KI4BoardNet

Bundesministerium
für Forschung, Technologie
und Raumfahrt

BMBF-Verbundprojekt:	MANNHEIM-KI4BoardNet
Förderkennzeichen:	16ME1106
Projektlaufzeit:	01.09.2025 bis 30.11.2025

Schlussbericht des Teilvorhabens

Erforschung und Entwicklung eines hochautomatisierten und modularen Automatisierungsbaukastens zur KI-gestützten Absicherung des Bordnetzes

im Rahmen des Verbundvorhabens

Integrale agile E/E-Entwicklung für fusionierte und standardisierte Energie- und Datenbordnetze

Version:	0.05
Erstelldatum:	08.07.2025

Autoren: Jannica Langner (ACONEXT Sparks GmbH)
Waldemar Zimbelmann (ACONEXT Sparks GmbH)
Alex Stark (ACONEXT Sparks GmbH)

Zuwendungsempfänger: ACONEXT Sparks GmbH (Fkz: 16ME1106)

Ansprechpartner: ACONEXT Sparks GmbH
Steffen Matt
Alemannenstraße 23, 71296 Heimsheim
steffen.matt@aconext.de

Das diesem Bericht zugrundeliegende Vorhaben wurde mit Mitteln des Bundesministeriums für Forschung, Technologie und Raumfahrt unter dem Förderkennzeichen 16ME1106 gefördert. Die Verantwortung für den Inhalt dieser Veröffentlichung liegt bei den Autoren.

Inhalt

1 Zusammenfassung

Die steigende Komplexität des Bordnetzes und das Aufkommen zukunftsweisender Technologien wie Connectivity, ADAS und autonomes Fahren machen es für Fahrzeughersteller zunehmend schwieriger, die meist zu jeder Generation neu entwickelte Automotive-Software sowie Komponenten und Gesamtsysteme abzusichern. Zudem verkürzt sich durch die schnelle technische Entwicklung der Produktzyklus eines Fahrzeugs. Dies hat zur Folge, dass sowohl der Umfang als auch die Komplexität der abzusichernden Komponenten im Bordnetz stark ansteigen. Testautomatisierungen wurden bereits früher zur Absicherung der Systeme auf Bus- und Schnittstellenebene eingesetzt. Die meisten Schritte im Testprozess wurden jedoch nach wie vor von menschlichem Testpersonal übernommen. Es waren daher Lösungen gefragt, die die tendenziell gleichbleibende Anzahl an Testern bei der Bewältigung der steigenden Menge an Testfällen unterstützen konnten.

Auch die Testfallerstellung, -dokumentation und das Reporting basierten zum Großteil auf manuellen Prozessen. Ziel des Teilvorhabens von ASPA war es daher, den gesamten Testprozess mithilfe KI-gestützter Tools so weit wie möglich zu automatisieren. Zu diesem Zweck wurde ein modularer Automatisierungsbaukasten (MAB) erforscht, der es ermöglicht, verschiedene Module unabhängig voneinander einzusetzen. Dies erlaubte eine optimale Verwertung der Ergebnisse des Teilbeitrags und einen flexiblen Einsatz in verschiedenen Dienstleistungen (z.B. Testgewerken) und Projekten.

Im Gesamtverbundprojekt ist das ASPA-Teilvorhaben essenziell für die Erforschung einer automatisierten Testumgebung. Das Teilvorhaben umfasste auch die Anforderungsanalyse, Datenerzeugung und Validierung der Testumgebung.

2 Ziele

2.1 Problemstellung und allgemeine Ziele des Gesamtvorhabens

Zum Projektbeginn sah sich die Automobilindustrie weltweit einem disruptiven Paradigmenwechsel ausgesetzt, da neue Mobilitäts- und Nutzungskonzepte die Anforderungen an das Automobil grundlegend veränderten: Das automatisierte und vernetzte Fahren stellte immer höhere technische Anforderungen an das Bordnetz. In seiner Doppelfunktion diente es sowohl der Energieversorgung als auch dem Informationsfluss zwischen Sensoren und Aktoren sowie der Kommunikation nach außen. Klassische Kabelbäume und der Ansatz, für jedes Feature ein eigenes Steuergerät zu integrieren, waren an ihre Grenzen gestoßen und wurden bereits durch neue Konzepte ergänzt bzw. ersetzt.

Das Ziel von KI4BoardNet war die Entwicklung eines intelligenten, durch Controller mit hoher Rechenleistung gesteuertes und aus Gründen der funktionalen Sicherheit teilredundantes Zonen- bzw. Zentralkonzept, welches durch die Konzentration vieler Funktionen in leistungsfähigen Energie- und Datenverarbeitungseinheiten eine reduzierte Anzahl von Leitungsverbindungen ermöglicht. Dadurch könnten als Grundlage für die Realisierung höherer Grade automatisierter Fahrfunktionen (Stufen 4 und 5 gemäß VDA) mit einer neuen Architektur gleichzeitig neue Möglichkeiten für eine Sensor- und Datenfusion eröffnen.

Das Bordnetz der Zukunft soll als Rückgrat des „System of Systems“ wirken, indem es die gekapselten, unterschiedlichen Subsysteme intelligent verknüpft. Hohe Datenraten und unterschiedliche Spannungsebenen von 3V über 48V bis 800V stellen dabei neue Herausforderungen an den Entwurf dieser E/E-Bordnetzarchitekturen. Neben der funktionalen Sicherheit ist die Datensicherheit von hoher Bedeutung (keine unautorisierten Eingriffe in das System, Datenübermittlung nur an vertrauenswürdige Ziele, Integrität der Daten). Hierfür müssen Konzepte für die umfangreiche Sensorik, Aktorik und entsprechende intelligente Datenverarbeitung sowie deren Zuverlässigkeit neu gedacht und weiterentwickelt werden.

Das Bordnetz als eigene intelligente E/E-Architektur muss flexibel und situationsgerecht auf Störungen und Ausfälle reagieren können. Dies stellt komplett neue Anforderungen an den E/E-Entwurf des Bordnetzes und seiner Komponenten. Zukünftige Architekturen müssen zudem funktional möglichst einfach erweiter- und anpassbar sein und die Integration von neuen Elektronikkomponenten je nach Anforderungen der Fahrzeugfamilie und den global unterschiedlichen regulatorischen Vorgaben erlauben. Die Trennung von Energie- und Datennetz ist mit einer zentralisierten Architektur und entsprechend langen, hochpoligen Leitungen nicht zukunftsfähig und bisherige Konzepte erreichen bezüglich Gewicht und Platzbedarf, der Energieversorgung des Systems, seiner Konfigurierbarkeit und insbesondere seiner elektromagnetischen Verträglichkeit (EMV) ihre Grenzen. Höchste Zuverlässigkeit bei gleichzeitig akzeptablen Kosten und Gewicht ist nur durch neue, gut strukturierte Architektur- und Diagnosekonzepte mit optimaler Nutzung der physikalischen Schichten, möglichst einheitlichen, standardisierten Schnittstellen auf Hard- und Software-Ebene sowie skalierbaren Anschlusskomponenten realisierbar. Hierbei muss der fail-operational Betrieb von Beginn an mitgedacht werden, was durch eine intelligente diagnosebasierte Selbstkonfiguration statt durch zusätzliche Redundanz gewährleistet werden soll und gleichbedeutend mit einem radikalen Paradigmenwechsel ist.

In den letzten Jahren ist der Aufwand für die Absicherung von Bordnetz und Steuergeräten im Rahmen der E/E-Entwicklung stark angestiegen. Die Forschungsziele innovativer Fahrzeugrechenplattformen, Elektronikkomponenten und -systeme waren daher unmittelbar mit erhöhten Anforderungen an die Absicherung neuer Entwicklungen verbunden. Das Teilvorhaben von ASPA zielte darauf ab, durch Testautomatisierung die Qualitätsansprüche an Neuentwicklungen zu gewährleisten.

2.2 Einordnung der Teilvorhabenziele in das Gesamtvorhaben und Motivation

Das Teilvorhaben von ASPA befasste sich mit der Automatisierung der Absicherung des Bordnetzes. Ziel war es, zu evaluieren, welche Teilschritte des Absicherungsprozesses sich sinnvoll automatisieren lassen. Im Anschluss sollte die Automatisierungen im Rahmen eines modularen Automatisierungsbaukastens (MAB) KI-basiert erforscht werden. Dabei wurde eine vollautomatisierte End-to-End-Absicherung angestrebt. Die automatisierte Absicherung minimiert das Risiko einer fehlerhaften Integration von Systemkomponenten (z. B. Steuergeräten) in das Bordnetz. Früher wurden die meisten Tests noch durch menschliches

Testpersonal manuell durchgeführt. Die Automatisierung senkt die Kosten für Prüfmittel und Prüflinge sowie die Arbeitszeit des Testpersonals, sodass eine qualitativ bessere Absicherung gewährleistet werden kann. Das Teilvorhaben von ASPA stellt im Gesamtverbund sicher, dass die Absicherung von Bordnetzen automatisiert durchgeführt werden kann. Fehler in einzelnen Komponenten und Netzwerken sollen automatisiert erkannt und gemeldet werden.

Zu Beginn des Projektes wurden fünf UseCases identifiziert. ACONEXT Sparks GmbH beteiligte sich an UseCase#2, #4 und #5.

UseCase#1 – Energie- und Datenbordnetze & Hybride Zonenkonzepte
UseCase#2 – Intelligente Sensoren, Algorithmen und Hardware zur KI-Beschleunigung
UseCase#3 – Intelligente Energiekomponenten
UseCase#4 – Methoden für Simulation, Modellierung & Datengenerierung für Bordnetzkomponenten
UseCase#5 – Agiler modellbasierter Entwurf & Verifikation

Tabelle 1: Verknüpfung der geplanten CARIAD-Projektaktivitäten mit den UseCases

UseCase#2:

Im Rahmen dieses UseCases beschäftigten wir uns mit Robotik, Sensorik, Bildverarbeitung und Audioverarbeitung zur Durchführung von hochautomatisierten Testfällen. Es wurde ein Demonstrator für die KI-gestützte Testautomatisierung mit Roboterarm aufgebaut. Folgende Fragestellungen und Aufgaben ergaben sich konkret:

- Validierung, Demonstration und Bewertung eines Robotiksystems und der Sensorik.

UseCase#4:

In UseCase#4 haben wir uns mit Anomalie Erkennung in Fahrzeugbusdaten eingebracht. Spezialisierte KI-Modelle für die Anomalie Erkennung in CAN- und Ethernet-Busdaten wurden trainiert. Unsere Aufgaben und Fragestellungen sahen dabei wie folgt aus:

- Validierung, Demonstration und Bewertung eines Demonstrators, der einen Fahrzeug-Trace KI-gestützt analysiert und Abweichungen detektiert.

UseCase#5:

In diesem UseCase beschäftigten wir uns mit Natural Language Processing, automatischer Testfallerstellung, Testauswertung und Testberichterzeugung. Wir suchten nach Ansätzen zur Automatisierung der Testfalldurchführung von Testfallerstellung bis zur Bug-Ticket-Erstellung. Folgende Fragestellungen und Aufgaben ergaben sich:

- Validierung, Demonstration und Bewertung eines Demonstrators, der aus Testspezifikationen automatisiert Test-Cases erzeugt, Testergebnisse beurteilt und Testberichte erzeugt.

Über die UseCases hinweg erfolgte eine enge Zusammenarbeit zwischen UseCase#2 und UseCase#5, um einen gemeinsamen Modularen Automatisierungs-Baukasten (MAB) zu erstellen, der den gesamten Testprozess abdeckt. Außerdem war ein intensiver Austausch in enger Zusammenarbeit mit den jeweiligen UseCase-Partnern elementar. Ein stetiger

Austausch und Aufbau auf den jeweiligen Kernkompetenzen der Partner führt zu einer Kompetenzsteigerung innerhalb des gesamten UseCases und Verbundes.

2.3 Wissenschaftliche und/oder technische Ziele des Teilvorhabens

ASPA setzte sich in den Forschungskernbereichen des Teilvorhabens die folgenden technischen Arbeitsziele:

- Validierung aller automatisierten Teilaspekte des Modularen Automatisierungs-Baukastens (MAB): Die Validierung sollte mithilfe eines Demonstrators durchgeführt werden. Die Modularität des MAB sollte sichergestellt werden, sodass Automatisierungsmodule flexibel in verschiedene Prozessketten eingebettet werden können.

2.4 Ausgangssituation und Voraussetzungen, unter denen das Vorhaben durchgeführt wurde

2.4.1 Der Stand von Wissenschaft und Technik

Zu Beginn des Projektes erfolgte die Absicherung meistens noch mittels konventioneller Hardware-Interfaces für diverse Fahrzeugschnittstellen. Aus Gründen der Flexibilität wurde aber zunehmend Hardware aus eigener Entwicklung des ASPA eingesetzt, um auf die vielfältigen Testgegebenheiten schnell und kostengünstig zu reagieren. Der zunehmende Einsatz eingebetteter Systeme hatte eine signifikante Verkürzung der Reaktionszeiten auf neue Gegebenheiten zur Folge.

Für die Automatisierung von Testumfängen wurden sowohl externe als auch interne Softwarekomponenten eingesetzt. Darüber hinaus wurde zu dieser Zeit ein Test-Framework entwickelt, welches zu diesem Zeitpunkt noch keine KI-Unterstützung beinhaltete. Im Rahmen des Projektes wurde jenes als Grundlage für die Erweiterung um KI-Funktionalitäten herangezogen.

2.4.2 Bisherige Arbeiten des Antragstellers

11-ASPA

Unternehmensvorstellung

Das Projekt wurde von der Firma SPARKS Solutions GmbH initiiert, jedoch wurde die Firma im Projektverlauf Aufgrund ihrer Insolvenz von der Firma ACONEXT übernommen und in ACONEXT Sparks GmbH umfirmiert. SPARKS Solutions, als Bestandteil der SPARKS GmbH, trat als Infotainment-Spezialist in Erscheinung und verfügte über eine langjährige Expertise in der Automobilindustrie. Als ACONEXT Sparks GmbH werden die Geschäfte nahtlos fortgeführt. Die ACONEXT Sparks GmbH bietet eine breite Palette von Dienstleistungen an, mit denen sie Projekte von der Anforderung bis zur Implementierung begleitet. Die Grundlage bildet ein Spezialistenteam, welches seine individuellen Kompetenzen in sämtlichen Geschäftsbereichen einsetzt, um die Zukunft von Infotainmentsystemen zu gestalten.

Die Geschäftsbereiche der ACONEXT Sparks GmbH erstrecken sich dabei von entwicklungsbegleitenden Leistungen über Projektmanagement bis hin zur Entwicklung von Hard- und Softwarekomponenten.

Relevante Kompetenzen und bisherige Arbeiten

Insbesondere im Bereich der Hard- und Softwareentwicklung hat sich das Unternehmen in jüngerer Vergangenheit auf die Automatisierung von Testprozessen fokussiert. ASPA entwickelt dafür Eingebettete Systeme, Hardwareschnittstellen sowie Softwarebausteine, um diese für automatisierte Testabläufe zu kombinieren.

ASPA hat Erfahrung in der Entwicklung von Softwareanalysetools zur Durchführung von Simulations- und Absicherungsaufgaben an den gängigen Fahrzeugbussystemen mit der Entwicklung des SPARKScommander gesammelt. Der SPARKSio, ein Multibusinterface für die gängigen Hardwareschnittstellen im Fahrzeug, ist mit dem SPARKScommander kompatibel. Mit SPARKSspy100 und SPARKSspy1000 wurden zwei Automotive-Ethernet-Hardwareinterfaces entwickelt, die ebenfalls mit dem SPARKScommander verwendet werden können. Die SPARKS Testing-Suite fungiert als zentrales Testing-Tool zur Abbildung des gesamten Testprozesses. Darüber hinaus wurden verschiedene Projekte zu Hard- und Softwareinterfaces für Restbussimulationen und Testautomatisierung sowie weitere Testautomatisierungstools, HMI-Automatisierung und Schnittstellentests entwickelt.

3 Technische Ergebnisse

3.1 Arbeitspaket 5: Validierung auf Fahrzeug-/System- und Komponentenebene mittels Demonstratoren

Anhand der in Arbeitspaket 5 gestaltenden Demonstratoren-Ausprägungen sollten die drei Arbeitsfelder Kabelbaum/Bordnetz + KI-Ansatz - Elektronik-Komponenten und eda/KI (KI im Entwicklungs-prozess und Agile Entwicklung) adressiert werden.

Es wurden die erzielten F+E-Ergebnisse aus AP 1 – 4 angewendet und so deren Realisierbarkeit mittels Demonstratoren (Proof-of-Concept) nachgewiesen. Die Demonstrator-Ausprägungen decken Schlüsselpositionen der Automobilindustrie und damit die jeweiligen Wertschöpfungsketten ab.

Die jeweils geschaffenen Instanzen zum Arbeitsfeld eda/KI decken wesentliche Bereiche der jeweiligen Wertschöpfungskette ab und können von Nutzern sowohl für die KI-basierte Automatisierung wertschöpfungsketten-übergreifender Entwurfsprozesse als auch für die Automatisierung einzelner im Entwurfsprozess kritischer Schritte eingesetzt werden.

Die Spezifikationen aus AP1/TA1.4 (elektronische Bordnetzmodule/Demo-Bordnetze + Testumgebungen Elektronik- + SW-Module) wurden zur Realisierung der Demonstratoren (TA 5.1 – 5.6) herangezogen. Zu den Validierungen wurden sowohl stationäre Demonstrator-Aufbauten (z.B. Chassis) als auch Testfahrzeug(e) herangezogen.

Die Ergebnisse aus AP4 (Testumgebungen für Integrations- + System-/Komponententests auf Demonstratorbasis (Elektronik-/SW-/Bordnetz-Komponenten)) wurden ebenfalls ab Q8/M22 auf die Anwendungs-Ebenen: Kommunikation - Energiesysteme - Sensorik - Digitale Zwillinge - KI-Beschleuniger - KI-Module/KI-Hardware/Elektronikmodule abgebildet:

Im Detail wurden in AP5 die folgenden Themen bearbeitet (Proof-of-Concept):

- Kommunikation: Test/Messungen Bordnetz PoDL/ETH/Serdes; Abgleich Simulationsergebnisse; Evaluierung Zonenkonzepte; Validierung KI-Module. Test und Verifikation der jeweiligen Ausprägung der digitalen Zwillinge.
- Energiesysteme. Test/Messungen Energie-Bordnetz; Abgleich Simulationsergebnisse; Evaluierung Zonenkonzepte; Validierung KI-Module.
- Sensorik. Test/Messungen Sensornetze; Abgleich Simulationsergebnisse; Evaluierung Zonenkonzepte; Validierung KI-Module.
- Digitale Zwillinge. Digitale Zwillinge an einem kontrollierten Aufbau mittels Bordnetzprüf-stand testen/verifizieren.
- KI-Beschleuniger. Nutzung der entwickelten Plattform zur Analyse, Entwicklung und Optimierung der geplanten KI-Beschleuniger für elektronische Bordnetz-Komponenten; Nutzung der entwickelten Plattform zur Demonstration einer typischen Applikation des modell-basierten Lernens.

