DEFEnD

DEvelopment For SEcured
Autonomous Driving

Schlussbericht zum Teilvorhaben

Jürgen Becker, Tobias Dörr und Timo Sandmann

16. November 2021

Verbundprojekt:	DEvelopment For SEcured Autonomous Driving (DEFEnD)
Teilvorhaben:	Modellbasierte Entwurfsmethodik für Safety-Security-Co-Design im Kontext des automatisierten Fahrens
Verbundpartner:	Karlsruher Institut für Technologie (KIT)
Projektleitung:	Prof. Dr.-Ing. Dr. h. c. Jürgen Becker
Projektlaufzeit:	1. September 2018 – 31. Mai 2021
Förderkennzeichen:	16KIS0886

Das diesem Bericht zugrunde liegende Vorhaben wurde mit Mitteln des Bundesministeriums für Bildung und Forschung unter dem Förderkennzeichen 16KIS0886 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Inhaltsverzeichnis

1 Gesamtüberblick

1.1 Aufgabenstellung

Ziel des Teilvorhabens war die Entwicklung einer modellbasierten Methodik zum Entwurf sicherer eingebetteter Systeme für zukünftige autonome Fahrzeuge. Hierfür sollte ein ganzheitliches Konzept entwickelt werden, das „Security-by-Design“ mit konstruktiven Ansätzen zur Erfüllung von Safety-Anforderungen kombiniert. Ein besonderer Fokus lag dabei auf der Betrachtung zukünftiger Elektrik/Elektronik-Architekturen (E/E-Architekturen), für die eine zunehmende Konsolidierung verschiedener Funktionen auf wenigen, aber leistungsfähigen Steuergeräten (z. B. auf Basis heterogener Mehrkernprozessoren) prognostiziert wird [7]. Grundsätzlich ist davon auszugehen, dass ein sequentieller Entwurf, bei dem die relevanten Anforderungen an Safety und Security nachgelagert betrachtet werden, erheblich höhere Aufwände erzeugt, als wenn dies bereits in frühen Entwurfsphasen und gesamtheitlich erfolgt.

Vor diesem Hintergrund war es die Aufgabe des Teilvorhabens, entsprechende Anforderungen an Entwurfswerkzeuge zu identifizieren und diese in Metamodelle, die angestrebte Entwicklungsmethodik sowie Tools zur praktischen Umsetzung der Methodik zu überführen. Abschließend sollte die Methodik unter Verwendung der entwickelten Tools auf ein repräsentatives Szenario aus dem Kontext des autonomen Fahrens angewandt werden, um eine anschauliche Demonstration und eine realitätsnahe Validierung der erarbeiteten Konzepte zu ermöglichen.

1.2 Voraussetzungen zur Durchführung des Vorhabens

In der Hightech-Strategie der Bundesregierung stellen Informations- und Kommunikationstechnologien (IKT) ein Themenfeld essentieller Bedeutung dar. DEFEnD als Verbundprojekt hat Lösungen im Bereich der IT-Sicherheit für autonomes Fahren geschaffen und damit aktiv zur digitalen Transformation der Mobilität beigetragen. Im Rahmen der Fördermaßnahme „KMU-innovativ: Informations- und Kommunikationstechnologien (IKT)“ hat der Verbund aus KMUs und Forschungseinrichtungen insbesondere die Förderschwerpunkte „Sichere und vertrauenswürdige IKT-Systeme“ sowie „IT-Sicherheit in Anwendungsfeldern“ vorangetrieben.

Die durch den Zuwendungsempfänger grundsätzlich verfolgten Forschungsschwerpunkte reichen vom Entwurf und der Implementierung neuartiger Hardwarekomponenten (wie z. B. effiziente Hardwarebeschleuniger) über die Entwicklung optimierter Systemarchitekturen mit speziellen Safety- oder Security-Eigenschaften bis hin zur Konzipierung werkzeuggestützter Methoden für einen solchen Architekturentwurf. Er hat diese Expertise eingebracht, um die in Abschnitt 1.1 beschriebenen Aufgaben als Teil des Gesamtvorhabens zu absolvieren. Dabei lag der Fokus insbesondere auf der Betrachtung hardware- und implementierungsnaher Heraus-

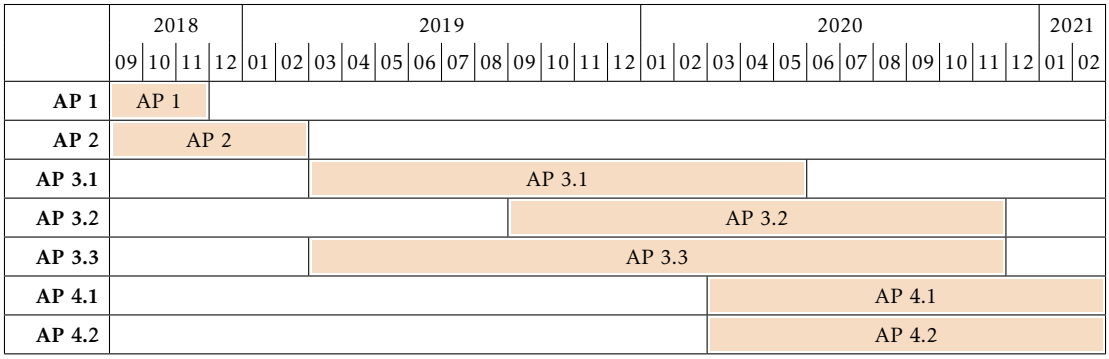

Abbildung 1: Gantt-Diagramm der Arbeitspakete mit Beteiligung des KIT

forderungen. Für auf höheren Abstraktionsebenen angesiedelte Safety- und Security-Analysen konnte auf die Arbeit der Partner im Verbund zurückgegriffen werden. Umgekehrt hatten auf diesen Abstraktionsebenen tätige Projektpartner die Möglichkeit, die konkrete Umsetzung ausgewählter Anforderungen an die Methoden und Werkzeuge des KIT zu delegieren.

1.3 Planung und Ablauf des Vorhabens

Das durch den Verbund verfolgte Vorhaben war in die folgenden Arbeitspakete gegliedert:

- **AP 1**: Szenarien und Security-Anforderungen für autonomes Fahren
- **AP 2**: Security-Requirements-Engineering und Auswirkungsanalyse
- **AP 3**: Methoden und Werkzeuge
  - **AP 3.1**: Security by Design – Modellbasierte Methoden und Konzepte
  - **AP 3.2**: Methoden und Prozesse für Security-Validierung und -Test
  - **AP 3.3**: Tooling für Security-Methoden und -Konzepte
- **AP 4**: Evaluation und Demonstratoren
  - **AP 4.1**: Implementierung des Demonstrators
  - **AP 4.2**: Validierung der Projektergebnisse
- **AP 5**: Analyse, Modellierung und Überführung rechtlich-technischer Vorgaben

Der Zuwendungsempfänger hatte die Leitung von AP 1 sowie AP 4 inne und war darüber hinaus in AP 2 und AP 3 aktiv. Abbildung 1 zeigt den ursprünglich, d. h. zu Projektbeginn, gültigen Zeitplan der Arbeitspakete mit KIT-Beteiligung.

Aufgrund zeitlicher Verzögerungen auf Verbundebene von ein bis drei Monaten ist die Laufzeit des Gesamtprojekts über den im Gantt-Diagramm dargestellten Zeitraum um drei Monate, d. h. bis zum 31. Mai 2021, verlängert worden.

In seiner Rolle als Leiter von AP 4 hat der Zuwendungsempfänger an der Verlängerung teilgenommen, um dem Abschluss dieses Arbeitspakets organisatorisch begleiten und die Ergebnisdokumentation durchführen zu können.

1.4 Stand der Technik zu Projektbeginn

Die jüngsten Entwicklungen im Automobilsektor lassen einen klaren Trend zur Zentralisierung der E/E-Architektur erkennen [7, 10]. Damit steigt das Potential für unerwünschte Interferenzen zwischen verschiedenen Subsystemen und die Erfüllung von Anforderungen wie Echtzeitfähigkeit oder Zuverlässigkeit kann sich zu einer anspruchsvollen Aufgabe entwickeln [13]. Darüber hinaus waren die vergangenen Jahre durch eine stetige Zunahme der Konnektivität von Fahrzeugen geprägt. Insbesondere externe Schnittstellen erhöhen die Wahrscheinlichkeit der Existenz von Schwachstellen, die für gezielte Angriffe auf die E/E-Architektur genutzt werden können [9]. Ein mahnendes Beispiel für einen solchen Angriff auf ein Serienfahrzeug ist 2015 von Miller und Valasek publiziert worden [18]. Da ein erfolgreicher Angriff bei enger Integration von Funktionen auch Safety-relevante Vorfälle (z. B. in Form physischen Schaden an Insassen) hervorrufen kann [24], stellt die Kombination von Zentralisierung und steigender Konnektivität ein herausforderndes Spannungsfeld dar.

Explizit mit dem Ziel der On-Chip-Isolation entwickelte Hardwareeinheiten, wie sie z. B. in [20] und [25] beschrieben werden, können zur logischen Segregation verschiedener Subsysteme auf komplexen Mehrkernprozessoren eingesetzt werden. Kommerziell verfügbare Plattformen wie der Zynq UltraScale+ MPSoC von Xilinx [6] verfügen über Einheiten dieser Art. Hierbei handelt es sich um Mechanismen, die grundsätzlich zur Behandlung der oben beschriebenen Problematik geeignet, in der Regeln aber nicht in modellbasierte Ansätze zur automatisierten Erfüllung von Safety- oder Security-Anforderungen integriert sind.

Ansätze zur modellbasierten Analyse sowie automatischen Synthese sicherheitskritischer eingebetteter Systeme sind grundsätzlich bekannt [11, 21], mit den neuartigen Mechanismen zur Isolation aber bisher nicht umfassend kombiniert worden. Der Zuwendungsempfänger hat diesen Umstand als Lücke im Stand der Technik identifiziert.

1.5 Angabe verwendeter Fachliteratur

Veröffentlichungen, an die des Teilvorhaben angeknüpft hat, sind im Literaturverzeichnis dieses Berichts bzw. als Referenzen der eigenen Veröffentlichungen [1–3] aufgeführt.

Darüber hinaus sind insbesondere die folgenden Standards eingesetzt worden:

- SAE J3061 (2016) – *„Cybersecurity Guidebook for Cyber-Physical Vehicle Systems“*
- SAE AS5506C (2017) – *„Architecture Analysis & Design Language (AADL)“*
- ISO 26262:2018 – *„Road vehicles — Functional safety“*
- ISO/SAE DIS 21434:2020 – *„Road vehicles — Cybersecurity engineering“*

1.6 Zusammenarbeit mit anderen Stellen

Im Rahmen des Teilvorhabens ist eine intensive Zusammenarbeit mit den Konsortialpartnern erfolgt. Über diese Kooperation hinaus hat der Zuwendungsempfänger im direkten Projektkontext mit keinen weiteren Stellen zusammengearbeitet.

2 Ergebnisse des Teilvorhabens

2.1 Szenarien und Anforderungen für autonomes Fahren (AP 1)

Fokus von AP 1 war die Beschreibung von Use-Cases, Szenarien und Anforderungen für das betrachtete Umfeld. In enger Abstimmung mit den Projektpartnern ist der betrachtete Automatisierungsgrad auf Stufe 5 der SAE-Definition (gemäß J3016) festgelegt worden.

Um die damit einhergehende Komplexität beherrschbar zu machen, ist mit den Partnern eine Definition der relevanten Akteure und Use-Cases entwickelt worden. Abbildung 2 zeigt einen Ausschnitt der dabei spezifizierten Use-Cases als Use-Case-Diagramm in UML-Notation.

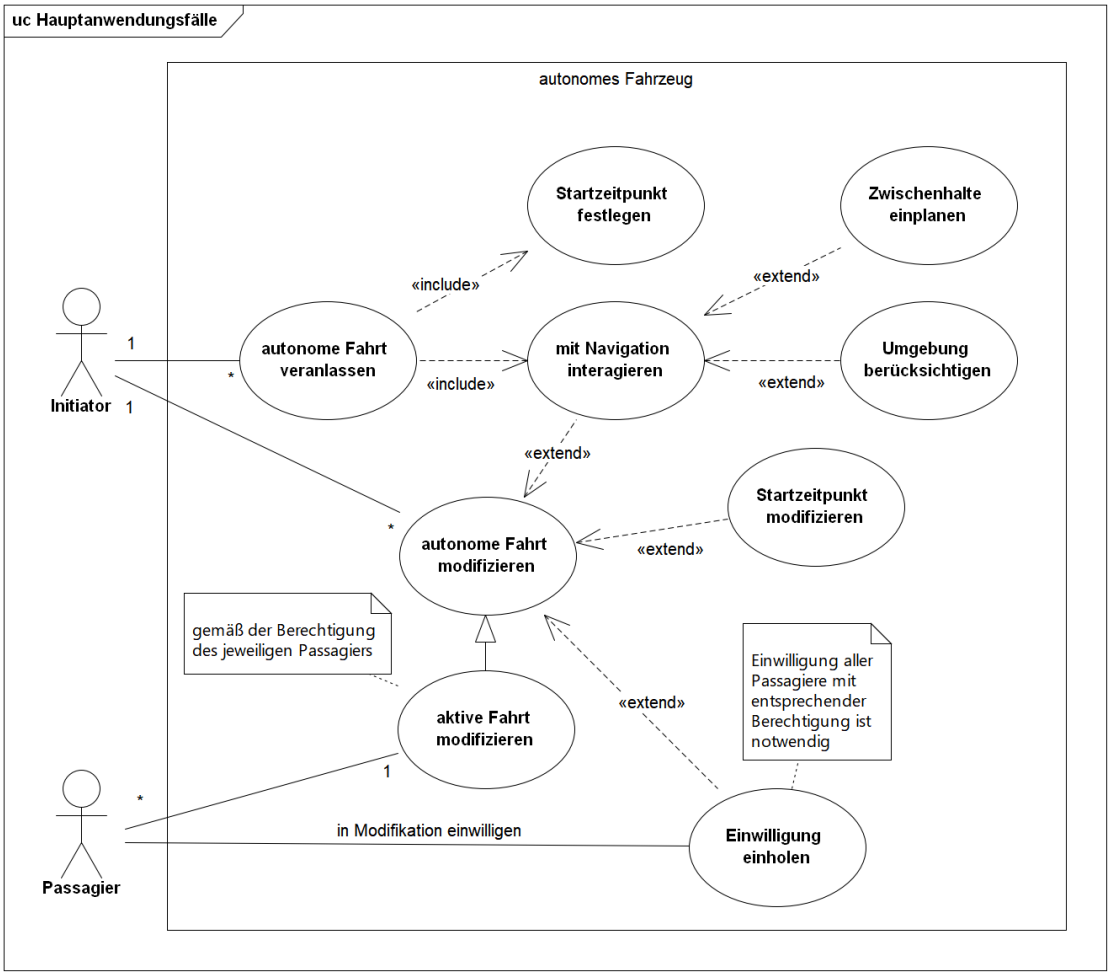

Abbildung 2: Auswahl an betrachteten Use-Cases aus Sicht der Akteure

Um den repräsentativen Charakter der Use-Cases zu verifizieren, sind sie beispielhaft auf ein Privatfahrzeug und einen Linienbus angewandt worden. Hierbei konnte gezeigt werden, dass sie zur Behandlung dieser verschiedenen Szenarien grundsätzlich geeignet sind.

