KI WISSEN
Automotive AI Powered by Knowledge

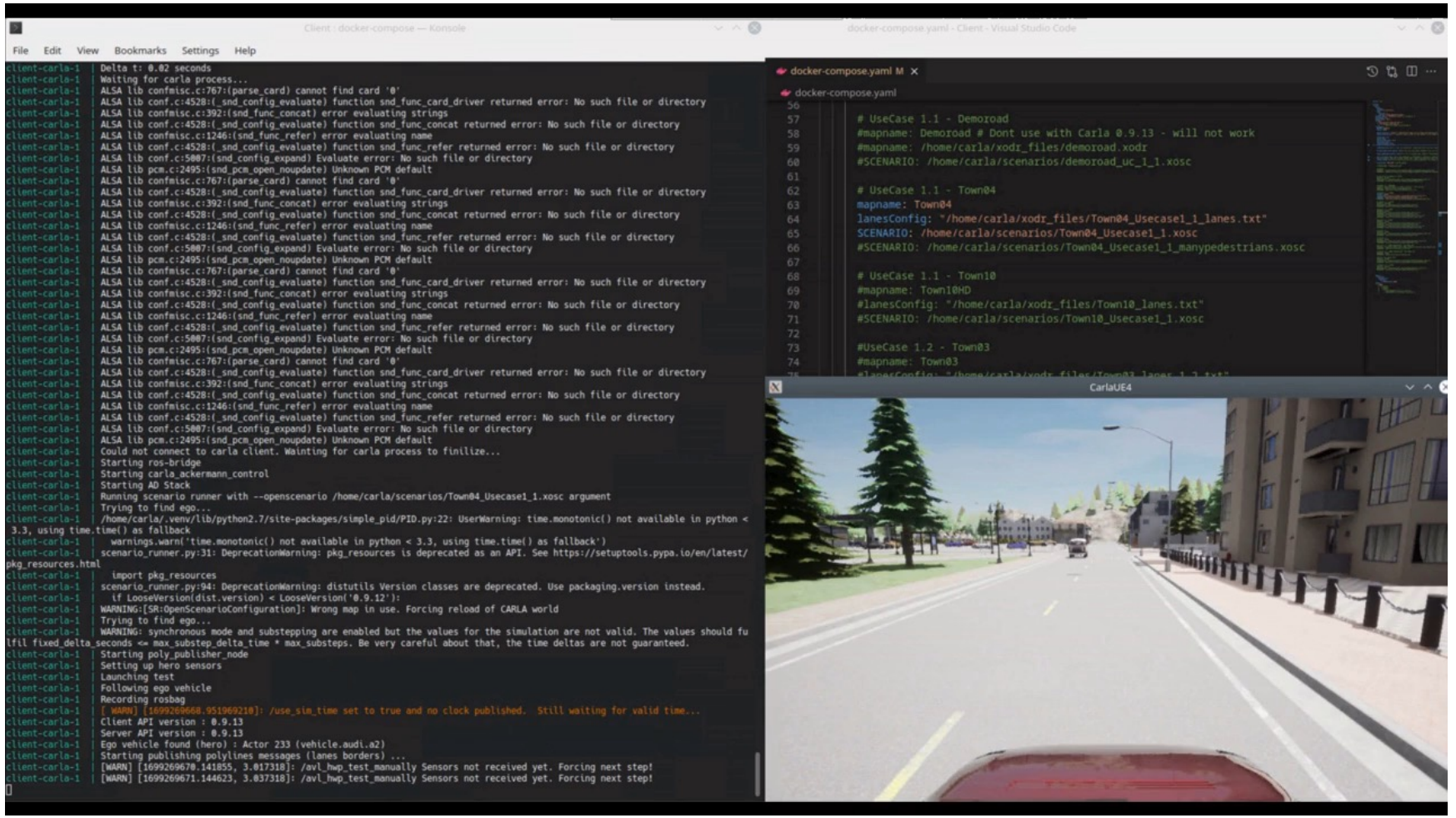
KI Wissen Final Event | 21-22 March 2024

TP4 Demonstrator

Thomas Knorr | AVL

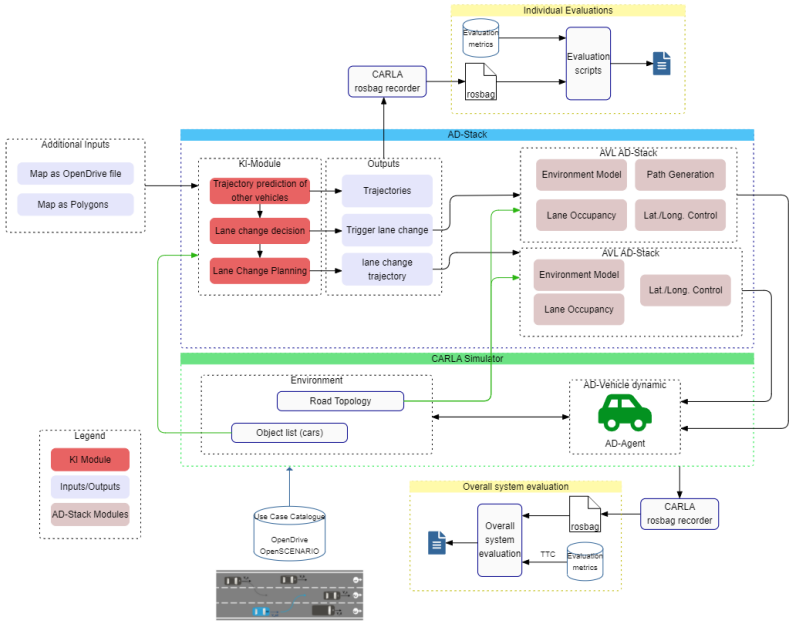
Demonstrator Example

```
SimPC:~$ docker-compose up
```

