation fortgeführt werden. Bei Messungen mit über einer Million überwachter Connected Cars über einen Zeitraum von 90 Tagen haben Andrade et al. [ABG+17] festgestellt, dass bei den meisten großen Downloads drei bis zehn Basisstationen involviert sind. Der Handover zwischen den Funkzellen nimmt Zeit in Anspruch, während der die Verbindung gestört sein kann [SLY+16].

4.3. Sicherheit

Connected Cars sind Systeme, bei denen die Automatisierung eine zentrale Rolle spielt. Sie müssen in der Lage sein, sich selbstständig in einer ungewissen Umgebung zurechtzufinden. Dabei muss sich auf die eigenen Sensoren oder die Messungen anderer Fahrzeuge verlassen werden. Diese geben Auskunft über den Zustand der Straße, Wetterverhältnisse, Position und Fahrverhalten anderer

Verkehrsteilnehmer und vielem mehr. Mit diesen Daten müssen Connected Cars in der Lage sein auch bei hohen Geschwindigkeiten zuverlässig zu funktionieren. Zumal es sich zumeist um den Transport von Menschen handelt, können kleine Fehler fatale Konsequenzen haben, bis hin zum Verlust von Leben. Um dies zu verhindern, ist der Aspekt der Sicherheit in Connected Cars von fundamentaler Bedeutung.

Sicherheit unterteilt sich in Safety und Security. Safety beschäftigt sich mit der Verhinderung von Schäden an der Umgebung durch das System. Dazu zählen unter anderem Schäden am Menschen und der Natur. Bei Security handelt es sich hingegen darum, dass die Umgebung dem System nicht schaden kann, wie etwa dem Schutz der Daten vor Hackerangriffen [LNRT06].

Ist Security nicht gewährleistet, kann es auch keine Safety geben, da nicht garantiert werden kann, dass das System die Passagiere schützt [Hus16]. Bei Umfragen haben 59% der Deutschen ihre Angst vor Hackerangriffen bei Connected Cars bekundet [LML15]. Checkoway et al. [CMK+11] haben diesbezüglich eine Reihe von Experimenten durchgeführt. Es wurde versucht, über die folgenden Angriffspunkte in ein Connected-Car-System einzudringen:

- Indirekten physischen Kontakt, z.B. das Einlegen einer CD in den Media Player
- Kurzstrecken Kommunikation, z.B. Bluetooth
- Langstrecken Kommunikation, z.B. Funk

Durch Reverse Engineering ist es den Autoren gelungen, für jeden dieser Wege je zwei erfolgreiche Angriffe auszuführen und Zugriff auf das interne Netz zu bekommen. Koscher et al. [KSR+10] haben bereits gezeigt, dass dies ausreicht, um sich zusätzlich Zugriff auf weitere Komponenten, die mit dem internen Netz verbunden sind, verschaffen zu können. Folglich bieten sich dem Angreifer viele Möglichkeiten Schaden zuzufügen. Nach Gesprächen mit Vertretern der Automobilindustrie sind Checkoway et al. [CMK+11] zu dem Schluss gekommen, dass sich die Hersteller aufgrund der Neuheit der Technologie noch nicht mit Ernstfällen auseinandersetzen mussten. Beim klassischen, unvernetzten Fahrzeug kam es aufgrund der fehlenden Verbundenheit nicht zu einem Angriff auf das interne Netz. Dafür wäre physischer Kontakt nötig gewesen. Angreifer, die dies schaffen, können allerdings auch über direktem Wege dem Fahrzeug Schaden zufügen, wie dem Durchschneiden der Bremskabel. Aufgrund der bisher ausbleibenden digitalen Angriffe sah sich die Industrie nicht gezwungen präventive Maßnahmen zu ergreifen [CMK+11].

Es existiert eine Vielzahl von Angriffen und Zielen, die verteidigt werden müssen. Das Verbreiten falscher Daten kann eine falsche Realität vorspielen und falsche Fahrmanöver herbeiführen. Dabei ist unerheblich, ob die falschen Daten unbeabsichtigt sind und z.B. aus fehlerhaften Sensoren stammen, oder ob es sich um einen beabsichtigten Angriff handelt. Dennoch wird im Folgenden vorwiegend auf den Fall eines Angriffs eingegangen. Ein Connected Car könnte z.B. genutzt werden, um ein anderes anzugreifen, indem es falsche Basic Safety Messages (BSMs) versendet. Diese sind in SAE J2735 [SAE09] spezifiziert und dienen zur Information über Position, Fahrtrichtung, Geschwindigkeit und mehr. Genauso könnten falsche Angaben zur Verkehrssituation oder Unfällen gemacht werden [GS18]. Die eingebauten Sensoren eines Fahrzeuges könnten ebenfalls manipuliert oder deaktiviert werden und so dem Fahrzeug die Orientierung nehmen. Angriffe wie Denial of Service (DoS), bzw. Distributed Denial of Service (DDoS) können das gesamte System überlasten und somit unbrauchbar machen. DoS kann auch durch Störsender herbeigeführt werden, welche die gesamte Kommunikation zwischen Connected Cars lahmlegen [HCL04]. Die physischen Angriffspunkte sind vielfältig, wie Dinesh Kumar et al. [DNK+18] ausführen: Das GPS und das

LiDAR können durch falsche Signale fehlgeleitet werden und in Kameras verwendete Complementary Metal-Oxide-Semiconductor (CMOS) Sensoren können durch Hochleistungslampen geblendet werden. Direkter Schaden kann angerichtet werden, indem der Angreifer die Kontrolle über Steuerelemente übernimmt. Neben leichten Angriffen, wie der Steuerung des Media Players können auch die Grundfunktionen des Fahrzeuges, wie dem Abschließen der Türen, Beschleunigen und Bremsen kompromittiert werden [CMK+11; Hus16]. Auch das Installieren von Ransomware ist denkbar [GS18].

Die Herausforderungen beim Schutz privater Daten wurde bereits in Abschnitt 4.1 herausgestellt. Dies betrifft jedoch nur den Normalbetrieb. Bei Hackerangriffen ist der Einbruch in die Privatsphäre noch größer, sodass sogar Daten, die nie das Fahrzeug verlassen sollten, gestohlen werden können. Dazu zählen personenbezogene Daten, wie z.B. Kreditkarteninformationen. Der Diebstahl von Daten kann auch Teil einer Impersonation Attack sein, die dazu dient die Identität eines anderen Fahrzeuges zu übernehmen [HCL04]. Heutzutage ist es üblich das Smartphone mit dem Fahrzeug verbinden zu können, um z.B. Musik abzuspielen. Dies bietet bereits eine Schnittstelle, um in das System einzudringen. Dennoch werden solche Angriffspunkte mit zunehmender Digitalisierung und neuen Features zahlreicher. BMW kündigte 2010 das Feature *iPod Out* [BMW10] an, mit dem es möglich war ein iPhone mit dem Bordmonitor zu verbinden. Zwar werden den Nutzern damit mehr Funktionen geboten, allerdings öffnet dies die Schnittstelle für Angriffe noch weiter [CMK+11]. Diese Schnittstellen sind eine besondere Gefährdung, da sie die Möglichkeit eröffnen das System so zu manipulieren, dass es auch nach dem Einbruch und auf Distanz die Möglichkeit des Einbruchs bietet. Checkoway et al. [CMK+11] zeigten dies auf, als es ihnen gelang eine Media Access Control (MAC)-Adresse auf eine ECU einzuspeichern. Auf diesem Weg sorgten sie dafür, dass immer eine Bluetooth-Verbindung mit dem Gerät dieser MAC-Adresse zustande kommen soll. Folglich war ein leichtes Eindringen über Bluetooth möglich, selbst nachdem der ursprüngliche Angriff bereits beendet war. Dies zeigt, wie wichtig es ist, dass jeder Angriff in das System sofort bemerkt wird. Nur so ist es möglich, rechtzeitig Gegenmaßnahmen einzuleiten und in einen sicheren Zustand zurückzukehren [Mcc17].

Die Verschlüsselung von Daten ist ein Mittel, um den Nachrichtenverkehr zwischen Connected Cars zu schützen. Dies bringt jedoch Nachteile mit sich, welche die Nützlichkeit reduzieren können. Für die Berechnung der Verschlüsselungsfunktion wird Zeit aufgewendet, wodurch die Nachrichten verspätet ausgesendet werden [NJ05]. Selbst wenn es sich nur um Millisekunden handelt, kann diese Latenz bereits für Beeinträchtigungen sorgen. Zum Beispiel müssen im Falle eines Unfalls nachkommende Fahrzeuge umgehend gewarnt werden, um keinen Auffahrunfall zu verursachen. In diesem Szenario kann es einen großen Unterschied machen, wenn ein Bremsvorgang erst leicht verspätet ausgelöst wird. Je höher die Latenz, desto schädigender kann die Auswirkung sein. Ist die Latenz zu hoch, bedarf es zusätzlich einer speziellen Erkennung. Die Bedeutung von Nachrichten verfällt, wenn Zeit vergeht. Bei zu spätem Erscheinen der Nachricht kann sie sogar einen negativen Effekt haben. Ein Ausweichmanöver darf nur im richtigen Moment geschehen. Wird es ausgeführt nachdem die Situation bereits bereinigt ist, kann es nur vermeidbaren Schaden anrichten. Neben dem Alter der Nachricht ist auch die Korrektheit wichtig. Nachrichten können falschen oder sogar böswilligen Inhalt haben. Auch fehlende Nachrichten können Schaden anrichten, wie etwa die ausbleibende Information über eine unerwartete Straßensperrung, die von einem vorausfahrenden Fahrzeug erkannt wurde [PKÅ+20]. Eine weitere Herausforderung beim Schutz von Connected Cars ist das Zusammenfügen individueller Komponenten. Selbst wenn jede Komponente einen Schutz für sich selbst aufweist, bedeutet dies nicht, dass das Gesamtkonstrukt zusammengefügter Komponenten ebenfalls geschützt ist [Mcc17].

Eine der größten Gefahren bei Connected Cars ist, dass Angriffe auf ein einzelnes Fahrzeug auf weitere überspringen können. Für eine erfolgreiche Zusammenarbeit der Fahrzeuge muss dies verhindert werden [PKÅ+20]. Gleichzeitig sind Angriffe auf individuelle Komponenten kritischer als beim klassischen Fahrzeug. Dadurch, dass jedes Fahrzeug einer Serie dieselben Komponenten verbaut hat, kann ein erfolgreicher Angriff auf ein einzelnes Fahrzeug, leicht die gesamte Flotte befallen [Mcc17]. In einer systematischen Literaturrecherche haben Ram et al. [RMFR18] untersucht welche Bedrohungen und mögliche Lösungen für Connected Cars in der Literatur identifiziert wurden und diese in die folgenden Kategorien unterteilt:

- *Communication Threat* umfasst Angriffe auf die Kommunikation zwischen Fahrzeugen
- *Identity Management* umfasst Angriffe durch Authentifikation und Autorisierung
- *Embedded Security* umfasst Angriffe auf dem Physical- und Data Link Layer des OSI-Modells

Der Großteil der identifizierten Bedrohungen bezieht sich auf die Kommunikation. Diese fallen sehr vielfältig aus und beinhalten beispielsweise Self-propagating Worm Attacks, Malware, unsichere Warnsysteme und unsichere Firmware Updates. Die zu den Attacken gefundenen Lösungen beschäftigen sich hauptsächlich mit der Erkennung und Abmilderung des Problems und werden somit als unzureichend bewertet. Zusätzlich wird herausgestellt, dass die meisten Lösungen nur in passenden Simulationen Erfolg gefunden haben, aber im Kontext der realen Welt nicht effektiv wären. Für ein System das sowohl Security, als auch Safety bieten soll, ist es wichtig die Sicherheit so früh wie möglich einzuplanen. Eine tiefe Integration garantiert mehr Erfolg als Sicherheitslösungen auf ein bestehendes System aufzusetzen [Hus16; Mcc17; RMFR18].

4.4. Komplexität

Eine große Herausforderung von Datenpipelines ist deren Komplexität. Durch Unterteilung komplexer Prozesse in Teilschritte steigt die Anzahl der Komponenten und deren Verbindungen. Dies ermöglicht deren Wiederverwendung, jedoch werden dadurch weitere Verbindungen zwischen zuvor unabhängigen Komponenten geschaffen. Zum Beispiel muss eine Funktion zur Konvertierung von Temperaturwerten zwischen Fahrenheit und Celsius nur einmal implementiert werden. Wird diese Funktion allerdings an vielen Stellen benötigt, beeinträchtigen die benötigten Verbindungen die Modularisierung der Funktionalitäten. Wenn einzelne Komponenten viele Verbindungen haben, reduziert dies die Verständlichkeit des gesamten Systems. Wie zuvor in Abschnitt 2.2 erklärt, ist es nicht ratsam, wenn Komponenten mehr als eine Funktionalität implementieren. Allerdings hat dieses Vorgehen den Nachteil, dass die Anzahl der nötigen Komponenten und Verbindungen steigt und sich somit die Komplexität erhöht [MBO20].

Die Verständlichkeit der Modelle komplexer Datenpipelines muss gewahrt werden. Dennoch muss die Flexibilität der Pipeline erhalten bleiben. Das Hinzufügen, Bearbeiten und Entfernen von Datenquellen und Verarbeitungsschritten muss ohne großen Aufwand möglich sein. Dabei ist die Eigenschaft der Verteiltheit zu berücksichtigen. Da in einem Connected Car eine Vielzahl ECUs eingesetzt werden, gehört zur Modellierung der Datenpipeline die Zuordnung welcher Verarbeitungsschritt auf welcher ECU ausgeführt wird. Erneut erhöht dies die Komplexität und fügt die Notwendigkeit hinzu, diese leicht zu überschauen. Es muss schnell erkennbar sein, wie die Verteilung der Komponenten auf den ECUs ist, um sicherzustellen, dass die benötigten Ressourcen

vorhanden sind. Zur Unterstützung des Managements verschiedener Modelle ist es wichtig, diese importieren und exportieren zu können. Auf diese Weise können existierende Modelle schnell bearbeitet und wieder eingesetzt werden.

4.5. Interoperabilität

Eine hohe Verbreitung von Connected Cars ist wichtig, um von ihnen zu profitieren. Je mehr Fahrzeuge in der Lage sind miteinander zu kommunizieren, desto weiter spannen sich Kommunikationsnetze und desto höher ist der Informationsaustausch. Für dessen Realisierung müssen zum einen die Fahrzeuge mit der notwendigen Technologie ausgestattet sein. Zum anderen müssen die Fahrzeuge in der Lage sein, die empfangenen Daten zu interpretieren. Bei einer hohen Vielfalt an Fahrzeugherstellern ist dies schwierig zu gewährleisten. Globale Standards sind eine Möglichkeit die Kommunikation der Fahrzeuge zu definieren [PKÅ+20]. Jedoch stellt ein Mangel an Standards zurzeit die größte Hürde für die Sicherstellung von Interoperabilität dar [Age15]. Aufgrund der Heterogenität der Fahrzeughersteller erscheint eine Einigung auf Standards ein weit entferntes Ziel. Jeder Hersteller bietet unterschiedliche Services an, verbaut andere Software und hat eigene Vorstellungen davon, wie die Kommunikation gestaltet sein soll. Dazu kommt, dass die Kommunikation mit RSUs ebenfalls berücksichtigt werden muss. Systeme dieser Größe sind sehr dynamisch und deren Komponenten entwickeln sich stetig unabhängig voneinander weiter [RCL14]. Aufgrund dieser Eigenschaften wird eine Interoperabilität erschwert. Dennoch muss während der Evolution solcher Systeme jederzeit eine Zusammenarbeit möglich sein [SPE17].

Zur Interoperabilität gehört die Definition von Schnittstellen, um den Austausch von Daten zu ermöglichen [NLF+15]. Durch diese Schnittstellen werden Daten übertragen. Jedoch ist dies nicht ausreichend, um von den Empfängern interpretiert werden zu können. Zusätzlich muss ein gemeinsames Verständnis für das Format und den Inhalt ausgetauschter Daten etabliert werden [AV10]. Dabei behandelt die *syntaktische Interoperabilität* das Format einer Nachricht. Diese bezieht sich auf die Kodierung und die Syntax der Nachricht. Hingegen definiert *semantische Interoperabilität* das gemeinsame Verständnis des Inhalts und wie die Daten zu interpretieren sind [RCL14]. Nur wenn beide Arten der Interoperabilität gegeben sind, kann die Zusammenarbeit der Connected Cars funktionieren.

4.6. Datenverarbeitung

Für die Verarbeitung gesammelter Daten müssen verschiedene Anforderungen beachtet werden. Wie zuvor beschrieben, ist für viele Funktionen eine Verarbeitung in Echtzeit wichtig. Dazu kommt, dass einige Verarbeitungsschritte einen hohen Ressourcenverbrauch haben, wie z.B. das Verarbeiten von Videomaterial. Eine elastische Bereitstellung von Ressourcen ist mit Cloud-basierten Lösungen möglich. Um von diesen zu profitieren, werden gesammelte Daten in die Cloud hochgeladen, anstatt im Fahrzeug verarbeitet zu werden. Das Ergebnis der Berechnung wird zurück zum Fahrzeug übertragen. Bei dieser serverseitigen Lösung wären lediglich Client-Anwendungen im Fahrzeug nötig. Aufgrund der Elastizität der Cloud ist der Ressourcenverbrauch keine Herausforderung. Allerdings bringt das Streaming der Daten Probleme mit sich. Bei der Übertragung der Daten zwischen Fahrzeug und Cloud können Latenzen entstehen, welche verhindern Entscheidungen

in Echtzeit zu treffen [BIS+18; RBP19]. Oftmals ist es zeitlicher, die Daten nahe an dem Ort zu verarbeiten, an dem das Ergebnis eingesetzt wird [KAEM13]. Eine Verarbeitung auf den ECUs innerhalb des Fahrzeuges hätte somit die geringste Latenz. Dennoch kann je nach Anwendungsfall eine Verarbeitung in der Cloud sinnvoll sein, da Echtzeitverarbeitung nicht immer nötig ist und somit Rechenressourcen im Fahrzeug gespart werden können.

Ein Kompromiss aus geringer Latenz und einem größeren Ressourcenangebot bietet das Edge-Computing [MCC+16]. Bei diesem werden Daten am topologischen Rand des Netzwerkes verarbeitet. RSUs können als nutzbare Ressourcen angeboten werden, sodass Verarbeitungen auf diesen stattfinden können. Aufgrund der Nähe zum Fahrzeug sind, im Gegensatz zur Cloud, nicht mehrere Hops nötig, um die Ressourcen zu erreichen. Dadurch ist der Effekt auf die Latenz weniger negativ. Gleichzeitig werden Ressourcen zur Verarbeitung verfügbar, auch wenn diese nicht die Elastizität der Cloud haben. Ein Nachteil des Edge-Computings ist die Mobilität der Fahrzeuge. Die Edge-Nodes sind nur für geringe Zeit in Reichweite der Fahrzeuge, wodurch diese für die Verarbeitung langwieriger Prozesse ungeeignet sind. Wie zuvor zeigt sich, dass der optimale Ort für die Verarbeitung von Daten abhängig vom Anwendungsfall ist. Ein zusätzlicher Faktor, den es zu berücksichtigen gilt, ist die potenzielle Einschränkung zum Schutz der Privatsphäre. Abhängig der zu verarbeitenden Daten kann es vorkommen, dass diese das Fahrzeug nicht verlassen dürfen. Sollte dies geschehen, verliert der Inhaber der Daten die Kontrolle über die weitere Verarbeitung. Der Schutz der Privatsphäre kann somit nicht garantiert werden.

Eine weitere Anforderung an die Datenverarbeitung ist die Wahl der Verbindung zwischen den Verarbeitungsschritten. Kriterien, wie Verarbeitung in Batches oder Streams, die Latenz und die Menge an Daten haben Einfluss auf diese Wahl. Zusammenfassend ist eine präzise Konfiguration der Datenverarbeitung wichtig, um verschiedenen Anforderungen gerecht zu werden.