Darauf aufbauend sind die zur Umsetzung der Use-Cases nötigen Kommunikationsvorgänge und Schnittstellen identifiziert worden. Als hierbei betrachtete Systemgrenze wurde das autonome Fahrzeug mit allen unmittelbar an das Fahrzeug angeschlossenen Anlagen (z. B. eine Bluetooth-, Mobilfunk- oder WiFi-Schnittstelle) definiert.

Für das verfolgte Teilvorhaben war es anschließend wichtig, den Use-Cases konkrete Systemfunktionen gegenüberzustellen. Darüber hinaus mussten auch nicht durch Use-Cases erfasste, aber für das System trotzdem essentielle Funktionen modelliert werden.

Im nächsten Schritt sind daher vom System (d. h. dem autonomen Fahrzeug mit allen seinen unmittelbaren Schnittstellen) zu erbringende Funktionen identifiziert und unter Verwendung des von der EnCo Software GmbH ins Projekt eingebrachte SOX-Werkzeug modelliert worden. Diese lassen sich fünf übergeordneten Funktionskategorien zuordnen: Fahrtenverwaltung, Fahrtenabwicklung, Rollenverwaltung, Infrastrukturschnittstelle und Herstellerschnittstelle.

Während die unter Fahrtenverwaltung gruppierten Funktionen stark mit dem in Abbildung 2 dargestellten Ausschnitt der Use-Cases zusammenhängen, stehen die der Fahrtenabwicklung zugeordneten Funktionen mit der von Akteuren weitestgehend unabhängigen Durchführung einer autonomen Fahrt in Verbindung.

Die im Rahmen dieser Systemfunktionen über externe Schnittstellen ausgetauschten Informationen wurden identifiziert und bezüglich ihrer Safety- bzw. Security-Relevanz analysiert. Beispielsweise könnte ein unbefugter Zugriff auf die Fahrtenhistorie, als dem Initiator als Teil der Fahrtenverwaltung zur Verfügung steht, aus Sicht der Passagiere vergangener Fahrten schützenswerte Daten preisgeben.

Abschließend und insbesondere in Vorbereitung auf AP 4 wurden gemeinsam mit den Projektpartnern repräsentative Fahrszenarien identifiziert. Für das hier beschriebene Teilvorhaben wurde eines dieser Fahrszenarien („Rechtsabbiegen an einer innerstädtischen Kreuzung unter Berücksichtigung einer kommunizierenden Ampel und kreuzendem Fußverkehr“) als primär betrachtetes Szenario ausgewählt.

2.2 Security-Requirements-Engineering und Auswirkungsanalyse (AP 2)

Ziel der Arbeiten in AP 2 war es, die für den Systementwurf relevanten Anforderungen abzuleiten, diese hinsichtlich möglicher Wechselwirkungen von Safety- und Security-Maßnahmen zu analysieren und eine Teilmenge zur weiteren Betrachtung in AP 3 und AP 4 auszuwählen.

Hierbei war der Zuwendungsempfänger stets bestrebt, die abgeleiteten Anforderungen so aufzubereiten, dass Safety- und Security-Aspekte im Anschluss auf eine möglichst einheitliche Art und Weise betrachtet werden können. Außerdem wurde aus den in Abschnitt 1.1 genannten Gründen insbesondere auf solche Anforderungen an das Systemdesign hingearbeitet, die vor allem die implementierungsnahen Abstraktionsebenen des Designs (wie den Einsatz und die Konfiguration konkreter Steuergeräte oder Mikrocontroller) umfassen.

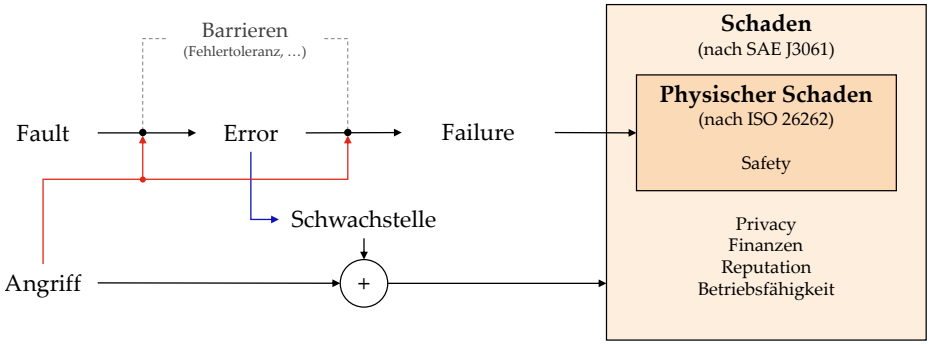

Abbildung 3: Schadensbegriffe und -ursachen aus ISO 26262 und SAE J3061

Dafür sind die ISO 26262 und die SAE J3061 zunächst auf ihren Betrachtungsrahmen und die aus ihren Vorgaben hervorgehenden Anforderungen an das Systemdesign untersucht worden.

Bezüglich der ISO 26262 ist festzuhalten, dass sie ausschließlich Safety-Auswirkungen (d. h. physischen Schaden) durch Defekte (*faults*) der E/E-Architektur berücksichtigt. Darüber hinaus umfasst der Betrachtungsrahmen der SAE J3061 auch Auswirkungen auf die Privacy, die Reputation, die Betriebsfähigkeit oder finanzieller Art. Ursächlich werden hierbei allerdings ausschließlich Security-relevante Aspekte (auf Basis von Schwachstellen) berücksichtigt.

Abbildung 3 stellt diese Erkenntnisse grafisch dar. Hierbei fasst die „Schwachstelle“ alle systematischen Fehler, die zur Durchführung eines Angriffs genutzt werden können, zusammen. Demgegenüber bezieht sich die als „Fault“ aufgeführte Situation auf einen „abnormalen Umstand“ nach ISO 26262. Die in der Abbildung blau und rot eingezeichneten Relationen ergeben sich aus der im Rahmen von AP 2 durchgeführten Analyse von Abhängigkeiten zwischen Safety und Security. So ist es insbesondere möglich, dass ein Angriff nicht auf die Nominalfunktionalität eines Systemelements, sondern auf eine explizit aus Safety-Gründen hinzugefügte Barriere ausgeführt wird und somit nur indirekt zu einem physischen Schaden führt. Analog dazu ist es möglich, dass ein zufälliger Defekt eine Schwachstelle erzeugt.

Über die Szenarien in Abbildung 3 hinaus ist es denkbar, dass eine während der Entwicklung hinzugefügte Barriere eine Schwachstelle überhaupt erst entstehen lässt (wie z. B. ein angreifbarer redundanter Prozessor) oder dass ein Mechanismus, der explizit zur Vermeidung von Schwachstellen hinzugefügt wird, physische Schäden begünstigt (z. B. ein Zugriffsschutzmodul, das einen essentiellen Datenpfad unterbricht).

Aufgrund der Vielzahl der verschiedenen Abhängigkeiten ist die explizite Berücksichtigung aller dieser Aspekte nicht zielführend. Von großem Interesse sind daher Mechanismen, die es während des Entwicklungsprozesses ermöglichen, einzelne Schwachstellen und Fehlerpropagationspfade effizient und ohne unerwünschte Seiteneffekte zu schließen. Eine genauere Analyse der Standards hat ergeben, dass insbesondere die logische Isolation physikalisch dicht integrierter Subsysteme einen solchen Mechanismus darstellt. Gemäß der Vorgaben aus der ISO 26262 ist es zur Erfüllung der Safety-Anforderungen oft unerlässlich, kritische Subsysteme von weniger kritischen Subsystemen zu isolieren [14]. Ebenso führt die SAE J3061 eine solche Isolation als essentielles Mittel zur Erfüllung vieler praktisch relevanter Security-Anforderungen auf [23].

Zur weiteren Betrachtung im Teilvorhaben wurden daher genau solche Mechanismen zur logischen Isolation ausgewählt, die gemäß Abschnitt 1.4 an den Stand der Technik moderner Mehr-

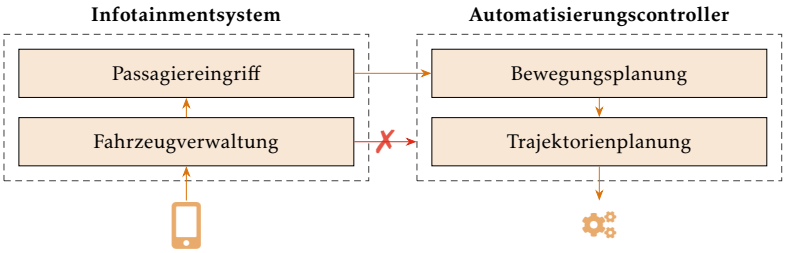

Abbildung 4: Beispielarchitektur mit zulässigen und unzulässigen Informationsflüssen

kernprozessoren anknüpfen und zur Nutzung in einer modellbasierten Entwicklungsmethodik mit zuvor genannten Ziel grundsätzlich geeignet sind.

Abschließend ist im Rahmen von AP 2 ein beispielhafter Ausschnitt einer E/E-Architektur entwickelt worden, um die durch logische Isolation erreichbaren Garantien anhand eines unterbundenen Angriffs auf die Safety eines autonomen Fahrzeugs zu veranschaulichen [2]. Ausgangspunkt für diesen Ausschnitt sind die Fahrtenverwaltung und Fahrtenabwicklung nach Definition in Abschnitt 2.1, wobei die genaue Ausgestaltung der Fahrtenabwicklung unter Verwendung der Ausführungen in [26] weiter detailliert wurde.

Abbildung 4 zeigt ein vereinfachtes Blockdiagramm dieser Architektur. Sie besteht aus zwei leistungsfähigen Steuergeräten, die durch jeweils einen modernen Mehrkernprozessor umgesetzt sind. Die beiden Steuergeräte realisieren wie in der Abbildung dargestellt ein Infotainmentsystem sowie eine als „Automatisierungscontroller“ bezeichnete Komponente, die alle im direkten Zusammenhang mit der Fahrtenabwicklung stehenden Aufgaben erfüllt.

Es wird angenommen, dass das Infotainmentsystem über eine externe Schnittstellen (wie etwa Bluetooth) verfügt und es Nutzern so erlaubt, durch die Verwendung eines geeigneten Endgeräts mit dem System zu interagieren. Insbesondere ist es Passagieren (gemäß der Use-Cases aus Abschnitt 2.1) möglich, über diese Schnittstelle in die aktive autonome Fahrt einzugreifen. Um solch einen Eingriff technisch umzusetzen, ist ein Informationsfluss vom Endgerät des Nutzers über Bewegungs- und Trajektorienplanung zum Antriebsstrang bzw. zur Lenkung nötig. Die orange hervorgehobenen Pfeile in Abbildung 4 veranschaulichen diesen Informationsfluss.

Zusätzlich wird in der entwickelten Architektur davon ausgegangen, dass vom Automatisierungscontroller empfangene Nachrichten (z. B. über einen CAN-Bus) grundsätzlich in der Lage sind, die Trajektorie des Fahrzeugs direkt zu beeinflussen. Um zu verhindern, dass Angreifer über externe Schnittstellen des Infotainmentsystems einen physischen Schaden hervorrufen, ist eine strikte Kontrolle der vom Infotainmentsystem an den Automatisierungscontroller versandten Nachrichten erforderlich. In der Beispielarchitektur wird dies primär durch eine hinreichend zuverlässige Auslesung der als „Passagiereingriff“ bezeichneten Funktion sichergestellt. Durch die Realisierung des Infotainmentsystems auf Basis eines modernen Mehrkernprozessors ist allerdings nicht ausgeschlossen, dass Angreifer diese Sicherheitsfunktion umgehen, um den rot dargestellten Informationsfluss anzustoßen.

Dies lässt sich durch eine sorgfältig ausgelegte, logische On-Chip-Isolation, wie sie die anvisierte Entwurfsmethodik anstrebt, per Design unterbinden.

2.3 Methoden und Werkzeuge (AP 3)

Ziel der Arbeiten in AP 3 war es, die in den vorausgehenden Arbeitspaketen erlangten Resultate zu nutzen, um Methoden und Werkzeuge für ein Vorgehen zum „Safety and Security by Design“ nach Abschnitt 1.1 zu entwickeln.

Dafür ist in AP 3.1 zunächst ein Konzept zum modellbasierten E/E-Architekturentwurf entwickelt und durch die formale Definition von Metamodellen, die in den nächsten Schritten in die entwickelten Werkzeuge zu integrieren sind, ergänzt worden.

Die im Rahmen von AP 3.2 durchgeführten Arbeiten hatten das Ziel, den Safety und Security verzahnenden Entwicklungsprozess selbst auf Basis der entwickelten Metamodelle zu definieren. Hierbei war es insbesondere von Bedeutung, diesen vom KIT definierten Prozess in den vom Konsortium anvisierten Gesamtprozess zu integrieren.

Abschließend wurden die in den vorausgehenden Teilarbeitspaketen entwickelten Metamodelle und Methoden in AP 3.3 in Form eines Java-basierten Softwarewerkzeugs implementiert. Das so entstandene Tool zur modellbasierten On-Chip-Isolation ist grundsätzlich eigenständig verwendbar, fügt sich konzeptionell aber nahtlos als hardware- bzw. implementierungsnaher Schritt in den Gesamtprozess des Konsortiums ein.

Orthogonal zu der oben genannten Struktur sind im Rahmen des Teilvorhabens zwei Ausprägungen der entwickelten Methodik entstanden. Die erste Ausprägung wird für die Zwecke dieses Berichts als „Codesign-Methodologie“ bezeichnet. Sie stellt das primäre Arbeitsergebnis dar, zeichnet sich durch eine hohe Flexibilität aus und wird in Abschnitt 2.3.1 genauer erläutert. Bei der zweiten Ausprägung handelt es sich um eine explizit auf Security ausgerichtete Weiterentwicklung der Codesign-Methodologie. Obwohl sie sich in einem gewissen Umfang ebenfalls zur verzahnten Betrachtung von Safety und Security eignet, ist ihre Flexibilität zugunsten einer vereinfachten Behandlung von Angriffen und Schwachstellen im Security-Sinne eingeschränkt. Sie wird in diesem Bericht daher als „Security-Methodologie“ bezeichnet. Abschnitt 2.3.2 enthält eine genauere Beschreibung der Anpassungen, die zu dieser Ausprägung geführt haben.

