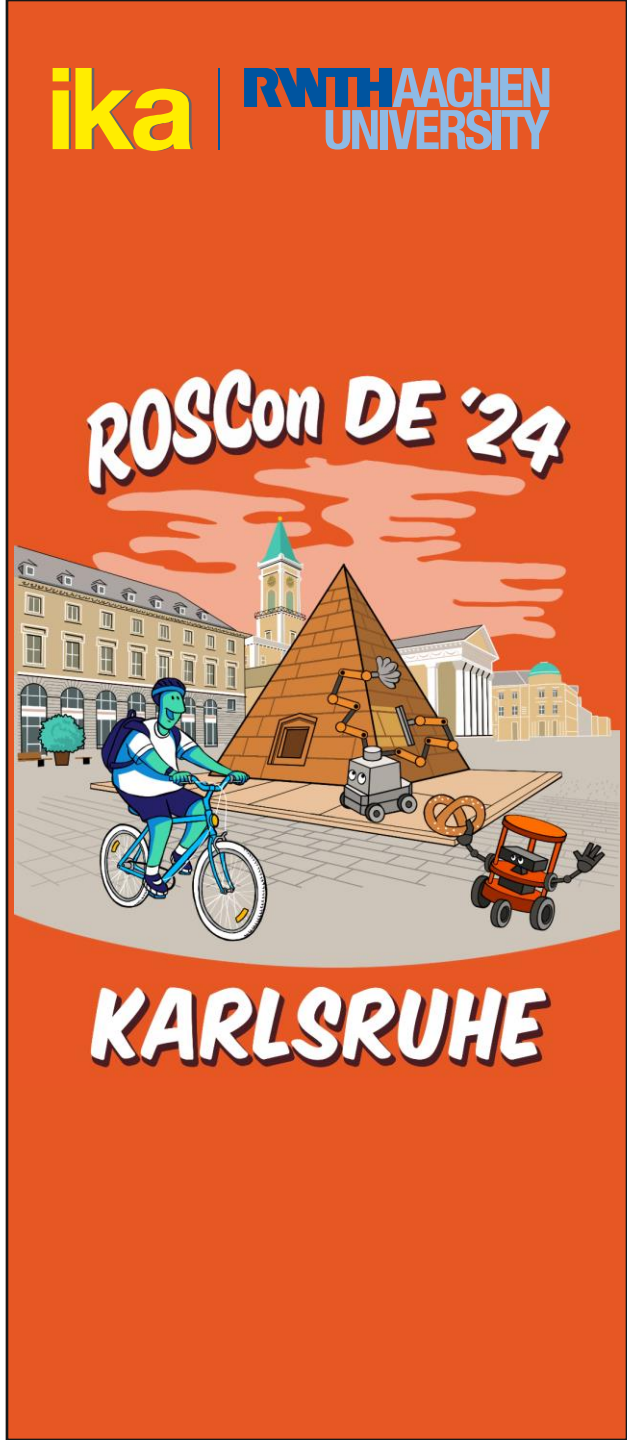
ROSCon DE

23. – 24. November 2024

Karlsruhe, Deutschland

CARLOS

An Open, Modular, and Scalable Simulation Framework for the Development and Testing of Software for C-ITS



Karlsruhe, 03.12.2024

Silas Damaschke

Institut für Kraftfahrzeuge (ika), RWTH Aachen University

Institut für Kraftfahrzeuge, RWTH Aachen University
Führendes Institut für Bildung und Forschung in der Fahrzeugtechnik

~ 90
Wissenschaftliche Mitarbeitende

~ 250
Mitarbeitende in Verwaltung, Werkstätten & Studierende Mitarbeitende

~ 10m €
Jährliches Projektvolumen

LEHRE

ÖFFENTLICH GEFÖRDERT
FORSCHUNG

AUFTRAGS-FORSCHUNG
INDUSTRIE

- Fahrzeugkonzepte & HMI
- Fahrdynamik & Akustik
- Energiemanagement & Antriebe
- Fahrzeugintelligenz & Automatisiertes Fahren
- Verkehrspsychologie & Akzeptanz

- Funktionsentwicklung Automatisiertes Fahren
- Absicherung & Wirksamkeit
- Vernetzung
- Human-Centered Driving Simulation

