

# Schlussbericht

zum Vorhaben

Konzeption und RT-kompatible Erweiterung von  
Zentralrechenplattformen und Embedded Compute Netzwerken für  
zukünftige hochautomatisierte Fahrzeuge (EMDRIVE)

FÖRDERKENNZEICHEN 16ME0455



**FZI Forschungszentrum Informatik**

Haid-und-Neu-Str. 10-14

76131 Karlsruhe

Autoren: Victor Pazmino Betancourt, Marius Kreutzer, Maximilian Kirschner

Telefon: +49 721 9654-190

E-Mail: pazmino@fzi.de

## Inhaltsverzeichnis

Schlussbericht.....	1
1 Thema und Zielsetzung des Vorhabens .....	5
1.1    Zielsetzung des Teilvorhabens.....	6
1.2    Planung und Ablauf des Vorhabens.....	6
1.3    Stand der Wissenschaft und Technik zu Beginn des Vorhabens .....	8
1.4    Angaben bekannter Konstruktionen, Verfahren und Schutzrechte, die für die Durchführung des Vorhabens benutzt wurden .....	9
1.5    Zusammenarbeit mit anderen Stellen .....	9
2    Ergebnisse.....	10
2.1    AP1: Anforderungen an künftige Rechnerarchitekturen in Automotive / Industrie .....	10
2.1.1    Beitrag des FZI und abgeleitete Kernanforderungen .....	10
2.2    AP2: Enhanced Embedded Computing Systemarchitektur.....	11
2.3    AP3: Dynamische Betriebsstrategien .....	11
2.3.1    Generische Schnittstelle zur dynamischen Beschleuniger-Anbindung .....	11
2.3.2    Migration von Anwendungen in heterogenen Systemen mittels WebAssembly.....	13
2.3.2.1    Grundlagen: Isolation und sichere Kommunikation .....	13
2.3.2.2    Heterogene Migration durch Ahead-of-Time (AOT) Kompilierung .....	14
2.3.2.3    Optimierung der Migrationslatenz .....	14
2.3.2.4    Validierung und Messergebnisse.....	15
2.3.3    Policy-basiertes Task-Reallocation .....	16
2.4    AP4: HW/SW-Komponenten und Module: Field2Edge .....	16
2.4.1.1    Integration der KI-Funktionalität in dynamische Betriebsstrategien.....	16
2.4.1.2    Untersuchung von Echtzeiteigenschaften und weiterführende Arbeiten .....	16
2.5    AP5: RT-Monitoring-Diagnose-Infrastruktur .....	17
2.6    AP6: Use Cases Automotive / Automotive Produktion.....	17
2.7    AP7: Simulation, Test & Validation, Demonstration .....	18
2.7.1    Konzeption der Demonstrations- und Simulationsinfrastruktur .....	18
2.7.2    Umsetzung und Integration in thematischen Arbeitsgruppen.....	18
3    Nutzen und Verwertbarkeit der Ergebnisse und Erfahrungen .....	20
4    Fortschritt auf dem Gebiet an anderer Stelle .....	21

5	Veröffentlichung der Ergebnisse und Publikationen.....	22
6	Referenzen.....	23

## Abbildungsverzeichnis

Abbildung 1 : Schematische Darstellung Fahrzeug-Bordnetz und embedded Computing: Sensor2Edge .....	5
Abbildung 2 : Arbeitspaketübersicht und Informationsfluss.....	7
Abbildung 3 : Architektur der ROS2-basierten Beschleunigerschnittstelle .....	12
Abbildung 4 : Speicherisolierung durch die WASM-Laufzeitumgebung (Runtime/Embedder) .....	13
Abbildung 5 : Interaktionsmodell der WebAssembly-Laufzeitumgebung .....	13
Abbildung 6 : Gesamtarchitektur der WASM-Laufzeitumgebung .....	14
Abbildung 7 : Migrationsmethoden zur Latenzreduktion .....	15
Abbildung 8: Messergebnisse der heterogenen WASM-Migration .....	15
Abbildung 9 : Gesamtkonzept zur Integration und Demonstration der Projektergebnisse aus den Arbeitspaketen 3-7 .....	18

# 1 Thema und Zielsetzung des Vorhabens

Künftige embedded Compute-Bordnetze im Fahrzeug zeichnen sich durch zweierlei aus. (1) Stärkere Zentralisierung unter Verwendung von Hochleistungs-Car-Servern, die eine Bündelung unterschiedlicher Applikationen erlauben. (2) Sensor-/aktuatornahe Integration von Rechenleistung zur dezentralen Datenvorverarbeitung/Steuerung, für die Entlastung der energieintensiven Onboard-Kommunikation.

Forschungsleitende Hypothese: Zur wirtschaftlichen Realisierung der Compute-Bordnetze bei hohem Saftley-Level (ASIL-D) braucht es KI-basierte HW-gestützte Monitoring- und Diagnose-Systeme und dynamische Allokation der vorhandenen Rechenressourcen - auch um HW-Redundanz im System zu minimieren

Die Systemarchitekturen im Automobil befinden sich im radikalen Wandel. Die jüngste Entwicklungsexplosion in der Sensor-Technologie für die Unterstützung aktiver Sicherheit, Trends wie IoT, Fahrer-Assistenzsysteme, Elektro-Mobilität oder Autonomes Fahren machen eine Abkehr von traditionellen Architekturen aus vernetzten ECUs bzw. PLCs unabdingbar. Die Steuerung und Überwachung der Vielzahl hinzugekommener Sensoren und Aktoren durch individuelle ECUs/PLCs ist schlicht nicht mehr wirtschaftlich. Hinzu kommt die Problematik mit der zusätzlichen Verkabelung und der daraus resultierenden Minderung der Zuverlässigkeit. Der Trend zum universellen Einsatz von KI in den Steuerungen, Fragestellungen wie SOTA und der damit zusammenhängenden Cyber-Security verschärfen die Lage weiter.

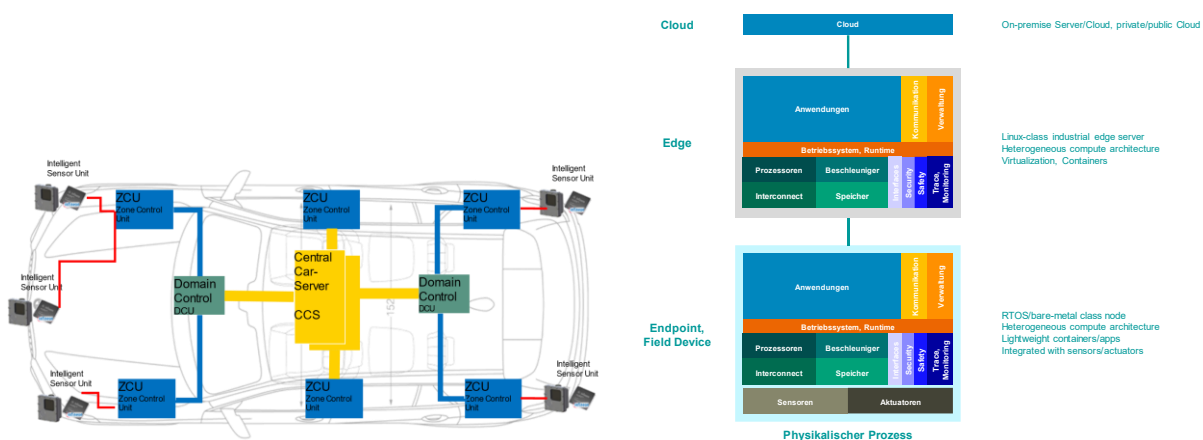


Abbildung 1: Schematische Darstellung Fahrzeug-Bordnetz und embedded Computing: Sensor2Edge

Durch die Einführung von Zonen-/Domänen-Controllern bzw. eines leistungsfähigen Car- oder Edge Servers können ECUs virtualisiert, und somit die Zahl der ECUs/PLCs signifikant reduziert werden. Voraussetzung ist allerdings eine Unix-basierte Rechereinheit mit MMU und Hypervisor. Dies geht einher mit einer signifikanten Reduktion der Verkabelung, aber zu Lasten der Kommunikations-Bandbreite auf den Netzwerken. Ein weiterer zu beachtender Aspekt sind die zusätzlichen Anforderungen, um auf solchen Multi-Funktions-Controllern funktionale- und Cyber-Sicherheit zu gewährleisten. Vollautonomes Fahren nach Level 5 erfordert darüber hinaus eine Auslegung nach ASIL-D, was sehr hohe Anforderungen an die Zuverlässigkeit der Systeme und Halbleiter-Bauelemente stellt.

