Schlussbericht

Vorhabensbezeichnung:	MANNHEIM-EMDRIVE
Teilvorhaben:	Safety-Konzepte für dynamische Betriebsstrategien
Förderkennzeichen:	16ME0450
Zuwendungsempfänger:	Elektrobit Automotive GmbH Am Wolfsmantel 46 91058 Erlangen
Laufzeit des Vorhabens:	01.02.2022 - 31.01.2025
Autor(en):	Florian Aigner Uwe Hildebrand Oliver Jesorsky
Version/Datum:	V1.0/25.07.2025

Inhaltsverzeichnis

I. Kurzdarstellung des Vorhabens

I.1 Aufgabenstellung und wissenschaftlicher Stand

Mit dem Ziel „Autonome Fahrzeuge“ demonstrieren Versuche die technische Machbarkeit perfektionierter Systeme. Nicht aber den wirtschaftlichen Einsatz als zuverlässiges Massenprodukt. Energieverbrauch, Echtzeit-Rechenleistung und Hardware-Kosten werden künftigen Anforderungen an Fahrzeug-Compute-Bordnetze nicht gerecht. Mit immer neuen Applikationen steigt die Menge zu verarbeitender Informationen sowie die gesamte Komplexität um ein Vielfaches.

Ein Ausweg sind neue Ansätze, die Datenflut skalierbar mit minimalem Energie- und Kostenaufwand zu verarbeiten - funktional sicher, manipulationsfest, zuverlässig über die gesamte Lebensdauer.

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 Software-komponenten 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.

Übergeordnetes Ziel von EMDRIVE ist die Entwicklung eines hierarchischen, skalierbaren Plattformkonzepts für verteilte heterogene Automotive-Echtzeit Rechennetzwerk.

In seinem Teilvorhaben entwickelt Elektrobit hierarchische Sicherheitsarchitektur für den Einsatz des L4Re-Hypervisors in einem sicherheitskritischen Kontext. Teilziel des Projekts ist dabei die Integration dynamischer Migrationsmechanismen als Teil eines L4Re-Systems in dieses Sicherheitskonzept.

I.2 Ablauf des Vorhabens

Das MANNHEIM EMDRIVE Projekt ist organisatorisch in mehrere Arbeitspakete (AP) gegliedert, wie in Abbildung 1 dargestellt. Die unterschiedlichen Arbeitspakete decken dabei das gesamte Spektrum von Festlegung der Anforderungen und Architektur, über Implementierungen, bis zu Demonstrationen von Anwendungsfällen ab.

Der überwiegende Teil der Aktivitäten von Elektrobit fällt in den Bereich von AP3 „Dynamische Betriebsstrategien“, dessen Leitung auch bis Mitte 2020 von Elektrobit wahrgenommen wurde. Darüber hinaus hat Elektrobit auch in anderen Arbeitspaketen mitgewirkt und vor allem Unterstützung in AP1 and AP2 geleistet. In Abschnitt I.3 sind die wichtigsten Arbeiten Elektrobits in den einzelnen Arbeitspaketen beschrieben.

Zum Abschluss des Projekts (im Rahmen des Projekt-Abschluss-Reviews als Teil der AP3-Ergebnisse) hat Elektrobit einen Demonstator vorgeführt, der in einem „Mixed-criticality“ Szenario die Überwachungsmechanismen für sicherheitskritische Anwendungen zeigt.