- KI-Module/KI-Hardware/Elektronikmodule: Durchführung umfangreicher Tests (Best Practice) für die auf System-/Komponentenebene entwickelten Demonstratoren.

Die Evaluierung der Leistungsfähigkeit der Demonstratoren erfolgte auch mit realen Fragestellungen aus dem Feld (z.B. Planung eines Kabelbaums oder Schaltungsentwurf für eine Anwendung MicroContoller/integrierter Sensor). Das daraus gewonnene Feedback wurde genutzt, um eine weitere Iteration der Demonstrator-Ausprägungen vorzunehmen.

Für alle geplanten Demonstratoren sollte auf Industrieseite TRL 5 erreicht werden. Die Forschungspartner sollten minimal TRL4 und in Einzelfällen TRL5 erreichen.

3.1.1 Teilaufgabe 5.6: Validierung eines KI-gestützten Robotiksystem zur automatisierten Abtestung von Mensch-Maschine-Schnittstellen sowie einer KI-gestützten Testautomatisierungsumgebung auf Schnittstellenebene

Ziele

Im Rahmen der Validierung von KI-Module/KI-Hardware/Elektronikmodule sollten umfangreiche Tests (Best Practice) für die auf System-/Komponentenebene entwickelten Demonstratoren erarbeitet werden. Damit sollten Validierungsergebnisse und Bewertung zu dem KI-gestütztes Robotiksystem, dem HW-Interface und der KI-gestützten Testautomatisierungsumgebung auf Schnittstellenebene erzielt werden.

Problemstellung

5.6.1: Validierung, Demonstration und Bewertung eines Robotiksystems, welches auf ein räumlich beliebig orientiertes HMI reagiert, Bedienelemente erkennen und Bedienungen ausführt.

5.6.2: Validierung, Demonstration und Bewertung eines Robotiksystems, welches sich auf geänderte Bedingungen (z.B. neues HMI-Layout) anpasst.

5.6.3: Validierung, Demonstration und Bewertung eines Robotiksystems, welches aus Testspezifikationen Spracheingaben erzeugt.

5.6.4: Validierung, Demonstration und Bewertung eines Robotiksystems, welches Sprachausgaben eines HMI interpretiert und damit einen Testcase bedient.

5.6.5: Validierung, Demonstration und Bewertung eines Demonstrators der einen Fahrzeug-Trace KI-gestützt analysiert und Abweichungen detektiert.

5.6.6: Validierung, Demonstration und Bewertung eines Demonstrators, der aus Testspezifikationen automatisiert Test-Cases erzeugt, die sich für eine Automatisierte Testumgebung verwenden lassen.

5.6.7: Validierung, Demonstration und Bewertung eines Demonstrators der anhand von Testspezifikationen und Testdaten einen Test als bestanden oder als nicht bestanden beurteilt bzw. die Testkriterien als erfüllt oder nicht erfüllt kennzeichnet.

5.6.8: Validierung, Demonstration und Bewertung eines Demonstrators, der anhand von Testspezifikationen und Testdaten automatisiert Eingaben für ein Problemmanagementsystem erzeugen soll.

5.6.9: Validierung, Demonstration und Bewertung eines Demonstrators, der anhand von Testspezifikationen und Testdaten automatisiert Testberichte erzeugen soll.

Lösungsweg und Ergebnisse

Ergebnisse zu 5.6.1 bis 5.6.4:

Demonstrator Roboter-KI-Agent mit YOLO und LLM (UC2+5)

Der Demonstrator zeigt einen Roboter-KI-Agenten, der klassische Objekterkennung mit YOLOv8 [1] und ein Large Language Model kombiniert. Der Roboterarm ist in eine HiL-Anlage integriert und mit einer Stereo-Tiefenkamera zur Entfernungsmessung, einer 4k-Kamera für hochauflösende Bilder und einem Touchpen ausgestattet, um umfassende End-to-End-Tests zu ermöglichen. Der Agent führt automatisierte HMI-Tests am Infotainmentsystem mithilfe des Roboterarms und der 3D-Kamera durch. Bei Erhalt eines neuen Infotainmentsystem-Aufbaus, muss der Aufbau einmalig gescannt, in einer Datenbank gespeichert werden und mit einem QR-Code versehen werden, sodass der Roboterarm sich anhand des QR-Codes präzise zum Display ausrichten kann. Dies ist auch bei unterschiedlichen oder leicht verschobenen Aufbauten möglich. Die Software des Infotainmentsystems muss dabei allerdings die gleiche bleiben, da das YOLO-Modell für einen spezifischen Typ trainiert wurde. Testfälle werden in natürlicher Sprache formuliert, vom LLM interpretiert und in konkrete Testschritte umgesetzt. Fehlende Schritte können automatisch ergänzt werden, basierend auf der visuellen Analyse des Displays durch YOLO.

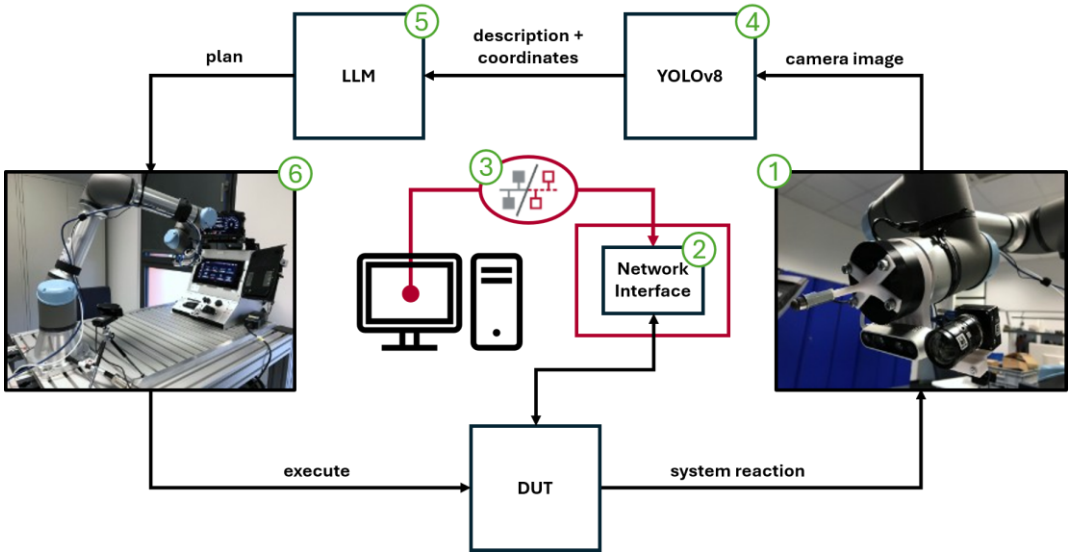

Abbildung 1 Anpassung des HIL-Aufbaus unter Einbeziehung eines 6DOF-Roboterarms, Bildverarbeitung mit YOLO und Large Language Model

Die Komponenten des robotergestützten HiL-Setups sind:

1. **Zwei Kameras:**
   Die Verwendung einer Stereo-Tiefenkamera anstelle eines Image Grabber ermöglicht eine präzise Entfernungsmessung zum Touchscreen, was für die Navigation des Roboterarms an die richtige Stelle unerlässlich ist. Die 4k-Kamera wurde hinzugefügt, um jedes kleine Detail des Infotainment-Systems zu erfassen (da die Tiefenkamera nur Full-HD-Bilder liefert, was als unzureichend empfunden wurde). Die zusätzliche Verwendung der 4k-Kamera anstelle eines Bildgrabbers war für einfache Koordinatenberechnungen erforderlich.
2. **Netzwerk Interface:**
   Das Netzwerk Interface dient dem gleichen Zweck wie unter Abbildung 6 beschrieben.
3. **Restbussimulation:**
   Das restliche Bussystem wird simuliert, damit der Agent auf so viele Menüs wie möglich zugreifen kann.
4. **YOLOv8:**
   YOLO (You Only Look Once) ist ein Computer Vision Modell zur Objekterkennung. Es kann mehrere Objekte in einem Bild lokalisieren und klassifizieren und so auch Bedienelemente in User Interfaces erkennen. Wir haben es mit gelabelten Bildern von „unserem“ Infotainmentsystem für unseren UseCase traninert. Zusätzlich wurde PaddleOCR [2] zur Texterkennung eingesetzt. Mit Hilfe der beiden Tools können wir die Bilder unserer Kamera live auswerten und alle erkannten Objekte und Texte ausgeben. Alle gesammelten Informationen werden von einem Algorithmus gruppiert und zusammengefasst und so für die Weiterverarbeitung mittels LLM aufbereitet.

- **Beispiel für aufbereitete Daten von YOLO und PaddleOCR:**

```
[
        "The image displays the Settings submenue of an automotive infotainment system.",
        "Non-clickable Text with text \"Settings\".",
        "Button with text \"Connected devices\", which is clickable.",
        "Button marked with text \"Connection settings\" that is clickable.",
        "Clickable Button with a \"media\" icon and text \"Entertainment\".",
        "Clickable Button with a \"language keyboard\" icon and text \"Language& keyboard\".",
        "Button marked with text \"Display& brightness\" that is clickable.",
        "Clickable Button with text \"System maintenance\".",
        "Button featuring a \"media\" icon and text \"A\", which is clickable.",
        "Button featuring a \"phone\" icon and text \"C\", which is clickable.",
        "Button marked with a \"tone\" icon and text \"Sound\" that is clickable.",
        "Clickable Button with a \"radio\" icon.",
        "Clickable Button featuring a \"settings\" icon and text \"General\".",
        "Button featuring a \"home\" icon, which is clickable.",
        "Button marked with a \"car\" icon that is clickable.",
        "There is an indicator that shows you can scroll left and right."
]
```

Jede Zeile in der Liste entspricht dabei einem erkannten Bedienelement. Die erkannten Informationen von YOLO und PaddleOCR sind hier bereits vereint und wurden zur besseren Weiterverarbeitung mit dem LLM zu vollständigen Sätzen zusammengefügt. In Versuchen hat sich gezeigt, dass dieses Vorgehen zu weitaus besseren Ergebnissen führt, als wenn alle Daten in ihrem Rohformat vom LLM verarbeitet werden würden. Zu jedem Bedienelement gehört außerdem eine Bounding Box aus Koordinaten, die hier aber nicht dargestellt wurden.

5. **LLM:**
   Die Aufgabe des Large Language Models ist es, anhand der gestellten Aufgabe und den Daten von YOLO zu entscheiden mit welchen Bedienelement interagiert werden soll. Die Informationen, von YOLO enthalten eine Beschreibung des Bedienelementes und die Koordinaten auf dem aktuellen Screenshot. Ein Algorithmus rechnet die Koordinaten vom Screenshot in Koordinaten im Roboterkoordinatensystem um. Dies ist möglich durch die Messung der Tiefenkamera, welche am Endeffektor des Roboters angebracht ist.
   Beispiel:
   Angenommen der Agent hat folgende Aufgabe erhalten: *"Change the language to english."* und den Inhalt von YOLO wie im obigen Beispiel, dann wäre die korrekte Wahl aus der Liste *"Clickable Button with a \"language keyboard\" icon and text \"Language& keyboard\"."* Mit der Action *"tap"*, um zum nächsten Menü zu gelangen.
6. **Roboterarm mit Touchpen:**
   Der verwendete Roboterarm (UR5e [3]) ist ein Cobot, der für die Zusammenarbeit zwischen Mensch und Maschine vorgesehen ist und sich daher perfekt für den Einsatz in einer Laborumgebung eignet. Er ist mit einer 3D-gedruckten Touchpen-Halterung sowie den beiden Kameras ausgestattet. Der Roboterarm hat sechs Freiheitsgrade (6DOF) und kann so das vor ihm platzierte Infotainmentsystem ohne Probleme erreichen (siehe Abbildung 1).

ReAct Agent im LangChain Framework

LangChain [4] ist ein OpenSource Framework, das es ermöglicht auf sehr einfache Weise KI Agenten zu erstellen. Ein zentrales Element in LangChain sind sogenannte Chains, die das grundlegende Prinzip darstellen, diverse KI-Komponenten zu vereinen, um kontextsensitive Antworten zu generieren. Eine Chain repräsentiert dabei eine Sequenz automatisierter Aktionen, die von der Anfrage bis zur Modellausgabe reicht. LangChain implementiert eine Standardschnittstelle für große Sprachmodelle, was den Austausch von KI-Modellen extrem einfach und schnell möglich macht.

Ein ReAct-Agent ist ein KI-Agent, der das „**Re**asoning and **Act**ing“ (ReAct)-Framework [5] nutzt, um Chain-of-Thought-Reasoning (CoT) mit dem Einsatz externer Tools zu kombinieren. Das ReAct-Framework verbessert die Fähigkeit von großen Sprachmodellen (LLMs), komplexe Aufgaben und Entscheidungsfindungen in agentischen Arbeitsabläufen zu bewältigen.

In LangChain ist es wie folgt umgesetzt:

1. **System Prompt:**
   a. **Hintergrundbeschreibung:**
      Allgemeine Beschreibung der Rolle/Situation/Aufgabe im Prompt.
   b. **Tool Beschreibungen:**
      Detaillierte Beschreibung von jedem Tool, das vom Agenten verwendet werden kann. Das LLM kann dann ein Tool anhand der Beschreibung und des Namens passend zur Situation auswählen. Das LangChain Framework sorgt im Hintergrund automatisch für die Ausführung der Tool Funktionen, was die Entwicklung von Agenten mit diesem Framework so einfach macht.

c. **Formatvorgabe (ReAct Pattern Format):**
   Vorgabe einer Struktur (z.B. JSON), die das LLM beim Antworten einhalten muss, damit die Antwort korrekt verarbeitet werden kann. Die relevanten Informationen werden durch Textverarbeitung aus der Antwort des LLM extrahiert.
d. **Beispiele (Few Shot Examples):**
   Beim Few-Shot Prompting erhält ein KI-Modell einige Beispiele für eine Aufgabe, aus denen es lernen kann. Anschließend generiert es eine Antwort und nutzt die Beispiele, um seine Leistung bei ähnlichen Aufgaben zu verbessern.
e. **Konkrete Frage/Aufgabe:**
   Der aktuelle Test, der durchzuführen ist, wird in natürlicher Sprache angegeben. Durch das Reasoning des KI-Agenten, ist es nicht notwendig alle Testschritte vollständig zu beschreiben, da der Agent in der Lage ist, die notwendigen Schlüsse aus der jeweiligen Situation zu ziehen.

2. **Agent Scratchpad:**
   a. **Speicherung von Beobachtungen:**
      Dies dient als temporärer Speicher, um Erkenntnisse aus der Interaktion mit externen Tools oder der Analyse von Daten während des Denkprozesses zu bewahren. Diese Beobachtungen werden anschließend in den Ursprungsprompt integriert, wodurch das Sprachmodell seine Denkweise verfeinert und die ursprüngliche Abfrage effektiver beantwortet.
   b. **Injektion von Kontext:**
      Um die Zustandslosigkeit von Sprachmodellen zu überwinden, bewahrt das Agent Scratchpad die Historie vergangener Konversationen oder relevanter Informationen aus vorherigen Schritten im mehrstufigen Denkprozess. Diese Injektion von Kontext ermöglicht es dem Modell, Zusammenhänge zu wahren und auf einem vorherigen Verständnis während der Interaktion aufzubauen. Ein Nachteil ist allerdings, dass die Verarbeitung immer langsamer wird, evtl. hohe Kosten verursacht und die Kontextlänge des KI-Modells überschreiten kann, bevor die Aufgabe erfüllt ist.
   c. **Reasoning + Acting:**
      Das LLM gibt eine Aktion samt Begründung an, die ausgeführt werden soll. Die Angabe der Begründung unterstützt den „Denkprozess“ des LLM und führt erfahrungsgemäß zu besseren Ergebnissen (ReAct), hat allerdings den Nachteil, dass es den Prozess der Inferenz des Modells verlangsamt. Um sicherzustellen, dass die Funktionen erfolgreich aufgerufen werden können, stellt das LangChain-Framework verschiedene Output-Parser zur Verfügung. Diese Parser ermöglichen die Umwandlung des unstrukturierten Textes des Sprachmodells in ein JSON-Objekt, sodass die jeweilige Funktion samt ihrer Parameter korrekt aufgerufen werden kann.

ReAct Agent im LangGraph Framework

LangChain und LangGraph [6] wurden beide vom gleichen Hersteller *LangChain Inc.* entwickelt. Beide sind Open Source und beide wurden entwickelt, um Entwicklern bei der Erstellung von LLM-basierten Anwendungen zu helfen.

LangChain basiert auf der Idee der Verkettung von Operationen, wie eine Pipeline, in der jeder Schritt vom Ergebnis des vorherigen abhängt. LangGraph hingegen wurde für komplexere, zustandsbehaftete Workflows entwickelt. Es wurde speziell für die Erstellung von Multi-Agenten-Systemen und zur Verarbeitung nichtlinearer Prozesse entwickelt. Die Verwendung von LangChain ist für Einsteiger etwas einfacher.

LangGraph basiert auf einer Graphstruktur, in der jede Aktion ein Knotenpunkt und die Übergänge zwischen den Aktionen Kanten sind. Jeder Knoten ist eine einzelne Funktion, welche z.B. ein Aufruf eines LLM mit einem Prompt sein kann. Dies hat den Vorteil, dass jeder Prompt hochspezialisiert ist, was dazu führt, dass die Abarbeitung schnell erfolgt und somit Ressourcen gespart werden können.

Die Graph Struktur ermöglicht auch Schleifen und das Zurückkehren zu früheren Zuständen, wodurch sie sich ideal für interaktive Systeme eignet, bei denen der nächste Schritt von sich verändernden Bedingungen oder Benutzereingaben abhängt. Ein Vergleich des Ablaufes unseres Testing-Agenten in LangChain und LangGraph (Abbildung 2) zeigt die Unterschiede deutlich auf.