Eine detaillierte Beschreibung beider Ausprägungen liegt in Form von Konferenzbeiträgen und einem Journalartikel vor. Die in [1] enthaltenen Ausführungen erläutern dabei den Zwischenstand der Codesign-Methodologie, wie er im Januar 2020 zur Verfügung stand. Die finale Version der Codesign-Methodologie wird in [2] behandelt. Die zur Entwicklung der Security-Methodologie angewandten Anpassungen sowie die Auswirkungen dieser Anpassungen auf den Entwicklungsprozess sind in [3] beschrieben. Abschnitt A.1 im Anhang setzt die englischen Begriffe aus [1–3] mit den im vorliegenden Bericht genutzten Übersetzungen in Bezug.

2.3.1 Codesign-Methodologie

Bei der Codesign-Methodologie handelt es sich um ein modellbasiertes Konzept, das es Entwicklern erlaubt, anvisierte On-Board-Netzwerke in Kombination mit Anforderungen hinsichtlich erlaubter sowie zu unterbindender Informationsflüsse in diesem Netzwerk zu beschreiben. Basierend auf einer solchen Beschreibung werden automatisch Implementierungsartefakte erzeugt, die auf die Erfüllung der Informationsflussvorgaben hinwirken. Gleichzeitig erfolgt eine automatische Verifikation, die sicherstellt, dass die erzeugten Artefakte zur Erfüllung der Anforderungen hinreichend sind. Unter der Voraussetzung, dass diese Verifikation erfolgreich

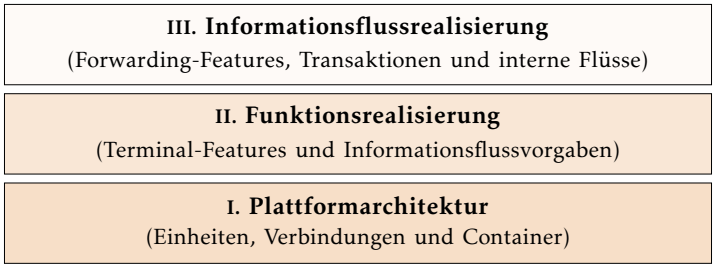

Abbildung 5: Überblick über die Modellschichten und ihre zentralen Elemente

ist, erhält der Entwickler beim Einsatz der Artefakte gewisse Informationsflussgarantien, die er zur Auslegung eines aus Safety- und Security-Perspektive sicheres System nutzen kann.

Bei den generierten Implementierungsartefakten handelt es sich um Konfigurationen für Hardwareeinheiten zur On-Chip-Isolation, wie sie in Abschnitt 1.4 beschrieben wurden. Im Folgenden wird eine solche Isolationseinheit wie in [1–3] als „access protection unit“ bezeichnet und als APU abgekürzt.

Metamodell und domänenspezifische Sprache (DSL)

Um dem Nutzer der Methodologie die Beschreibung der nötigen Informationen zu ermöglichen, ist ein dreischichtiges Metamodell (siehe Abbildung 5) entwickelt worden. Eine konkrete Instanz dieses Metamodells wird als „Modellinstanz“ bezeichnet.

Die Plattformarchitektur einer Modellinstanz (Schicht I) beschreibt die Struktur des betrachteten On-Board-Netzwerks. „Einheiten“ repräsentieren Hardwaremodule wie einen Prozessorkern (inkl. der relevanten Ausführungsumgebung für Applikationen). „Verbindungen“ stellen On-Chip- oder Off-Chip-Kommunikationskanäle zwischen einzelnen Einheiten dar. „Container“ sind Elemente, die zur hierarchischen Strukturierung des Designs dienen. So ist es beispielsweise sinnvoll, alle auf einem bestimmten integrierten Schaltkreises vereinten Elemente als Einheiten oder Verbindungen des gleichen Containers zu modellieren. Ein als Wurzelcontainer ausgezeichneter Container enthält alle Elemente der Plattformarchitektur.

Durch die Funktionsrealisierung (Schicht II) werden alle vom anvisierten System zu erbringenden Funktionen (wie z. B. eine konkrete Applikation, die als Prozess auf einem Betriebssystem ausgeführt wird) modelliert und den jeweils ausführenden Einheiten zugeordnet. Dies geschieht durch die Definition von „Terminal-Features“, die jeweils eine Funktion repräsentieren und auf genau eine in der Plattformarchitektur modellierte Einheit abgebildet werden. Zusätzlich werden auf dieser Modellschicht alle Informationsflussvorgaben spezifiziert. Hierfür ist die Auflistung aller akzeptierten und aller erforderlichen Informationsflüsse zwischen Terminal-Features durch den Entwickler notwendig.

Durch die Spezifikation der Informationsflussrealisierung (Schicht III) modelliert der Entwickler, wie die erforderlichen Informationsflüsse im On-Board-Netzwerk technisch umgesetzt werden sollen. Hierfür können „Forwarding-Features“ eingesetzt werden, die z. B. die Funktionalität eines CAN-Controllers erfassen. Darüber hinaus können angedachte Kommunikationskanäle zwischen Features beschrieben werden. Abhängig davon, ob es sich dabei um eine Kommunikation innerhalb einer Einheit oder über eine Verbindung zwischen zwei Einheiten handelt, handelt es sich dabei um „interne Flüsse“ oder „Transaktionen“.

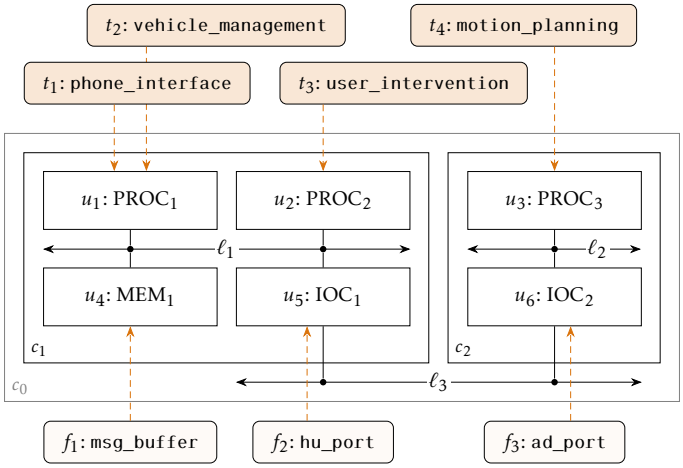

Abbildung 6: Formalisierung der Beispielarchitektur unter verwendung des Metamodells

Über die Modellschichten hinweg ist es möglich, einzelne Einheiten und Features als „zuverlässig“ zu deklarieren. Mit einer solchen Deklaration gehen gewisse Anforderungen einher, die der Entwickler eigenständig sicherstellen muss. Durch die automatische Berücksichtigung dieser Anforderungen im weiteren Prozess ist es der Methodologie möglich, weniger pessimistische Annahmen bezüglich der potentiell möglichen Informationsflüsse zu treffen.

Das dreischichtige Metamodell ist im Rahmen des Teilvorhabens formal, d. h. unter Verwendung mathematischer Mengen, Relationen und Funktionen, spezifiziert worden. So lässt sich eine konkrete Plattformarchitektur beispielsweise als 7-Tupel darstellen:

$$M_I = (U,\ L,\ C,\ c_0,\ \varphi_C,\ \varphi_L,\ \delta_U)$$

Hierbei stellen U, L und C die Mengen der Einheiten, Verbindungen und Container dar. $c_0 \in C$ repräsentiert den Wurzelcontainer. $\varphi_C : U \cup L \cup (C \setminus \{c_0\}) \to C$ ordnet jedes Element mit Ausnahme des Wurzelcontainers dem Container zu, der dieses Element enthält. $\varphi_L : L \to \mathcal{P}(U)$ ordnet jede Verbindung den Einheiten zu, die an diese Verbindung angeschlossen sind. Abschließend gibt $\delta_U : U \to \{0,1\}$ die Zuverlässigkeit jeder Einheit an.

Abbildung 6 zeigt die Plattformarchitektur sowie Terminal- und Forwarding-Features, wie sie als Fortführung des mit Abbildung 4 begonnenen Beispiels denkbar sind. Die beiden Mehrkernprozessoren sind hier als dedizierte Container (c_1 und c_2) beschrieben und über eine Verbindung wie etwa einen CAN-Bus verbunden (ℓ_3). In der formalen Modellinstanz wäre ℓ_3 beispielsweise per $\varphi_C(\ell_3) = c_0$ dem Wurzelcontainer zugeordnet und per $\varphi_L(\ell_3) = \{u_5, u_6\}$ als an die I/O-Controller der beiden integrierten Schaltkreise angeschlossen modelliert.

Zur besseren Handhabbarkeit der Methodologie wurde darüber hinaus eine domänenspezifische Sprache (DSL) entwickelt, die eine für den Nutzer leicht zu verfassende Beschreibung von Modellinstanzen ermöglicht. Für weitere Details, insbesondere zur formalen Definition der übrigen Modellschichten, sei an dieser Stelle auf die Ausführungen in [2] verwiesen.

Die oben beschriebenen Arbeiten zum Metamodell und der DSL für die Codesign-Methodologie sind im Rahmen von AP 3.1 absolviert worden.

Definition des Entwicklungsprozesses

Anschließend wurde in AP 3.2 der Prozess, der die Codesign-Methodologie definiert, entwickelt. Abbildung 7 veranschaulicht in einem ein Flussdiagramm, wie dieser durch den Entwickler eines eingebetteten Systems zu nutzen ist. Ausgangspunkt für die automatisierten Prozessschritte ist eine Modellinstanz.

Bei der Modellierung der Funktionsimplementierung (II) muss dafür unter anderem eine Spezifikation der Informationsflussvorgaben vorgenommen werden. Während die Spezifikation der erforderlichen Informationsflüsse lediglich notwendig ist, um automatisiert zu verifizieren, dass die resultierenden APU-Konfigurationen nicht zu streng sind, steht die Spezifikation der akzeptierten Informationsflüsse in direktem Zusammenhang mit den zugrundeliegenden Anforderungen an Safety und Security. Konkret müssen Erkenntnisse aus zuvor durchgeführten Safety- und Security-Betrachtungen in eine Spezifikation der erlaubten Informationsflüsse umgesetzt werden. Im definierten Entwicklungsprozess wird davon ausgegangen, dass alle nicht explizit als akzeptierte Informationsflüsse zwischen einem geordneten Paar von Terminal-Features zuverlässig unterbunden werden müssen.

Eine gegebene Modellinstanz wird anschließend durch zwei automatisierte Schritte verarbeitet: Informationsflussanalyse und die plattformspezifische Generierung.

Bei der **Informationsflussanalyse** werden plattformspezifische Aspekte (wie die Existenz einer konkreten APU) vernachlässigt. Stattdessen wird auf Basis der spezifizierten Transaktionen eine abstrahierte APU-Konfiguration erzeugt, die so strikt wie möglich ist, aber die Durchführung aller dieser Transaktionen ermöglicht. Ziel des Schritts ist es dann, zu verifizieren, dass

1. alle erforderlichen Informationsflüsse nominal möglich sind und durch eine Kombination von Transaktionen und internen Flüssen realisiert wurden, aber gleichzeitig
2. jeder potentiell mögliche Informationsfluss gemäß der Informationsflussanforderungen erlaubt („akzeptiert“) ist.

Anders formuliert wird in diesem Schritt sichergestellt, dass die Modellinstanz eine Generierung von APU-Konfiguration ermöglicht, die in Summe (1) weder zu strikt noch (2) unzureichend zur

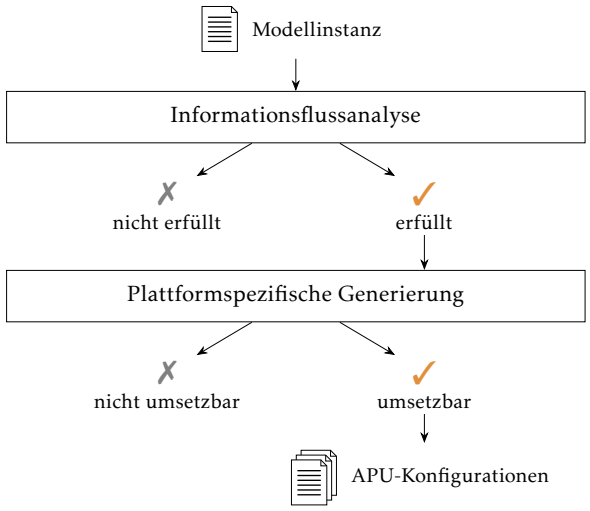

Abbildung 7: Flussdiagramm der entwickelten Methodologie

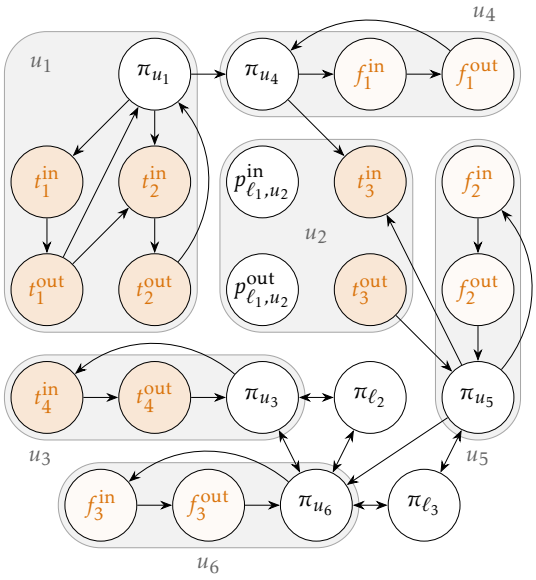

Abbildung 8: Beta-Graph (G_β) für die beispielhafte Modellinstanz

Erfüllung der Safety- und Security-Anforderungen sind. Für jeden dieser Fälle ist ein graphenbasierter Algorithmus entwickelt worden. Diese beiden Algorithmen wandeln die gegebene Modellinstanz in einen Alpha-Graphen (G_α) bzw. einen Beta-Graphen (G_β) um. Diese erfassen alle nominal spezifizierten Informationsflüsse bzw. alle zur Laufzeit potentiell möglichen Informationsflüsse unter der Annahme, dass der vollständige Satz generierter APU-Konfigurationen auf der Hardware eingesetzt wird.

