1 Thema

KoKoVa: Kontinuierliche und **Ko**mpositionelle **Va**lidierung für Autonomes Fahren

2 Zeitraum

Wintersemester 2022 / 2023 und Sommersemester 2023

3 Veranstalter

Simon Dierl <simon.dierl@cs.tu-dortmund.de>, LS14, Tel. 7543, OH12 / 2.008
Tim Tegeler <tim.tegeler@tu-dortmund.de>, LS5, Tel. 5802, OH14 / 113
Dominic Wirkner <dominic.wirkner@tu-dortmund.de>, LS5, Tel. 5806, OH14 / 107
Prof. Dr. Falk Howar <falk.howar@tu-dortmund.de>, LS14, Tel. 7945, OH12 / 2.009

4 Aufgabe

Im letzten Jahrzehnt verfolgt die Automobilindustrie zunehmend die Vision autonomer Fahrzeuge. In den USA sind in zahlreichen Regionen bereits autonome Taxis – sowohl mit als auch ohne Sicherheitsfahrer – im Testbetrieb. In Deutschland testet z. B. MOIA[1] in Hamburg ein autonomes Ride-Pooling-Fahrzeug. Auch im Rennsport wird es in Zukunft autonomes Fahren als Disziplin geben. Um die Sicherheit von autonomen Fahrfunktionen zu prüfen, wird heute meist eine Kombination aus Simulationen und Testfahrten gesetzt: Viele Kilometer in der Simulation schaffen ein grundlegendes Vertrauen in das System bevor Testfahrten auf der Straße oder Rennstrecke das in der Simulation gewonnene Vertrauen validieren.

Eine große Herausforderung bei der Entwicklung autonomer Fahrfunktionen ist die Anzahl verschiedener Komponenten, die benötigt werden, von der Wahrnehmung über die Planung zur Steuerung des Fahrzeugs. Diese Komponenten basieren auf unterschiedlichen Paradigmen (u. a. neuronale Netze, Optimierungsalgorithmen und Kontrolltheorie). Diese Komponenten werden oft unabhängig von unterschiedlichen Teams entwickelt und müssen dann auf dem Fahrzeug integriert werden. Um die Folgen von Änderungen an einzelnen Komponenten zu beherrschen, ist es wichtig, diese Änderungen direkt im Gesamtsystem zu testen. Hier kann Automatisierung einen großen Beitrag leisten.

Vision

Die notwendigen Werkzeuge für eine solche Automatisierung sind für andere Arten von Systemen bereits etabliert: Im Bereich der Webanwendungen hat sich unter dem Begriff DevOps [3] ein iteratives Vorgehen etabliert, um die Qualität der Software kontinuierlich zu erhöhen. DevOps adressiert den Lebenszyklus von Software von der Entwicklung, über die Auslieferung, bis hin zum Betrieb des Gesamtsystems. In der Praxis umfasst es Praktiken wie *Continuous Integration* und *Continuous Deployment* (CI/CD), Qualitätsprüfungen, Konfigurationsmanagement, automatische Releases und Monitoring. Während diese Praktiken bereits große Verbreitung in der Industrie gefunden haben im Bereich der Cyber-physischen Systeme noch wenig verbreitet. Erste Ansätze sind u. a. in [2] beschrieben.

[1] https://www.moia.io/

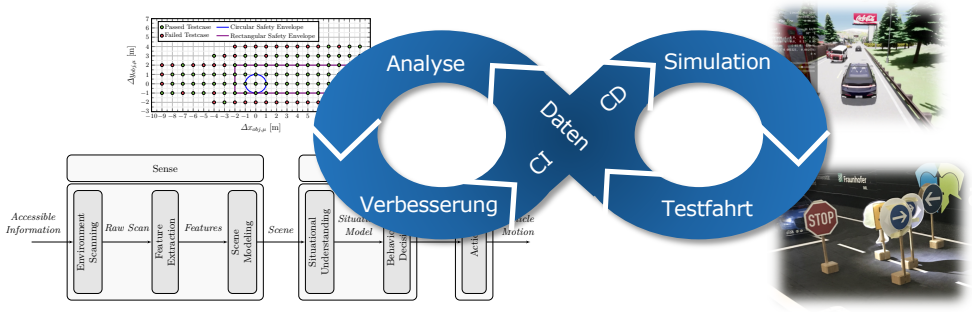


Abbildung 1: DevOps für autonome Fahrfunktionen.

Eine Übertragung auf den Bereich des autonomen Fahrens könnte, wie in Abbildung 1 angedeutet, aussehen: inkrementelle lokale *Verbesserungen* werden per CI/CD mit der Gesamtsoftware für die Simulation und für Testfahrzeuge integriert und ausgeliefert. In der *Simulation* werden automatisiert vordefinierte Szenarien abgefahren, welche die Sicherheit der Fahrfunktion grundlegend zu testen, und um z. B. optimale Parameter für die Fahrfunktion zu ermitteln. Mit diesen Parametern wird die Fahrfunktion dann auf der Teststrecke validiert. In der Simulation und in Testfahrten erhobene Daten werden in den Entwicklungsprozess gespielt und bilden die Basis für das Akzeptieren oder Ablehnen einzelner Änderungen. So wird insgesamt ein verteilter, komponentenbasierter Entwicklungsprozess mit kurzen Entwicklungszyklen möglich.