## 1.1 Zielsetzung des Teilvorhabens

Die verteilte Verarbeitung großer Datenmengen und komplexer rechenintensiver Funktionen wie z. B. KI-basierte Anwendungen bringen eine große Herausforderung bei der Realisierung eines effizienten, robusten und widerstandsfähigen Systems mit sich. Die Komplexität und Menge an alternativen Verteilungen erfordert intelligente dynamische Betriebsstrategien, die automatisch auf z. B. Funktionsausfälle, Angriffe, neue Komponenten usw. reagieren und sich anpassen können.

**Aufgabe:** Derzeit werden große KI-basierte Anwendungen im autonomen Fahren hauptsächlich in einer zentralen Einheit berechnet, wobei zunehmend die Berechnungen für eine übergreifende Funktion wie die Umgebungserkennung einer komplexen Kreuzung oder das Fahren in Kolonnen verteilt werden können. Das FZI erforscht in diesem Projekt Methoden zur intelligenten und selbstadaptiven Lastverteilung, um diese mit geeigneten neuen Hardwarearchitekturansätzen zu unterstützen.

**Vorgehen:** Im Rahmen des Projekts wurden verteilte Funktionen und bestehende Methoden analysiert und weiterentwickelt. Insbesondere wurden Mechanismen zur Umsetzung von dynamischen Betriebsstrategien und Lastverteilungsansätzen untersucht, die mit Hardwareunterstützung realisiert werden können. Dazu wurden Container- und Orchestrierungslösungen mit Hypervisor-Lösungen verglichen und deren Einsatz evaluiert. Auf dieser Basis wurden Lösungen für die Ansteuerung von HW/SW-Modulen und Beschleunigern unter Berücksichtigung von Safety- und Security-Bedingungen erforscht. Darüber hinaus wurden Simulationslösungen zur Demonstration und Evaluation der Ergebnisse integriert. Somit wird die HW/SW-Plattform und der Kommunikation so betrachtet, dass eine robuste und selbstadaptive Verarbeitung erreicht werden kann.

**Ziel:** Eine Hardware-Plattform, die dynamischen Betriebsstrategien unterstützt, um robuste und ausfallsichere Edge-Computing-Systeme zu realisieren.

**Output:** Moderne HW/SW-Architekturansätze, die den Anforderungen an eine robuste und ausfallsichere verteilte Datenverarbeitung gerecht werden, neue Methoden zur Lastverteilung und Selbstadaption unter Berücksichtigung der Hardware.

## 1.2 Planung und Ablauf des Vorhabens

Das Vorhaben folgte dem untenstehenden Arbeits- und Zeitplan zur Erreichung der Projektziele.

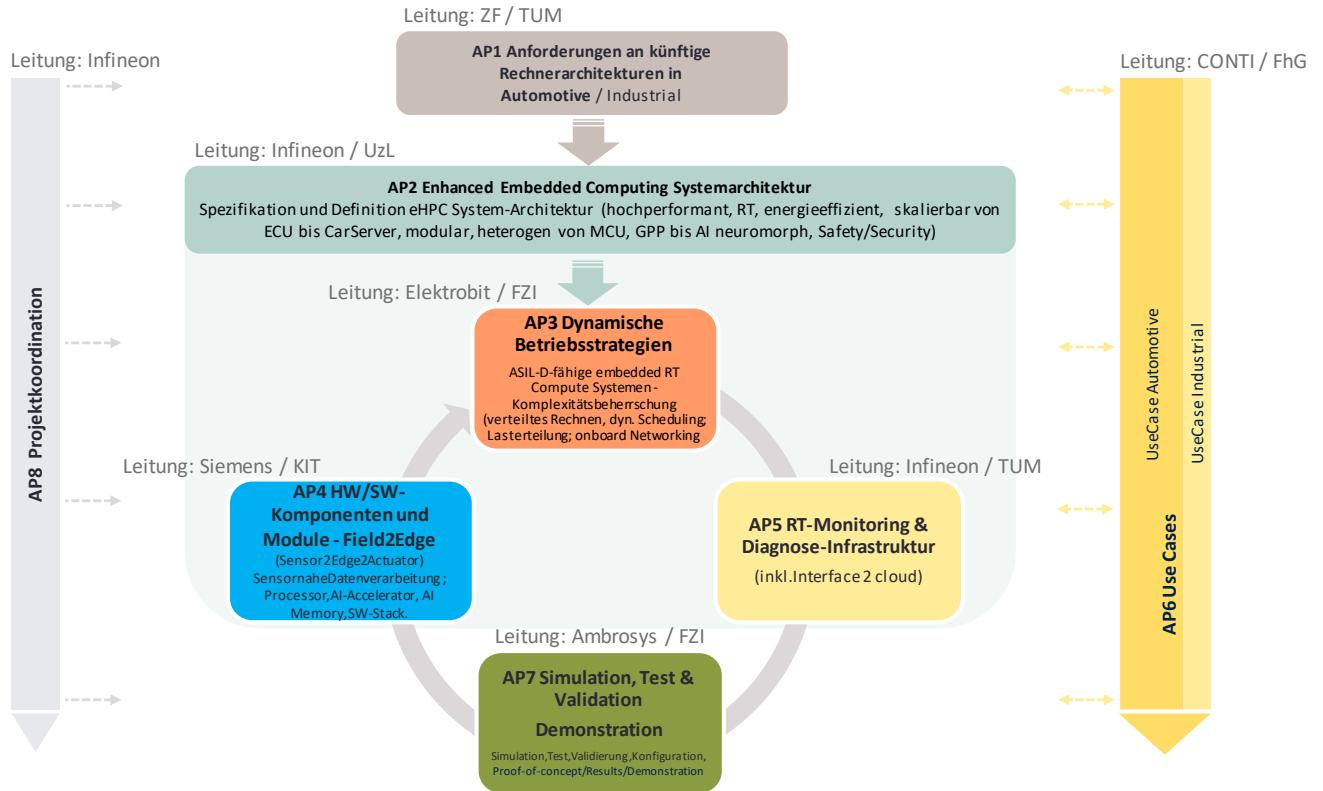


Abbildung 2 Arbeitspaketübersicht und Informationsfluss

	2022				2023				2024			
	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4	Q1	Q2	Q3	Q4
AP1		M1.1		M1.2								
AP2						M2.1					M2.2	
AP3						M3.1					M3.2	
AP4						M4.1				M4.2		
AP5						M5.1					M5.2	
AP6						M6.1					M6.2	
AP7								M7.1				M7.2