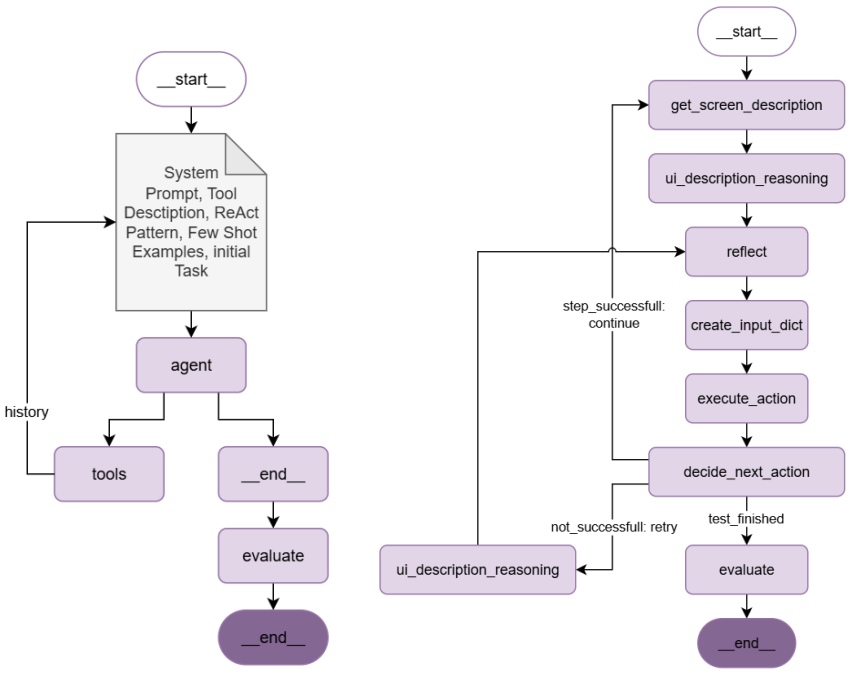

Abbildung 2 LangChain (links) vs. LangGraph (rechts) Framework

Demonstrator Roboter-KI-Agent mit VLM und LAM (UC2+5)

Der entwickelte Demonstrator stellt eine Weiterentwicklung der Roboter-basierte HMI-Testautomatisierung dar. Jüngste Fortschritte bei VLMs haben den Weg für eine neue Generation von Agenten geebnet, die zu komplexeren Interaktionen fähig sind, insbesondere in Bereichen, die sowohl visuelles Verständnis als auch die Ausführung von Aktionen erfordern. Wir nutzen daher diese Fortschritte, um einen neuen autonomen Agenten für HMI-Tests im Automobilbereich zu entwickeln.

Anstelle von YOLOv8 werden hier Vision Language Models (VLM) in Kombination mit Large Action Models [7] (LAM) und Large Language Models (LLM) verwendet. Während ein VLM zur semantischen Analyse des Bildschirminhalts eingesetzt wird, übernimmt das LLM die Interpretation natürlichsprachlicher Testanweisungen und die Ableitung der erforderlichen Interaktionen aus der Bildschirmbeschreibung. Das LAM, eine spezialisierte Klasse von VLM, ermöglichen die präzise Lokalisierung von Bedienelementen durch die Rückgabe exakter Koordinaten, was mit konventionellen VLM (selbst von führenden Anbietern wie OpenAI oder Google) nicht zuverlässig realisierbar ist. Diese Kombination adressiert die bisherige Limitation der YOLO-basierten Lösung und erlaubt eine robuste, adaptive und hochpräzise Ausführung von Testschritten durch den Roboterarm. Das Testsystem ist nicht mehr beschränkt auf ein bestimmtes Infotainmentsystem mit dem das YOLO-Modell trainiert wurde, sondern kann ganz ohne zusätzliches Training auf beliebige Infotainmentsysteme angewendet werden. Die Architektur demonstriert damit einen signifikanten Fortschritt in der KI-gestützten Automatisierung komplexer Interaktionsszenarien im Automotive-HMI-Bereich.

Während traditionelle VLMs sich in erster Linie auf die Interpretation und Beschreibung visueller Inhalte konzentrieren, zeichnen sich LAMs durch ihre Fähigkeit aus, Verständnis in konkrete Handlungen innerhalb ihrer Betriebsumgebung umzusetzen. Dieser handlungsorientierte Ansatz unterscheidet sie deutlich von ihren Vorgängern. Mehrere wesentliche Merkmale unterscheiden LAMs:

- **Aktionsorientierte Ausrichtung:**
  Im Gegensatz zu Modellen, die für die Textgenerierung oder die Informationsbeschaffung entwickelt wurden, sind LAMs grundsätzlich darauf ausgelegt, Aktionen auszuführen. Dieses zentrale Designprinzip ermöglicht es ihnen, aktiv mit ihrer Umgebung zu interagieren und diese zu verändern, eine Fähigkeit, die herkömmlichen Sprachmodellen fehlt.
- **Kontextuelle Intelligenz:**
  LAMs verfügen über ein ausgeprägtes Verständnis für Kontexte. Dieses tiefe Kontextbewusstsein ermöglicht es ihnen, Aktionen auszuführen, die nicht nur relevant sind, sondern auch sinnvoll auf die jeweilige Situation abgestimmt sind.
- **Zielorientierter Betrieb:**
  LAMs arbeiten in der Regel unter der Anleitung expliziter Ziele oder Vorgaben. Unabhängig davon, ob das Ziel darin besteht, eine bestimmte Aufgabe zu erfüllen, ein Problem zu lösen oder einen bestimmten Prozess zu optimieren, sind diese Modelle so konstruiert, dass sie konkrete, vorher festgelegte Ergebnisse erzielen.
- **Räumliches Denken:**

Traditionelle VLM wie z.B. GPT-4o von OpenAI haben große Schwierigkeiten mit Aufgaben, die eine präzise räumliche Lokalisierung erfordern, wie beispielsweise das Erkennen von Positionen von Objekten auf Bildern. LAM sind spezialisiert auf diese Aufgabe und in der Lage präzise Koordinaten von Objekten zu liefern (je nach Modell in Form eines Punktes oder einer Box).

In unserem neuen Agenten wurden YOLO, PaddleOCR und das LLM durch ein VLM und ein LAM ersetzt. Das VLM hat die Fähigkeit den Kontext des aktuellen Screens zu erkennen und darauf basierend eine Beschreibung und einen Testplan zu liefern. Das LAM hat die Fähigkeit, den Plan vom VLM zu analysieren und ein Objekt auf den Screenshot auszuwählen, mit dem interagiert werden soll. Es liefert außerdem direkt die Koordinaten für die Interaktion mit.

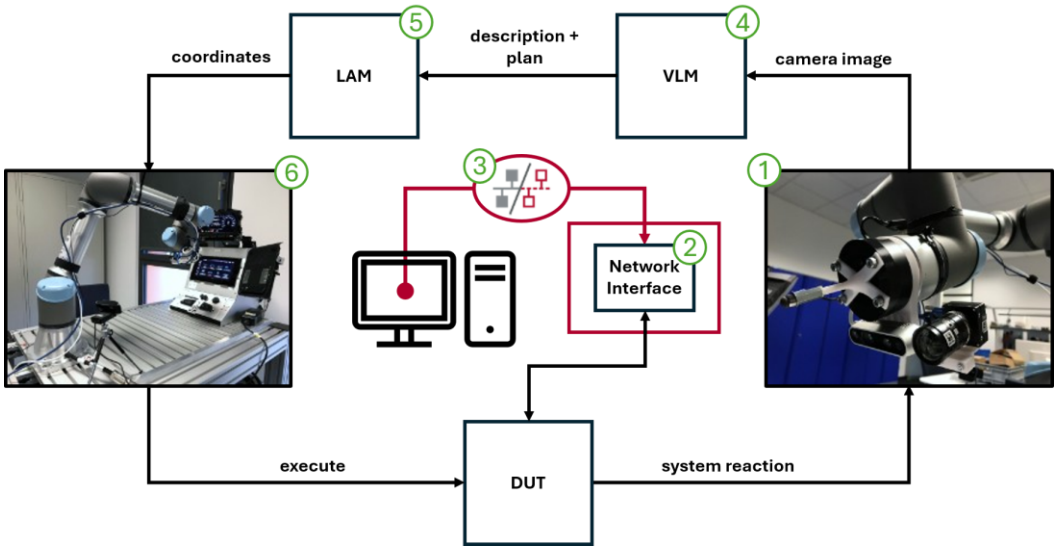

Abbildung 3 Anpassung des HIL-Aufbaus unter Einbeziehung eines 6DOF-Roboterarms und Vision Language Models

4. - 5. VLM und LAM:

LAM und VLM werden anstelle von OCR und Template Matching für die präzise Koordinatenbestimmung und visuelle Schlussfolgerungen verwendet. Wir verwenden ein spezielles LAM mit visuellen Grounding-Fähigkeiten für die Koordinatenbestimmung und ein hochmodernes VLM für die Bildanalyse und Schlussfolgerungen. Alle Modelle können leicht ausgetauscht werden, sodass verschiedene Modelle getestet und das für jede Aufgabe am besten geeignete gefunden werden kann.

(Für 1. – 3. und 6. siehe Abbildung 1)

Plan-and-Execute Agent

Unsere Implementierung des Agenten, insbesondere die Entscheidungsfindung darüber, welches UI-Element angeklickt werden soll, basierte zuvor ausschließlich auf textuellen Eingaben. Diese wurden von unserem eigens für die Infotainment-Domäne trainierten YOLO-Modell erkannt, welches Benutzeroberflächen analysieren und deren Inhalte in vordefinierten Sprachmustern als Text ausgeben kann. Allerdings haben sich große Sprachmodelle zunehmend zu multimodalen Alleskönnern entwickelt. Allen voran sind dies proprietäre

Lösungen wie GPT-4 von OpenAI, Claude von Anthropic sowie Gemini von Google. Populäre Open-Weight-Modelle wie z. B. Qwen, InternLm und Mistral haben nun ebenfalls die Fähigkeit zur Bild-, Sprach- und Videoverarbeitung erlangt. Um mit dem technologischen Fortschritt Schritt zu halten, haben wir unseren bisher eindimensionalen Agenten auf die direkte Verarbeitung von Screenshots des aktuellen Infotainment-Zustands umgestellt.

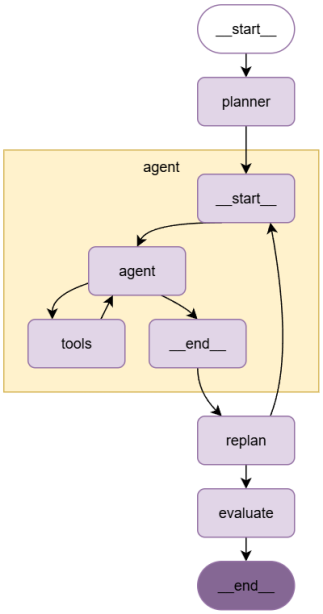

Abbildung 4 Plan-and-Execute Agent

Die Vorgehensweise des Agenten basiert auf einer Plan-and-Execute-Prompting-Strategie. Dieser Ansatz orientiert sich an den etablierten Methoden Plan-and-Solve [8] und ReAct-Prompting [5] und erweitert diese durch die Einbindung von Echtzeit-Feedback aus dem Infotainment-System. Konkret nutzt der Agent einen Screenshot des aktuellen HMI-Zustands, um seine Entscheidungen zu treffen. Durch diese Integration ist der Agent in der Lage, innerhalb einer Testschleife die folgenden Schlüsselfunktionen auszuführen:

- **Plan:**
  Zunächst wird das Modell angewiesen, einen Plan zu erstellen, der die gesamte Aufgabe in kleinere Teilaufgaben unterteilt. Anschließend führt das Modell die Teilaufgaben gemäß dem erstellten Plan aus.
- **Tools:**
  Der Agent verfügt über eine Reihe von Aktionen für die direkte Systeminteraktion, vor allem „Tippen“ und „Wischen“ in bestimmte Richtungen, wodurch er die HMI manipulieren kann. Die Tool-Komponente selbst stellt ebenfalls einen Agenten dar, der auf dem ReAct-Prompting-Muster basiert, sodass unser System de facto ein Multi-Agenten-System ist.
- **Reflective Replanning (Anpassung des Zustands):**
  Nach jeder ausgeführten Aktion begibt sich der Agent in eine Reflexionsphase. Er bewertet den aktuellen Zustand der HMI neu und passt seinen Plan dynamisch an, um etwaigen Änderungen oder unerwarteten Ereignissen, wie z.B. Popup-Dialogen,

Rechnung zu tragen. Diese adaptive Neuplanung, inspiriert vom Konzept der Selbstreflexion in KI-Systemen, gewährleistet die Robustheit und Genauigkeit des Testprozesses.

- **Evaluierung:**
  Basierend auf dem Konzept von LLM-as-a-Judge enthält der Agent einen Schritt zur Zustandsbewertung. Nach Abschluss eines Testschritts oder eines vollständigen Testfalls bewertet der Agent das Ergebnis, erstellt einen Bericht, in dem alle beobachteten Unregelmäßigkeiten detailliert aufgeführt sind, und gibt einen eindeutigen Pass/Fail-Status für den ausgeführten Test an.

Optimierung des Plan-and-Execute-Agenten für das Testen von Infotainmentsystemen in Fahrzeugen

Durch praktische Experimente mit dem Plan-and-Execute-Agenten haben wir bestimmte Bereiche identifiziert, die optimiert werden müssen, um seine Leistung und Eignung für das Testen von Infotainment-HMI in Kraftfahrzeugen zu verbessern. Unsere Beobachtungen haben die folgenden wesentlichen Einschränkungen aufgezeigt. Um diese identifizierten Mängel zu beheben und die Effektivität des Agenten zu verbessern, haben wir mehrere wichtige Modifikationen an der Plan-and-Execute-Architektur vorgenommen.

1. **Übermäßige Token-Nutzung und ineffiziente Modellnutzung:**
   Das ursprüngliche Agent-Design führte zu einer hohen Token-Nutzung, da die meisten Eingabeaufforderungen Bilder enthielten. Darüber hinaus erwies sich die Verwendung eines großen und rechenintensiven VLM für jeden Schritt als ineffiziente Nutzung von Ressourcen. In vielen Fällen konnte ein kleineres VLM oder sogar ein kleines LLM bestimmte Teilaufgaben innerhalb des Arbeitsablaufs des Agenten angemessen bewältigen.

   Lösung: Strategische Modellnutzung:

   Zur Optimierung der Ressourcennutzung haben wir eine Strategie zur selektiven Modellnutzung eingeführt. Das große VLM wird nun ausschließlich für die anfängliche Planungsphase verwendet. In dieser Phase generiert es eine textuelle Beschreibung sowie Gedanken (Thoughts) über den aktuellen Bildschirmstatus, anstatt einen starren, vordefinierten Plan. Anschließend kommt ein kleineres LLM zum Einsatz, um die vom VLM generierte Beschreibung und die damit verbundenen Gedanken zu konkreten Testschritten zu verdichten und den Aufruf von Tools zu handhaben. Dadurch wird komplexes visuelles Denken effektiv von der einfacheren Ausführung von Aktionen entkoppelt.

2. **Fehlende Bildschirmhistorie für Kontextbewusstsein:**
   Die zustandslose Natur des Agenten, dem die Erinnerung an vorherige Screenshots fehlte, stellte insbesondere bei der Navigation in scrollbaren Listen eine Herausforderung dar. Die Unfähigkeit, sich an vergangene Bildschirmzustände zu erinnern, beeinträchtigte die Fähigkeit des Agenten, das Ende einer Liste zu erkennen, was zu potenziellen Fehlern bei der Testausführung führte.

   Lösung: Traditioneller Bildvergleich beim Scrollen:

Zur Scroll-Aktion wurde eine spezielle Funktion hinzugefügt, die vor und nach dem Scrollen jeweils einen Screenshot aufnimmt. Anschließend werden beide Screenshots mit klassischen Bildverarbeitungsalgorithmen verglichen, um ihre Ähnlichkeit zu ermitteln. Wird ein festgelegter Grenzwert bei der Ähnlichkeit überschritten, kann davon ausgegangen werden, dass das Ende der Liste erreicht wurde und das Scrollen nicht weiter fortgesetzt werden sollte. Diese Information wird an das VLM und LLM weitergegeben, sodass der Plan entsprechend angepasst werden kann.

3. **Schlechte Erkennung von Togglebutton Zuständen:**
   Beim Testen stellte sich heraus, dass VLMs nicht in der Lage sind, den Zustand eines Togglebuttons (Ein/Aus) zuverlässig zu erkennen. In den Tests führte dies dazu, dass Togglebuttons in den falschen Zustand versetzt wurden, da das VLM davon ausging, dass der Togglebutton angetippt werden musste, obwohl er sich bereits im korrekten Zustand befand. Versuche mit LAMs zeigten hingegen, dass sich diese Technologie sehr gut für die Erkennung des Zustands von Togglebuttons eignet.

   Lösung: Spezialtool für die Interaktion mit Togglebuttons:

   Um die Einschränkungen allgemeiner VLMs bei der Handhabung von Umschaltknöpfen zu überwinden, haben wir ein zusätzliches Tool integriert, welches ein LAM nutzt, um den Zustand von Togglebuttons vor jeder Interaktion genau zu beurteilen. Durch die Abfrage des LAM kann der Agent zuverlässig feststellen, ob sich ein Togglebutton derzeit im Status „Ein" oder „Aus" befindet, was eine robustere und genauere Testplanung für Szenarien mit diesen UI-Elementen ermöglicht.

Vergleich und Validierung von Roboter KI-Agenten (UC2+5)

Zur Validierung der entwickelten Roboter-KI-Agenten wurde eine dedizierte Sandbox-Testumgebung implementiert, die den systematischen Vergleich unterschiedlicher Agentenarchitekturen, KI-Modelle und Prompt-Strategien ermöglicht. In diesem Rahmen erfolgte ein direkter Vergleich zwischen dem initialen Agenten, der auf YOLOv8 und einem Large Language Model (LLM) basiert, und der erweiterten Architektur, die Vision Language Models (VLM) sowie Large Action Models (LAM) integriert. Darüber hinaus wurde die Leistungsfähigkeit des zu diesem Zeitpunkt besten Open-Weight-VLM (Qwen) gegenüber dem führenden proprietären VLM (GPT-4o) untersucht. Die Evaluierung fokussierte auf Genauigkeit, Robustheit und Effizienz bei der Ausführung komplexer HMI-Testabläufe, um die Eignung der jeweiligen Modellkombination für den Einsatz in automatisierten Testprozessen im Automotive-Bereich zu bestimmen.

Der modulare Aufbau unseres Agenten-Frameworks bietet die Flexibilität, das zugrunde liegende VLM oder LLM problemlos auszutauschen. Diese Fähigkeit ermöglicht es uns, vergleichende Bewertungen verschiedener hochmoderner Modelle durchzuführen, darunter sowohl proprietäre Lösungen wie GPT-4 von OpenAI, Claude von Anthropic und Gemini von Google als auch immer leistungsfähigere Open-Weight-Modelle wie Qwen, InternLM und Mistral, die nun multimodale Verarbeitung von Bildern, Sprache und Videos bieten. Durch die systematische Erprobung dieser verschiedenen Modelle in unserem Agenten-Framework

wollen wir die für unsere spezifischen Ziele im Bereich der HMI-Tests für Automobile am besten geeigneten Modelle identifizieren und dabei Leistungskennzahlen wie Genauigkeit, Ausführungsgeschwindigkeit und Rechenaufwand bewerten.