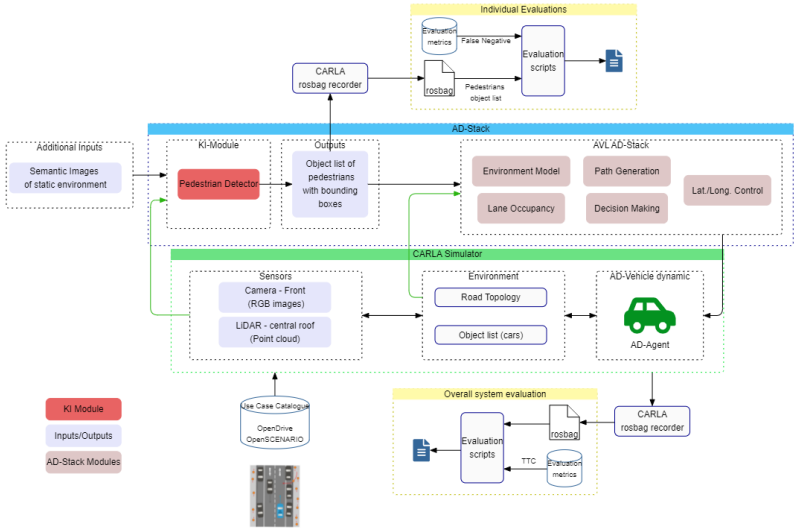


Demonstrator Requirements

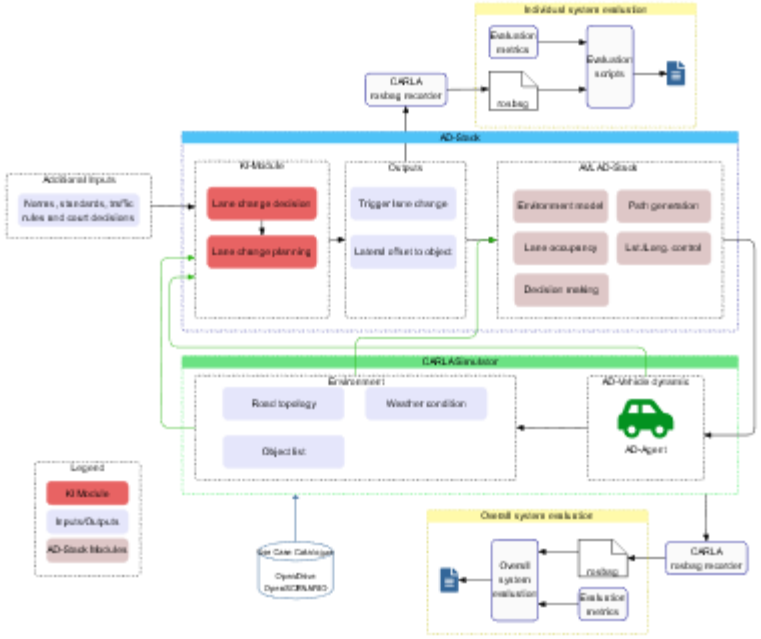
- UC1 – Pedestrian Detection



- UC2 – Lane Change

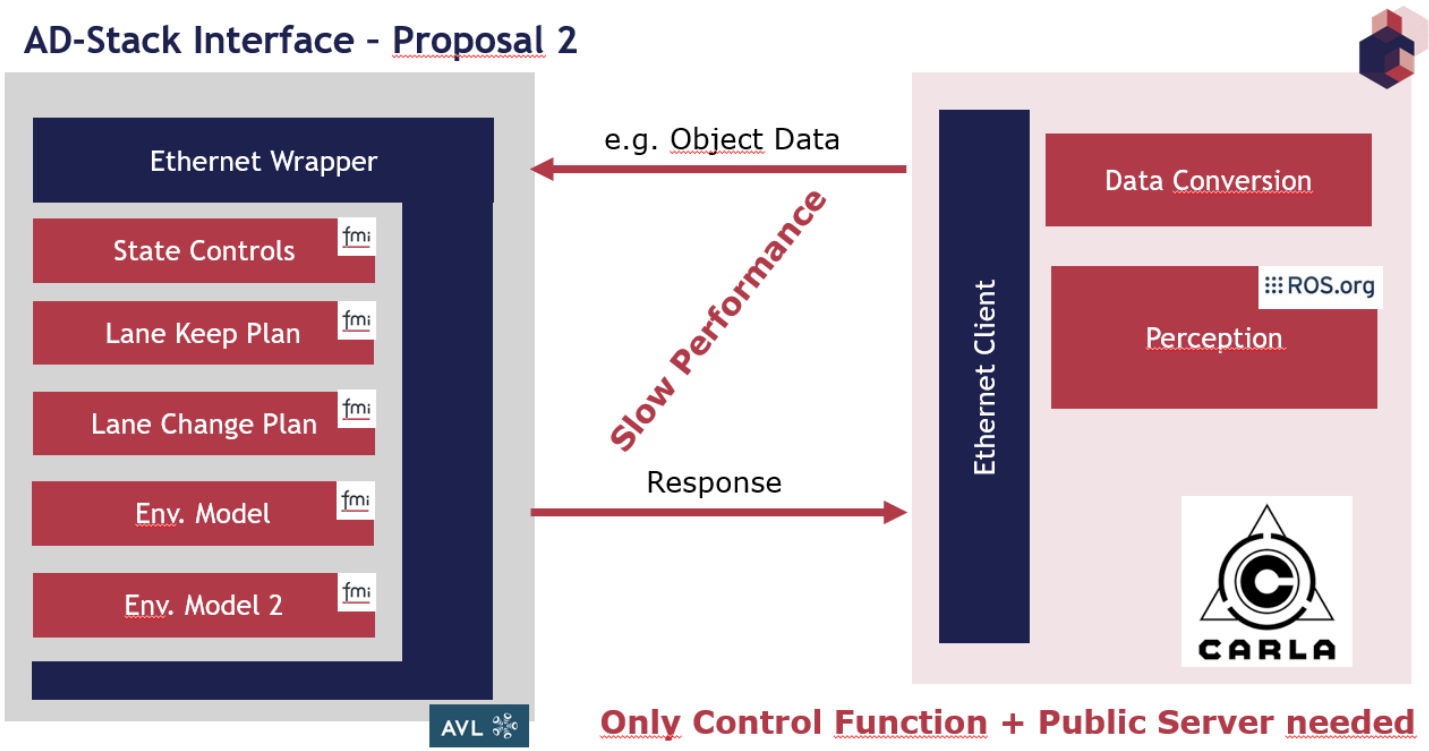


- UC3 – Controlled Rule Exception



E4.2.3:
https://confluence.vdali.de/display/KIWissen/Generic+AD+Architecture

Demonstrator Concept(s)



E4.2.2:
https://confluence.vdali.de/display/KIWissen/Demonstrator%27s+Interface+Architecture+between+AD-Stack+and+developed+modules+and+methods