Abbildung 8 zeigt beispielsweise den Beta-Graphen, der in [2] für die Fortführung des vorausgehenden Beispiels generiert worden ist. Die als t_i^{in} und t_i^{out} bezeichneten Knoten repräsentieren die Eingänge und Ausgänge des Terminal-Features $t_i \in T$, während gerichtete Kanten einen potentiell möglichen Informationsfluss zwischen zwei direkt miteinander in Verbindung stehenden Systemkomponenten wiedergeben. Durch die Betrachtung aller gerichteten Pfade zwischen einem Paar von Terminal-Features lässt sich so ermitteln, ob ein direkter oder indirekter Informationsfluss von den Ausgängen eines Terminal-Features zu den Eingängen eines anderen Terminal-Features potentiell möglich ist. Durch den Algorithmus werden alle potentiell vom Terminal-Feature $t \in T$ erreichbaren Informationssenken automatisch bestimmt und in Form der Funktion

$$\beta : T \rightarrow \mathcal{P}(T)$$

zurückgeliefert. Diese werden dann mit der expliziten Spezifikation der akzeptierten Informationsflüsse abgeglichen. Sind alle potentiell möglichen Flüsse aus β akzeptiert, ist dieser Teil der Informationsflussanalyse erfolgreich. Ist dies nicht der Fall, muss der Entwickler eine Anpassung der Modellinstanz vornehmen. Für weitere Details und eine Beschreibung des Vorgehens, das auf den Alpha-Graphen angewandt wird, sei an dieser Stelle wieder auf [2] verwiesen.

Führt die Informationsflussanalyse einer gegebenen Modellinstanz zu einem positiven Ergebnis, geht der Prozess automatisch zur **plattformspezifischen Generierung** über. Ziel dieses Schritts ist es, die zuvor abstrakt generierten APU-Konfigurationen in konkrete, auf der realen Hardware einsetzbare Implementierungsartefakte zu überführen. Hierfür sind die detaillierten Eigenschaften der betrachteten Ausführungsplattformen (inkl. verfügbarer APUs) von großer Re-

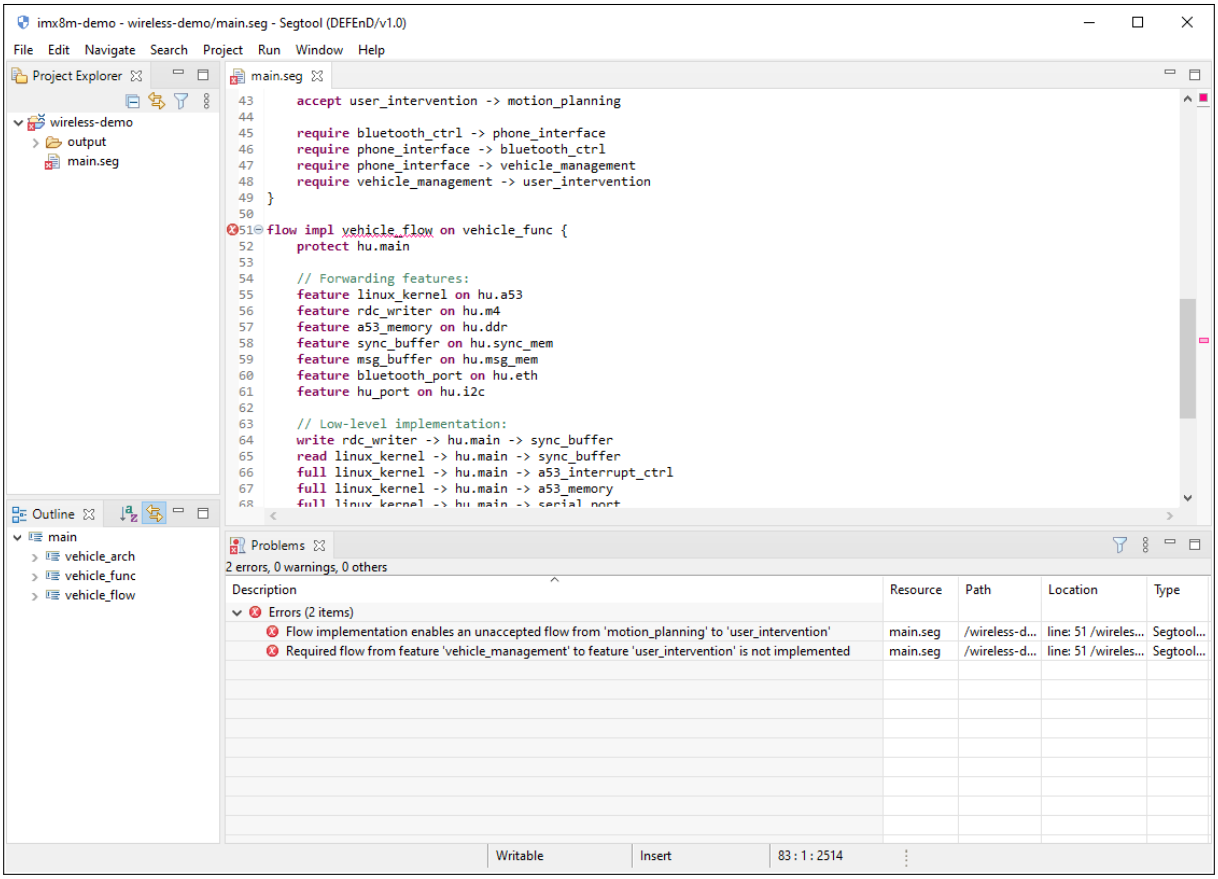

Abbildung 9: Segtool bei der Bearbeitung einer Modellinstanz

levanz und können nicht weiter vernachlässigt werden. Um diese zusätzliche Information in den Prozess einzubringen, wurde das Metamodell aus AP 3.1 entsprechend erweitert. Container, die zum Beispiel einen konkreten Mehrkernprozessor repräsentieren, können unter Verwendung dieser Erweiterung auf konkrete Hardwareplattformen (wie z. B. den i.MX 8M von NXP) abgebildet werden.

Aufgrund der hohen Variabilität in der Auslegung realer APUs ist nicht garantiert, dass eine abstrakte APU-Konfiguration aus dem vorausgehenden Schritt auf der für einen Container ausgewählten Plattform tatsächlich umsetzbar ist. Wie in Abbildung 7 dargestellt, erfolgt daher zunächst eine Prüfung, ob die gegebene Modellinstanz unter Berücksichtigung aller relevanten APUs in reale Konfigurationen überführbar ist. Fällt diese positiv aus, terminiert der Prozess mit der automatischen Generierung aller angeforderten APU-Konfigurationen. Andernfalls ist wieder eine Anpassung der Modellinstanz nötig.

Implementierung als Softwarewerkzeug

Aufbauend auf den Ergebnissen von AP 3.1 und AP 3.2 ist in AP 3.3 die technische Umsetzung des zuvor beschriebenen Metamodells, der DSL sowie eines Softwarewerkzeugs zur Durchführung des beschriebenen Entwicklungsprozesses erfolgt.

Alle im Teilvorhaben entwickelten Implementierungen sind Java-basiert und setzen das Eclipse Modeling Framework (EMF) zur Abbildung des Metamodells ein. Für die Realisierung der DSL wurde das Xtext-Framework [12] verwendet. Die entwickelte Toolchain selbst liegt als Kom-

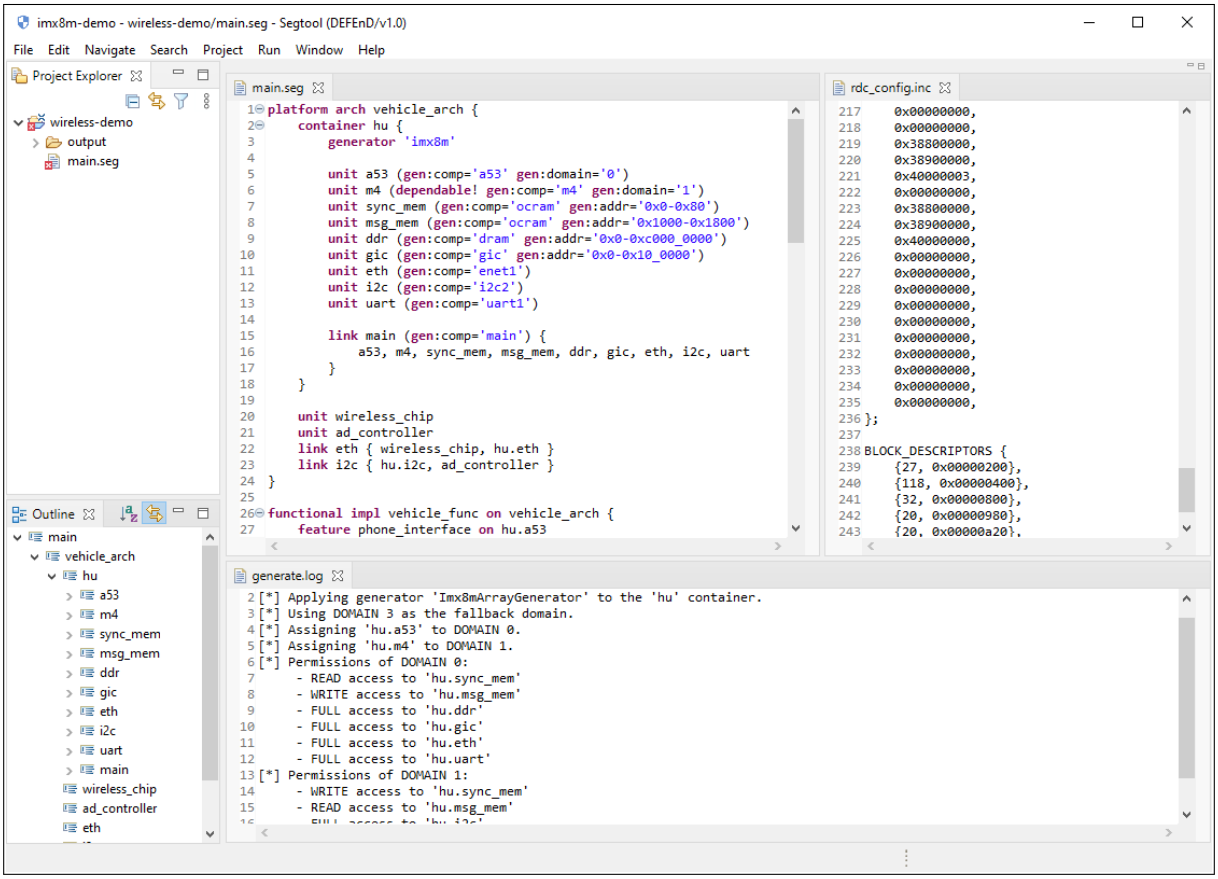

Abbildung 10: Segtool nach der plattformspezifischen Generierung für den i.MX 8M

mandozeilenanwendung sowie als Eclipse-basierten Anwendung mit grafischem Frontend vor. Abbildung 9 zeigt einen Screenshot dieses als „Segtool“ bezeichneten Frontends bei der Bearbeitung der DSL-Formulierung aus [2].

Bei der Entwicklung wurde auf eine flexible und erweiterbare Auslegung der plattformspezifischen Generierung hingearbeitet. So ist es möglich, zusätzliche Backends für diesen Prozessschritt in Form einer gekapselten Java-Klasse zu realisieren. Beispielhaft ist im Rahmen des Teilvorhabens ein plattformspezifischer Generator für den i.MX 8M von NXP entwickelt worden. Abbildung 10 zeigt, wie es möglich ist, einen Container der Modellinstanz dem plattformspezifischen Generator unter Verwendung der DSL zuzuordnen (siehe Zeile 3 des Codes). Wird eine so mit dem Generator verknüpfte DSL-Formulierung im Segtool gespeichert, erfolgt die automatische Generierung einer Konfiguration (`rdc_config.inc`) für die APU dieses Chips.

2.3.2 Security-Methodologie

Die in Abschnitt 2.3.1 beschriebene Methodologie, die insbesondere bis Juni 2020 entwickelt wurde, konnte im Rahmen von AP 4 frühzeitig evaluiert werden. Die dabei erlangten Erkenntnisse sind im Anschluss in das hier beschriebene AP 3 zurückgeflossen und wurden für Optimierungen der Methodologie genutzt. Obwohl sich bei dieser Evaluation ergeben hat, dass die Codesign-Methodologie auf den Entwurf von On-Board-Netzwerken grundsätzlich effizient anwendbar ist, wurde ein potentielles Skalierbarkeitsproblem identifiziert: Modellinstanzen, die durch eine signifikante Anzahl von Transaktionen, internen Flüssen und unzuverlässigen Mo-

dellelementen (wie ein $u \in U$ mit $\delta_U(u) = 0$) charakterisiert sind, können eine Vielzahl potentiell möglicher Informationsflüsse hervorrufen. Wie zuvor beschrieben, müssen alle diese Informationsflüsse explizit als erlaubt spezifiziert worden sein, damit die Informationsflussanalyse ein positives Resultat zurückliefert. Im ungünstigsten Fall wächst der Aufwand, der beim Entwickler für die Aufstellung dieser Liste akzeptierter Informationsflüssen zwischen Terminal-Features anfällt, quadratisch mit der Zahl an Terminal-Features in der Modellinstanz an.

Um auf eine Verbesserung der Skalierbarkeit hinzuwirken, sind im Rahmen von AP 3 weiterführende Konzepte untersucht worden. Als eine vielversprechender Maßnahme hat sich dabei die direkt ins Metamodell integrierte Erfassung ausgewählter Security-Anforderungen erwiesen. Auf diese Weise ist in Zusammenarbeit mit der ERNW Enno Rey Netzwerke GmbH die bereits zu Beginn von Abschnitt 2.3 genannte „Security-Methodologie“ entwickelt worden.

Hierbei handelt es sich um eine direkte Weiterentwicklung der Codesign-Methodologie. Sowohl die Struktur des dreischichtigen Metamodells aus Abbildung 5 als auch der in Abbildung 7 dargestellte Fluss der Prozessschritte ist grundsätzlich weiter gültig. Die entscheidende Änderung, die zur Security-Methodologie führt, ist im Metamodell auf der Schicht der Funktionsrealisierung (II) vorgenommen worden und bezieht sich auf die Spezifikation der akzeptierten Informationsflüsse. Während diese bei der Codesign-Methodlogie durch eine explizite Auflistung geordneter Paare von Terminal-Features vorgenommen werden musste, werden die akzeptierten Informationsflüsse in der Security-Methodologie automatisch auf Basis einer modellbasierten Annotation von Security-Anforderungen ermittelt.

Durch die erweiterte Methodologie können sowohl Anforderungen hinsichtlich Vertraulichkeit („Confidentiality“) als auch hinsichtlich Integrität („Integrity“) behandelt werden.