Um eine strenge und kontrollierte Modellbewertung zu gewährleisten, haben wir eine simulierte Testumgebung oder „Sandbox“ entwickelt, anstatt uns ausschließlich auf das physische End-to-End-Testsystem zu verlassen. Diese Sandbox wurde unter Verwendung eines Wissensgraphen erstellt, der die visuellen und interaktiven Inhalte des Ziel-Infotainmentsystems umfassend darstellt. Der Wissensgraph enthält eine ausgewählte Sammlung von Fotos der Benutzeroberfläche des Infotainmentsystems, die mit der am Roboter montierten 4K-Kamera aufgenommen wurden. Alle relevanten anklickbaren Schaltflächen wurden mit Verweisen auf die nachfolgenden Bildschirme versehen. Alle diese strukturierten Informationen sind im GraphML-Format festgehalten. Dieser Sandbox-Ansatz bietet mehrere wichtige Vorteile für die Modellbewertung. Er gewährleistet konsistente Testbedingungen und eliminiert Störvariablen wie Änderungen in der Umgebungsbeleuchtung oder unbeabsichtigte Software-Updates des Live-Infotainment-Systems.

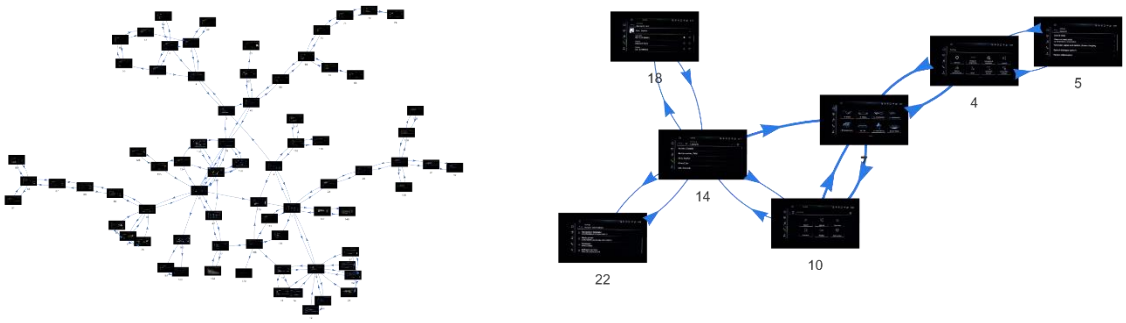

Abbildung 5 Knowledge Graph

Diese kontrollierte Umgebung ermöglicht eine präzisere und zuverlässigere Bewertung der Modellleistung. Darüber hinaus reduziert die Sandbox die Evaluierungszeit erheblich, da mehrere Testläufe parallel durchgeführt werden können. Außerdem erleichtern die vordefinierten Testsequenzen die Auswertung der Sandboxtests, da die optimale Abfolge von Bildern und Interaktionen in der Testsequenz festgelegt werden kann und dann nur noch mit der tatsächlichen Abfolge verglichen werden muss.

Vergleich der KI-Agenten mit Tabelle.

Zum damaligen Zeitpunkt der Auswahl wurde sich für den ThinkingAgent in Kombination mit dem Open-Weight-Modell Qwen2.5-VL-32B-Instruct [9] entschieden. Ausschlaggebend war die Möglichkeit das Modell InHouse zu betreiben, als auch die zu bewertenden Screenshots des aktuellen HMI-Zustands mit korrekten Pixelgenauen Koordinaten. Besonders die Koordinaten einzelner Elemente in einem Screenshot wurden von anderen KI-Modellen, mit Vision Funktionalität, eher zufällig bestimmt und waren somit nicht nutzbar.

Andere KI-Agenten wie bspw. der M18-Agent stellt keine Vision Funktionalität bereit. Dieser verwendet ausschließlich textuelle Beschreibungen der zu sehende Elemente eines

Screenshots. Jene müssen vorerst von einem weiteren Tool bereitgestellt werden. (bspw. Yolo)

Um eine interne Beurteilung des ThinkingAgent mit verschiedenen KI-Modellen durchzuführen, wurde bspw. gpt-4o-in Kombination mit Molmo (zur Bereitstellung korrekter Bildkoordinaten) gegenüber ThinkingAgent mit Qwen evaluiert.

Alle verwendeten KI-Agenten nutzten, für den direkten Vergleich untereinander, die gleichen KI-Modelle und es wurde ein, an reale Testsituationen orientiertes, Set von Prompts erstellt. Dadurch werden die einzelnen Testschritte bis hin zum vollständigen Testfall abgedeckt.

Beispiel eines realen Testfalls:

Testfall: Wechsel des aktuellen DriveSelect-Profils auf Comfort.

Im Test muss folgendes geprüft werden:

- Relevantes Menü und SubMenü vorhanden
  - Home - CAR
  - CAR - DriveSelect
- Menüs betretbar (Nicht Ausgegraut)
- Profil in DriveSelect vorhanden und bedienbar
- Auswahl des DriveSelect-Profils

Daraus resultierende Testcase-Prompts:

["Is 'CAR'== available in 'Current Screen'"]	Verfügbarkeitsprüfung
['Go To the CAR menue']	Bedienbarkeitsprüfung
['Go To "CAR"', 'Go To "DriveSelect"', 'Select "comfort"']	Vollständiger Testcase

Unterschiedliche Prompt-Strukturen für die einzelnen Testschritte:

['Set Comfort as Active DriveSelect Profile']
['Enter CAR menu', 'Enter Drive Select Menu', 'Select "comfort" driving mode in CAR DriveSelect Menu']
['Go To "CAR"', 'Go To "DriveSelect"', 'Select "comfort"']

Alle Prompts wurde jeweils 100-mal ausgeführt, vorausgesetzt die ersten 20 Durchläufe waren nicht 100 % fehlerhaft (z. B. durch Exceptions, Nichterreichen des richtigen Endscreens etc.).

Verwendete Prompts:

- **Verfügbarkeitsprüfung:**
  - "Check if 'CAR' is visible"
  - "Is 'CAR'== available in 'Current Screen'"
  - "Swipe right and check for 'User' menu entries"
  - "Swipe Right", "Check for 'User' Menue availablity"
- **Bedienbarkeitsprüfung:**
  - "Enter 'CAR"
  - "Go To the CAR menue"
  - "Enter the CAR Driver Assistance Menue"

- "Go To CAR","Go To Driver Assistance"

Vollständiger Testcase:

- **DriveSelect Profile:**
  - "Select Driving Profile \"comfort\" in the CAR Drive Select Menue"
  - "Set Comfort as Active DriveSelect Profile"
  - "Enter CAR menu", "Enter Drive Select Menu", "Select \"comfort\" driving mode in CAR DriveSelect Menu"
  - "Go To \"CAR\"", "Go To \"DriveSelect\"", "Select \"comfort\""
- **DriverAssist FatigueWarning:**
  - "Go To CAR Driver Assistant FatigueWarning and disable the menu entry"
  - "Deactivate \"Fatigue Warning\" in CAR DriverAssistant"
  - "Go To \"CAR\"", "Go To \"Driver Assistant\"","Scroll Down", "Scroll down", "Scroll down", "Scroll Down", "Select \"Fatigue Warning\" Submenu"
  - "Go To the CAR Driver Assistant FatigueWarning Submenu"

Für die Auswertung der KI-Agenten & -Modelle wurden folgende Daten erhoben:

- Prompt Erfolgsquote in %
- Prompt-Result (TRUE bei Prompt-Erfolgsquote >=97%)
- Durchschnittliche Zeit der Prompt-Durchführung in Sekunden
- (M18-Agent) Erreichter & Erwarteter Screen am Testende
- (ThinkingAgent) Target & Actual Trajectory von Start- zu Endscreen

KI4BN Auswertung Standarisierte Tests												
Prompt	M18 Agent						ThinkingAgent					
	Langchain						Qwen			gpt-4o		
	Qwen			gpt-4o								
	Result	P-Quote	Zeit(s)	Result	P-Quote	Zeit(s)	Result	P-Quote	Zeit(s)	Result	P-Quote	Zeit(s)
["Check if 'CAR' is visible"]	FALSE	0,00%	--	TRUE	100,00%	2,00	FALSE	96,00%	--	TRUE	100,00%	13,94
["Is 'CAR'== available in 'Current Screen'"]	FALSE	0,00%	--	TRUE	100,00%	3,05	FALSE	85,00%	--	FALSE	86,00%	--
["Swipe right and check for 'User' menu entries"]	FALSE	90,00%	--	FALSE	90,00%	--	FALSE	93,00%	--	FALSE	8,00%	--
['Swipe Right', "Check for 'User' Menue availablity"]	FALSE	0,00%	--	TRUE	100,00%	6,14	TRUE	100,00%	51,39	TRUE	99,00%	20,41
["Enter 'CAR'"]	FALSE	0,00%	--	TRUE	100,00%	5,83	FALSE	8,00%	--	FALSE	0,00%	--
['Go To the CAR menue']	FALSE	80,00%	--	TRUE	100,00%	6,05	TRUE	100,00%	42,16	FALSE	77,32%	--
['Enter the CAR Driver Assistance Menue']	FALSE	0,00%	--	TRUE	100,00%	8,98	TRUE	100,00%	52,60	FALSE	23,00%	--
['Go To CAR', 'Go To Driver Assistance']	TRUE	99,00%	17,61	TRUE	100,00%	7,85	TRUE	100,00%	54,34	FALSE	23,00%	--
['Select Driving Profile "comfort" in the CAR Drive Select Menue']	TRUE	100,00%	25,69	TRUE	100,00%	11,79	TRUE	100,00%	67,53	TRUE	100,00%	45,27
['Set Comfort as Active DriveSelect Profile']	FALSE	42,00%	--	TRUE	100,00%	11,59	TRUE	100,00%	79,15	TRUE	100,00%	40,32
['Enter CAR menu', 'Enter Drive Select Menu', 'Select "comfort" driving mode in CAR DriveSelect Menu']	FALSE	0,00%	--	TRUE	100,00%	11,86	TRUE	100,00%	73,45	TRUE	100,00%	43,35
['Go To "CAR"', 'Go To "DriveSelect"', 'Select "comfort"']	TRUE	100,00%	21,21	TRUE	100,00%	11,92	TRUE	100,00%	72,19	TRUE	100,00%	48,65
['Go To CAR Driver Assistant FatigueWarning and disable the menu entry']	FALSE	0,00%	--	FALSE	0,00%	--	TRUE	97,00%	142,68	TRUE	98,00%	83,64
['Deactivate "Fatigue Warning" in CAR DriverAssistant']	FALSE	0,00%	--	FALSE	0,00%	--	FALSE	73,00%	--	FALSE	6,00%	--
['Go To "CAR"', 'Go To "Driver Assistant"', 'Scroll Down', 'Scroll down', 'Scroll down', 'Scroll Down', 'Select "Fatigue Warning" Submenu']	FALSE	0,00%	--	FALSE	45,00%	--	FALSE	95,00%	--	FALSE	92,31%	--
['Go To the CAR Driver Assistant FatigueWarning Submenu']	FALSE	0,00%	--	FALSE	0,00%	--	FALSE	94,00%	--	--	--	--

Abbildung 6 Auszug aus Statistischer Auswertung

Ergebnisse zu 5.6.5:

Demonstrator Multivariate Zeitreihen Anomalie Detektion (UC4)

Die Möglichkeit des Einsatzes von Deep-Learning-Methoden zur Anomalieerkennung in multivariaten automobilen Zeitreihen wurde untersucht. Ziel war es, unterschiedliche Architekturen, wie Convolutional Neural Networks (CNN), Long Short-Term Memory Netze (LSTM) und Convolutional Autoencoder (CAE), systematisch zu vergleichen und ihre Leistungsfähigkeit unter realen Testbedingungen zu bewerten. Aufbauend auf den theoretischen Grundlagen zur Zeitreihenanalyse und den Charakteristika von Anomalien wurden reale Fahrzeugsensordaten herangezogen, um praxisnahe und belastbare Ergebnisse zu erzielen. Die Integration authentischer Fahrzeugdaten anstelle synthetischer Benchmarks gewährleistete eine hohe Relevanz für reale Entwicklungs- und Testumgebungen.

Datenumfang und Vorverarbeitung

Das Messsystem zeichnete sämtliche Bussignale des Fahrzeugs mit einer Abtastrate von 200 Hz auf, wodurch ein hochdimensionaler Datensatz mit über 1100 Signalen entstand. Zur Reduktion des Rechenaufwands und zur Verbesserung der Interpretierbarkeit wurde eine gezielte Merkmalsauswahl vorgenommen. Dabei wurden acht physikalisch interpretierbare Größen beibehalten, die das Fahrverhalten des Fahrzeugs in seiner Dynamik am besten beschreiben:

Merkmal	Beschreibung	Einheit	Bedeutung
v_ref	Fahrzeugreferenzgeschwindigkeit	km/h	Längsdynamik
Strom_HV	Hochvoltbatteriestrom	V	Leistungsanforderung des Antriebs
Bremsdruck	Hydraulischer Bremsdruck	bar	Bremsintensität
Lenkradwinkel	Absoluter Lenkwinkel	°	Fahrerinput
Lenkradw_Geschw	Winkelgeschwindigkeit des Lenkrads	°/s	Lenkraddynamik
Gierrate	Drehgeschwindigkeit um Hochachse	°/s	Fahrzeugrotation
Laengsbeschl	Längsbeschleunigung	g	Vorwärts-/Rückwärtsbewegung
Querbeschl	Querbeschleunigung	g	Kurven- und Seitenkräfte

Tabelle 2 Übersicht verwendeter Fahrdynamikgrößen

Diese Merkmale decken die zentralen Dimensionen der Fahrdynamik ab und stellen ein ausgewogenes Verhältnis zwischen Informationsgehalt und Modellkomplexität sicher.

Zur Vorbereitung auf das Modelltraining wurden sämtliche Eingangsvariablen standardisiert, um Verzerrungen durch unterschiedliche Wertebereiche zu vermeiden.

Labeling der Inferenzdatensätze

Zur quantitativen Bewertung der Modelle wurde ein manuelles Labeling der Inferenzdatensätze durchgeführt. Grundlage hierfür bildeten die im Test durchgeführten Manöver, die in sieben Datensätzen (DS1–DS7) dokumentiert sind. Jedes Dataset repräsentiert ein spezifisches Fahrverhalten, von starker Beschleunigung und ABS-Bremsung bis hin zu Drift- und Slalomversuchen.

DS	Maneuver Description	Purpose/Relevance	Anomaly Type	Representative Features
1	Strong acceleration to 120 km/h followed by ABS braking	Tests powertrain performance and ABS effectiveness in high-speed scenarios	Extreme acceleration and deceleration	v_ref, Strom_HV, Bremsdruck, Laengsbeschl
2	Medium acceleration to 80 km/h and braking to 20 km/h	Test vehicle performance in moderate speed scenarios	Moderate acceleration and deceleration	v_ref, Strom_HV, Bremsdruck, Laengsbeschl
3	Drifting	Tests vehicle stability and control during oversteer	Oversteer dynamics/ stability shift	Lenkradwinkel, Strom_HV, Lenkradw_Geschw, Gierrate, Querbeschl
4	Swaying at 50 km/h	Tests vehicle roll stability and suspension behavior at moderate speeds	Induced lateral oscillations	Lenkradwinkel, Lenkradw_Geschw, Querbeschl
5	Heavy braking lane changes	Tests stability and control during emergency evasive maneuvers with braking	Sudden direction change while braking	Lenkradwinkel, Laengsbeschl, Lenkradw_Geschw, Bremsdruck, Strom_HV
6	Constant circle drive steering angle 130 deg	Tests steady-state cornering behavior and understeer/ oversteer characteristics	Prolonged high lateral acceleration	Lenkradwinkel, Querbeschl, Gierrate
7	Slalom full steering lock	Tests quick direction changes and steering response at maximum steering angle	Rapid alternating lateral forces	Lenkradwinkel, Lenkradw_Geschw, Querbeschl

Abbildung 7 Übersichtstabelle der durchgeführten Fahrmanöver

Das Labeling erfolgte auf Basis von Expertenwissen und domänenspezifischem Verständnis der Fahrdynamik. Für jede Fahrsequenz wurden Zeitbereiche identifiziert, in denen sich signifikante Abweichungen vom Normalverhalten zeigten – z. B. extreme Ausschläge in Bremsdruck, Gierrate oder Querbeschleunigung.

Ein Beispiel sind die Datensätze DS1 und DS3 (starke Beschleunigung mit anschließender Vollbremsung bzw. Drifiting), bei dem die gleichzeitigen Änderungen von Geschwindigkeit, Stromaufnahme und Bremsdruck die ABS-Aktivierung klar erkennen lassen.

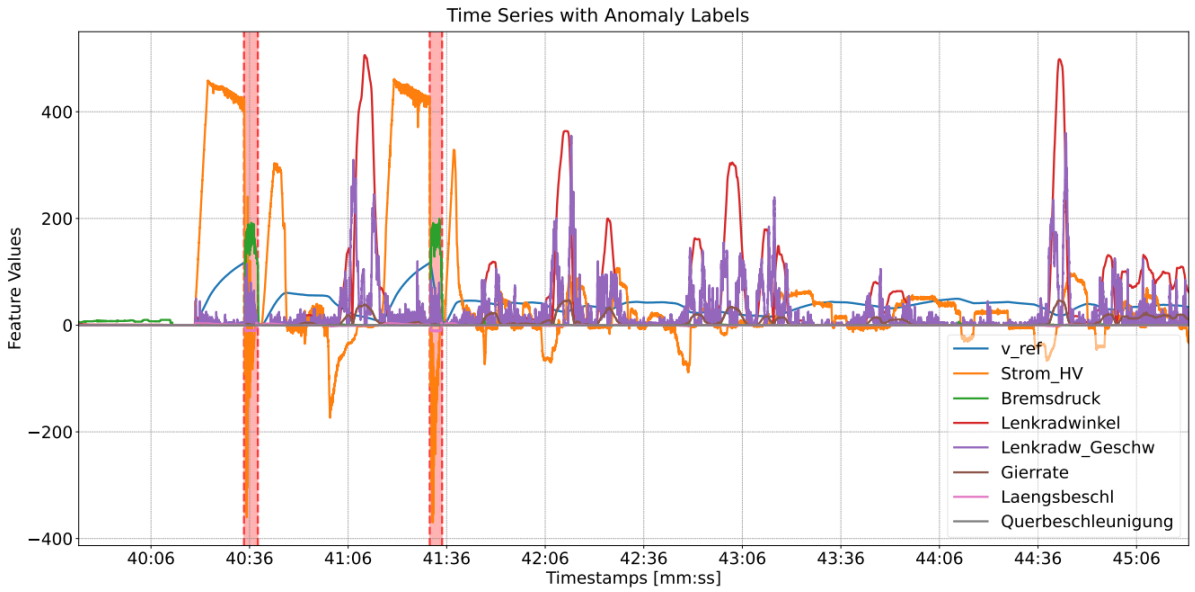

Abbildung 8 Zeitreihenplot von DS1 mit markierten anomalen Abschnitten.