Demonstrator System Architecture

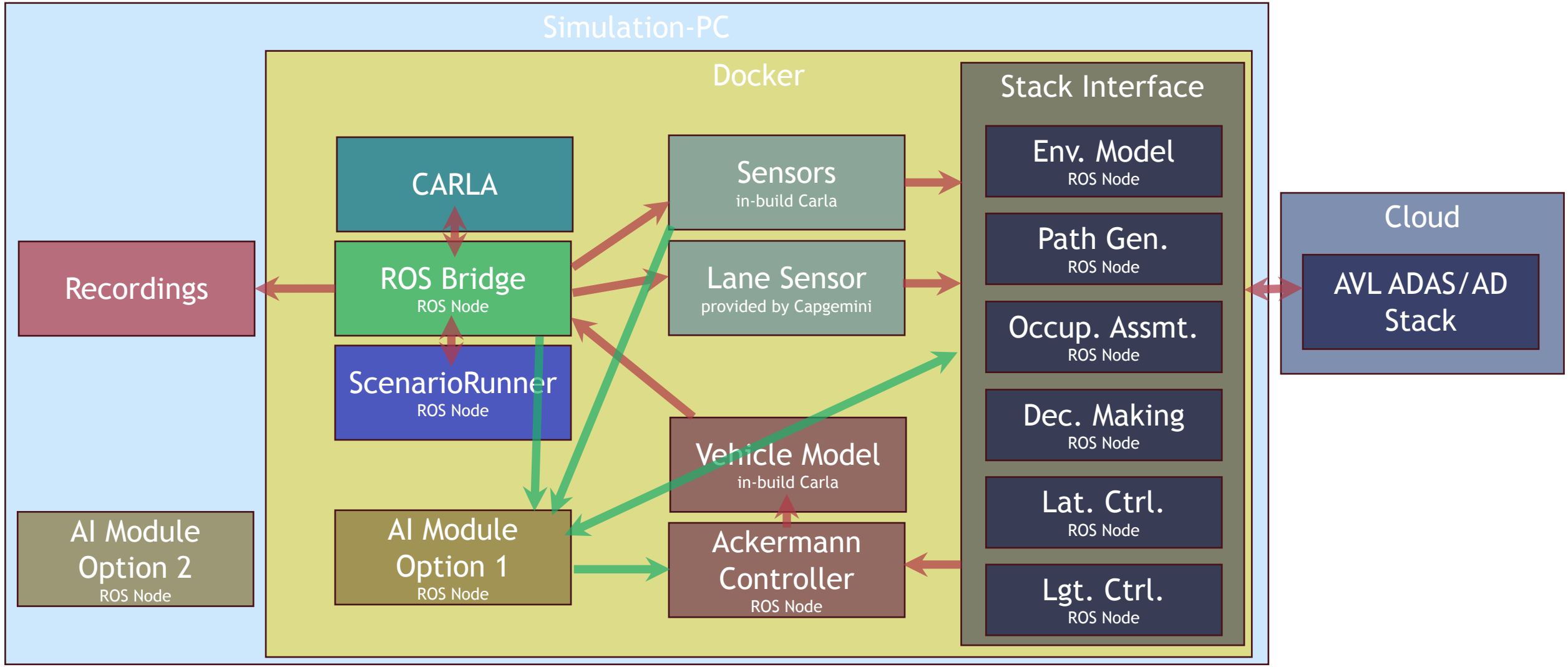
Overview / Remarks

- CARLA toolchain used for scenario, environment and sensor simulation
- Docker used to make it easy-to-use and still flexible
- ROS used as middleware
- All CARLA sensor can be used and setup flexible
- Recording can be done automatically.
- Simulation is running in simulated real-time
- AVL ADAS/AD-Stack is running in the cloud and each component is interfaced as independent ROS-Node
- The ADAS/AD-Stack can be run in two modes:
  - Combined: Every inital input will be send as once and will internally distributed to the modules on the cloud server. After everything is calculated for one step the output will be send back again as once.
  - Splitted: Every signal is send and can be accessed and modified independently per module.

E4.3.2
https://confluence.vdali.de/display/KIWissen/E4.3.2+2.Demonstrator+Implementation

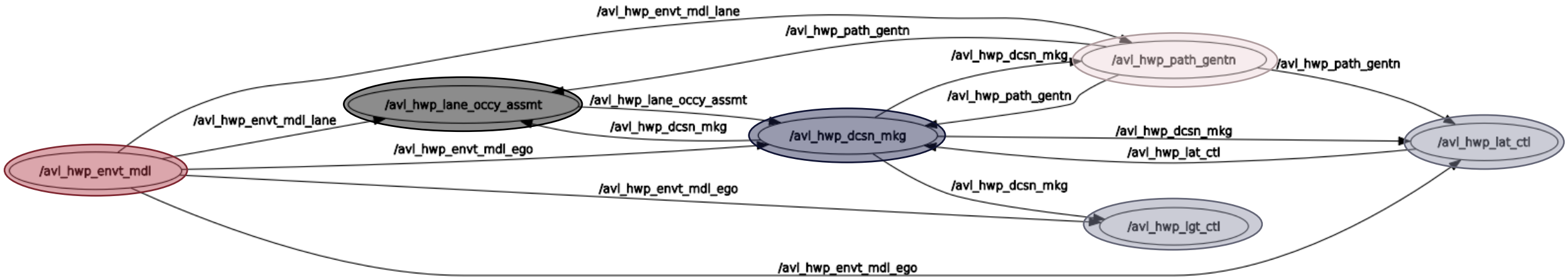
Demonstrator System Architecture

Diagram



Demonstrator System Architecture:

ROS Nodes & Topics

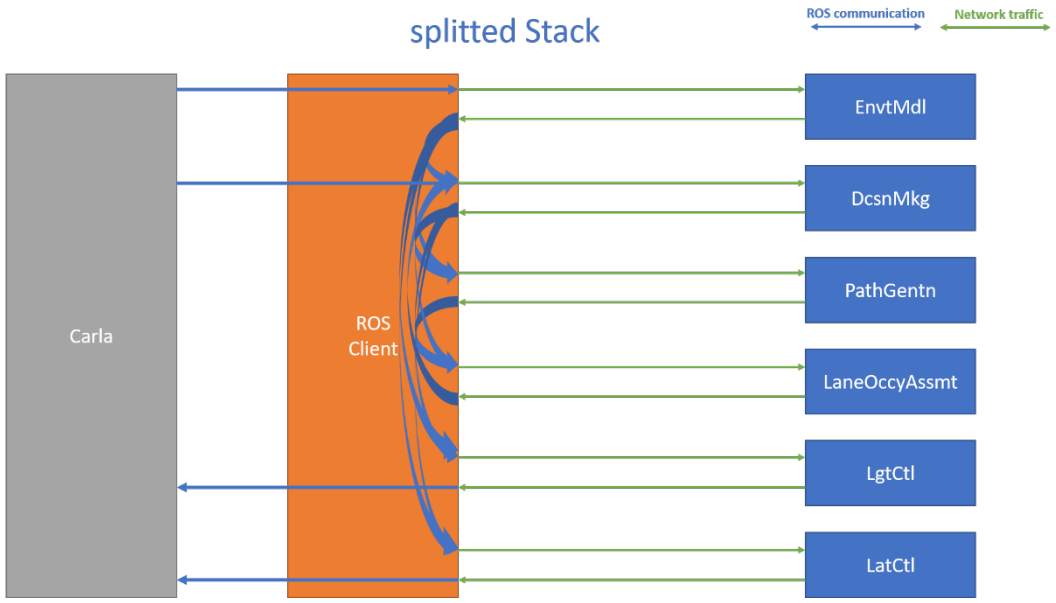


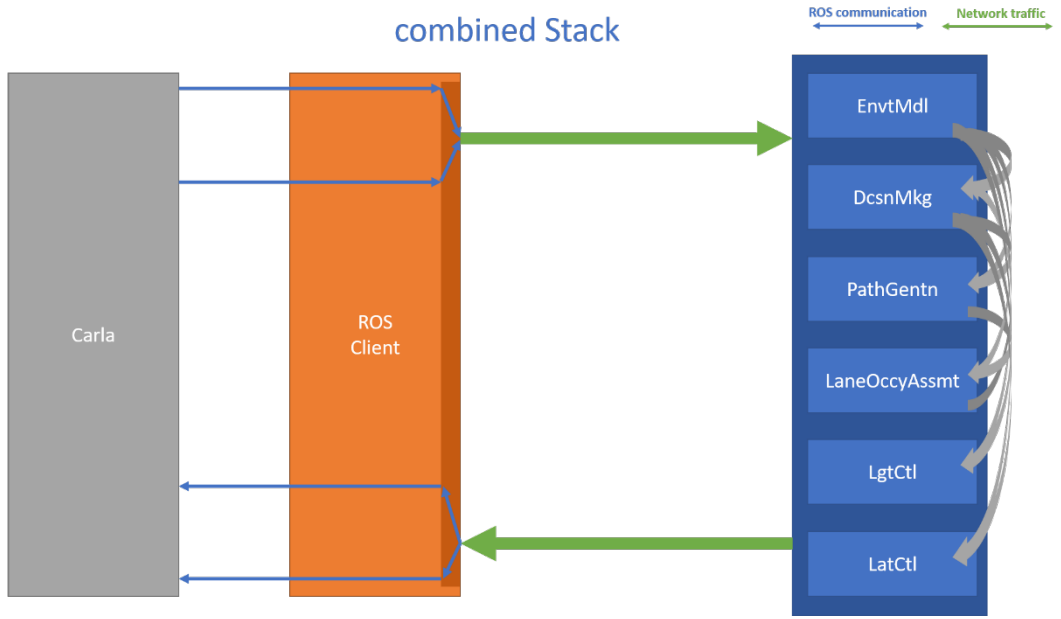
https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp/-/blob/master/Documentation/ROS_Stack_Graph.png