5. Konzeptionierung

In diesem Kapitel wird das entwickelte Konzept zur Ermöglichung von verteilten Datenpipelines in Connected Cars vorgestellt. Dabei wird sich an die in Kapitel 4 vorgestellten Anforderungen orientiert. Für diese werden Möglichkeiten zur Bewältigung aufgezeigt. Teil des Konzeptes ist die Vorgehensweise zur Modellierung von Datenpipelines. Daher wird die Vision eines Modellierungswerkzeuges und dessen Funktionalitäten vorgestellt. Die konkrete Implementierung dieser Vision wird in Kapitel 6 anhand eines Prototypen vorgestellt. Fokus dieser Arbeit ist die Modellierung des Datenflusses, weshalb auf den Eintritt in Datensenken nicht gesondert eingegangen wird. Deshalb werden diese aus Sicht der Modellierung als ein weiterer Verarbeitungsschritt betrachtet. Datenquellen hingegen werden separat behandelt.

Zu Beginn dieses Kapitels wird in Abschnitt 5.1 auf die Modellierung in ihren Grundzügen eingegangen. Darauf folgen in Abschnitt 5.2 Konzepte, die auf die zuvor vorgestellte Modellierung aufbauen und genauer die Funktionen des Modellierungswerkzeuges erklären. Zum Schluss wird in Abschnitt 5.3 auf den Lebenszyklus eines Modells, bestehend aus Deployment und Wartung eingegangen.

5.1. Modellierung

Datenpipelines können auf vielfältige Weise modelliert werden. Bei Connected Cars handelt es sich um einen komplexen Anwendungsfall mit vielen Datenquellen und Datensenken. Die Vision dieser Arbeit ist es daher ein Konzept für ein Modellierungswerkzeug zu entwickeln, das diese Komplexität auflösen kann. Da sich für die Systemarchitektur eines Connected Cars das Pipes-and-Filters-Konzept [AZ05] empfiehlt, eignet sich ein Modellierungsstil, der daran angelehnt ist. Vor allem eine leichte Verständlichkeit und Bedienung ist für die Realisierung der Vision wichtig. Auf Basis dessen ist eine flussdiagramm-orientierte Modellierung über eine graphische Benutzeroberfläche sinnvoll. Die Visualisierung ermöglicht einen schnellen Überblick und ein leichtes Verständnis über komplexe Zusammenhänge. Sie erlaubt zusätzlich den Einsatz von Instrumenten, wie Formen und Farben. Mit diesen lassen sich Informationen codieren und zugänglich in das Gesamtmodell integrieren. Indem die Notwendigkeit Informationen durch Lesen aufzunehmen auf ein Minimum reduziert wird, können diese effektiver aufgenommen werden. Ein Flussdiagramm stellt den tatsächlichen Datenfluss in einem Connected Car anschaulich dar. Die Filter des Pipes-and-Filters-Konzeptes sind im Konzept dieser Arbeit als *Datenquellen* und *Filter* umgesetzt. Diese werden im Folgenden zusammenfassend als *Komponenten* bezeichnet. Datenquellen generieren Daten und Filter verarbeiten diese. Da Datensenken nicht zum Rahmen dieser Arbeit gehören, werden diese für die Modellierung nicht berücksichtigt. Die Funktion der Pipes ist der gerichtete Datenaustausch zwischen Datenquellen und Filtern. Datenquellen können Nachrichten an Filter senden und diese wiederum an weitere Filter. Wie dies aussieht ist in Abbildung 5.1 anhand des in dieser Arbeit entwickelten Modellierungswerkzeuges demonstriert.

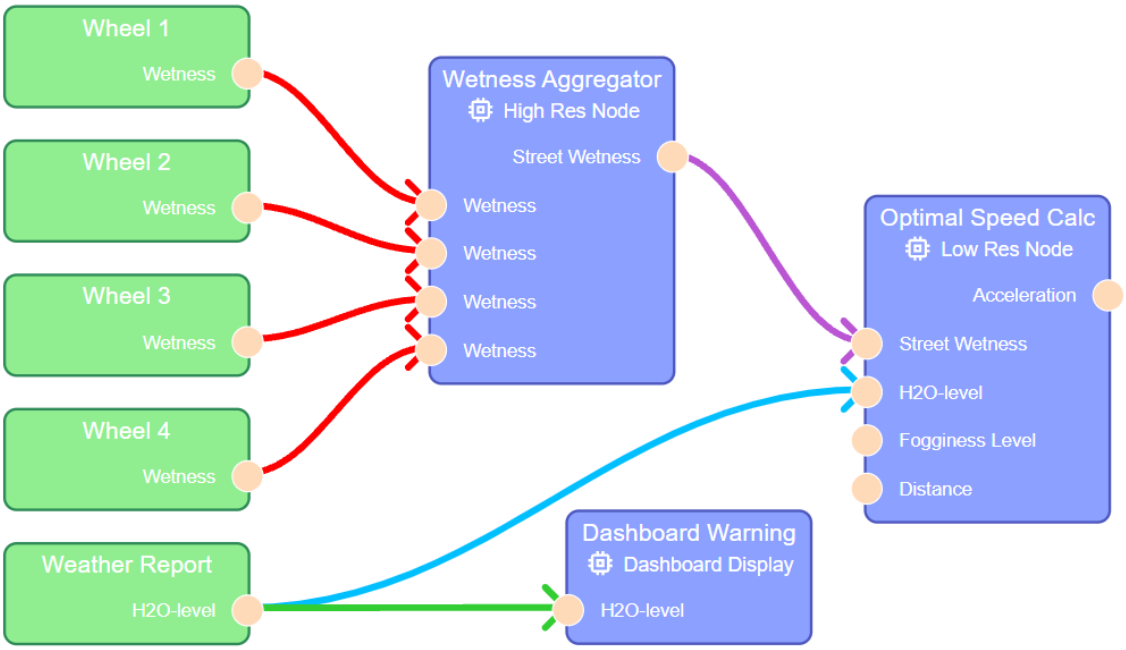


Abbildung 5.1.: Prototyp des auf Pipes-and-Filters basierenden Konzeptes mit Datenquellen, Filtern und Verbindungen.

Dieser Prototyp wird in Kapitel 6 im Detail ausgeführt. Zu erkennen ist das Pipes-and-Filters-Konzept mit Datenquellen, Filtern und deren Verbindungen. Im Rest dieses Abschnitts werden die entwickelten Konzepte bezüglich der allgemeinen Modellierung ausgeführt.

Fokus auf Anwendungen Wie in Abschnitt 2.1.1 aufgezeigt, besteht ein Connected Car nicht aus einer einzelnen Recheneinheit. Dieser Faktor muss ebenfalls modelliert werden. Der Einsatz vieler ECUs mit unterschiedlichen Ressourcen ist von zentraler Bedeutung. Für die Modellierung werden diese im Folgenden als *Knoten* bezeichnet. Auf jedem Knoten können mehrere Anwendungen betrieben werden, die potenziell mit Anwendungen anderer Knoten kommunizieren müssen. Bei Big-Data-Datenflüssen ist es wichtig diese Kommunikation zu berücksichtigen, um eine effiziente Auslastung der Knoten zu realisieren. Dies wirft die Frage auf, ob bei der Modellierung die Knoten oder die darauf laufenden Anwendungen als Filter gewählt werden sollen. Bei der Wahl von Knoten als Filter liegt der Fokus auf der Verteilung der physischen Ressourcen. Die Teilschritte zur Erfüllung des Geschäftsziels, d.h. die Funktionalität des Connected Cars, würden in den Hintergrund geraten und deren Darstellung komplexer werden. Liegt der Fokus allerdings auf den Anwendungen, bietet dies eine übersichtliche Veranschaulichung der komplexen Verarbeitungsschritte. Da hier die Hauptkomplexität liegt, lohnt es sich diese Darstellung zu wählen und über die im weiteren Verlauf dieses Kapitels erwähnten Konzepte zu vereinfachen. Die Übersicht über die Knoten kann dennoch mit den in Abschnitt 5.2 vorgestellten Konzepten bewahrt werden.

Aufbau eines Filters Filter haben beliebig viele Eingänge und Ausgänge. Im Folgenden werden diese zusammenfassend als *Schnittstelle* bezeichnet. Eine Schnittstelle ist ein Kommunikationspunkt der darunterliegenden Anwendung. Je nach Verbindungsart kann dies z.B. ein Topic bei

Publish/Subscribe oder eine Netzwerkadresse sein. Sofern die Verbindungsart es zulässt, kann eine einzelne Schnittstelle beliebig viele eingehende bzw. ausgehende Verbindungen unterstützen. Auf diese Weise muss ein Filter nicht pro Kommunikationspartner eine neue Schnittstelle bereitstellen. Dadurch wird die Modellierung übersichtlicher und in der darunterliegenden Anwendung kann Code wiederverwendet werden.

Aufbau einer Datenquelle Eine Datenquelle kann vielfältige Bedeutung haben. Es kann sich etwa um einen verbauten Sensor handeln, oder auch eine Quelle außerhalb des Fahrzeuges sein. Dazu zählen z.B. Wetterdienste. Von anderen Connected Cars übertragene Daten stellen für das eigene Fahrzeug ebenfalls eine Datenquelle dar. Als Teil des Modellierungskonzeptes sind Datenquellen ähnlich wie Filter. Datenquellen haben allerdings nur exakt einen Ausgang und keine Eingänge. Identisch zu den Filtern reicht es aus eine einzelne Datenquelle zu modellieren, selbst wenn diese mehrere Filter mit Daten versorgen soll. Einzelne Sensoren, die mehrere Daten erfassen, werden als getrennte Datenquellen modelliert, da es unerheblich ist, ob Daten von demselben Sensor oder von unterschiedlichen Sensoren gemessen werden. Dies ermöglicht eine bessere Trennung zusammenhangsloser Daten und eine übersichtlichere Modellierung.

Templates & Instanzen Um Wiederverwendbarkeit der Komponenten zu gewährleisten sind diese in Templates und Instanzen aufgeteilt. Diese gibt es sowohl für Datenquellen als auch Filter. Templates definieren, wie Instanzen aufgebaut sind. Dazu zählt unter anderem, wie viele Eingänge und Ausgänge es gibt. Aus Templates können beliebig viele Instanzen erzeugt werden. Die in den Templates definierten Informationen gelten für jede Instanz. Bei der Erzeugung von Instanzen können weitere Informationen gespeichert werden, die nur für sie gültig sind. Auf diese Weise müssen Sensoren oder Anwendungen, die mehrfach vorhanden sind, nur einmal definiert werden. Dies sorgt für leichtere Verwaltung und Konsistenz zwischen den Instanzen. Zum Beispiel muss der Aufwand für die Modellierung der Räder eines Fahrzeuges nicht mehrfach betrieben werden. Zusätzlich kann leichter mit Redundanz, die vor allem in Sicherheitskritischen Systemen wichtig ist, gearbeitet werden.

Modellierung von Knoten Knoten sind die Repräsentation von Recheneinheiten wie z.B. ECUs, RSUs und Ressourcen der Cloud. Diese benötigen keine Templates und sind nicht Teil des Flussdiagramms. Filter können Knoten zugewiesen werden, um sicherzustellen, dass die benötigten Ressourcen vorhanden sind. Zusätzlich ist die Modellierung der Zuweisungen von Filter und Knoten für das in Abschnitt 5.3.1 beschriebene Konzept des Deployments wichtig. Beispielsweise haben Filter für die Analyse des aufgezeichneten Videomaterials der Außenkameras einen außergewöhnlichen Anspruch an Ressourcen und können nur auf ganz bestimmten Knoten, die ausreichende Ressourcen zur Verfügung stellen, betrieben werden. Filter für die Verarbeitung von Temperaturdaten hingegen haben andere Ansprüche und können auch auf leistungsschwächeren Knoten arbeiten. Datenquellen können keinen Knoten zugewiesen werden, da diese lediglich aufgezeichnete Daten aussenden. Für diesen Zweck sind Datenquellen sozusagen ihre eigenen Knoten und bedürfen daher keiner speziellen Modellierung.

5.2. Konzepte

Im Folgenden werden Konzepte vorgestellt, die es erleichtern verteilte Datenpipelines im Anwendungsfall der Connected Cars zu modellieren. Die Konzepte unterstützen die Modellierung, um die in Kapitel 4 vorgestellten Anforderungen zu bewältigen. Diese Konzepte können direkt oder indirekt umsetzbar sein. Eine direkte Umsetzung ist ein Feature oder eine Konfiguration, die das Modellierungswerkzeug zur Verfügung stellt. Eine indirekte Umsetzung ist ein Konzept, wie etwas modelliert werden kann, und bedarf keiner zusätzlichen Einstellung.

Verschleierungsfilter Private Daten gilt es zu schützen und nur offen zu zeigen, wenn es notwendig ist. In Abschnitt 4.1 wurde gezeigt, dass die Akzeptanz für die Offenlegung privater Daten unter anderem vom Verwendungszweck und der inbegriffenen Angebrachtheit abhängt. Für einige Daten gibt es mehrere Verwendungszwecke. Abhängig von diesen ist es unterschiedlich stark angebracht die Daten zu veröffentlichen, auch wenn deren Inhalt derselbe ist. Deshalb ist es notwendig für jeden Anwendungsfall dieselbe Datenmenge auf unterschiedliche Weisen zu behandeln, bevor diese geteilt wird. Zum Beispiel werden Standortdaten auf vielfältige Weise verwendet. Wenn sie dazu genutzt werden, umliegende Fahrzeuge über die eigene Position zu informieren, müssen diese präzise sein, um Unfälle zu vermeiden. Geht es darum, dass ein Verkehrsservice Fahrzeugpositionen zur Erkennung von Staus sammelt, müssen diese nicht präzise sein. In diesem Fall können zum Schutz der Privatsphäre die Koordinaten verschleiert werden. Ein Verschleierungsfilter kann, über eine eingehende Verbindung, die tatsächlichen Koordinaten empfangen und über eine Funktion so verändern, dass eine Position innerhalb eines bestimmten Radius um die tatsächliche Position wiedergegeben wird. Die neue Position ist nicht auf ein Fahrzeug zurückzuführen und kann in dieser Form an den Verkehrsservice versendet werden. Der Nutzen des Dienstes wird nicht behindert, da die Erkennung von Staus immer noch möglich ist. Die genaue Verschleierungsfunktion ist von der darunterliegenden Anwendung implementiert. Diese Filter können beliebig wiederverwendet werden.

Deployment Tagging Für jede Anwendung muss die richtige Umgebung auf den ECUs vorhanden sein. Dies kann während der Modellierung sichergestellt werden. Jedem Knoten und Filter können Tags zugewiesen werden, welche die Beschaffenheit und Anforderungen ausweisen. Dazu zählen z.B. physische Ressourcen und installierte Laufzeitumgebungen. Zusätzlich können über das Tagging-System Anforderungen zur Privatsphäre, Sicherheit o.ä. festgehalten werden. So kann unter anderem für jeden Filter spezifiziert werden, ob er auf einem privaten Knoten innerhalb des Fahrzeuges gehostet werden muss, oder ob auch ein Deployment auf RSUs oder in der Cloud zulässig ist. Dieselbe Zuweisung findet für Knoten statt. Während der Zuweisung von Filtern zu Knoten kann überprüft werden, ob alle Bedingungen eines Filters erfüllt sind. Damit sind zusätzlich die Voraussetzungen erfüllt, um das Deployment zu automatisieren und einem Programm die Entscheidung über die richtige Umgebung für einen Filter zu überlassen.

Codierungsfilter Codierungsfilter umfassen Filter für die Verschlüsselung und Entschlüsselung. Bei diesen handelt es sich um designierte Filter, deren einzige Aufgabe es ist, sich um die Codierung von Nachrichten zu kümmern. Diese sollten vor allem an eingehenden und ausgehenden Verbindungen des Fahrzeuges platziert werden, um das interne Netzwerk vor der Außenwelt abzusichern.

Indem Codierungen explizit modelliert werden, kann diese Aufgabe von allen anderen Filtern und somit der Implementierung der Anwendungen entfernt werden. Zusätzlich ist ein Vorteil einer expliziten Modellierung, dass auf Codierungsfilter verzichtet werden kann, ohne die Anwendungen ändern zu müssen. Viele Nachrichten benötigen keine Codierung und können somit lesbar sein, ohne dass Schaden entsteht. Wenn dies in zeitkritischen Anwendungsfällen der Fall ist, kann eine Verschlüsselung und Entschlüsselung wichtige Millisekunden bedeuten. Im Falle eines Unfalls müssen nachkommende Fahrzeuge sofort benachrichtigt werden und es gibt keine Notwendigkeit eine solche Nachricht geheim zu halten. Einen Verzicht auf Codierungsfilter modellieren zu können, ist für eine solche Situation wichtig und kann ein zu spätes Bremsmanöver verhindern.

Interne Verschlüsselung Codierungsfilter sind eine Möglichkeit eine gesicherte Kommunikation mit externen Kommunikationspartnern sicherzustellen. Im fahrzeuginternen Netzwerk ist eine sichere Verbindung damit nicht gewährleistet. Die Verbindungen zwischen Filtern sind weiterhin ungeschützt. Selbst bei der Nutzung von Codierungsfiltern ist dieser Schutz nicht auf deren eingehenden Verbindungen anwendbar. Das interne Netzwerk benötigt daher einen separaten Schutzmechanismus. Wie zuvor gilt jedoch, dass manche Verbindungen zeitkritisch sind und keine Geheimhaltung der Nachrichten benötigen. Daher ist es notwendig für jede Verbindung des Modells individuell einstellen zu können, ob diese gesichert werden müssen.

Validierungsfilter Connected Cars sind auf Daten voneinander angewiesen, um zuverlässig funktionieren zu können. Dabei sind ihre eigenen Fahrentscheidung von anderen abhängig. Das birgt ein Risiko, da nicht davon ausgegangen werden kann, dass die empfangenen Daten korrekt sind. Ursachen sind z.B. Angreifer, die bewusst böswillige Daten aussenden. Gleichzeitig können auch fehlerhafte Messungen anderer Fahrzeuge unbeabsichtigt für falsche Daten sorgen. Um dies auszugleichen, ist es wichtig auf eine übereinstimmende Mehrheit zu hören. Durch das VANET ist bekannt, welche Fahrzeuge sich an welchem Ort befinden. Daraus kann abgeleitet werden, welche Fahrzeuge in der Lage sein sollten, stattfindende Ereignisse zu messen. Wird z.B. eine Nachricht über Gegenstände auf der Fahrbahn empfangen, ist es wichtig auf Bestätigung umliegender Fahrzeuge zu achten, die in der Lage sein sollten dies ebenfalls zu vernehmen. Vermelden diese allerdings keine Ereignisse, ist dies ein Indiz für eine Falschmeldung. Die Logik hinter einer solchen Validierung ist komplex und hängt von vielen Faktoren ab, wie etwa der Art des Ereignisses und die Wahrnehmbarkeit umliegender Fahrzeuge. Wie diese Logik aussieht liegt daher außerhalb des Rahmens dieser Arbeit. Stattdessen ist wichtig, wie die Validierung modelliert werden kann. Für diese können zustandsspeichernde Validierungsfilter eingesetzt werden. Diese Filter können für eingehende Nachrichten prüfen, ob eine Validierung notwendig ist. Ist dies der Fall, kann über die implementierte Validierungsfunktion und ggf. weiteren eingehenden Nachrichten das Ereignis validiert und darauf reagiert werden. Dabei muss beachtet werden, dass andere Fahrzeuge das Ereignis selbst wahrgenommen haben müssen. Auf diese Weise kann zwischen mehreren unabhängigen Meldungen und einer vielfältig propagierten Falschmeldung einer einzelnen Quelle differenziert werden.

