Technische Hochschule Ingolstadt
FORTEC
Forschungsförderung, -transfer und -controlling

Schlussbericht zum Projekt

Exploratives Projekt 4 (EP): Cloudbasierte Softwarearchitektur für hochautomatisierte Fahrzeuge (DRAFT)

im Rahmen der FH-Impuls-Partnerschaft „SAFIR: Safety for all – Innovative Research Partnership on Global Vehicle and Road Safety Systems“ der Technischen Hochschule Ingolstadt

Teil I: Kurzbericht

Zuwendungsempfänger: Technische Hochschule Ingolstadt
Forschungsförderung, -transfer und –controlling (FORTEC)
Esplanade 10
85049 Ingolstadt

Projektpartner: -

Förderkennzeichen: 13FH7E04IA

Projektleitung: Prof. Dr.-Ing. Werner Huber
☎ +49 (0) 841 / 9348-2523
✉ werner.huber@thi.de

Projektlaufzeit: 01.06.2023 bis 31.07.2024

Berichtszeitraum: 01.06.2023 bis 31.07.2024

I. Kurzbericht zum Projekt DRAFT

Hochautomatisierte Fahrzeugfunktionen, mehr Entertainment, starke Vernetzung innerhalb der Fahrzeuginfrastruktur, kürzer werdende Entwicklungszyklen, over-the-air Updatefähigkeit und viele andere Faktoren stellen Fahrzeughersteller zukünftig, vor allem im Bereich der Softwareentwicklung, vor enorme Herausforderungen. Ein Kerngedanke zur Eindämmung der Komplexität ist die Entkopplung von Software(SW)-Funktionalität in eine Software-Schicht (z.B. Betriebssystem, Cloud-Schnittstelle), sowie einer unabhängigen Hardware(HW)-Schicht. Dieser Kerngedanke wird in Verbindung mit einer Architekturdefinition auch als „Software-Defined Vehicle“ (SDV) verstanden.

Ziel von DRAFT war es, eine neuartige SDV-Fahrzeugarchitektur zu entwerfen, die Impulse aus unterschiedlichen Bereichen (z. B. IoT, Softwareentwicklung) zusammen mit aktuellen Forschungsansätzen aus dem Automobilsektor zu einer voll integrierten Lösung für die Forschung im Bereich des sicheren, hochautomatisierten Fahrens zusammenführt. Kapselung, Abstrahierung, Lastverteilung durch Orchestrierung, Continous Integration sowie Continous Deployment als Technologieimpulse wurden in eine modulare cloudbasierte Architekturdefinition übernommen.

DRAFT setzt auf das erfolgreich beendete SAFIR Projekt EP 1 – AutoBit (Förderkennzeichen: 13FH7I3IA) auf. Der dort aufgebaute Versuchsträger dient hierbei als Entwicklungsumgebung über den Projektzeitraum für eine solche cloudbasierte Softwarearchitektur.

Im Zuge dieses Projekts wurden zunächst sorgfältige Recherchen durchgeführt, um eine priorisierte Anforderungsliste (mit Impulsen aus Automobil-fremden Bereichen) für eine cloudbasierte Softwarearchitektur zu erstellen. Verschiedene literarische und thematische Ressourcen wurden untersucht. Hierbei wurden sowohl Closed-Source-Lösungen als auch Open-Source-Plattformen in Betracht gezogen, wobei die Priorität auf letztere gelegt wurde. Die gewählten Plattformen wurden jeweils tieferlegend untersucht und ihre Übertragbarkeit auf den Versuchsträger geprüft. Überbegriffe der Anforderungsliste sind „Modularität / Integrationsfreiheit“, „SDV-Ansätze“, „Cloud-Nutzung“ und „Tests“.

Auf Basis dessen wurde eine High-Level-Softwarearchitektur definiert. Diese Architektur wurde so ausgelegt, dass sie in der Lage ist, verschiedene Integrationsebenen sowie verschiedene Realitäten (z.B. virtuelle Entitäten) zu berücksichtigen. Insgesamt wurden zwei Versionen erstellt: Eine kompakte Variante, die im Projekt weiter betrachtet wurde und eine allgemeingültige Variante, die vollumfänglich die Anforderungen der zuvor erstellten Anforderungsliste abbildet.

Anschließend wurden Softwarekomponenten aus den recherchierten Fahrzeugarchitekturen ausgewählt. Der Fokus lag dabei vor allem auf Open-Source-Plattformen. Die Referenzarchitektur SOAFEE erfüllte dabei die meisten der gestellten Anforderungen und wurde somit für die anschließende Implementierungsphase ausgewählt. SOAFEE bietet eine breite Unterstützung von Industriepartnern mit einer aktuellen Softwarekette. SOAFEE konnte umfassend durch eine anpassbare Konfigurationskette (weiter) an die Bedürfnisse von DRAFT angepasst werden, welche nach Ausführung eine komplett integrative Lösung bereitstellt. Herausforderungen (u.a.

Anpassung der Ziel-Architektur und limitierte Dokumentation) wurden bewertet und abgewogen.