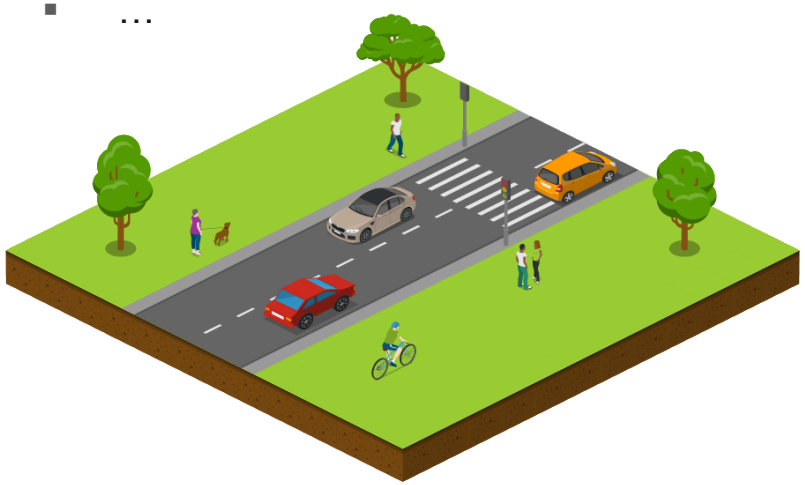
Motivation

Herausforderungen bei der Entwicklung von automatisierten Fahrfunktionen

Komplexität

Interaktion mit

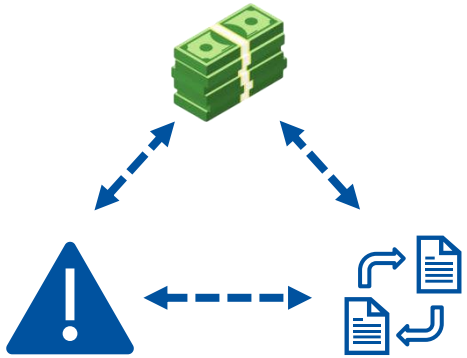
- Fahrzeugen
- Fußgängern
- Infrastruktur
- …



Kosten

Reale Tests

- Teuer und aufwändig
- Riskant
- Schwer reproduzierbar



Skalierbarkeit

Software Prototyping & Testing

vs.

Hohe Sicherheitsanforderungen im realen Straßenverkehr

- **Skalierbarkeit** durch enorme **Mengen an gefahrenen Kilometer**?
  - ➢ Weiterhin **keine Reproduzierbarkeit**
  - ➢ **Nicht parallelisierbar**

Lösungsansatz: Validierung durch Fahrsimulation

Zusammenfassung: Reale Tests

- **Automatisierung** und **Skalierung** nicht möglich
- Nicht **reproduzierbare** oder **vordefinierbare** Testbedingungen
- Hohe **zeitliche** und **finanzielle Investments** notwendig

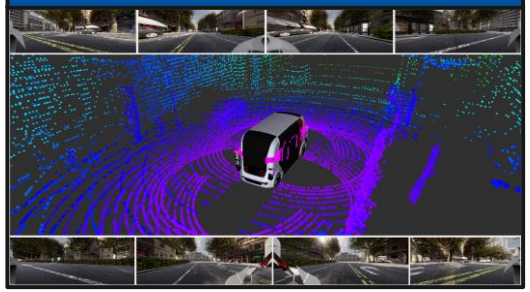
Testen in der **Simulation**

- Ermöglicht **schnelle Validierung** von entworfenen Fahrfunktionen
- **Datengetriebene Entwicklung** durch Simulationsumgebung
- **Tests** sind **automatisierbar** und **reproduzierbar**

Kombination von realen und simulativen Tests

Software Prototyping

Data Driven Development



Automated Testing

Anforderungen an eine generische Simulationsarchitektur

Simulationsframework

CARLA

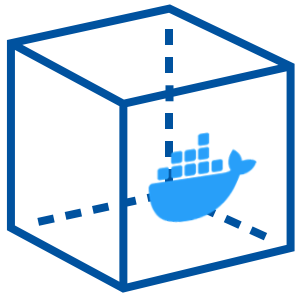
Standardisierte Schnittstellen

ROS 2

ASAM

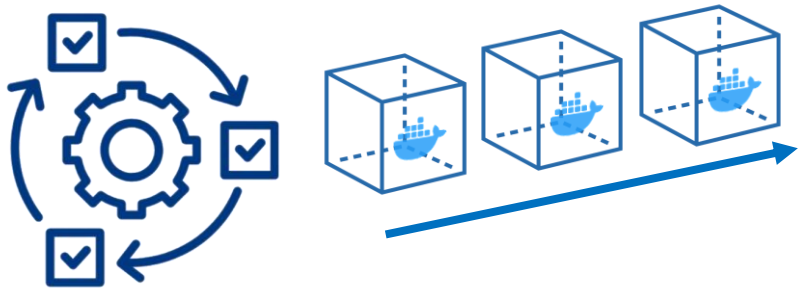
ETSI

Modulare Containerisierung



docker

Automatisiertes Testen



Fokus: Simulationsframework

Simulationsframework

Simulationen müssen bestimmte **Qualitätskriterien** erreichen

- Hohe Simulationsgüte von
  - Fahrdynamiken
  - Sensoren
  - Verkehrsfluss
- Integration mit relevanten Tools und Standards

CARLA Simulator

- Basiert auf UE4 (Migration zu UE5 bereits in Arbeit)
  - Hochgenaue Grafiken und Modellierungen
- Umfangreiches Ökosystem
  - ROS-Bridge, Scenario Runner, etc.