Von einem Toolchainnutzer, der die Security-Methodologie einsetzt, wird zunächst erwartet, ein zur Betrachtung des Systems genutztes Vertraulichkeits- bzw. Integritätsframework zu spezifizieren. Bei einem Framework dieser Art handelt es sich um eine konkrete Menge sogenannter Security-Levels. Ein solches Level wird im Confidentiality-Fall auch als Vertraulichkeitslevel bezeichnet und quantifiziert die Vertraulichkeit von Informationen oder Komponenten im System. Analog dazu wird es im Integrity-Fall als Integritätslevel bezeichnet und quantifiziert die Integrität von Informationen oder Komponenten. Bei der Definition dieser Levels wurde insbesondere auf das Bell-LaPadula-Modell für Confidentiality [4] sowie des Biba-Modell für Integrity [5] zurückgegriffen. Zwischen den Security-Levels eines konkreten Frameworks besteht eine partielle Ordnung in Form einer Dominanzrelation.

Durch den Entwickler erfolgt eine Annotation der Terminal-Features mit Security-Levels. Hierbei können die internen Ausgänge eines Terminal-Features mit einer Vertraulichkeitsanforderung versehen werden, während seine internen Eingänge mit der von diesem Feature bereitgestellten Vertraulichkeit annotiert wird. Um die Vertraulichkeitsanforderung abzuleiten, betrachtet der Entwickler die internen und externen Assets, über die das Terminal-Feature verfügt. Für die Bestimmung der bereitgestellten Vertraulichkeit sind die an das Terminal über externe Schnittstellen verbundenen Gefahrenquellen (wie ein potentieller menschlicher Angreifer) relevant. Für den Integritätsfall wird analog dazu vorgegangen, wobei die Rollen der geforderten und bereitgestellten Levels vertauscht sind: An die internen Ausgänge des Features wird die vom Feature bereitgestellte Integrität annotiert, während die internen Eingänge mit der von diesem Feature gestellten Integritätsanforderung versehen werden (siehe Abbildung 11).

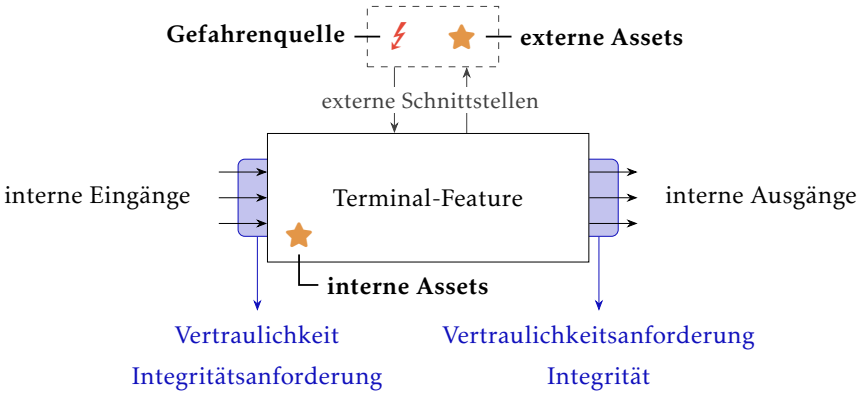

Abbildung 11: Annotation von Security-Levels an ein Terminal-Feature

Während der Informationsflussanalyse werden Security-Levels (d. h. Vertraulichkeitsanforderungen und Integritäten) entlang der Kanten des Beta-Graphen propagiert. Konvergieren dabei zwei oder mehr potentiell mögliche Informationsflüsse, wird das resultierende Security-Level so gewählt, dass ein Vertraulichkeitslevel grundsätzlich nicht abnimmt, während ein Integritätslevel grundsätzlich nicht zunimmt. Abschließend wird geprüft, ob alle an interne Eingänge propagierten Vertraulichkeitsanforderungen in Einklang mit der entsprechenden Vertraulichkeit stehen. Analog dazu müssen die propagierten Inegritäten mit der entsprechenden Integritätsanforderung kompatibel sein. Sind beide Bedingungen erfüllt, liefert die Informationsflussanalyse ein positives Ergebnis zurück. Andernfalls ist nicht auszuschließen, dass eine Gefahrenquelle (wie ein menschlicher Angreifer) einen Informationsfluss hervorrufen kann, der die Anforderungen nach Confidentiality bzw. Integrity nicht erfüllt.

Alle im Anschluss an die Informationsflussanalyse durchgeführten Schritte können von der Codesign-Methodologie unverändert übernommen werden. Insbesondere die plattformspezifische Generierung ist unabhängig von den Informationsflussvorgaben, sodass an dieser Stelle eine vollständige Kompatibilität beider Methodologien erreicht werden konnte.

Für eine detaillierte Beschreibung der Security-Methodologie, insbesondere durch die exakte Definition der Struktur eines Security-Level-Frameworks und einer Erläuterung des darauf aufbauenden Algorithmus zur Propagation von Security-Levels entlang der Kanten von G_β, sei an dieser Stelle auf die Ausführungen in [3] verwiesen.

2.4 Evaluation und Demonstration (AP 4)

Plangemäß hatte das KIT die Leitung von AP 4 inne. Hierbei wurde insbesondere auf die Definition eines Demonstratorkonzepts hingewirkt, das die Arbeiten aller im Verbundprojekt tätigen Partner anhand eines praxisrelevanten Beispiels veranschaulicht und in Bezug setzt. Darüber hinaus hat das KIT die Realisierung eines konkreten Teils des gemeinsamen Konzepts übernommen. Die folgenden Ausführungen beziehen sich auf diese Realisierung.

2.4.1 Demonstratorkonzept

Das entwickelte Demonstratorkonzept basiert auf der Codesign-Methodologie und veranschaulicht deren Fähigkeiten. Ausgangspunkt hierfür ist der in Abbildung 12 dargestellte Ausschnitt

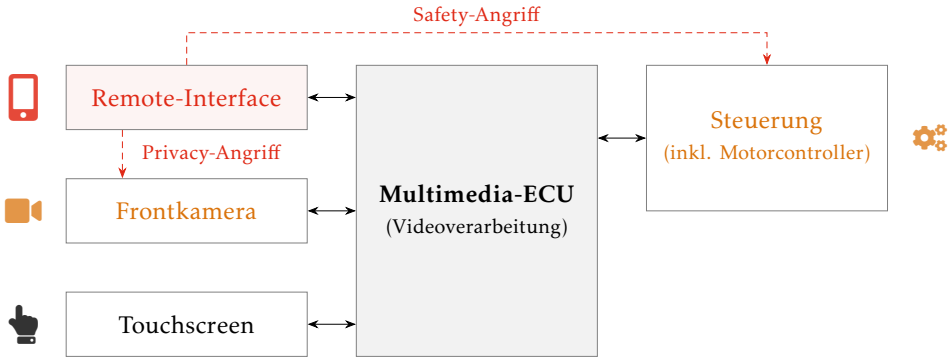

Abbildung 12: Gesamtüberblick des betrachteten Demonstratorszenarios

einer E/E-Architektur, der unter Verwendung der Ergebnisse aus AP 3 gegen Angriffe auf Safety und Privacy, jeweils im Sinne von [23], geschützt werden soll.

Die primär betrachtete Komponente ist hierbei die „Multimedia-ECU“, die eine Komponente auf Basis eines modernen Mehrkernprozessors mit APU-Unterstützung darstellt. Sie wird zur Erfüllung aller im Zusammenhang mit Videoverarbeitung stehenden Ausgaben eingesetzt. Insbesondere soll die ECU von der Frontkamera gelieferte Videosequenzen empfangen und diese (nach einer eventuellen Vorverarbeitung) an verschiedene Komponenten weiterleiten:

- Für die Zwecke des Infotainmentsystems sollen Kameraframes sowohl über das im Fahrzeug befindliche Touchscreen als auch über kabellos verbundene Endgeräte angezeigt werden. Hierzu müssen sie an diese Komponenten weitergeleitet werden.
- Um eine Trajektorienplanung für autonome Fahrten zu ermöglichen, sollen Kameraframes zur Umfelderkennung herangezogen werden. Hierfür müssen die Frames an die als „Steuerung“ bezeichnete Komponente weitergeleitet werden.

Grundsätzlich ist davon auszugehen, dass aufgezeichnete Kameraframes personenbezogene Daten enthalten (z. B. durch erkennbare Gesichter). Aus diesem Grund wird die beispielhafte Anforderung aufgestellt, dass Daten dieser Art anonymisiert werden müssen, bevor sie das autonome Fahrzeug verlassen (→ Privacy-Anforderung). Dies soll erreicht werden, indem jeder von der Multimedia-ECU das Remote-Interface weitergeleitete Kameraframe anonymisiert worden sein muss. Analog hierzu wird angenommen, dass die als „Steuerung“ bezeichnete Komponente einen direkten Einfluss auf die zurückgelegte Trajektorie hat. Dies führt zu der beispielhaften Anforderungen, dass die von der Multimedia-ECU an die Steuerung weitergeleiteten Frames gegen bewusste Manipulationen über das Remote-Interface geschützt werden müssen (→ Safety-Anforderung).

Durch die enge Integration der oben beschriebenen Aufgaben auf einem Chip ergeben sich aus Sicht der Cybersecurity zwei konkrete Angriffspfade, gegen die das eingebettete System geschützt werden muss: Angreifer, die über das Remote-Interface auf die Multimedia-ECU zugreifen können, sind theoretisch in der Lage,

1. die an die Steuerung weitergeleiteten Frames zu manipulieren und das Fahrzeug potentiell in seiner Trajektorienplanung zu beeinflussen (→ Safety-Angriff) sowie
2. noch nicht anonymisierte Kameraframes abzugreifen und somit unberechtigt auf in den Frames enthaltene personenbezogene Daten zuzugreifen (→ Privacy-Angriff).

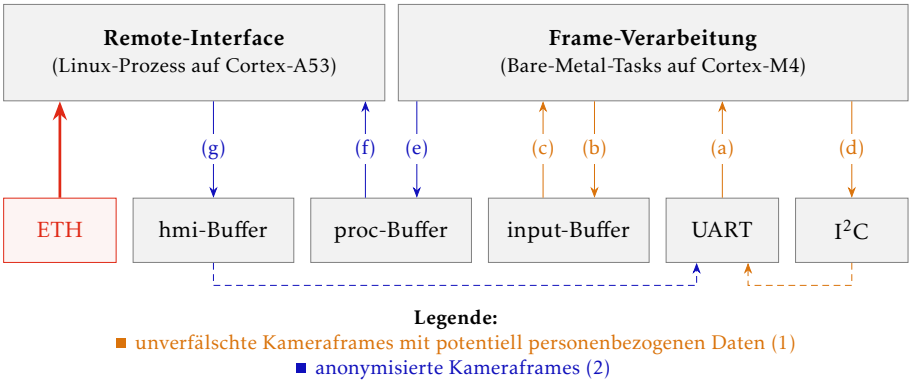

Abbildung 13: On-Chip-Datenflüsse im Demonstrationsszenario (im Nominalfall)

Das übergeordnete Ziel des KIT-Demonstrators ist es dabei, beide potentiellen Angriffspfade durch die Anwendung der Codesign-Methodologie effektiv und per Design zu unterbinden.

2.4.2 Realisierung des Demonstrators

Das oben beschriebene Konzept ist in AP 4.1 umgesetzt worden. Die folgenden Unterabschnitte beschrieben die dafür erfolgten Arbeiten.

Hardwareplattform für die Multimedia-ECU

Die zentrale Multimedia-ECU ist durch ein MCIMX8M-EVK-Evaluationsboard von NXP realisiert worden. Dieses basiert auf dem i.MX 8M von NXP, einem heterogenen System-on-Chip auf Mehrkernprozessorbasis (MPSoC) mit APU-Unterstützung. Als primäre Recheneinheiten verfügt dieser Chip über einen Cortex-A53- und einen Cortex-M4-Prozessor.

Softwarerealisierung, nominale Informationsflüsse und Angriffspfade

Ein auf dem Cortex-A53-Prozessor ausgeführtes Linux ist genutzt worden, um die Kommunikation mit dem Remote-Interface umzusetzen.

Um die Komplexität zu begrenzen und den Fokus auf die On-Chip-Isolation zu legen, wurden alle Aufgaben zur Handhabung von Kameraframes durch Bare-Metal-Anwendungen auf dem Cortex-M4-Prozessor absolviert. Hierbei handelt es sich um eine Vereinfachung, da die Performanz dieses Prozessors zur Anonymisierung realer Kameraframes nicht ausreicht. Um dennoch eine realistische Demonstration zu ermöglichen, wurden im gesamten Setup statt realer Kameraframes entsprechende IDs voraufgezeichneter Frames ausgetauscht. Die Aufgabe, einen gegebenen Frame zu anonymisieren, hat sich damit auf das Ersetzen der ursprünglichen ID mit der ID des anonymisierten Gegenstücks beschränkt. In der Praxis ist es insbesondere denkbar, zur Anonymisierung die auf dem Chip vorhandene GPU einzusetzen.

Abbildung 13 zeigt den Informationsfluss innerhalb der Multimedia-ECU. Von der Kamera erzeugte Frame-IDs werden zunächst über UART empfangen (a) und von einer Anwendung auf dem Cortex-M4 in den „input-Buffer“ geschrieben (b). Eine weitere Anwendung auf diesem Prozessor liest die zwischengespeicherten Frames periodisch aus (c) und leitet sie über I^2C an die Steuerung weiter (d). Darüber hinaus werden die ausgelesenen Frames wie oben beschrieben anonymisiert und in den „proc-Buffer“ übertragen (e). Diese anonymisierten Frames liest ein

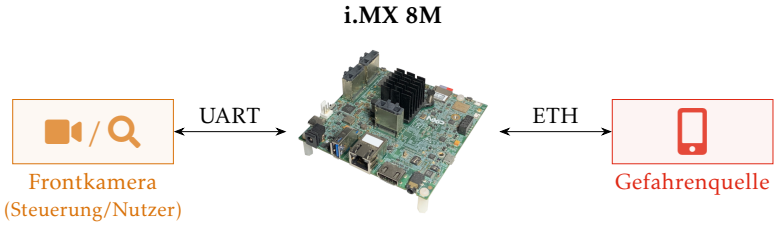

Abbildung 14: Evaluationsaufbau auf Basis eines i.MX-8M-Boards

Linux-Prozess (f) aus und erzeugt daraus für das Remote-Interface und den Touchscreen aufbereitete HMI-Inhalte (g). Angreifer, die über das Linux-OS vollen Zugriff auf die Ports des Cortex-A53 erlangen, können den input-Buffer direkt auslesen (→ Privacy-Angriff) oder den input-Buffer überschrieben (→ Safety-Angriff).

Um die jeweils aktiven Frames zu demonstrativen Zwecken leicht abgreifen zu können, ist eine Kopie der über I^2C versandten bzw. in den „hmi-Buffer“ geschriebenen Daten permanent per UART zurückgeliefert worden (siehe die gestrichelten Linien in Abbildung 13).

