Build Your Own Demonstration Platform for Autonomous Cars

An Introduction to the Lab Course

Master Lab Course
October 19th

Sina Shafaei
Dr. Mohd Hafeez Osman
Dr. rer. nat. Stefan Kugele
Vadim Cebotari
Demonstration Platform for Autonomous Cars
Setting Up Your Demo Car

• Necessary equipment will be provided to you
• Instructions are available in the moodle
• You will use this demo car for testing and evaluation
• Working place is our lab at 03.07.011 (24/7)
Setting Up Your Demo Car

• Applications to implement on the car:
  • Lane following (one Vs. two)
  • Obstacle avoidance (Sensoric approach)

• Application to implement remotely on your own computer:
  • An state-of-the-art object detection algorithm
  • Objectives and details will be released to you over moodle
Assignments and Deadlines

• 3 Categories of:
  • Hardware and AI
  • Safety
  • SOA

• Assignments as follow:
  • (2 weeks) 1st Assignment: Setting up the demo car (19.10 - 2.11)
  • (4 weeks) 2nd Assignment: SOA / Analysis, Logic, Implementation Design (2.11 - 30.11)
  • (5 weeks) 3rd Assignment: Implementing the object detection algorithm (state-of-the-art) (30.11 - 11.01)
  • (2 weeks) 4.1st Assignment: Anomaly detection (11.01 - 25.01)
  • (3 weeks) 4.2nd Assignment: Safety platform and integration (25.01 - 15.02)
  • (2 weeks) Testing and evaluation (will be discussed later)
Coordination

• Time and Location: 03.07.011 / Irregular Meetings
• Each group will get an access card from our Secretary (03.07.052)
• Main coordination through the moodle of the lab course and gitlab repository
• Please find a team member and fill the respective form

https://campus.tum.de
http://www6.in.tum.de/en/teaching/ws-1819/lab-course-build-your-own-demonstration-platform-for-autonomous-cars/
Build Your Own Demonstration Platform for Autonomous Cars

Safety

Master Lab Course
October 19th

Sina Shafaei
Dr. Mohd Hafeez Osman
Dr. rer. nat. Stefan Kugele
Vadim Cebotari
Toyota "Unintended Acceleration" Has Killed 89 Acceleration

Unintended acceleration in Toyota vehicles may have been involved of 89 people over the past decade, upgrading the number of deaths to the massive recalls, the government said Tuesday.

In the fall of 1908, Tesla launched new software over the air to cars in the United States and elsewhere that added safeguards to its Autopilot system to prevent drivers from taking away from the road keeping mode by taking off the steering wheel for long periods of time.

In the fall of 1908, Tesla launched new software over the air to cars in the United States and elsewhere that added safeguards to its Autopilot system to prevent drivers from taking away from the road keeping mode by taking off the steering wheel for long periods of time.

The driver suffered minor injuries and told police she was using the car's driver-assisting Autopilot mode.

The crash has similarities to other incidents, including a fatal crash in Florida where the driver's "over-reliance on vehicle automation" was determined as a probable cause.
Functional Safety Overview

**Item Definition**
- Describe the item with regard to its functionality, interfaces, environmental conditions, legal requirements, hazards, etc.
- Define boundary of the item and its interfaces
- Assumptions concerning other functions, systems and components

*Figure 2 — Safety lifecycle [ISO 26262]*
Functional Safety Overview

Initiation of the Safety Lifecycle
• New development lifecycle or modification of existing lifecycle

Figure 2 — Safety lifecycle
Functional Safety Overview

Hazard Analysis and Risk Assessment
- Determine Automotive Safety Integrity Level (ASIL)
- Define safety goal for each hazardous event

Figure 2 — Safety lifecycle [ISO 26262]
Functional Safety Overview

**ASIL = \{QM, A, B, C, D\}**

<table>
<thead>
<tr>
<th>Class</th>
<th>S0</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>No injuries</td>
<td>Light and moderate injuries</td>
<td>Severe and life-threatening injuries (survival probable)</td>
<td>Life-threatening injuries (survival uncertain), fatal injuries</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
<th>E4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Incredible</td>
<td>Very low probability</td>
<td>Low probability</td>
<td>Medium probability</td>
<td>High probability</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Class</th>
<th>C0</th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Controllable in general</td>
<td>Simply controllable</td>
<td>Normally controllable</td>
<td>Difficult to control or uncontrollable</td>
</tr>
</tbody>
</table>

Table 4 — ASIL determination

<table>
<thead>
<tr>
<th></th>
<th>C1</th>
<th>C2</th>
<th>C3</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>E1</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td>S2</td>
<td>E1</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>S3</td>
<td>E1</td>
<td>QM</td>
<td>QM</td>
</tr>
<tr>
<td></td>
<td>E2</td>
<td>QM</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>E3</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>E4</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
Functional Safety Overview

Safety Goal
• Top-level safety requirements for the item.
• Lead to functional safety requirements needed to avoid unreasonable risk.
• ASIL determined for the hazardous event shall be assigned to the corresponding safety goal.