Demonstrator System Architecture:
Splitted Stack vs Combined Stack

- Manipulation of signals possible
- Replacement of individual functions is possible
- Full control of the signals between each component
- Recommended for Use Case 2 + 3

- Faster execution
- Easy integration due to limited Input/Outputs
- Recommended for Use Case 1





Basic Parameterization & Options

- Simulation setup
  - module_converter
  - module_launchscript
- Scenario setup
  - mapname
  - laneConfig
  - scenario
- Recording options
  - exit_on_scenario_end
  - exit_on_timeout
  - record_rosbag

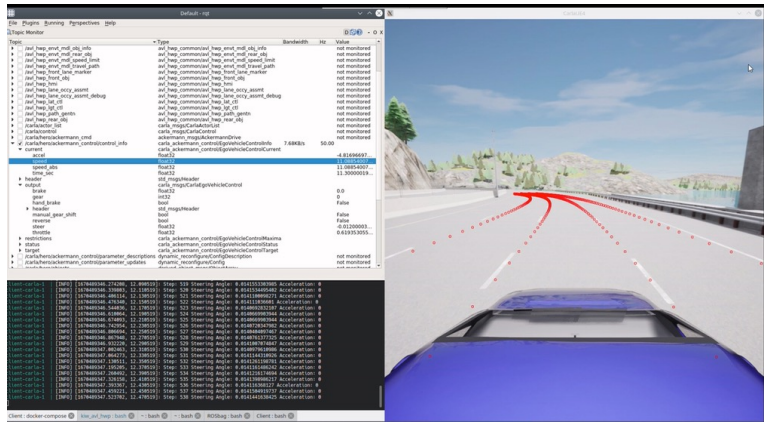
docker-compose.yaml

```yaml
module_converter: "… && rosrun avl_hwp_test_manually main.py"
module_launchscript: "… && rosrun myCompany_myModule main.py"

# UseCase 1.1 - Town04
mapname: Town04
lanesConfig: "/home/carla/xodr_files/Town04_Usecase1_1_lanes.txt"
scenario: /home/carla/scenarios/Town04_Usecase1_1.xosc

...

# Usecase 2.5
#mapname: Town04
#lanesConfig: "/home/carla/xodr_files/Town04_Usecase2_5_lanes.txt"
#SCENARIO: /home/carla/scenarios/Town04_Usecase2_5.xosc

...
...
...

exit_on_scenario_end: false
#exit_on_timeout: 120
record_rosbag: false
```

Advanced Parameterization & Options

- follow_ego *true (default value)*
- debug *false (default value)*
- use_splitted_stack *false (default value)*
- only_carla *false (default value)*
- visualize_paths *false (default value)*
- setup_hero_sensors *false (default value)*

follow_ego: *true (default value)*
debug *false (default value)*
use_splitted_stack *false (default value)*
only_carla ***true***
visualize_paths *false (default value)*



follow_ego: *true (default value)*
debug ***true***
use_splitted_stack: ***true***
only_carla: *false (default value)*
visualize_paths ***true***

https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp/-/tree/master/Client

Scenario Parameterization & Options

Example based on UseCase 1.1

- <ParameterDeclaration name="pedestrian_speed" parameterType="double" value="1.5"/>
- <ParameterDeclaration name="pedestrian_start_distance" parameterType="double" value="35.0"/>

Scenario: Town04_Usecase1_1.xosc

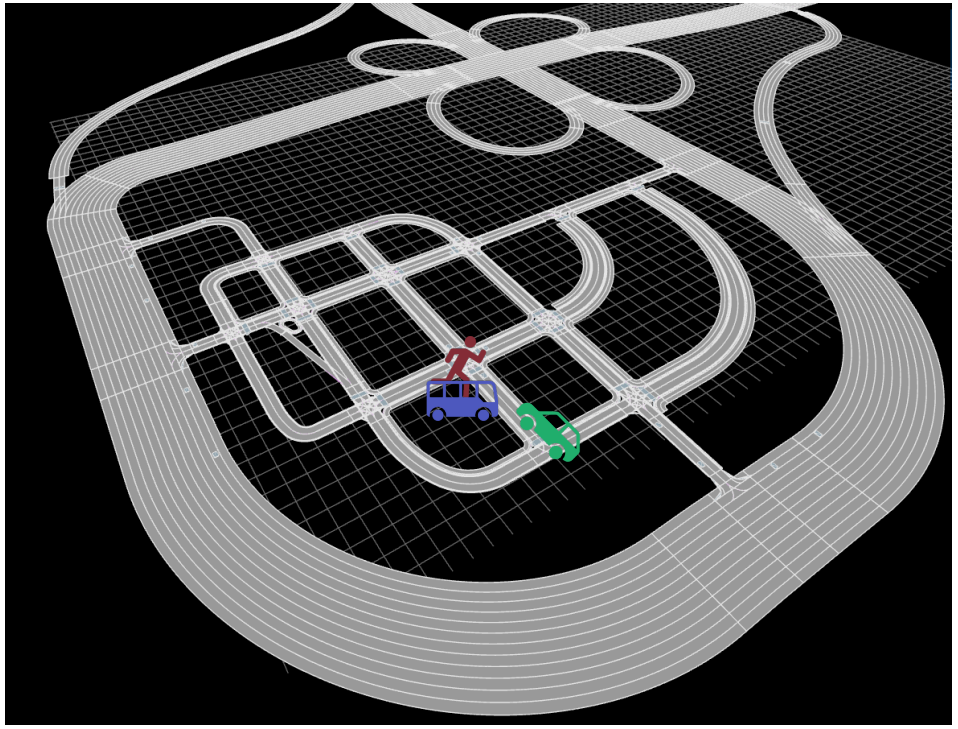
Default Parameters: 1.5m/s, 35m

Explicite overshooted Parameters: 10m/s, 10m

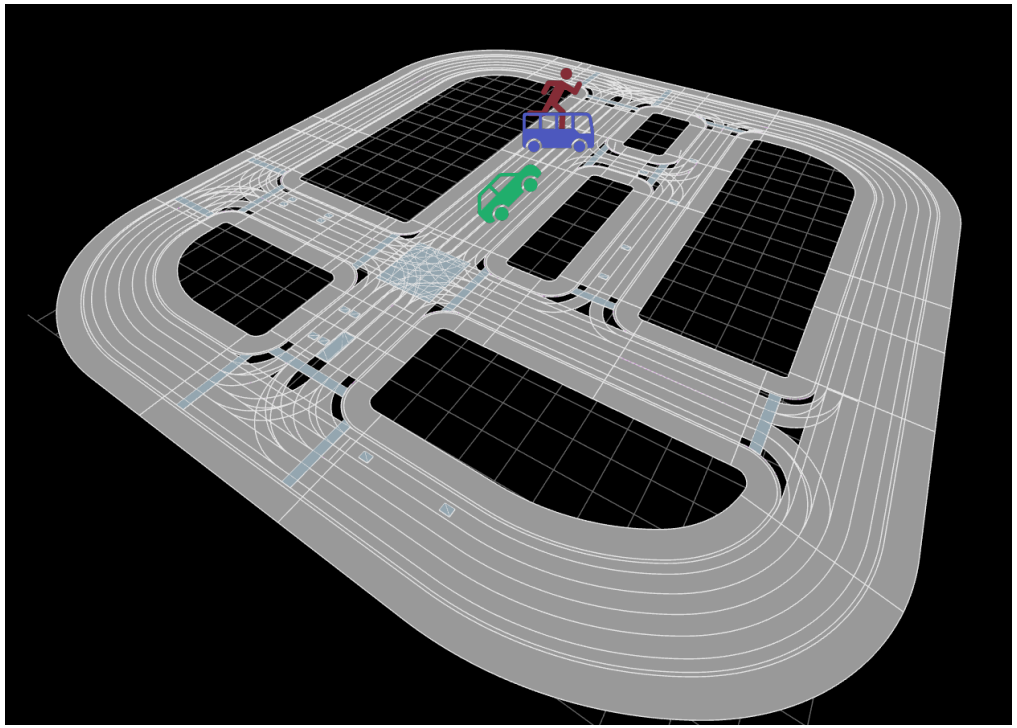
Scenario Repositioning (Deutsch: „Verortung“)