Erzeugung von Isolationskonfigurationen durch die Toolchain

Nach der grundsätzlichen Implementierung des Nominalverhaltens der Multimedia-ECU ist eine DSL-Formulierung der für die Codesign-Methodologie relevanten Modellinstanz aufgestellt worden. Die vollständige Formulierung kann Abschnitt A.2 im Anhang entnommen werden und wurde im Anschluss verwendet, um die Toolchain aus AP 3 auszuführen. Die von der Toolchain generierte APU-Konfiguration stand daraufhin zum Einsatz auf dem i.MX 8M zur Verfügung.

Demonstration per Software-Applikation auf dem Host-PC

Um den Effekt dieser APU-Konfiguration im demonstrierten Szenario zu veranschaulichen, ist eine Software-Applikation für den Host-PC entwickelt worden, die sich per UART mit dem Cortex-M4 sowie per SSH (über ETH) mit dem Linux-OS auf dem Cortex-A53 verbindet.

Wie in Abbildung 14 dargestellt, simuliert diese Anwendung die Frontkamera, indem sie IDs von Kameraframes generiert und über UART an die Multimedia-ECU liefert. Darüber hinaus ist sie mit einer Funktion ausgestattet, die jeweils letzte per I^2C versandte bzw. aktuelle im hmi-Buffer befindliche ID periodisch über UART auszulesen. Zusätzlich baut die Anwendung über Ethernet eine Verbindung zum Linux-OS auf, um auf Anforderung entweder einen Safety- oder einen Privacy-Angriff anzustoßen.

Abbildung 15 zeigt einen leicht modifizierten Screenshot der Applikation für den Host-PC, wenn bei ungeschützter Hardware (d. h. ohne Verwendung der durch die Toolchain generierten APU-Konfiguration) kein Angriff ausgeführt wird. Die tatsächlich in der Applikation sichtbaren Frames enthalten Daten, die (gemäß des grundsätzlichen Demonstratorkonzepts) potentiell einzelnen Personen zugeordnet werden könnten. Für die Zwecke dieses öffentlich zugänglichen Berichts wurden die drei in Realität sichtbaren Kameraframes daher unkenntlich gemacht und zur weiteren Bezugnahme mit Ziffern versehen. Hierbei bezeichnet (1) den ursprünglich von der Kamera gelieferten Frame, während (2) sich auf den durch Anonymisierung aus (1) hervorgegangenen Frame bezieht. Abbildung 13 verdeutlicht den nominalen Fluss dieser Frames unter Verwendung dieser Ziffern.

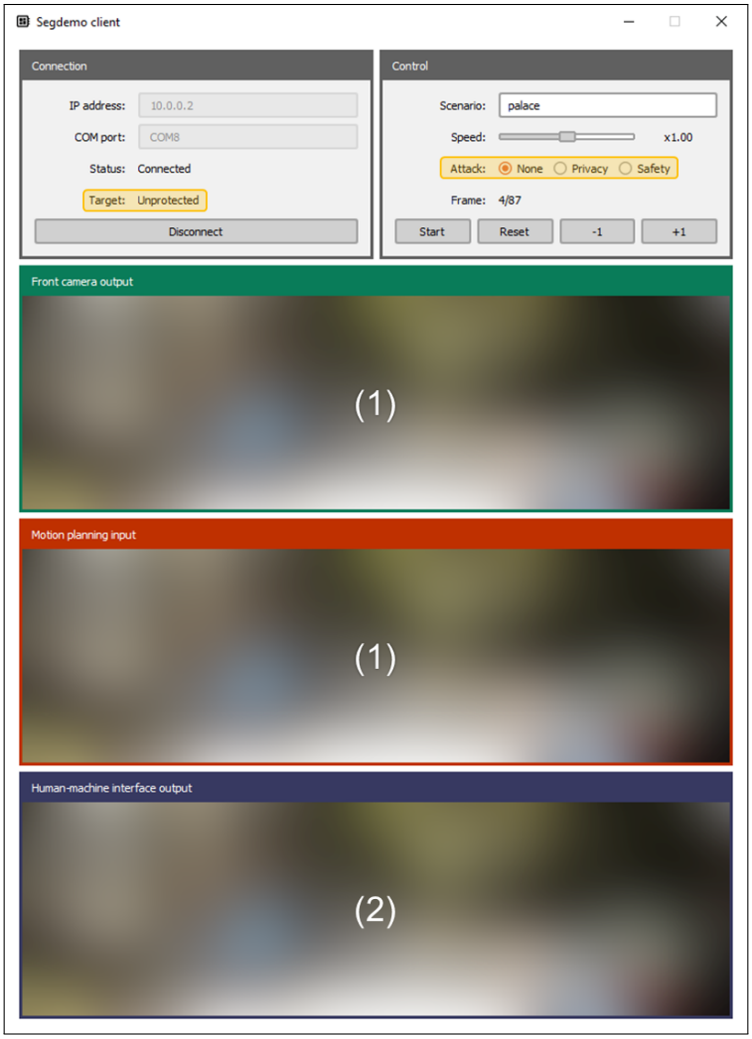

Abbildung 15: Visualisierungssoftware ohne Angriff bei ungeschützter Hardware

Grundsätzlich zeigt die Applikation immer Frames folgender drei Quellen an:

- In der oberen Box (grün) wird immer derjenige Frame angezeigt, den die Applikation zum entsprechenden Zeitpunkt als Ausgabe der Frontkamera simuliert. Bezogen auf den Fluss in Abbildung 13 handelt es sich dabei um den Frame, der als (a) über UART empfangen und vom Cortex-M4 verarbeitet wird.
- Die mittlere Box (rot) zeigt den letzten per I^2C versandten Frame an. Hierbei handelt es sich um (d) in Abbildung 13, d. h. um den an die Steuerung versandten Frame.
- Die untere Box (blau) wird zum Inhalt des hmi-Buffers (g) synchron gehalten. Es es sich um den Frame, den ein Nutzer über das Remote-Interface abrufen kann.

Wie in Abbildung 15 zu sehen ist, entspricht das visualisierte Systemverhalten exakt der Erwartung: Der an die Steuerung weitergeleitete Frame (rot) ist identisch zum durch die Kamera selbst simulierten Frame (grün). Gleichzeitig ist über das Remote-Interface (blau) nur der anonymisierte Frame sichtbar.

Führt man bei weiterhin ungeschützter Hardware einen Safety-Angriff durch, wie es Abbildung 16 zeigt, ist eine Änderung des an die Steuerung weitergeleiteten Frames (rot) erkennbar.

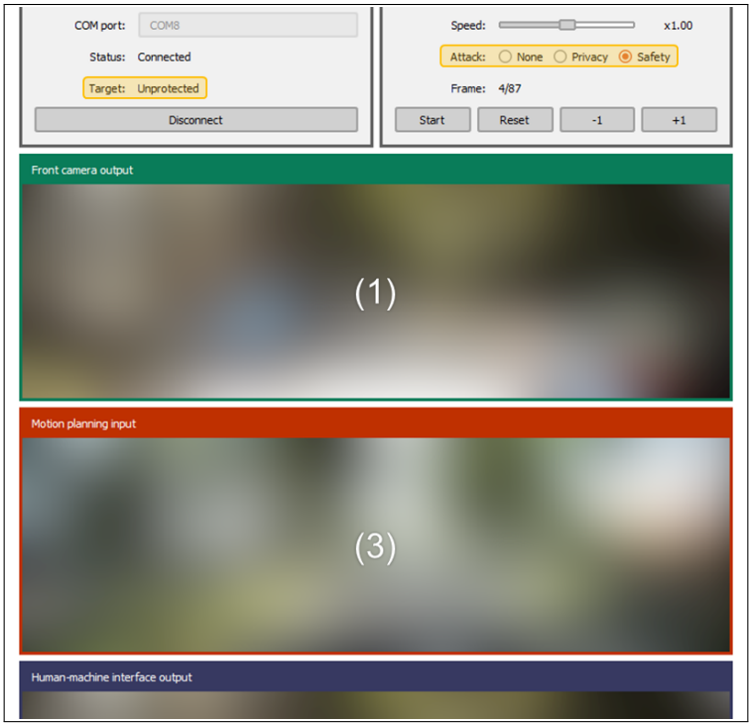

Abbildung 16: Safety-Angriff bei ungeschützter Hardware

Dieser wird nun mit (3) bezeichnet und entspricht einer durch den Angreifer vorgetäuschten Fahrsituation. Durch diese Demonstration wird deutlich, dass es ein fehlender Zugriffsschutz der Linux-Applikation ermöglicht, den input-Buffer und damit sowohl (c) als auch (d) in Abbildung 13 zu manipulieren. Analog dazu veranschaulicht die Applikation einen Privacy-Angriff dadurch, dass nicht anonymisierte Frames (1) per Remote-Interface abgreifbar sind, d. h. in der blauen Box angezeigt werden können. Auf die explizite Darstellung dieses Falls wird im Rahmen dieses Berichts aus Platzgründen verzichtet.

Wird die generierte APU-Konfiguration verwendet, zeigt dies die Applikation wie in Abbildung 17 hervorgehoben an. Aus dem Screenshot geht hervor, dass die APU-Konfiguration in der Lage ist, den Safety-Angriff effektiv zu verhindern: Der erneute Versuch, die Frames des input-Buffers zu überschreiben, bleibt diesmal erfolglos. Die mittlere Box (rot) enthält den zum entsprechenden Zeitpunkt von der Kamera gelieferten Frame (1), was wieder dem erwarteten Nominalverhalten entspricht. Auch der Privacy-Angriff schlägt fehl: Der originale Frame (1) kann der Linux-Applikation nun nicht mehr in den hmi-Buffer übertragen werden.

2.4.3 Evaluationsergebnisse

In AP 4.2 wurde abschließend eine Evaluation der beiden entwickelten Methodologien vorgenommen. Unter anderem ist hierfür der unter Verwendung der Codesign-Methodologie entwickelte Demonstrator aus AP 4.1 herangezogen worden. Um darüber hinaus auch eine praxisnahe Bewertung der Security-Methodologie vornehmen zu können, ist eine entsprechend modifizierte Version des Demonstrators entwickelt worden. In dieser ist zur automatischen Generierung der APU-Konfigurationen ein Vertraulichkeitsframework und ein Integritätsframework zum Einsatz gekommen. Bei der Aufstellung einer entsprechenden Modellinstanz für die Security-Methodologie sind keine Probleme aufgetreten. Somit konnte gezeigt werden, dass

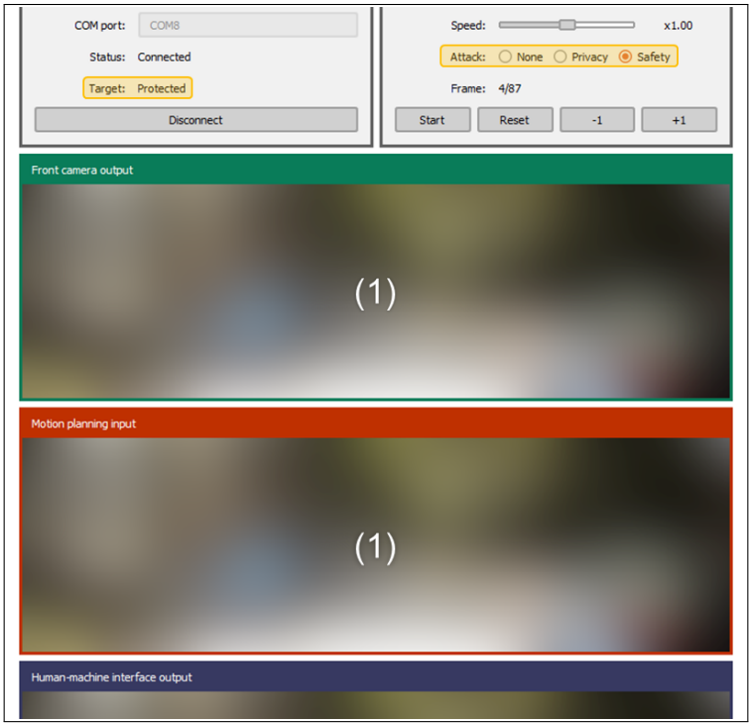

Abbildung 17: Safety-Angriff bei geschützter Hardware

auch diese Methodologie auf praktische Problemstellungen anwendbar ist. Insgesamt lag der Fokus der Evaluation auf folgenden Aspekten:

1. **Automatisierungsgrad**: Um einen konkreten Mehrwert während der Entwicklung zu liefern, müssen die Methodologien einen hohen Automatisierungsgrad bieten.
2. **Flexibilität**: Trotz des hohen Automatisierungsgrad muss möglich sein, praxisrelevante Systemdesigns so abzubilden, dass während der Entwurfsphase keine inakzeptable Einschränkung entsteht. Dies umfasst insbesondere die Möglichkeit, manuelle Implementierungsentscheidungen weitgehend uneingeschränkt treffen zu können, sofern diese aus bestimmten Gründen (wie Safety, Security, . . .) notwendig sind.

Um den Automatisierungsgrad zu evaluieren, ist vor allem die Notwendigkeit für manuelle Eingriffe an verschiedenen Stellen im anvisierten Entwicklungsprozess analysiert worden. Hierbei konnte bei keiner der beiden entwickelten Methodologien ein Bedarf für manuelle Eingriffe identifiziert werden, der aus Sicht der Zuwendungsempfängers als unangemessen oder unwirtschaftlich einzuschätzen ist.

Bezüglich der Flexibilität wurde insbesondere evaluiert, inwiefern praktisch relevante Szenarien höherer Komplexität durch die Methodologien behandelt werden können. In diesem Zusammenhang wurde insbesondere durch die erfolgreiche Integration des Linux-OS gezeigt, dass die in AP 3 entwickelten Methodologien keine prinzipiell inakzeptablen Einschränkungen hervorrufen.

3 Verwertung und Einordnung

3.1 Bezug zum zahlenmäßigen Nachweis

Die mit Abstand umfangreichste Position des zahlenmäßigen Nachweises bilden Personalkosten für wissenschaftliche Mitarbeiter, die am Institut tätig waren und über die Projektlaufzeit hinweg zur Bearbeitung des Teilvorhabens zum Einsatz gekommen sind. Vor allem zu Projektbeginn wurden die Mitarbeiter durch studentische Hilfskräfte bei Recherchearbeiten und der Inbetriebnahme der genutzten Hardwareplattform (insb. in Vorbereitung auf AP 4) unterstützt.

Der ursprüngliche Projektplan hat vorgesehen, große Teile des in AP 4 entwickelten Demonstrators im Labor und mit Unterstützung studentischer Hilfskräfte zu realisieren. Aufgrund verschiedener Einschränkungen, die sich im Zusammenhang mit der COVID-19-Pandemie ergeben haben, war dies allerdings nicht wie angestrebt möglich. Die diesbezüglich geplanten Arbeiten sind daher an die bestehenden Einschränkungen angepasst und von wissenschaftlichen Mitarbeitern vorgenommen worden, was entsprechende Umplanungen nötig gemacht hat.