In der Implementierungsphase wurde EWAOL, eine Referenzimplementierung von SOAFEE, auf die spezifischen Anforderungen des Projekts zugeschnitten. Meilensteine der Implementierung war 1) der Wechsel der EWAOL Ziel-Architektur von ARM zu AMD 2) das Einbringen von NVIDIA-Treibern inkl. CUDA-Unterstützung 3) die Installation zusätzlicher Software, die seitens Forschungsträger benötigt wird, sowie 4) die Integration der EWAOL-Testumgebung. Zur Integration wurden zwei Wege verfolgt: 1) eine Installation Im Forschungsfahrzeug und 2) eine virtuelle Testumgebung, welche die Fahrzeughardware emuliert.

Der letzte Schritt im Projekt war eine Testphase und eine darauf basierende Problemanalyse der neuartigen Softwarearchitektur. Durch die Verwendung von Testfällen, die EWAOL bereitstellte, sowie von im Bauprozess integrierten Prüfungsroutinen, wurde das System ausführlich getestet. Alle gelieferten Testfälle wurden erfolgreich durchgeführt. Die Testergebnisses, die gesamte Konfigurationskette, der Bauprozess der Softwarekette, die Integration (auf Forschungsfahrzeug und virtuell via QEMU) und dessen Inbetriebnahme wurde dokumentiert und mittels git zur Verfügung gestellt.

Zusammenfassend wurden im Projekt alle gesteckten Ziele erreicht: Die Entwicklung einer anforderungsgerechten High-Level Softwarearchitektur (Meilenstein 1), deren Einbettung in den Versuchsträger (Meilenstein 2) und die erfolgreiche Durchführung einer Testphase (Meilenstein 3). Eine hardwareunabhängige Inbetriebnahme von standardisierten Software-Komponenten (Workloads) und Schnittstellen konnte aufgezeigt werden. Eine schnelle, einfache, robuste und sichere Datenerhebung für den Forschungsbetrieb wird durch DRAFT gewährleistet.

Technische Hochschule Ingolstadt
FORTEC
Forschungsförderung, -transfer und -controlling

Schlussbericht zum Projekt

Exploratives Projekt 4 (EP): Cloudbasierte Softwarearchitektur für hochautomatisierte Fahrzeuge (DRAFT)

im Rahmen der FH-Impuls-Partnerschaft „SAFIR: Safety for all – Innovative Research Partnership on Global Vehicle and Road Safety Systems“ der Technischen Hochschule Ingolstadt

Teil II: Eingehende Darstellung

Zuwendungsempfänger: Technische Hochschule Ingolstadt
Forschungsförderung, -transfer und –controlling (FORTEC)
Esplanade 10
85049 Ingolstadt

Projektpartner: -

Förderkennzeichen: 13FH7E04IA

Projektleitung: Prof. Dr.-Ing. Werner Huber
☎ +49 (0) 841 / 9348-2523
✉ werner.huber@thi.de

Projektlaufzeit: 01.06.2023 bis 31.07.2024

Berichtszeitraum: 01.06.2023 bis 31.07.2024

Inhalt

I. Kurzzusammenfassung 4

II. Erzielte wissenschaftlich-technische Ergebnisse und andere wesentliche Ereignisse 4

***1. AP 1: Anforderungserhebung und Architekturrecherche*.....4**

***2. AP 2: Definition einer High-Level SW-Architektur für zukünftige Fahrsysteme* 9**

***3. AP 3: Auswahl und Analyse von einzusetzenden Softwarekomponenten (z.B. aus IoT)*10**

***4. AP 4: Implementierung und Integration einer neuartigen SW-Architektur für zukünftige Fahrsysteme*.....11**

***5. AP 5: Performancetest und Problemanalyse der neuartigen SW-Architektur* .12**

III. Ergebnisse von dritter Seite, die für die Durchführung des Vorhabens relevant sind 14

IV. Fortschreibung des Verwertungsplans.....14

1. Literaturverzeichnis 14

B. Abbildungen

Abbildung 1: High-Level SW-Architektur für den Forschungsträger. Die blaue Farbe kennzeichnet die Container Funktionalität und Orchestrierung, die fliederfarbenen Bereiche sind containerisierte Softwarekomponenten. 9

I. Kurzzusammenfassung

Hochautomatisierte Fahrzeugfunktionen, mehr Entertainment, starke Vernetzung innerhalb der Fahrzeuginfrastruktur, kürzer werdende Entwicklungszyklen, over-the-air Updatefähigkeit und viele andere Faktoren stellen Fahrzeughersteller zukünftig, vor allem im Bereich der Softwareentwicklung, vor enorme Herausforderungen. Ein Kerngedanke zur Eindämmung der Komplexität ist die Entkopplung von Software(SW)-Funktionalität in eine Software-Schicht (z.B. Betriebssystem, Cloud-Schnittstelle), sowie einer unabhängigen Hardware(HW)-Schicht. Dieser Kerngedanke wird in Verbindung mit einer Architekturdefinition auch als „Software-Defined Vehicle“ (SDV) verstanden.

Ziel von DRAFT war es, eine neuartige SDV-Fahrzeugarchitektur zu entwerfen, die Impulse aus unterschiedlichen Bereichen (z. B. IoT, Softwareentwicklung) zusammen mit aktuellen Forschungsansätzen aus dem Automobilsektor zu einer voll integrierten Lösung für die Forschung im Bereich des sicheren, hochautomatisierten Fahrens zusammenführt. Kapselung, Abstrahierung, Lastverteilung durch Orchestrierung, Continous Integration sowie Continous Deployment als Technologieimpulse wurden in eine modulare cloudbasierte Architekturdefinition übernommen.