Nr.	Meilenstein	AP	Fällig
M1.1	Anforderungskatalog Rechnerarchitekturen und –netzwerke Automotive, Industrie V1	AP1	Q2 2022
M1.2	Anforderungskatalog Rechnerarchitekturen und –netzwerke Automotive, Industrie, V final	AP1	Q4 2022
M2.1	Generisches Konzept: Enhanced Embedded Computing Systemarchitektur, Version V1	AP2	Q2 2023
M2.2	Generisches Konzept: Enhanced Embedded Computing Systemarchitektur, V final	AP2	Q3 2024
M3.1	Dynamische Betriebsstrategien zur situations-optimierten Steuerung von Auto/Industrie RT-Rechennetzwerken, V1	AP3	Q2 2023
M3.2	Dynamische Betriebsstrategien zur situations-optimierten Steuerung von Auto/Industrie RT-Rechennetzwerken, V final	AP3	Q3 2024
M4.1	Architekturkonzept für die dynamische Lastverteilung und Beschleunigung von komplexen Workloads	AP4.2	M12
M4.2	Konzeptauswahl für Safety- und Security-Maßnahmen abgeschlossen	AP4.3	M18
M4.3	Beschleunigerentwurf und Architekturexploration abgeschlossen	AP4.4	M24
M4.4	Hardwareimplementierung und Software für die Architekturkomponenten	AP4.4	M30
M4.5	Demonstration und Evaluation der Komponenten	AP4.5	M36
M5.1	Definition RT-Monitoring-Diagnose-Infrastruktur	AP5	Q4 2022
M5.2	Prototypische Realisierung RT-Monitoring-Diagnose-Infrastruktur	AP5	Q3 2024
M6.1	Definition Use Case Automotive und Use Case Industrie	AP6	Q4 2022
M6.2	Validierung Use Case Automotive und Use Case Industrie	AP6	Q4 2024
M7.1	Simulations- und Implementierungsinfrastruktur Automotive / Industrie	AP7	Q4 2023
M7.2	Demonstration Use Case Automotive und Use Case Industrie	AP7	Q4 2024

### 1.3 Stand der Wissenschaft und Technik zu Beginn des Vorhabens

EMDRIVE untersucht neue Edge-Computing-Plattformen und -Systeme für die Sensor-Signalverarbeitung und Aktuator-Steuerung mit Schwerpunkt Automotive und Transfer in die Industrie. Angesichts stets steigender Komplexität gilt es, bessere On-Chip Monitoring-Lösungen zu integrieren. EMDRIVE setzt auf Monitoring-Lösungen in Verbindung mit KI-Verfahren für die Eigenüberwachung und die Erkennung von IT-Sicherheitsangriffen. Dazu werden geeignete Hardwarebeschleuniger eingesetzt.

#### Edge-Plattformen und Beschleuniger

Stand Wissenschaft/Technik: Mit den großen Datenmengen, sowohl in Form von Sensor- als auch Diagnosedaten von Prozessoren, kommen Methodiken für die Verarbeitung, Analyse und Kompression der Daten auf. Um Latenzen und Datenaustausch zu minimieren, ist eine Analyse auf der Edge (statt Cloud) von Vorteil. Allerdings erweitert das die gängigen Anforderungen an Edge-Plattformen. Die Voraussetzungen an Edge-Plattformen um Hochleistungsalgorithmen auszuführen, wie sie etwa zur Realisierung hochautomatisierter Fahrzeuge benötigt werden, setzen sich wesentlich aus folgenden Faktoren zusammen: Speicherbandbreite, Kommunikation, General-Purpose CPU, KI-spezifische GPU und Beschleuniger, sowie Vor- und Nachverarbeitung verschiedener Sensordaten. Da sie nach Stand der Technik in der Regel in einem zentralen Steuergerät integriert werden, entsteht durch die Datentransporte eine höhere Verlustleistung, Um dem entgegenzuwirken, liegt ein Fokus laufender Forschung auf heterogenen CPU-FPGA Plattformen. Wie z.B. in [1] dargestellt, erlauben sie eine Rekonfiguration der HW-Beschleuniger entsprechend den Applikationsanforderungen zur Steigerung von Performanz und Energieeffizienz. Beschleuniger für neuronale Netze sind seit Jahren Gegenstand der Forschung. Sie bieten optimierten Datenfluss und Energieeffizienz [2-6], müssen jedoch anwendungsspezifisch gewählt und angepasst werden. Für das Cloud und Edge-Computing kann z.B. in der Microsoft Azure Cloud mit Akri (<https://github.com/deislabs/akri>) Unterstützung für Edge-Devices integriert werden. Akri ist in der Lage, die Sensoren und Aktoren der On-Site IoT Landschaft managen. Eine Alternative bietet die Kombination von Edge Version von Kubernetes. Desweiteren können Edge-Devices mit KubeEdge (<https://kubedge.io/>) oder OpenYurt (<https://openyurt.io>) in public und private Clouds durch Kubernetes verwaltet werden. Auch hier ist die Anbindung von IoT Sensoren und Aktoren vorgesehen. Lösungsansatz: Der komplexe Entwurfsprozess eines KI-Beschleunigers kann durch Simulations- und Entwurfsexplorationsframeworks vereinfacht und verallgemeinert werden. Bislang aber sind sie entweder durch eine bereits vorgegebene Struktur beschränkt [7-9] oder können lediglich den Datenfluss modellieren [10]. In EMDRIVE wird das Algorithmen/Beschleuniger Co-Design weiter entwickelt. Ziel ist es, automatisiert für die entsprechende Anwendung einen hinsichtlich Energieeffizienz, Performance und Kosten optimierten Beschleuniger zu entwerfen. Auf der Betriebssystem-Ebene wird die Virtualisierung der Akzeleratoren auf Edge-Plattformen untersucht, um dynamische Rekonfigurierbarkeit zu erlangen [11]. Hersteller wie Denso, Tesla Ford und Mercedes haben das Problem erkannt und arbeiten an einer auf Kubernetes aufbauenden Lösung – was die Relevanz der Thematik unterstreicht. (<https://thechief.io/c/editorial/how-kubernetes-is-shaping-the-future-of-cars/>. <https://customers.microsoft.com/en-us/story/784791-mercedes-benz-r-and-d-creates-container-driven-cars-powered-by-microsoft-azure>). Jedoch steht u.a. die Sicherheit dieser Systeme infrage; die hier ent-wickelten Lösungen werden den Sicherheitsaspekt besser einbeziehen.

#### Betriebssystem (OS)

Stand Wissenschaft/Technik: Der Trend zu zentralen Steuergeräten, die aus multi-core Computersystemen und Beschleunigern aufgebaut sind, erfordert auch neue Ansätze bei der zugrundeliegenden Softwarearchitektur des Gesamtsystems. Die Anforderungen an ein solches System

sind, dass verschiedenste Softwarekomponenten unabhängig auf der gleichen leistungsstarken Hardware ausgeführt werden können und sowohl IT-Sicherheit wie auch Betriebssicherheit eingehalten werden. Gerade auch durch die Vernetzung der Systeme und die gestellten Anforderungen der Dynamik des Systems ist dies eine Herausforderung. Mikrokernsysteme, wie L4Re, bilden die Grundlage, die geforderten Eigenschaften umzusetzen. Lösungsansatz: Das L4Re Betriebssystem wurde in seiner Historie mit Hinblick auf diese Dynamik konstruiert und kann dazu beitragen, den Übergang von statischen Systemkonfigurationen in automobilen Anwendungen hin zu dynamischen Systemen umzusetzen.

