**Web Simulator Interface for Autonomous Vehicle Software Stack.**

**This thesis is confidential.**

|  |  |
| --- | --- |
|  | |
| Master Thesis 3263 | |
| Submitted at the University of Stuttgart by  **Sarthak Bapat** | |
| INFOTECH | |
|  | |
| Examiner: | Prof. Dr.-Ing. Michael Weyrich |
| Supervisor: | Msc. Hannes Vietz |
| External Supervisor: | Prof. Dr.-Ing. Christof Ebert |
|  | 19/01/2022 |

Remarks regarding this template:

General

The thesis is a major part of your examination. You are welcome to use other theses as examples or talk about your structure with your supervisor. However, the specific content and composition of your thesis should be your own and more than a mere pasting of the process model’s documents. Your goal should be a description of the main steps and results of your thesis project laid out in a scientific language to allow outsiders to understand why you made which decisions and to what purpose.

Please replace any place holders in square brackets (e.g. „<End Date>“) with the correct information and delete the brackets afterwards (e.g. “2017-01-01”).

Please remove all blue remarks (including this whole page).

Please use MS Word features (e.g. for creating lists and tables, figure and table captions and cross-references) wherever possible.

Title Page

Please do not abbreviate the type of thesis (e.g. “Master Thesis” instead of “MT”).

Please remove all contents of the line “External Supervisor” if you don’t have one.

Please remove the text box “This thesis is confidential” if you your thesis is not confidential.

# **Table of** **Contents** Style Template: Überschrift 1 / Headline 1 (delete the number)

Table of Contents *Style Template: Überschrift 1 / Headline 1 (delete the number)* iii

Table of Figures *Style Template: Überschrift 1 / Headline 1 (delete the number)* iv

Table of Tables *Style Template: Überschrift 1 / Headline 1 (delete the number)* v

Table of Abbreviations *Style Template: Überschrift 1 / Headline 1 (delete the number)* vi

Glossary *Style Template: Überschrift 1 / Headline 1 (delete the number)* vii

Abstract viii

1 Introduction 9

1.1 What is Autonomous Driving Software? 9

1.2 Current Setup 10

1.3 Scope of the Thesis 10

2 Analysis and Design of AV Software stack 12

2.1 Existing Interface Architecture 12

2.1.1 Apollo AV software stack 13

2.1.2 Apollo Cyber-RT 13

2.2 Interface options between Simulators and Apollo 13

2.2.1 Cyber RT Interface 13

2.2.2 ROS or ROS2 Interface 15

2.3 Problems with VTD connection to Apollo 16

2.4 Decision on AV Software 16

3 System Model 17

Bibliography 18

Declaration of Compliance *Style Template: Überschrift 1 / Headline 1 (delete the number)* 19

# Table of Figures Style Template: Überschrift 1 / Headline 1 (delete the number)

Style Template for Table of Figures: IAS\_TableOfFigures

**No table of figures entries found.**

# Table of Tables

**No table of figures entries found.**

# Table of Abbreviations Style Template: Überschrift 1 / Headline 1 (delete the number)

Convention: Mark the letters that make up the abbreviations by making them bold

|  |  |
| --- | --- |
| ASCII  ROS  AV | **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange  **R**obot **O**perating **S**ystem  **A**utonomous **V**ehicle |

# Glossary Style Template: Überschrift 1 / Headline 1 (delete the number)

Style Template: IAS\_Glossary (make main term bold)

|  |  |
| --- | --- |
| **Actuator** | Component for the conversion of an information carrying, low-energy control signal to a high energy signal capable of process intervention |

# Abstract

**Key Words:**

# Introduction

Autonomous systems in general are widely used in today’s world. Autonomous driving has been the latest trend in the market with rapid advancement in the AI, sensor, and the software technology. Major players in the automotive market are conducting research in autonomous driving and shifting towards increased levels of autonomy. A completely autonomous car with level 5 autonomy is expected to drive without any human intervention and handle any situation safely. This means that the vehicle should be intelligent enough to be able to take decisions in worst of the situations that are on par with the human decisions [1]. For this, the autonomous vehicle should be tested thoroughly under various scenarios such that they are safe enough to operate on roads. That brings us to the most important and wide research topic in autonomous driving – Validation of Autonomous Driving.