Anforderungen an eine generische Simulationsarchitektur

Simulationsframework

CARLA

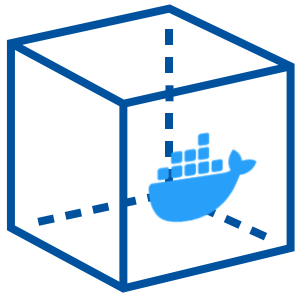
Standardisierte Schnittstellen

ROS 2

ASAM

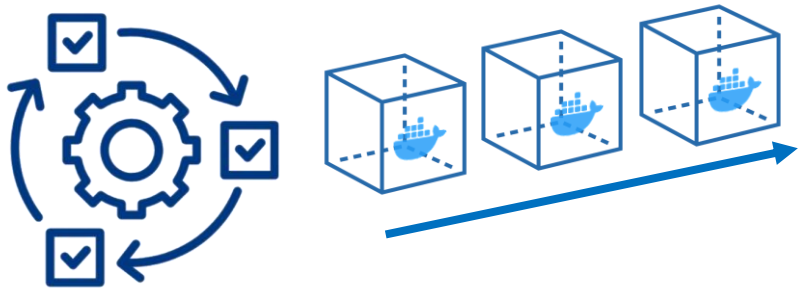
ETSI

Modulare Containerisierung



docker

Automatisiertes Testen

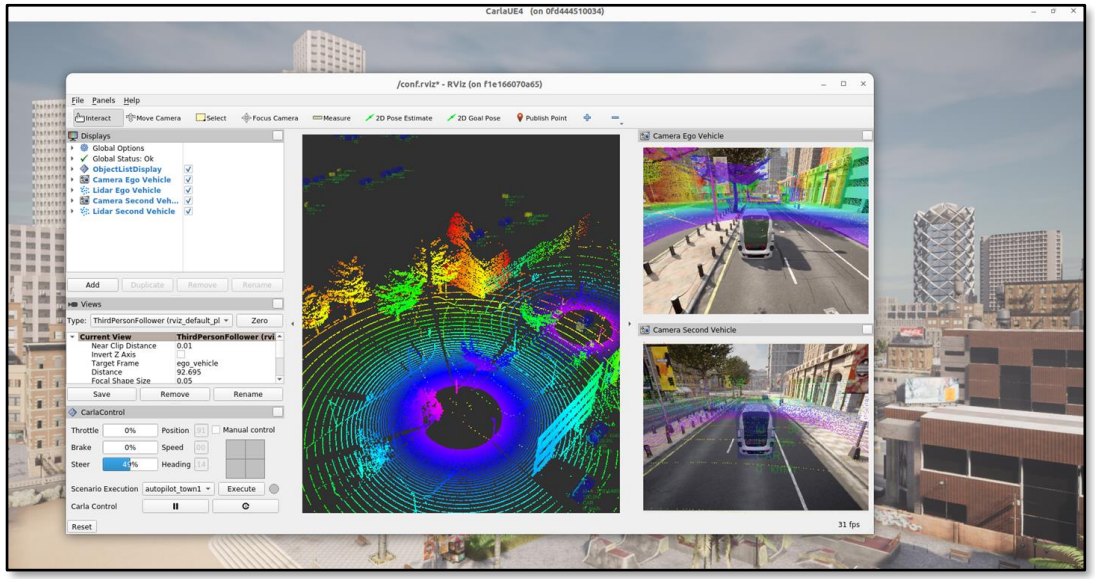


Fokus: Standardisierte Schnittstellen

Standardisierte Schnittstellen

CARLA ⟷ ROS 2 bridge

- Daten- und Steuerungsschnittstelle für **ROS 2**
  - Objekt- und Sensordaten (Bilder, Punktwolken, ...)
  - Simulationssteuerung (Start, Stopp, Szenariobearbeitung, ...)
  - Fahrzeugsteuerung (Lenkwinkel, Beschleunigung, ...)
- **Modulare Container**, welche die CARLA Python API nutzen
- **ika** arbeitet an einer **C++ API** Implementierung