Eine solche neuartige Fahrzeugarchitektur wurde in DRAFT definiert, implementiert und getestet. Eine hardwareunabhängige Inbetriebnahme von standardisierten Software-Komponenten (Workloads) und Schnittstellen konnte aufgezeigt werden. Exemplarisch wurde eine solche Architektur auf einem Versuchsträger des erfolgreich beendeten SAFIR Projekt EP 1 – AutoBit (FKZ: 13FH7I3IA) in Betrieb genommen. Eine schnelle, einfache, robuste und sichere Datenerhebung für den Forschungsbetrieb wird durch DRAFT gewährleistet.

II. Erzielte wissenschaftlich-technische Ergebnisse und andere wesentliche Ereignisse

1. AP 1: Anforderungserhebung und Architekturrecherche

Im Rahmen des ersten Arbeitspaketes konnte erfolgreich eine priorisierte Anforderungsliste an eine Cloud-basierte Software (SW)-architektur erstellt werden. Der Stand der Technik von Softwarearchitekturen wurde erarbeitet und eine Branchen- und technologieübergreifende Recherche zu neuartigen Ansätzen / „best Practices“ aus Automobil-fremden Brachen wurde durchgeführt. Eine Literaturrecherche in den Datenbanken „ACM Digital Library“ und „IEEE Xplore Digital Library“ sowie dem Bibliothekskatalog der THI wurde durchgeführt.

Im Rahmen der Literaturrecherche wurden folgende Veröffentlichung ausführlich betrachtet und als Basis der weiteren Arbeitspakete identifiziert - siehe Tabelle 1.

Tabelle 1: Im Detail betrachtete relevante Literatur für DRAFT

Titel	DOI
Architecture platforms for future vehicles: a comparison of ROS2 and Adaptive AUTOSAR	10.1109/ITSC55140.2022.9921894

JUPITER – ROS based Vehicle Platform for Autonomous Driving Research	10.1109/ROSE56499.2022.9977434
Bridging the Gap between Open Source Software and Vehicle Hardware for Autonomous Driving	10.1109/IVS.2019.8813784
How to build and validate a safe and reliable Autonomous Driving stack? A ROS based software modular architecture baseline	10.1109/IV51971.2022.9827271
A Multi-Domain Software Architecture for Safe and Secure Autonomous Driving	10.1109/RTCSA52859.2021.00017
A Standard Driven Software Architecture for Fully Autonomous Vehicles	10.1109/ICSA-C.2018.00040
Concept Paper on an open European software-defined vehicle platform	https://digital-strategy.ec.europa.eu/en/library/concept-paper-open-european-software-defined-vehicle-platform

Im Rahmen der Literatur- und Themenrecherche konnten eine Vielzahl von kommerziellen Lösungen gesichtet werden. Eine gute Übersicht hierzu bietet [1, p. Abbildung 1], wobei deutlich wurde, dass Closed Source Lösungen nur schwer im Detail analysiert werden konnten. Da der Schwerpunkt auf einer offenen Plattform für die Forschung lag, wurden diese daher weitestgehend ausgeklammert und die Suche auf Open-Source-Lösungen beschränkt. Kommerzielle Lösungen (z.B. „APTIV Smart Vehicle Architecture“ oder „Nvidia Hyperion“) wurden wegen Verbindungen zu angrenzenden Forschungsprojekten kurz betrachtet, für die Machbarkeitsprüfung jedoch, aus oben aufgelisteten Gründen, ausgespart.

Folgend wurden Open Source SW-Plattformen identifiziert und gesichtet, die als Stand der Technik angesehen werden können, siehe Tabelle 2.

Tabelle 2: Open-Source SW-Plattformen

Name	Kurzbeschreibung	Bewertung
Baidu Apollo	Apollo ist eine offene, modulare Softwareplattform für autonome Fahrzeuge, entwickelt von Baidu (OEM). Enthält umfassend Funktionen um Sense-Plan-Act	**Alleinstellungsmerkmal:** - Umfassende Ende-zu-Ende-Lösung (eigene Implementierung) - Industrienahe Entwicklung (durch Baidu) **Pro:** - Vollständige Dokumentation - Aktive Entwicklung (jährliche Major-Releases)

<table>
<tr><td></td><td>(Wahrnehmung, Planung, Steuerung) abzubilden.</td><td>Kontra / Einschränkung:
- Starke Abhängigkeit von Baidu bzgl. eingesetzter Hardware, Kleineres Ökosystem verglichen mit Autoware
- Ende-zu-Ende-Lösung von Baidu (OEM). Ein offenes SDV-Ökosystem wird weniger angestrebt.</td></tr>
<tr><td rowspan="3">Autoware</td><td rowspan="3">Autoware ist eine offene, modulare Softwareplattform inkl. Referenzimplementierung. Basiert auf dem Robot Operating System (ROS) 2 und war das Erste Framework, das Sense-Plan-Act (Perception, Prediction, Planning und Control) abbildete.</td><td>Alleinstellungsmerkmal:
- Unterteilung zwischen Core Modulen der Autoware Foundation und Universe Modulen der Autoware Community
- Ende-zu-Ende Lösung basierend auf ROS (2)</td></tr>
<tr><td>Pro:
- Breiter Einsatz in Industrie und Forschung
- Aktive Entwicklung (rolling Releases)
- Kompatibilität mit ROS</td></tr>
<tr><td>Kontra / Einschränkung:
- Teils werden Cloud-Dienste nur geschlossen (Closed-Source) zur Verfügung gestellt.
- Keine Einsicht in Trainingsdaten der Sense-Plan-Act-Anwendung</td></tr>
<tr><td rowspan="2">Open AD Kit</td><td rowspan="2">Erweitert das Autoware-Ökosystem um einen Software Definded Vehicle (SDV) Anteil. Ziel ist es, Autoware von Hard- und Software zu entkoppeln.

