INTEGRATING ECLIPSE KUKSA WITH AN OPENADX TOOLCHAIN
Eclipse OpenADx WG
OpenADx Toolchain

- A community effort to enable compatibility between toolchain components for implementing autonomous driving functions
Create a **cross-vendor** connected vehicle platform that relies on **open standards** and uses **open source software** to leverage the potential of a **large developer community**!

Eclipse Kuksa

Vision
Eclipse Kuksa
Ecosystem
**Eclipse Kuksa**

**Kuksa In-Vehicle Platform**

**Application layer:**
- Runs 3rd party apps on the platform
- Contains a Sandbox Environment & Additional Services

**Middleware layer (Yocto layer):**
- APIs to abstract the vehicles’ E/E architecture (W3C VISS, Sensoris...)
- Communication Services to manage network access and provide data from the vehicle
- Includes communication libs, protocols, security layers,...

**OS layer:**
- Reuse of OE’s existing services, layers, HW abstractions, services, etc.
Eclipse Kuksa
Kuksa Cloud Platform

- App Store
- Big Data Analysis
- Visualization
- Report Generation
- Data Management
- Third Party Services
- Core Services
- Cloud Platform
- Keycloak
- (Automatic) Deployment
Eclipse Kuksa
Kuksa Val (Vehicle Abstraction Layer)

- Implements **Vehicle Signal Specification (VSS)** data model proposed by GENIVI
  - e.g. web-socket interface based on **W3C VISS (Vehicle Information Service Specification)** which relies on VSS data model

- VSS specifies a domain taxonomy for vehicle signals
  - 43 car attributes, 451 branches and 1060 signals currently

- Examples:
  - **Vehicle.OBD.Status.MIL** (Used by Malfunction Indicator Light (MIL) service)
  - **Vehicle.Engine.EOT** (engine oil temperature)

http://genivi.github.io/
Touchpoints between Eclipse Kuksa and OpenADx

- Ingest/Store:
  - Ingest data from In-Vehicle platform in model from Val and store in Cloud

- Test drive:
  - Use In-Vehicle platform for execution of actual function and test-drive aspects (e.g. data recording)
    [Kuksa.InVehicle]

- Optional Deep Learning:
  - Realization of Deep Learning component as specific function of general concept of function development and modeling for In-Vehicle applications and Cloud services
Touchpoints between Eclipse Kuksa and OpenADx
Proposed Integration

- Simulation and test / Simulation-based validation:
  - Assure functionality of developed Kuksa In-Vehicle applications and Cloud services by using simulations
  - Especially next-generation mobility services in the context of autonomous driving require sophisticated validation
  - Simulation of all relevant aspects of the physical world required:
    - Vehicles, bicycles, pedestrians, and further traffic participants
    - Traffic lanes, intersections, cross traffic, traffic rules etc.
    - Environmental conditions such as weather or daytime
Example Use Case
Kuksa Cloud Dashboard

Example Vehicle Signal Tree [1]

- engine
- body
- rpm
- speed
- mirrors
- doors
- weight
- fuel_type
- heated
- dimmed
- ...

Kuksa Cloud

- Grafana
- MQTT
- AMQP
- Cloud Dashboard Service

Kuksa In-Vehicle

- Hono MQTT Adapter

Simulation of Client Application to assure its functionality

[1] https://www.w3.org/TR/vehicle-information-service/
Integrating simulation steps
Architecture of proposal

1. Push code and test config file into, e.g. github repo
2. Triggers verification process component
3. Pulls code and test config from repo
4. Prepare input for Carla
5. Simulation via Carla
6. Result of simulation

App Publisher

Kuksa IDE

GitHub

Verification

Kuksa Cloud

Kuksa In-Vehicle

App Store

hawkBit

W3C API

In-Vehicle Platform

Deploy App on In-Vehicle Gateway via App Manager

App Deployment Process

Verification Process
Integrating simulation steps
Testing Kuksa In-Vehicle Applications

- Feed In-Vehicle application with sensor values from a simulated vehicle to assure functionality
- Enable assessment of software in early development phases

Challenges:
- Connecting Kuksa In-Vehicle platform within Carla
  - Carla-ROS-Bridge as blueprint
- Formal simulation scenario description (Test Config)
- Matching of Carla sensor values with, for example, VSS
- Only subset of scenarios are supported due to available sensor values

Kuksa In-Vehicle

- Recorder
  - Record produced data output of the In-Vehicle application
- Mapper
  - Mapping between Carla sensor values and VISS
- Connector
  - Verify that the recorded data match with the expected data
  - Retrieve scenario-specific sensor values from simulated vehicle using Carla API

- Verification

- W3C API
- Kuksa Dashboard
- Client
- App Manager
- Recorder
- Kuksa In-Vehicle Platform

- Get expected data schema
- Get Test Scenario parameters
Integrating simulation steps

Testing Kuksa Cloud Services

- Obtain sensor values from simulation and send it to the respective Kuksa Cloud service to assure functionality of service

- Challenges:
  - Providing mass data from simulation based on a vehicle fleet
    - Running co-simulation with SUMO
  - Integration of Command & Control, i.e. sending commands to the simulated vehicle to modify its behavior
  - Metrics to assure the functionality of the deployed Cloud service
    - E.g. measure scalability or detect Microservices Anti-patterns
QUESTIONS???

More Information:  https://www.eclipse.org/kuksa/
Mailing List:       kuksa-dev@eclipse.org

*Bi-weekly Zoom meeting every Thursday on even calendar weeks from 1-2pm (CET/CEST)*