Bridging the Gap between SUMO & Kuksa: Using A Traffic Simulator for Testing Cloud-based Connected Vehicle Services

*SUMO User Conference 2019*

**Philipp Heisig**¹, Sven Erik Jeroschewski², Johannes Kristan², Robert Hötterg¹, Ahmad Banijamali³, and Sabine Sachweh¹

¹ Dortmund University of Applied Sciences and Arts
² Bosch Software Innovations
³ University of Oulu
MOTIVATION
Connected Vehicles

Connected Vehicle Services

• Connected vehicles are expected to be the next frontier in automotive revolution
  • Extraction, storage, processing, analysis, and usage of vehicle data within the IoT
  • Integration of further data sources, e.g. Smart City or Smart Home

• Innovative and data-driven mobility services

• High degree of data-monetization and value creation based on disruptive business models

• Various application domains benefit from connectivity
  • Road safety
  • Smart, efficient, and green transportation
  • Location-dependent services

Source: https://www.bosch-mobility-solutions.com/
Connected Vehicle Testbed

- Scalable and reliable connected vehicle services required
  - Increasing number of connected vehicles on the road
  - Connected vehicles operate in a safety-critical and time-sensitive environment
- How to test the service functionality and ensure a suitable software architecture?
- Focus on implementation rather than setting up a test environment

Does my service scale with a large amount of vehicles?

Is my service reliable also under changing circumstances?

Do my service work as expected within different scenarios?
Traffic Simulators for Testing Connected Vehicle Services

- Testbed for evaluating the functionality of the according software components
  - Setting up a large number of hardware and vehicle nodes is not practical due to economical and operational constraints
  - Context-specific automotive data from real-word scenarios that goes beyond few rudimental or fake data-sets
- Using an appropriate traffic simulator for ...
  - ...proof-of-concept regarding the software architecture design
  - ...an evaluation of service functionality
- Simulation on microscopic level to provide vehicle-specific data
- Various reasons for using Eclipse SUMO
  - Designed for microscopic simulations
  - Supports large road networks and the modeling of intermodal traffic systems
  - Well established in the research community
  - Fosters real-world scenarios from areas such as Luxembourg, Bologna, Cologne, or Monaco
  - Provides a lot of modeling tools and APIs
ECLIPSE KUKSA
APPSTACLE

• **APPSTACLE**: open standard Application Platform for carS and TrAnsportation vehicles
  - Pave the way for connected driving

• Development of an Open Source Connected Vehicle Ecosystem
  - Open source automotive IoT Cloud Platform
    - Architectural considerations for the cloud platform
    - Establishment of standardized interfaces to the vehicle
  - Service enablers for car-to-cloud connectivity
    - Network infrastructure considerations
    - Next generation mobile networks
  - Open source in-vehicle platform
    - Safe and secure gateway to the cloud
    - In-vehicle data access mechanism and application platform

• **Eclipse Kuksa**: Open Source project that host the developed connected vehicle technology stack
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**!
Bridging the Gap between SUMO & Kuksa

Eclipse Kuksa
Bridging the Gap between SUMO & Kuksa

Eclipse Kuksa

Third Party Services

Core Services

App Store

Big Data Analysis

Visualization

Report Generation

Data Management

(Automatic) Deployment

Core Services

Keycloak

Hawkbit

Ditto

Hono

InfluxDB

Grafana

mongoDB
CONNECTING SUMO WITH KUKSA
Connecting Eclipse SUMO with Eclipse Kuksa

Bridging the Gap between SUMO & Kuksa
Connecting SUMO with Eclipse Kuksa

```python
def connect_to_message_gateway():
    client.reinitialise(client_id="python_client", clean_session=True, userdata=None)
    client.username_pw_set(username, password)
    client.on_connect = on_connect
    client.connect(host, port, 60)
    client.loop_start()

def run():
    step = 0
    while traci.simulation.getMinExpectedNumber() > 0:
        traci.simulationStep()
        for vehicleID in traci.vehicle.getIDList():
            co2emission = traci.vehicle.getCO2Emission(vehicleID)
            noxemission = traci.vehicle.getNOxEmission(vehicleID)
            x, y = traci.vehicle.getPosition(vehicleID)
            longitude, latitude = traci.simulation.convertGeo(x, y)
            json_data = json.dumps({'Vehicle_ID': vehicleID, 'Latitude': latitude, 'Longitude': longitude,
                                      'CO2_Emission': co2emission, 'NOx_Emission': noxemission})
            client.publish(topic_to_publish, json_data, 0, False)
            step += 1
    traci.close()
    sys.stdout.flush()
```
TESTING CONNECTED VEHICLE SERVICES
Integration of SUMO into the Kuksa application development process

1. Service implementation
   ◦ Example service: *Air Quality Monitor*
   ◦ Visualizes the air pollution for certain areas on a real-world heatmap
   ◦ Java-based Spring application
   ◦ InfluxDB as underlying database

```
VehicleDTO
- time: Instant
- vehicleID: String
- speed: Double
- latitude: String
- longitude: String
- noxEmissions: Double
- co2Emissions: Double

InfluxDBClient
- influxDB: InfluxDB
- dbName: String

getVehicleData(query: String): List<VehicleDTO>

VehicleDataController
+ getCurrentVehicleEmissions(): List<VehicleDTO>
+ getAggregatedVehicleEmissions(interval: int): List<VehicleDTO>
```
Integration of SUMO into the Kuksa application development process

2. Setting up the test environment
   - Eclipse SUMO 1.1.0 (Windows 10 with Intel Core i7-5600U CPU at 2.60GHz and 16GB RAM)
   - Python 3.6.4
   - Scenarios: Monaco and TAPASCologne
   - Eclipse Kuksa Cloud within Azure Kubernetes Service (Eclipse Hono 0.8)
   - Air Quality Monitor runs on Ubuntu 16.04 VM
Integration of SUMO into the Kuksa application development process

3. Testing and enhancing the service
   ◦ Developer gets early feedback in the development process

• Different problems occurred with initial design of the Air Quality Monitor service:
  ◦ Database schema
    ◦ Initial not designed for querying large data sets in an efficient and flexible way
    ◦ Improved the response time of the service
  ◦ Presentation
    ◦ Map gets unclear and overcrowded with larger amounts of vehicles
    ◦ Changed the weighting of points dynamically
  ◦ API usage
    ◦ Heatmap was not updated correctly
    ◦ Changing vehicle data from running simulation ease debugging
SUMMARY AND OUTLOOK
Future Work

• Implementation and evaluation with different simulation scenarios and more sophisticated connected vehicle services
  • Consideration of Command & Control
  • Integration of other data sources, e.g. from the smart city

• Supporting real-time to test safety-critical services regarding responsiveness and reaction time
  • Record data and use time-stamps
  • What about Command & Control?

• Considering changing connectivity for end-to-end scenarios between vehicles and cloud backend
  • VSimRTI

• Investigation how to integrate with existing test processes, pipelines, and frameworks
Cooperation Possibilities

• Eclipse Kuksa Open Source project
  • Contribute with own ideas and development
  • Use and try the software
  • Be part of the development community from the beginning

• APPSTACLE Advisory board
  • Advice and proof of current development
  • Getting up to date results
  • Be part and shape the open source results
  • Cooperation based on open source solutions

• More information needed? https://www.eclipse.org/kuksa/