Example based on UseCase 1.1

Goal: Modify UseCase 1.1 Scenario to use Town10 instead of Town04



Town04.xodr



Town10.xodr

Special needs for the demonstrator

- Trajectory Prediction feature
  - Option: „numOfPredictions“
  - Multiple instances of „DcsnMkg“, „LgtCtl“, „LatCtl“ modules of the AD-Stack will be created, most critical prediction will be applied and forwarded to the vehicle model as input.
- Scenario options
  - Option „use_ego_speed_from_scenario“
  - Ego speed can be taken from scenario file or from internal configuration.
- Modification of Map / Import of special assets
  - Option: „import_additional_assets“
  - Own Maps (xodr) and Assets (e.g. special traffic signs) can be imported.
  - Maps need to be pre-compiled by an special carla version compiled from source

Occured problems and solutions using the CARLA toolchain

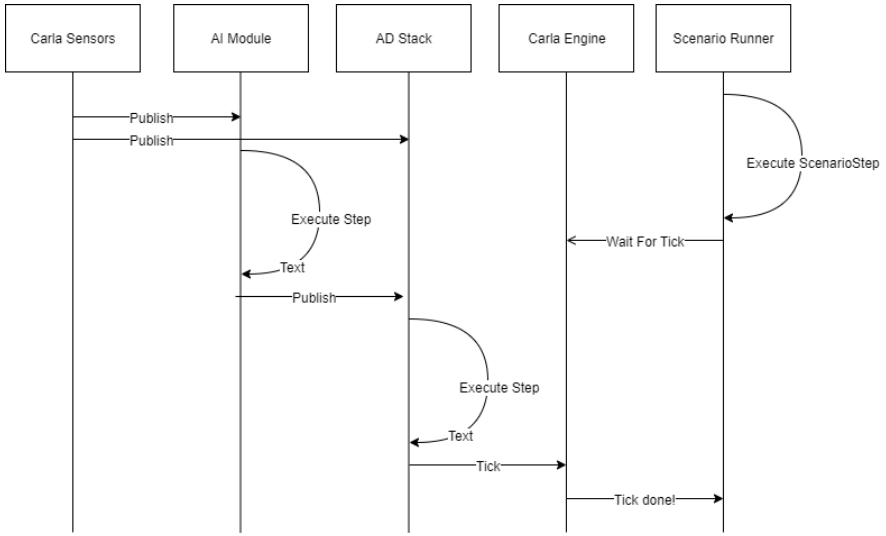
- Use of own maps
  - CARLA does not support all 3D-Models. No straight forward structure which models are valid or not.
  - Maps need to be compiled (very long compile time) and tested manually after compilation. No detailed error message which asset causing issues. CARLA just crashing with fatal unknown error.
- Modification of maps
  - Internal maps in CARLA need to be overwritten inside the container. Only internal fixed pathes given by Carla are valid.
- ScenarioRunner
  - Only part of OpenScenario is supported.

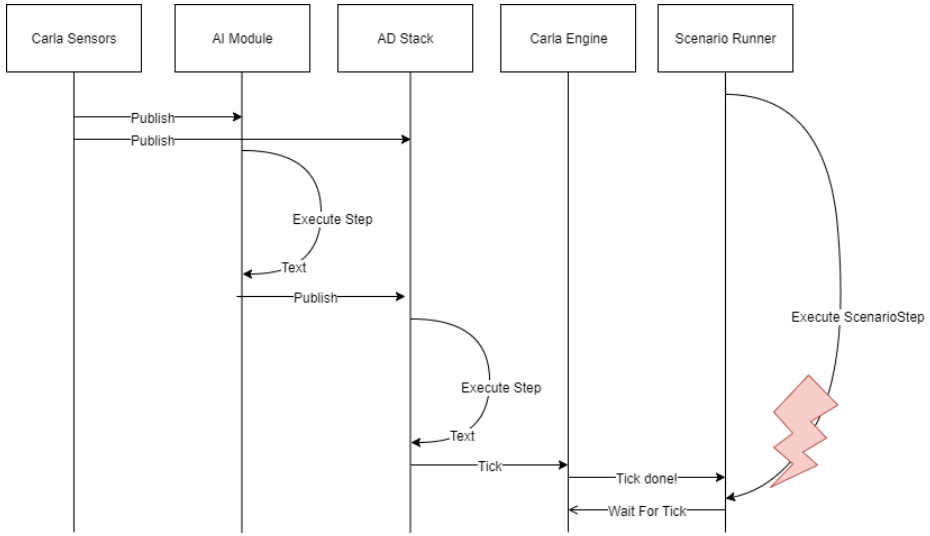
    (Even parts which are mentioned as suppored are not working, e.g. private trajectory)
  - In general very buggy

    (Description need to be set with special strings. e.g. „CARLA:“, Start time can‘t be true with given edge type, Units in parameter declaration will cause crash, Storys with special condition triggers are not getting executed, …)

Occured problems and solutions using the CARLA toolchain

- Vehicle Model
  - CARLA Vehicle Model is not very accurat.
  - Inputs varieties are very limitted. Special controller needed to be implmented.
- Synchronisation
  - Synchronisation algorithm/process provided by CARLA, Carla-Ros-Bridge and Scenario Runner is buggy.
  - Each sensor and actor needs to be synced manually





KI WISSEN

Automotive AI Powered by Knowledge

Thomas Knorr | AVL | Thomas.Knorr@avl.com

KI Wissen is a project of the KI Familie. It was initiated and developed by the VDA Leitinitiative autonomous and connected driving and is funded by the Federal Ministry for Economic Affairs and Climate Action.