Das „Grundgerüst“ dazu stellt SOAFEE. Eine</td><td>Alleinstellungsmerkmal:
- Erweitert Autoware um den SDV-Gedanken</td></tr>
<tr><td>Pro:
- Breite Industrieunterstützung (in SOAFEE)
- Referenzimplementierung vorhanden und öffentlich verfügbar
- Kompatibilität mit ROS</td></tr>
</table>

	Architekturbeschreibung die, mittels Standardisierung, Fahrzeugarchitekturen einschließlich Impulsen aus der Softwareentwicklung, Industriezusammenarbeit und Simulationsanteilen SDVs zu ermöglichen. Autoware wird im Fall von Open-AD-Kit als „Workload“ oder „Domain-Specific Application“ (Container) eingesetzt, welche hardwareunabhängig (edge, cloud) ausgeführt wird. Im Rahmen von DRAFT wurde die Referenzimplementierung ***EWAOL*** für SOAFEE betrachtet.	**Kontra / Einschränkung:** - Referenzimplementierung EWAOL verwendet ARM-Referenzarchitektur, welche inkompatibel mit angedachtem Versuchsträger ist. - Dokumentation bildet nicht vollumfänglich den Umfang ab. - Implementierung hängt Autoware nach (Zum Zeitpunkt des Projektes wurde Open AD Kit V2 noch nicht, V1 im Oktober 2022 veröffentlicht)

Diese Plattformen wurden jeweils tieferlegend untersucht und ihre Übertragbarkeit auf den Versuchsträger (aus SAFIR EP 1: AutoBit, FKZ 13FH7E01IA) geprüft. Die Recherche zeigte: Die Austauschbarkeit von Software-Modulen wird oft nicht berücksichtigt, welche eine in der Forschung wichtige Kooperationsmöglichkeit erschwert (Ziel: Modularität). Weiter ist die Integrationsfreiheit eingeschränkt, da Referenzimplementierungen sich meist auf einzelne (oder wenige) Fahrzeuge und/oder Sensoren ausrichten. Durchgängige Prozessketten müssen abgebildet werden, um einen vollständigen Software Development Lifecycle (SDLC) abzubilden. Cloud-Dienste müssen Open-Source, nahtlos und ortsunabhängig verfügbar sein. Hierzu wurden im Rahmen von DRAFT auch Impulse aus Automobil-fremden Bereichen berücksichtigt. Abschließend sollen vollumfängliche Testfälle integriert sein, die den gültigen Bauprozess sicherstellen, sowie virtuell / real den sicheren Betrieb gewährleisten.

Im Projektzeitraum konnte deshalb eine Anforderungsliste (mit Impulsen aus Automobil-fremden Bereichen) an eine Cloud-basierte SW-Architektur, optimiert für den Einsatz in der Forschung, abgeleitet werden (siehe Tabelle 3).

Tabelle 3: Anforderungsliste an eine Cloud-basierte Softwarearchitektur für SDV

Priorität	Überbegriff	Detailanforderung
1	Modularität / Integrations-freiheit	**Anforderung**: Ein möglichst modularer Aufbau, der einen einfachen Austausch von Hardware- und Softwarekomponenten zulässt, ist für die Forschung essenziell. Die Freiheit: - jegliche Hardware flexibel einzusetzen - containerisierte Softwarekomponenten zu unterschützen
		Ziel: Die Abstrahierung von Hard- und Software
		Mittels: - Softwarekette, die Hardwareunabhängig agiert (z.B. Yocto) und Virtualisierung von Hardware (z.B. Hypervisor) bereitstellt. - Unterstützung von Softwareabstraktion mittels Containerisierung.
2	Software Developme nt Lifecycle - Ansätze	**Anforderung**: Eine durchgängige Prozesskette, von der Planung, über die Programmierung bis zum produktiven Einsatz für den Forschungsbetrieb ist durch die Nutzung von gängigen Tools für Anwendungsentwicklung und Versionierung sicherzustellen
		Ziel: Prozesskette, die mit Hilfe von Continous Integration (CI) und Continous Development (CD) einen Software Development Life Cycle (SDLC) abbildet. Nutzung von Impulsen aus der Softwareentwicklung wie Versionierbarkeit, Abbildung von Abhängigkeiten, Rückrollmöglichkeiten, ACID-Prinzip und Stage-Umgebungen.
		Mittels: CI/CD-Pipeline, die eine automatische Ausführung von versionierten Testkampagnen anstrebt und eine (komfortable) Auslieferung der Software bietet.
3	Cloud-Nutzung	**Anforderung**: Eine Möglichkeit, Software (Open-Source), unabhängig des Ortes, auszuführen und zu skalieren.
		Ziel: Verteilte und skalierbare Ausführung der Fahrzeugsoftware, unabhängig vom Standort.
		Mittels: Orchestrierung (der Container)
4	Test(s)	**Anforderung**: Eine vollumfängliche Testroutine muss vorhanden sein, die die gültige Integration und Inbetriebnahme gewährleistet.
		Ziel: Prüfbare Inbetriebnahme virtuell oder im Versuchsträger
		Mittels: - Testfällen die den sicheren Betrieb gewährleisten

		- Prüfroutinen den gültigen Bauprozess sicherstellen

Im weiteren Verlauf des Projektes werden diese Anforderungen an die Architektur des Versuchsträger als Zielvorlage angenommen. Im Rahmen des Projektes DRAFT richtet sich die Aufmerksamkeit auf einen möglichst modularen Aufbau im Sinne einer voll integrativen Lösung mit Forschungsabsicht.