Examples:
• Anti-lock control must be performed only when required.
• Brake Controller must engage brake when required.
• Steering Controller must change steering angle when required.
Functional Safety Overview

- Derive functional safety requirement from safety goals.
- Specify basic safety mechanism such as:
  - Fault detection and failure mitigation
  - Transitioning to safe state
  - Fault tolerance

**Functional Safety Concept**

*Figure 2 — Safety lifecycle [ISO 26262]*
Functional Safety Overview

Product Development: System Level
- V-model approach
- Technical safety requirements
- System design
- Testing
- Integration
- Verification
- Validation
- Assessment of Functional Safety

Figure 2 — Safety lifecycle [ISO 26262]
Functional Safety Overview

Production & Operation Planning
- Planning for production and operation

Other Technologies
- Other technologies different from E/E-technology covered ISO 26262

Controllability
- From Hazard Analysis and Risk Assessment

External Measures
- Measures outside the system (e.g. guardrails, fire-fighting system)

Release for Production
- Final subphase of product development
  - Provides item signoff
Functional Safety Overview

Production
- Ensure the required functional safety is achieved during production process.

Operation, Service and Decommissioning
- Define the scope of customer information, maintenance and repair instructions regarding the safety-related products
- Provide the requirements concerning activities addressing safety issues before disassembly.

Figure 2 — Safety lifecycle [ISO 26262]
Functional Safety & Ai-based Systems

“The future E/E architecture will be highly influenced by Intelligent systems.”

• How to ensure safety for the uncertainty application software (e.g. AI-based application, Autonomous/Unmanned features)?

• How to identify safety violation from the dynamic application or from a runtime application? Can it be done automatically?

• Since AI application seems to be a black box application that the nominal behavior is hard to identify (or find reason), how to monitor safety on this type of application?
Functional Safety & Ai-based Systems

- Managing safety for Ai-based systems
- Focus on After SOP Phase
  - Safety Monitor
  - Safety Control
  - Safety Analysis

Figure 2 — Safety lifecycle
“The (run-time) Safety Management Platform”

Service Profile
- Input (Range/limiter)
- Output (Range/limiter)
- Input/output threshold
- Safety log Criteria
- Backup Function / Fail-over Operation and Safety Mechanism

Functionality/Service
- **Identify** – Identification of AI-based Software
  - Manage Service Identification (ID)
  - Backup Function / Safety Mechanism / Tactics
  - Fail-over Operation
- **Monitor** – Monitoring Input and output
  - Monitor input, output and threshold
  - Safety log
- **Analyze** – Analyzing safety violation
  - Analysis of safety violation
  - Safety violation statistic
  - Safety violation pattern*
- **Response** – Updating safety information
  - Uploading safety information (safety log)
  - Safety Alert

Online safety control
- **Alert**
  - Warning
- **Log Monitoring**
  - Safety Log statistic
- **Update Safety Information**
  - Update safety information
    - Modify, Delete, New
  - Safety ID creation
  - Safety profile verifier*

Dr. Mohd Hafeez Osman
Ai-based systems function is a safety function that is basically used to:

• control the Ai-based systems by using safety profile,
• holds the safety limiters value,
• impose safety based on the safety limiters, and
• send the log and safety violation events to the *In-car safety management services* (ICSM).

"The (run-time) Safety Management Platform"
“The (run-time) Safety Management Platform”

The main functionality of ICSM is to:
• centrally manage the safety of the AI-based systems
• interact with the AI-based systems and the Safety Control (RSaC) component.
• handle all safety update for AI-based systems (internal/in-car).
[RSaC] Safety Control (Remote)

- The safety control is basically an online control system that interacts with ICSM.
- This component is used to control the safety and monitor the AI-based systems safety related events.
“The (run-time) Safety Management Platform”

• **Challenge**
  - Anomaly detection on AI-based systems input
    - Feature selection
    - Anomaly Detection approach
      - Statistical methods
      - Machine Learning methods
Build Your Own Demonstration Platform for Autonomous Cars

Service Oriented Architecture (SOA) / Part II

Master Lab Course
October 19th

Sina Shafaei
Dr. Mohd Hafeez Osman
Dr. rer. nat. Stefan Kugele
Vadim Cebotari
From a Function to a Technical Realization

We develop functions on different levels of abstraction.

- **Service Architecture**
  - Models the decomposition of systems into services, their interrelation and interaction.