Individuelle Verbindungstechnologien Wie zuvor erläutert gibt es in Connected Cars ECUs mit unterschiedlichen Ressourcen, um die vielfältigen Anforderungen der Anwendungen zu unterstützen. Allerdings gibt es solche Anforderungen auch für die Verbindungen. Diese müssen auf verschiedene Kriterien zugeschnitten sein, wie etwa die Größe und Menge der Daten, sowie stream- oder

batchbasierte Verarbeitungsweise der Filter. Zur Erfüllung dieser Kriterien, ist die Wahl der grundlegenden Verbindungstechnologie wichtig. Für leichtgewichtige Sensordaten, die potenziell viele Empfänger haben ist z.B. eine Publish/Subscribe-basierte Technologie, wie MQTT [BBBG19] geeignet. Bei der Übertragung von Live-Videomaterial zur Analyse des aufgezeichneten Materials sollte eine andere Technologie, wie z.B. Real-Time Transport Protocol (RTP) [SCFJ03] verwendet werden. Daher ist es wichtig, diese Einstellung modellieren zu können. Das Konzept ist anwendbar auf weitere Schichten des OSI-Modells, um z.B. zwischen TCP und UDP auswählen zu können. Bestmöglich kann dies für jede Verbindung individuell eingestellt werden, um eine flexible Zusammenstellung der Filter zu ermöglichen, ohne sich ungeeigneter Technologie anpassen zu müssen.

Priorisierung von Schnittstellen Für viele, vor allem sicherheitskritischen Anwendungen, muss es möglich sein, zwischen wichtigen und unwichtigen Ereignissen zu unterscheiden. In dringenden Situationen müssen Nachrichten, die zeitkritisch sind, schneller verarbeitet werden als andere. Selbst wenn Nachrichten als zeitkritisch markiert sind, kann eine priorisierte Behandlung nicht garantiert werden. Es muss davon ausgegangen werden, dass Nachrichten, z.B. aufgrund eines Rückstaus in einer Message Queue, nicht rechtzeitig gelesen werden können. Existieren dedizierte Verbindungen für zeitkritische Ereignisse, können diese genutzt werden, um Nachrichten sicherer zuzustellen. Daher ist es wichtig, im Modell bestimmte Eingänge für Filter priorisieren zu können. Mit dieser Information können Filter Nachrichten dieser Verbindung bevorzugt behandeln. Es kann ebenfalls von Vorteil sein, Ausgänge zu priorisieren. Verbindungen können unterschiedliche Eigenschaften haben, wie etwa die genutzte Verbindungstechnologie. Auf diese Weise lässt sich für Filter definieren, dass diese z.B. den Ausgang über 5G bevorzugt gegenüber dem Ausgang über 4G behandeln sollen. Weiterhin lässt sich bestimmen, welcher Filter bevorzugt als nächster Verarbeitungsschritt ausgewählt werden soll. Geht dieser Filter offline, kann auf eine niedriger priorisierte Alternative zurückgefallen werden.

Obsolete Nachrichten abweisen Vor allem im Automobilbereich ist es wichtig, dass Nachrichten zeitnah über Ereignisse berichten. Der Zustand der Umgebung kann sich aufgrund der hohen Mobilität schnell ändern. Nachrichten, die mit zu großer Latenz ankommen, können daher häufig irreführend sein. Autonome Fahrzeuge könnten durch obsolete Nachrichten zu Aktionen geführt werden, die in der neuen Umgebung Schaden verursachen. Daher ist es bei eingehenden Nachrichten wichtig, die Aktualität des Inhalts zu überprüfen. Wenn Informationen mehrere Filter passieren, handelt es sich bei jeder Verbindung um eine neue Nachricht. Aufgrund dessen ist nicht das Alter der Nachricht selbst, sondern das Alter des Ereignisses, über das sie berichtet, von Bedeutung. Korrekt ist die Wahl des Zeitpunktes, an dem die Sensoren das Ereignis wahrgenommen haben. Da Filter Informationen aus verschiedenen eingehenden Quellen zusammensetzen können, muss für jeden Eingang individuell modellierbar sein, ab wann eine Nachricht obsolet ist. Je nach erwarteter Nachricht, kann dies unterschiedlich ausfallen. Zum Beispiel haben Nachrichten, die den Abstand zu benachbarten Fahrzeugen berichten, ein schnelles Verfallsdatum. Hingegen entsprechen Nachrichten, die über Wetterverhältnisse informieren, lange der Realität.

Verbindungsqualität Die Verbindungsqualität definiert die Anforderungen an die Verbindung. Verbindungen, die einen hohen Qualitätsanspruch haben, benötigen zusätzlichen Aufwand, um diesem gerecht zu werden. Allerdings ist dies in vielen Anwendungsfällen nicht nötig und belastet

stattdessen die Verbindung. Wie eine Qualitätseinstellung aussehen kann, hängt von der Art der verwendeten Technologie ab. Beispielsweise lässt sich bei einem QoS zwischen *Exactly once*, *At least once* und *At most once* unterscheiden. Zeichnet ein Sensor mehrmals pro Sekunde Werte auf, z.B. die Temperatur des Motors, kann es unerheblich sein, ob einzelne Werte fehlen oder Duplikate empfangen werden. Kann hingegen bei einem Filter jede einzelne eingehende Nachricht einen Unterschied machen, muss die Verbindungsqualität entsprechend eingestellt sein. Neben dem QoS sind weitere Qualitätskonzepte denkbar. Solch eine Einstellung sollte pro Verbindung modelliert werden können.

Containerisierung Mit einer bereits sehr hohen und immer weiter steigenden Anzahl an Sensoren in Connected Cars steigt ebenfalls die Komplexität der Modellierung. Für jede Datenquelle kann es mehrere Filter zur Verarbeitung der Daten geben. Diese können komplex miteinander verwoben sein, da jeder Filter beliebig viele Abhängigkeiten zu anderen Filtern haben kann. Mit dem Fortschritt der Technologie werden Fahrzeuge mit immer mehr Sensoren und Features ausgestattet. Einen Überblick über das Modell zu bewahren, wird mit jedem Filter erschwert. Dennoch muss es möglich sein, komplexe Modelle zu erzeugen, zu warten und zu erweitern, um ein zukunftsfähiges Produkt zu gewährleisten. Ein Verfahren zum Verbergen von Komplexität ist die Containerisierung. Ein Container erscheint wie ein regulärer Filter mit Eingängen und Ausgängen. Hinter diesem verbirgt sich jedoch keine Anwendung und er wird nicht auf einem Knoten gehostet. Andere Eigenschaften eines Filters, z.B. die zuvor beschriebene Priorisierung von Schnittstellen, werden ihm ebenfalls nicht zugeschrieben. Container dienen dazu Teile des Modells auf einer höheren Abstraktionsebene zusammenzufassen. Ein Container kann geöffnet werden, um eine Abstraktionsebene tiefer zu gehen und den von ihm verborgenen Teil des Modells offenzulegen. Zum Beispiel kann es auf der obersten Ebene des Modells einen Container für die Funktion des Spurenhalteassistenten geben. Dieser Service ist als einzelner Filter auf dieser Ebene dargestellt. Wird der Container geöffnet, zeigen sich die Teilschritte bestehend aus Datenquellen und Filtern, deren Komposition den Spurenhalteassistenten ermöglichen. Durch diese Abstraktion ist es möglich, die Teilschritte vom Rest des Modells zu verbergen. Die Eingänge und Ausgänge des Containers auf der höheren Ebene sind auf die niedrigere Ebene repliziert. Dadurch können Verbindungen zwischen Filtern unterschiedlicher Ebenen erzeugt werden. Datenquellen können ebenfalls innerhalb von Containern modelliert werden, wodurch es möglich ist, vollständige Datenflüsse von Quelle bis Senke innerhalb eines Containers zu abstrahieren. Durch diese Art der Strukturierung ist es möglich, das große Gesamtmodell in verschiedene kleinere Ansichten zu unterteilen. Da es sich lediglich um andere Ansichten handelt, verliert die Modellierung trotz Einsatz von Containerisierung nicht an Mächtigkeit. Die Containerisierung kann rekursiv unbegrenzt fortgeführt werden.

Abhängigkeiten hervorheben Trotz Containerisierung kann es zu komplexen Modellen derselben Abstraktionsebene kommen, wenn diese sich nicht logisch unterteilen lässt. Um innerhalb derselben Ebene eine bessere Übersicht zu erhalten, können Abhängigkeiten von Datenquellen und Filtern hervorgehoben werden. Abhängigkeiten gibt es in Richtung des Datenflusses und entgegen dieser. Die Abhängigkeiten eines Filters F in Flussrichtung beinhalten alle Filter, auf die eine ausgehende Nachricht von F über alle Ausgänge transitiv Einfluss haben kann. Dies ist ebenfalls auf Datenquellen anwendbar. Abhängigkeiten entgegen der Flussrichtung beinhalten alle Datenquellen und Filter, deren Nachrichten transitiv Auswirkung auf F haben können. In Abbildung 5.2 sind diese Abhängigkeiten beispielhaft demonstriert. F hat Abhängigkeiten in Flussrichtung mit E und entgegen dieser mit

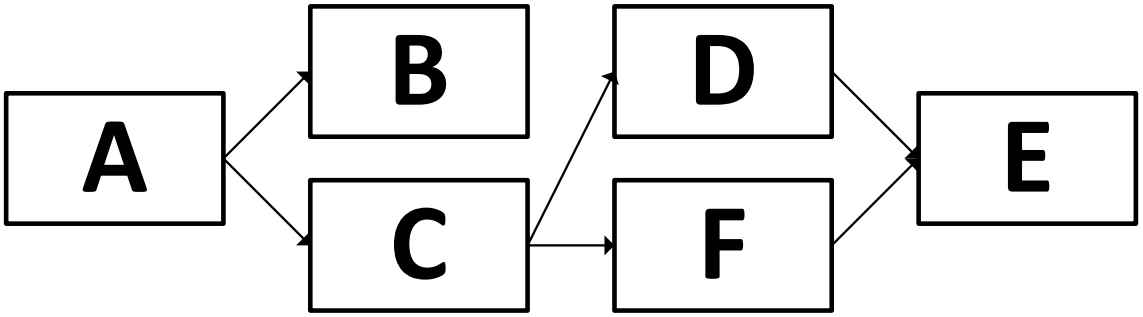


Abbildung 5.2.: Abhängigkeiten von Filtern. F hat Abhängigkeiten mit A, C und E.

A und C. Filter B und D sind demnach nicht von Relevanz für F. Soll ein Filter hervorgehoben werden, um eine bessere Übersicht über dessen Datenfluss zu erzeugen, würden irrelevante Filter ausgeblendet werden. Zu beachten ist, dass dies nur eine Ansicht ist und ausgeblendete Filter weiterhin Teil des Modells sind. Verbindungen zwischen Filtern werden nur modelliert, wenn sowohl Startfilter als auch Zielfilter nicht ausgeblendet sind.

Filter kategorisieren & hervorheben Wie zuvor erwähnt, kann Komplexität durch das Ausblenden irrelevanter Filter aufgelöst werden. Mehr Gestaltungsmöglichkeiten gibt es durch das Kategorisieren von Filtern. Durch das Zuweisen von Kategorien zu Filtern können diese beliebig gruppiert werden. Auf diese Weise kann z.B. nach Funktionalitäten getrennt werden. Durch Einblenden aller Filter, die einer bestimmten Kategorie angehören, wird die Übersicht reduziert. Jegliche Komplexität von Filtern, die nicht mit dieser Kategorie zusammenhängen, ist somit aufgelöst.

Modelle importieren & exportieren Management und Deployment verschiedener Modelle kann durch eine Import- und Exportfunktion möglich gemacht werden. Importierte Modelle werden in das Modellierungswerkzeug geladen, sodass Änderungen vorgenommen werden können. Bei einem Import kann entschieden werden, ob das importierte Modell das bereits existierende Modell überschreiben oder diesem integriert werden soll. Durch eine Integration kann ein Gesamtmodell aus mehreren Teilen zusammengesetzt werden. Auf diese Weise ist eine Trennung von Verantwortlichkeiten möglich und die Modellierung kann besser auf verschiedene Teams aufgeteilt werden. Exportierte Modelle können für das Deployment in einem Connected Car verwendet werden. Wie dies aussieht ist genauer in Abschnitt 5.3.1 beschrieben.

Übersicht aller Knoten Bei Knoten handelt es sich um ECUs. Diese sind in ihren Ressourcen limitiert und können nur eine begrenzte Menge an Filtern unterstützen. Ziel der Modellierung ist es, diese Grenze nicht zu überschreiten und somit ein lauffähiges Modell zu garantieren. Viele Faktoren bestimmen über den Ressourcenverbrauch von Filtern, wozu z.B. der eingehende Datenverkehr zählt. Bei diesem kann es allerdings von Laufzeitbedingungen abhängig sein, wie die Größe und Frequenz des Datenempfangs aussieht. Mit einer großen Menge an Filtern, die über ein komplexes Modell verteilt sind, ist es umso unvorhersehbarer wie hoch die potenzielle Belastung pro Knoten in diesem Modell ausfällt. Um den Modellierern eine bessere Einschätzung zu ermöglichen, ist eine Übersicht über alle Knoten und den darauf laufenden Filtern notwendig.

Referenzen modellieren In großen, mit Containern verschachtelten Modellen ist die Wiederverwendung von Komponenten erschwert. Ein Filter, der für viele Zwecke verwendet wird, hat eingehende- und ausgehende Verbindungen, die sich über das gesamte Modell erstrecken. Die Lesbarkeit des Modells ist somit stark reduziert. Zusätzlich können Datenquellen, welche mehrere Filter unterschiedlicher Abstraktionsebenen mit Daten versorgen, nicht sinnvoll mittels Containern abstrahiert werden. Mit dem Einsatz von Templates ist es möglich, mehrere Instanzen gleich aufgebauter Datenquellen und Filter zu erzeugen. Diese Art der Wiederverwendung erleichtert die Modellierung. Für jede modellierte Datenquelle muss jedoch ein zugehöriger Sensor oder sonstige Quelle existieren. Dieselbe Zusammengehörigkeit gibt es mit Filtern und den repräsentierten Anwendungen. Durch diese Abhängigkeit ist eine freie Wiederverwendung eingeschränkt. Stattdessen ist es sinnvoll, mehrere Datenquellen und Filter anlegen zu können, welche dieselbe physische Entität referenzieren. Mittels dieser Modellierung ist es möglich, jede Komponente an dem Punkt zu modellieren, an dem sie gebraucht wird. Häufig verwendete Filter benötigen so keine Verbindungen, die sich in verschiedene Sektionen des Modells erstrecken. Stattdessen kann jede Sektion eine Referenz des Filters erzeugen, ohne dass im Hintergrund weitere Anwendungen gestartet werden müssen. Gleiches ist anwendbar auf Datenquellen, die somit wiederverwendet werden können, obwohl es nur einen physischen Sensor gibt.

Typisierte Verbindungen Zur Gewährleistung der funktionierenden Zusammenarbeit zweier Komponenten ist jeder Schnittstelle ein Datentyp zugewiesen. Komponenten definieren an ihren Ausgängen den Typ, den die Nachrichten besitzen, welche die Anwendung für diese Verbindung produziert. An den Eingängen werden die von den eintreffenden Nachrichten erwarteten Typen festgelegt. Diese Informationen werden miteinander abgeglichen, um festzustellen, ob eine Verbindung über die betreffenden Schnittstellen valide ist. Verbindungen zweier nicht übereinstimmender Datentypen werden vom Modellierungswerkzeug abgewiesen. Somit werden Verbindungen effektiv zu *Datatype Channels* [HW04]. Die Anwendungen, welche durch die Filter widergespiegelt werden, können auf diese Weise davon ausgehen, dass sie nur Nachrichten vom korrektem Typ empfangen. Die aussendenden und empfangenden Komponenten benötigen ein einheitliches Verständnis der Datentypen, damit das Konzept funktioniert. Auf diese Weise wird, ebenfalls zwischen verschiedenen Herstellern, eine Übereinkunft über das Format von Nachrichten erzwungen.

Transformationsfilter Für Connected Cars ist Zusammenarbeit essenziell. Dazu zählen ebenfalls Fahrzeuge anderer Hersteller. Diese haben andere Sensoren und Anwendungen zur Datenverarbeitung. Mit einer großen Vielfalt an Herstellern und verwendeten Technologien ist eine Zusammenarbeit erschwert. Neben Connected Cars muss ebenfalls die Kommunikation mit RSUs gewährleistet sein. Diese existieren in noch vielfältiger Ausführung und führen weitere Parteien ein, die ihre eigenen Datenformate definieren. Eine Einigung auf einen globalen Standard ist daher ein schwer zu erreichendes Ziel, vor allem aufgrund sich ständig wandelnder Technologie. Zusätzlich werden dadurch Abhängigkeiten erzeugt. Um eine lose Kopplung, nicht nur zwischen all diesen Parteien, sondern auch den eigenen Filtern zu wahren, können Transformationsfilter eingesetzt werden. Die einzige Funktion eines Transformationsfilter ist es, eingehende Nachrichten in ein anderes Format umzuwandeln. Der Inhalt bleibt dabei gleich. Durch diese Abstraktionsschicht muss nicht jede Anwendung mehrere Datenformate unterstützen, sondern kann sich auf eines festlegen. Hersteller

können sich voneinander abkoppeln und müssen lediglich Transformationsfilter einsetzen. Auch innerhalb desselben Systems können Transformationsfilter zwischen Anwendungen eingesetzt werden, sodass diese nur noch ein Datenformat unterstützen müssen.

Metadaten & Konfiguration Es wurden bereits viele Konfigurationsmöglichkeiten erwähnt, um Komponenten für den Connected-Cars-Anwendungsfall zu modellieren. Jedoch kann es sich dabei nur um generische Konfigurationen handeln, da Datenquellen und Filter aus Sicht der Modellierung Black Boxes sind. In der Realität sind Filter grundlegend unterschiedlich und benötigen individuelle Einstellungen. Zum Beispiel muss einem Filter zur Überwachung der Motortemperatur mitgeteilt werden, ab welcher Temperatur Überhitzungsgefahr besteht. Solch spezifische Konfigurationen sind auf einzelne Filter zugeschnitten und dem Modellierungswerkzeug nicht bekannt. Um diese dennoch abbilden zu können, ist es wichtig den Modellierern viel Freiheit in ihren Einstellungen zu geben, auch wenn diese vom Modell selbst nicht mehr interpretiert werden können. Gleiches gilt für die Zuweisung von Metadaten. Einige Komponenten benötigen Zusatzinformationen aus dem Modell. So kann z.B. eine Anwendung zur Analyse des Reifendrucks nicht wissen, um welchen Reifen es sich handelt. Diese Information wird beim Modellieren festgelegt und muss daraufhin beim Deployment der zuständigen Anwendung übergeben werden. Zwar handelt es sich nicht um eine Konfiguration, dennoch ist diese Information wichtig, um die Fahrer bei zu niedrigem Reifendruck angemessen informieren zu können. Um dies im Modell abzubilden, reicht es aus, ein freies Textfeld zur Verfügung zu stellen. Das Format und der Inhalt des Textes ist für das Modell nicht relevant, da es diese nicht interpretiert, sondern lediglich weitergibt. Die Modellierer müssen allerdings sicherstellen, dass die empfangenden Sensoren und Anwendungen die übergebenen Daten korrekt interpretieren können.

Zyklenerkennung Zyklische Verbindungen zwischen Filtern können in vielen Modellen zu Fehlern führen. Bei Logik, die sich auf mehrere Filter ausbreitet, kann dies zwar gewollt sein, dennoch bedarf es großer Vorsicht. In komplexen Modellen können Zyklen unerkannt bleiben und unerwartete Nebeneffekte haben. Diese machen sich erst während der Laufzeit, das bedeutet beim Fahren, bemerkbar und können deshalb zu lebensgefährdenden Situationen führen. Unvorhergesehene Zyklen sind ein Risiko für das gesamte System, da die Nachrichten sich immer weiter aufstauen und die gesamte Pipeline blockieren können. Daher ist es wichtig bereits während der Modellierung eine automatische Erkennung von Zyklen einzusetzen. Folglich können diese z.B. vollständig verboten werden. Ist es vom Anwendungsfall jedoch so gewollt, ist zumindest ein Hinweis auf jeden zyklischen Pfad eine angebrachte Sicherheitsmaßnahme.