Darüber hinaus sind Kosten für die Beschaffung von Hardwareplattformen (insb. Evaluationsboards auf Basis des i.MX 8M) und elektrischen Verbrauchsmaterialien angefallen. Sie haben der Entwicklung von Backends für das in Abschnitt 2.3.1 beschriebene Tool und der darauf aufbauenden Visualisierung in AP 4 gedient. Abschließend sind Kosten für Reisen (insb. in der ersten Hälfte der Projektlaufzeit zur Abstimmung mit den Projektpartnern) und zur Beschaffung technischer Standards (siehe Abschnitt 1.5) angefallen.

3.2 Notwendigkeit und Angemessenheit der Arbeiten

Wie in Abschnitt 1.4 beschrieben, ist vor Projektbeginn eine konkrete Problematik im Zusammenhang mit dem Entwurf zukünftiger E/E-Architekturen identifiziert worden, die der Stand der Technik nicht bzw. nur unzureichend adressiert.

Durch die im Teilvorhaben erbrachten Arbeiten ist es gelungen, diese Lücke im Stand der Technik zu schließen und durch eine praxisnahe Validierung nachzuweisen, dass die entwickelten Ansätze zu einem konkreten Mehrwert beim Entwurf von E/E-Architekturen für autonome Fahrzeuge führen. Damit konnte ein wichtiger Beitrag zur Absicherung von IKT-Systemen in zukünftigen Mobilitätslösungen geleistet werden.

Alle in den Arbeitspaketen durchgeführten Arbeiten haben zu diesem Ziel beigetragen und waren daher notwendig. Konkret mussten die in AP 1 geleisteten Arbeiten durchgeführt werden, um den Betrachtungsrahmen des Projekts partnerübergreifend zu definieren und in späteren Arbeitspaketen auf praxisrelevante Beispielszenarien zurückgreifen zu können. Der geleistete Bei-

trag zu AP 2 war notwendig, um anhand eines solchen Beispielszenarios eine initiale Planung der angestrebten Lösung vorzunehmen und sie in den Kontext des Gesamtvorhabens zu setzten. Wie später in Abschnitt 3.4 ausgeführt, hat sich der dabei gewählte Schwerpunkt rückblickend als sehr relevant erwiesen. AP 3 bildet den Kern des Teilvorhabens und wurde genutzt, um die identifizierte Lücke im Stand der Technik zu schließen. Als ebenfalls von zentraler Bedeutung ist AP 4 anzusehen, weil dort neben der Entwicklung eines anschaulichen Demonstrators konkretes Feedback aus der Validierung zurück an AP 3 geflossen ist. So konnte während des Projekts eine iterative Optimierung der Zwischenergebnisse vorgenommen werden, was beispielsweise zur Definition der Security-Methodologie geführt hat.

3.3 Nutzen und Verwertbarkeit der Ergebnisse

Ziel des KIT war es, die erlangten Erkenntnisse in wissenschaftlichen Publikationen, in weiteren Projekten und im Rahmen der Lehre zu verwerten.

Insbesondere die Verwertung durch wissenschaftliche Publikationen ist aus Sicht des Zuwendungsempfängers bereits umfassend erfolgt. So sind die Ergebnisse des Teilvorhabens in Form von zwei Konferenzbeiträgen und einem Journalartikel der Öffentlichkeit zugänglich gemacht worden (siehe Abschnitt 3.5). Auf diese Weise konnte ein relevanter Beitrag zum Stand der Technik geleistet werden. Obwohl derzeit keine direkt auf das Teilvorhaben zurückgehenden Veröffentlichungen mehr geplant sind, wird der Zuwendungsempfänger in Zukunft auf den bisherigen Ergebnissen aufbauen und abgeleitete Erkenntnisse nach Möglichkeit in entsprechende Publikationen überführen.

Bezüglich der Verwertung in der Lehre war initial angestrebt, die im Teilvorhaben erlangten Erkenntnisse über Abschlussarbeiten und Lehrveranstaltungen an die Studierendenschaft des KIT zurückzuspiegeln. Während der Projektlaufzeit ist es in diesem Kontext gelungen, zwei direkt auf den entwickelten Methodologien aufbauende Masterarbeiten zu organisieren. Auch in Aufgabenstellungen für weitere Abschlussarbeiten, die mit DEFEnD nicht unmittelbar in Zusammenhang standen, sind die aus dem Teilvorhaben erlangten Erkenntnisse bereits eingeflossen. Darüber hinaus konnten Problemstellungen, die im Teilvorhaben identifiziert und betrachtet worden sind, in ausgewählten Lehrveranstaltungen als Praxisbeispiele herangezogen werden. Mit diesen Verwertungsaktivitäten wird in Zukunft fortgefahren.

Das im Umfeld des autonomen Fahrens erlangte Know-How konnte darüber hinaus genutzt werden, um weitere in diesem Kontext angesiedelte Aktivitäten zu initiieren. So ist es insbesondere gelungen, ein auf dem hier beschriebenen Teilvorhaben aufbauendes Forschungsvorhaben für das bewilligte UNCOVER-Projekt (16KIS1414) zu definieren.

Eine direkte wirtschaftliche Verwertung der Erkenntnisse ist bisher nicht erfolgt, prinzipiell aber möglich. Die beiden im Projekt entwickelten Methodologien automatisieren Aufgaben, die beim Einsatz moderner Plattformen im sicherheitskritischen Umfeld unvermeidbar sein werden. So entsteht ein konkreter wirtschaftlicher Mehrwert. Während der Projektlaufzeit konnte die grundsätzliche Praktikabilität des Ansatzes erfolgreich gezeigt werden. Es ist daher denkbar, ein entsprechendes Werkzeug in Form einer Firmenausgründung auf dem Markt zu platzieren.

3.4 Bekannt gewordener Fortschritt anderer Stellen

Über den Projektkontext hinaus ist die Safety autonomer Fahrzeuge aus rein anwendungsorientierter Perspektive, d. h. unabhängig von der funktionalen Sicherheit der E/E-Architektur, intensiv betrachtet worden. Beispielsweise enthält [15] eine physikalische Betrachtung sicherer Bremsvorgänge im Kontext des autonomen Fahrens. Überlegungen, die dieser Abstraktionsebene zuzuordnen sind, wurden im Projektrahmen durch Konsortialpartner erbracht. Die durch das Teilvorhaben erarbeiteten Methodologien schließen sich im Projekt auf niedrigerer Abstraktionsebene an. Daher stehen sie mit Arbeiten wie der oben genannten in keinem Widerspruch, sondern ergänzen.

Eine große Zahl an Arbeiten ist über die Projektlaufzeit hinweg zur Security vernetzter, automatisierter oder autonomer Fahrzeuge publiziert worden. Die in [17] veröffentlichte Übersicht kategorisiert eine Vielzahl von Publikationen, die im Zusammenhang mit Angriffsvektoren auf On-Board-Netzwerke, auf die externe Kommunikation von Fahrzeugen oder einzelne Fahrzeugkomponenten (wie Sensoren) relevant sind. Die beiden in Abschnitt 2.3 beschriebenen Methodologien beziehen sich insbesondere auf On-Board-Netzwerke und fügen sich in die Betrachtungen daher nahtlos ein. Insbesondere unterstreichen die Ausführungen in [17], dass drahtlos mit externen Einheiten kommunizierende Infotainment-Systeme in vielen bisher erfolgreich durchgeführten Angriffen auf das On-Board-Netzwerk eine entscheidende Rolle gespielt haben. Darüber hinaus verdeutlicht die Zusammenstellung, dass die Safety-Relevanz potentieller Angriffe in der Praxis von entscheidender Bedeutung ist. Diese beiden Aspekte bestätigen die Relevanz des in Abschnitt 2.4.1 entwickelten Demonstratoszenarios.

Weiterhin ist eine Vielzahl von Arbeiten zum Entwurf abgesicherter On-Board-Netzwerke erfolgt. So wird in [22] beispielsweise ein neuartiges Vorgehen zur Risikoanalyse in diesem Kontext vorgestellt. Allerdings handelt es sich dabei um ein Verfahren, das die Notwendigkeit geeigneter Mechanismen zur Erfüllung nichtfunktionaler Anforderungen systematisch herleitet, während die im Teilvorhaben erarbeitete Methodologie zur automatisierten Realisierung eines solchen Mechanismus dient. Als weiteres Beispiel sei an dieser Stelle das Intrusion Detection System (IDS) aus [16] genannt. Durch ein trainiertes neuronales Netzes werden Angriffe auf den CAN-Bus zur Laufzeit detektiert und entsprechend behandelt. Verglichen mit der im Teilvorhaben entwickelten Security-Methodologie handelt es sich hierbei um eine Arbeit auf gleicher Abstraktionsebene, allerdings mit einem starken Fokus auf die Laufzeit des Systems. Die im Rahmen dieses Teilvorhabens entwickelte Security-Methodologie kann diesbezüglich als komplementär zum Stand der Technik gesehen werden. Sie ergänzt die aus der Literatur bekannten Ansätze um eine automatisierte Methode zur Umsetzung bestimmter Security-Anforderungen unterstützen. Dabei werden On-Chip-Aspekte von Grund auf als Teil des On-Board-Netzwerks aufgefasst, was in Arbeiten dritter Stellen oft nicht oder nur indirekt der Fall ist.

Abschließend sind an dieser Stelle noch Publikationen mit einem Fokus auf Safety (im Sinne der funktionalen Sicherheit von E/E-Architekturen) zu nennen. So wird in [8] beispielsweise ein Mechanismus zum Laufzeittest von Speicher im Kontext sicherheitskritischer Multicore-Systeme vorgestellt. Obwohl diese konkrete Arbeit vom Teilvorhaben weitgehend unabhängig ist, beschäftigt sie sich mit einem vergleichbaren Problemumfeld, nämlich der zunehmenden Relevanz von Mehrkernprozessoren in sicherheitskritischen eingebetteten Systemen. Wie in Abschnitt 1.4 beschrieben, war die Beobachtung dieses Trends ausschlaggebend für den konkret im Teilvor-

haben verfolgten Arbeitsplan. Eine weitere Arbeit mit einem stärkeren Fokus auf Safety, die auf On-Board-Netzwerke ausgerichtet ist und auch Security-Anforderungen bei der Kommunikation berücksichtigt, wurde in [19] vorgestellt. Sie beschreibt einen modellbasierten Ansatz, der die Kommunikation zwischen Steuergeräten per Design echtzeitfähig auslegt und kritische Nachrichten gleichzeitig per Nachrichtenauthentifizierungscode absichert. Damit handelt es sich um eine Methodik, die aus der Perspektive dieses Berichts als Safety/Security-Codesign-Ansatz im engeren Sinne aufzufassen ist. Sowohl diese Tatsache als auch die betrachtete Abstraktionsebene ist als Gemeinsamkeit mit Codesign-Methodologie aus Abschnitt 2.3.1 zu nennen. Die konkrete Ausrichtung auf Off-Chip-Kommunikation über CAN stellt dahingegen einen entscheidenden Unterschied zur Codesign-Methodologie dar.

Zusammenfassend lässt auf Basis der beispielhaft genannten Publikationen seit Projektstart festhalten, dass die von dritter Seite durchgeführten Arbeiten zwar häufig das gleiche übergeordnete Ziel wie das Teilvorhaben verfolgen, den hier geleisteten Arbeiten aber nicht widersprechen oder ihre Relevanz mindern. Vielmehr können die entwickelten Methodologien als komplementär zum fortschreitenden Stand der Technik gesehen werden.

3.5 Erfolgte oder geplante Veröffentlichungen

Die erarbeiteten Ergebnisse sind in Form der Konferenzbeiträge [1, 3] und dem Journalartikel [2] veröffentlicht worden. Details zum jeweiligen Fokus dieser Publikationen sind insbesondere Abschnitt 2.3 zu entnehmen. Derzeit ist keine weitere Veröffentlichung der unmittelbar aus dem Teilvorhaben hervorgegangenen Ergebnisse geplant.

Literaturverzeichnis

Publikationen des Teilvorhabens

[1] Tobias Dörr, Timo Sandmann und Jürgen Becker. „A Formal Model for the Automatic Configuration of Access Protection Units in MPSoC-Based Embedded Systems“. In: *2020 23rd Euromicro Conference on Digital System Design (DSD)*. 2020. DOI: 10.1109/DSD51259.2020.00098.

[2] Tobias Dörr, Timo Sandmann und Jürgen Becker. „Model-based configuration of access protection units for multicore processors in embedded systems“. In: *Microprocessors and Microsystems* (2021). Im Erscheinen. DOI: 10.1016/j.micpro.2021.104377.

[3] Tobias Dörr, Timo Sandmann, Hannes Mohr und Jürgen Becker. „Employing the Concept of Multilevel Security to Generate Access Protection Configurations for Automotive On-Board Networks“. In: *2021 24th Euromicro Conference on Digital System Design (DSD)*. 2021. DOI: 10.1109/DSD53832.2021.00026.

Literatur mit Projektbezug

[4] D. E. Bell und L. J. LaPadula. *Secure Computer System: Unified Exposition and MULTICS Interpretation*. Forschungsbericht. MITRE Corporation, 1976.

[5] K. J. Biba. *Integrity Considerations for Secure Computer Systems*. Forschungsbericht. MITRE Corporation, 1975.

[6] Vamsi Boppana, Sagheer Ahmad, Ilya Ganusov u. a. „UltraScale+ MPSoC and FPGA families“. In: *2015 IEEE Hot Chips 27 Symposium (HCS)*. Cupertino, California, 2015. DOI: 10.1109/HOTCHIPS.2015.7477457.

[7] Ondrej Burkacky, Johannes Deichmann und Jan Paul Stein. *Automotive software and electronics 2030*. Bericht. McKinsey & Company, Juli 2019.

[8] Ciro Donnarumma, Alessandro Biondi, Francesco De Rosa und Stefano Di Carlo. „Integrating Online Safety-related Memory Tests in Multicore Real-Time Systems“. In: *2020 IEEE Real-Time Systems Symposium (RTSS)*. Houston, TX, USA, Dez. 2020. DOI: 10.1109/RTSS49844.2020.00035.

[9] Mahmoud Hashem Eiza und Qiang Ni. „Driving with Sharks: Rethinking Connected Vehicles with Vehicle Cybersecurity“. In: *IEEE Vehicular Technology Magazine* 12.2 (2017), S. 45–51. DOI: 10.1109/MVT.2017.2669348.

[10] Rolf Ernst. „Automated Driving: The Cyber-Physical Perspective“. In: *Computer* 51.9 (2018), S. 76–79. DOI: 10.1109/MC.2018.3620974.

[11] Marie-Aude Esteve, Joost-Pieter Katoen, Viet Yen Nguyen u. a. „Formal correctness, safety, dependability, and performance analysis of a satellite“. In: *2012 34th International Conference on Software Engineering (ICSE)*. Zurich, 2012, S. 1022–1031. DOI: 10.1109/ICSE.2012.6227118.