2. AP 2: Definition einer High-Level SW-Architektur für zukünftige Fahrsysteme

Aus den Ergebnissen aus AP 1 (sowie Ergebnisse aus IP1, IP10 und IP6) wurde eine SW-Architektur definiert. Unterschiedliche Integrationsebenen, sowie eingesetzte Realitäten (Integration von simulierten Entitäten) wurden berücksichtigt.

Die für den späteren Einsatz angedachte Architektur ist in Abbildung 1 aufgezeigt. Diese Reflektiert die Architektur, die im Rahmen von DRAFT umgesetzt werden soll. Diese Architektur soll für mögliche anschließende Forschungsprojekte als Referenz dienen und weitere Einsatzmöglichkeiten aufzeigen. Hierzu können die eingeführten Abkürzungsdefinitionen aus diesem Abschnitt analog betrachtet werden.

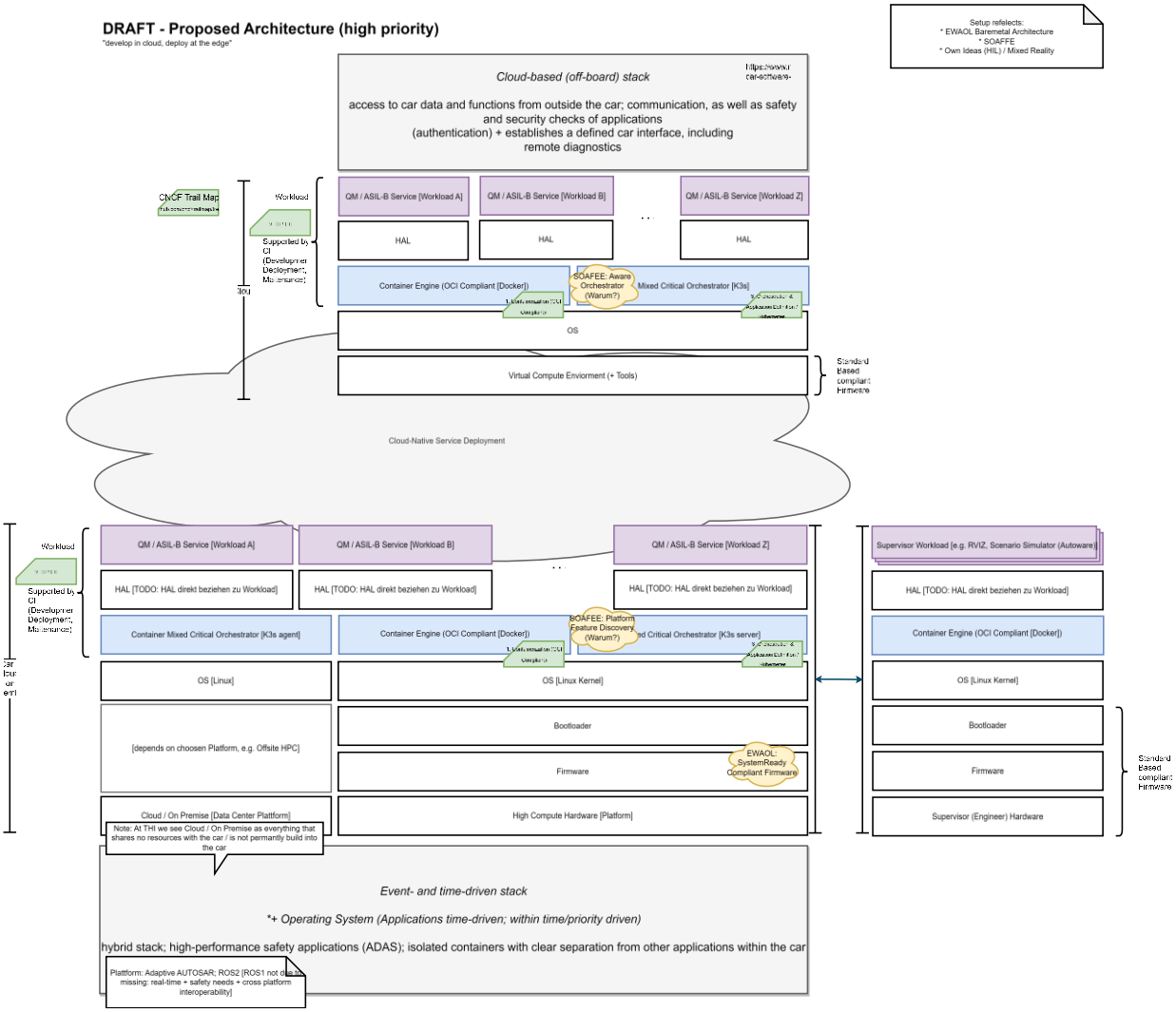

Abbildung 1: High-Level SW-Architektur für den Forschungsträger. Die blaue Farbe kennzeichnet die Container Funktionalität und Orchestrierung, die fliederfarbenen Bereiche sind containerisierte Softwarekomponenten.