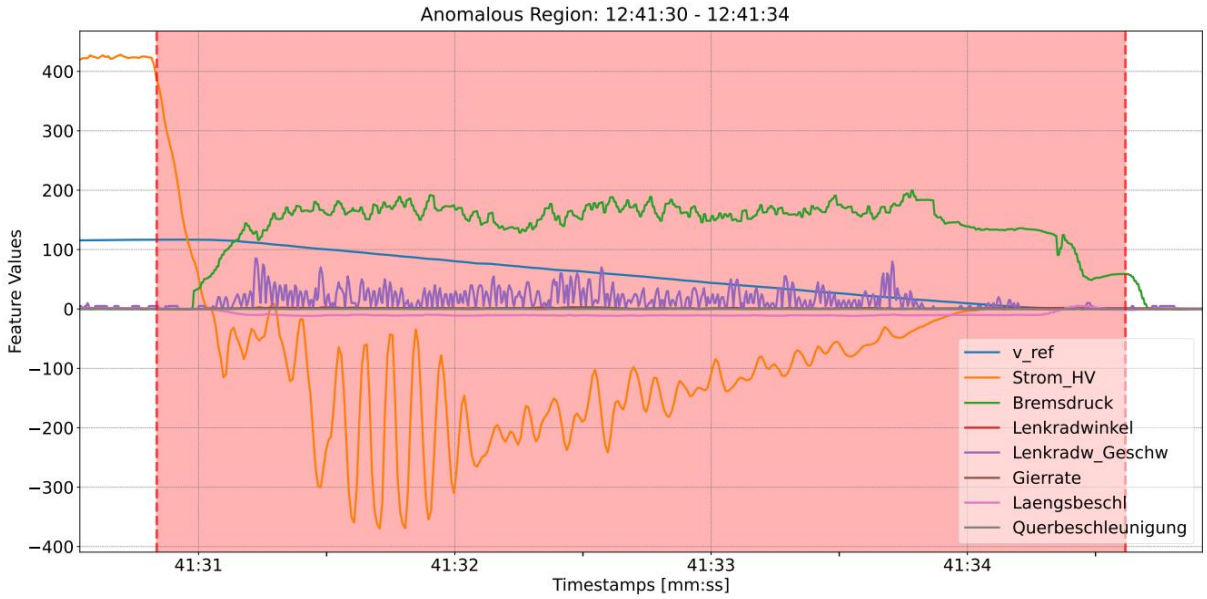

Abbildung 9 Zeitreihenplot von DS1 mit markierten anomalen Abschnitten (Zoom bei zweitem Abschnitt).

Die Kennzeichnung dieser Bereiche ermöglicht eine objektive Auswertung anhand von Metriken wie Precision, Recall, F1-Score und ROC-AUC. Dabei ist zu beachten, dass das Labeling trotz größter Sorgfalt subjektiven Einflüssen unterliegt, etwa durch Beobachter-, Erwartungs- oder Expertise-Bias. Daher werden die Labels als informierte Referenz, nicht als absolute Wahrheit betrachtet. Sie bilden dennoch die essenzielle Grundlage für die Evaluierung der Anomalieerkennungsmodelle in den folgenden Kapiteln.

Modellarchitekturen

Die stetige Weiterentwicklung der Deep Learning Verfahren hat das Feld der Zeitreihenanalyse grundlegend verändert. Besonders in der Prognose und Anomalieerkennung haben sich neuronale Netze als leistungsfähige Werkzeuge etabliert, um komplexe, zeitabhängige Muster zu erfassen und ungewöhnliche Abweichungen zuverlässig zu identifizieren.

Aufbauend auf diesen Erkenntnissen wurden im Rahmen der Forschungsarbeit drei modellbasierte Ansätze implementiert, die unterschiedliche Prinzipien des Deep Learning kombinieren: Convolutional Neural Networks (CNN) [10] , Long Short-Term Memory Netze (LSTM) [11] und Autoencoder (AE) [12]. Ziel war der Vergleich ihrer Eignung zur multivariaten Anomalieerkennung in Fahrzeugzeitreihen.

Da im Rahmen der Forschungsarbeit mit dem LSTM-basierten Ansatz die besten Ergebnisse erzielt wurden, wird in den nächsten Abschnitten nur dieser Ansatz näher erläutert.

LSTM-basierte Anomalieerkennung

Die LSTM-basierte Anomalieerkennung basiert auf dem von Pankaj Malhotra entwickelten LSTM-AD-Modell [11]. LSTM-Netze sind in der Lage, zeitliche Abhängigkeiten über lange Sequenzen zu modellieren, indem sie vergangene Informationen in Speicherzellen („gates“) über mehrere Zeitschritte hinweg bewahren.

Das Netzwerk besteht aus mehreren gestapelten LSTM-Schichten, die Eingangsdaten mit m Dimensionen verarbeiten und $d \times pw$ Ausgabeeinheiten erzeugen, wobei d die vorherzusagenden Dimensionen und pw den Vorhersagehorizont bezeichnen.

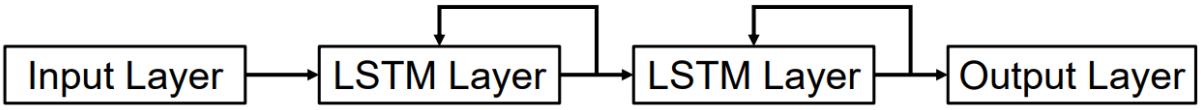

Abbildung 10 Struktur des LSTM-AD mit gestapelten LSTM-Ebenen.

Nach dem Training auf normalem Fahrverhalten berechnet das Modell für jede Zeit t einen Fehlervektor e_t, der die Differenz zwischen Vorhersage und Beobachtung enthält. Diese Fehlervektoren werden als multivariate Gauß-Verteilung N(μ, Σ) modelliert, wobei μ den mittleren Fehler und Σ die Kovarianzmatrix beschreibt. Anomalien ergeben sich aus Punkten mit geringer Likelihood p_t unter dieser Verteilung.

Zur Festlegung des Entscheidungs-Schwellwerts wird die F-β-Metrik genutzt. Mit $\beta < 1$ wird die Präzision stärker gewichtet, um Fehlalarme zu minimieren. Im Rahmen der Forschungsarbeit wurde zur Vergleichbarkeit der Modelle ein fester Schwellenwert implementiert, um eine einheitliche quantitative Bewertung zu gewährleisten.

Das LSTM-Verfahren zeigt insbesondere bei zeitlich korrelierten Anomalien seine Stärken, da es sequenzielle Abhängigkeiten explizit berücksichtigt, erfordert jedoch im Vergleich zu CNNs längere Trainingszeiten und eine sorgfältige Hyperparameter-Abstimmung.

Vergleich der Ansätze

Alle drei Architekturen basieren auf lernbasierten Konzepten, unterscheiden sich jedoch in ihrer Herangehensweise an die Modellierung zeitlicher und räumlicher Zusammenhänge:

Modell	Charakteristik	Vorteile	Herausforderungen
CNN	Fensterbasierte Kurzzeit-Vorhersage	Schnelle Berechnung, gute Mustererkennung	Eingeschränkte Langzeit-Abhängigkeiten
LSTM	Sequenzielle Langzeitmodellierung	Erfassung komplexer Zeitbeziehungen	Höherer Rechenaufwand, Hyperparametrik
CAE	Rekonstruktionsbasiert, unüberwacht	Keine Label nötig, robust gegen Rauschen	Sensitivität gegenüber Schwellenwert-Wahl

Tabelle 3 Gegenüberstellung der Modelle mit ihren spezifischen Charakteristiken

Alle Modelle teilen den grundlegenden Ansatz, das normale Systemverhalten zu erlernen und Abweichungen davon als potenzielle Anomalien zu interpretieren. Während das CNN auf kurzfristige Abweichungen reagiert, erkennt das LSTM vor allem längerfristige Veränderungen. Der CAE bietet schließlich eine vollständig unüberwachte Alternative, die insbesondere dann vorteilhaft ist, wenn keine ausreichenden Labeldaten verfügbar sind.

Ergebnisse und Diskussion

Nach der Implementierung der Modelle und der Definition der Versuchsdaten stand im nächsten Schritt die **Bewertung der Modellleistung** im Vordergrund. Ziel war es, die Fähigkeit der verschiedenen Deep-Learning-Ansätze zur Erkennung von Anomalien in multivariaten Fahrzeugdaten quantitativ und qualitativ zu beurteilen. Die Analyse umfasst sieben Datensätze (DS1–DS7), die unterschiedliche Fahrmanöver und Anomaliearten repräsentieren.

Die Ergebnisse werden zunächst quantitativ anhand etablierter Metriken bewertet, anschließend qualitativ interpretiert und im letzten Teil in Bezug auf die Forschungsfragen und Hypothesen diskutiert.

Quantitative Bewertung

Zur objektiven Beurteilung der Modellleistung wurden die Metriken Precision, Recall, F1-Score und AUC-ROC verwendet. Dabei wurden jeweils zwei Schwellenwertmethoden verglichen – ein empirischer Schwellenwert (TH_pq), der auf den obersten 1 % der Validierungsfehler basiert, und ein statistischer Schwellenwert (TH_st), der aus zwei Standardabweichungen der Residuen berechnet wird.

Diese Doppelbewertung erlaubt eine präzise Einschätzung, wie empfindlich die Modelle auf verschiedene Schwellenwertdefinitionen reagieren und wie sich die Balance zwischen Fehlalarmen (FP) und verpassten Anomalien (FN) verändert.

LSTM-AD (rekurrentes Modell)

Das LSTM-basierte Modell zeigte seine Stärken insbesondere bei zeitlich stark korrelierten Anomalien, wie sie bei abrupten Zustandswechseln auftreten. In DS1 und DS5 wurden F1-Scores von 0.94 bis 0.95 erzielt, was die Fähigkeit des Modells zur Modellierung sequenzieller Muster unterstreicht.

Dataset	DS1		DS2		DS3		DS4		DS5		DS6		DS7	
Threshold	Pq	St	Pq	St	Pq	St	Pq	St	Pq	St	Pq	St	Pq	St
Precision	1.00	1.00	0.19	0.18	1.00	1.00	0.90	0.94	1.00	1.00	0.94	0.87	1.00	1.00
Recall	0.89	0.90	0.00	0.00	0.18	0.32	0.04	0.13	0.81	0.83	0.01	0.02	0.13	0.20
F1	0.94	**0.95**	0.00	0.00	0.30	0.48	0.08	0.23	0.89	**0.90**	0.01	0.03	0.22	0.33
AUC-ROC	0.99		0.44		0.98		0.92		0.99		0.47		0.98	

Abbildung 11 LSTM-AD Ergebnisse zu Precision, Recall, F1, AUC.

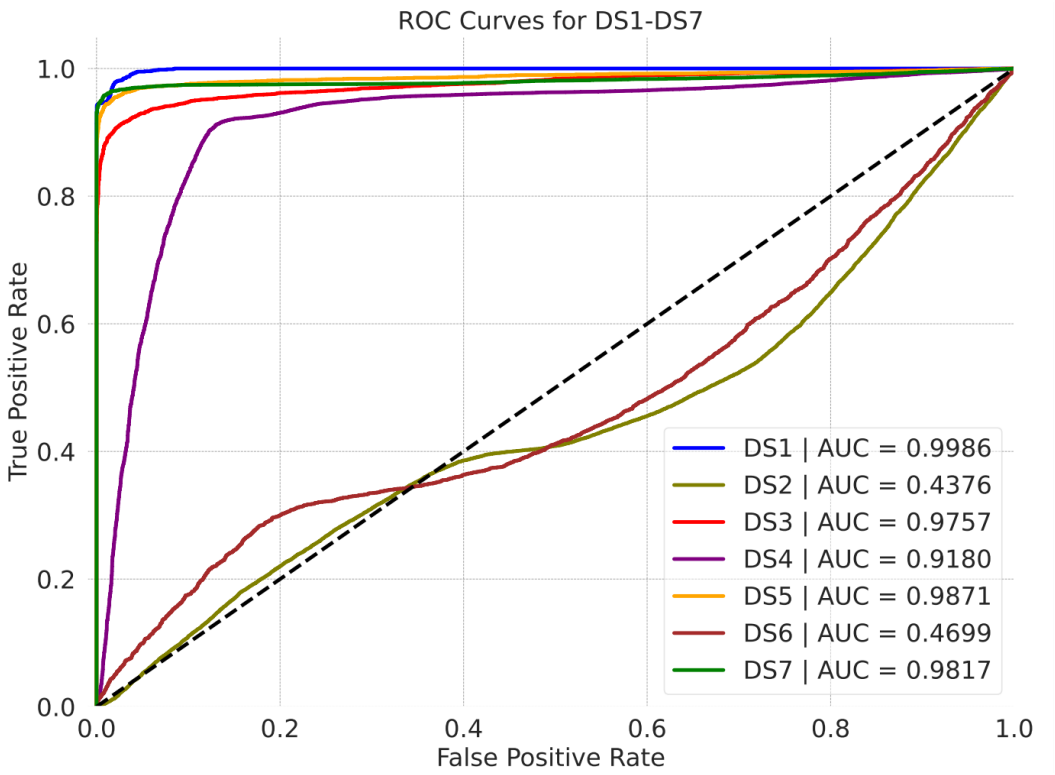

Abbildung 12 ROC-Kurven des LSTM-AD-Modells über DS1–DS7.

Allerdings zeigte sich ein deutlicher Recall-Verlust bei subtilen Anomalien (z. B. DS2, DS6), wo F1-Werte gegen null gingen. Der Grund: das LSTM „normalisiert“ graduelle Veränderungen, wodurch moderate Abweichungen nicht mehr als anormal erkannt werden.

Eine Schwellenwertoptimierung führte zu deutlichen Verbesserungen in DS3, DS4 und DS7 (F1 bis 0.98), während DS1 aufgrund zu strenger Schwellen sogar schlechter abschnitt.

Insgesamt belegt das Modell hohe Präzision, aber eine starke Abhängigkeit von der Kalibrierung der Entscheidungsgrenzen.

Qualitative Bewertung

Neben der quantitativen Auswertung wurde die Modellleistung auch visuell überprüft, um Korrelationen zwischen tatsächlichen Fahrmanövern und den modellgenerierten Anomaliescores zu identifizieren, vgl. Abbildungen 35 und 36.

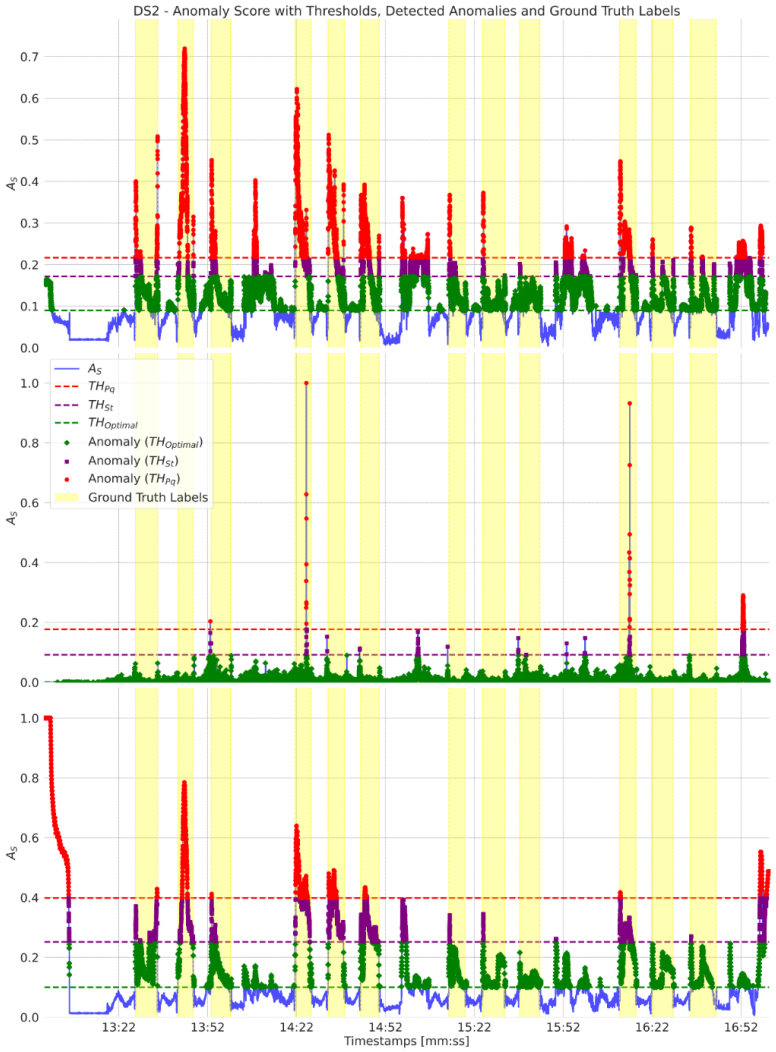

Abbildung 13 Anomaliescores für DS2 über alle Modelle mit Schwellenwerten und Ground-Truth-Labels. DeepAnt, LSTM-AD, ConvAE von oben nach unten.

Datensatz DS2 – Moderate Beschleunigung/Bremsung

Alle Modelle zeigten in diesem Szenario Schwächen:

- **DeepAnT**: sporadische Peaks → viele False Positives.
- **LSTM-AD**: zu konservativ → viele False Negatives.

- **CAE**: erkennt Anomaliephasen korrekt, aber mit unzureichender Trennschärfe.

Die qualitative Analyse bestätigte, dass die Modelle die „richtigen“ Bereiche zwar erkennen, ihre Scores aber meist unterhalb der festen Schwellen liegen. Erst durch den optimalen Schwellenwert (TH_opt) wurde DS2 korrekt erfasst (F1 = 0.81 beim CAE).

Diskussion

Die vergleichende Betrachtung aller Modelle zeigt, dass kein einzelnes Verfahren in allen Szenarien überlegen ist. Vielmehr ergänzen sich die Modelle in ihren Stärken:

- **DeepAnT (CNN)** → robust und konstant, besonders bei klaren Zustandswechseln.
- **LSTM-AD** → höchstes theoretisches Diskriminierungspotenzial, aber schwankend bei subtilen Mustern.
- **CAE** → sehr präzise bei extremen Abweichungen, jedoch inkonsistent bei schwachen oder gleichförmigen Anomalien.

Model / Dataset		DS1	DS2	DS3	DS4	DS5	DS6	DS7
DeepAnT	F1	0.87	0.41	**0.72**	**0.71**	0.93	**0.87**	**0.98**
	AUC-ROC	(0.97)	(0.80)	(0.67)	(0.80)	(0.99)	(0.92)	(0.88)
LSTM-AD	F1	0.95	0.00	0.48	0.23	0.90	0.03	0.33
	AUC-ROC	(0.99)	(0.44)	(0.98)	(0.92)	(0.99)	(0.47)	(0.98)
CAE	F1	**0.97**	**0.44**	0.43	0.00	**0.96**	0.61	0.78
	AUC-ROC	(0.98)	(0.89)	(0.46)	(0.16)	(0.99)	(0.76)	(0.61)

Abbildung 14 Übersicht der besten F1- und AUC-ROC-Werte über alle Modelle.