Anforderungen an eine generische Simulationsarchitektur

Simulationsframework

CARLA

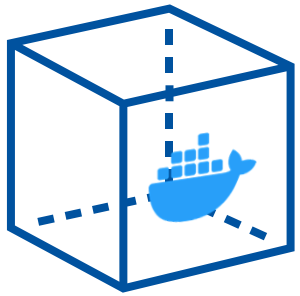
Standardisierte Schnittstellen

ROS 2

ASAM

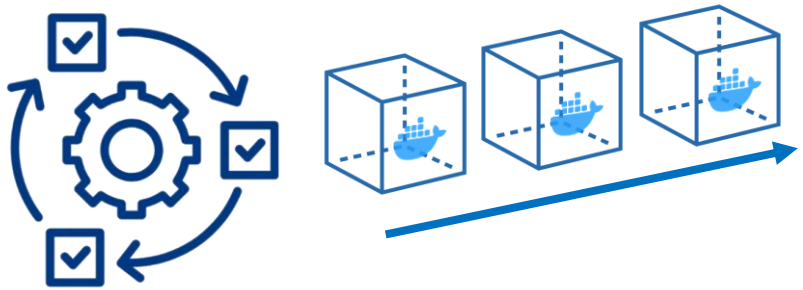
ETSI

Modulare Containerisierung



docker

Automatisiertes Testen

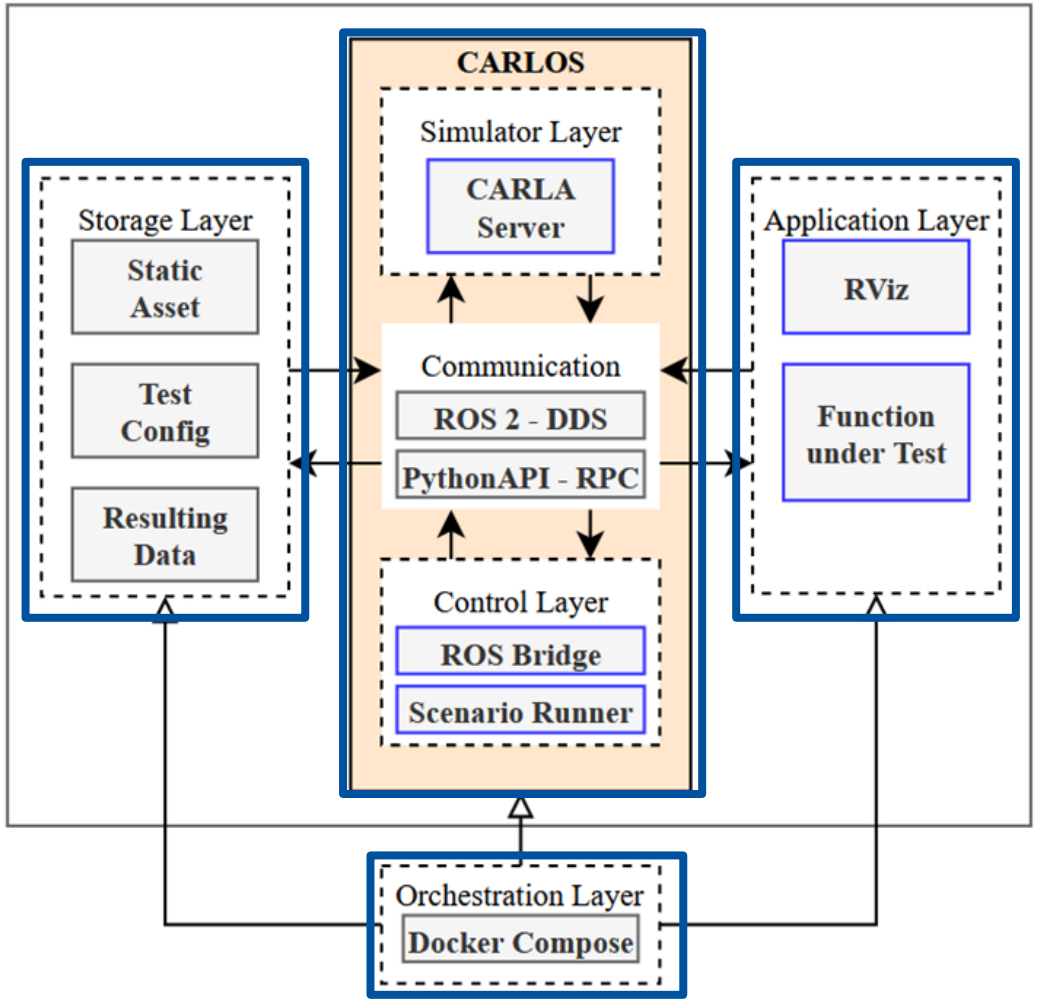


Fokus: Modulare Containerisierung

Modulare Containerisierung

Modulare Microservicearchitektur

- Flexibles Framework für alle **Use-Cases**
- **Funktionalität** eingeteilt in **mehrere Ebenen**
  - **Simulation**
  - **Datenspeicherung**
  - **Anwendung**
  - **Orchestrierung**