### **Self-X Analyse der Elektronik-Hardware**

Stand Wissenschaft/Technik: Die erfassten Sensor- und Monitoring-Daten gestatten zudem weitere Auswertungen, die auch Verschleiß und Lebensdauer des Systems überwachen. Ermüdung, Riss und Bruch in Elementen der Aufbau- und Verbindungstechnik sind zentrale Risiken für die Zuverlässigkeit elektronischer Komponenten und Systeme. Die typische Lebensdauer des Systems unter Testbedingungen lässt sich per Simulation abschätzen [31]. Eine zuverlässigkeitskonforme Auslegung (Design for Reliability) und der Produktionsstart neuer Produkte ist damit sehr rasch möglich. Für den Nutzer jedoch zählt die Restlebensdauer der Elektronik des Einzelgeräts unter konkreten Betriebsbedingungen. Die nötige funktionale Sicherheit auch für autonome Systeme allein durch Redundanz und Sicherheitsfunktionen zu realisieren, wäre nicht akzeptabel [32]. Entwicklungsaufwand, Material- und Energiebedarf, Gesamtgewicht und nicht zuletzt die Kosten der zahlreicher werdenden Redundanzen würden zu hoch. Auch bleibt unbekannt, wie lange die redundante Komponente den Ausfall der ersten überlebt. Daher laufen seit gut 5 Jahren intensive Forschungsaktivitäten zur Eigenüberwachung der Elektronik und zum Abschätzen der konkreten Restlebensdauer [33-37], um Wartung, Reparatur und Austausch ressourcensparsam bedarfsgerecht aber dennoch rechtzeitig vor einem Funktionsversagen vornehmen zu können. Der Schwachpunkt dieser Anstrengungen ist, dass die ermittelte Restlebensdauer geringer sein kann, als die Erfüllung der aktuellen Mission erfordert. Benötigt werden aktive Vorkehrungen zur bestmöglichen Aufrechterhaltung der Funktion bis zur nötigen Wartung. Lösungsansatz: Das Problem mittels eines integrierten, intelligenten, nicht-invasiven, sicheren und verteilten Monitoring und Diagnoseansatz angehen. Integriert, weil bereits existierende Monitore und Diagnoseinstanzen in Systemkomponenten (z.B. Aurix MCDS) in den neuen Ansatz einbezogen werden. Intelligent, weil KI für die Diagnose von Effekten zwischen verschiedenen Komponenten/Codes, sowie das Identifizieren bisher unkekannter Fehler/Effekte im Edge Device eingesetzt wird und auch während Ruhephasen des Systems vorhandene Ressourcen zu Diagnosezwecken genutzt werden sollen („Dreaming“).

## **1.4 Angaben bekannter Konstruktionen, Verfahren und Schutzrechte, die für die Durchführung des Vorhabens benutzt wurden**

Es konnten keine Patente identifiziert werden, die diesem Vorhaben entgegengestanden oder die Ausübungsfreiheit eingeschränkt hätten.

## **1.5 Zusammenarbeit mit anderen Stellen**

Das Vorhaben erforderte keine weitere Zusammenarbeit mit anderen Stellen außerhalb des Projekts.

## 2 Ergebnisse

### 2.1 AP1: Anforderungen an künftige Rechnerarchitekturen in Automotive / Industrie

Die zunehmende Komplexität und Vernetzung moderner Fahrzeugarchitekturen, insbesondere im Kontext des autonomen Fahrens, stellt enorme Anforderungen an die zugrundeliegende Rechenplattform. Um Aspekte wie Energieeffizienz, funktionale Sicherheit (Safety), Angriffssicherheit (Security) sowie Resilienz ganzheitlich zu adressieren, ist eine präzise Definition der Systemanforderungen unerlässlich. Das zentrale Ziel von AP1 war daher die Erstellung eines vereinheitlichten Anforderungskatalogs, der als gemeinsame Grundlage für die technischen Entwicklungen in den nachfolgenden Arbeitspaketen (AP2-AP7) dient.

Dieser Katalog sollte nicht nur statische Anforderungen erfassen, sondern insbesondere die in EMDRIVE angestrebten dynamischen Betriebsstrategien berücksichtigen. Diese ermöglichen es, die Systemressourcen zur Laufzeit adaptiv an die jeweilige Fahrsituation und den Systemzustand anzupassen.

#### 2.1.1 Beitrag des FZI und abgeleitete Kernanforderungen

Das FZI brachte in diesem Arbeitspaket seine Expertise im Bereich dynamischer Betriebsstrategien und flexibler Softwarearchitekturen ein, welche den Schwerpunkt der FZI-Arbeiten in AP3 bilden. In gemeinsamer Arbeit mit den Projektpartnern wurden die Anforderungen an die zukünftige Compute-Architektur konsolidiert. Aus der Perspektive der dynamischen Lastverteilung und des flexiblen Einsatzes von Software-Komponenten ergaben sich für das FZI die folgenden zentralen Anforderungen an die Systemarchitektur, die in den Katalog eingebracht wurden:

- **Multicore-HPC-System mit Virtualisierungsunterstützung:** Die Entwicklung von dynamischen Betriebsstrategien, wie die Migration von Anwendungen (siehe AP3), erfordert eine strikte Kapselung der einzelnen Software-Komponenten. Eine Virtualisierung auf Hardware-Ebene ist die Grundvoraussetzung, um diese Isolation zu gewährleisten und die flexible Zuweisung von Rechenressourcen zu virtualisierten Maschinen oder Containern zu ermöglichen.
- **Heterogene Rechenplattform:** Um die Vorteile dynamischer Lastverteilung demonstrieren zu können, muss die Architektur verschiedene Arten von Recheneinheiten umfassen. Dazu gehören sowohl universelle Prozessoren (z. B. ARM-Multicore) als auch spezialisierte Hardware wie FPGA- oder ASIC-basierte KI-Beschleuniger. Nur auf einer solchen heterogenen Plattform kann die im Projekt erforschte, situationsbedingte Verlagerung von Aufgaben auf die jeweils am besten geeignete Hardware validiert werden.
- **Bedarf an realitätsnahen Anwendungen und Eingangsdaten:** Dynamische Betriebsstrategien reagieren auf veränderliche Lasten, die durch die Ausführung von Anwendungen entstehen. Um die entwickelten Mechanismen zur Lastverteilung und Migration unter realistischen Bedingungen testen und demonstrieren zu können, ist ein Satz von relevanten Anwendungen (z. B. aus dem Bereich der Umfeldwahrnehmung) samt zugehöriger Eingangsdaten – ob aus Datensätzen oder Simulationen – zwingend erforderlich.

Diese im Arbeitspaket 1 definierten Kernanforderungen bildeten eine entscheidende Grundlage für die weiteren Forschungs- und Entwicklungsarbeiten des FZI in den Arbeitspaketen AP3, AP4 und AP7.

## **2.2 AP2: Enhanced Embedded Computing Systemarchitektur**

Aufbauend auf dem in AP1 erstellten Anforderungskatalog war das Ziel von AP2 die Konzeption und Spezifikation einer modularen, skalierbaren und sicheren Enhanced Embedded Computing Systemarchitektur. Diese Architektur dient als technischer Entwurfsrahmen für die in den Folge-Arbeitspaketen zu entwickelnden Hard- und Softwarekomponenten.

Der Beitrag des FZI lag hierbei in der unterstützenden Analyse und Definition von Architektureigenschaften aus der Perspektive der dynamischen Betriebsführung. Um die in AP3 entwickelten dynamischen Mechanismen, wie die anwendungs-basierte Lastverteilung, effektiv umsetzen zu können, muss die Systemarchitektur entsprechende Voraussetzungen erfüllen. Das FZI hat daher die Anforderungen an die CPU-Leistung und die Systemschnittstellen im Hinblick auf die dynamischen Anwendungsfälle untersucht. Ein besonderer Fokus lag auf der Betrachtung der Balance zwischen zentralen Servern, Domänen-Controllern und Edge-Devices.

Des Weiteren unterstützte das FZI bei der Ableitung von Schnittstellendefinitionen, die für die Ansteuerung und Orchestrierung durch die in AP3 entwickelten Module notwendig sind. Dies stellte sicher, dass die in AP2 entworfene Systemarchitektur die erforderlichen Funktionalitäten bereitstellt, um eine adaptive und ressourcenoptimierte Lastverteilung zur Laufzeit zu ermöglichen und die entwickelten Softwarebausteine nahtlos zu integrieren.