Ziele und Vorgehen

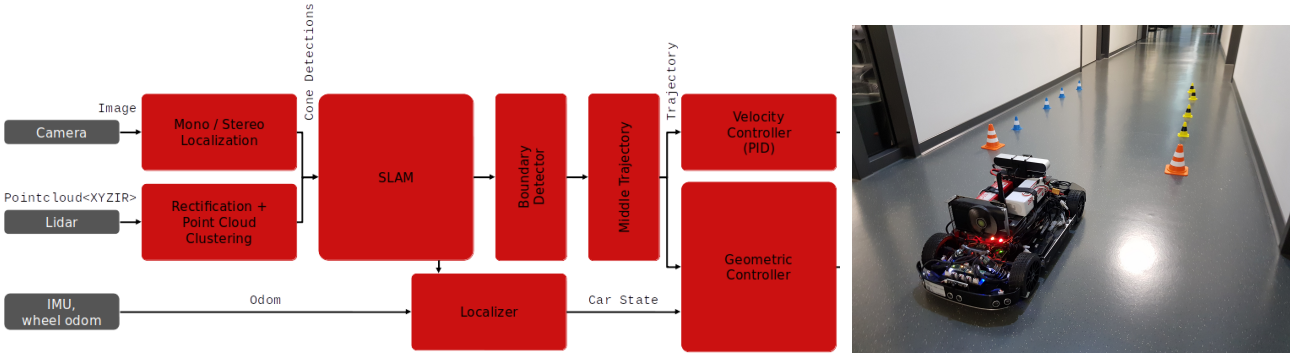
Die Aufgabe der Projektgruppe ist es, den skizzierten DevOps-basierten Entwicklungsprozess für autonome Fahrfunktionen und die dafür notwendigen Werkzeuge (u. a. simulationsbasiertes Testen, Continuous Integration, Auswertung von Fahrdaten) zu entwickeln. Als Fallstudie soll eine Fahrfunktion für Formula Student-ähnliche Rennfahrten entwickelt werden. Die Formula Student ist ein Wettbewerb, in dem studentische Teams von Hochschulen aus aller Welt in der Entwicklung eines Rennwagens gegeneinander antreten. Analog zu der industriellen Entwicklung hat sich auch hier der Fokus auf elektrisches und autonomes Fahren erweitert.

Kompositionelle Entwicklung der Fahrfunktion

Traditionell wurde Software für Automobile im Rahmen der industrietypischen Entwicklungsprozesse – d. h., wasserfallartig und produktspezifisch – entwickelt. Spätestens im Bereich des autonomen Fahrens scheint dieses Modell an seine Grenzen zu kommen.

Das Framework ROS[2] ermöglicht eine andere Herangehensweise durch ein Modul- und RPC-System, das die Entwicklung von Funktionen in Form dedizierter Knoten mit wohldefinierten Schnittstellen ermöglicht. Am Beispiel von ROS und der Zielplattformen Simulation, Modell- und Realauto soll ein Katalog an Basiskomponenten für autonomes Fahren entwickelt werden, der die Probleme kamerabasierte Wahrnehmung, SLAM, Steckenerkennung und Pfadplanung wiederverwendbar und zuverlässig umsetzt. Als Basis stehen eine vom GET racing Dortmund e. V. entwickelte Software, JARVIC, auf Basis von ROS 1 sowie Modellautos im Maßstab 1:8 gegenüber einem Audi Q2 zur Verfügung, die in Abbildung 2 dargestellt sind.

[2] `https://www.ros.org/`



(a) Architektur von JARVIC (b) AADC-Modellauto

Abbildung 2: Softwarearchitektur und Plattform für die GET racing-Fahrfunktion.

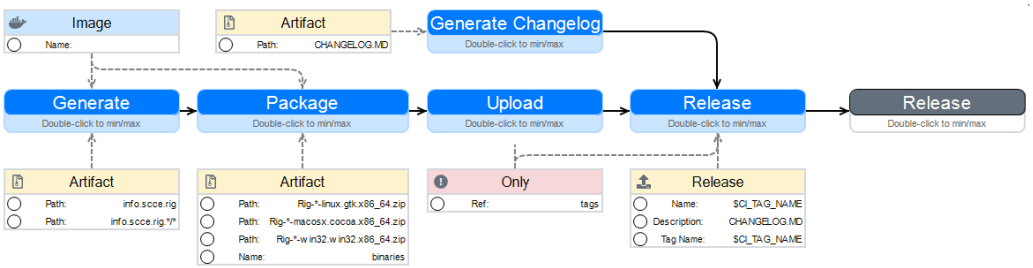


Abbildung 3: Grafisches Modell eines beispielhaften CI/CD-Workflows.