Anforderungen an eine generische Simulationsarchitektur

Simulationsframework

CARLA

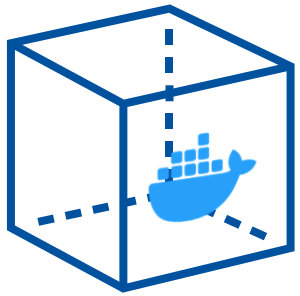
Standardisierte Schnittstellen

ROS 2

ASAM

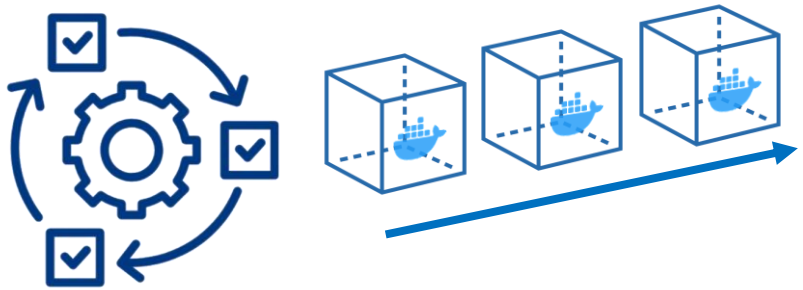
ETSI

Modulare Containerisierung



docker

Automatisiertes Testen



Fokus: Automatisiertes Testen

Automatisiertes Testen

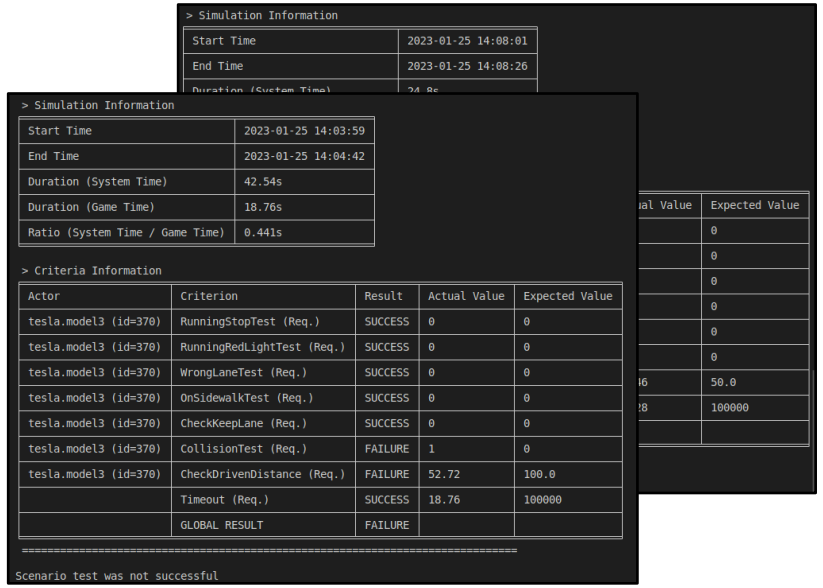
CARLA **„scenario runner“**

- Ermöglicht genau **definierte** und **reproduzierbare** Tests
- **OpenSCENARIO** Unterstützung mit vordefinierten Szenarien

Ziel: Nutzung von **CI-Pipelines** für Tests

- Ausgelöst über GitHub oder GitLab
- Modulare Teststrategie und parallele Ausführung von Szenarien

ASAM OpenSCENARIO®



> Simulation Information

Start Time	2023-01-25 14:08:01
End Time	2023-01-25 14:08:26

> Simulation Information

Start Time	2023-01-25 14:03:59
End Time	2023-01-25 14:04:42
Duration (System Time)	42.54s
Duration (Game Time)	18.76s
Ratio (System Time / Game Time)	0.441s

> Criteria Information

Actor	Criterion	Result	Actual Value	Expected Value
tesla.model3 (id=370)	RunningStopTest (Req.)	SUCCESS	0	0
tesla.model3 (id=370)	RunningRedLightTest (Req.)	SUCCESS	0	0
tesla.model3 (id=370)	WrongLaneTest (Req.)	SUCCESS	0	0
tesla.model3 (id=370)	OnSidewalkTest (Req.)	SUCCESS	0	0
tesla.model3 (id=370)	CheckKeepLane (Req.)	SUCCESS	0	0
tesla.model3 (id=370)	CollisionTest (Req.)	FAILURE	1	0
tesla.model3 (id=370)	CheckDrivenDistance (Req.)	FAILURE	52.72	100.0
	Timeout (Req.)	SUCCESS	18.76	100000
	GLOBAL RESULT	FAILURE		