[12] Moritz Eysholdt und Heiko Behrens. „Xtext: Implement Your Language Faster than the Quick and Dirty Way“. In: *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion (SPLASH '10)*. Reno/Tahoe, NV, USA, 2010. DOI: 10.1145/1869542.1869625.

[13] Mohamed Hassan. „Heterogeneous MPSoCs for Mixed-Criticality Systems: Challenges and Opportunities“. In: *IEEE Design & Test* 35.4 (2018). DOI: 10.1109/MDAT.2017.2771447.

[14] International Organization for Standardization. *ISO 26262-9: Road vehicles — Functional safety — Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses*. Standard. 2018.

[15] Philip Koopman, Beth Osyk und Jack Weast. „Autonomous Vehicles Meet the Physical World: RSS, Variability, Uncertainty, and Proving Safety“. In: *Computer Safety, Reliability, and Security (SAFECOMP 2019)*. Cham: Springer International Publishing, 2019. DOI: 10.1007/978-3-030-26601-1_17.

[16] Vipin Kumar Kukkala, Sooryaa Vignesh Thiruloga und Sudeep Pasricha. „INDRA: Intrusion Detection Using Recurrent Autoencoders in Automotive Embedded Systems“. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.11 (Nov. 2020), S. 3698–3710. DOI: 10.1109/TCAD.2020.3012749.

[17] Anthony Lopez, Arnav Vaibhav Malawade, Mohammad Abdullah Al Faruque u. a. „Security of Emergent Automotive Systems: A Tutorial Introduction and Perspectives on Practice“. In: *IEEE Design & Test* 36.6 (Dez. 2019), S. 10–38. DOI: 10.1109/MDAT.2019.2944086.

[18] Charlie Miller. „Lessons Learned from Hacking a Car“. In: *IEEE Design & Test* 36.6 (2019). DOI: 10.1109/MDAT.2018.2863106.

[19] Hyeran Mun, Kyusuk Han und Dong Hoon Lee. „Ensuring Safety and Security in CAN-Based Automotive Embedded Systems: A Combination of Design Optimization and Secure Communication“. In: *IEEE Transactions on Vehicular Technology* 69.7 (Juli 2020), S. 7078–7091. DOI: 10.1109/TVT.2020.2989808.

[20] Tohru Nojiri, Yuki Kondo, Naohiko Irie u. a. „Domain Partitioning Technology for Embedded Multicore Processors“. In: *IEEE Micro* 29.6 (2009). DOI: 10.1109/MM.2009.96.

[21] Jason Oberg, Wei Hu, Ali Irturk u. a. „Information flow isolation in I2C and USB“. In: *Proceedings of the 48th Design Automation Conference*. San Diego, CA, USA, 2011. ISBN: 978-1-4503-0636-2. DOI: 10.1145/2024724.2024782.

[22] Dominik Püllen, Nikolaos Anagnostopoulos, Tolga Arul und Stefan Katzenbeisser. „Safety Meets Security: Using IEC 62443 for a Highly Automated Road Vehicle“. In: *Computer Safety, Reliability, and Security (SAFECOMP 2020)*. Bd. 12234. Cham: Springer International Publishing, 2020. DOI: 10.1007/978-3-030-54549-9_22.

[23] SAE International. *J3061: Cybersecurity Guidebook for Cyber-Physical Vehicle Systems*. Standard. 2016.

[24] Selma Saidi, Sebastian Steinhorst, Arne Hamann u. a. „Special Session: Future Automotive Systems Design: Research Challenges and Opportunities“. In: *2018 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. Turin, 2018. DOI: 10.1109/CODESISSS.2018.8525873.

[25] Benjamin Tan, Morteza Biglari-Abhari und Zoran Salcic. „Towards decentralized system-level security for MPSoC-based embedded applications“. In: *Journal of Systems Architecture* 80 (2017). DOI: 10.1016/j.sysarc.2017.09.001.

[26] Julius Ziegler, Philipp Bender, Markus Schreiber u. a. „Making Bertha Drive—An Autonomous Journey on a Historic Route“. In: *IEEE Intelligent Transportation Systems Magazine* 6.2 (2014), S. 8–20. DOI: 10.1109/MITS.2014.2306552.

Anhang

A.1 Bezug zur bisher publizierten Terminologie

In den Veröffentlichungen [1–3] kommt für Elemente des entwickelten Metamodells eine englische Terminologie zum Einsatz. Die folgenden Tabellen stellen diese Veröffentlichungen und den vorliegenden Bericht in Bezug, indem relevante Originalbegriffe ihren hier eingesetzten Übersetzungen gegenübergestellt werden.

	Originalbegriff	**Übersetzung**
	feature	Feature
	dependability	Zuverlässigkeit
I	*platform architecture*	Plattformarchitektur
	execution unit, siehe Definition in [1]	Ausführungseinheit
	unit, siehe Definition in [2]	Einheit
	communication link	Verbindung
	container	Container
	root container	Wurzelcontainer
II	*functional implementation*	Funktionsrealisierung
	terminal feature	Terminal-Feature
	desired information flow policy	Informationsflussvorgaben
	accepted information flow	akzeptierter Informationsfluss
	required information flow	erforderlicher Informationsfluss
III	*information flow implementation*	Informationsflussrealisierung
	forwarding feature	Forwarding-Feature
	transaction	Transaktion
	internal flow	interner Fluss

Tabelle 1: Übersetzung der Terminologie zur Codesign-Methodologie

Tabelle 1 bezieht sich auf die Codesign-Methodologie nach [1] und [2]. Sofern möglich, wird ein Begriffspaar hierbei seiner jeweiligen Ebene des Metamodells (I, II oder III) zugeordnet. Analog dazu bezieht sich Tabelle 2 auf die zusätzlich für die Security-Methodologie nach [3] relevanten Begriffe. Alle in [3] vorgenommenen Erweiterungen finden auf der Ebene der Funktionsrealisierung (II) statt.

	Originalbegriff	Übersetzung
II	*security level*	Security-Level
	confidentiality	Vertraulichkeit
	confidentiality requirement	Vertraulichkeitsanforderung
	confidentiality level	Vertraulichkeitslevel
	confidentiality framework	Vertraulichkeitsframework
	integrity	Integrität
	integrity requirement	Integritätsanforderung
	integrity level	Integritätslevel
	integrity framework	Integritätsframework
	threat agent	Gefahrenquelle
	internal asset	internes Asset
	external asset	externes Asset

Tabelle 2: Übersetzung der Terminologie zur Security-Methodologie

A.2 DSL-Beschreibung der Modellinstanz aus AP 4

In diesem Anhang folgt die DSL-Formulierung, die im Rahmen von AP 4 zur Demonstration der Codesign-Methodologie verwendet wurde. Hierbei beschreibt der Code in Listing 1 die Plattformarchitektur, der Code in Listing 2 die Funktionsrealisierung und der Code in Listing 3 die Informationsflussrealisierung der entsprechenden Modellinstanz.

```
platform arch arch01 {
   container mpsoc {
      generator 'imx8m'

      unit a53 (gen:comp='a53' gen:domain='0')
      unit m4 (dependable! gen:comp='m4' gen:domain='1')

      unit i2c2 (gen:comp='i2c2')
      unit uart1 (gen:comp='uart1')
      unit uart2 (gen:comp='uart2')
      unit enet (gen:comp='enet1')

      unit sync_mem (gen:comp='ocram' gen:addr='0x0-0x80')
      unit input_mem (gen:comp='ocram' gen:addr='0x1000-0x1080')
      unit proc_mem (gen:comp='ocram' gen:addr='0x1080-0x1100')
      unit hmi_mem (gen:comp='ocram' gen:addr='0x1100-0x1180')
      unit ddr (gen:comp='dram' gen:addr='0x0-0xc000_0000')
      unit gic (gen:comp='gic' gen:addr='0x0-0x10_0000')

      link main (gen:comp='main') {
         a53, m4, ddr, uart1, uart2, i2c2, enet, gic, sync_mem,
         input_mem, proc_mem, hmi_mem,
      }
   }

   unit hmi_port
   unit camera (dependable!)
   unit motor_controller (dependable!)
   link global_uart { mpsoc.uart2, camera }
```

```
    link global_ethernet { mpsoc.enet, hmi_port }
    link global_i2c { mpsoc.i2c2, motor_controller }
}
```

Listing 1: Plattformarchitektur (I) der Modellinstanz

```
functional impl func01 on arch01 {
    feature hmi_task on mpsoc.a53
    feature interrupt_controller on mpsoc.gic
    feature raw_frame_dispatcher on mpsoc.m4 (dependable!)
    feature frame_processor on mpsoc.m4 (dependable!)
    feature visualization_port on mpsoc.m4 (dependable!)
    feature motor_controller on motor_controller
    feature hmi_port on hmi_port
    feature camera on camera

    accept all between {
        interrupt_controller, frame_processor, hmi_task,
        hmi_port, visualization_port
    }

    accept all between {
        camera, raw_frame_dispatcher, motor_controller, frame_processor
    }

    require camera -> raw_frame_dispatcher
    require raw_frame_dispatcher -> motor_controller, frame_processor
    require frame_processor -> hmi_task
    require hmi_task -> hmi_port, visualization_port
}
```

Listing 2: Funktionsrealisierung (II) der Modellinstanz

```
flow impl flow01 on func01 {
    protect mpsoc.main

    feature uart1 on mpsoc.uart1
    feature uart2 on mpsoc.uart2
    feature i2c2 on mpsoc.i2c2
    feature enet on mpsoc.enet

    feature a53_memory on mpsoc.ddr
    feature kernel on mpsoc.a53

    feature sync_buffer on mpsoc.sync_mem
    feature input_buffer on mpsoc.input_mem
    feature proc_buffer on mpsoc.proc_mem
    feature hmi_buffer on mpsoc.hmi_mem

    full kernel -> mpsoc.main -> interrupt_controller

    read kernel -> mpsoc.main -> sync_buffer
    write visualization_port -> mpsoc.main -> sync_buffer

    full kernel -> mpsoc.main -> a53_memory
    full hmi_task -> mpsoc.main -> enet

```

```
25      read hmi_port -> global_ethernet -> enet
26
27      full kernel -> mpsoc.main -> uart1
28
29      write camera -> global_uart -> uart2
30      full raw_frame_dispatcher -> mpsoc.main -> uart2
31
32      write frame_processor -> mpsoc.main -> proc_buffer
33      read hmi_task -> mpsoc.main -> proc_buffer
34
35      write hmi_task -> mpsoc.main -> hmi_buffer
36      read visualization_port -> mpsoc.main -> hmi_buffer
37
38      write raw_frame_dispatcher -> mpsoc.main -> input_buffer
39      read frame_processor -> mpsoc.main -> input_buffer
40
41      full raw_frame_dispatcher -> mpsoc.main -> i2c2
42      write i2c2 -> global_i2c -> motor_controller
43  }
```

Listing 3: Informationsflussrealisierung (III) der Modellinstanz

Berichtsblatt

1. ISBN oder ISSN	2. Berichtsart (Schlussbericht oder Veröffentlichung) Schlussbericht

3. Titel
KMU-innovativ – Verbundprojekt: DEvelopment For SEcured Autonomous Driving (DEFEnD); Teilvorhaben: Modellbasierte Entwurfsmethodik für Safety-Security-Co-Design im Kontext des automatisierten Fahrens

4. Autor(en) [Name(n), Vorname(n)]	5. Abschlussdatum des Vorhabens Mai 2021
Becker, Jürgen Dörr, Tobias Sandmann, Timo	6. Veröffentlichungsdatum November 2021
	7. Form der Publikation

8. Durchführende Institution(en) (Name, Adresse)	9. Ber. Nr. Durchführende Institution
Karlsruher Institut für Technologie (KIT) Kaiserstraße 12 76131 Karlsruhe	10. Förderkennzeichen 16KIS0886
	11. Seitenzahl 35

12. Fördernde Institution (Name, Adresse)	13. Literaturangaben 26
Bundesministerium für Bildung und Forschung (BMBF) 53170 Bonn	14. Tabellen 2
	15. Abbildungen 17

16. Zusätzliche Angaben

17. Vorgelegt bei (Titel, Ort, Datum)

18. Kurzfassung
Zukünftige autonome Fahrzeuge sind durch einen hohen Grad an Konnektivität und eine zunehmende Zentralisierung der Elektrik/Elektronik-Architektur geprägt. Mit dieser Entwicklung geht ein steigendes Potential für Cyberangriffe und bedeutende Probleme mit der funktionalen Sicherheit einher. In diesem Kontext war es das Ziel des in diesem Abschlussbericht beschriebenen Teilvorhabens, ein ganzheitliches Konzept zu entwickeln, das „Security-by-Design“ mit konstruktiven Ansätzen zur Erfüllung von Safety-Anforderungen kombiniert. Ausgehend von einer identifizierten Lücke im Stand der Technik wurden dafür zwei konkrete Ausprägungen einer Entwurfsmethodologie entwickelt und durch die Anwendung auf eine beispielhafte Fahrzeugarchitektur für das autonome Fahren validiert.

19. Schlagwörter
autonomes Fahren, modellbasierter Entwurf, Safety, Cybersecurity

20. Verlag	21. Preis

Document Control Sheet

1. ISBN or ISSN	2. type of document (e.g. report, publication) Final report

3. title
KMU-innovativ – Joint project: DEvelopment For SEcured Autonomous Driving (DEFEnD); Subproject: Model-based design methodology for safety/security codesign in the context of autonomous driving

4. author(s) (family name, first name(s))	5. end of project
Becker, Jürgen Dörr, Tobias Sandmann, Timo	May 2021
	6. publication date November 2021
	7. form of publication
8. performing organization(s) (name, address) Karlsruhe Institute of Technology (KIT) Kaiserstraße 12 76131 Karlsruhe	9. originator's report no.
	10. reference no. 16KIS0886
	11. no. of pages 35
12. sponsoring agency (name, address) Bundesministerium für Bildung und Forschung (BMBF) 53170 Bonn	13. no. of references 26
	14. no. of tables 2
	15. no. of figures 17

16. supplementary notes

17. presented at (title, place, date)

18. abstract
Future autonomous vehicles are characterized by a high degree of connectivity and an increasing centralization of the electric/electronic architecture. This trend is associated with a growing potential for cyberattacks and functional safety issues. In this context, the goal of the subproject described in this final report was to develop a holistic approach combining the security-by-design concept with constructive approaches to meet safety requirements. Based on an identified gap in the state of the art, two specific manifestations of a design methodology were therefore developed and validated by applying them to a sample vehicle architecture for autonomous driving.

19. keywords
Autonomous driving, Model-based design, Safety, Cybersecurity

20. publisher	21. price