## **2.3 AP3: Dynamische Betriebsstrategien**

In AP3, welches vom FZI gemeinsam mit Elektrobit geleitet wurde, lag der Forschungsschwerpunkt auf der Entwicklung und Umsetzung von dynamischen Betriebsstrategien. Ziel war es, eine flexible Verarbeitungsarchitektur zu schaffen, die es ermöglicht, auf Basis von Systemzuständen, sich ändernden Anforderungen oder drohenden Hardware-Ausfällen eine adaptive und ressourcenoptimierte Lastverteilung im Gesamtsystem – von der Sensorik bis zu den zentralen Recheneinheiten – vorzunehmen. Die Arbeiten des FZI konzentrierten sich auf zwei technologische Säulen, die für die Realisierung solcher Strategien fundamental sind: eine generische Schnittstelle zur dynamischen Anbindung von Hardware-Beschleunigern und ein Framework zur Migration von Anwendungen über heterogene Rechenknoten hinweg.

### **2.3.1 Generische Schnittstelle zur dynamischen Beschleuniger-Anbindung**

Moderne automobiler Systeme sind zunehmend mit heterogener Hardware ausgestattet, die spezialisierte Beschleuniger für KI-Anwendungen umfasst. Eine statische Zuweisung dieser wertvollen Ressourcen zu einzelnen Anwendungen führt zu einer ineffizienten Auslastung. Um dieses Problem zu adressieren, wurde im FZI ein Konzept für eine generische, service-orientierte Beschleunigerschnittstelle entwickelt.

Der Vorteil einer solchen Abstraktionsebene liegt darin, die im Gesamtsystem vorhandene Beschleunigerhardware dynamisch für verschiedene Anwendungen nutzbar zu machen. Mehrere

Anwendungen können sich so einen oder mehrere Beschleuniger teilen, ohne dass die Anwendungsentwickler detaillierte Kenntnisse über die darunterliegende Hardware besitzen müssen. Dies erhöht die Ressourceneffizienz und die Portabilität von Software über verschiedene Hardware-Plattformen hinweg.

Die konzipierte Architektur basiert auf dem Robot Operating System 2 (ROS2) als Kommunikations-Middleware und besteht aus drei Kernkomponenten, die in Abbildung 3 zu sehen sind.

- **Hardware Manager:** Diese zentrale Instanz verwaltet alle verfügbaren Beschleunigerressourcen im System. Sie nimmt Anfragen von Anwendungen entgegen und entscheidet, auf welchem Beschleuniger eine Aufgabe (z. B. die Inferenz eines KI-Modells) ausgeführt wird.
- **Acceleration Runner:** Für jeden Beschleunigertyp im System existiert ein spezifischer "Runner". Er ist dafür verantwortlich, Modelle auf die jeweilige Hardware zu laden, auszuführen und die Ergebnisse zurückzuliefern. Diese Runner sind als standardisierte "Plug-ins" konzipiert, um eine hohe Modularität zu gewährleisten.
- **Modell-Datenbank:** In dieser Datenbank werden KI-Modelle gespeichert. Um maximale Kompatibilität zu gewährleisten, wurde das Open Neural Network Exchange (ONNX) Format als Standard für die Modelle definiert. Dies erlaubt es, die jeweiligen herstellereigenen, hochoptimierten Toolchains zur Konvertierung der Modelle für die Zielhardware zu nutzen.

Dieses in AP3 entworfene Software-Framework schafft die grundlegenden Voraussetzungen, um eine dynamische und flexible Nutzung von Hardware-Beschleunigern in einem komplexen Gesamtsystem zu ermöglichen.

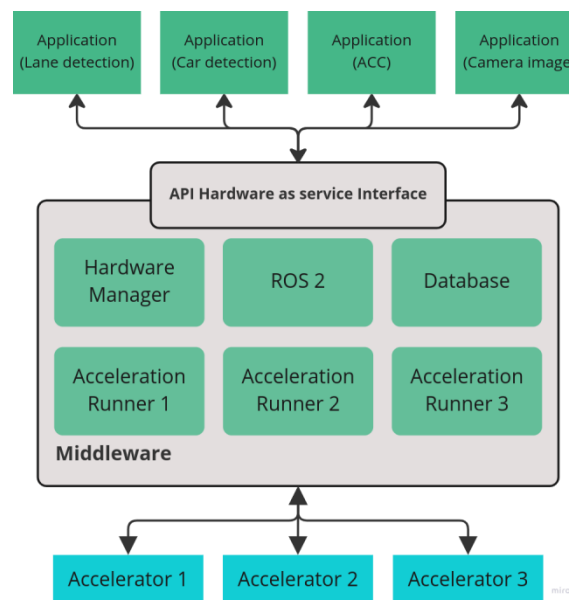


Abbildung 3: Architektur der ROS2-basierten Beschleunigerschnittstelle

## 2.3.2 Migration von Anwendungen in heterogenen Systemen mittels WebAssembly

Der zweite Forschungsschwerpunkt des FZI war die Entwicklung eines Frameworks zur Migration von gekapselten Anwendungen zwischen verschiedenen Recheneinheiten zur Laufzeit. Dies ermöglicht es, auf Lastspitzen zu reagieren, Energie durch das Abschalten ungenutzter Einheiten zu sparen oder Anwendungen von einem von Degradation betroffenen Knoten zu evakuieren. Als Basistechnologie wurde WebAssembly (WASM) gewählt, da es eine plattformunabhängige, effiziente und sichere Ausführung von Code in einer Sandbox ermöglicht.

### 2.3.2.1 Grundlagen: Isolation und sichere Kommunikation

Um Interferenzen zwischen den betrachteten Anwendungen und dem restlichen System zu verhindern, wurde die strikte Speicherisolation, die von WASM vorgegeben wird, weiterverwendet. Wie in Abbildung 4 dargestellt, wird jede WASM-Anwendung in einer eigenen Sandbox mit einem dedizierten Speicherbereich und Stack ausgeführt.

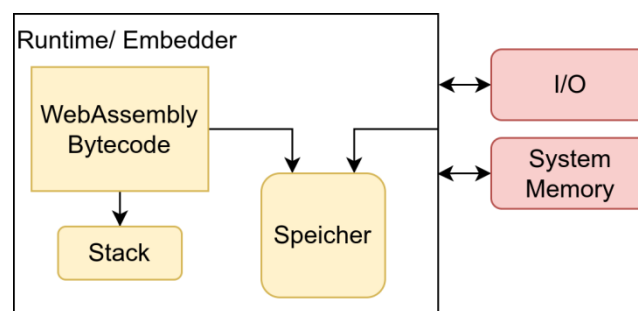


Abbildung 4: Speicherisolation durch die WASM-Laufzeitumgebung (Runtime/Embedder)

Der Zugriff auf I/O-Schnittstellen oder andere Systemressourcen wird ausschließlich von der Laufzeitumgebung (Runtime) verwaltet. Diese kopiert benötigte Kommunikationsnachrichten gezielt in den Speicher der Anwendung und leitet ausgehende Nachrichten weiter. Die Interaktionen zwischen der Runtime und den WASM-Modulen sind klar definiert und umfassen Ereignisse wie Initialisierung, Stopp, zeitgesteuerte Auslöser oder das Eintreffen von Kommunikationsnachrichten (Abbildung 5).