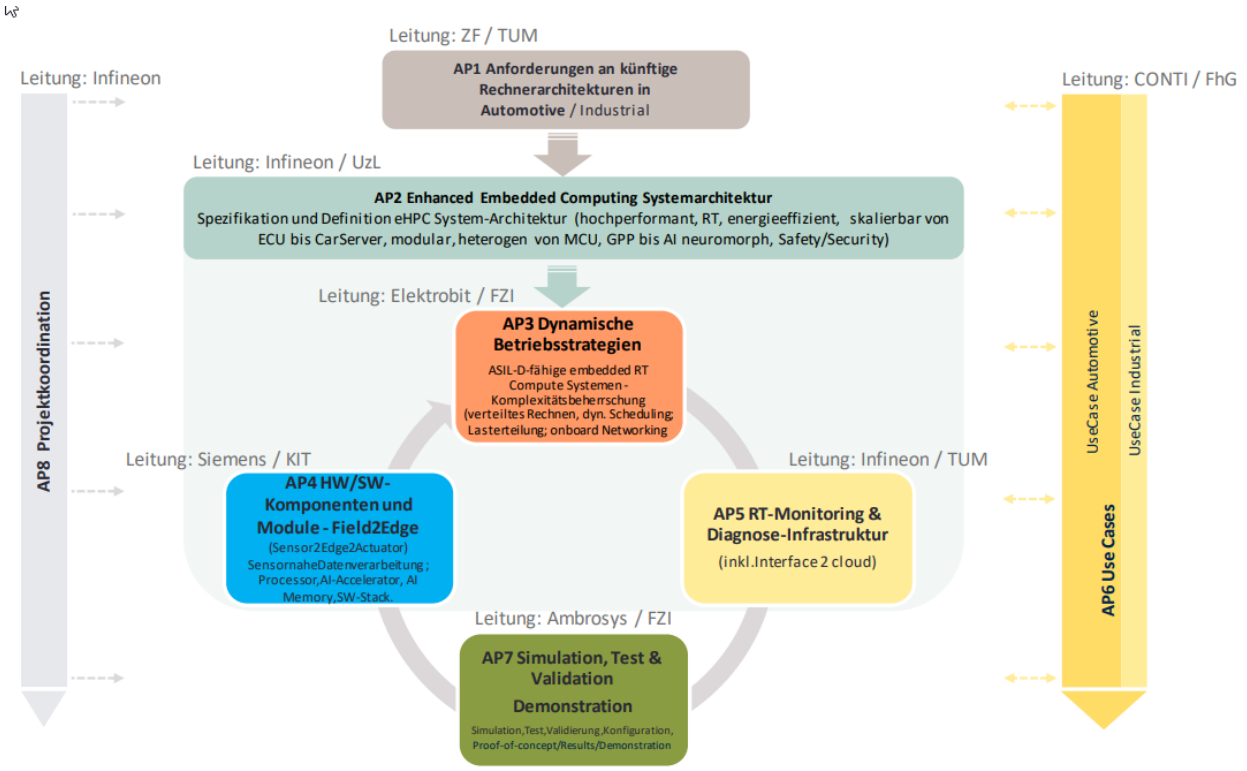

Abbildung 1: EMDRIVE Arbeitspakete

I.3 Wesentliche Ergebnisse

Die folgende Aufzählung fasst die wesentlichen Ergebnisse der Elektrobit Automotive GmbH zusammen:

Der Fokus von Elektrobits Arbeiten stand in Bezug zu AP3 „Dynamische Betriebsstrategien“ und umfasste folgende Arbeitsschwerpunkte:

- Aufbau eines Safe Execution Environments, mit Arbeiten an sicherheitsrelevanten Hypervisor-Komponenten, die letztendlich für die Bereitstellung einer Virtuellen Maschine mit sicherheitsrelevantem Integritäts-Level („Safe VM“) durch den Hypervisor notwendig sind.
- Überwachung des Gesamtsystemzustands, mit Arbeiten an dedizierten sicherheits-relevanten Hypervisor-Applikationen zur Überwachung (der Ausführung) von

sicherheitskritischen Anwendungen, sowohl von sicherheitsrelevanten Hypervisor-Applikationen, also auch von sicherheitskritischen Anwendungen in einer „Safe VM“.

- Kommunikationsschnittstellen zwischen sicherheitskritischen und nicht-sicherheits-kritischen Hypervisor-Applikationen unter besonderer Berücksichtigung von Aspekten der zeitlichen Rückwirkungsfreiheit.
- Migration von VMs/Containern zwischen ECUs und der Cloud, mit Arbeiten zum Thema Mechanismen der Softwareauslieferung (Updates, in Teilen auch Migration), sowie in Zusammenarbeit mit Infineon AG an einem Konzept für VM-Migration im Kontext von Power Management-und Error Recovery.