Before an autonomous vehicle is made available in the market, it must be validated and deemed fit enough to operate safely on the roads under any circumstances. The public trust in autonomous vehicles can only be achieved by continuous validation of the autonomous vehicle software [1]. One way of performing the validation is to directly test the vehicle on road. This technique, though possible to use is not convenient in terms of the number of miles that need to be driven to certify safety of autonomous vehicle and poses threats to the human life and property. To demonstrate that the failure rate of autonomous vehicles is statistically significantly lower than the human driver failure rate they need to be driven an approximately 5 billion miles [2]. This is highly inconvenient validation activity and it would take years to cover all weather conditions, road and traffic conditions in different parts of the world.

The other way of validation is to perform the simulation of the scenarios and test the performance of the autonomous vehicle software. A scenario consists of different weather, road, light and traffic conditions in different parts of the world, on different terrains and with contrasting driving conventions. For the autonomous vehicle software to be deemed safe, it must perform optimally under various scenarios as mentioned. The autonomous vehicle under test in the simulation is controlled by the autonomous vehicle software depending on the inputs received from the simulation environment. In this master thesis the validation of autonomous driving system using the simulation technique is considered.

## What is Autonomous Driving Software?

Autonomous driving software is the main component or the control unit of the autonomous vehicle. It basically receives the environment data from various sensors embedded in the vehicle, processes and makes decisions on the received data and sends the control commands back to the autonomous vehicle. The control commands include the throttle, gear, brake and acceleration values that determine the next position and the action of the autonomous vehicle. Any autonomous vehicle software stack consists of Localization, Perception, Prediction, Planning and Control as its core components.

The ‘Localization’ module is responsible for single digit centimeter level accuracy localization of the vehicle in the environment. Generally, the data from GPS, IMU (Inertial Measurement Unit) and LiDAR sensors is input to this component. ‘Perception’ module takes in the data from LiDAR, RADAR, Camera sensors and performs tasks such as classification, detection, segmentation and learning convolutional neural networks to perceive the surroundings of the vehicle. ‘Prediction’ module predicts the movement of other traffic elements like cars and pedestrians with respect to the autonomous vehicle. This information is consumed by the ‘Planning’ module to generate the right trajectory for the vehicle. Finally, the ‘Control’ component ensures that steering, throttles and brake commands are issued to the vehicle to execute the planned trajectory [3].

## Current Setup

The Robo-Test demonstrator has the LGSVL simulator connected to the Apollo autonomous vehicle software stack, a proprietary simulator VTD (Virtual Test Drive) and a web interface designed to create the test scenarios for the LGSVL as well as the VTD simulators. The web application was designed in MT-3136 [4] and VTD was introduced and integrated at a basic level in the web application in MT-3183 [5].

MT-3192 [6] has established an interface between the VTD simulator and the Apollo autonomous software stack using ROS [6]. The interface doesn’t support all the sensor data exchange between VTD and Apollo, also it introduces a few problems that are analyzed further in this thesis.

## Scope of the Thesis

This master thesis explores the possibility of creating a flexible interface between the LGSVL, VTD simulators and the Apollo software stack using ROS or ROS2. A feasibility check regarding a ROS based interface is performed before taking the decision. Depending on the outcome of the feasibility check, the existing Robo-Test web application interface is enhanced with new features and the details are discussed in the sections to follow.

Furthermore, in this thesis a detailed literature survey regarding the traceability techniques between the requirements and the test results is performed and a concept has been worked out for the current setup of the Robo-Test system. The concept tries to link the requirements engine to the test execution and test results. A detailed literature review regarding the techniques for automatic generation of test cases from requirements is also presented.

The rest of the report is structured as follows. Chapter 2 discusses the analysis of the current system and the fundamental design decisions. In the Chapter 3, we discuss the software architecture and corresponding system model. The implementation details are discussed in the Chapter 4, followed by the results in Chapter 5. Chapter 6 presents the concept for traceability for our system along with a literature review of the automated test case generation techniques. User manual is provided in the Chapter 7, followed by the conclusion.

# Analysis and Design of AV Software stack

For the selection and connection of an autonomous vehicle software stack with LGSVL and VTD simulators using a flexible software interface, the analysis of the current system architecture must be performed and the interface options must be explored. In the following sections, we will analyze these aspects and arrive to a decision with the help of certain criteria.

## Existing Interface Architecture

Let’s understand the existing interface architecture for our simulator and autonomous software stack system.