Die angedachte Architektur (Abbildung 1) lässt sich grob in drei Abschnitte unterteilen. Einem „Event- and time-driven stack“, welcher die Softwarekomponenten innerhalb des Fahrzeuges bedient. Unter diesem wird ein hybrider Ansatz zur Dringlichkeitsbewertung verwendet. Anwendungen werden isoliert verwaltet und nach deren Kritikalität ausgeführt. Innerhalb von Anwendung können harte Zeitanforderungen, oder auch die „best effort“-Dringlichkeit eingesetzt werden. Hierzu wird, in Bezug zur Anforderungsliste, das Betriebssystem (OS) mittels Container abstrahiert. Diese sollen OCI (Open Container Initiative) gültig sein und entkoppeln die im Container enthaltenen Funktionen von ihrer Umgebung. Diese entkoppelten Container können mittels dem einzusetzenden Orchestrator dynamisch auf unterschiedlichen Rechenressourcen ausgeführt werden. Dieser hat generell die Aufgaben, Containeranwendungen in ihrer Bereitstellung und Skalierung in einem verteilten System zu koordinieren und zu verwalten. Dem Container kann mittels eines Hardware Abstraction Layer (HAL) Zugriffe auf die Hardware zur Verfügung gestellt werden. Der Aufbau des „Event- and time-driven stack“ wird durch eine CI/CD Fähigkeit erweitert, um alle Anforderungsbedingungen zu erfüllen, so dass die unterschiedlichen Wissenschaftler ihre Funktionen dynamisch, sicher und in einem vorgegebenen Prozess auf die Versuchsplattform deployen können. Hinzukommend ist angedacht, dass Arbeitslasten auch vor Ort („on Premise“), aber dennoch außerhalb des Versuchsträgers verteilt werden können. Diese Anforderung wurde aufgenommen, da Rechenleistung respektive Ressourcen wie Strom und Platzvolumen im Versuchsträger begrenzt sind.

Weiter wird ein „Cloud-based (off-board) stack” angedacht, der analog zum vorherigen „Event- and time-driven stack“ agiert. Unterschied ist hierbei, dass der Cloud-Stack unabhängig zum Fahrzeug Arbeitslast bedient. Auch hier wird eine CI/CD-Fähigkeit integriert. Beispielhaft könnten Arbeitslasen ausgelagert werden, welche keine kritische Latzenzzeit benötigen.

Zuletzt wurde ein „Supervisor“-System angedacht, das komplementäre Arbeitsleistungen, für den Forschungsbetrieb, bearbeitet. Hierzu wird vor allem Software verstanden die zum (sicheren) Testbetrieb benötigt wird. Bei der Integration von Open AD Kit umfasst dies beispielsweise ein kontinuierliches Monitoring der Testdaten, eine Szenariensimulation oder das Abführen von Testdaten.

Der Meilenstein 1 („MS 1: Eine High-Level SW-Architektur für zukünftige Fahrsysteme ist erstellt.“) wurde erfolgreich abgeschlossen.

3. AP 3: Auswahl und Analyse von einzusetzenden Softwarekomponenten (z.B. aus IoT)

Die Open Source Software Plattformen aus AP 1 wurden im Rahmen dieses Arbeitspakets weiter untersucht und Softwarekomponenten ausgewählt. Ziel war es, vor der Implementierung (in AP 4) eine möglichst umfangreiche Sichtung von Softwarekomponenten zu erzielen.

Ausgehend von den erarbeiteten Anforderungen aus Tabelle 3: Anforderungsliste an eine Cloud-basierte Softwarearchitektur für SDV wurden Teilkomponenten aus den

gesichteten SW-Plattformen (siehe Tabelle 2) gewählt. Ein „kompletter Neustart“ ist im Rahmen von DRAFT nicht durchführbar. Die Entscheidung fiel deshalb auf das Open AD Kit, welches Autoware um SDV-Anteile erweitert. Technisch wurde vor allem die Referenzarchitektur SOAFEE analysiert. SOAFEE bietet eine breite Unterstützung von Industriepartnern mit einer aktuellen Softwarekette. [2] Die Architektur bietet eine sehr gute Grundlage, um einen modernen Forschungsbetrieb zu gewährleisten. Um die Kontra Punkte der Tabelle 2 zu negieren, wurden die dazugehörigen Herausforderungen (siehe Tabelle 4) identifiziert und Forschungsträger spezifische Anforderungen aufgenommen. Einschränkungen bezüglich der Anwendung selbst („Workload“) wurden als unkritisch identifiziert, da im Forschungsbetrieb eigene Anwendungen (z.B. eine Perception-Algorithmen) eingesetzt werden.