Ein zentrales Ergebnis ist der Einfluss der Schwellenwertdefinition: Die standardisierten Schwellen (TH_pq, TH_st) erweisen sich als zu konservativ; erst datensatzspezifische Optimierungen schöpfen das Potenzial der Modelle aus.

Darüber hinaus bestätigte die Untersuchung, dass Vorverarbeitung, Feature-Skalierung und Sequenzlänge maßgeblich die Erkennungsleistung beeinflussen. Die Ergebnisse stützen die Hypothese, dass Modellleistung nicht nur von der Architektur, sondern in gleichem Maße von der Datenaufbereitung abhängt.

Schließlich wurde durch die AUC-ROC-Analyse belegt, dass LSTM-AD die beste theoretische Trennschärfe aufweist, DeepAnT die stabilste Gesamtperformance liefert und der CAE seine Stärken bei stark ausgeprägten Mehrdimensionalanomalien ausspielt.

Fazit und Ausblick

Die vorliegenden Berichtsergebnisse untersuchten den Einsatz von Deep-Learning-Methoden zur Anomalieerkennung in multivariaten automobilen Zeitreihen. Ziel war es, unterschiedliche Architekturen – Convolutional Neural Networks (CNN), Long Short-Term Memory Netze

(LSTM) und Convolutional Autoencoder (CAE) – systematisch zu vergleichen und ihre Leistungsfähigkeit unter realen Testbedingungen zu bewerten.

Aufbauend auf den theoretischen Grundlagen zur Zeitreihenanalyse und den Charakteristika von Anomalien wurden reale Fahrzeugsensordaten herangezogen, um praxisnahe und belastbare Ergebnisse zu erzielen. Die Integration authentischer Fahrzeugdaten – anstelle synthetischer Benchmarks – gewährleistete eine hohe Relevanz für reale Entwicklungs- und Testumgebungen.

Ergebnisse zu 5.6.6 bis 5.6.9:

Entwicklung einer domänenspezifischen Sprache (DSL) zur Testbeschreibung für Infotainment-Displays mit JetBrains MPS (UC2+5)

Zur Automatisierung von Tests für Infotainmentsystem-Steuergeräte in Fahrzeugen wurde eine domänenspezifische Sprache (DSL) mit JetBrains MPS [13] entwickelt, die eine formale und zugleich benutzerfreundliche Beschreibung von Testfällen ermöglicht. Die DSL bietet eine grafische Benutzeroberfläche zur Erstellung von Testszenarien, die zum einen in ausführbaren Code zur Testdurchführung umgewandelt werden kann und zum anderen auch als standardisierte Testspezifikation exportiert werden können. Zum Testen wurden Methoden zur Anbindung verschiedener Kommunikationsschnittstellen wie CAN-Bus, Automotive Ethernet, serielle Schnittstellen, REST-APIs und ADB implementiert. Zur Validierung visueller Inhalte des Infotainmentsystems wurde ein Vision-Language-Modell integriert, während Text-to-Speech- und Speech-to-Text-Modelle die Absicherung von Sprachdialogsystemen ermöglichen. Die Plattform unterstützt zudem die automatische Auswertung von Testergebnissen und generiert Testberichte in kundenspezifischem Format, wodurch eine durchgängige und skalierbare Testpipeline realisiert wird.

Demonstrator Testautomatisierungsumgebung

Im vorliegenden Abschnitt erfolgt eine detaillierte Vorstellung der Testautomatisierungs-plattform, die unter dem Projektnamen "Octoscript" entwickelt wurde. Die Plattform zeichnet sich durch einen als "modularen Automatisierungsbaukasten" konzipierten Ansatz aus und umfasst KI-Features sowie die Einbindung eines Roboterarms. Darüber hinaus wird eine Übersicht über die Rückmeldung der Nutzer sowie ein Vergleich mit klassischem Testing präsentiert.

Octoscript: Entwicklung einer domänenspezifischen Sprache (DSL) zur Testbeschreibung für Infotainment-Displays mit JetBrains MPS

Ein zentraler Bestandteil dieses Projekts war die Entwicklung einer domänenspezifischen Sprache (DSL), um Testfälle für automobile Infotainment-Displays strukturiert, wartbar und automatisierungsfreundlich zu beschreiben. Zum Einsatz kam dabei JetBrains MPS (Meta Programming System). JetBrains MPS ist ein leistungsfähiges Sprachentwicklungswerkzeug, das auf strukturierter Bearbeitung von Code und Sprachvererbung basiert.

Motivation für eine DSL im Infotainment-Testumfeld

Testfälle im Bereich Infotainment sind häufig komplex, wiederkehrend und stark an die jeweilige Fachdomäne gebunden. Benutzerinteraktionen wie Touch-Gesten, Spracheingaben oder Zustandsüberprüfungen lassen sich in allgemeinen Programmiersprachen meist nur umständlich und mit hohem Boilerplate-Anteil formulieren.

Die Verwendung einer spezialisierten DSL ermöglicht eine höhere Abstraktionsebene, die näher an der Sprache der Fachanwender:innen liegt. Daraus ergeben sich mehrere Vorteile:

- **Bessere Verständlichkeit:** Auch nicht-technische Stakeholder können Tests nachvollziehen oder selbst definieren.
- **Fehlerminimierung:** Durch feste Sprachelemente und strukturierte Eingabe sinkt die Fehleranfälligkeit.
- **Automatisierbarkeit:** Die klare Struktur der DSL erlaubt eine einfache Umwandlung in ausführbaren Testcode.
- **Wiederverwendbarkeit:** Wiederkehrende Interaktionsmuster oder UI-Komponenten lassen sich modular abbilden.

Umsetzung mit JetBrains MPS

JetBrains MPS verfolgt einen editorzentrierten Ansatz: Anstatt Text zu parsen, arbeiten Nutzer:innen direkt mit einem abstrakten Syntaxbaum. Dies erlaubt die Definition eigener Sprachkonstrukte, domänenspezifischer Editoren und kontextsensitiver Validierungen.

In der entwickelten DSL wurden zentrale Konzepte des Infotainment-Testings abgebildet, darunter:

- **Bildschirmansichten** (Screens) und **UI-Elemente** (Buttons, Slider, Listen etc.)
- **Benutzeraktionen** (Tippen, Texteingabe, Umschalten von Schaltern, Spracheingaben)
- **Systemreaktionen und Validierungen**
- **Testabläufe** mit Zustandsübergängen und erwarteten Ergebnissen

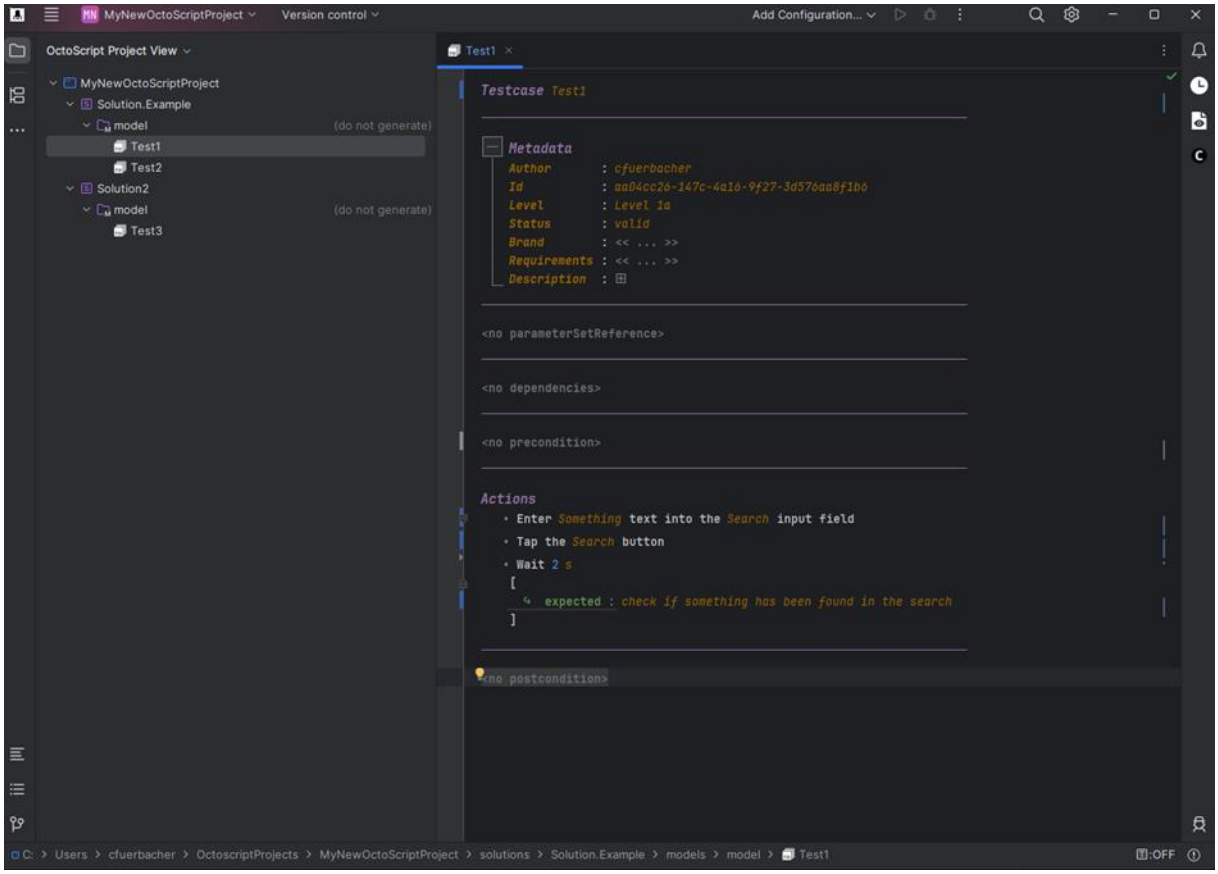

Abbildung 1 Screenshot Octoscript

Die Abbildung zeigt einen Screenshot der Modellierungsumgebung JetBrains MPS (Meta Programming System), die in einem konkreten Anwendungsfall zur Entwicklung von Testfällen für Automotive Infotainment Displays verwendet wird. JetBrains MPS ist eine projektspezifische, sprachbasierte Entwicklungsumgebung, die es erlaubt, domänenspezifische Sprachen (DSLs) zu definieren und zu nutzen.

Die Projektstruktur ist im linken Bereich der Benutzeroberfläche ersichtlich und zeigt eine klare Organisation in Form von Lösungen (Solutions) und Modellen (Models). Im dargestellten Projekt *MyNewOctoScriptProject* sind zwei Lösungen enthalten: Solution.Example und Solution2, jeweils mit einem eigenen model. Diese Modelle beinhalten die eigentlichen Testfälle (Test1, Test2, Test3), welche auf einer domänenspezifischen Sprache basieren, die auf die Anforderungen im Automotive-Testbereich zugeschnitten ist.

Im zentralen Editorfenster ist der Inhalt des Testfalls Test1 dargestellt. Die Testfallstruktur folgt einem typischen MPS-basierten Aufbau, der semantisch reichhaltige Metadaten und handlungsorientierte Testbeschreibungskomponenten integriert. Die relevanten Abschnitte umfassen:

- **Metadata:** Enthält Informationen wie Autor, eindeutige Test-ID, Schwierigkeitslevel, Status und Anforderungen. Diese Metadaten sind essenziell für Rückverfolgbarkeit, Versionierung und Testmanagement.

- **Actions:** Beschreibt die konkreten Schritte zur Ausführung des Tests. In diesem Fall wird ein Texteingabefeld (Search) getestet. Der Ablauf umfasst die Eingabe eines Suchbegriffs, das Auslösen einer Suchaktion und eine darauffolgende Verifikationsphase.
- **Validation/Assertion:** Nach einer Wartezeit von 2 Sekunden erfolgt eine Erwartungsprüfung, die kontrolliert, ob ein Suchergebnis vorhanden ist.

Die verwendeten Schlüsselwörter sind farblich hervorgehoben und typografisch strukturiert, was zur Lesbarkeit und zur intuitiven Navigation im Editor beiträgt.

Im unteren Bereich der IDE sind Pfadangaben zur Datei sichtbar, wodurch eine eindeutige Lokalisierung im Projektverzeichnis gewährleistet wird. Auch weitere IDE-typische Funktionen wie Versionskontrolle, Konfigurationsmanagement (oben rechts), sowie Teststatusanzeigen und Hintergrundprozesse (unten rechts) sind Teil des Gesamtsystems und unterstützen die Entwickler:innen bei der systematischen Testfallmodellierung.

KI-gestützte Testausführung mittels visueller Sprachmodelle (VLM)

Traditionelle Testautomatisierungsmethoden arbeiten häufig mit festen x-y-Koordinaten für Interaktionen auf dem Display. Diese sind jedoch fehleranfällig und schwer wartbar, da sie stark von Layout, Design und Bildschirmauflösung abhängen. Schon kleinere UI-Anpassungen erfordern dann umfangreiche Anpassungen der Testskripte.

Zur Lösung dieses Problems wurde in diesem Projekt ein visuelles Sprachmodell (Visual Language Model, VLM) integriert, das auf Grundlage eines Screenshots die relevanten UI-Elemente lokalisiert – nicht anhand ihrer Position, sondern über deren visuelle Merkmale und semantische Beschreibung.

Beispielhafte Aktionen und deren KI-gestützte Ausführung

In der DSL können Benutzeraktionen deklarativ beschrieben werden, etwa:

- Tap "Settings" Button
- Enter Text "Berlin" into "SearchField"
- Change "Volume" Slider to 70
- Toggle "Bluetooth" Switch to ON

Bei der Ausführung analysiert das VLM den aktuellen Bildschirmzustand und sucht das passende UI-Element auf Basis der Beschreibung (z. B. "Bluetooth Switch"). Die Aktion wird anschließend an der vom Modell bestimmten Position ausgeführt – etwa durch ein simuliertes Tippen.

Rechts zu sehen (Abbildung 15) ist ein einfaches Beispiel eines Tests, der die Mute Optionen in den Einstellungen aktiviert.

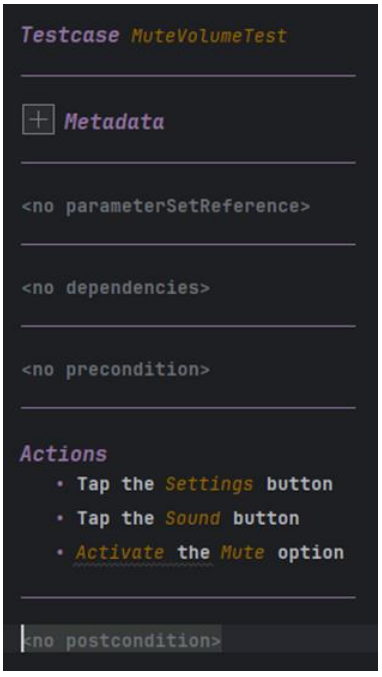

Abbildung 15 Beispieltest in Octoscript

Vorteile des KI-gestützten Ansatzes

- **Unabhängigkeit vom Layout:** Tests bleiben robust gegenüber UI-Änderungen, da die semantische Bedeutung im Fokus steht.
- **Weniger Wartungsaufwand:** Änderungen im Design erfordern keine Neupositionierung der Aktionen.
- **Natürlichere Testbeschreibung:** Die DSL bleibt lesbar und verständlich, ohne technische Details wie Koordinaten.
- **Erhöhte Robustheit:** Das Modell erkennt auch leicht abweichende Darstellungen, etwa durch Icons, Sprachen oder Themes.

Freitextbasierte Testdefinition in natürlicher Sprache

Um die Erstellung von Testfällen weiter zu vereinfachen, wurde zusätzlich ein Freitextmodus eingeführt, der es erlaubt, Testschritte in natürlicher Sprache zu formulieren. Anstelle strukturierter DSL-Befehle können Anwender:innen auch freie Beschreibungen wie:

„Schiebe den Lautstärkeregler ganz nach rechts“

verfassen. Diese werden automatisch interpretiert und in passende DSL-Aktionen übersetzt. Hier ein paar Beispiele, die von der KI verarbeitet werden können.

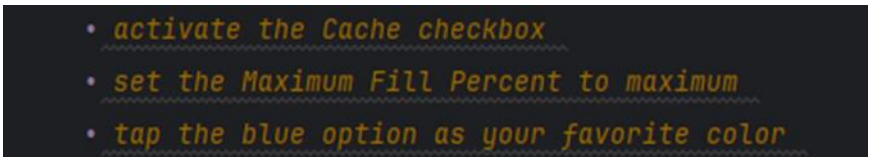

Abbildung 16 Beispiele für Freitext-Aktionen

Vorteile natürlicher Spracheingabe

Viele Nutzer:innen, insbesondere aus nicht-technischen Rollen, bevorzugen eine alltagssprachliche Formulierung von Tests. Der Freitextmodus senkt die Einstiegshürde deutlich und beschleunigt die Testfallerstellung. Formulierungen wie…

- „Tippe auf den Startknopf“
- „Schalte Bluetooth ein“
- „Stelle die Temperatur auf 22 Grad“

…werden automatisch analysiert und in strukturierte Testaktionen überführt.

Technische Umsetzung: Parsing und Kontextanalyse

Die Freitextverarbeitung basiert auf einem Natural Language Processing (NLP)-Modul, das je nach Testkontext (z. B. Setup, Interaktion, Validierung) passende Aktionsmuster erkennt. Dabei werden maschinelles Lernen, regelbasierte Parser und UI-Metadaten kombiniert.

Beispielhafte Umwandlungen:

Freitext	Interpretiert als DSL-Aktion
„Tippe auf den Startknopf“	Tap "StartButton"
„Gib 'Berlin' in das Suchfeld ein“	EnterText "Berlin" into "SearchField"
„Schiebe den Regler ganz nach rechts“	ChangeSliderValue "Regler" to 100 *(implizit)*

„Der Modus muss auf 'Eco' stehen“	ExpectValue "DrivingMode" == "Eco"

Tabelle 4 Natural Language Processing in Octoscript

Auch qualitative Angaben wie „ganz nach rechts“, „mittig“ oder „minimal“ werden erkannt und durch entsprechende Zielwerte ersetzt basierend auf UI-Definitionen.

Validierungen in Freitext

Neben Aktionen können auch Validierungsschritte frei formuliert werden:

- „Die Navigation sollte gestartet sein“
- „Der Bildschirm zeigt die Karte“
- „Die Akzentfarbe ist nun Dunkelblau“

Das System erkennt automatisch, dass es sich um eine Validierung handelt, und erzeugt den entsprechenden Ausdruck. Hier Beispiele der Validierung.