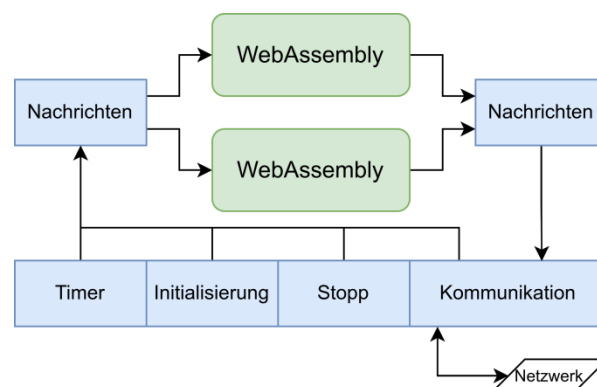


Abbildung 5: Interaktionsmodell der WebAssembly-Laufzeitumgebung

Ein integrierter Access-Control-Filter für ein- und ausgehende Nachrichten stellt zudem sicher, dass eine Anwendung nur auf die für sie freigegebenen Teile des Systems zugreifen kann.

### 2.3.2.2 Heterogene Migration durch Ahead-of-Time (AOT) Kompilierung

Eine zentrale Herausforderung war die Migration über unterschiedliche CPU-Architekturen hinweg (z. B. von x86\_64 auf ARM). Obwohl WASM plattformunabhängig ist, muss der Code für eine performante Ausführung in nativen Maschinencode übersetzt werden. Hierfür wurde der Ansatz der Ahead-of-Time (AOT) Kompilierung gewählt. Das entwickelte Konzept unterstützt die Migration sowohl des ursprünglichen WASM-Moduls als auch der bereits AOT-kompilierten, plattformspezifischen Artefakte. Dies ermöglicht maximale Flexibilität: Auf ressourcenstarken Knoten kann die Kompilierung direkt auf dem Gerät erfolgen, während für ressourcenbeschränkte Knoten die Artefakte per Cross-Kompilierung auf einem externen System erzeugt und dann übertragen werden können. Die Gesamtarchitektur ist in Abbildung 6 gezeigt.

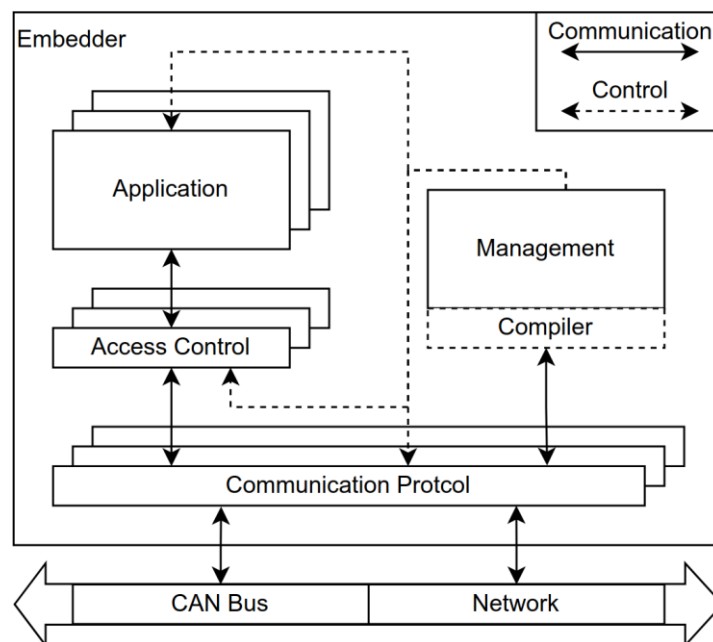


Abbildung 6: Gesamtarchitektur der WASM-Laufzeitumgebung

### 2.3.2.3 Optimierung der Migrationslatenz

Die simultane Übertragung des WASM-Moduls, der kompilierten Artefakte und des aktuellen Speicherzustands kann in zeitkritischen automotive Umgebungen zu inakzeptablen Latenzen führen. Aus diesem Grund wurde ein mehrstufiger Migrationsprozess entwickelt, der in Abbildung 7 veranschaulicht wird:

1. Vorab-Übertragung (Light/Copy Migration): Der Code des WASM-Moduls und die kompilierten Artefakte sind unveränderlich (immutable). Sie können daher unabhängig vom dynamischen Speicherzustand und in weniger zeitkritischen Phasen auf den Zielknoten übertragen werden.
2. Kritische Übertragung (State Migration): Zum eigentlichen Zeitpunkt der Migration muss nur noch das typischerweise deutlich kleinere Speicherabbild (der dynamische Zustand) transferiert werden.

Durch diese Aufteilung wird die Latenz im kritischen Pfad der Migration signifikant reduziert.

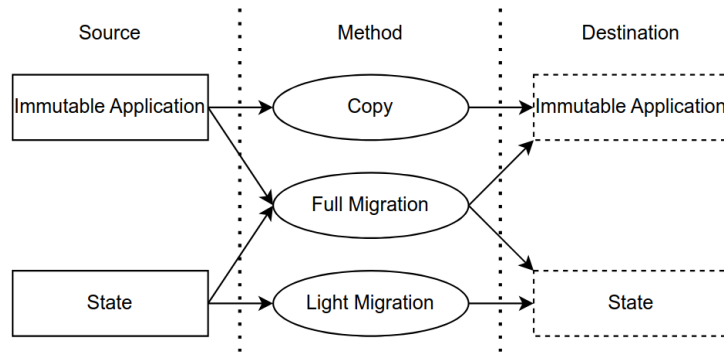


Abbildung 7: Migrationsmethoden zur Latenzreduktion

### 2.3.2.4 Validierung und Messergebnisse

Das Migrationskonzept wurde zwischen einem AMD-basierten Computer (x86\_64) und einem Raspberry Pi (ARM) validiert. Die Messungen ergaben eine durchschnittliche Migrationszeit von nur 50,73 ms für den Wechsel zwischen diesen heterogenen Systemen. Zum Vergleich: Ein Standardverfahren wie das Erstellen und Wiederherstellen eines Docker-Containers mittels Checkpoint/Restore-Technologien benötigte auf demselben Computer, ohne Netzwerkübertragung und ohne Architekturwechsel, durchschnittlich 391 ms. Dieses Ergebnis unterstreicht eindrucksvoll die Effizienz des entwickelten WASM-basierten Ansatzes.

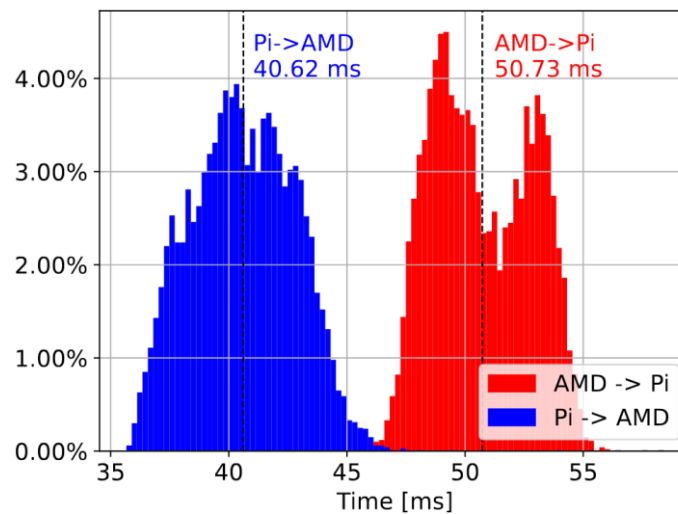


Abbildung 8 Messergebnisse der heterogenen WASM-Migration

Eine zusätzliche Untersuchung zur Übertragung von reinen Speicheränderungen (Delta-State) anstelle des gesamten Speicherabbilds wurde ebenfalls durchgeführt. Aufgrund des signifikanten Performance-Overheads durch die kontinuierliche Nachverfolgung der Speicherzugriffe wurde dieser Ansatz jedoch nicht weiterverfolgt. Die erzielten Ergebnisse wurden auf den internationalen Konferenzen EMSOFT und IEEE ICFC vorgestellt und publiziert.