Darüber unterstützten Mitarbeiter von Elektrobit die Partner bei der projektweiten Festlegung von Anforderungen im Rahmen von AP1 und bei Fragestellungen zu den Themen Hardware und Architektur im Rahmen von AP2.

II. Eingehende Darstellung des Vorhabens

II.1 Verwendung und Ergebnisse

Für den Themenkomplex „Migration von VMs/Containern zwischen ECUs und der Cloud“ haben die Arbeiten der Elektrobit Automotive GmbH eine erste Prototyp-Implementierung eines Ansatzes für das initiale Aufspielen und Laden der Software von verschiedenen ECU-internen und externen Speicherorten beigesteuert. Des Weiteren wurde in Zusammenarbeit mit der Infineon AG an einem verfeinerten Konzept gearbeitet, wie Migration von virtuellen Maschinen im Kontext von übergeordneten Systemfunktionen für „Power Management“-und „Error Recovery“ verwendet werden könnte.

Ein Schwerpunkt der Arbeiten der Elektrobit Automotive GmbH lag im Bereich der Hypervisor-basierten Sicherheitsarchitektur, die das Ausführen sicherheitskritischer Anwendungen auch in einem „Mixed-criticality“ Szenario erlaubt. Die Weiterentwicklung und Verfeinerung des Hypervisor-basierten Sicherheitskonzepts stellte einen Teil dieser Arbeiten dar. Weiterhin umfassten die Arbeiten sowohl Architektur- und Design-Aufgaben für sicherheitsrelevante Hypervisor-Komponenten („Hypervisor Safety Applications“), wie auch die (Prototyp-)Implementierung der meisten dieser SW-Komponenten.

Die zugrundeliegende Hypervisor-basierte Sicherheitsarchitektur ist in Abbildung 2 dargestellt. Der als Safe Root Task bezeichneten Hypervisor-SW-Komponente kommt dabei eine zentrale Bedeutung zu. Der Safe Root Task trägt zur erforderlichen Rückwirkungsfreiheit („Freedom from Interference“, FFI) für sicherheitsrelevante SW-Komponenten durch Einführen zweier logischer Partitionen bei: Eine Partition für den sicherheitsrelevanten Teil („Hypervisor Safety Partition“) und eine weitere Partition für den nicht-sicherheitsrelevanten Teil („Hypervisor QM Partition“).

Während der Aufstartphase stellt der Safe Root Task durch geeignete Synchronisationsmechanismen sicher, dass zunächst alle in der Konfiguration festgelegten sicherheitsrelevanten Hypervisor-Anwendungen gestartet werden und dass diese auch ihre konfigurierten Ressourcen erhalten. Erst danach erfolgt das Starten der ersten nicht-sicherheitsrelevanten Hypervisor-Anwendung („QM Root Task“). Dadurch kann immer ein sicherer Startvorgang der Hypervisor Safety Applications ohne Einwirkungen aus der Hypervisor QM Partition gewährleistet werden. Durch geeignete Konfiguration der Ressourcen kann für die Hypervisor Safety Partition auch die räumliche und zeitliche Rückwirkungsfreiheit („Spatial FFI“, „Temporal FFI“) erreicht werden. Während der Laufzeit sorgt der Safe Root Task weiterhin für Rückwirkungsfreiheit, indem er als Proxy für sicherheitsrelevante Ressource-Anfragen aus der Hypervisor QM Partition fungiert. Solch sicherheitsrelevante Ressourcen können beispielsweise Speicherbereiche oder Prozessorkerne sein.