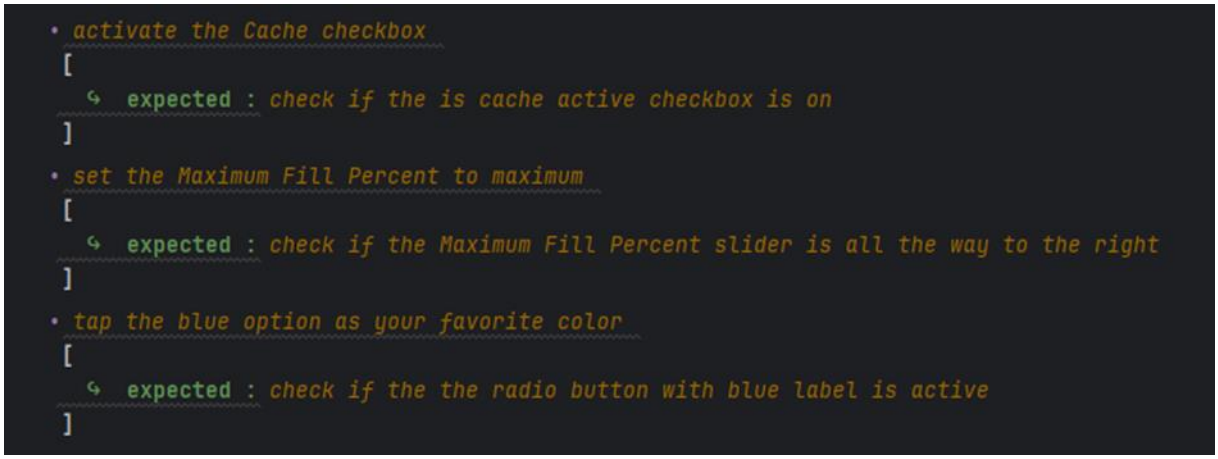

Abbildung 17 Validierungsmöglichkeiten in Octoscript

Vorteile der domänenspezifischen Sprache (DSL) gegenüber dem bisherigen Vorgehen

In der Vergangenheit basierte die Testfallerstellung im Infotainment-Bereich auf einer rein manuellen Vorgehensweise, bei der Testfälle von einem dedizierten Team in Excel-Tabellen gepflegt wurden. Diese Tabellen enthielten oft informelle oder uneinheitliche Beschreibungen von Testschritten, die anschließend von Tester:innen manuell am Prüfstand ausgeführt wurden. Nur ein Teil der Tests wurde nachträglich durch ein Entwicklerteam automatisiert – ein Prozess, der nicht nur zeitintensiv war, sondern auch personell aufwendig. Die Durchführung erforderte mehrere Übergaben zwischen Testdesign, Testdurchführung und Automatisierung, wodurch sowohl die Durchlaufzeit als auch das Fehlerrisiko stiegen.

Mit der Einführung einer domänenspezifischen Sprache (DSL) auf Basis von JetBrains MPS wurde dieser Prozess grundlegend verbessert.

Vergleich der Effizienz

Kriterium	Altes Vorgehen (Excel + manuell)	Neues Vorgehen (MPS + DSL)
Erstellungsaufwand	Hoch – durch manuelle Pflege und fehlende Struktur	Niedrig – durch strukturierte Eingabe in der DSL
Fehleranfälligkeit	Hoch – Freitextform und fehlende Validierung	Niedrig – formale Sprachelemente mit Syntaxprüfung

Verständlichkeit	Eingeschränkt – nur für technische Expert:innen	Hoch – auch für Fachabteilungen zugänglich
Automatisierung	Teilweise, hoher Entwicklungsaufwand	Vollständig – durch automatisierte Generierung aus DSL
Wiederverwendbarkeit	Gering – Copy-Paste zwischen Excel-Zellen	Hoch – modulare Bausteine und Templates in der DSL
Personalbedarf	Hoch – mehrere Teams involviert	Reduziert – integrierter End-to-End-Prozess

Tabelle 5 Effizienzverbesserung durch Octoscript

Qualität und Wartbarkeit

Durch den Einsatz von Vision-Language-Modellen (VLM) zur Objekterkennung entfällt die Notwendigkeit, UI-Elemente manuell über Koordinaten oder technische Selektoren anzusprechen. Testfälle bleiben dadurch trotz UI-Änderungen stabil und wartungsarm. Die Kombination aus DSL und VLM macht Testfälle nicht nur robuster, sondern auch für fachfremde Personen verständlich und bearbeitbar, was die Lesbarkeit, Qualität und Wartung erheblich vereinfacht.

Zusammenfassung: Flexibilität durch hybride Testdefinition

Die Kombination aus strukturierter DSL, KI-gestützter UI-Erkennung und natürlicher Spracheingabe schafft eine flexible und leistungsfähige Testumgebung, die sowohl technischen als auch nicht-technischen Nutzergruppen gerecht wird. Testfälle lassen sich:

- verständlich und intuitiv formulieren,
- robust gegenüber Designänderungen ausführen,
- und direkt in automatisierte Testsysteme überführen.

Durch diesen hybriden Ansatz wird eine moderne, skalierbare Teststrategie ermöglicht, die sich ideal an die Anforderungen komplexer Infotainment-Systeme anpasst.

Demonstrator zur Absicherung von KI-Sprachassistenten (UC2+5)

Zur Absicherung von KI-basierten Sprachassistenten im Fahrzeug wurde ein prototypischer Demonstrator entwickelt. Der Demonstrator verwendet Speech-to-Text-(STT)-Modellefür die Spracheingabe und Text-to-Speech-(TTS)-Modellen zur Erfassung der Sprachausgabe. Ein Large Language Model (LLM) wird eingesetzt, um die semantische Korrektheit und Konsistenz der Antworten des Sprachassistenten zu bewerten. Zusätzlich werden Debug-Nachrichten des Sprachdialogsystems kontinuierlich protokolliert, um eine detaillierte Fehleranalyse zu ermöglichen. Abschließend erfolgt die automatische Generierung eines Testberichts im kundenspezifischen Format, wodurch eine transparente und reproduzierbare Qualitätssicherung gewährleistet wird.

Ergebnisse:

Demonstrator: Absicherung von KI-Sprachassistenten

Mit der zunehmenden Integration von Sprachdialogsystemen (SDS) in moderne Fahrzeuge [14] wächst der Bedarf an effizienten, skalierbaren und reproduzierbaren Testverfahren. Während Sprachassistenten früher klar abgegrenzte Befehle auswerteten und sich dadurch

relativ leicht testen ließen, erweitern aktuelle Systeme ihr Funktionsspektrum kontinuierlich – unter anderem durch generative KI, mehrsprachige Interaktion und kontextbezogene Informationsabfragen. Insbesondere durch die Einführung von ChatGPT in Audi-Modellen ab dem Modelljahr 2021 ergeben sich neuartige Herausforderungen für die Validierung: Das System muss zuverlässig erkennen, ob eine Nutzeranfrage die Steuerung einer Fahrzeugfunktion, eine Navigationsaufgabe, eine Wissensfrage oder eine Interaktion mit dem generativen KI-Modell betrifft. Gleichzeitig dürfen bestehende Funktionen nicht durch neue Features beeinträchtigt werden.

Die klassische manuelle Testdurchführung ist diesen Anforderungen kaum gewachsen. Die Vielzahl möglicher Eingaben, die starke Varianz natürlicher Sprache, die Abhängigkeit von Sprechermerkmalen sowie die Komplexität der Systemreaktionen führen zu hohem Zeitaufwand und unzureichender Reproduzierbarkeit. Insbesondere die Sprachvielfalt stellt ein Problem dar: Für umfangreiche Testreihen werden native Sprecher benötigt, was Testzeiten verlängert und Kosten erhöht. Die Automatisierung bietet daher ein erhebliches Potenzial, diese Aufgaben robuster und effizienter durchzuführen.

Um diese Herausforderungen zu adressieren, wurde ein prototypischer Automatisierungsdemonstrator entwickelt, der sämtliche Schritte eines SDS-Testfalls softwareseitig abbildet. Die Lösung umfasst die automatisierte Sprachausgabe über **Text-to-Speech** (TTS), die Erkennung der Fahrzeugantwort mittels **Speech-to-Text** (STT), die Analyse der **Debug-Ausgaben** des Infotainmentsystems sowie die Auswertung der Systemantwort durch ein **Large Language Model** (LLM). Ein zentrales Ziel bestand darin, die vollständige Interaktion zwischen Testsystem und Fahrzeug so weit zu automatisieren, dass keine manuelle Beteiligung mehr notwendig ist und Tests vollständig reproduzierbar werden.

Das Gesamtsystem basiert auf modularen Komponenten: **Whisper** [15] wird zur Transkription von Antworten eingesetzt, während **Piper** [16] als TTS-System zur Erzeugung der Testeingaben dient. Ergänzend wurde eine automatische Übersetzungsschicht integriert, um verschiedene Sprachen ohne native Sprecher testen zu können. Die Auswertung der Systemantwort erfolgt anhand eines LLMs, das für jeden Testfall beurteilt, ob die Anfrage korrekt verstanden und die richtige Domäne ausgewählt wurde. Dieser Aufbau ermöglicht einen vollständig automatisierten Durchlauf eines SDS-Testfalls inklusive Aktivierung des Sprachassistenten über das Wake-Word, Generierung der Nutzereingabe, Analyse der Fahrzeugreaktion und Bewertung der Antwortqualität.

Ablauf eines SDS-Testfalls

Der automatisierte Testablauf verläuft dabei in mehreren Schritten: Zunächst wird der Testfall, der in Textform vorliegt, durch das TTS-System in gesprochene Sprache umgewandelt und über einen Lautsprecher an das Infotainmentsystem des Fahrzeugs übermittelt. Die Fahrzeugantwort wird anschließend akustisch erfasst und durch das STT-Modell transkribiert. Parallel dazu werden sämtliche Debug-Nachrichten des Fahrzeugsystems aufgezeichnet, um eine tiefergehende Analyse der Domänenzuordnung zu ermöglichen. Anschließend vergleicht das LLM die erwartete und die tatsächlich erkannte Antwort. Dabei bewertet das Modell sowohl die inhaltliche Korrektheit als auch die Domänenzuordnung und berücksichtigt, ob das Fahrzeug die Anfrage an ChatGPT weitergeleitet oder intern beantwortet hat.

Der Ablauf eines Testfalls sieht dabei wie folgt aus:

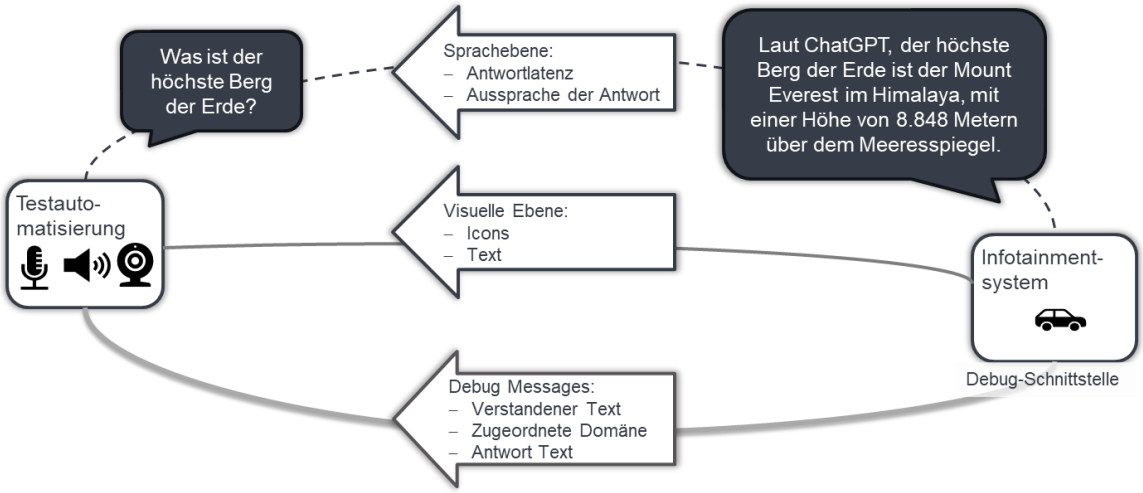

Abbildung 18 SDS Testablauf

Vergleich: Manuelles vs. automatisiertes SDS-Testing

Die Gegenüberstellung manueller und automatisierter SDS-Tests verdeutlicht das erhebliche Effizienzpotenzial. Ein manuelles Testen von 50 Testfällen dauert erfahrungsgemäß rund acht Stunden und erfordert zudem mehrere Tester, um die verschiedenen Zielsprachen abzudecken. Zusätzlich müssen Debug-Traces manuell gesammelt und analysiert werden, und viele Tests müssen aufgrund menschlicher Unsicherheiten wiederholt werden. Der automatisierte Ansatz hingegen kann dieselbe Testmenge in etwa 30 Minuten durchführen. Selbst bei vierfacher Wiederholung zur Erhöhung der Robustheit liegt die Dauer bei lediglich zwei Stunden. Betrachtet man die Anzahl der produzierten Ergebnisse, entsteht ein Vielfaches der Informationsmenge im Vergleich zu manuellen Tests.

Tabelle 6 Vergleich manuelles und automatisiertes SDS Testing

Art des Tests	Gesamtdauer	Reports erstellt
manuell	8 Stunden	1
automatisiert	2 Stunden	4

Kosten-Nutzen-Analyse

Auch die wirtschaftliche Perspektive wurde untersucht. Die initialen Entwicklungskosten des Automatisierungssystems liegen bei etwa 12.000 Euro. Aufgrund der hohen Effizienz und der reduzierten Testaufwände amortisiert sich die Investition jedoch bereits nach wenigen Testaufträgen. Besonders bei mehrsprachigen SDS-Validierungen zeigt sich, dass die Automatisierung einen deutlichen Kostenvorteil bietet. Abbildung 19 verdeutlicht den Break-even-Point anhand beispielhaft angenommener Kosten und zeigt die steigende Effizienz mit zunehmender Testanzahl.

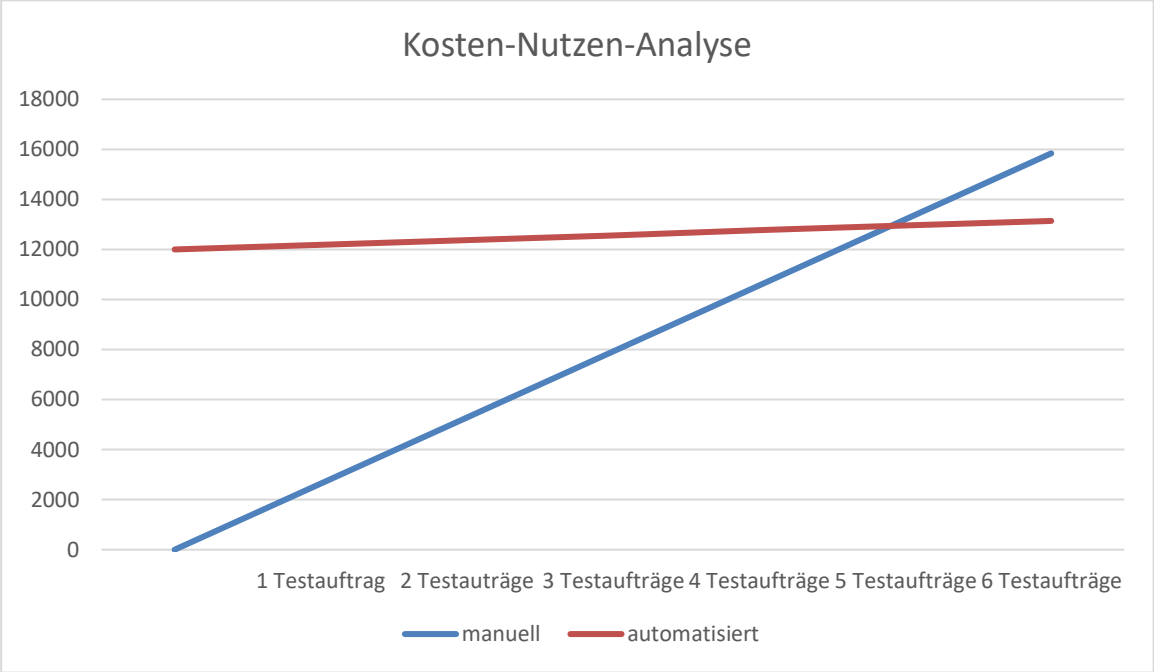

Abbildung 19 Kosten-Nutzen-Analyse SDS Automatisierung

Demonstration und Vergleich von Voice Agents zum Testen von Sprachdialogsystemen (UC2+5)

Die Entwicklung von Voice Agents zur Validierung von Sprachdialogsystemen könnte erhebliche Vorteile gegenüber rein manuellen oder skriptbasierten Testansätzen bieten. Klassische Testautomatisierungsmethoden stoßen beim Testen von Sprachdialogsystemen mitlerweile an Grenzen, insbesondere bei der Simulation realistischer Gesprächsverläufe mit Rückfragen und variabler Syntax. Voice Agents würden eine automatisierte, reproduzierbare und skalierbare Durchführung solcher Tests ermöglichen, einschließlich mehrstufiger Dialoge und dynamischer Abweichungen vom erwarteten Ablauf. Im Rahmen der Demonstration wurden zwei unterschiedliche Voice-Agent-Ansätze entwickelt und in einer Testumgebung erprobt, um deren Leistungsfähigkeit bei komplexen Interaktionsszenarien zu bewerten.

Voice Agents zur automatisierten Interaktion

Zur Realisierung vollständig autonomer Interaktionen wurden zwei Agenten entwickelt, die in der Lage sind, SDS-Systeme selbstständig zu testen. Der SDS State-Machine-Agent folgt einem State Machine Ansatz und reagiert abhängig vom Dialogzustand mit einem vordefinierten Ablauf. Der zweite Ansatz, der Conversational AI Agent, nutzt ein LLM für frei formulierte Echtzeitreaktionen. Dadurch entsteht ein flexibles Testsystem, das in der Lage ist, dynamische, nicht vorhersehbare Dialogabläufe zu simulieren.