Tabelle 4: Herausforderungen der gewählten Softwarekette

Beschreibung	Herausforderung
Implementierung / Integration von SOAFEE (mittels EWAOL-Referenzimplementierung)	Die bestehenden Referenzimplementierungen basieren auf ARM-Architektur (N1SDP und AVA Plattformen) und müssen auf eine AMD-Architektur angepasst werden. Der Forschungsträgeraufbau benötigt und profitiert von einer gültigen NVIDA-Treiber Installation, um u.a. auf Hardwarebeschleunigung zugreifen zu können.
Demo oder Use-Case Präsentation von SOAFEE (mittels Open AD Kit, Autoware)	Es existierte nur ein Limitierte Dokumentation innerhalb von Open AD Kit. Ein Abgleich der Dokumentationen von Open AD Kit V1 und V2, SOAFEE und EWAOL war nötig.

SOAFEE hat trotz dieser Herausforderungen dennoch ein großes Potential. In EWAOL sind umfangreiche Testfälle und ein vollständiges „Software Development Kit“ (SDK) implementiert, die eine Inbetriebnahme vereinfachen. Weiter werden die Anforderungen (siehe Tabelle 3) bedient. Vor allem kann die Modularität, die Integration interner Software und Testkampagnen, der SDV-Ansatz sowie der Cloud-Betrieb übernommen werden.

Die Softwareintegration wird mittels Docker[1] bereitgestellt. Der Cloud-Ansatz (hardware- und ortsunabhängiger Betrieb) mittels K3s[2]. Die Integration einer CI/CD Pipeline basiert auf GitLab[3].

4. AP 4: Implementierung und Integration einer neuartigen SW-Architektur für zukünftige Fahrsysteme

In diesem Arbeitspaket wurde, aufbauend auf allen vorangegangenen Arbeitspaketen, die Implementierung und Integration der definierten SW-Architektur vorgenommen. Ziel war es eine voll integrative Lösung zu implementieren.

[1] https://www.docker.com/ (freie Software zur Isolierung von Anwendungen mit Hilfe von Containern)
[2] https://k3s.io/ (leichtgewichtige Orchestrierungs-Software basierend auf Kubernetes)
[3] https://gitlab.com (Onlinedienst zur Softwareentwicklung und Versionsverwaltung für Softwareprojekte)

Wie in auf AP3 beschrieben, musste EWAOL an die Anforderungen von DRAFT angepasst werden. EWAOL unterscheidet für die Integration zwei Ansätze: Baremetal und Virtualization. Virtualization erweitert die Baremetal-Architetkur um die Option, Anwendungen in gekapselten virtuellen Maschinen zu starten. Diese Anforderung wird im Forschungsbetrieb nicht benötigt (eine Hardwareplattforum), weshalb im Weiteren der Fokus auf die Baremetal-Architektur gelegt wurde.

EWAOL bietet eine umfassend anpassbare Konfigurationskette, die nach Ausführung eine komplett integrative Lösung bereitstellt. Diese lässt sich sofort in den Versuchsträger integrieren oder virtuell (mittels QEMU) ausführen und testen. Im Rahmen von DRAFT wurden die nachfolgenden Anpassungen durchgeführt.

Wechsel des EWAOL ARM-Maschinentyp zur AMD-(x64 Architektur)
Hierzu wurde die Konfigurationskette so gengepasst, dass anstatt ARM-spezifischer Abstraktionstreiber passende AMD-Alternativen eingeladen werden. Mitunter mussten hierzu eine eigene Kernel-Konfiguration eingepasst werden, um die Kompatibilität mit dem Forschungsträger zu gewährleisten. EWAOL spezifische Einstellungen wurden so verändert, dass ein automatischer, gültiger und nachvollziehbarer Bauprozess für eine AMD-Architektur sichergestellt ist.

Einbringen von NVIDIA-Treibern inkl. CUDA-Unterstützung (inkl. Container)
Um spezifische Hardware im Forschungsträger anzusprechen, wurde die Konfigurationskette um NVIDIA-Treiber erweitert. Neben der Sicherstellung der Kompatibilität mit EWAOL befanden sich auch Abhängigkeiten zum Betriebssystem (Linux), der NVIDIA-Version und dem Bauprozess selbst. Es konnte a) die Unterstützung von NVIDIA-Grafikarten, b) CUDA (auf dem Host-Betriebssystem) und c) volle Hardware-beschleunigung innerhalb laufender Container mittels NVIDIA-Container-Toolkit sichergestellt werden.

Installation zusätzlicher Software, die seitens Forschungsträger benötigt wird
Hinzukommend wurde proprietäre Software seitens des Hardwarezulieferers für den Forschungsträger in die Konfigurationskette aufgenommen. Zuletzt wurden geringfügige Anpassungen der Installationsroutine eingebracht (z.B. Tastaturlayout, vim-Editor) um eine Inbetriebnahme zu vereinfachen.

Bereitstellung EWAOL Testumgebung
Zuletzt wurde die Konfigurationskette so angepasst, dass die optionalen EWAOL-Testfälle (Testkampagnen) zur Abnahmeprüfung eingeladen werden. Die vollständige Konfigurationskette wurden intern mittels git-Repository veröffentlicht.

Der Meilenstein (MS 2) „In einem Versuchsträger wurde die in Arbeitspaket 2 definierte High-Level SW-Architektur exemplarisch implementiert und integriert“ wurde erreicht.

5. AP 5: Performancetest und Problemanalyse der neuartigen SW-Architektur

Dieses Arbeitspaket umfasste den Test der in AP4 implementierten und integrierten SW-Architektur.