Robo-Test Web Application

VTD Connector

LGSVL Connector

VTD Simulator

LGSVL Simulator

ROS Interface

Apollo Cyber RT

Apollo 5.0

Apollo 6.0

Figure 2.1: Architecture of Simulators and Apollo AV software stack.

The architecture shown above consist of the VTD and LGSVL simulators connected to the Robo-Test web application using the individual connector components. The web application is a tool developed to generate test scenarios easily, without knowing the process to create scenarios on different simulators. The process and scenario creation details for the LGSVL simulator using the web application can be found in MT-3136 [4], whereas the details for VTD are in MT-3183 [5].

Both the simulators are connected to different versions of Apollo software stack using different interfaces. The LGSVL connects to Apollo 5.0 using the Cyber-RT based interface and the VTD connection is using ROS interface to Cyber-RT developed in MT-3192 [6]. The Cyber-RT layer is a common layer that connects both the simulators to Apollo. To better understand this, let’s look at these concepts in more details.

### Apollo AV software stack

Apollo is an open-source, high performance, flexible software architecture developed by Baidu for development, testing and deployment of autonomous vehicles [7]. It provides all the AV software stack components like localization, perception, prediction, planning, control and its own HD map format specifications. The Apollo versions upto 3.0 support the ROS interface to connect any external component. Since Apollo 3.5, Baidu has developed a robust Cyber-RT layer that takes care of Apollo’s interactions with the external components. So, the versions starting from 3.5 don’t support a ROS interface with Apollo directly as there is a Cyber-RT layer in the middle.

### Apollo Cyber-RT

The Apollo Cyber-RT is a robust framework and offers developers with a high-performance computing to support complex tasks of autonomous driving [8]. It enables a high concurrency of execution, low latency and high throughput [8]. The newer versions of Apollo have improved since the introduction of Cyber-RT making it one of the most robust open-source AV software stacks available. Integration of external software components with Cyber-RT framework is easy due to its plug-and-play architecture.

## Interface options between Simulators and Apollo

As per the current architecture of our system, we have Apollo versions 5.0 and 6.0 which come with a Cyber-RT layer on top of it. Thus, any interface for our simulators with Apollo must interact with Cyber-RT layer to establish communication with Apollo. The analysis of different possibilities of interface with Apollo is discussed in this section.

### Cyber RT Interface

One way of connecting both the simulators to Apollo AV software stack is to interface them directly with Apollo’s Cyber-RT layer. The LGSVL simulator offers this functionality and offers support for a connection with Cyber-RT. In the architecture that we showed, the LGSVL is interfaced to Apollo using Cyber-RT without any additional middleware component. The LGSVL simulator offers four types of bridges where each bridge has its own message format – ROS, ROS Apollo, ROS2 and Cyber-RT [9]. Using the Cyber-RT message format, it is convenient to exchange data between both the components. Let us look at a couple of important factors that must be considered for the interface between the simulators and the AV software stack with respect to our system.

#### Addition of a new Simulator

What if a new simulator must be integrated into our system? A standardized middleware interface layer will ensure the flexibility to accommodate a new simulator. The table below shows a few simulators and the interface support that they provide.

Table 2.1: Simulators and Interface Support.

|  |  |  |
| --- | --- | --- |
| Simulator | Type | Interface Support |
| LGSVL | Open Source | ROS, ROS2, ROS Apollo and Cyber-RT. |
| VTD | Proprietary | ROS and ROS2. |
| Carla | Open Source | ROS and ROS2. |

Table 2.1 indicates that the VTD simulator does not explicitly offer any support for Cyber-RT interface data communication. Being a proprietary simulator, there are quite a few limitations on the exchange of data into and out of VTD. Since there is no Cyber-RT message format supported in VTD, there is a need of an additional layer to convert the messages into Cyber format and vice versa. Also, most simulators provide a ROS/ROS2 based interface support for further data exchange.

#### Migration to a new AV software stack

How about the software interface if a migration to a new AV software stack is desired? The following table contains different AV software stacks and the interface support they provide to communicate with the external components.