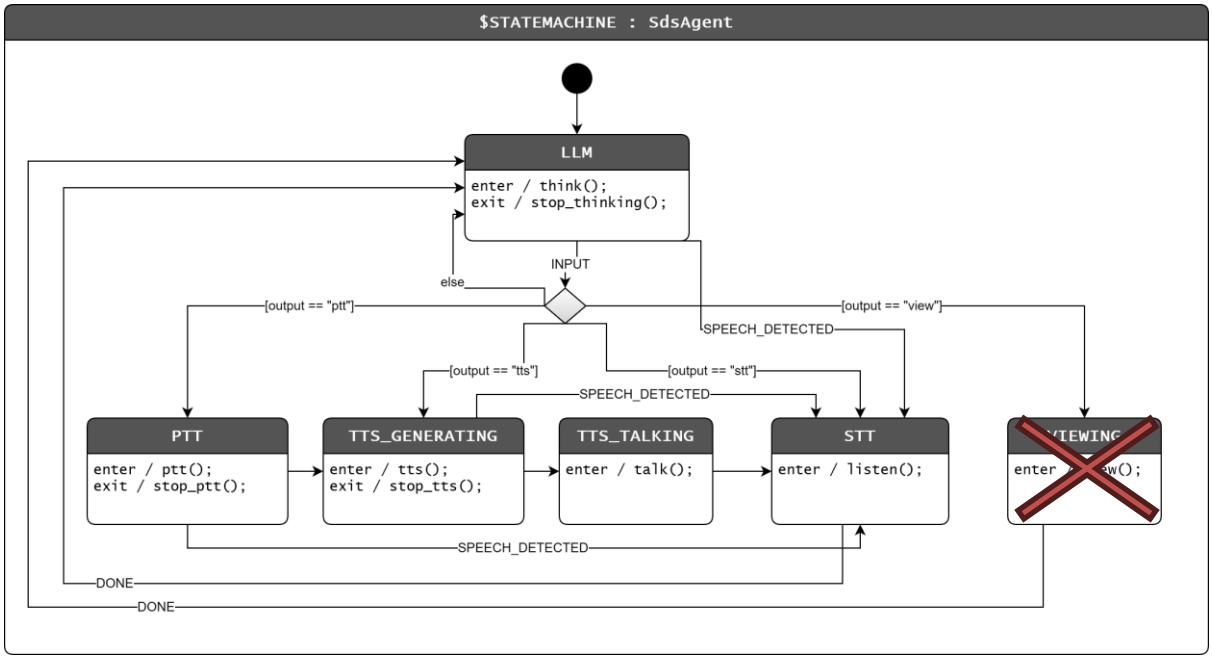

Abbildung 20 SDS State Machine Agent

Zum Vergleich der Agenten wurden zwei standardisierte Testfälle mit mehreren Testschritten entworfen. Der erste Testfall bestand darin, einen Telefonanruf zu einem bestimmten Kontakt zu starten. Da der Kontakt über mehrere Telefonnummern verfügt, musste diese Aufgabe in mehreren Schritten ausgeführt werden. Der zweite Testfall bestand darin, nach einem Restaurant zu suchen und die Navigation zu starten. Dabei hatte der Agent außerdem die Möglichkeit, das Restaurant genauer zu spezifizieren. Die Gegenüberstellung der beiden Agenten ergibt ein differenziertes Bild: Während der Conversational Agent eine höhere Erfolgsquote aufweist und natürlicher interagiert, zeigte der State-Machine-Agent im Testfall „Restaurant" eine geringere Varianz. Die Kombination beider Modelle könnte daher eine besonders vielversprechende Strategie für zukünftige Testumgebungen sein.

telephone, conversational

Berechnung	Wert
Mittelwert	24,37
Standardabweichung	0,56
Median	24,28
Spannweite	2,33
Korrekt	100%

telephone, StateMachine

Berechnung	Wert
Mittelwert	35,80
Standardabweichung	23,78
Median	27,30
Spannweite	143,15
Korrekt	90%

restaurant, conversational

Berechnung	Wert
Mittelwert	69,25
Standardabweichung	113,82
Median	34,92
Spannweite	630,16
Korrekt	84%

restaurant, StateMachine

Berechnung	Wert
Mittelwert	46,05
Standardabweichung	14,92
Median	40,96
Spannweite	122,14
Korrekt	76%

Tabelle 7 Vergleich der Ergebnisse der SDS Agenten

Fazit

Voice Agents haben das Potenzial, die Entwicklung und das Testen von Sprachdialogsystemen grundlegend zu verändern. Durch gezieltes Prompting und die Integration leistungsfähiger Tools können sie komplexe Testabläufe inklusive mehrstufiger Interaktionen und in verschiedenen Sprachen automatisiert durchführen. Dies ermöglicht eine erhebliche Zeitersparnis und eine höhere Testabdeckung. Zudem lassen sich technische Parameter wie die Systemlatenz präzise erfassen und mit definierten Schwellenwerten vergleichen.

Allerdings stoßen Voice Agents an ihre Grenzen, wenn es um die subjektive Wahrnehmung der Nutzer geht. Aspekte wie die Natürlichkeit der Sprache, die emotionale Wirkung der Stimme oder die Qualität des Gesprächsflusses können nur schwer maschinell bewertet werden. Für ein ganzheitliches Qualitätsurteil bleibt die menschliche Perspektive daher unverzichtbar, insbesondere bei der Frage, ob sich die Interaktion wirklich intuitiv und angenehm anfühlt.

4 Verwertung und voraussichtlicher Nutzen

Durch die Möglichkeiten umfassende Forschung zur Automatisierung von Testschritten zu betreiben hatte ASPA die Chance neben dem regulären Projektgeschäft, das zum Großteil auf manuellem Testen basiert, Kapazitäten in die Erforschung von Testautomatisierungen zu investieren.

Automatisierung führt dazu, dass mit der gleichen Menge an Personal sowohl quantitativ als auch qualitativ besser die Integration von Steuergeräten ins Bordnetz abgesichert werden kann. OEMs erhalten somit eine bessere und umfänglichere Absicherung und Integration von Bordnetzkomponenten und dem Bordnetzgesamtsystem. Dies bietet uns die Möglichkeit uns qualitativ und vor allem preislich von der Konkurrenz abzuheben.

11-ASPA

Ausgangslage

ACONEXT Sparks GmbH (ehemals SPARKS Solutions GmbH) ist ein Dienstleister in der Automotivbranche, der Testgewerke für namhafte deutschen Automobilhersteller bearbeitet. Häufiger ergaben sich für ASPA bereits Entwicklungsbeauftragungen für Testautomatisierungen und Simulationen. Daher versprach sich ASPA, dass positive Ergebnisse aus dem Projekt unmittelbar nach Projektende gewinnbringend eingesetzt werden könnten.

Es wurde zudem davon ausgegangen, dass in Zukunft das Feld der Absicherung für das Bordnetz durch den Technologiefortschritt und Trends wie autonomes, elektrisches und vernetztes Fahren immer umfangreicher werden würde, weswegen mit größeren Beauftragungen in der Absicherung zu rechnen wäre. Mit einem Modularen Automatisierungsbaukasten (MAB) könnte ASPA effizienter und flexibler ein größeres Arbeitspensum abarbeiten. Vor allem die Automatisierung im gesamten Testprozess und flexibler Absicherung von Mensch-Maschine-Schnittstellen könnten ASPA deutlich von Mitbewerbern in der Absicherung abheben. Durch Testautomatisierung sind Testaufträge besser skalierbar. Daher versprach sich ASPA durch die Innovation einen voll- oder hochautomatisierten Testbaukasten im Portfolio zu haben, dass mit der gleichen Menge an Personal sowohl quantitativ als auch qualitativ besser die Integration von Steuergeräten ins Bordnetz abgesichert werden könnte. OEMs würden somit eine bessere und umfänglichere Absicherung und Integration von Bordnetzkomponenten und dem Bordnetzgesamtsystem erhalten.

Von den Forschungsergebnissen und späteren, abgeleiteten Entwicklungen würden primär Kunden aus der Automobilbranche profitieren. Durch die Modularität der Entwicklungen wäre es jedoch auch möglich in anderen Branchen (z.B. Komfortelektronik, Mensch-Maschine-Schnittstellen, etc.) Ergebnisse zu verwerten und neue Kunden zu akquirieren. Bei einer erfolgreichen Verwertung der Forschungsergebnisse nach Projektende wären wirtschaftliche Erfolge durch den innovativen MAB und unser bestehendes Kundennetzwerk zu erwarten.

Zu Beginn des Projektes wurde angenommen, dass die wirtschaftlichen Risiken bei positiven Forschungsergebnissen niedrig lägen, da davon ausgegangen wurden, dass Alternativen zur Automatisierung in der Absicherung in Zukunft nicht rentabel sein würden. Es zeichnet sich

jetzt jedoch der Trend ab, Absicherungsaufträge in Niedriglohnländer in Osteuropa oder Sudostasien auszulagern. Aber auch hier sehen wir gute Verwertungsaussichten, da unser einfach zu bedienender MAB auch von weniger erfahrenen Testern zur Testautomatisierung eingesetzt werden kann.

Ergebnisse in KI4BoardNet

Das Hauptergebnis von ASPA ist die Entwicklung des geplanten Modularen Automatisierungsbaukasten (MAB), im Weiteren genannt Octoscript. Octoscript basiert auf einer Domänenspezifische Sprache (DSL). Eine DSL kann verglichen werden mit einer Programmiersprache, die speziell an ein Fachgebiet angepasst wurde. Sie vereinfacht die Kommunikation zwischen Steuergerätetestern und Softwareentwicklern, indem sie Testern erlaubt Testfälle in ihrer gewohnten Sprache zu verfassen und die Tätigkeiten der Entwickler auf die Entwicklung neuer Funktionalitäten beschränkt. Octoscript ist modular aufgebaut und kann sehr einfach um neue Features erweitert werden.

Ein Einsatzgebiet von Octoscript ist das Testen der Buskommunikation von Steuergeräten. Hierfür wurde eine Anbindung an das Tool CANoe von Vector geschaffen. Geplant ist später auch wieder eigene Hardwareinterfaces und welche von anderen Herstellern, wie z.B. PEAK-System, zu unterstützen.

Ein weiteres wichtiges Einsatzgebiet von Octoscript ist das Abtesten von visuellen Schnittstellen, wie dem Infotainment-Display oder dem Kombi-Display im Fahrzeug. Auf diese graphischen User Interfaces (GUI) kann über Octoscript auf verschiedene Weisen zugegriffen werden: Android ADB Schnittstelle (wenn vorhanden), digitale Bildgrabber oder Kameras. Zur Analyse der Bilder kommen verschiedene Techniken zum Einsatz: Template Matching Algorithmen, Optical Character Recognition (OCR), YOLO Objekterkennungsmodelle und Vision Language Modelle (VLM).

Zum Abtesten der visuellen Schnittstellen gehört auch das Bedienen der User Interfaces, um in bestimmte Menüs zu gelangen, oder Eingaben zu machen. Hierfür haben wir eine innovative Lösung entwickelt: Die Testschritte können in natürlicher Sprache verfasst werden. Sie werden dann zusammen mit einem Screenshot des aktuellen Menüs von einem VLM interpretiert, welches die Nutzerintention extrahiert und eine Aktion (z.B. Klicken oder Swipen) zurückliefet.

Zum Zwecke der Bedienung von GUIs wurde auch der Einsatz von KI-Agenten erprobt. Wir wollten KI-Agenten einsetzen, um auf unvorhergesehene Ereignisse während des Testens reagieren zu können oder unvollständige Testschritte in natürlicher Sprache zu erweitern. Es konnte bewiesen werden, dass das Grundprinzip funktioniert. Ein prototypischer Demonstrator wurde vorgestellt. Dieser setzte auf einen Roboterarm, der End-to-End Tests am Infotainmentsystem durchführen kann. Die Möglichkeit zur End-to-End Absicherung mit Roboterarm ist zusätzlich auch über Octoscript gegeben. Über das Projekt hinweg wurde der KI-Agent durch neue KI-Technologien immer weiter verbessert. Leider konnte nie die gewünschte Genauigkeit, die zum produktiven Einsatz benötigt werden würde, erreicht werden. Mit dem allgemeinen technischen Fortschritt und den immer besser werdenden KI-

Modellen, könnte dies jedoch in absehbarer Zeit der Fall sein und wird daher weiterverfolgt werden.

Zum Testen von auditiven Schnittstellen wurden automatische Spracherkennung und Sprachsynthese in Octoscript integriert. Für einen speziellen Anwendungsfall wurde ein separates Tool zu Octoscript erprobt. Hierbei ging es um die Absicherung von KI-Sprachassistenten im Fahrzeug. Dank der Modularität von Octoscript, kann dieses spezielle Tool auf dasselbe Backend zur Spracherkennung und Sprachsynthese aufbauen wie Octoscript, was den Entwicklungsaufwand erheblich reduzierte. Mit diesem Tool konnte gezeigt werden, dass sich der Testaufwand um mindestens 75% reduzieren lässt.

Es wurde außerdem erforscht, wie Datenanomalien in Busdaten mit KI erkannt werden können. Dafür wurden verschieden Methoden evaluiert und verschiedene Modelle trainiert. Es konnte bewiesen werden, dass die Erkennung von Datenanomalien in Fahrzeugbusdaten mit verschiedenen Verfahren möglich ist. Diese Modelle müssen auf jeden spezifischen Anwendungsfall angepasst werden. SPARKS hatte sich durch die Teilnahme an KI4BoardNet das nötige Wissen angeeignet, um in Zukunft neue Modelle trainieren zu können. Das Wissen und die Mitarbeiter gingen an ACONEXT Sparks GmbH über, so dass sichergestellt ist, dass die Ergebnisse weiterverwendet werden können.

Nutzen und Verwertbarkeit

In einer Evaluierungsphase konnte gezeigt werden, dass Octoscript uns eine große Produktivitätssteigerung gegenüber dem vorherigen Testing- und Automatisierungsprozess liefert. Der vorherige Prozess sah vor, dass Tester (Domänenexperten) zuerst Testfälle ausdefinieren (Excel-Tabellen) und diese dann manuell auszuführen. Nach der Erstellung der Testfälle erhielten die Softwareentwickler die Dokumente, um die Testfälle mit Hilfe eines auf Vector CANoe basierenden Frameworks zu automatisieren. Da die Softwareentwickler um weniger Domänenwissen verfügen als die Tester, kam es während des Automatisierungsprozesses zu vielen sehr Rückfragen, was den Prozess stark verlangsamte. Vieles an Wissen wurde

Der neue Prozess sieht vor, dass die Softwareentwickler nur die Domänenspezifische Sprache und die Funktionen dahinter bereitstellen und die Tester diese verwenden, um ihre Testfälle zu definieren und zu automatisieren. Wenn vom Kunden gewünscht, ist es möglich die Testfälle als Excel-Tabelle aus Octoscript zu exportieren.

Octoscript befindet sich auf dem Technischen Reifegrad Level 6 und soll ab Januar 2026 erstmalig bei einem Kundenprojekt produktiv eingesetzt werden.

Was den Einsatz von automatischer Spracherkennung und Sprachsynthese angeht, so konnte in einem Versuch eine Zeitersparnis von mindestens 75% festgestellt werden. Diese kommt zustande durch die automatische Dokumentation und Berichtserstellung durch die KI. Weiterhin können Tests durch diese Technologien in viel mehr Sprachen durchgeführt werden, was die Testabdeckung erhöht. Ein Einsatz dieser Automatisierung ist für eine Ausschreibung im Jahr 2026 angedacht.

Ab Beginn 2026 ist außerdem eine Kooperation mit einer Schwesterfirma von ACONEXT Sparks GmbH geplant, bei der KI-Modelle zur Erkennung von Datenanomalien in Fahrzeugbusdaten trainiert und eingesetzt werden sollen.

Vor diesem Hintergrund besteht bei der ACONEXT Sparks GmbH die Möglichkeit, in dieser Arbeit gewonnenen Erkenntnisse weiterzuführen und praxisorientiert in bestehende Testumgebungen zu integrieren. Insbesondere die Implementierung einer skalierbaren Pipeline zur automatisierten Anomalieerkennung auf Bus- oder Sensordaten bietet erhebliches Potenzial, um die Effizienz von Entwicklungs- und Testprozessen nachhaltig zu steigern. Damit bildet die Arbeit nicht nur einen wissenschaftlichen Beitrag, sondern zugleich eine praktische Grundlage für zukünftige industrielle Anwendungen und die Fortführung des Forschungsansatzes im Rahmen realer Kundenprojekte.

5 Veröffentlichungen im Projektzeitraum

Im Bearbeitungszeitraum vom 01.09.2025 – 30.11.2025 sind keine Veröffentlichungen entstanden.

6 Quellen

[1] Ultralytics, „Explore YOLOv8,“ [Online]. Available: https://yolov8.com/. [Zugriff am 05 02 2024].

[2] PaddlePaddle, „PaddleOCR,“ Github, [Online]. Available: https://github.com/PaddlePaddle/PaddleOCR. [Zugriff am 05 02 2024].

[3] Universal Robots, „Über Universal Robots,“ [Online]. Available: https://www.universal-robots.com/de/uber-universal-robots/. [Zugriff am 08 02 2024].

[4] LangChain Inc., „LangChain overview,“ [Online]. Available: https://docs.langchain.com/oss/python/langchain/overview. [Zugriff am 01 12 2025].

[5] J. Z. D. Y. N. D. I. S. K. N. Y. C. Shunyu Yao, „ReAct: Synergizing Reasoning and Acting in Language Models,“ 2024.

[6] LangChain Inc., „LangGraph overview,“ [Online]. Available: https://docs.langchain.com/oss/python/langgraph/overview. [Zugriff am 01 12 2025].

[7] D. J. S. u. L. J. Patricia Kelbert, „Large action models (LAMs), tool learning, function calling and Agents,“ 19 03 2024. [Online]. Available: https://www.iese.fraunhofer.de/blog/large-action-models-multi-agents/. [Zugriff am 01 12 2025].

[8] W. X. Y. L. Z. H. Y. L. R. K.-W. L. E.-P. L. Lei Wang, „Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models,“ 2023.

[9] Qwen, „Qwen2.5-VL-32B-Instruct,“ [Online]. Available: https://huggingface.co/Qwen/Qwen2.5-VL-32B-Instruct. [Zugriff am 01 12 2025].

[10] M. M. e. al., „DeepAnT: A Deep Learning Approach for Unsupervised Anomaly,“ *IEEE Access 7,* pp. 1991-2005, 2019.

[11] P. M. e. al., „Long Short Term Memory Networks for Anomaly Detection,“ in *The European Symposium on Artificial Neural Networks*, 2015.

[12] F. A. a. H. M. Narges Ehsani, „Convolutional autoencoder anomaly detection and classification based on distribution PMU measurements,“ *IET Generation, Transmission & Distribution 16.14,* pp. 2816-2828, 2022.

[13] JetBrains, „Domänenspezifische Sprachen,“ [Online]. Available: https://www.jetbrains.com/de-de/mps/concepts/domain-specific-languages/. [Zugriff am 01 12 2025].

[14] Volkswagen, „ChatGPT ist ab sofort in vielen Volkswagen Modellen verfügbar,“ 21 06 2024. [Online]. Available: https://www.volkswagen-newsroom.com/de/pressemitteilungen/chatgpt-ist-ab-sofort-in-vielen-volkswagen-modellen-verfuegbar-18459.

[15] OpenAI, „Wir stellen vor: Whisper,“ 21 09 2022. [Online]. Available: https://openai.com/de-DE/index/whisper/. [Zugriff am 01 12 2025].

[16] OHF-Voice, „Piper,“ [Online]. Available: https://github.com/OHF-Voice/piper1-gpl. [Zugriff am 01 12 2025].