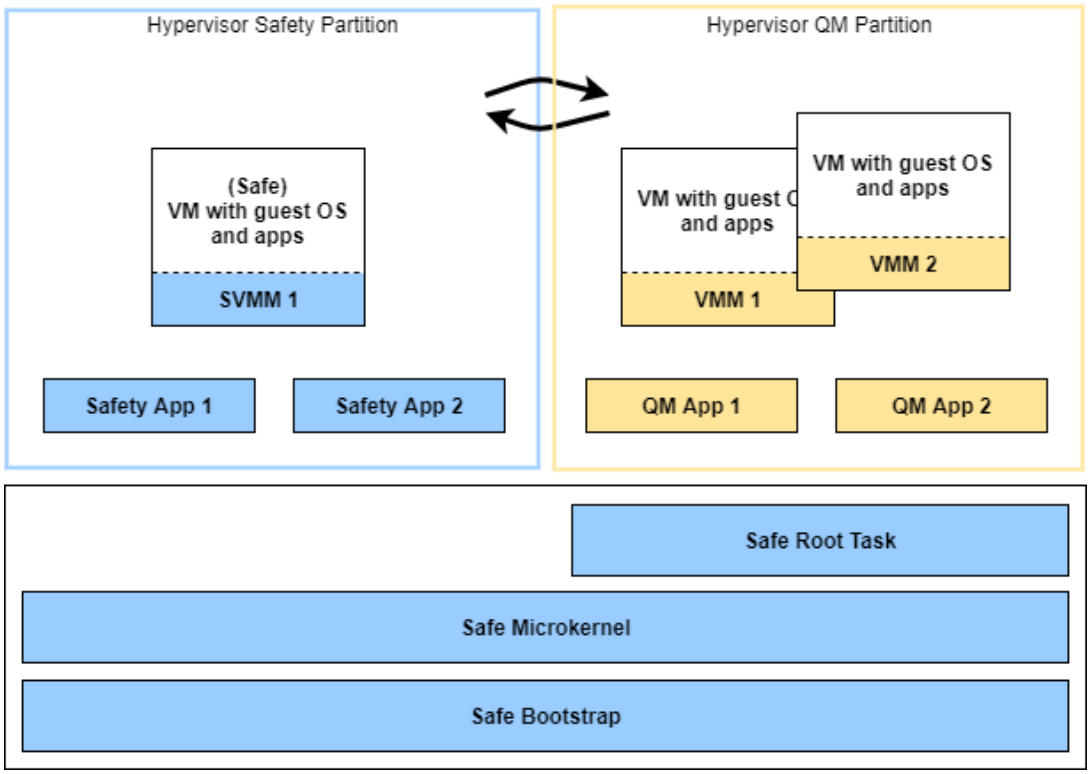

Abbildung 2: Hypervisor Sicherheitsarchitektur

Im Zusammenhang mit der beschriebenen Hypervisor-basierten Sicherheitsarchitektur wurden auch Architektur- und Designarbeiten, sowie Prototyp-Implementierungen für weitere sicherheitsrelevante Hypervisor-Anwendungen durchgeführt. Dies gilt vor allem für die unten näher beschriebene Überwachungs-Anwendung „Safe Health Monitor“. Aber auch für einen „Safe Virtual Machine Monitor“ (SVMM) als Hypervisor Safety Application, für den erste Prototyp-Implementierungen existieren. Ähnliches gilt für eine andere Hypervisor Safety Application, welche in der gewählten Architektur notwendige Funktionalitäten und Abstraktionen für die Ausführung des SVMM kapselt und bereitstellt.

Ein weiterer Arbeitsschwerpunkt lag auf einem Konzept zur Überwachung der Ausführung von sicherheitsrelevanten Anwendungen. Dabei wurden als sicherheitsrelevante Anwendungen sowohl native Hypervisor-Applikationen als auch Applikationen eines Gastbetriebssystems in einer virtuellen Maschine in Betracht gezogen. Im Fehlerfall oder bei ausbleibendem Lebenszeichen einer überwachten sicherheitsrelevanten Anwendung ist der resultierende unsichere Systemzustand in geeigneter Weise einer übergeordneten Kontollinstanz anzuzeigen.

Die zeitlich korrekte Ausführung einer sicherheitsrelevanten Anwendung in einer virtuellen Maschine kann im allgemeinen Fall von der Sicherheitsstufe des Gastbetriebssystems abhängen, auf dem diese Anwendung ausgeführt wird. Deshalb kann in einem derartigen Szenario die Überwachung einer sicherheitsrelevanten Anwendung innerhalb der virtuellen Maschine selbst unter Umständen nicht ausreichend sein.