Scenario test was not successful

Anforderungen an eine generische Simulationsarchitektur

Simulationsframework

CARLA

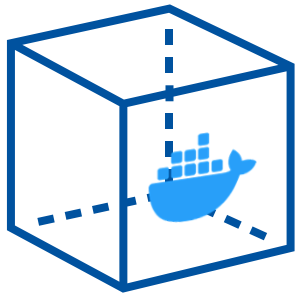
Standardisierte Schnittstellen

ROS 2

ASAM

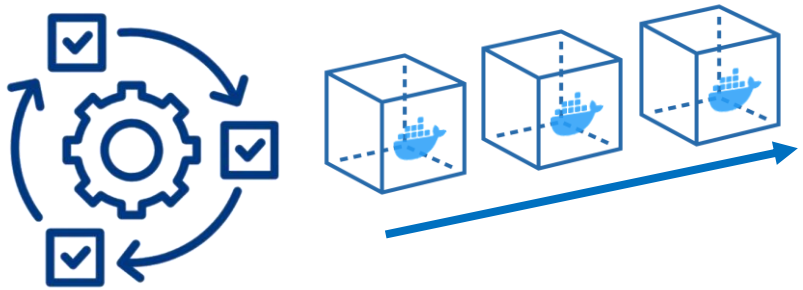
ETSI

Modulare Containerisierung



docker

Automatisiertes Testen



CARLOS: Ein offenes, modulares, and skalierbares Simulations-Framework

ROS-basierte Entwicklung

- Aufbau in flexibler **Microservice-Architektur**
- **Einfache Integration** von ROS-basierten Funktionen
- **CARLA-ROS Bridge** als Kommunikationsakteur zwischen beiden Welten

Toolchain zur **Datengenerierung**

- Beispiel: Erfassung benutzerdefinierter Wahrnehmungsdaten
- Flexibel und skalierbar, um die **Datenvielfalt zu erhöhen**

Automatisiertes Testen von Szenarien

- Automatisierung über CI-GitHub/GitLab-Pipelines
- Auswertung einer großen Anzahl von **benutzerdefinierten Szenarien** im OpenSCENARIO-Format

CARLOS: Beispiel einer Funktionsintegration

Integration einer **benutzerdefinierten Funktion** als neuer Docker-Service:

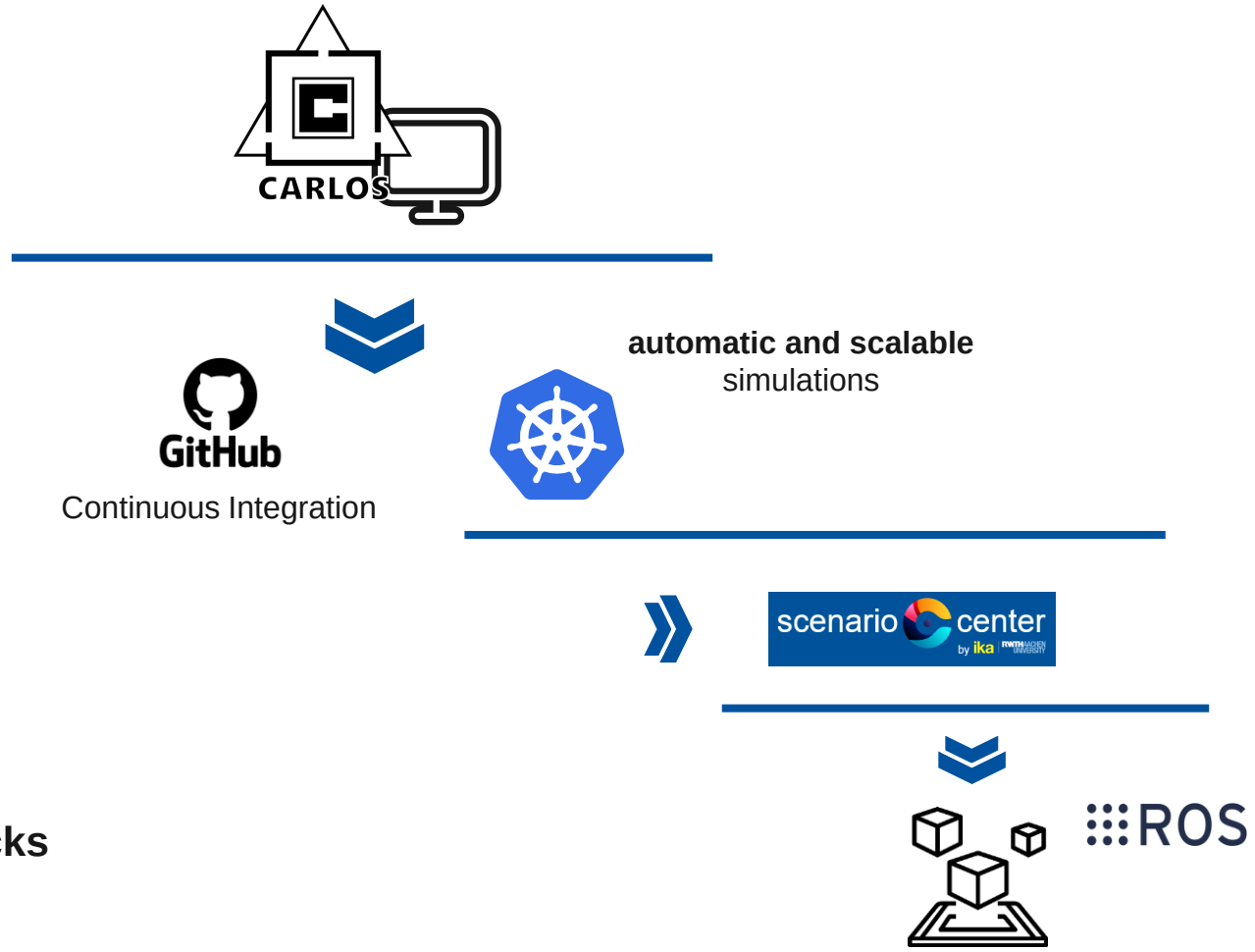
```
image-segmentation:
    extends:
        service: gpu-service
    image: gitlab.ika.rwth-aachen.de:5050/fb-fi/its-modules/perception/image_segmentation:latest
    command: ros2 launch image_segmentation image_segmentation.launch.py image_topic:="/carla/ego_vehicle/image"
```

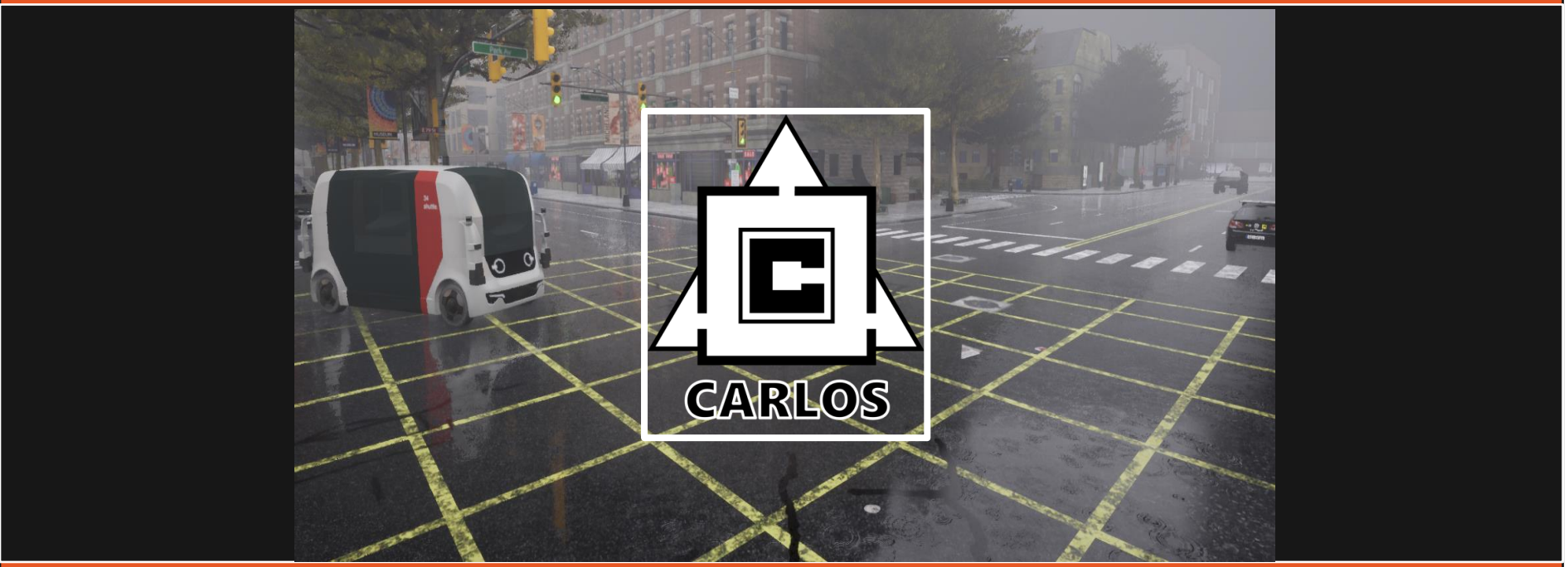
Entwickelt mit **docker-ros**:

github.com/ika-rwth-aachen/docker-ros

Ausblick: Mehr Monitoring, Skalierung und Testautomatisierung

- **CARLAB: Ein Browser-basiertes Simulationslabor**
- **Skalierung der Simulationstests mit Kubernetes**
  - Helm-Deployments
  - Parallele Ausführung von Szenarien
- Einbindung an eine große Anzahl von **Szenarien**
  - Verbindung zum ika **scenario.center**
- Komplette **Integration** des ika **ROS-AD Software Stacks**





Vielen Dank für die Aufmerksamkeit!

Fragen?

Kontakt

Silas Damaschke

silas.damaschke@ika.rwth-aachen.de

+49 241 80 26713

CARLOS

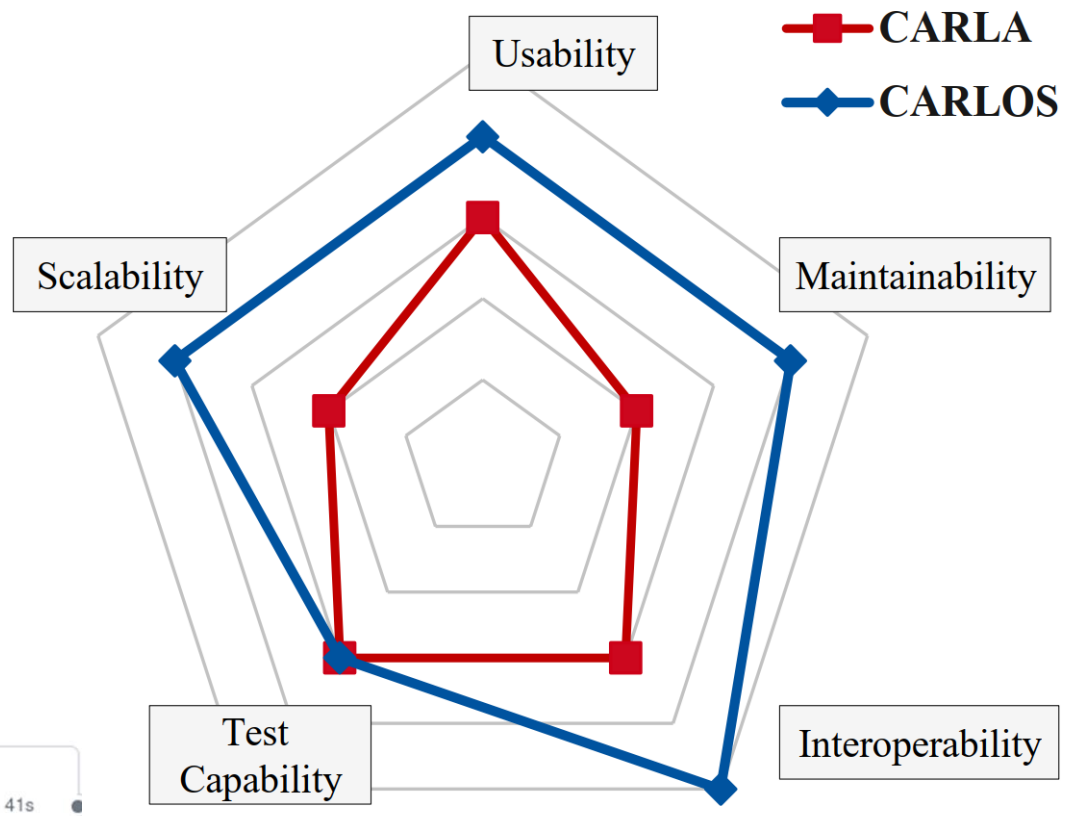
Probiere die Demo aus unter

Öffentlich verfügbar
github.com/ika-rwth-aachen/carlos

CARLOS: Evaluation und Vergleich

Qualitative Bewertung der Fähigkeiten

- Vorteile insbesondere bei:
  - Benutzerfreundlichkeit
  - Wartbarkeit
  - Skalierbarkeit



Beispiel für **Skalierbarkeit**

- Direkte Metrik Auswertung mit CARLOS innerhalb von GitHub Actions
- **OpenSCENARIO**-Ausführung