DevOps-basierter Entwicklungsprozess

Im Bereich der Webanwendungen ist es bereits industrielle Praxis, Änderungen an der Software kontinuierlich zu validieren. Ermöglicht wird dies durch die formale Beschreibung und vollständige Automatisierung des Auslieferungsprozesses auf Basis von CI/CD. So können vorgefertigte Tests und Qualitätsprüfungen, ohne menschliche Interaktion, durchgeführt werden. Am Lehrstuhl 5 wird für die grafische Erstellung von CI/CD-Workflows (siehe Abbildung 3) eine eigenständige Modellierungsumgebung, genannt Rig[3], entwickelt.

Da real existierende Hardware oft nicht ohne menschliche Hilfe testbar ist, ist bereits die automatische Ausführung von Tests nicht möglich. Die Nutzung von simulationsbasierten Tests [1, 4] mithilfe von Simulationswerkzeugen wie Carla[4] oder Gazebo[5] bietet allerdings einen möglichen Ansatz, dauerhaft und automatisch entwickelte Komponenten zu validieren.

Dies stellt hohe Anforderungen: die Simulationen müssen möglichst repräsentativ für das Verhalten in der echten Welt sein und die Ergebnisse müssen in einer für Entwickler nutzbaren Form aufbereitet und dargestellt werden. In der gewählten Beispieldomäne sollen geeignete Tests einen Rennwagen über eine Vielzahl an gültigen Strecken schicken und die Performance z. B. nach dem Formula Student-Punktesystem bewerten, um unmittelbares Feedback zur fahrerischen Leistung zu generieren.

Ziel ist Simulations-basiertes Testen mit den gängigen DevOps-Praktiken, wie Automatisierung und kontinuierlichen Feedback, umzusetzen. Eine zentrale Rolle spielen dabei CI/CD-Workflows (s. Abbildung 1), die im Rahmen der Projektgruppe mit Rig modelliert werden sollen. Dabei ist es die Aufgabe der Projektgruppe, für diesen Anwendungsfall mögliche Einschränkungen von Rig aufzudecken und passende Lösungskonzepte zu erarbeiten. Dies kann sowohl die

[3] https://scce.gitlab.io/rig/
[4] http://carla.org/
[5] https://gazebosim.org/

Weiterentwicklung an der Modellierungssprache als auch Änderungen am Codegenerator nach sich ziehen.

5 Teilnahmevoraussetzungen

Für eine Teilnahme sind folgende Kenntnisse *notwendig*:

- Sichere Beherrschung einer Programmiersprache (z. B. C++, Java, oder Python)
- Grundkenntnisse in der Arbeit mit Linux

Folgende Kenntnisse sind *hilfreich*, werden jedoch nicht vorausgesetzt:

- Grundkenntnisse zu cyber-physischen Systemen (Regelungstechnik etc.)
- Erfahrungen mit ROS
- Grundkenntnisse im Bereich DevOps
- Vorwissen im Bereich des autonomen Fahrens

6 Minimalziel

Die Minimalziele der PG sind:

- Realisierung einer Fahrfunktion als Fallstudie für Formula Student-ähnliche Rennfahrten.
- Entwicklung wiederverwendbarer Komponenten für autonomes Fahren, insbesondere für Wahrnehmung, SLAM, Racetrack-Steckenerkennung und eine optimierte Pfadplanung.
- Einsatz und Weiterentwicklung von DevOps-Werkzeugen, insbesondere Rig, zur automatischen Validierung von Fahrfunktionen, die die Validierung der entwickelten Komponenten ermöglichen.

7 Literatur

[1] A. Bussler, L. Hartjen, R. Philipp, and F. Schuldt. Application of evolutionary algorithms and criticality metrics for the verification and validation of automated driving systems at urban intersections. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 128–135, Oct. 2020. ISSN: 2642-7214.

[2] B. Combemale and M. Wimmer. Towards a model-based DevOps for cyber-physical systems. In J.-M. Bruel, M. Mazzara, and B. Meyer, editors, *Software Engineering Aspects of Continuous Development and New Paradigms of Software Production and Deployment*, pages 84–94, 2020.

[3] L. Leite, C. Rocha, F. Kon, D. Milojicic, and P. Meirelles. A survey of DevOps concepts and challenges. *ACM Computing Surveys*, 52(6):127:1–127:35, Nov. 2019.

[4] R. Philipp, H. Qian, L. Hartjen, F. Schuldt, and F. Howar. Simulation-based elicitation of accuracy requirements for the environmental perception of autonomous vehicles. In T. Margaria and B. Steffen, editors, *Leveraging Applications of Formal Methods, Verification and Validation*, pages 129–145, Cham, 2021.

8 Rechtlicher Hinweis

Die Ergebnisse der Projektarbeit und die dabei erstellte Software sollen der Fakultät für Informatik uneingeschränkt für Lehr- und Forschungszwecke sowie dem GET racing Dortmund e. V. für den Betrieb autonomer Fahrsysteme zur freien Verfügung stehen. Darüber hinaus sind keine Einschränkungen der Verwertungsrechte an den Ergebnissen der Projektgruppe und keine Vertraulichkeitsvereinbarungen vorgesehen.