Das erarbeitete Überwachungskonzept basiert im Wesentlichen auf einer dedizierten Hypervisor Safety-Monitoring-Anwendung („Safe Health Monitor“) und ist für eine exemplarische Konfiguration in Abbildung 3 zu sehen, die auch eine Hypervisor-Anwendung „Health Signal Handler“ als übergeordnete Kontollinstanz zeigt.

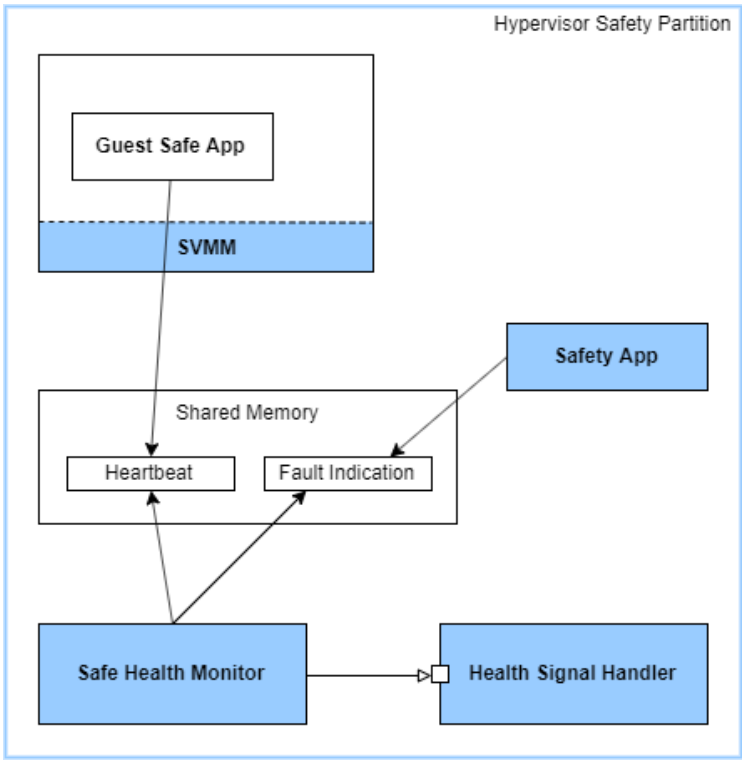

Abbildung 3: Überwachungskonzept sicherheitsrelevanter Anwendungen

Eine Instanz des Safe Health Monitors erlaubt es, jeweils eine konfigurierte Anzahl von (sicherheitsrelevanten) Anwendungen zu überwachen. Die Überwachung kann dabei entweder über regelmäßige Lebenszeichen („heartbeats“) oder über dedizierte Fehlermeldungen erfolgen und die gewünschte Überwachungsmethode kann für jede überwachte Anwendung separat festgelegt werden. Für sicherheitsrelevante native Hypervisor-Anwendungen können dedizierte Fehlermeldungen ausreichend sein, da durch die Sicherheitsstufe aller involvierten Hypervisor-Komponenten die korrekte Ausführung einer solchen sicherheitsrelevanten Anwendung bereits gewährleistet ist. Aus Gründen der Rückwirkungsfreiheit wird „Shared Memory“ als Schnittstelle zwischen Safe Health Monitor und den von ihm überwachten sicherheitsrelevanten Anwendungen verwendet.