KI FAMILIE

VDA LEITINITIATIVE

Funded by the European Union
NextGenerationEU

Supported by:
Federal Ministry for Economic Affairs and Climate Action
on the basis of a decision by the German Bundestag

www.kiwissen.de @KI_Familie KI Familie

AD Stack Interfaces

Introduction

- Abstraction of Carla and AD Stack using Docker Container

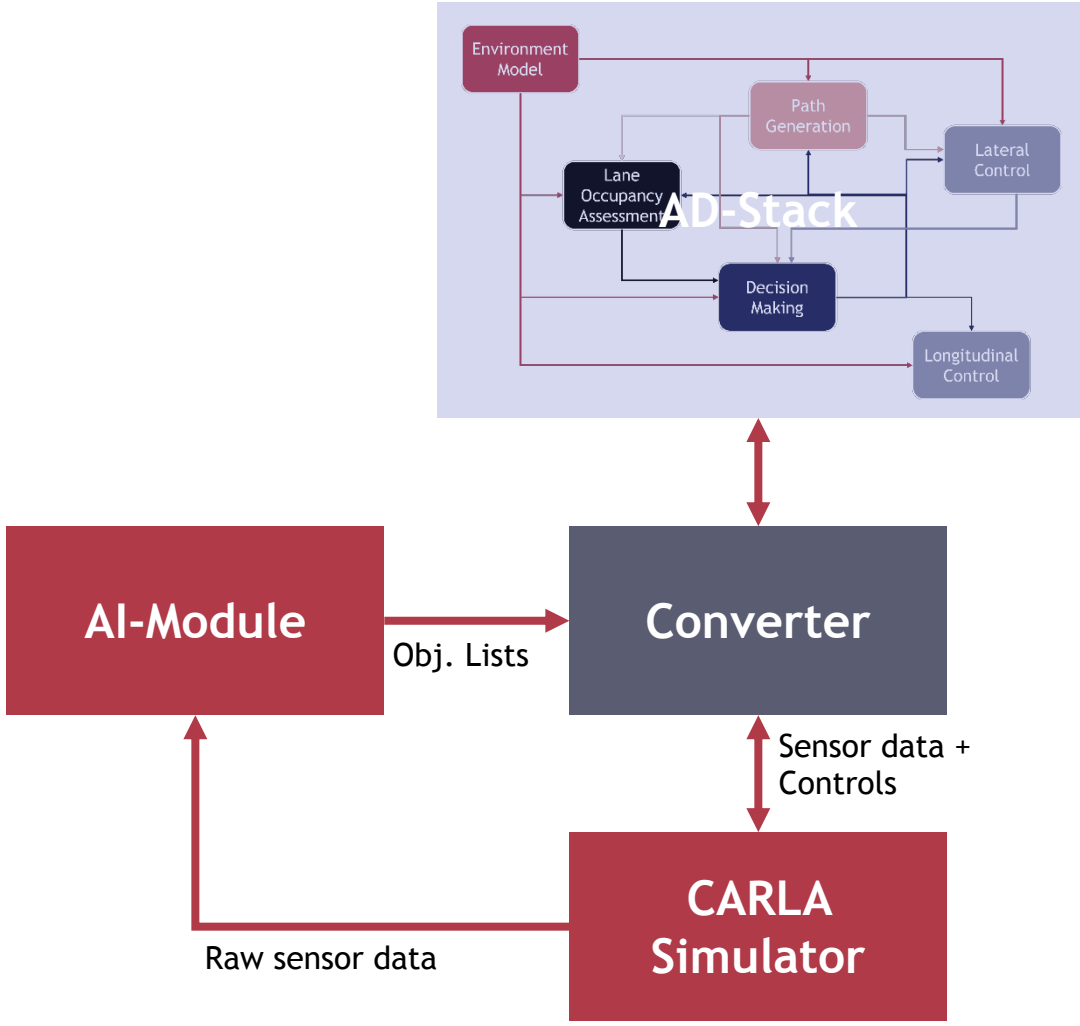
Reasons

- Easy to install / use
- Alligned setup between all partners

Demonstration

➢ **Live Demo**

- See: https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp



Docker Container - Limitations & Prerequirements

- Preferred Operating System: Ubuntu 18
- Network connection needs to connect to AVL AD Stack (Port 443)

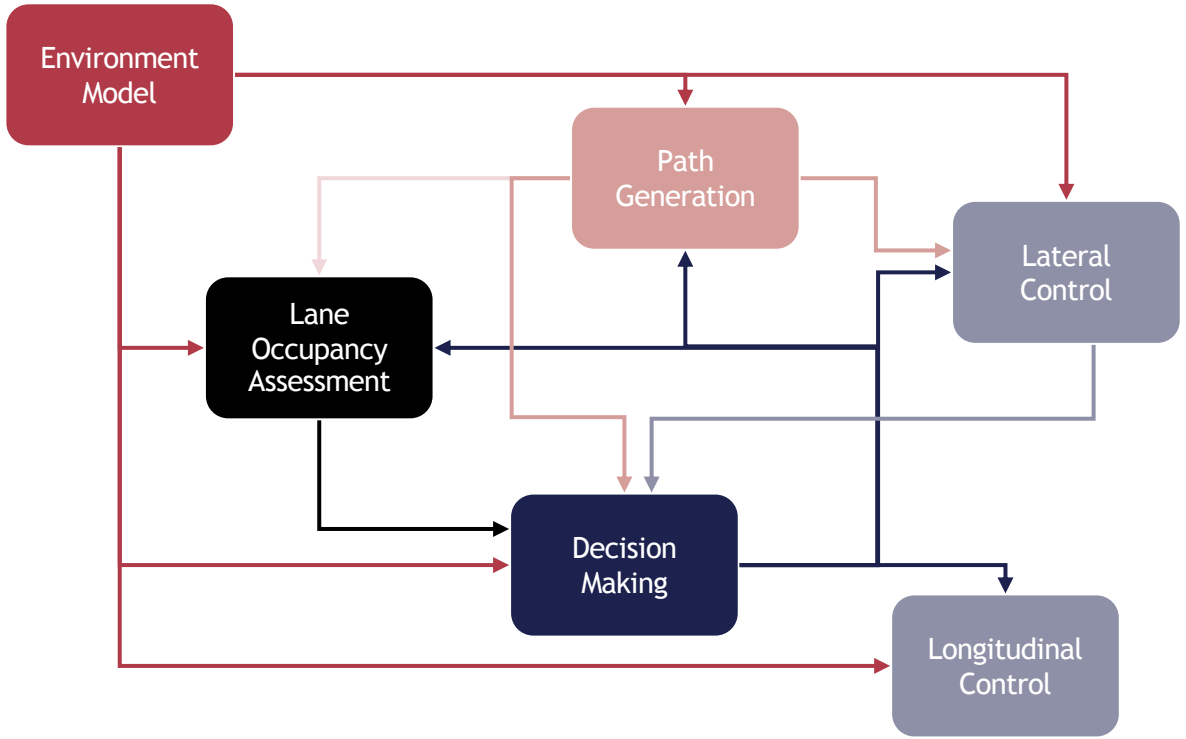
- Demonstrator is **not** designed for data learning
  - Its designed for executing scenario-based test cases for validation

- Demonstrator is running in simulated real-time, not physical real-time
  - Execution Performance is highly depending of CPU and Graphic Card
    (High End Simulation PC strongly recommended)

https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp/-/tree/master/Client

AD Stack – Background and Block Diagram

- Deterministic Highway Pilot Stack
- SAE L2+ Automation level
- Controls Modules only (perception not included)
- Features:
  - ACC
  - LKA
  - LCA
- Best compatibility with UC2 scenarios



https://confluence.vdali.de/display/KIWissen/AVL+Control+AD+Stack+-+Interface+Description