Eine Übersicht aller Konzepte ist in Tabelle 5.1 dargestellt. Anforderungen von Connected Cars bezüglich Privatsphäre, Sicherheit und des Netzwerkes können durch Konfigurationsmöglichkeiten, wie z.B. der Priorisierung bestimmter Schnittstellen bewältigt werden. Zusätzlich helfen Modellierungstechniken, wie z.B. dem Einsatz von speziellen Filtern zur Verschleierung und Kodierung der Daten. Die meisten Konzepte behandeln die Komplexität der Modellierung und ermöglichen große Modelle übersichtlich darzustellen. Die beste Lösung zur Sicherstellung der Interoperabilität sind Transformationsfilter, auch wenn diese für die Konvertierung weiterhin Informationen über die Datentypen anderer Hersteller benötigen. Mit der Zuteilung von Filtern zu Knoten erhalten die Modellierer Kontrolle über den Ort an dem potenziell private Daten verarbeitet werden und den Verbrauch der bereitgestellten Ressourcen.

	Privatsphäre	Sicherheit	Netzwerk	Komplexität	Interoperabilität	Datenverarbeitung
Fokus auf Anwendungen				X		
Aufbau eines Filters				X		
Aufbau einer Datenquelle				X		
Templates & Instanzen				X		
Modellierung von Knoten						X
Verschleierungsfilter	X					
Deployment Tagging	X	X				X
Codierungsfilter	X	X	X			
Interne Verschlüsselung	X	X	X			
Validierungsfilter		X				
Individuelle Verbindungstechnologien			X			
Priorisierung von Schnittstellen		X	X			X
Obsolete Nachrichten abweisen		X				
Verbindungsqualität		X	X			
Containerisierung				X		
Abhängigkeiten hervorheben				X		
Filter kategorisieren & hervorheben				X		
Modelle importieren & exportieren				X		
Übersicht aller Knoten				X		
Referenzen modellieren				X		
Typisierte Verbindungen					X	X
Transformationsfilter					X	
Metadaten & Konfiguration						X
Zyklenerkennung						X

Tabelle 5.1.: Übersicht über die vorgestellten Konzepte und die Anforderungen, die sie bewältigen.

5.3. Lebenszyklus eines Modells

Mit dem vorgestellten Konzept ist es möglich, verteilte Datenpipelines speziell für Connected Cars zu modellieren. Um die erzeugten Modelle in Betrieb zu nehmen, wird folgend in Abschnitt 5.3.1 erläutert, wie ein Deployment ablaufen kann. Daraufhin wird in Abschnitt 5.3.2 beschrieben, wie verwendete Modelle gewartet und ausgetauscht werden können.

5.3.1. Deployment

Beim Deployment muss das erstellte Modell auf ein oder mehrere Connected Cars angewendet werden. Dies kann auf unterschiedliche Weisen umgesetzt werden. Über die zuvor beschriebene Export-Funktionalität kann das Modell vom Modellierungswerkzeug gelöst werden. Das Format

der exportierten Datei ist dabei unerheblich. Ein solcher Export ermöglicht es, das Modellierungswerkzeug von der Umsetzung des Deployments und somit jeglichen Installations- und Konfigurationsaufgaben zu trennen. Dadurch muss das Modellierungswerkzeug keine direkte Verbindung zu den Connected Cars besitzen. Diese Verantwortlichkeit kann z.B. auf ein Managementwerkzeug für die Connected-Cars-Flotte übertragen werden. Schließlich kann ein integrierter Interpreter das Modell einlesen und anwenden. Eine andere Möglichkeit ist, diese Funktionalitäten zu verknüpfen und das Deployment direkt über das Modellierungswerkzeug auszuführen. Dafür ist es nötig dieses mit den Connected Cars zu verbinden, sodass Zugriff auf jede ECU und jeden Sensor besteht und Installationen ausgeführt werden können.

Wenn Anwendungen und deren Umgebungen bereits auf den ECUs vorinstalliert sind, ist es möglich, ein Deployment lediglich für die Konfiguration des Systems zu nutzen. Im Modell sind die Relationen von Filtern und Knoten festgehalten, wodurch eine Zuordnung der Komponenten des Modells auf das reale System möglich ist. Auf Basis dessen kann für jede Anwendung die zugehörige Konfiguration gelesen und angewendet werden. Schließlich müssen die Verbindungen der Komponenten gelesen und ebenfalls auf das reale System angewendet werden, um das Deployment abzuschließen. Soll das Modell jedoch nicht bloß zur Konfiguration, sondern für ein vollständiges Deployment verwendet werden, müssen die zu installierenden Anwendungen ebenfalls zum Modell gehören. Da es eine direkte Zusammengehörigkeit zwischen Filter und Anwendung gibt, ist vorstellbar die Anwendungen, z.B. in Form von ausführbaren Installationsskripten, jedem Filter hinzuzufügen. Auf diese Weise erhalten die Modellierer mehr Kontrolle über das Aussehen des Systems und muss keine Annahmen über bereits installierte Anwendungen treffen. Die Entscheidung, welche Anwendung auf welcher ECU installiert werden soll, kann auf verschiedene Weisen modelliert werden. Möglich ist eine direkte Zuweisung als Konfiguration für jeden Filter. Eine weitere Option ist das zuvor erwähnte Konzept des Deployment Taggings, in dem für jeden Filter und Knoten die Anforderungen und Beschaffenheiten, angegeben werden. Dadurch kann eine Verteilung der Filter auf Knoten berechnet und automatisiert durchgeführt werden.

Ähnlich verhält es sich mit den benötigten Verbindungstechnologien. Für jede Verbindung kann eine eigene Technologie modelliert werden. Ob z.B. Kafka- oder MQTT-Broker [Apa; BBBG19] benötigt werden, ist in dem Modell festgehalten. Um keine Annahmen über die vorhandene Infrastruktur in den Fahrzeugen treffen zu müssen, kann dies ebenfalls aus dem Modell gelesen und installiert werden. Flexibilität in der Wahl der Technologie wird ermöglicht, indem sichergestellt ist, dass jede Anwendung sich mit dieser verbinden kann. Allerdings sollte diese Anforderung nicht im Code jeder Anwendung gelöst werden. Stattdessen können die von Hohpe und Woolf [HW04] vorgestellten Messaging Endpoints eingesetzt werden. Diese bilden eine Schnittstelle zwischen Anwendung und Verbindungstechnologie, sodass jeweils nur ein Interface bereitgestellt werden muss. Somit bedarf es beim Deployment zusätzlich einer Installation und Konfiguration dieser Endpoints.

5.3.2. Wartung

Während des Betriebs müssen auf dem System eines Connected Cars Wartungsarbeiten und Updates möglich sein. Dazu gehören Änderungen am Modell. Um Änderungen vorzunehmen, können Modelle in das Modellierungswerkzeug importiert und bearbeitet werden. Die neue Version kann daraufhin exportiert, und wie zuvor besprochen, eingesetzt werden. Abhängig von den Veränderungen im neuen Modell, müssen gegebenenfalls Anwendungen entfernt und installiert werden. Auch Verbindungen, einschließlich derer verwendeten Verbindungstechnologien, können

sich ändern. Um solche Arbeiten zu erleichtern, ist es sinnvoll kein vollständiges Deployment des neuen Modells auszuführen. Stattdessen sollte die Differenz des alten und des neuen Modells berechnet und nur dieser Teil angewendet werden. Auf diese Weise wird die benötigte Zeit für ein neues Deployment reduziert. Mit Hilfe des Blau-Grün-Deployments ist es theoretisch möglich, diese Zeit noch weiter zu reduzieren. Bei diesem wird, neben dem bereits laufendem Modell, die neue Version parallel hochgefahren. Danach wird der Datenverkehr des alten Modells auf das Neue umgeschaltet. Herausforderung dieses Verfahrens im Connected-Cars-Anwendungsfall ist die Verteiltheit der Anwendungen. Diese muss in beiden Modellen sichergestellt sein, wodurch es zu doppelter Belastung auf jeder ECU und somit einer potenziellen Sicherheitsgefährdung kommt. Stattdessen ist es ratsam, Updates durchzuführen während das Fahrzeug nicht in Nutzung ist.

6. Implementierung

Zur Umsetzung des Konzeptes wird ein Prototyp eines Modellierungswerkzeuges entwickelt. Mit diesem lassen sich die Modellierung der Datenquellen, Filter, Knoten und Verbindungen umsetzen. Zusätzlich sind die zuvor erklärten Konfigurationsmöglichkeiten modellierbar. Das Kapitel beginnt in Abschnitt 6.1 mit einer Beschreibung der verwendeten Technologien und des allgemeinen technischen Aufbaus der Anwendung. In Abschnitt 6.2 wird das Modellierungswerkzeug vorgestellt und erläutert, wie sich die Konzepte modellieren lassen. Abschnitt 6.3 befasst sich mit dem verwendeten Datenmodell und gibt Aufschluss darüber, wie die Komponenten aus technischer Sicht zusammenhängen. Zuletzt wird in Abschnitt 6.4 eine Demonstrations-Umgebung vorgestellt. In dieser wird ein Connected Car simuliert und gezeigt, wie ein Deployment für die Anwendung eines Modells aussehen könnte.

6.1. Technischer Aufbau

Bei dem Prototypen handelt es sich um eine mit React [Fac21] erstellte Browseranwendung. Zur Realisierung der Zeichenfläche, für die graphische Modellierung der Datenquellen, Filter und deren Verbindungen, dient das Framework Rete.js [Sto21]. Im Folgenden wird dieses als Editor bezeichnet. An dem Prototypen ist keine Datenbank angeschlossen. Jegliche Zustandsverwaltung während des Modellierens wird in einem Redux-Store [AC21] erfasst und nicht sitzungsübergreifend persistiert. Die gesamte Anwendung ist in JavaScript [ECM21] geschrieben. Das Modellierungswerkzeug kommuniziert mit dem Editor über direkte Funktionsaufrufe. Zum Beispiel kann auf diese Weise, beim Anlegen eines Filters im Store, das Gegenstück dem Editor hinzugefügt werden. Die Kommunikation der Gegenrichtung, wenn z.B. im Editor eine Verbindung erzeugt wird, geschieht über Trigger des Editors. Diesen können Funktionen übergeben werden, welche vom Trigger mit den erforderlichen Parametern aufgerufen werden. Es muss jedoch zurückverfolgbar sein, um welche Elemente es sich im Editor handelt, wenn ein Trigger auslöst. Dafür wird jedem Element beim Anlegen eine ID zugewiesen, welche dem Gegenstück im Store entspricht. Die Funktionen, welche vom Trigger aufgerufen werden, können anhand dieser IDs den neuen Zustand des Editors auf den Store reproduzieren. Der Großteil der Kommunikation zwischen den Komponenten geschieht durch solch eine Übertragung von IDs. Werden zusätzliche Informationen benötigt, werden diese aus dem Store geladen. Das Design der Benutzeroberfläche beruht auf Material UI [Mat21]. Alle verwendeten Frameworks laufen unter der MIT-Lizenz.

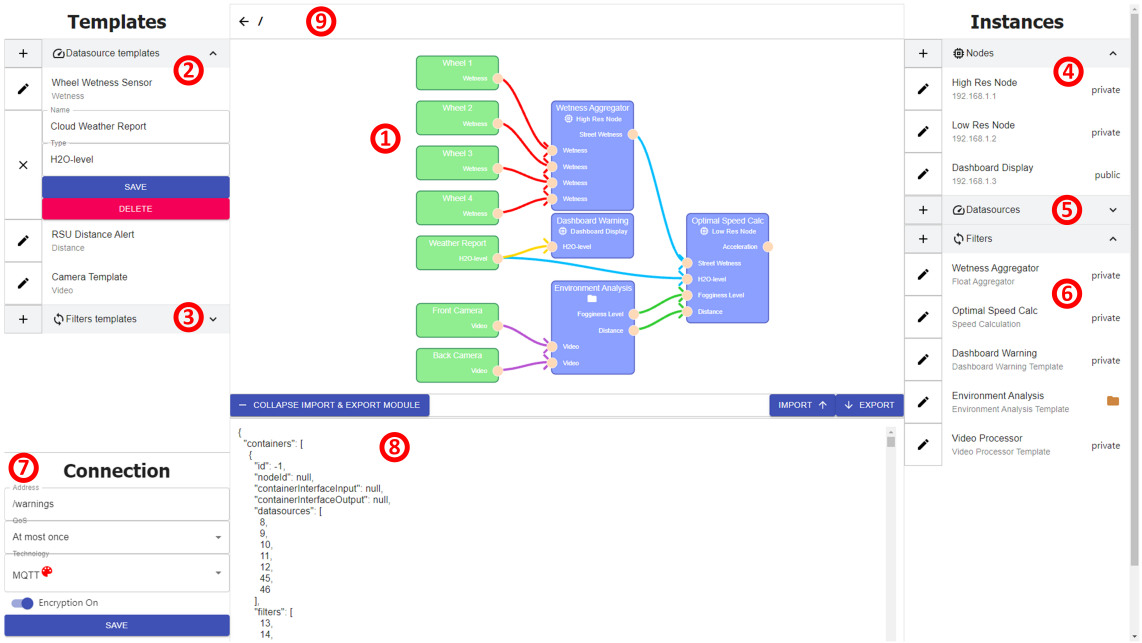


Abbildung 6.1.: Übersicht über das Modellierungswerkzeug.

6.2. Modellierungswerkzeug

Im Folgenden wird der Prototyp eines Modellierungswerkzeuges des in Kapitel 5 erklärten Konzeptes anhand von Bildschirmaufnahmen vorgestellt. Zur besseren Veranschaulichung ist in diesem bereits ein Beispiel eines Modells angelegt. Das Modell zeigt einen Anwendungsfall, in dem Daten von Feuchtigkeitssensoren an den Rädern, dem Wetterbericht und den Außenkameras dazu verwendet werden, die optimale Geschwindigkeit des Fahrzeuges zu berechnen. In verschiedenen Zwischenschritten werden Teilergebnisse ermittelt. Ein Filter berechnet die Straßennässe anhand der Feuchtigkeitssensoren an den Rädern. Die Kameras ermöglichen Aufschluss über eine Einschränkung der Sichtbarkeit durch potenziellen Nebel. Zusätzlich kann anhand der Kameras und unterstützenden RSUs der Abstand zum vorausfahrenden Fahrzeug berechnet werden. Auf Basis der berechneten Straßennässe, der Nebelstärke, des Abstandes und eines Wetterberichtes wird die optimale Fahrgeschwindigkeit ermittelt. Außerdem soll der Wetterbericht auf einem Dashboard im Fahrzeug angezeigt werden. All diese Funktionen werden auf drei Knoten gehostet. Eine vollständige Übersicht über das Modellierungswerkzeug und des modellierten Beispiels ist in Abbildung 6.1 dargestellt. Die Funktion und implementierten Konzepte der Komponenten werden in den folgenden Abschnitten anhand der Markierungen (1) bis (9) in der Abbildung näher erläutert.

6.2.1. Editor

Im Zentrum des Modellierungswerkzeuges, bei (1), befindet sich der Editor. Im Editor werden angelegte Datenquellen, Filter und Verbindungen angezeigt. Abbildung 6.2 zeigt ein Beispiel für eine Datenquelle, die mit einem Filter verbunden ist. Zur leichten Differenzierung sind Datenquellen grün und Filter blau eingefärbt. Verbindungen können über Drag and Drop (D&D) zwischen

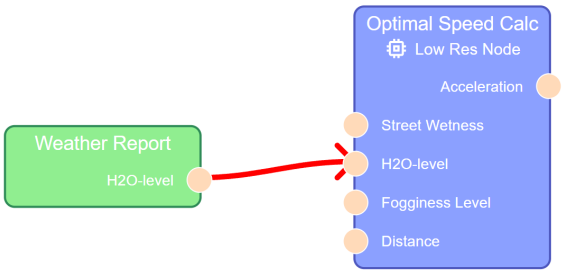
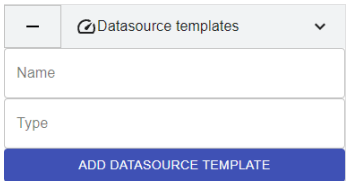
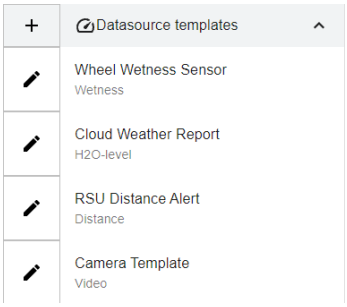


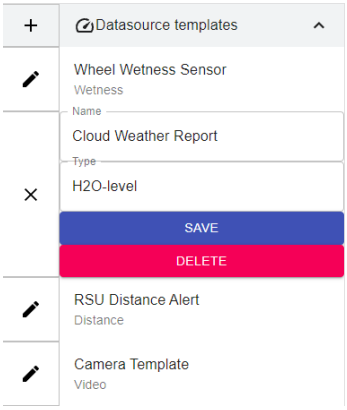
Abbildung 6.2.: Verbindung zwischen einer Datenquelle und einem Filter.



(a) Hinzufügen eines Templates.



(b) Übersicht aller Templates.



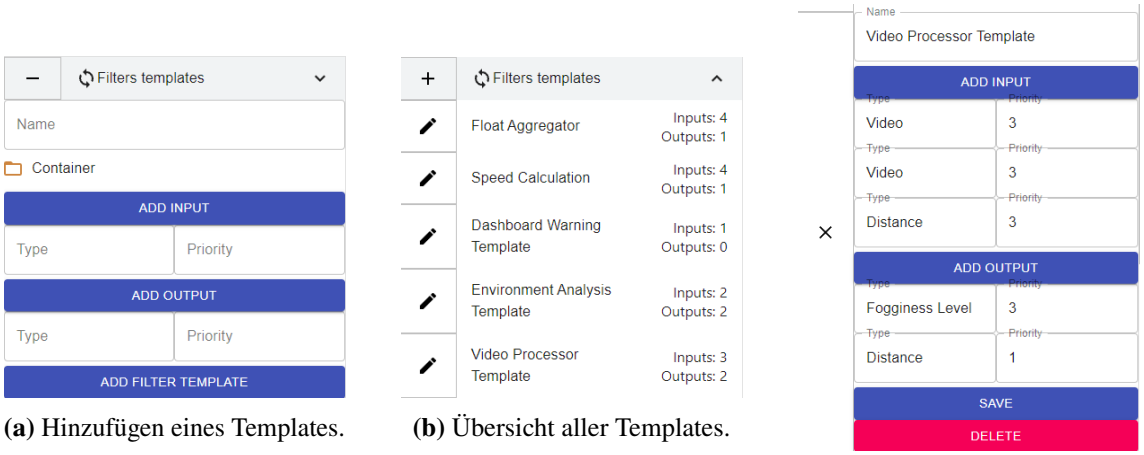
(c) Editieren eines Templates.

Abbildung 6.3.: Menü für Templates von Datenquellen.

den in beige gefärbten Ausgängen und Eingängen erzeugt werden. Verbindungen können an den Eingängen der Filter wieder aufgehoben und somit entfernt werden. Alle Komponenten im Editor sind über D&D bewegbar. Dadurch haben die Modellierer freie Gestaltung bei der Anordnung der Komponenten. Die Verbindungen bleiben dabei erhalten und werden automatisch mit verschoben. Die Ansicht des Editors kann über D&D der Hintergrundfläche verschoben werden, um verschiedene Sektionen des Modells in Perspektive zu bringen. Durch Rollen des Mausrads wird an das Modell heran- bzw. herausgezoomt.

6.2.2. Templates für Datenquellen

Bei (2) befindet sich das in Abbildung 6.3 abgebildete Menü der Templates von Datenquellen. Mit Klick auf den Plus-Knopf öffnet sich ein Menü zum Erstellen eines Templates (6.3a). Dazu wird die Angabe eines Namens und eines Datentyps benötigt. Durch den Datentyp wird modelliert, welche Daten von dieser Datenquelle produziert werden und welches Format sie haben. Über einen Klick auf den Hinzufügen-Knopf wird das Template angelegt. Alternativ kann über den Minus-Knopf das Menü wieder geschlossen werden. Mit Klick auf den Titel des Drop-Down-Menüs öffnet sich die Liste aller angelegten Templates (6.3b). Für jedes Template ist zur Identifizierung der Name und der Datentyp angegeben. Zusätzlich ist jedem Template ein Bearbeiten-Knopf zugeordnet. Mit Klick auf diesem wird ein Menü zum Editieren des Templates geöffnet (6.3c). In diesem können die



(a) Hinzufügen eines Templates. **(b)** Übersicht aller Templates. **(c)** Editieren eines Templates.