### **2.3.3 Policy-basiertes Task-Reallocation**

Um die entwickelten Migrationsmechanismen in eine übergeordnete, regelbasierte Systemsteuerung einzubetten, wurde zusätzlich ein Policy-Modell für die Task-Reallokation zur Laufzeit erforscht. Dieses in der Publikation „Policy Model for Task Reallocation at Runtime in Edge Computing Infrastructures“ auf der IEEE IoTaiS vorgestellte Modell erlaubt es, Anforderungen an Systemkomponenten zur Entwurfszeit zu beschreiben. Diese Informationen werden zur Laufzeit wiederverwendet, um die Neuweisung von Tasks automatisch zu berechnen und auf Konformität mit den Designvorgaben zu prüfen. Dies erhöht die Zuverlässigkeit und Verfügbarkeit des Gesamtsystems und reduziert Ausfallzeiten.

Zusammenfassend lässt sich sagen, dass die umfangreichen Arbeiten des FZI in AP3 die wesentlichen technologischen Bausteine für eine dynamische und adaptive Steuerung von automobilen Rechnernetzwerken lieferten und somit einen entscheidenden Beitrag zur Erreichung der Projektmeilensteine M3.1 und M3.2 darstellten.

## **2.4 AP4: HW/SW-Komponenten und Module: Field2Edge**

Das übergeordnete Ziel von Arbeitspaket 4 war die Konzeption, Implementierung und Integration der Hardware- und Softwarekomponenten, die die Sensor2Edge-Gesamtplattform bilden. Dies umfasste insbesondere die Entwicklung von anwendungsoptimierten KI-Beschleunigern, sowie unterstützende Software.

### **2.4.1.1 Integration der KI-Funktionalität in dynamische Betriebsstrategien**

Die Rolle des FZI in diesem Arbeitspaket war es, die Verbindung zwischen den in AP3 entwickelten, softwarebasierten Konzepten für dynamische Betriebsstrategien und den in AP4 von den Partnern geschaffenen Hardware- und Software-Komponenten zu realisieren. Der Fokus lag auf der praktischen Anwendung, Integration und Validierung der in AP3 vorgestellten generischen Beschleunigerschnittstelle.

In enger Abstimmung wurde das Wissen aus beiden Arbeitspaketen genutzt, um die Anbindung der neuen Beschleuniger an das Gesamtsystem zu ermöglichen. Diese Integrationsarbeit umfasste die Entwicklung konkreter Acceleration Runner für die spezifische Hardware. Dabei wurden verschiedene Werkzeuge und Laufzeitumgebungen, wie der Apache TVM Machine Learning Compiler, evaluiert, um eine effiziente Ausführung zu gewährleisten. Die praktischen Erfahrungen aus dieser Arbeit flossen direkt in die Verfeinerung des in AP3 entwickelten Konzepts ein und führten zur Entscheidung, ONNX als übergreifenden Standard zu etablieren, um die Kompatibilität über die diverse Hardwarelandschaft, der heute verwendeten Beschleuniger hinweg sicherzustellen.

### **2.4.1.2 Untersuchung von Echtzeiteigenschaften und weiterführende Arbeiten**

Über die reine Integration hinaus untersuchte das FZI die Eignung aktueller FPGA-basierter KI-Beschleuniger für den Einsatz in Systemen mit harten Echtzeitanforderungen. Dabei wurden die

Schwankungen der Ausführungszeiten bei verschiedenen Workloads analysiert, um den Einfluss von Systemaspekten wie der Host-CPU-Anbindung oder dem Betriebssystem zu charakterisieren. Die Ergebnisse dieser Untersuchung wurden bei der IEEE International Conference on Dependable, Autonomic & Secure Computing vorgestellt und liefern wertvolle Erkenntnisse für die Entwicklung zukünftiger, sicherheitskritischer Systeme.

## **2.5 AP5: RT-Monitoring-Diagnose-Infrastruktur**

Die zunehmende Komplexität von Fahrzeugsystemen erfordert fortschrittliche Methoden zur Überwachung und Diagnose, um auch sporadisch auftretende Fehler im Feld zu erkennen und zu behandeln, was für die Erreichung hoher Sicherheitslevel wie ASIL-D unerlässlich ist. Das Ziel von Arbeitspaket 5 war daher die Konzeption einer integrierten, intelligenten und verteilten Monitoring- und Diagnose-Infrastruktur.

Der Beitrag des FZI bestand darin, die Brücke zwischen der in diesem Arbeitspaket entwickelten Diagnose-Infrastruktur und den in AP3 konzipierten dynamischen Betriebsstrategien, als auch mit dem Demonstrator zu schlagen.

Eine effektive Diagnose in einem dynamischen System muss dessen veränderlichen Zustand berücksichtigen. Das FZI unterstützte daher den Entwurf der Diagnose-Architektur mit der Perspektive der dynamischen Betriebsführung. Es wurde sichergestellt, dass die in AP3 entwickelten Ansätze zur dynamischen Lastverlagerung für die Entscheidung wichtige Informationen liefern. Dies konnte durch die Nutzung geteilter Runner und der gekapselten WASM-Module gewährleistet werden.

In der finalen Integrations- und Demonstrationsphase lag der Fokus des FZI auf der Evaluierung des Zusammenspiels beider Systeme. Hierbei wurden Szenarien betrachtet, in denen die Diagnose-Infrastruktur (AP5) einen Fehler oder eine Leistungsdegradation erkennt. Diese Information dient als Auslöser für das in AP3 entwickelte Framework, um eine geeignete Gegenmaßnahme, wie die Migration einer kritischen Anwendung auf eine andere Recheneinheit, einzuleiten. Die Arbeit des FZI hat somit dazu beigetragen, die Wirksamkeit dieser Ursache-Wirkungs-Kette im Gesamtdemonstrator zu bewerten und die durch die Kombination beider Technologien gesteigerte Systemresilienz nachzuweisen.

## **2.6 AP6: Use Cases Automotive / Automotive Produktion**

Arbeitspaket 6 diente der Definition und Umsetzung von konkreten Anwendungsfällen (Use Cases), um die in den technischen Arbeitspaketen entwickelten Forschungsergebnisse auf ihre Praxistauglichkeit zu untersuchen und zu demonstrieren. Die beiden zentralen Domänen waren das autonome Fahren (Automotive) und die hochautomatisierte Fertigung (Automotive-Produktion).

Die Rolle des FZI war es, die in AP3 entwickelten Methoden zur dynamischen Lastverteilung und Anwendungs-Migration in die definierten Use Cases konzeptionell einzubetten. Nach der anfänglichen Mitarbeit an der Spezifikation der Anwendungsfälle, wie dem Ausweichmanöver im teilautomatisierten Fahrbetrieb oder der vorausschauenden Wartung von Industriekomponenten, unterstützte das FZI bei der Analyse der Algorithmenverteilung. Ziel war es, die Analysealgorithmen je nach Bedarf flexibel zwischen ressourcenbeschränkten Edge-Sensoren und zentralen Recheneinheiten zu verteilen.



Durch die Implementierung dieses Konzepts im konkreten Anwendungsfall der vorausschauenden Wartung konnte eindrucksvoll demonstriert werden, wie Software-Tasks zur Laufzeit flexibel zwischen Edge-Knoten und zentralen Einheiten verlagert werden können, um auf geänderte Anforderungen zu reagieren. Diese praktische Umsetzung der FZI-Technologien in einem realitätsnahen Szenario war ein wesentlicher Beitrag zur finalen Validierung der Projektziele.

Zusammenfassend hat das FZI den Rahmen für die erfolgreiche Integration und Demonstration der Projektergebnisse mitgeschaffen und durch die Umsetzung konkreter Demonstratoren die Praxistauglichkeit der eigenen Forschungsergebnisse nachgewiesen.