- **Logical Architecture**
  - Models the logical structure and involved data-flow between subsystems.

- **Technical Architecture**
  - Models the Hardware Topology, deployment information, and implementation.
Service Architecture

The Service Architecture is a means to understand the system, their decomposition, and interaction.

The Service Architecture models the decomposition of a high-level service into sub-services: Services at level $L_n$ can only use (i.e., subscribe to) services from layers $\leq n$.

Interactions can be strict or non-strict. For non-strict service usages we need a rationale.

The lower a service is, the more hardware-dependent it is.

The higher a service, the more hardware agnostic.

Layered Architecture:

Strict and non-strict service usage
Logical Architecture

The Logical Architecture is a means to understand the system, their decomposition, logical data-flow.

The **Logical Architecture** models the **decomposition** into sub-systems and the involved **data-flow**.

There can be several interaction patterns:

a) Request / Response,

b) Periodic,

c) Event-based, …
Technical Architecture

The Technical Architecture is a means to understand the technical realization of a system.

The Technical Architecture models hardware topology, the deployment, and software architecture.

It is the foundation for the implementation.
Problem Description

Objectives

We demand that you provide a **Service Architecture** including the **functions** that need to be modeled as services.

You need to provide services to access **sensors** and **actuator**.

We want that you understand that the functions may **interact** (aka **feature interactions**) in certain operational situations.

Your implementation needs to be **aware** of that and moreover react appropriately on **changing situations** (e.g. no connection to the laptop, sensor failure, etc.).

Modeling the Logical and Technical Architecture will help to understand better the system under development.
Build Your Own Demonstration Platform for Autonomous Cars

Service Oriented Architecture (SOA) / Part I

Master Lab Course
October 19th

Sina Shafaei
Dr. Mohd Hafeez Osman
Dr. rer. nat. Stefan Kugele

Vadim Cebotari
A service is a software program that makes its functionality available via a published API that is part of the service contract.

A service contract consists of the service API and the Service Level Agreement (SLA). The Service Level Agreement describes additional quality-of-service guarantees, behaviors, and limitations.
Service-Orientation as a Design Paradigm

- **Service** as a reusable unit of solution logic characterized through a service contract
- **Service composition** as means to assemble services to provide the functionality required by a specific business task or process
- **Service registry** as a catalog of information about available services and their locations
Service Interaction

Static service binding

Dynamic service binding

Client → Service

Service Registry

Service

Service Consumer

Client

Static service binding

Dynamic service binding

Service Consumer

Client

Service

Service Registry

Find

Publish
Service-Oriented Analysis

**Goal:** Identification of service candidates based on functional requirements

1. Investigate service properties
   i. Granularity
   ii. Scope
   iii. Runtime behavior
   iv. Functional behavior
   v. Communication pattern
   vi. Deployment environment

2. Model service dependencies

**Deliverables:** Service conceptual model
Service-Oriented Design

**Goal:** Definition of service contracts
1. Service API
2. Service-level agreement (SLA)

**Deliverables:** Service contract specifications
Service Implementation

**Goal:** Implementation of services

**Deliverables:** Technical specification of services
Service-Orientation Design Principles

- Standardized Service Contract
- Service Loose Coupling
- Service Abstraction
- Service Reusability
- Service Autonomy
- Service Statelessness
- Service Discoverability
- Service Composability
Service-Orientation Design Principles

**Standardized Service Contract**

Services follow the same contract design standards

- Base for intrinsic interoperability
Usage of consistent data models for information exchange
  - SOAP-based webservices: WSDL and XML Schema
  - Restfull webservices: uniform API provided by HTTP
Service-Orientation Design Principles

**Service Loose Coupling**
Promotes independent design and evolution of service logic while still guaranteeing baseline interoperability

- Strong emphasis on reducing dependencies between services
Service-Orientation Design Principles

Service Abstraction
Promotes hiding of implementation details
- Information about services is limited to what is published in service contracts
Service Reusability
Promotes reuse of services in different contexts
- Increased opportunities to use service logic for multiple purposes
- Increased opportunities to combine services in different configurations
- Solves software system integration problems at enterprise level
Service-Orientation Design Principles

**Service Autonomy**
Promotes design of independent services from underlying runtime execution environment

- Increase in service’s runtime reliability, performance and predictability
Service-Statelessness
Supports the design of agnostic services and improves the potential of service reuse
- Increase service scalability
Service-Orientation Design Principles

**Service Discoverability**
Promotes effective discoverability and interpretation of services

- Usage of service discovery mechanism (e.g. service registry) or metadata to increase the discoverability of services
Service-Orientation Design Principles

Service Composability
Promotes effective composition of services, regardless of the size and complexity of composition

- Ability to effectively compose services to solve business problems as fundamental goal of service-oriented paradigm