Abbildung 6.4.: Menü für Templates von Filtern.

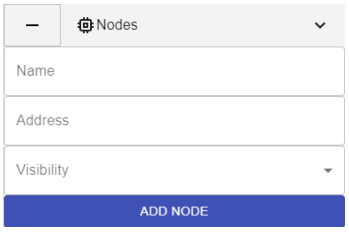
gespeicherten Informationen bearbeitet werden. Über den Speichern-Knopf werden die Änderungen übernommen. Es kann ebenfalls das Template gelöscht oder das Menü, ohne die Änderungen zu speichern, geschlossen werden.

6.2.3. Templates für Filter

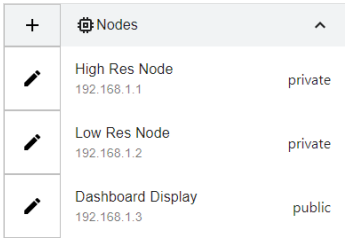
Abbildung 6.4 und (3) zeigen das Menü der Filter-Templates. Die Liste, das Hinzufügen und das Editieren sind für alle Menüs konzeptionell gleich. Unterschiede gibt es nur in den zu modellierenden Eigenschaften. Anders als bei den Templates der Datenquellen kann ein Filter eine beliebige Anzahl an Eingängen und Ausgängen besitzen. Diese können über die Knöpfe zum Hinzufügen der jeweiligen Schnittstelle dem Template angefügt werden (6.4a). Jede Schnittstelle benötigt Informationen über den zu erwartenden Datentypen und ihre Priorität. Bei der Wahl der Priorität ist kein Schema vorgeschrieben. Dabei kann es sich um eine numerische Rangfolge oder auch Schlüsselwörter handeln. Wichtig ist, dass die Anwendungen in der Lage sind diese Information zu interpretieren. Außerdem kann für Templates bei Klick auf das Ordner-Icon ausgewählt werden, ob es sich um Container handelt. Da Container keine Anwendungen widerspiegeln, werden für deren Schnittstellen keine Prioritäten benötigt. Ist diese Option ausgewählt, verschwinden die Textfelder für die Prioritäten von jeder Schnittstelle. Nach dem Hinzufügen des Templates wird es der Liste angefügt (6.4b). Neben dem Namen wird pro Template die Anzahl der Eingänge und Ausgänge eingeblendet. Beim Editieren des Eintrages kann jeder gesetzte Wert geändert werden (6.4c). Bei Containern gibt es wie zuvor keine Eingabe für die Priorität der Schnittstellen.

6.2.4. Knoten

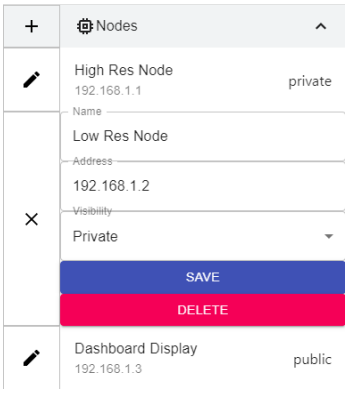
Die Modellierung von Knoten, bei (4), ist in Abbildung 6.5 abgebildet. Beim Hinzufügen müssen Name, Adresse und eine Sichtbarkeit ausgewählt werden (6.5a). Die Adresse ist notwendig, damit beim Deployment die Anwendungen auf die richtige ECU geladen werden können. Bei der Sichtbarkeit handelt es sich um eine reduzierte Variante des Deployment Taggings. Sie stellt



(a) Hinzufügen eines Knotens.

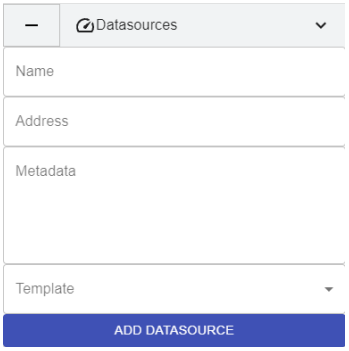


(b) Übersicht aller Knoten.

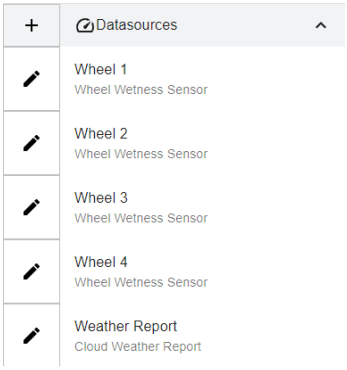


(c) Editieren eines Knotens.

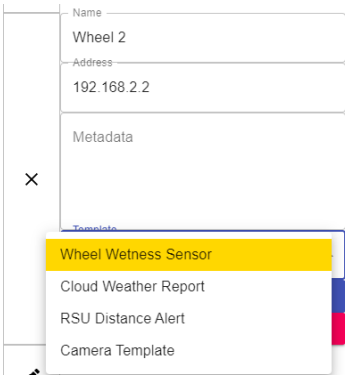
Abbildung 6.5.: Menü für Knoten.



(a) Hinzufügen einer Datenquelle.



(b) Übersicht aller Datenquellen.



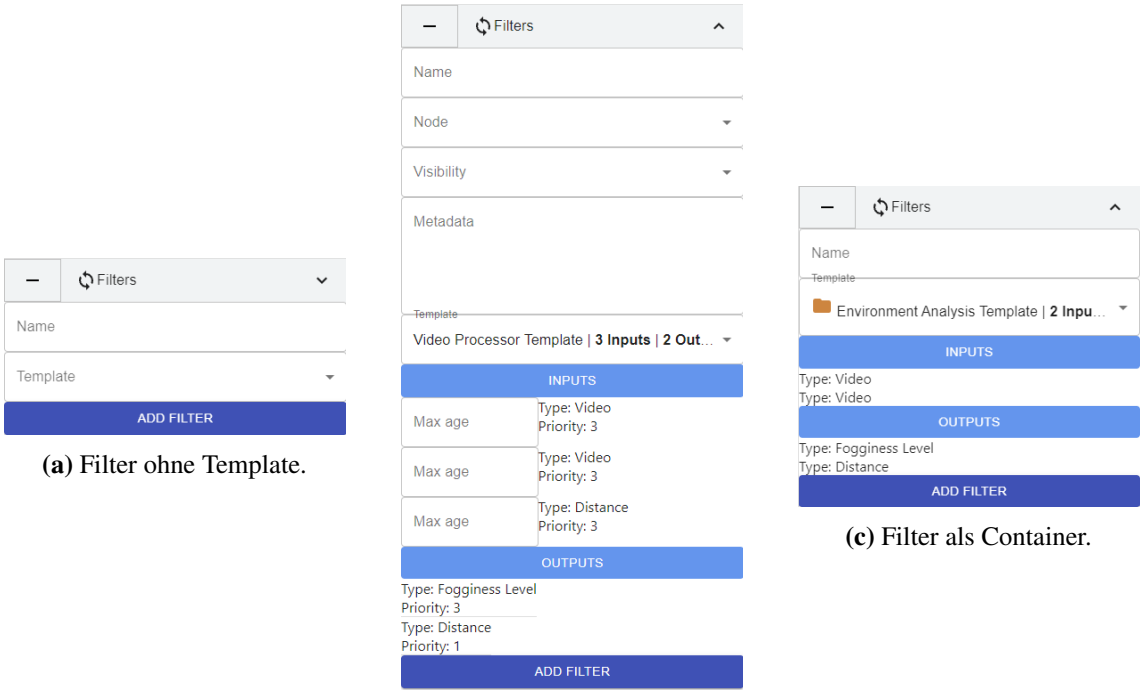
(c) Editieren einer Datenquelle.

Abbildung 6.6.: Menü für Datenquellen.

einen möglichen Tag dar, der Teil eines vollständigen Tagging Systems sein könnte. Über diesen werden Eigenschaften des Knotens festgelegt, die ein Filter gegebenenfalls voraussetzt. In diesem Prototyp können die Sichtbarkeiten *Öffentlich* oder *Privat* gewählt werden. Keine Auswahl ist ebenfalls möglich. Die Werte stehen für die Einstellung, ob der Knoten Teil des Connected Cars oder ein öffentlicher Knoten, wie z.B. eine RSU, ist. Auf diese Weise kann entschieden werden, auf welchem Knoten eine Anwendung bereitgestellt werden darf. In der Liste wird für jeden Knoten der Namen, die Adresse und die eingestellte Sichtbarkeit angezeigt (6.5b). Beim Editieren ist jeder Wert änderbar (6.5c).

6.2.5. Datenquellen

Datenquellen können, wie in Abbildung 6.6 dargestellt, bei (5) modelliert werden. Jede Datenquelle hat einen Namen, eine Adresse, Metadaten und ein Template (6.6a). Die Adresse wird erneut für das Deployment benötigt, falls der Sensor konfiguriert oder Messaging Endpoints installiert werden müssen. Wie im Konzept erwähnt, ermöglichen Metadaten den Modellierern Freiheiten in der Konfiguration. Zuletzt muss eines der modellierten Templates ausgewählt werden, damit die



(a) Filter ohne Template.

(b) Filter mit Template.

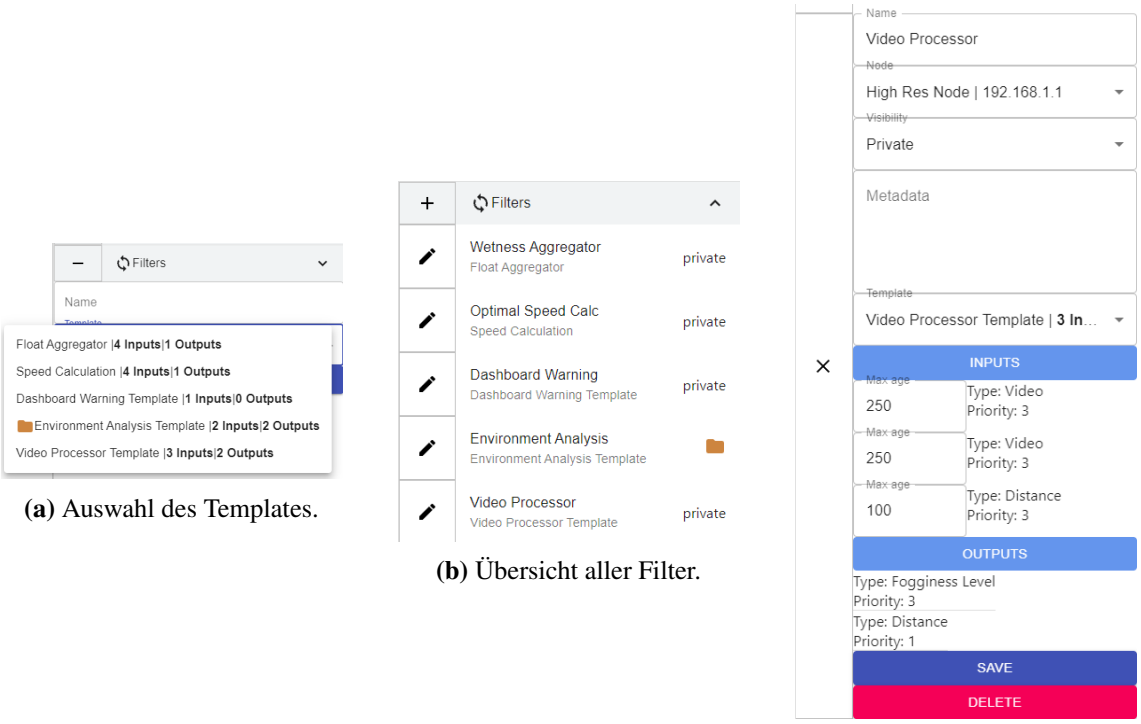
(c) Filter als Container.

Abbildung 6.7.: Hinzufügen eines Filters.

Datenquelle ihre Schnittstelle kennt. Beim Speichern wird die Datenquelle dem Editor hinzugefügt. In der Übersicht aller Datenquellen können diese anhand ihres Namens und des Templates identifiziert werden (6.6b). Das Editieren ermöglicht die Änderung jedes Wertes. Die entsprechende Datenquelle im Editor wird dabei aktualisiert. Falls neue Templates modelliert werden, aktualisiert sich die Auswahlmöglichkeiten für diese (6.6c).

6.2.6. Filter

Bei (6) befindet sich das Menü für Filter. Abbildung 6.7 stellt dar, wie abhängig von der Art des Templates, ein Filter unterschiedliche Eigenschaften haben kann. Jedem Filter kann ein Name und ein Template zugewiesen werden (6.7a). Wird ein Template ausgewählt, bei dem es sich nicht um einen Container handelt, werden weitere Datenfelder aufgedeckt (6.7b). Da es sich folglich um einen Filter handelt, der eine Anwendung repräsentiert, können zusätzlich zu dem Knoten die Sichtbarkeit und Metadaten angegeben werden. Über die Angabe des Knotens kann eine direkte Zuweisung für das Deployment getroffen werden. Dies widerspricht der automatisierten Zuweisung über das Deployment Tagging, weshalb diese Option auch frei bleiben kann, um sie nicht zu nutzen. Die Sichtbarkeit stellt das Gegenstück zu demselben Datenfeld der Knoten dar (vgl. Abschnitt 6.2.4). In diesem Fall steht es für die Anforderungen an Knoten. Mit diesen Feldern kann automatisiert eine Verteilung von Filtern auf Knoten berechnet und beim Deployment durchgeführt werden. Wie auch bei Datenquellen können Metadaten gesetzt werden, welche die Konfiguration unterstützen. Für jeden Eingang, der im Template modelliert wurde, kann ein maximales Alter der eingehenden Nachrichten bestimmt werden. Dies konfiguriert die Erkennung obsoleter Nachrichten. Zur Identifizierung der



(a) Auswahl des Templates.

(b) Übersicht aller Filter.

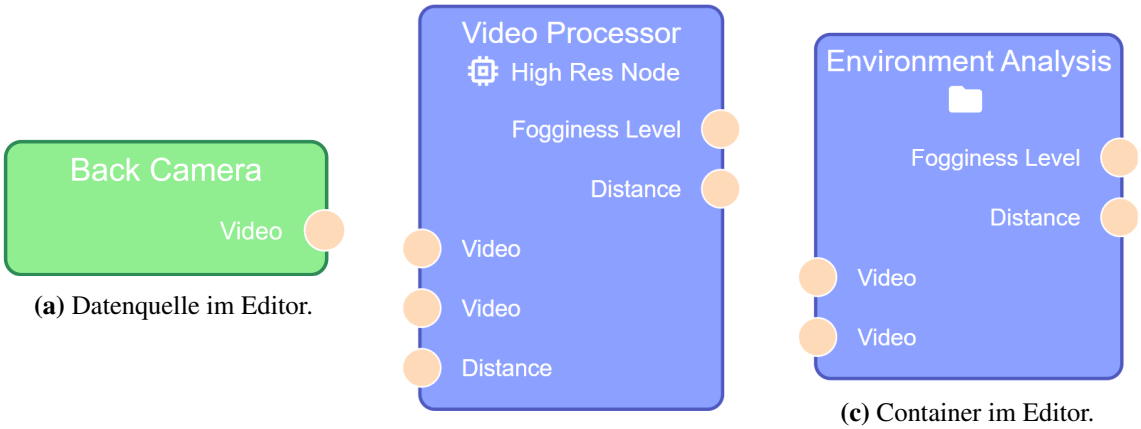
(c) Editieren eines Filters.

Abbildung 6.8.: Menü für Filter.

Schnittstellen sind zusätzlich der Datentyp und die Priorität, welche im Template modelliert wurden, dargestellt, aber nicht editierbar. Diese können, wie zuvor beschrieben, nur im Menü der Templates für Filter editiert werden. Für die Ausgänge des Filters gibt es keine weiteren Modellierungsoptionen. Dennoch werden aus Informationszwecken die zuvor festgelegten Datentypen und Prioritäten angezeigt. Wenn für den Filter ein Template ausgewählt wird, bei dem es sich um einen Container handelt, fallen diese Konfigurationen weg (6.7c). Es bleibt bei der Auswahl des Namens und des Templates. Die Datentypen der Schnittstellen des Templates werden weiterhin angezeigt, sind aber nicht editierbar.

Aufgrund der Vielfalt der Templates werden bei deren Auswahl weitere Merkmale angezeigt. Abbildung 6.8a zeigt das Auswahlmenü für Templates. Neben dem Namen wird die Anzahl der Eingänge und Ausgänge dargestellt. Handelt es sich um einen Container, wird zusätzlich ein Ordner-Icon eingeblendet. Für jeden Filter ist in der Übersicht der Name, das ausgewählte Template und die Sichtbarkeit einsehbar (6.8b). Hat der Filter als Template einen Container ausgewählt, und besitzt somit keine Sichtbarkeit, wird statt dieser ein Ordner-Icon angezeigt. Beim Editieren eines Filters lassen sich alle zuvor beschriebenen Datenfelder ändern (6.8c). Hierbei fallen erneut Konfigurationen weg, wenn es sich um einen Container handelt.

Abbildung 6.9 zeigt wie die modellierten Komponenten im Editor aussehen. Datenquellen besitzen einen grünen Hintergrund und sind mit ihrem Namen annotiert (6.9a). Ihr einziger Ausgang trägt den Datentyp der Nachrichten, die sie produzieren. Filter haben einen blauen Hintergrund (6.9b). Unter ihrem Namen befindet sich der Name des Knotens, auf dem die repräsentierte Anwendung bereitgestellt wird. Erneut ist jede Schnittstelle mit dem erwarteten Datentyp annotiert. Mit diesen



(a) Datenquelle im Editor.

(b) Filter im Editor.

(c) Container im Editor.

Abbildung 6.9.: Komponenten im Editor.

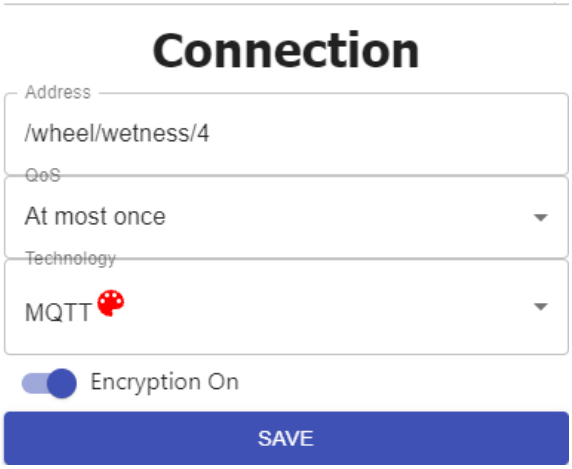


Abbildung 6.10.: Menü einer Verbindung.

fällt es leichter, passende Verbindungen zu identifizieren. Ein Container unterscheidet sich im Aussehen von einem Filter nur darin, dass er anstelle des Namens des Knoten ein Ordner-Icon abbildet (6.9b). Dieses dient zur Unterscheidung von regulären Filtern. Außerdem signalisiert der fehlende Knoten, dass Container keinen Einfluss auf das Deployment haben.