Die primäre Funktion des Safe Health Monitors ist die Überwachung der periodischen Lebenszeichen (oder gegebenenfalls dedizierten Fehlermeldungen) aller überwachten Anwendungen und das Erzeugen eines periodischen aggregierten Ausgabesignals. Durch das periodische Senden dieses Ausgabesignals wird angezeigt, dass der Gesamtzustand aller überwachten Anwendungen (noch) „intakt“ ist. Empfänger des Ausgangssignals ist eine übergeordnete Kontroll-Komponente, die dann eine geeignete Reaktion auf das

Ausbleiben des periodischen Signals (also auf einen sicherheitskritischen Zustand) implementieren muß. Eine entsprechende Prototyp-Implementierung eines Health Signal Handlers mit minimaler Funktionalität zeigt zum Beispiel einen „intakten“ Zustand durch periodisches Umschalten eines Chip-HW-Ausgangssignals („GPIO Toggling“). Andernfalls bliebe der Pegel des HW-Signals dann konstant unverändert.

Elektrobit Automotive GmbH arbeitet in Themenbereich „Kommunikationsschnittstellen zwischen sicherheitskritischen und nicht-sicherheitskritischen Hypervisor-Applikationen“ an einem Kommunikationsprotokoll zwischen Applikationen verschiedener Kritikalität. Dies bedeutet, dass eine bidirektionale Kommunikation möglich sein soll ohne Bedingungen bezüglich zeitlicher Isolation bzw. Rückwirkungsfeiheit zu verletzen. Diese Arbeiten fokussieren sich hauptsächlich auf die Hypervisor-Schicht bzw. Kommunikation zwischen Hypervisor-Applikationen.

Die simpelste Umsetzung stellt ein sogenanntes „Shared memory“ dar, welches ohne zeitlichen Einfluss der Gegenseite gelesen und geschrieben werden kann. Allerdings hat dies Nachteile, wie zum Beispiel Latenzen bis die Nachricht empfangen wird sowie eine gewisse Grundlast (verursacht durch Polling des „Shared memory“).

Dem gegenüber erlaubt eine Interrupt-basierte Kommunikation eine fast latenzfreie Kommunikation. Der Kommunikationspartner wird nur aktiv, wenn auch neue Daten angefallen sind. Allerdings verletzt dieses Grundkonzept die zeitliche Isolation bzw. Rückwirkungsfeiheit, da die Partner unbegrenzt Kommunikationsanfragen stellen können.

Werden die beiden oben beschriebenen Ansätze kombiniert, dann kann mit einem geeigneten Betriebssystem ein Ansatz basierend auf „Reply capabilities“ genutzt werden. Hierbei erteilt die höher privilegierte/vertrauenswürdige Applikation einem niedriger privilegierten/vertrauenswürdigen Kommunikationspartner explizit die Berechtigung für Komunikationsanfragen. Dadurch kann eine Art Budgetierung von Anfragen gestaltet werden. Die weniger privilegierte Applikation kann sich diesem Konzept nicht entziehen, da es ein Designelement des Hypervisor-Microkernels darstellt. Die daraus resultierenden Kommunikationsmuster sind in der folgenden Abbildung schematisch dargestellt und nachfolgend näher beschrieben.

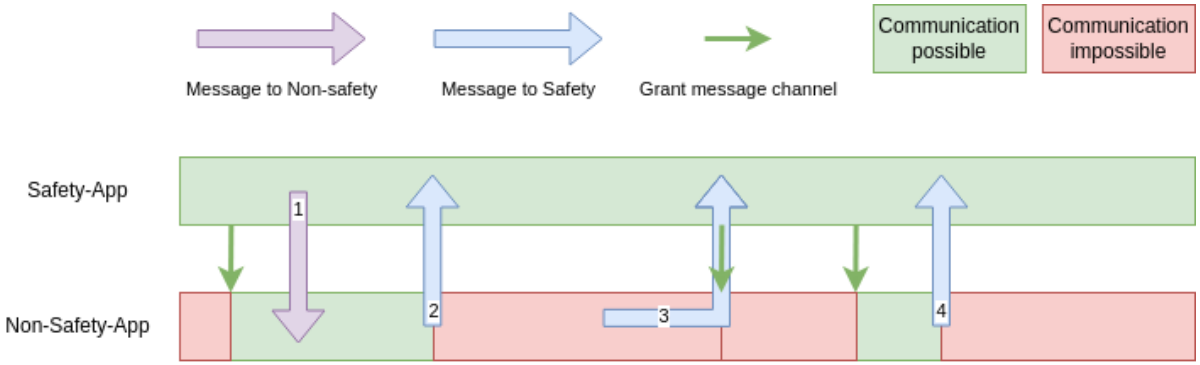

Abbildung 4: Kommunikation zwischen Applikationen unterschiedlicher Kritikalität

Abbildung 4 beschreibt die folgenden vier Szenarien:

1. Die Safety-App kann die Non-Safety-App stets erreichen, da sie höher vertrauenswürdig ist. Dies ist Teil des hierarchischen Designs.
2. Das Gewähren der Kommunikation ist unabhängig von der Nachricht in 1. Die Non-Safety-App ist noch immer in der Lage, die Safety-App zu erreichen, da die erteilte Anfrageberechtigung (grüner Pfeil in der Abbildung) noch nicht in Anspruch genommen bzw. konsumiert wurde.
3. Die Non-Safety-App möchte eine Nachricht an die Safety-App senden, obwohl die Safety-App dies noch nicht erlaubt hat. In diesem Kontext wird das Versenden in eine Warteschlange gelegt. Sobald die Safety-App eine neue Nachricht zulässt, antwortet die Non-Safety-App schnellstmöglich. In diesem Falle tritt eine nennenswerte Latenz auf.
4. Identisch zu 1, allerdings ohne eine Kommunikation der Safety-App während eine erteilte Anfrageberechtigung besteht.

Durch diesen Ansatz werden Latenzfreiheit und minimale Grundlast kombiniert. In Anbetracht von "Microservice Strukturen" ist es nicht ungewöhnlich, dass hunderte solcher Anwendungen auf einer CPU laufen. Eine minimale Last in der Wartezeit ist daher essenziell für die Skalierung eines Systems. Da ein geeignetes Betriebssystem die nötige Funktionalität selbst abbildet, muss die empfangende Applikation nicht gefragt werden, ob sie Nachrichten zulässt oder nicht. Das Absenden der Nachricht führt schon innerhalb des Kontext des Senders zu einem Fehler.

Die Arbeiten zum beschriebenen Kommunikationsprotokoll zwischen Applikationen unterschiedlicher Kritikalität sind nicht in den zum Projekt-Abschluss gezeigten Demonstrator eingeflossen, waren aber Gegenstand einer separaten, dedizierten Prototyp-Implementierung. Bei Bedarf kann der Ansatz aufgegriffen und weiterverfolgt werden, um auch sicherheitskritische Anwendungen prinzipiell als Server-seitige Kommunikationpartner unter Berücksichtigung des Aspekts zeitlicher Rückwirkungsfreiheit zu erlauben.

II.2 Wichtigste Positionen des zahlenmäßigen Nachweises

Die wesentlichen Positionen des zahlenmäßigen Nachweises sind in der folgenden Tabelle dargestellt:

Position	Erläuterung
0837 Personalkosten	Personalkosten für die Projektbeteiligung von Elektrobit SW-Ingenieuren

Tabelle 1 - Wesentliche Positionen des zahlenmäßigen Nachweises

II.3 Voraussichtlicher Nutzen / Verwertbarkeit der Ergebnisse

Es ist geplant, das im Rahmen des Projekts erarbeitete Sicherheitskonzept in Verbindung mit dem Einsatz eines Linux Betriebssystems, für einen späteren Serieneinsatz weiterzuentwickeln und als Grundlage für Projekte mit internationalen Kunden im Automobilbereich einzusetzen.

Das Konzept kann hier voraussichtlich als direkte Basis für eine konkrete ASIL-Zertifizierung von auf EB Softwareprodukten aufbauenden Projektlösungen eingesetzt werden.

III. Literaturverzeichnis

[1] MANNHEIM EMDRIVE Projekt-Webseite,
https://elektronikforschung.de/projekte/mannheim-emdrive

[2] Elektrobit Automotive GmbH, Produkt-Webseite für EB corbos Linux for Safety Applications,
https://www.elektrobit.com/products/ecu/eb-corbos/linux-for-safety-applications/