### **3 Nutzen und Verwertbarkeit der Ergebnisse und Erfahrungen**

Wichtig für das FZI als gemeinnützige Forschungstransferinstitution ist insbesondere die wissenschaftliche Verwertung. Neben der industriellen Bedeutung und der Marktrelevanz der genannten Themen spiegelt sich darin auch die am FZI vorhandene Expertise wider, die in einer Vielzahl von geförderten Projekten genutzt wurde und derzeit ausgebaut wird, so dass die technischen Erfolgsaussichten als gut eingeschätzt werden können.

Durch die Vernetzung des EMDRIVE-Projekts mit anderen Projekten im Bereich der IT-Sicherheit und Automotive Elektronik, wie z.B. CeCaS, Shift2SDV, HAL4SDV und Rigoletto wird ein intensiver Forschungstransfer der Ergebnisse in die jeweiligen Arbeitsfelder gewährleistet. Damit besteht eine hohe Erfolgchance für die im Projekt definierten Ziele in wissenschaftlicher und technischer Hinsicht.

Des Weiteren bilden die Ergebnisse des EMDRIVE-Projekts die Grundlage für drei weitere Dissertationen, von denen eine im Rahmen des Projektes erfolgreich abgeschlossen wurde. Zusätzlich wurden im Rahmen des Vorhabens 5 studentische Abschlussarbeiten betreut. Die Veröffentlichungen der Ergebnisse des Teilvorhabens des FZI erfolgte in Form von vier Veröffentlichungen.

## 4 Fortschritt auf dem Gebiet an anderer Stelle

Im Verlauf des Projektzeitraums wurde ein signifikanter Fortschritt auf dem Gebiet der containerisierten Deployment-Lösungen für Automotive-Anwendungen bei anderen Forschungseinrichtungen und Industrieinitiativen beobachtet. Während zu Projektbeginn entsprechende Ansätze noch als irrelevant galten, zeigte sich insbesondere ab der Projektmitte eine zunehmende Relevanz, etwa durch die Integration solcher Technologien in Open-Source-Projekten der Eclipse SDV Community. Es entstanden mehrere externe Projekte zur Orchestrierung und zum Deployment containerisierter Anwendungen, wobei automotive-spezifische Anforderungen weiterhin Gegenstand aktiver Entwicklung sind.

Darüber hinaus wurden neue Hardware-Beschleuniger und unterstützende Software-Frameworks entwickelt, die im Rahmen des Vorhabens aufgegriffen und evaluiert wurden. Auch der Einsatz von WebAssembly in eingebetteten Systemen gewann gegen Ende des Projektzeitraums an Bedeutung, was sich in industrienahen wissenschaftlichen Publikationen und ersten industriellen Anwendungen widerspiegelt. Diese Entwicklungen zeigen, dass das Vorhaben sowohl von externen Fortschritten profitieren konnte als auch selbst Beiträge in Forschung und Entwicklung erbringen konnte.

## 5 Veröffentlichung der Ergebnisse und Publikationen

<i>Jahr</i>	<i>Konferenz</i>	<i>Ort, Land</i>	<i>Titel</i>	<i>Mehrwert für EMDRIVE</i>
2023	EMSOFT: International Conference on Embedded Software 2023	Hamburg, Deutschland	Work-in-Progress: Integrating WebAssembly into Service-Oriented Architectures for Edge Systems	Vorstellung des Konzeptes isolierte Anwendungen durch WebAssembly im automotive Kontext zu verwenden und Feedback aus dem Fachpublikum
2023	2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing	Abu Dhabi, United Arab Emirates	Characterization of Execution Time Variability in FPGA-Based AI- Accelerators	Vorstellung einer Untersuchung zur Real- Time Tauglichkeit verschiedener Hardware- Beschleuniger sowie Feedback aus dem Fachpublikum
2023	2023 IEEE International Conference on Internet of Things and Intelligence Systems (IoTaIS)	Bali, Indonesia	Policy Model for Task Reallocation at Runtime in Edge Computing Infrastructures	Konzept für das Scheduling von Lasten nach einem Regel- basierten Framework und Feedback aus dem Fachpublikum
2024	2024 IEEE 8th International Conference on Fog and Edge Computing (ICFEC)	Philadelphia, PA, USA	Migration of Isolated Application Across Heterogeneous Edge Systems	Vollständiges Konzept für die Verwendung von WebAssmebly-basierten Anwendungen in einem automotiven Umfeld und die Migration von Anwendungen zwischen heterogenen Recheneinheiten und Feedback aus dem Fachpublikum.

## 6 Referenzen

- [1] M. Damschen, M. Rapp, L. Bauer, und J. Henkel, „i-Core: A Runtime-Reconfigurable Processor Platform for Cyber-Physical Systems“, in *Embedded, Cyber-Physical, and IoT Systems: Essays Dedicated to Marilyn Wolf on the Occasion of Her 60th Birthday*, S. S. Bhattacharyya, M. Potkonjak, und S. Velipasalar, Hrsg. Cham: Springer International Publishing, 2020, S. 1–36. doi: 10.1007/978-3-030-16949-7\_1.
- [2] Y. Chen, T. Krishna, J. S. Emer, und V. Sze, „Eyeriss: An Energy-Efficient Reconfigurable Accelerator for Deep Convolutional Neural Networks“, *IEEE Journal of Solid-State Circuits*, Bd. 52, Nr. 1, S. 127–138, 2017.
- [3] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, und A. Moshovos, „Cnvlutin: Ineffectual-Neuron-Free Deep Neural Network Computing“, in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, 2016, S. 1–13.
- [4] S. Han und others, „EIE: Efficient Inference Engine on Compressed Deep Neural Network“, *CoRR*, Bd. abs/1602.01528, 2016, [Online]. Verfügbar unter: <http://arxiv.org/abs/1602.01528>
- [5] H. Kwon, A. Samajdar, und T. Krishna, „MAERI: Enabling Flexible Dataflow Mapping over DNN Accelerators via Reconfigurable Interconnects“, in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2018, S. 461–475. doi: 10.1145/3173162.3173176.
- [6] B. Zimmer et al., "A 0.32–128 TOPS, Scalable Multi-Chip-Module-Based Deep Neural Network Inference Accelerator With Ground-Referenced Signaling in 16 nm," in *IEEE Journal of Solid-State Circuits*, vol. 55, no. 4, pp. 920-932, April 2020, doi: 10.1109/JSSC.2019.2960488.
- [7] A. Parashar und others, „Timeloop: A Systematic Approach to DNN Accelerator Evaluation“, in *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2019, S. 304–315. doi: 10.1109/ISPASS.2019.00042.
- [8] A. Samajdar, Y. Zhu, P. Whatmough, M. Mattina, und T. Krishna, „SCALE-Sim: Systolic CNN Accelerator Simulator“, *arXiv preprint arXiv:1811.02883*, 2018.
- [9] F. Muñoz-Martínez, J. L. Abellán, M. Acacio, und T. Krishna, „STONNE: A Detailed Architectural Simulator for Flexible Neural Network Accelerators“, *ArXiv*, Bd. abs/2006.07137, 2020.
- [10] S. (Likun) Xi, Y. Yao, K. Bhardwaj, P. Whatmough, G.-Y. Wei, und D. Brooks, „SMAUG: End-to-End Full-Stack Simulation Infrastructure for Deep Learning Workloads“, *ACM Trans. Archit. Code Optim.*, Bd. 17, Nr. 4, Nov. 2020, doi: 10.1145/3424669.
- [11] T. Xia, Y. Tian, J.-C. Prévotet, und F. NOUVEL, „Ker-ONE: A new hypervisor managing FPGA reconfigurable accelerators“, *Journal of Systems Architecture*, Bd. 98, S. 453–467, 2019, doi: <https://doi.org/10.1016/j.sysarc.2019.05.003>.