6.2.7. Verbindungen

Bei (7) befindet sich das in Abbildung 6.10 dargestellte Menü für Verbindungen. Verbindungen können im Editor ausgewählt werden, wodurch sich das Menü für die ausgewählte Verbindung öffnet. Verbindungen werden im Editor erstellt und entfernt. Deshalb gibt es im Menü lediglich die Funktion diese zu editieren. Eine Verbindung hat eine Adresse, einen QoS, eine zugewiesene Verbindungstechnologie und eine Verschlüsselungsoption. Bei der Adresse handelt es sich um eine frei definierbare Einstellung. Diese kann je nach Technologie anders aussehen. Ist als Technologie MQTT gewählt, könnte diese z.B. ein Topic sein. Die Adresse wird beim Deployment beiden

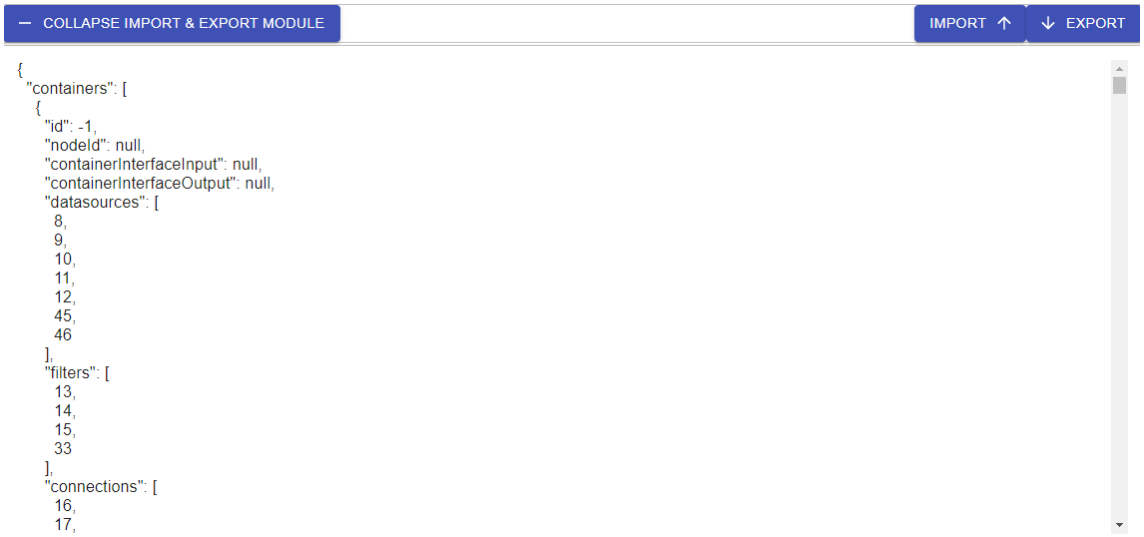
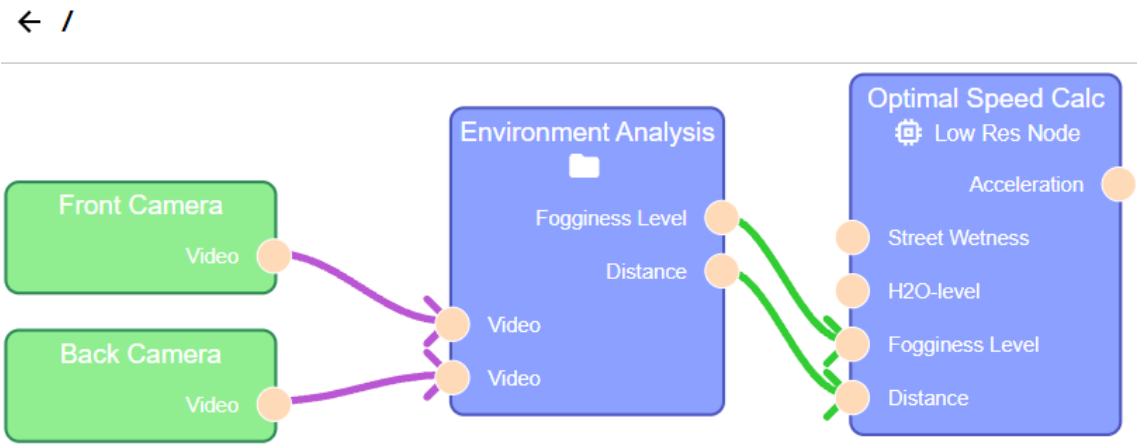


Abbildung 6.11.: Menü für das Importieren und Exportieren des Modells.

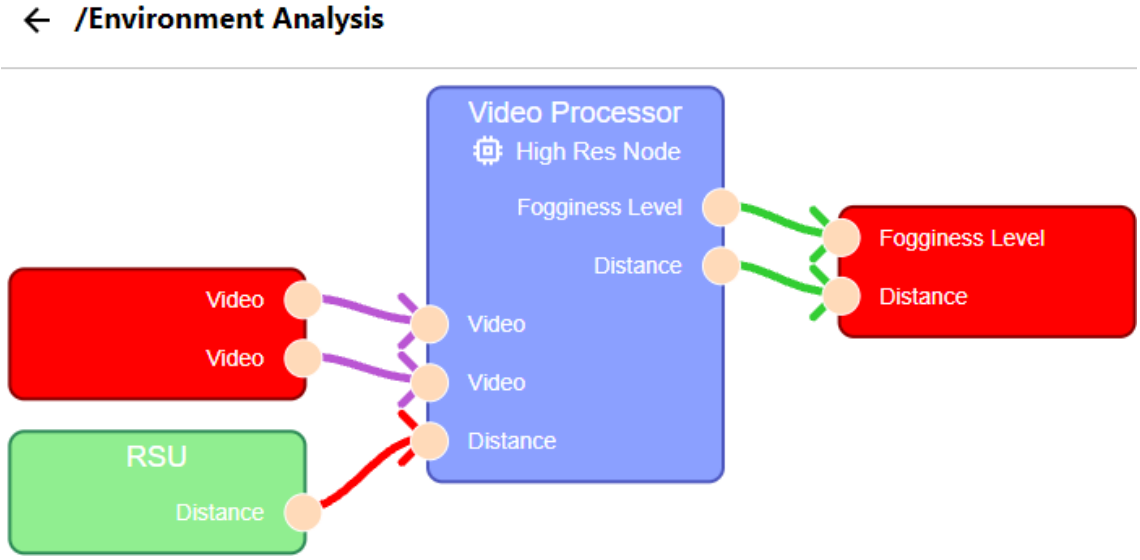
verbundenen Komponenten, d.h. den Sensoren und Anwendungen, mitgeteilt. Diese müssen in der Lage sein, die Adresse zu interpretieren. Auf diese Weise sind ebenfalls proprietäre Formate einsetzbar. Beim QoS handelt es sich um ein Beispiel für eine Einstellung der Verbindungsqualität. Jegliche andere Definitionen für diese sind ebenfalls denkbar. Über die Einstellung der Verbindungstechnologie kann z.B. zwischen MQTT, Hypertext Transfer Protocol (HTTP) oder RTP gewählt werden. Wie im Konzept erwähnt, ist eine einstellbare Verbindungstechnologie auch für die anderen Schichten des OSI-Modells denkbar. In diesem Protoypen wurde sich auch eine Schicht beschränkt. Abhängig der übertragenen Daten kann die Verbindungstechnologie für jede Verbindung anders ausfallen. Neben dem Format, in dem die Daten übertragen werden, beeinflusst diese Einstellung, ob weitere Anforderungen, wie z.B. Zuverlässigkeit und Latenz der Übertragung erfüllt werden können. Jede Technologie hat eine zugewiesene Farbe, welche die Verbindungen im Editor annehmen. Auf diese Weise ist die Einstellung erkennbar, ohne das Menü öffnen zu müssen. Zuletzt gibt es die Möglichkeit zu wählen, ob die Verbindung verschlüsselt werden soll. Diese Option ermöglicht die Wahl zwischen höherer Geheimhaltung und niedrigerer Latenz beim Erzeugen der Nachricht. Auf eine Übersicht aller Verbindungen in Form einer Liste wurde verzichtet, da eine Verbindung keinen Namen benötigt und stattdessen über die beiden verbindenden Komponenten identifiziert wird.

6.2.8. Import & Export

Abbildung 6.11 und (8) zeigen das Menü, in dem Modelle importiert und exportiert werden können. Im Ausgangszustand ist dieses geschlossen. Über einen Knopf lässt sich dieses öffnen und schließen. Im geschlossenen Zustand existiert nur der Knopf zum Öffnen. Im geöffneten Zustand sind weitere Knöpfe für den Import und Export eingeblendet, sowie ein Textfeld. In das Textfeld kann ein Modell eingetragen werden. Bei Betätigung des Import-Knopfes wird dieses in den Editor geladen und überschreibt das Modell, das zuvor zu sehen war. Über den Export-Knopf wird das aktuelle Modell in das Textfeld geschrieben. Von dort aus kann es zu einem anderen Speicherort kopiert werden.



(a) Außerhalb des Containers.



(b) Innerhalb des Containers.

Abbildung 6.12.: Container und Verzeichnisstruktur.

6.2.9. Container

Ein Beispiel für den Einsatz von Containerisierung ist in Abbildung 6.12 abgebildet. Die zugehörige Verzeichnisstruktur befindet sich bei (9), welche Auskunft über die Verschachtelung gibt. Container sind im Editor als Filter dargestellt und signalisieren über ein Ordner-Icon, dass es sich bei ihnen um Container handelt. Durch Container kann Komplexität hinter einem einzigen Filter verborgen werden. Abbildung 6.12a zeigt einen Container mit jeweils zwei Eingängen und Ausgängen. Die Verzeichnisstruktur gibt Aufschluss darüber, dass der Editor sich im Wurzel-Verzeichnis befindet. Über einen Doppelklick auf den Filter öffnet sich der Container. Der Editor wechselt die Ansicht zu dem Inhalt des Containers (6.12b). Dies wird ebenfalls durch die aktualisierte

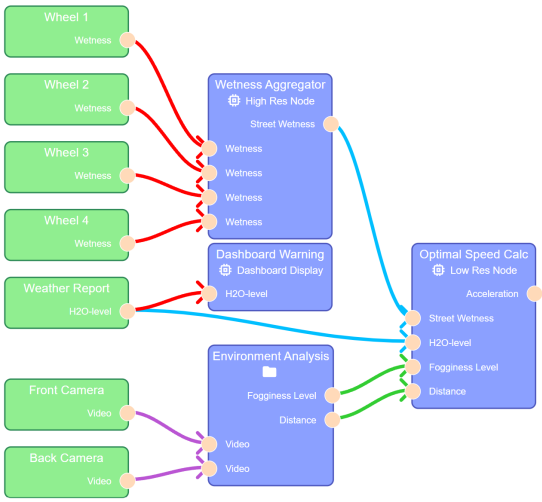
Verzeichnisstruktur signalisiert. Das Verzeichnis ist nicht mehr die Wurzel, sondern das des Containers und trägt daher dessen Namen. In einem Container können rekursiv weitere Container angelegt und geöffnet werden. Die Verzeichnisstruktur fügt den Namen des zuletzt geöffneten Containers dem Verzeichnispfad an. Durch Klick auf den nach links gerichteten Pfeil kann der aktuelle Container verlassen werden. Beim Verlassen eines Containers wird dessen Name aus dem Verzeichnispfad entfernt und der Editor aktualisiert seine Ansicht auf den Container des neuen Verzeichnisses. Innerhalb eines Containers existieren bis zu zwei rote Filter ohne Namen. Bei diesen handelt es sich um Eingangs- und Ausgangsfilter, welche automatisch generiert werden. Der Eingangsfilter repräsentiert alle Eingänge des geöffneten Containers, indem er diese als Ausgang besitzt. Selbes gilt für den Ausgangsfilter und die Ausgänge des Containers. Die Verbindungen des Containers werden auf diese Filter übertragen. Zum Beispiel besitzt der Container in Abbildung 6.12 zwei eingehende Verbindungen des Typs „Video“, die den Außenkameras entstammen. Diese Verbindungen sind auf den Eingangsfilter innerhalb des Containers reproduziert und können auf diese Weise auf der niedrigeren Abstraktionsebene weiterverwendet werden. Das bedeutet, dass die beiden Datenquellen „Front Camera“ und „Back Camera“ mit dem Filter „Environment Analysis“ innerhalb des Containers verbunden sind. Im Editor befindet sich somit zweimal die gleiche Verbindung. In der Realität existiert diese nur einmal. Die inneren Verbindungen sind lediglich ein Mittel, um die Containerisierung zu ermöglichen. Deshalb kopieren diese jegliche Konfigurationen der originalen Verbindungen, wie Adresse und QoS. Für das Deployment ist es somit unerheblich, wie sehr ein Modell verschachtelt ist. Es ändert nichts an der Anzahl der real zu erstellenden Verbindungen. Hat ein Container keine Eingänge oder Ausgänge, werden die repräsentierenden Eingangs- bzw. Ausgangsfilter nicht benötigt und somit nicht erzeugt. Wird ein Container gelöscht, werden alle sich darin befindlichen Filter und Verbindungen ebenfalls gelöscht.

6.2.10. Verschiedenes

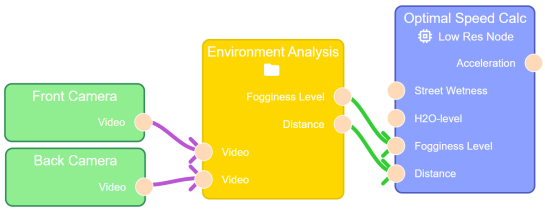
Viele Konzepte sind bei der vorigen Erläuterung zum Aufbau des Modellierungswerkzeuges bereits aufgezeigt worden. Im Folgenden werden weitere Konzepte demonstriert, die bisher nicht erkennbar waren.

Abhängigkeiten hervorheben Bei komplexen Modellen kann es hilfreich sein nur die Abhängigkeiten bestimmter Komponenten sehen zu können und Irrelevantes auszublenden. Dabei soll lediglich die Ansicht angepasst werden, ohne das Modell zu verändern. Im Modellierungswerkzeug ist dies durch einen Rechtsklick auf eine Komponente möglich. Dies gilt sowohl für Datenquellen als auch Filter. Abbildung 6.13 zeigt ein Beispiel für das Ausblenden von Komponenten ohne Abhängigkeiten. In Abbildung 6.13a ist das Ausgangsmodell gezeigt. Abbildung 6.13b bis Abbildung 6.13d zeigen wie das Modell aussieht, wenn die Abhängigkeiten der in gold markierten Komponenten hervorgehoben werden. Durch einen Rechtsklick außerhalb einer Komponente wird das Ausgangsmodell wiederhergestellt.

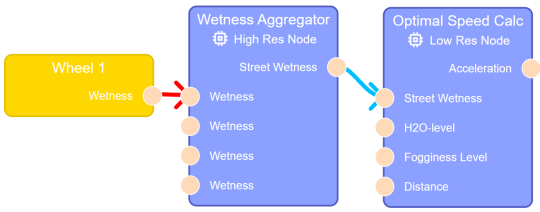
Zyklenerkennung Bei dem Anlegen jeder Verbindung wird überprüft, ob sie einen Zyklus verursachen würde. Dies schließt ebenfalls aus, dass Filter mit sich selbst verbunden sind. Wird eine zyklische Verbindung erkannt, erscheint eine Warnung im Browser, die auf den Zyklus hinweist. Bei Bestätigung der Warnung, wird die Verbindung automatisch entfernt.



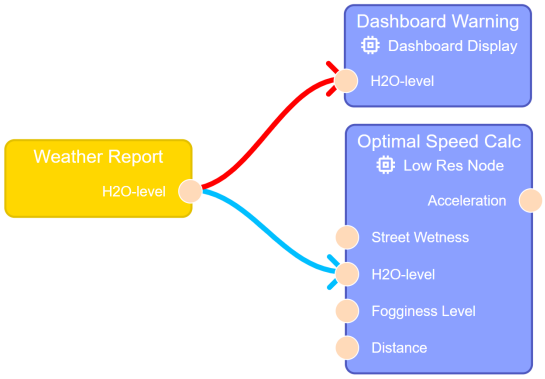
(a) Ausgangsmodell ohne hervorgehobene Komponenten.



(b) Abhängigkeiten des Filters „Environment Analysis“.



(c) Abhängigkeiten der Datenquelle „Wheel 1“.



(d) Abhängigkeiten der Datenquelle „Weather Report“.

Abbildung 6.13.: Hervorheben der Abhängigkeiten der in gold markierten Komponenten.

Benutzerfreundlichkeit Benutzerfreundlichkeit unterstützt die Bedienung des Modellierungswerkzeuges. Zuvor wurde gezeigt, dass Komponenten gold hervorgehoben werden können. Wie in Abbildung 6.14 demonstriert, gilt dies sowohl für Einträge in den Listen als auch den Komponenten im Editor. Die hervorgehobenen Einträge sind miteinander synchronisiert. Das bedeutet, dass beim Auswählen eines Eintrages in der Liste das repräsentierte Gegenstück im Editor hervorgehoben wird. Vice versa kann eine Komponente im Editor geklickt werden, um den Listeneintrag hervorzuheben. In den jeweiligen Listen bezieht sich die Hervorhebung sowohl auf die Komponente als auch deren Template. Auf diese Weise wird eine leicht erkennbare Verbindung der Gegenstücke hergestellt, um z.B. Änderungen vorzunehmen, ohne nach den Elementen suchen zu müssen. Beim Klicken von Verbindungen öffnet sich das Menü zur Bearbeitung. Gleichzeitig werden auch diese in gold hervorgehoben, um erkennbar zu machen auf welche Verbindung sich das Menü bezieht. Des Weiteren gibt es eine Zoom-Funktion, die es ermöglicht bestimmte Teile des Modells besser erkennbar zu machen, oder einen Überblick über größere Teile des Modells zu erhalten.

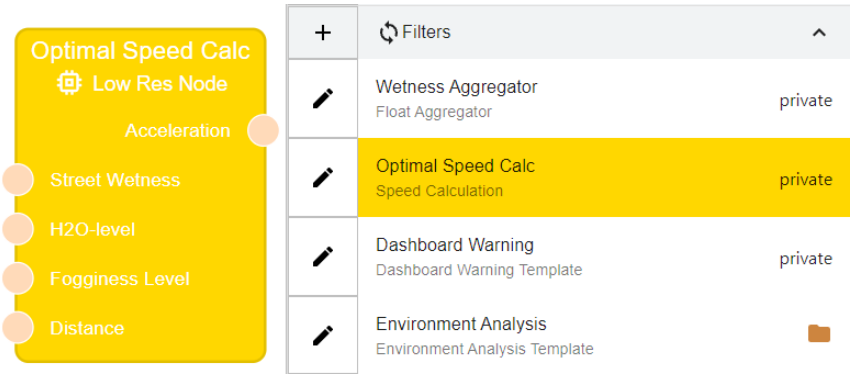


Abbildung 6.14.: Hervorheben angeklickter Komponenten im Editor und in der Liste.

Listing 6.1 Wurzelelemente des Datenmodells.

```
{
  "containers":[],
  "containerInterfaceTemplates":[],
  "containerInterfaces":[],
  "datatransfers":[],
  "qoss":[],
  "physicalNodes":[],
  "datasourceTemplates":[],
  "filterTemplates":[],
  "datasources":[],
  "filters":[],
  "connections":[]
}
```

6.3. Datenmodell

In diesem Abschnitt wird genauer auf das Datenmodell eingegangen. Dazu werden anhand eines gespeicherten Modells das Format und die Zusammenhänge der Daten erläutert. Dabei wird näher auf Filter, Verbindungen und Container eingegangen. Für diesen Zweck ist jeweils ein Beispiel des in diesem Kapitel verwendeten Modells angegeben. Zur besseren Lesbarkeit sind diese teilweise gekürzt. Die Teile des Datenmodells, auf die hier nicht eingegangen wird, sind in Anhang A zu finden. Das Modell wird in Redux [AC21] als JavaScript-Objekt gespeichert. Beim Exportieren wird dieses zu JavaScript Object Notation (JSON) konvertiert. Die grundlegende Struktur wird dabei nicht geändert.

Die Wurzelelemente des Datenmodells sind in Listing 6.1 aufgezeigt. Jedes Element enthält eine Liste. Die Templates und zugehörigen Instanzen werden voneinander getrennt gespeichert. Alle Elemente besitzen eindeutige IDs und werden über diese zueinander referenziert. Verbindungen werden ebenfalls getrennt der Datenquellen und Filter gespeichert.