AD Stack Interfaces

Introduction

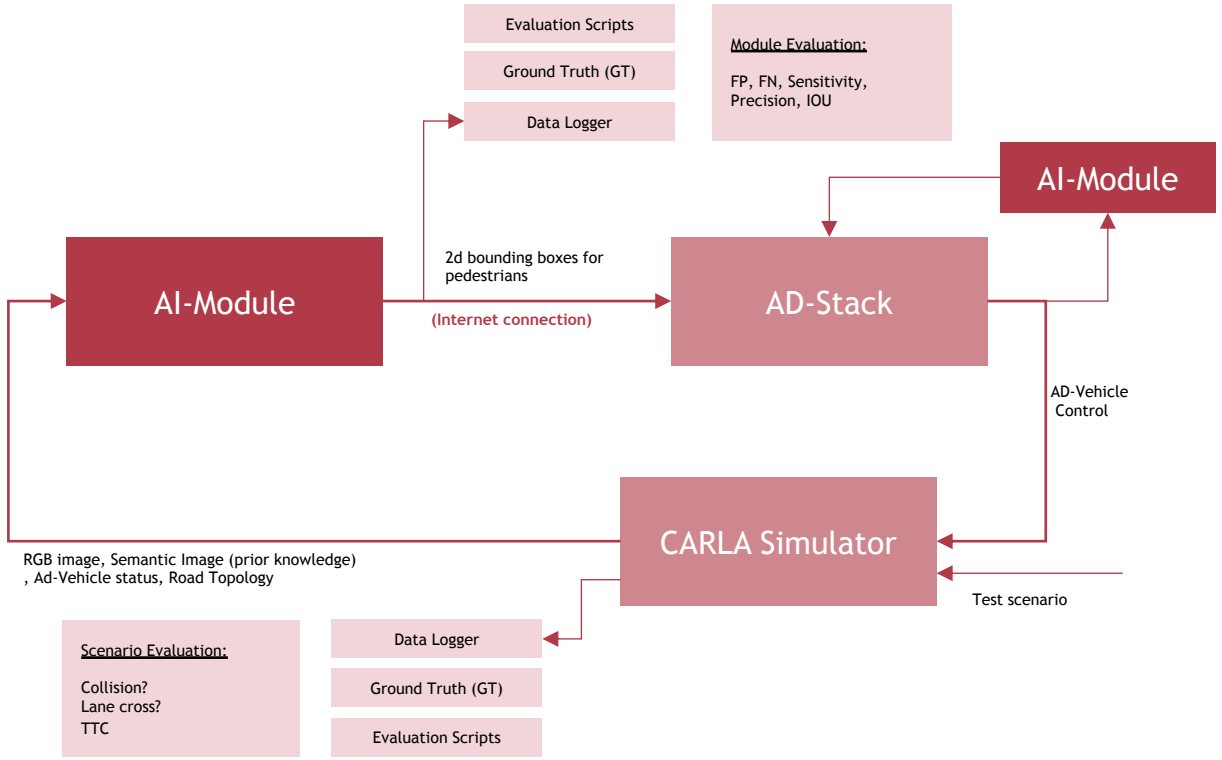
- Controls AD Stack running on public cloud server
- Accessible via Websocket
- Each module is abstracted as an ROS Node

Reasons

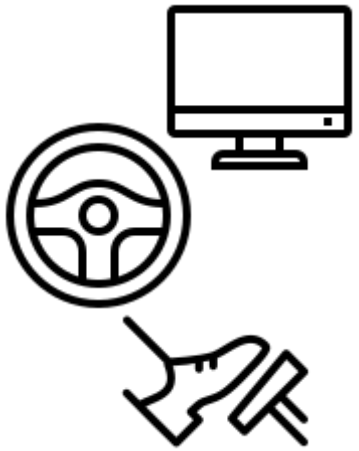
- AVL is not able to share source code
- ROS as aligned middleware for KI Wissen

Demonstration

- Link to Guidelines:
- See: https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp



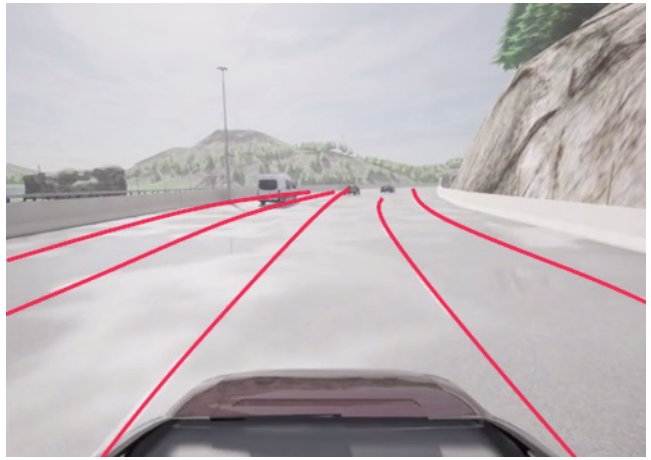
Interface Description - Vehicle Informations & HMI

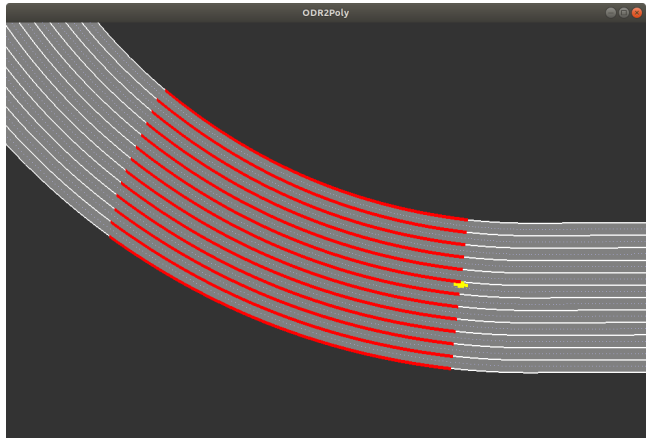


Signal Name	Description	Unit
ego_accx_in	Ego acceleration x	m/s^2
ego_delta	Ego yaw rate	°/s
ego_velx	Ego velocity x	m/s

Signal Name	Description	Unit	Recommended Value
Hmi_VDesrd	Cruise control vehicle set speed. Desired velocity, selected by driver	m/s	20 (0-33)
CC_PosnBrkPedl	Driver input brake pedal position	%	0
CC_PosnThrPedl	Driver input throttle pedal position	%	0
CC_StBlnkrAcvLe	Status left blinker active	-	0
CC_StBlnkrAcvRi	Status right blinker active	-	0
CC_StCrsCtlAdpv	Status cruise control adaptive	-	1
CC_StCtlLat	Status latitude control	-	1
CC_StCtlLgt	Status longitude control	-	1
CC_StsLca	Status lane change assist	-	1

Interface Description - Front Camera (line detection)





Signal Name	Description	Unit
lane_marker_???_c0	line coefficient c0. Lateral distance to vehicle ref. point and dir	m
lane_marker_???_c1	line coefficient c1. Yaw angle relative to vehicle dir	rad
lane_marker_???_c2	line coefficient c2. Horizontal curvature	1/m
lane_marker_???_c3	line coefficient c3. Change of horizontal curvature	$1/m^2$
lane_marker_???_quality	line quality	%
lane_marker_???_range	line range. Distance of last valid measurement	m
lane_marker_???_type	line type.	0=no_line, 1=durchgezogen, 2=gestrichelt, 3=Bot_Dots, 4=Fahrbanrand, 5=Leitplanke, 6=Bordstein, 7=Mauer, 8=Bitumenfuge, 9=Sonstige, 10=not_defined

Interface Description – Front/Rear Sensor (object detection)