Die Integration und Inbetriebnahme wurde, wie in den Anforderungen (siehe Tabelle 3) beschrieben, auf zwei Plattformen durchgeführt: 1) eine Installation Im Forschungsfahrzeug und 2) eine virtuelle Testumgebung, welche die Fahrzeughardware

emuliert. Hierzu wurde QEMU[4] verwendet. Der Projekterfolg wurde dementsprechend unabhängig von der Verfügbarkeit des Fahrzeugs während der Projektlaufzeit sichergestellt.

Die Testergebnisses, die gesamte Konfigurationskette, der Bauprozess der Softwarekette, die Integration (auf Forschungsfahrzeug und virtuell via QEMU) und dessen Inbetriebnahme wurde dokumentiert und als git-Repository zur Verfügung gestellt.

Übergeordnet und besonders ausschlaggebend zur Problemanalyse war die in EWAOL integrierte Testumgebung (siehe AP 4) und dessen Testfälle. Es konnten zum Abschluss des Projektes alle gelieferten Testfälle erfolgreich durchgeführt werden (siehe Tabelle 5: Durchgeführte Testfälle). Weiter wurden die im Bauprozess integrierten Prüfungsroutinen von EWAOL bestanden.

Tabelle 5: Durchgeführte Testfälle

Name	Kurzbeschreibung	Link	Status
Validate – EWAOL	Stellt sicher, dass alle für EWAOL nötigen Services (u.a. docker und k3s) laufen	link	`OK`

Run-Time Integration Tests	Diese validieren die EWAOL-Kernfunktionalitäten zur Laufzeit.
	
	Container-Engine-Tests: Sie stellen sicher, dass die Container ordnungsgemäß in Betrieb genommen werden können und in der Lage sind, miteinander zu kommunizieren.
	
	K3s-Orchestration-Tests: Validiert das deployment und Verfügbarkeit (Server Failure Tolerance) einer Test-Workload.
	
	User-Accounts-Test: Prüft, ob die für EWAOL notwenigen Nutzer korrekt konfiguriert und Zugriffsberechtigungen vorhanden sind.
		link1, link2, link3	`PASS:container-engine-integration-tests`
			`PASS: k3s-integration-tests`
			`PASS:user-accounts-integration-tests`

k3s check-config	Zur Prüfung der k3s-Einstellungen und Abgleich des Featureumfangs mit EWAOL	K3s Dokumentation	`STATUS: pass`
Reproducing the EWAOL Use-Cases	Testlauf vereinfachter Kernfunktionalitäten von EWAOL. Workloads werden	link	`Docker OK, k3s OK`

[4] https://www.qemu.org/ (Open Source Virtualisierungssoftware)

	mittels Docker und k3s deployed.		

Zuletzt wurde mithilfe von “Open AD Kit“ (siehe Tabelle 2: Open-Source SW-Plattformen) die Mächtigkeit des Ansatzes gezeigt, indem Bestandteile von Autoware innerhalb der neuartigen SW-Architektur gestartet wurden. Dieser Test fand wegen Ressourcenkonflikten in einer virtuellen Testumgebung statt.

Der Meilenstein (MS 3) „Die High-Level SW-Architektur ist getestet und Probleme identifiziert“ wurde erreicht.

III. Ergebnisse von dritter Seite, die für die Durchführung des Vorhabens relevant sind

Entwicklungen der gewählten Referenzimplementierungen wurden kontinuierlich verfolgt. Darunter werden auch die eingesetzten und davon abhängigen Komponenten (z.B. Yocto) verstanden. Einschlägige Datenbankrecherchen (z.B. IEEE, ACM) wurden kontinuierlich durchgeführt. Es sind keine Publikationen bekannt, die die Durchführung des Projektes beeinflussten.

IV. Fortschreibung des Verwertungsplans

Das Projekt hat Potenzial für Folgeprojekte, darunter das EU-Projekt CARISMA im Rahmen der Horizon Linie D6-01-01, das auf den Ergebnissen von DRAFT aufbaut. Ein zentrales Thema war die Integration von Off-Site Hardwarekapazitäten (im Fahrzeugaufbau von DRAFT) für rechenintensive Forschung. Dazu wird die Fahrzeugarchitektur um Edge-Computing erweitert. Ein weiteres Forschungsvorhaben betrifft das High-Definition-Testfeld (HDT) Ingolstadt, das Road-Side-Units (Off-Site Hardwarekapazitäten) in die Fahrzeuginfrastruktur integriert.

Weiter ist eine Veröffentlichung der Projektergebnisse geplant. Grundlage sind die in Tabelle 1 identifizierte Literatur. Ziel ist es, die Mächtigkeit der neuartigen SW-Architektur aufzuzeigen und damit Forschung voranzutreiben. Dies soll in Zusammenarbeit mit thematisch naheliegenden Projekten (u. A. SAFIR IP10: Lokalisierung und Steuerung in kombinierten Indoor- und Outdoor-Szenarien (KASINO): FKZ 13FH7I10IA) stattfinden, um die gesamte Anwendungskette darzustellen.

E. Literaturverzeichnis

1. Literaturverzeichnis

[1] E. Juliussen, „embedded.com,“ 13 04 2021. [Online]. Available: https://www.embedded.com/av-software-platform-providers-tally-design-wins/.

[2] „soafee,“ 17 04 2024. [Online]. Available: https://www.soafee.io/about/members.