Listing 6.2 Datenmodell eines Filters.

```
"filterTemplates":[
  {
    "id":34,
    "name":"Video Processor Template",
    "isContainer":false,
    "inputs":[
      {
        "index":"0",
        "type":"Video",
        "priority":"3"
      },
      [...]
    ],
    "outputs":[
      {
        "index":"0",
        "type":"Fogginess Level",
        "priority":"3"
      },
      [...]
    ]
  }
]

"filters":[
  {
    "id": 35,
    "name": "Video Processor",
    "physicalNode": 0,
    "template": 34,
    "maxAges": [250, 250, 100],
    "position": [0, -100],
    "visibility": "private",
    "metadata": ""
  }
]
```

Listing 6.2 zeigt den Aufbau eines Filters und seines zugehörigen Templates. Template und Instanz besitzen jeweils einen eindeutigen Namen, um für die Modellierer identifizierbar zu sein. Für die Referenzierung werden dennoch die IDs verwendet, um Zusammenhänge beizubehalten, selbst wenn die Namen bearbeitet werden. Die IDs bleiben den Modellierern verborgen. Templates von Filtern besitzen eine Flag, die erkennbar macht, ob es sich bei den erzeugten Instanzen um Container handelt. Unabhängig des Wertes der Flag ändert sich am Datenmodell des Templates und der Instanzen nichts. Die Auswirkungen der Flag werden in einem folgenden Abschnitt erläutert. Das Template besitzt jeweils ein Array für Eingänge und Ausgänge. Alle Elemente dieser Arrays sind gleich aufgebaut. Sie beinhalten einen Index zur Erkennung, um welchen Eingang, bzw. Ausgang es sich handelt, den erwarteten Datentyp und eine Priorität. Der Datentyp und die Priorität sind

Listing 6.3 Datenmodell einer Verbindung.

```
"connections":[
  {
    "id": 16,
    "outNode": 8,
    "outIndex": "0",
    "inNode": 13,
    "inIndex": "0",
    "qos": 51,
    "datatransfer": 23,
    "address": "/wheel/wetness/1",
    "encrypt": true
  }
]
```

wichtig für die Umsetzung des Konzeptes. Der Index ist die einzige Möglichkeit zur Identifizierung der Schnittstelle im Editor. Jeder Filter besitzt eine Referenz auf einen Knoten und ein Template. In einem Array wird das maximal erlaubte Alter von eingehenden Nachrichten gespeichert, um diese bei Obsoleszenz abweisen zu können. Die Wahl des Templates bestimmt, anhand der Anzahl der Eingänge, die Größe dieses Arrays. Die Zuordnung zwischen erlaubtem Alter und eingehender Verbindung geschieht über die Position im Array und dem Index des Einganges. Zusätzlich wird für jeden Filter eine zweidimensionale, räumliche Koordinate gespeichert, die seine Position im Editor widerspiegelt. Die Felder für die Sichtbarkeit und den Metadaten dienen zur Erfüllung der Konzepte. Handelt es sich bei dem Template um einen Container, existieren die nicht benötigten Datenfelder weiterhin, haben aber keinen Wert. Für Datenquellen und deren Templates ist der Aufbau grundsätzlich ähnlich. Der Unterschied liegt darin, dass die filterspezifischen Datenfelder, wie z.B. die Sichtbarkeit, wegfallen. Zusätzlich besitzen Datenquellen nur einen Ausgang, weshalb dieser nicht in einem Array gespeichert wird und keinen Index benötigt. Ein Beispiel für eine Datenquelle und das zugehörige Template befindet sich in Anhang A.

Der Aufbau einer Verbindung ist in Listing 6.3 angegeben. Diese werden, wie zuvor, über IDs identifiziert. Die verbundenen Komponenten sind über die ID der jeweiligen Komponente und dem Index der betreffenden Schnittstelle verbunden. Verbindungen speichern nicht den Datentyp, den sie übertragen. Dieser ist bei den Komponenten gespeichert und wird vor der Erstellung der Verbindung auf Übereinstimmung geprüft. Passen die Datentypen zusammen, ist es für die Verbindung nicht relevant, welchen Datentyp sie überträgt. In einem weiteren Feld wird die Adresse gespeichert. Diese wird beim Deployment den Komponenten mitgeteilt, um die Endpunkte aufzubauen. Weiterhin werden die Verbindungsqualität, Verbindungstechnologie und Verschlüsselungseinstellung gespeichert.

In Listing 6.4 ist der Aufbau eines Containers abgebildet. Diese werden automatisch angelegt, wenn Filter-Templates mit gesetzter Flag erstellt werden. Container speichern die zugehörigen Eingangs- und Ausgangsfilter, welche jeweils ein eigenes Template benötigen. Sie entsprechen den roten Filtern innerhalb des Containers, welche die Eingänge und Ausgänge repräsentieren. Zusätzlich besitzen Container Arrays für die IDs der Datenquellen, Filter und Verbindungen, die sie abstrahieren. Anhand dieser wird das Innere des Containers aufgebaut. Die Templates der

Listing 6.4 Datenmodell eines Containers.

```
"containers":[
  {
    "id":27,
    "nodeId":33,
    "containerInterfaceInput":29,
    "containerInterfaceOutput":30,
    "datasources":[41],
    "filters":[35],
    "connections":[36, 37, 38, 39, 42]
  }
]

"containerInterfaceTemplates":[
  {
    "id":31,
    "name":"Environment Analysis template_containerInterfaceInputGen",
    "interfaceType":"input",
    "containerTemplate":28,
    "isContainer":false,
    "outputs":[
      {
        "index":"0",
        "type":"Video",
        "priority":""
      },
      [...]
    ]
  },
  [...]
]

"containerInterfaces":[
  {
    "id":29,
    "interfaceType":"input",
    "template":31,
    "position":[-300, 0]
  },
  [...]
]
```

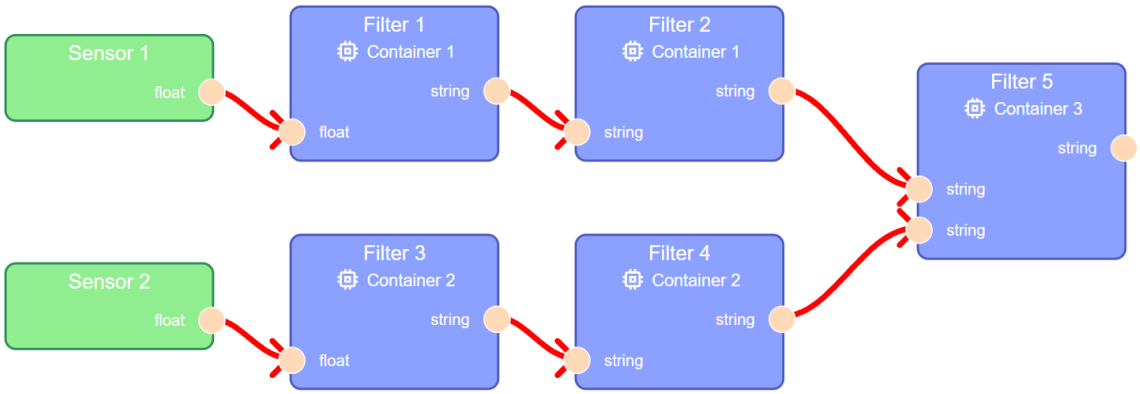


Abbildung 6.15.: Modell der Demonstrations-Umgebung.

Eingangs- und Ausgangsfilter werden automatisch angelegt, wenn ein Filter-Template, welches als Container markiert ist, erzeugt wird. Diese speichern eine Referenz des Filter-Templates, zu dessen Container sie gehören. Zusätzlich wird gespeichert, ob es sich bei dem Template um den Eingang, bzw. Ausgang handelt. Abhängig davon werden die eingehenden bzw. ausgehenden Schnittstellen kopiert. Das bedeutet, nach Anlegen eines Containers, wird ein Eingangsfilter erstellt, welcher die Eingänge des Containers als eigene Ausgänge speichert. Dies geschieht vice versa für den Ausgang. Die Eingangs- und Ausgangsfilter werden erzeugt, sobald ein Filter des als Container markierten Templates angelegt wird. Diese speichern ebenfalls die Angabe, ob es sich um den Eingang oder Ausgang handelt, sowie aus welchem Template sie erzeugt wurden. Zuletzt speichern diese ihre Position, welche mit vordefinierten Werten initialisiert wird, sodass der Eingangsfilter links und der Ausgangsfilter rechts erscheint.

In Anhang A sind Beispiele für eine Verbindungsqualität (A.1), einen QoS (A.2), einen Knoten (A.3) und einer Datenquelle mit Template (A.4) abgebildet. Auf diese wird im Folgenden nicht weiter eingegangen.

6.4. Deployment einer Demonstrations-Umgebung

Zur Demonstration, wie erstellte Modelle eingesetzt werden können, wird ein Deployment simuliert. Es wird eine Umgebung erstellt, die ein Connected Car nachbildet. Grundlage der Umgebung ist eine Virtual Machine (VM) mit installiertem Ubuntu 18.04 [Can21]. Auf dieser ist Docker [Doc21] installiert, und es ist eine Instanz der Multi-purpose Binding and Provisioning Platform (MBP) [SHS+20] hochgefahren. MBP ist eine Plattform für das Management von Internet of Things (IoT)-Umgebungen. Für diese Simulation wird MBP Sensordaten generieren. Das vollständige Deployment wird über Docker Compose gebaut und gestartet.

Das Modell, welches für die Demonstration verwendet wird, ist in Abbildung 6.15 dargestellt. Auf der VM befinden sich mehrere Docker Container, welche die Bestandteile eines Connected Cars widerspiegeln. In einem dieser Container ist ein MQTT-Broker hochgefahren und stellt die in dieser Demonstration verwendete Verbindungstechnologie dar. MBP wird dazu genutzt, Sensoren zu simulieren, indem willkürliche Gleitkommazahlen auf zwei vorgesehene Topics veröffentlicht werden. In einem realen Connected Car ist jede ECU eine separate Umgebung mit

Listing 6.5 Empfangene Nachrichten nach Durchlaufen der Demonstrations-Umgebung.

```
0.18716853
Filter 1
Filter 2
Filter 5

0.54298659
Filter 3
Filter 4
Filter 5
```

anderen Eigenschaften. Docker Container können diese Bedingungen nachstellen. Aus diesem Grund werden drei Container hochgefahren, um die ECUs, bzw. Knoten des Modells, zu simulieren. Auf „Container 1“ und „Container 2“ werden jeweils zwei Filter gehostet. Jeder dieser hat einen Eingang und einen Ausgang. Auf „Container 3“ wird ein Filter mit zwei Eingängen und einem Ausgang bereitgestellt. Auf diese Weise wird demonstriert wie Filter sowohl auf unterschiedlichen als auch auf denselben Knoten miteinander kommunizieren können. Zur Simulation der Filter wird in den Containern jeweils ein eigener Ordner mit der laufenden Anwendung und einer Konfigurationsdatei angelegt. Bei der Konfigurationsdatei handelt es sich um die im Modell spezifizierten Informationen. Die Anwendungen sind Java-Programme, welche zwei MQTT-Clients implementieren. Ein Client dient als Endpunkt für die Eingänge und abonniert die in der Konfigurationsdatei angegebenen Topics für die eingehenden Verbindungen. Der andere Client veröffentlicht Nachrichten auf das dafür vorgesehene Topic. Um den Nachrichtenfluss nachvollziehen zu können, verändert jeder Filter die eingehenden Nachrichten, indem er seinen Namen daran konkateniert.

Bei der Durchführung der Demonstration veröffentlicht MBP Werte auf Topics, die „Filter 1“ und „Filter 3“ abonniert haben. Den Ausgang von „Filter 5“ hat ein weiterer Client abonniert, der nicht Teil des Modells ist. Dieser dient lediglich dazu Nachrichten, die das System passiert haben, zu sammeln und zur Validierung des Ergebnisses auszugeben. In Listing 6.5 sind zwei empfangene Nachrichten abgebildet. Es ist zu erkennen, dass die beiden Nachrichten die getrennten Pfade des Modells durchlaufen haben. Dabei ist „Filter 1“ immer direkt vor „Filter 2“ passiert worden. Gleiches gilt für „Filter 3“ und „Filter 4“. Allein „Filter 5“ bildet für beide Pfade, wie im Modell vorgesehen, die letzte Station.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Konzept zur Modellierung von verteilten Datenpipelines in Software-Defined-Cars vorgestellt. Ein spezieller Fokus lag dabei auf Connected Cars. Das Konzept bewältigt Anforderungen bezüglich Privatsphäre, Sicherheit, Netzwerk, Komplexität, Interoperabilität und Datenverarbeitung, welche in einer Literaturrecherche erarbeitet wurden. Für die Modellierung wird sich an dem Pipes-and-Filters-Konzept orientiert. Datenquellen und Verarbeitungsschritte stellen die Filter dar, welche miteinander verbunden werden können. Die Datenquellen können entweder eigene Sensoren oder Daten aus externen Quellen, wie z.B. anderen Connected Cars und der Cloud sein. Die Verarbeitungsschritte entsprechen Anwendungen, welche im Connected Car bereitgestellt werden. Alle Filter sind individuell konfigurierbar, um ein Deployment zu ermöglichen. Dazu zählt, dass für jede Anwendung, bzw. dem repräsentierenden Filter, der Ort des Deployment bestimmt werden kann. Somit kann die Eigenschaft der Verteiltheit ausgenutzt werden. Für Anwendungen können die ECUs mit den passenden Ressourcen ausgewählt werden. Zudem ermöglicht dies Kontrolle darüber welche Daten das Fahrzeug verlassen und in die Cloud übertragen werden dürfen.

Aufbauend auf dieser Grundlage wurden Konzepte vorgestellt, welche die definierten Anforderungen bewältigen. Zum Beispiel lassen sich mit Verschleierungsfiltern Daten auf eine Weise manipulieren, in der sie ihren tatsächlichen Wert verbergen, aber noch verwendet werden können. Individuell einstellbare Verbindungstechnologien ermöglichen die Modellierung einer idealen Verbindung, um einen effizienten Datenfluss zu gestalten. Der Einsatz von Containerisierung ermöglicht Abstrahierung von technischer Tiefe, sodass die Modelle übersichtlich gestaltet werden können. Die Verwaltung der Modelle ist möglich durch eine Import- und Export-Funktionalität. Dies ermöglicht zusätzlich den Einsatz der Modelle für das Deployment in Connected Cars.

Neben der Entwicklung des Konzeptes wurde zur Demonstration ein implementierter Prototyp vorgestellt. Dabei handelt es sich um ein Modellierungswerkzeug in Form einer Browseranwendung. Die Modellierung wird über eine graphische Benutzeroberfläche ermöglicht. Über Formulare können Datenquellen und Verarbeitungsschritte angelegt, editiert und entfernt werden. Alle Komponenten werden einer Zeichenfläche hinzugefügt. Über D&D können die Komponenten positioniert und Verbindungen zwischen ihnen erstellt werden. Das Modellierungswerkzeug ermöglicht ein nutzerfreundliches Verfahren für das Management der Modelle. Durch die Export-Funktionalität kann ein Deployment erleichtert durchgeführt werden. Dazu wurde ein Deployment in einer Demonstrations-Umgebung beispielhaft vorgeführt.

Ausblick

Das in dieser Arbeit vorgeführte Deployment ist nicht automatisiert und muss stattdessen manuell durchgeführt werden. Bei regelmäßigen Updates würde der Aufwand für Deployments zu groß werden und sich zu einem Flaschenhals entwickeln. Der Nutzen einer flexiblen Modellierung ist begrenzt, wenn ein Deployment nicht im gleichen Maße flexibel umsetzbar ist. Um dies zu bewältigen ist die Integration einer *Continuous Deployment* Pipeline denkbar. Bei einem automatisierten Deployment können innerhalb kurzer Zeit mehrere Updates durchgeführt werden. Vor allem, da es sich bei Connected Cars um sicherheitskritische Anwendungsfälle handelt, ist eine schnelle Handlungsfähigkeit wichtig. Zur Realisierung könnte das Modellierungswerkzeug selbst mit Continuous Deployment ausgestattet werden. Eine andere Option wäre das Modellierungswerkzeug über die Export-Funktionalität an ein weiteres Managementwerkzeug anzubinden, welches unverzüglich die neuen Modelle empfängt und einsetzt.

Der Fokus der Modellierung in dieser Arbeit liegt auf Datenquellen und Verarbeitungsschritten. Datensenken wurden dabei als Verarbeitungsschritte betrachtet. Jedoch ist dies nicht auf die Realität übertragbar, da es sich bei Datensenken z.B. um Datenbanken und Aktuatoren handeln kann. Daher muss das Konzept erweitert werden und diese separat behandeln. Auf diese Weise können z.B. Steuerungselemente zur Ermöglichung von Platooning als Teil des Modells konfiguriert werden. Gleichzeitig ermöglicht dies eine automatisierte Konfiguration von Datensenken beim Deployment.

Literaturverzeichnis

[3GPa] 3GPP. *3GPP Release 14*. URL: https://www.3gpp.org/release-14 (zitiert auf S. 19).

[3GPb] 3GPP. *3GPP Release 15*. URL: https://www.3gpp.org/release-15 (zitiert auf S. 19).

[5G 17] 5G Automotive Association. „An assessment of LTE-V2X (PC5) and 802.11p direct communications technologies for improved road safety in the EU“. In: 2017 (zitiert auf S. 20).

[ABG+17] C. E. Andrade, S. D. Byers, V. Gopalakrishnan, E. Halepovic, D. J. Poole, L. K. Tran, C. T. Volinsky. „Connected cars in cellular network: a measurement study“. In: *Proceedings of the 2017 Internet Measurement Conference*. 2017, S. 235–241 (zitiert auf S. 29, 30).

[AC21] D. Abramov, A. Clark. *Redux*. 2021. URL: https://redux.js.org/ (zitiert auf S. 51, 63).

[ADA19] ADAC. *eCall: Elektronischer Schutzengel im Auto*. 29. Nov. 2019. URL: https://www.adac.de/rund-ums-fahrzeug/unfall-schaden-panne/unfall/ecall/ (zitiert auf S. 25).

[AG14] M. S. Anwer, C. Guy. „A survey of VANET technologies“. In: *Journal of Emerging Trends in Computing and Information Sciences* 5.9 (2014), S. 661–671 (zitiert auf S. 17–19).

[Age15] I. Agenda. „Industrial internet of things: unleashing the potential of connected products and services“. In: *White Paper, in Collaboration with Accenture* (2015) (zitiert auf S. 34).

[Ang13] A. D. Angelica. *Google's self-driving car gathers nearly 1 GB/sec*. 4. Mai 2013. URL: https://www.kurzweilai.net/googles-self-driving-car-gathers-nearly-1-gbsec (zitiert auf S. 17).

[Apa] Apache. *Kafka 3.0 Documentation*. URL: https://kafka.apache.org/documentation/ (zitiert auf S. 48).

[AV10] C. H. Asuncion, M. J. Van Sinderen. „Pragmatic interoperability: A systematic review of published definitions“. In: *IFIP International Conference on Enterprise Architecture, Integration and Interoperability*. Springer. 2010, S. 164–175 (zitiert auf S. 34).

[AZ05] P. Avgeriou, U. Zdun. „Architectural patterns revisited-a pattern language“. In: (2005) (zitiert auf S. 37).

[BBBG19] A. Banks, E. Briggs, K. Borgendale, R. Gupta, Hrsg. *MQTT Version 5.0*. 7. März 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html (zitiert auf S. 42, 48).

[BE12] C. U. Bas, S. C. Ergen. „Ultra-wideband channel model for intra-vehicular wireless sensor networks beneath the chassis: From statistical model to simulations“. In: *IEEE Transactions on Vehicular Technology* 62.1 (2012), S. 14–25 (zitiert auf S. 17, 29).

[BE99] L. Brankovic, V. Estivill-Castro. „Privacy issues in knowledge discovery and data mining“. In: *Australian institute of computer ethics conference*. Citeseer. 1999, S. 89–99 (zitiert auf S. 27).

[BIS+18] L. Bittencourt, R. Immich, R. Sakellariou, N. Fonseca, E. Madeira, M. Curado, L. Villas, L. DaSilva, C. Lee, O. Rana. „The internet of things, fog and cloud continuum: Integration and challenges“. In: *Internet of Things* 3 (2018), S. 134–155 (zitiert auf S. 35).

[Blu21] Bluetooth SIG. *Bluetooth Core Specification*. 13. Juli 2021. URL: https://www.bluetooth.com/specifications/specs/core-specification/ (zitiert auf S. 17).

[BMW10] BMW Group. *Die BMW Group kündigt den Support der neuen iPod Out Funktion für iPhone und iPod touch an.* 9. Juli 2010. URL: https://www.press.bmwgroup.com/austria/article/detail/T0082247DE/die-bmw-group-kuendigt-den-support-der-neuen-ipod-out-funktion-fuer-iphone-und-ipod-touch-an?language=de (zitiert auf S. 32).