Signal Name	Description	Unit
obj_length_in	Detected objects lengths	m
obj_raccx_in	Detected rear objects x accelerations	m/s^2
obj_rvelx_in	Detected objects relative x velocity	m/s
obj_type_in	Detected objects types	NONE 0, PLAYER_NONE 0, PLAYER_CAR 1, PLAYER_TRUCK 2, PLAYER_VAN 3, PLAYER_BIKE 4, PLAYER_PEDESTRIAN 5, PLAYER_PED_GROUP 6, POLE 7, TREE 8, BARRIER 9, OPT1 10, OPT2 11, OPT3 12, PLAYER_MOTORBIKE 13, PLAYER_BUS 14, STREET_LAMP 15, TRAFFIC_SIGN 16, HEADLIGHT 17, PLAYER_TRAILER 18, BUILDING 19, PARKING_SPACE 20, ROAD_WORKS 21, ROAD_MISC 22, TUNNEL 23, LEGACY 24, VEGETATION 25, MISC_MOTORWAY 26, MISC_TOWN 27, PATCH 28, OTHER 29, SEMI_TRAILER 30, RAILCAR 31, RAILCAR_SEMI_HEAD 32, RAILCAR_SEMI_BACK 33, VEH_LIGHT_FRONT_LEFT 34, VEH_LIGHT_FRONT_RIGHT 35, VEH_LIGHT_REAR_LEFT 36, VEH_LIGHT_REAR_RIGHT 37, VEH_CABIN 38
obj_width_in	Detected objects widths	m
obj_x_in	Detected objects x positions	m
obj_y_in	Detected objects y positions	m

Example: Feed Front Radar with data from object list

https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp/-/blob/master/Client/avl_hwp_ws/src/avl_hwp_test_carla/scripts/main.py

```
…
rospy.Subscriber("/carla/hero/objects", ObjectArray, set_object_callback)
…
def set_object_callback(data):
    for i in range(len(data.objects)):
        obj = data.objects[i]

        pos = obj.pose.position
        rel_pos = np.matmul(ego_mat_inv, np.array([[pos.x],[pos.y],[pos.z],[1]]))

        if (rel_pos_front[i] is None):
            rel_pos_front[i] = rel_pos

        rel_vel = (rel_pos - rel_pos_front[i]) / TDELTA
        rel_pos_front[i] = rel_pos

        data_avl_hwp_front_obj.obj_id_in[i] = obj.id           # Unique ID of the detectected vehicle
        data_avl_hwp_front_obj.obj_rvelx_in[i] = rel_vel[0]     # Speed of the detected vehicle
        data_avl_hwp_front_obj.obj_x_in[i] = rel_pos[0][0]      # X Distance relative to ego vehicle
        data_avl_hwp_front_obj.obj_y_in[i] = rel_pos[1][0]      # Y Distance relative to ego vehicle
        data_avl_hwp_front_obj.obj_type_in[i] = 1               # Type - need to be mapped, X = car
…
```

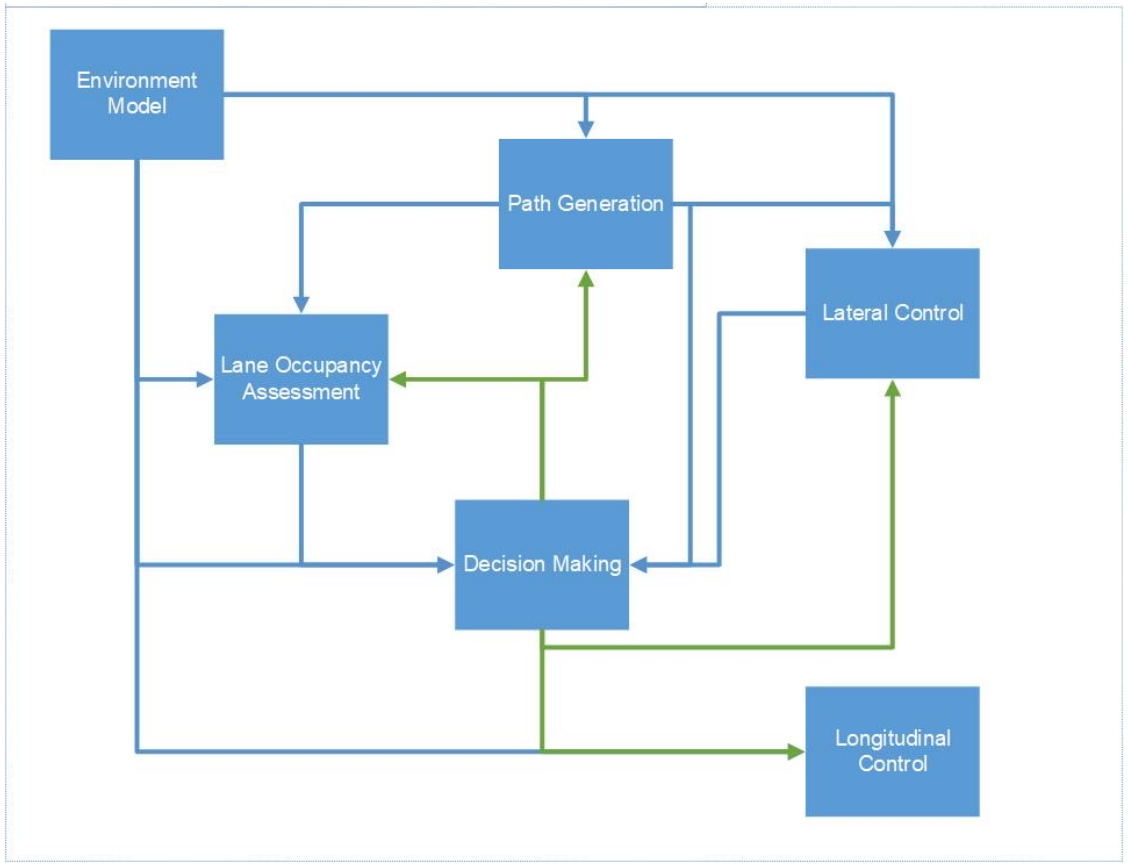
Example: Feed Front Radar with raw data

```
…
rospy.Subscriber("/carla/camera/rgb", CarlaRawData, my_callback)
…
def my_callback(data):
    # data = e.g. point cloud or rgb raw data

    # !!! Your Code - Fetching the AD-Stack relevanted informations from carla data

    data_avl_hwp_front_obj.obj_rvelx_in[i] = ...    # Speed of the detected vehicle
    data_avl_hwp_front_obj.obj_x_in[i] = ...        # X Distance relative to ego vehicle
    data_avl_hwp_front_obj.obj_y_in[i] = ...        # Y Distance relative to ego vehicle
    data_avl_hwp_front_obj.obj_type_in[i] = ...     # Type - need to be mapped, X = car
…
```

Internal Block Diagram

- **All AD-Stack components are connected / communicating via ROS topics/messages**
- **Alle components can be individually exchanged**

- Environment Model
- Lane Occupancy Assessment
- Path Generation
- Decision Making
- Longitudinal Control
- Lateral Control



Tool Versions

Python: 3.7

Carla: 0.9.12+

ROS: ROS1 Melodic (Python 2.7)

Docker: 19+

Links

- Getting Started (HowTo)
  - https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp/-/blob/master/ReadMe.md#howto-run-ros-environment-of-the-stack
- AD Stack Archtecture
  - https://confluence.vdali.de/display/KIWissen/Generic+AD+Architecture
- Interface Description
  - https://confluence.vdali.de/display/KIWissen/AVL+Control+AD+Stack+-+Interface+Description
  - https://gitlab.com/ki-wissen/tp4/ap4.2/kiw_avl_hwp/-/blob/master/ReadMe.md#message-explaination-for-inputsoutputs-of-the-whole-stack
- Insights
  - https://confluence.vdali.de/pages/viewpage.action?spaceKey=KIWissen&title=E4.3.1+-+Demonstrator+PoC