Table 2.2: AV stacks and Interface Support.

|  |  |  |
| --- | --- | --- |
| AV Stack | Type | Interface Support |
| Baidu Apollo | Open source | Apollo Cyber-RT |
| Autoware | Open source | ROS migrating to ROS2 |
| Nvidia Drive | Proprietary | NVIDIA Driveworks SDK |

In case of our system, the possibility of migration to Autoware AV stack is high as it is an open-source AV stack and supports open-source ROS based interface for data exchange. Based on the inputs from the above tables, a ROS/ROS2 based interface looks a better option. Thus, the idea of having a single Cyber-RT interface that handles data exchange for both the simulators is not possible due to the factors analyzed above. Moreover, the VTD doesn’t support a Cyber-RT type message exchange, so designing a full Cyber-RT interface is not an option.

### ROS or ROS2 Interface

The analysis so far suggests that it is better to design a ROS/ROS2 based software interface to connect the LGSVL and the VTD simulators to Apollo. There already exists a ROS interface between the VTD and the Apollo but the system is not fully functional with this interface. Also, the LGSVL is connected to Apollo using its Cyber-RT message support. Thus, we explore the possibility of a ROS2 interface between both the simulators and the Apollo AV software stack by migrating and enhancing the existing ROS interface. A comparison between ROS and ROS2 is presented based on certain features.

Table 2.3: Comparison between ROS and ROS2 [10].

|  |  |  |
| --- | --- | --- |
| Feature | ROS | ROS2 |
| Multi-machine application | Set one machine as ‘master’. | No ‘master’ required. Independent working. |
| ROS Master | Required. | No more necessary. |
| Parameter Server | Global parameter server. | Local parameters to nodes. |
| Underlying Middleware | Central discovery mechanism. | Based on DDS standard. |
| Nodes in a process | One node per process. | Multiple nodes per process. |
| Roslaunch | XML | Python (Complex logics) |
| Language Support | C++ 03 and Python 2. | C++ 11,14 and Python 3.5 or greater. |

Table 2.3 compares ROS and ROS2 on a few features and ROS2 prevails better due to its distributed mechanism, support for modern C++ and Python for application development, multi-master capabilities, multiple node support in a process and good support for a next few years depending on the version.

## Problems with VTD connection to Apollo

## Decision on AV Software

# System Model

# Bibliography

|  |  |
| --- | --- |
| [1]  [2]  [3]  [4]  [5]  [6]  [7]  [8]  [9]  [10] | C. Ebert and M. Weyrich, "Validation of Autonomous Systems," in IEEE Software, vol. 36, no. 5, pp. 15-23, Sept.-Oct. 2019, doi: 10.1109/MS.2019.2921037.  N. Kalra and S. M. Paddock, “Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability?,” *Transportation Research Part A: Policy and Practice*, vol. 94, pp. 182–193, 2016, doi: <https://doi.org/10.1016/j.tra.2016.09.010>.  “Apollo.” <https://apollo.auto/devcenter/devcenter.html> (accessed Nov. 27, 2021).  J. John, “Development of a tool for the collection, prioritization and application of test cases,” Institute of Industrial Automation and Software Engineering, University of Stuttgart, Stuttgart, 2020.  Lili Ji, “Modernization of a Simulator for the Improvement of Autonomous Driving Simulation Environment”, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2021.  Trieu Tach Vu, “Modernizing the autonomous vehicle software stack”, Institute of Industrial Automation and Software Engineering, University of Stuttgart, 2021.  *ApolloAuto/apollo*. Apollo Auto, 2021. Accessed: Nov. 30, 2021. [Online]. Available: <https://github.com/ApolloAuto/apollo>  “Apollo.” <https://apollo.auto/cyber.html> (accessed Nov. 30, 2021).  “Supported message types - LGSVL Simulator.” <https://www.svlsimulator.com/docs/archive/2020.06/simulator-messages/> (accessed Dec. 01, 2021).  V. Mazzari, “ROS vs ROS2,” *Génération Robots - Blog*, Dec. 17, 2019. <https://www.generationrobots.com/blog/en/ros-vs-ros2/> (accessed Dec. 02, 2021). |

# Declaration of Compliance Style Template: Überschrift 1 / Headline 1 (delete the number)

I hereby declare to have written this work independently and to have respected in its preparation the relevant provisions, in particular those corresponding to the copyright protection of external materials. Whenever external materials (such as images, drawings, text passages) are used in this work, I declare that these materials are referenced accordingly (e.g. quote, source) and, whenever necessary, consent from the author to use such materials in my work has been obtained.

Signature:

Stuttgart, on the <DD.MM.YYYY>