[BSC+12] C. Bergenhem, S. Shladover, E. Coelingh, C. Englund, S. Tsugawa. „Overview of platooning systems“. In: *Proceedings of the 19th ITS World Congress, Oct 22-26, Vienna, Austria (2012)*. 2012 (zitiert auf S. 18).

[BVF+10] M. Boban, T. T. Vinhoza, M. Ferreira, J. Barros, O. K. Tonguz. „Impact of vehicles as obstacles in vehicular ad hoc networks“. In: *IEEE journal on selected areas in communications* 29.1 (2010), S. 15–28 (zitiert auf S. 30).

[Can21] Canonical. *Ubuntu*. 2021. URL: https://ubuntu.com/ (zitiert auf S. 67).

[Car] Car Connectivity Consortium. *MirrorLink*. URL: https://mirrorlink.com/ (zitiert auf S. 20).

[CFR+16] C. Cooper, D. Franklin, M. Ros, F. Safaei, M. Abolhasan. „A comparative survey of VANET clustering techniques“. In: *IEEE Communications Surveys & Tutorials* 19.1 (2016), S. 657–681 (zitiert auf S. 17, 18).

[Cha09] R. N. Charette. „This car runs on code“. In: *IEEE spectrum* 46.3 (2009), S. 3 (zitiert auf S. 17).

[CHS+07] L. Cheng, B. E. Henty, D. D. Stancil, F. Bai, P. Mudalige. „Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 GHz dedicated short range communication (DSRC) frequency band“. In: *IEEE Journal on Selected Areas in Communications* 25.8 (2007), S. 1501–1516 (zitiert auf S. 30).

[CM16] R. Coppola, M. Morisio. „Connected car: technologies, issues, future trends“. In: *ACM Computing Surveys (CSUR)* 49.3 (2016), S. 1–36 (zitiert auf S. 13).

[CMK+11] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno et al. „Comprehensive experimental analyses of automotive attack surfaces.“ In: *USENIX Security Symposium*. Bd. 4. 447-462. San Francisco. 2011, S. 2021 (zitiert auf S. 29, 31, 32).

[com18] O.-R. A. D. (committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. Juni 2018. DOI: https://doi.org/10.4271/J3016_201806 (zitiert auf S. 16).

[Cot09] C. D. Cottrill. „Approaches to privacy preservation in intelligent transportation systems and vehicle–infrastructure integration initiative“. In: *Transportation research record* 2129.1 (2009), S. 9–15 (zitiert auf S. 27).

[Del21] Deloitte. *2021 Global Automotive Consumer Study*. 2021. URL: https://www2.deloitte.com/de/de/pages/consumer-industrial-products/articles/global-automotive-consumer-study.html (zitiert auf S. 25).

[DNK+18] A. Dinesh Kumar, K. Naga Renu Chebrolu, S. KP et al. „A Brief Survey on Autonomous Vehicle Possible Attacks, Exploits and Vulnerabilities“. In: *arXiv e-prints* (2018), arXiv–1810 (zitiert auf S. 16, 31).

[Doc21] Docker, Inc. *Docker*. 2021. URL: https://www.docker.com/ (zitiert auf S. 67).

[DS17] R. Dhall, V. Solanki. „An IoT Based Predictive Connected Car Maintenance.“ In: *International journal of interactive multimedia & artificial intelligence* 4.3 (2017) (zitiert auf S. 24).

[EAM18] S. Ercan, M. Ayaida, N. Messai. „How mobile RSUs can enhance communications in VANETs?“ In: *2018 6th International Conference on Wireless Networks and Mobile Communications (WINCOM)*. IEEE. 2018, S. 1–5 (zitiert auf S. 15, 19).

[ECM21] ECMA International. *ECMA-262, 12th edition, June 2021 ECMAScript® 2021 Language Specification*. 30. Juni 2021. URL: https://262.ecma-international.org/12.0/ (zitiert auf S. 51).

[EL14] A. S. Elmaghraby, M. M. Losavio. „Cyber security challenges in Smart Cities: Safety, security and privacy“. In: *Journal of advanced research* 5.4 (2014), S. 491–497 (zitiert auf S. 27).

[EQS17] H. Eichelberger, C. Qin, K. Schmid. „Experiences with the model-based generation of big data pipelines“. In: *Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband* (2017) (zitiert auf S. 21).

[Fac21] Facebook Inc. *React*. 2021. URL: https://reactjs.org/ (zitiert auf S. 51).

[Fle05] FlexRay Consortium. *FlexRay Communications System Protocol Specification Version 2.1 Revision A*. 2005 (zitiert auf S. 17).

[Gen09] C. Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009 (zitiert auf S. 28).

[GES12] J. Gonder, M. Earleywine, W. Sparks. „Analyzing vehicle fuel saving opportunities through intelligent driver feedback“. In: *SAE International Journal of Passenger Cars-Electronic and Electrical Systems* 5.2012-01-0494 (2012), S. 450–461 (zitiert auf S. 16).

[GS18] M. Gupta, R. Sandhu. „Authorization framework for secure cloud assisted connected cars and vehicular internet of things“. In: *Proceedings of the 23nd ACM on symposium on access control models and technologies*. 2018, S. 193–204 (zitiert auf S. 26, 31, 32).

[GSS+18] F. Giust, V. Sciancalepore, D. Sabella, M. C. Filippou, S. Mangiante, W. Featherstone, D. Munaretto. „Multi-access edge computing: The driver behind the wheel of 5G-connected cars“. In: *IEEE Communications Standards Magazine* 2.3 (2018), S. 66–73 (zitiert auf S. 19, 30).

[GZC18] J. Guerrero-Ibáñez, S. Zeadally, J. Contreras-Castillo. „Sensor technologies for intelligent transportation systems“. In: *Sensors* 18.4 (2018), S. 1212 (zitiert auf S. 15, 16, 19).

[HCL04] J.-P. Hubaux, S. Capkun, J. Luo. „The security and privacy of smart vehicles“. In: *IEEE Security & Privacy* 2.3 (2004), S. 49–55 (zitiert auf S. 25–27, 31, 32).

[HER19] D. von Hugo, G. Eichler, T. Rosowski. „A holistic communication network for efficient transport and enhanced driving via connected cars“. In: *International Conference on Innovations for Community Services*. Springer. 2019, S. 11–24 (zitiert auf S. 20).

[HFB09] J. Harri, F. Filali, C. Bonnet. „Mobility models for vehicular ad hoc networks: a survey and taxonomy“. In: *IEEE Communications Surveys & Tutorials* 11.4 (2009), S. 19–41 (zitiert auf S. 18).

[Hir18] P. Hirmer. „Anforderungsbasierte Modellierung und Ausführung von Datenflussmodellen“. Diss. Universität Stuttgart, 2018 (zitiert auf S. 23).

[HKGF09] T. Herpel, B. Kloiber, R. German, S. Fey. „Routing of safety-relevant messages in automotive ECU networks“. In: *2009 IEEE 70th Vehicular Technology Conference Fall*. IEEE. 2009, S. 1–5 (zitiert auf S. 17).

[Hus16] M. Hussain. „Security in connected cars“. In: *Proceedings of the European Automotive Congress EAEC-ESFA 2015*. Springer. 2016, S. 267–275 (zitiert auf S. 31–33).

[HW04] G. Hohpe, B. Woolf. *Enterprise integration patterns: Designing, building, and deploying messaging solutions*. Addison-Wesley Professional, 2004 (zitiert auf S. 45, 48).

[IEE10] IEEE Standards Association. *IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments*. 2010. URL: https://standards.ieee.org/standard/802_11p-2010.html (zitiert auf S. 18).

[IEE19] IEEE Standards Association. *IEEE 1609.12-2019 - IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Identifiers*. 2019. URL: https://standards.ieee.org/standard/1609_12-2019.html (zitiert auf S. 18).

[Int15] International Organization for Standardization. *CAN Standard*. Dez. 2015. URL: https://www.iso.org/standard/63648.html (zitiert auf S. 17).

[IV] T. M. D. W.-.-N. Inhalte, N. Verbraucherzentrale. „CONNECTED CAR NIMMT FAHRT AUF–WOHIN STEUERT DAS AUTO DER ZUKUNFT?“ In: () (zitiert auf S. 16).

[KAEM13] S. Kaisler, F. Armour, J. A. Espinosa, W. Money. „Big data: Issues and challenges moving forward“. In: *2013 46th Hawaii international conference on system sciences*. IEEE. 2013, S. 995–1004 (zitiert auf S. 35).

[KPD+09] M. Kafsi, P. Papadimitratos, O. Dousse, T. Alpcan, J.-P. Hubaux. „VANET connectivity analysis“. In: *arXiv preprint arXiv:0912.5527* (2009) (zitiert auf S. 15, 17, 18).

[KSR+10] K. Koscher, S. Savage, F. Roesner, S. Patel, T. Kohno, A. Czeskis, D. McCoy, B. Kantor, D. Anderson, H. Shacham et al. „Experimental security analysis of a modern automobile“. In: *2010 IEEE Symposium on Security and Privacy*. IEEE Computer Society. 2010, S. 447–462 (zitiert auf S. 29, 31).

[KT19] J. Kawtar, M. Tomader. „Study of connectivity aspect of connected car“. In: *2019 International Conference of Computer Science and Renewable Energies (ICCSRE)*. IEEE. 2019, S. 1–7 (zitiert auf S. 18).

[LCZ+14] N. Lu, N. Cheng, N. Zhang, X. Shen, J. W. Mark. „Connected vehicles: Solutions and challenges“. In: *IEEE internet of things journal* 1.4 (2014), S. 289–299 (zitiert auf S. 17–20, 29, 30).

[LML15] P. Lawson, B. McPhail, E. Lawton. „The Connected Car: Who is in the Driver's Seat? A Study on Privacy and Onboard Vehicle Telematics Technology“. In: (2015) (zitiert auf S. 26, 27, 31).

[LNRT06] M. B. Line, O. Nordland, L. Røstad, I. A. Tøndel. „Safety vs security?“ In: 2006 (zitiert auf S. 31).

[Mat21] Material-UI SAS. *Material UI*. 2021. URL: https://mui.com/ (zitiert auf S. 51).

[MBO20] A. R. Munappy, J. Bosch, H. H. Olsson. „Data Pipeline Management in Practice: Challenges and Opportunities“. In: *International Conference on Product-Focused Software Process Improvement*. Springer. 2020, S. 168–184 (zitiert auf S. 20, 21, 33).

[MCC+16] A. Mahmood, C. Casetti, C.-F. Chiasserini, P. Giaccone, J. Harri. „Mobility-aware edge caching for connected cars“. In: *2016 12th Annual Conference on Wireless On-demand Network Systems and Services (WONS)*. IEEE. 2016, S. 1–8 (zitiert auf S. 19, 35).

[Mcc17] B. Mccluskey. „Connected cars–the security challenge [Connected Cars Cyber Security]“. In: *Engineering & Technology* 12.2 (2017), S. 54–57 (zitiert auf S. 29, 32, 33).

[MLKS18] A. C. Marosi, R. Lovas, Á. Kisari, E. Simonyi. „A novel IoT platform for the era of connected cars“. In: *2018 IEEE International Conference on Future IoT Technologies (Future IoT)*. IEEE. 2018, S. 1–11 (zitiert auf S. 19, 25).

[NFM16] T. Nawrath, D. Fischer, B. Markscheffel. „Privacy-sensitive data in connected cars“. In: *2016 11th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE. 2016, S. 392–393 (zitiert auf S. 26, 29).

[NHCB18] Y. Ni, J. He, L. Cai, Y. Bo. „Data uploading in hybrid V2V/V2I vehicular networks: Modeling and cooperative strategy“. In: *IEEE Transactions on Vehicular Technology* 67.5 (2018), S. 4602–4614 (zitiert auf S. 18, 19).

[NJ05] A. Nadeem, M. Y. Javed. „A performance comparison of data encryption algorithms“. In: *2005 international Conference on information and communication technologies*. IEEE. 2005, S. 84–89 (zitiert auf S. 32).

[NLF+15] C. B. Nielsen, P. G. Larsen, J. Fitzgerald, J. Woodcock, J. Peleska. „Systems of systems engineering: basic concepts, model-based techniques, and research directions“. In: *ACM Computing Surveys (CSUR)* 48.2 (2015), S. 1–41 (zitiert auf S. 34).

[OEB18] N. Ostern, A. Eßer, P. Buxmann. „Capturing Users' Privacy Expectations To Design Better Smart Car Applications.“ In: *PACIS*. 2018, S. 97 (zitiert auf S. 26–28).

[PKÅ+20] P. Pelliccione, E. Knauss, S. M. Ågren, R. Heldal, C. Bergenhem, A. Vinel, O. Brunnegård. „Beyond connected cars: A systems of systems perspective“. In: *Science of Computer Programming* 191 (2020), S. 102414 (zitiert auf S. 13, 15, 18, 19, 24, 26, 29, 32–34).

[QWY10] F. Qu, F.-Y. Wang, L. Yang. „Intelligent transportation spaces: vehicles, traffic, communications, and beyond“. In: *IEEE Communications Magazine* 48.11 (2010), S. 136–142 (zitiert auf S. 29).

[RBOW20] A. Raj, J. Bosch, H. H. Olsson, T. J. Wang. „Modelling Data Pipelines“. In: *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2020, S. 13–20 (zitiert auf S. 20, 21).

[RBP19] E. G. Renart, D. Balouek-Thomert, M. Parashar. „An edge-based framework for enabling data-driven pipelines for iot systems“. In: *2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE. 2019, S. 885–894 (zitiert auf S. 35).

[RCL14] R. Rezaei, T. K. Chiew, S. P. Lee. „An interoperability model for ultra large scale systems“. In: *Advances in Engineering Software* 67 (2014), S. 22–46 (zitiert auf S. 34).

[RMFR18] P. Ram, J. Markkula, V. Friman, A. Raz. „Security and privacy concerns in connected cars: A systematic mapping study“. In: *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*. IEEE. 2018, S. 124–131 (zitiert auf S. 33).

[RSGJ13] K. Raman, A. Swaminathan, J. Gehrke, T. Joachims. „Beyond myopic inference in big data pipelines“. In: *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2013, S. 86–94 (zitiert auf S. 13, 20, 21).

[SAE] SAE. *SAE J1939 Standard*. URL: https://www.sae.org/standardsdev/groundvehicle/j1939a.htm (zitiert auf S. 17).

[SAE09] SAE. *Dedicated Short Range Communications (DSRC) Message Set Dictionary J2735_200911*. 19. Nov. 2009. URL: https://www.sae.org/standards/content/j2735_200911/ (zitiert auf S. 31).

[SCFJ03] H. Schulzrinne, S. L. Casner, R. Frederick, V. Jacobson. *RTP: A Transport Protocol for Real-Time Applications*. Juli 2003. URL: https://datatracker.ietf.org/doc/html/rfc3550 (zitiert auf S. 42).

[SHS+20] A. C. F. da Silva, P. Hirmer, J. Schneider, S. Ulusal, M. T. Frigo. „Mbp: Not just an iot platform“. In: *2020 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*. IEEE. 2020, S. 1–3 (zitiert auf S. 67).

[Sin19] N. Sinha. „Emerging technology trends in vehicle-to-everything connectivity“. In: *2019 Wireless Telecommunications Symposium (WTS)*. IEEE. 2019, S. 1–12 (zitiert auf S. 17–20).

[SJ08] R. Shaw, B. Jackman. „An introduction to FlexRay as an industrial network“. In: *2008 IEEE International Symposium on Industrial Electronics*. IEEE. 2008, S. 1849–1854 (zitiert auf S. 17).

[SLY+16] H. Seo, K.-D. Lee, S. Yasukawa, Y. Peng, P. Sartori. „LTE evolution for vehicle-to-everything services“. In: *IEEE communications magazine* 54.6 (2016), S. 22–28 (zitiert auf S. 15, 19, 20, 28, 30).

[SPE17] R. Spalazzese, P. Pelliccione, U. Eklund. „INTERO: an interoperability model for large systems“. In: *IEEE Software* 37.3 (2017), S. 38–45 (zitiert auf S. 34).

[Sta14] Statista Research Department. *Typische Lebensdauer von Autos in Deutschland nach Automarken*. 28. Juli 2014. URL: https://de.statista.com/statistik/daten/studie/316498/umfrage/lebensdauer-von-autos-deutschland/ (zitiert auf S. 29).

[Sto21] V. Stoliarov. *Rete.js*. 17. Juli 2021. URL: https://github.com/retejs/rete (zitiert auf S. 51).

[TKR21] M. N. Tahir, M. Katz, U. Rashid. „Analysis of VANET wireless networking technologies in realistic environments“. In: *2021 IEEE Radio and Wireless Symposium (RWS)*. IEEE. 2021, S. 123–125 (zitiert auf S. 20).

[Wis16] T. Wisman. „eCall and the Quest for Effective Protection of the Right to Privacy“. In: *Eur. Data Prot. L. Rev.* 2 (2016), S. 59 (zitiert auf S. 26).

[WL07] C. Wu, Y. Liu. „Queuing network modeling of driver workload and performance“. In: *IEEE Transactions on Intelligent Transportation Systems* 8.3 (2007), S. 528–537 (zitiert auf S. 16).

[WZX+16] D. Wu, L. Zhu, X. Xu, S. Sakr, D. Sun, Q. Lu. „Building pipelines for heterogeneous execution environments for big data processing“. In: *IEEE Software* 33.2 (2016), S. 60–67 (zitiert auf S. 20, 23).

[YKK+19] I. Yaqoob, L. U. Khan, S. A. Kazmi, M. Imran, N. Guizani, C. S. Hong. „Autonomous driving cars in smart cities: Recent advances, requirements, and challenges“. In: *IEEE Network* 34.1 (2019), S. 174–181 (zitiert auf S. 16, 19).

[YLSS21] H. Yousuf, M. Lahzi, S. A. Salloum, K. Shaalan. „Systematic review on fully homomorphic encryption scheme and its application“. In: *Recent Advances in Intelligent Systems and Smart Applications* (2021), S. 537–551 (zitiert auf S. 28).

[Zig15] ZigBee Alliance. *ZigBee Specification*. 5. Aug. 2015. URL: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf (zitiert auf S. 17).

[ZNM+16] E. Zabler, R. Neul, W.-M. Müller, U. Konzelmann, T. Schmidt-Sandte, B. Cramer, P. Weiberle, U. Papert, P. Spoden, G. Noetzel et al. „Sensoren“. In: *Sensoren im Kraftfahrzeug*. Springer, 2016, S. 112–171 (zitiert auf S. 16).

[ZNY+17] K. Zhang, J. Ni, K. Yang, X. Liang, J. Ren, X. S. Shen. „Security and privacy in smart city applications: Challenges and solutions“. In: *IEEE Communications Magazine* 55.1 (2017), S. 122–129 (zitiert auf S. 26, 28).

[ZYW+18] L. Zhu, F. R. Yu, Y. Wang, B. Ning, T. Tang. „Big data analytics in intelligent transportation systems: A survey“. In: *IEEE Transactions on Intelligent Transportation Systems* 20.1 (2018), S. 383–398 (zitiert auf S. 16).

Alle URLs wurden zuletzt am 03. 11. 2021 geprüft.

A. Weitere Teile des Datenmodells

Listing A.1 Datenmodell einer Verbindungsqualität.

```
"datatransfers":[
  {
    "id":23,
    "name":"MQTT",
    "color":"red"
  }
]
```

Listing A.2 Datenmodell eines QoS.

```
"qoss":[
  {
    "id":51,
    "name":"At most once"
  }
]
```

Listing A.3 Datenmodell eines Knotens.

```
"physicalNodes":[
  {
    "id":0,
    "name":"High Res Node",
    "address":"192.168.1.1",
    "visibility":"private"
  }
]
```

Listing A.4 Datenmodell einer Datenquelle.

```
"datasourceTemplates":[
  {
    "id": 3,
    "name": "Wheel Wetness Sensor",
    "type": "Wetness"
  }
]

"datasources":[
  {
    "id": 8,
    "name": "Wheel 1",
    "template": 3,
    "position": [-200,0],
    "metadata": "",
    "address": "192.168.2.1"
  }
]
```

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift