Eclipse SDV Blueprints

a collaborative initiative
to bring the "software defined vehicle" concepts to life

Naci Dai, eteration a.s
Sven Erik Jeroschewski, robert bosch gmbh

The Eclipse SDV Blueprints Project

collaborative initiative

- Eclipse SDV members
- SDV Concepts in action!
- Sandboxes for project collaboration and feedback

Eclipse SDV Blueprints

- Project hosts a collection of blueprints
- Standalone Exemplary

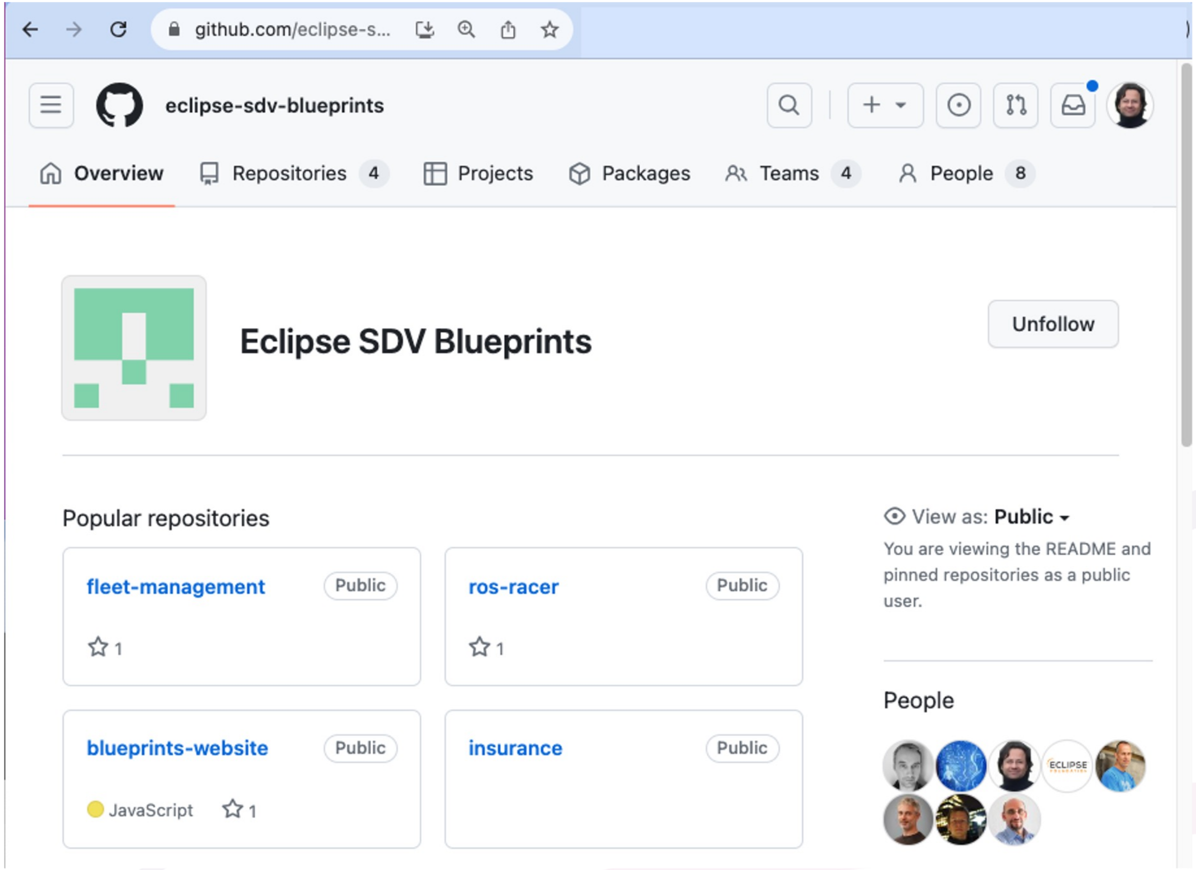
Learning

- Showcases for SDV technologies
- Blueprints facilitate self-learning
- Easily reproduce and run
- Well documented

Governance

An Eclipse Project

- Committers
- Github Repositories
- Issue Tracking
- Periodic Releases



github.com/eclipse-s...

eclipse-sdv-blueprints

Overview | Repositories 4 | Projects | Packages | Teams 4 | People 8

Eclipse SDV Blueprints

Unfollow

Popular repositories

fleet-management Public ☆ 1

ros-racer Public ☆ 1

blueprints-website Public ● JavaScript ☆ 1

insurance Public

View as: Public

You are viewing the README and pinned repositories as a public user.

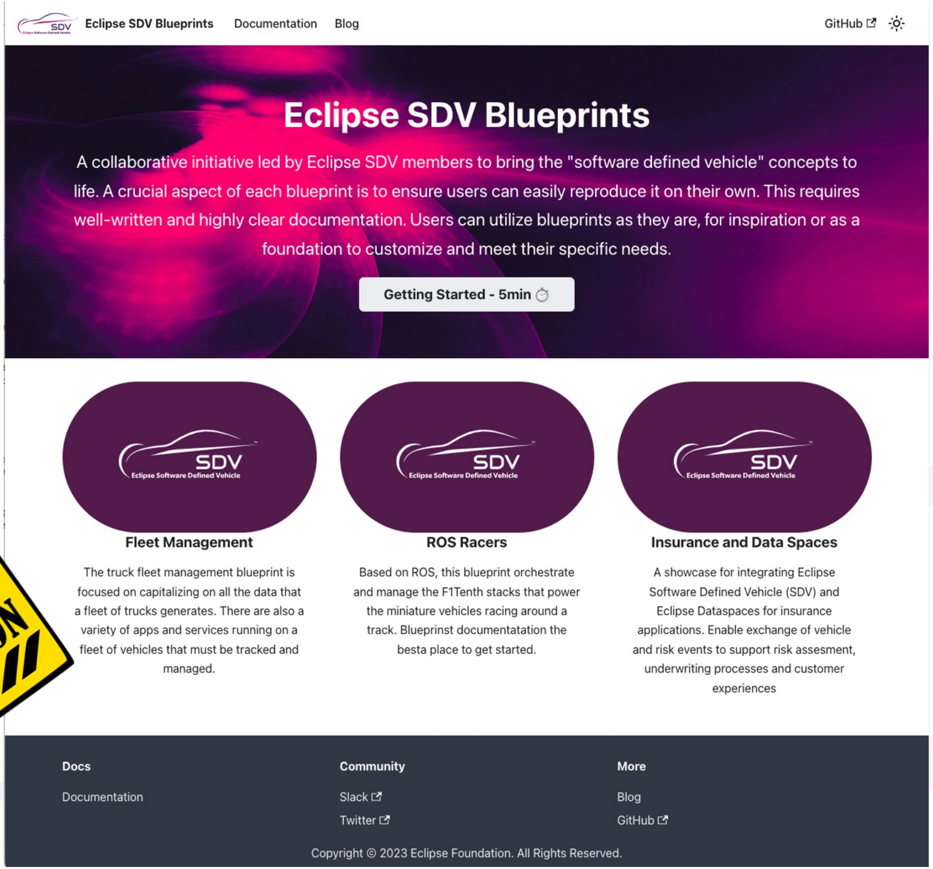
People

Documentation

Upto date

- ○ Blueprints is a learning resource
- ○ Documents and tutorial
- ○ Best practices and advice
- ○ Blueprints blog

UNDER CONSTRUCTION



Eclipse SDV Blueprints Documentation Blog GitHub

Eclipse SDV Blueprints

A collaborative initiative led by Eclipse SDV members to bring the "software defined vehicle" concepts to life. A crucial aspect of each blueprint is to ensure users can easily reproduce it on their own. This requires well-written and highly clear documentation. Users can utilize blueprints as they are, for inspiration or as a foundation to customize and meet their specific needs.

Getting Started - 5min

Fleet Management

The truck fleet management blueprint is focused on capitalizing on all the data that a fleet of trucks generates. There are also a variety of apps and services running on a fleet of vehicles that must be tracked and managed.

ROS Racers

Based on ROS, this blueprint orchestrate and manage the F1Tenth stacks that power the miniature vehicles racing around a track. Blueprinst documentatation the besta place to get started.

Insurance and Data Spaces

A showcase for integrating Eclipse Software Defined Vehicle (SDV) and Eclipse Dataspaces for insurance applications. Enable exchange of vehicle and risk events to support risk assesment, underwriting processes and customer experiences

Docs
Documentation

Community
Slack
Twitter

More
Blog
GitHub

Copyright © 2023 Eclipse Foundation. All Rights Reserved.

The Eclipse SDV Blueprints Project

collaborative initiative

- Eclipse SDV members
- SDV Concepts in action!
- Sandboxes for project collaboration and feedback

Eclipse SDV Blueprints

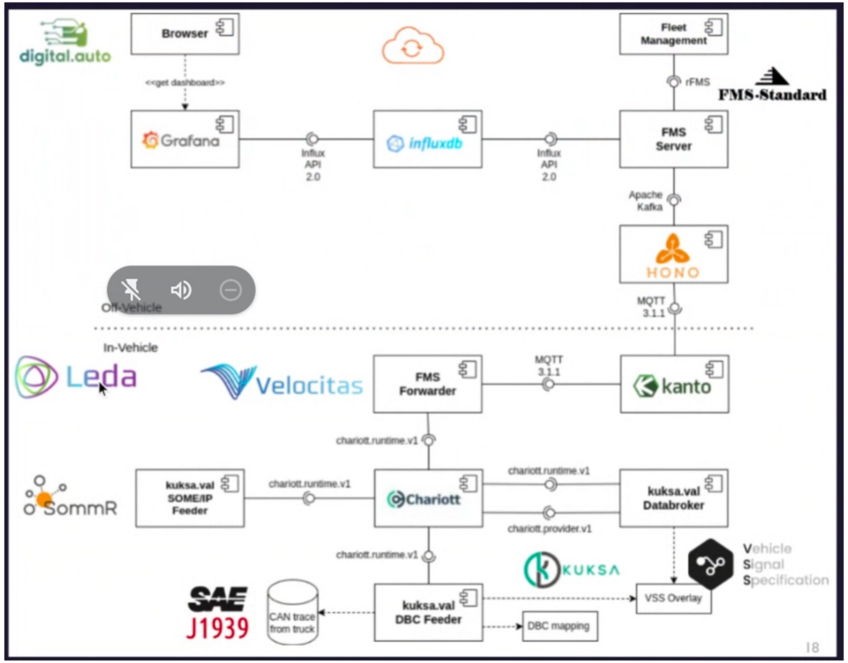
- Project hosts a collection of blueprints
- Standalone Exemplary

Learning

- Showcases for SDV technologies
- Blueprints facilitate self-learning
- Easily reproduce and run
- Well documented

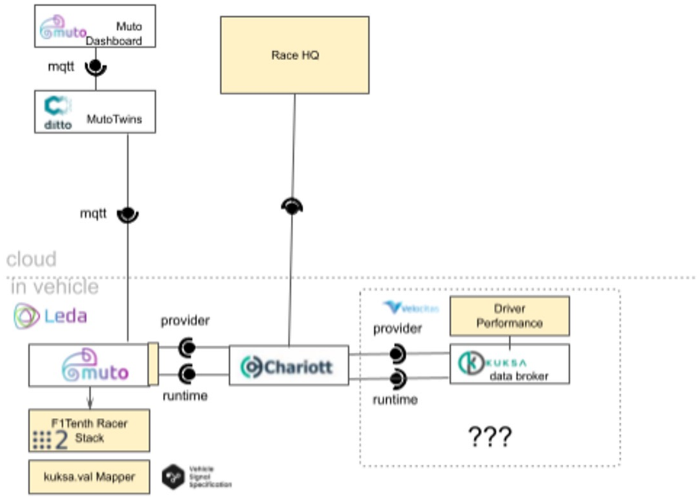
FMS - The Fleet Management Usecase

Summary	A close to "real-life" showcase for truck fleet management where trucks run SDV stacks so that logistics fleet operators can manage apps, data and services for a diverse set of vehicles.
Includes	Data collection from vehicle, In vehicle data brokers, VSS Signal Specification, FMS servers, Apps and Services using this data
Projects	Eclipse Leda, Eclipse Velocitas, Eclipse Kuksa.val, Eclipse SommR (?), Eclipse Charriot
Other Tech	InfluxDB, Prometheus, VSS, Eclipse Hono, Eclipse Kanto, Digital.Auto, Eclipse Sumo
Coordinator	Kai Hudalla



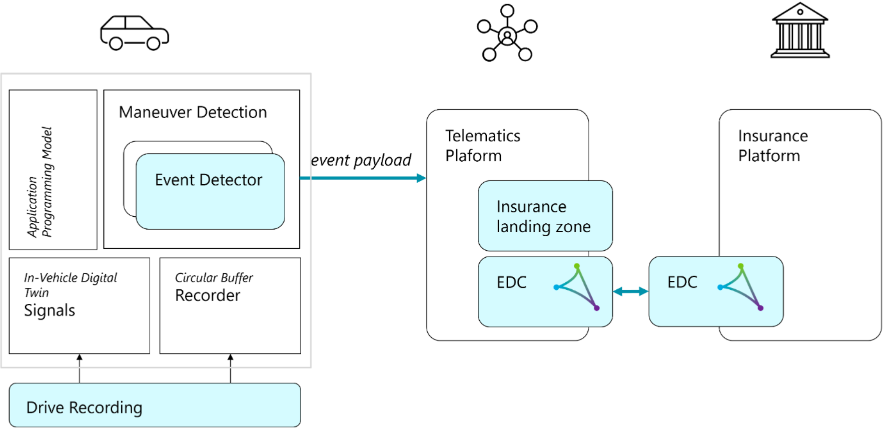
ROS - The ROS Racer Usecase

Summary	A showcase for integrating Eclipse Software Defined Vehicle (SDV) and Eclipse Dataspaces for insurance applications. Enable exchange of vehicle and risk events to support risk assesment, underwriting processes and customer experiences
Includes	Orchestration of ros software on racers and data collection from vehicles, providing remote control, race Apps and Services using SDV stacks
Projects	Eclipse Muto, Eclipse Leda, Eclipse Charriot, (Eclipse Kuksa?, Eclipse Velocitas?)
Other Tech	OSRF ROS2, Eclipse Kanto, Eclipse Ditto
Coordinator	Naci Dai



Insurance and Data Exchange

Summary	A ROS based showcase where autonomous racers that run F1Tenth stacks that are orchestrated and managed by SDV
Includes	dentification of common driver risk events (maneuver detection) such as tailgaiting or speeding using CAN bus signals. Data capture / snapshotting and transmission of risk events to a telemetics platform using MQTT pub. Data storage in a Insurance Landing Zone. Data exchange using Eclipse Dataspace components between two entities representing the OEM and the Insurance Platform
Projects	Eclipse Charriot, Ibeji, Kuksa
Other Tech	Eclipse Dataspaces, MQTT
Coordinator	Mario Ortegon



Blueprints are a Sandbox

- Discover
  - Blueprints have common architectural components and requirements
    - Cloud connectivity, publishing and receiving data
    - Security & Policy
    - Edge middleware
    - Orchestration
- Prototype
  - Reusable “glue” code
  - Contributed back to SDV projects

SDV™
Eclipse Software Defined Vehicle

Blueprints

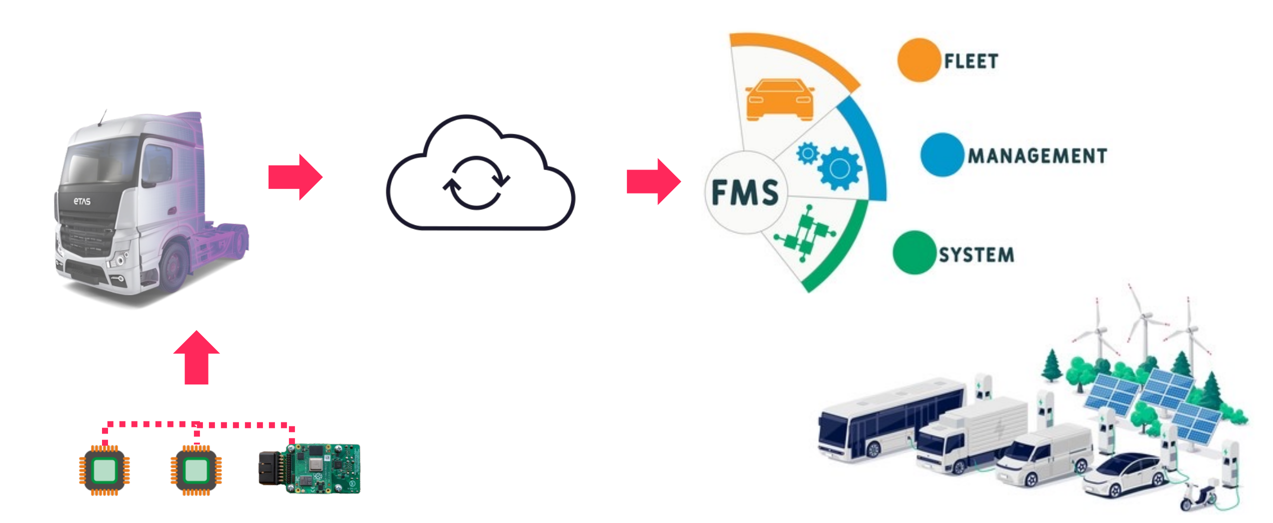
THANK YOU!

Eclipse SDV Blueprints Fleet Management

Fleet Management Systems

Problem Statement :

Fleet Operators are required by law to collect certain types of data points like Fuel Consumption, Mileage, Driver Information and others from Commercial Vehicles



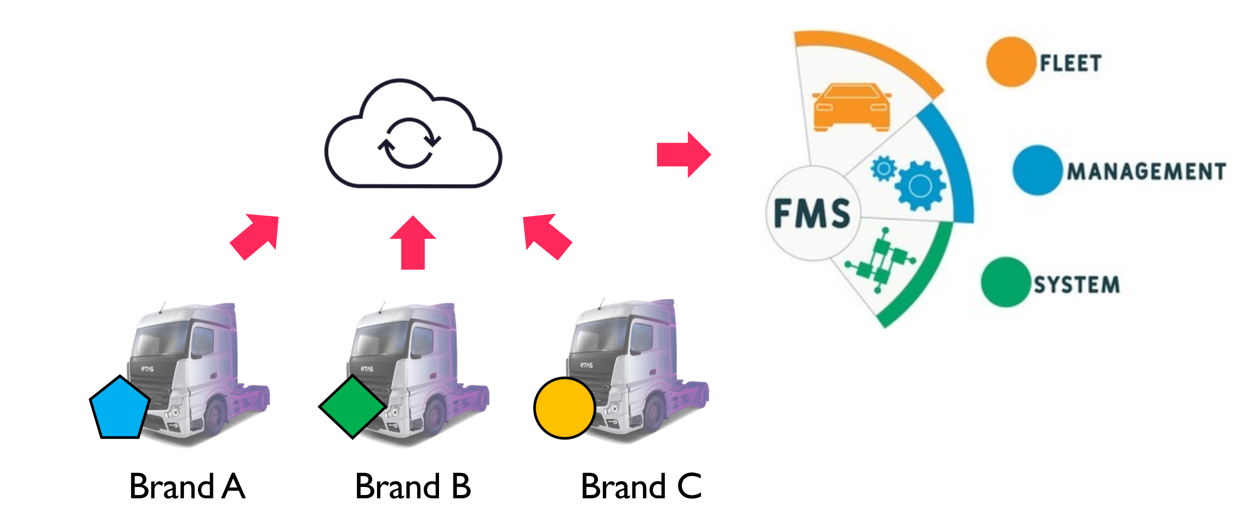
The Challenge :

How do we get this very specific data out of the vehicles and into the Fleet Management System?

FMS > How it is done today

Status Quo:

FMS vendors and/or TIER's build dedicated *Telematics Units* to put into commercial vehicles.



The Challenge :

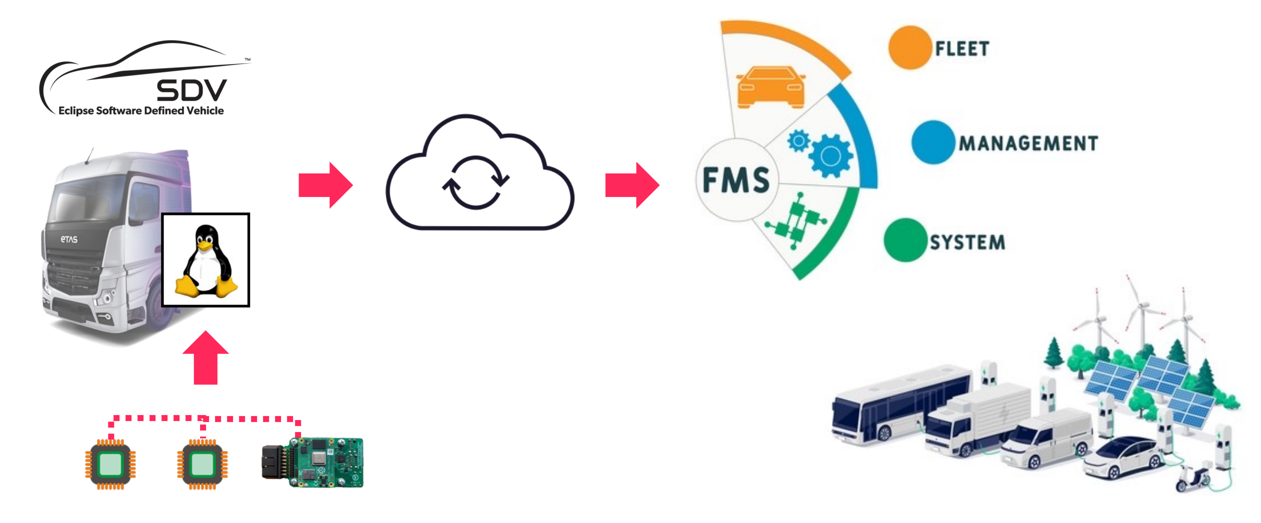
This needs to be done for each and every commercial vehicle brand in an FMS proprietary way

This is slow and costly and particularly tedious across brands and FMS vendors

FMS > How does SDV help ?

SDV potential:

Having a *generic* SDV in-vehicle computer allows to adapt any given commercial vehicle HW environment to any backend FMS simply by deploying some SW



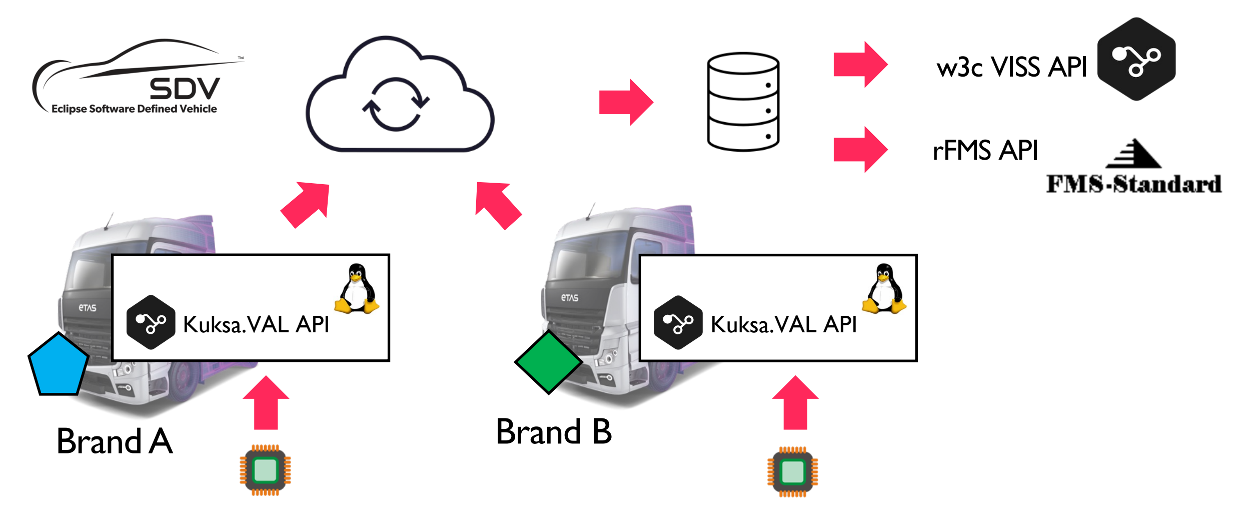
The Challenge :

We still however need to have different SW that is specific to each Commercial Vehicle brand and FMS

FMS > How do standards help?

Standards potential:

Using standard APIs for accessing data allows for employing the same hardware & software components in Commercial Verhicles across brands and FMS vendors.



The Solution :

This is the final stage in making the transition towards a fully software defined vehicle by leveraging Open standards and technology we can now

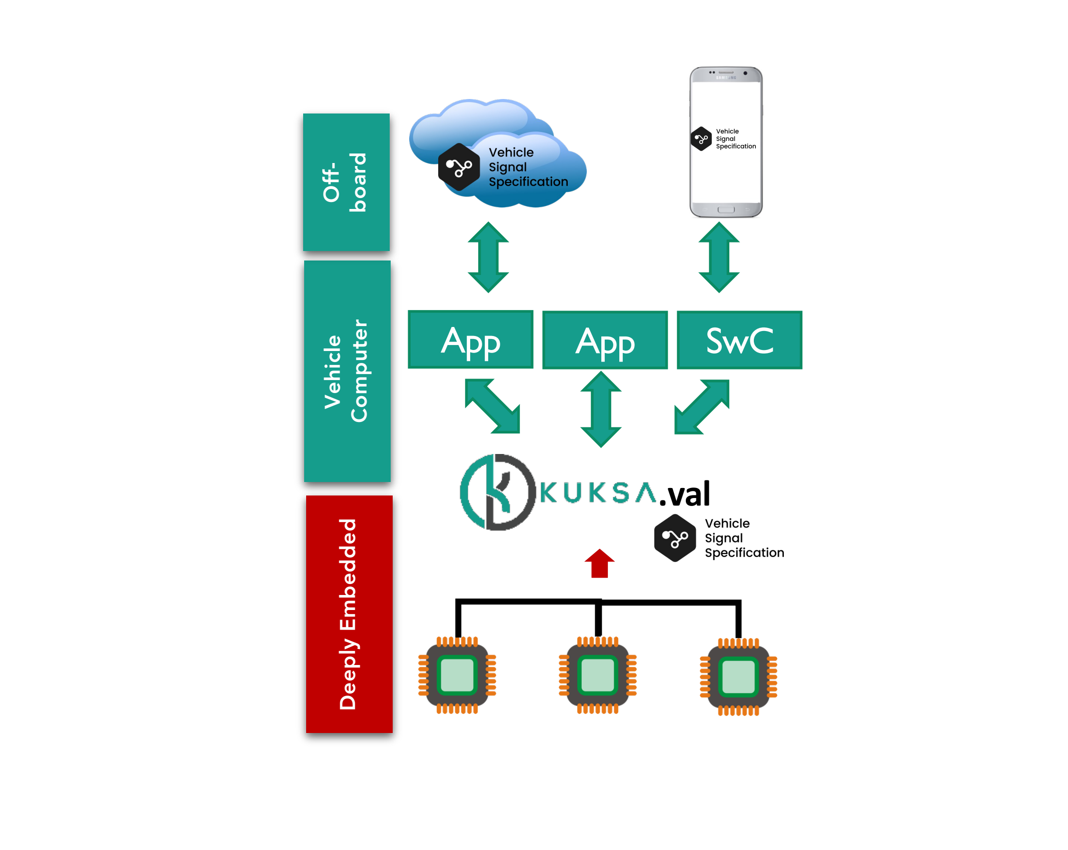
- ✓ Run Fleet SW on any vehicle
- ✓ Make changes to data reqs. & scale
- ✓ Simplify HW & focus on API's

How ?

To achieve it &

what is done already

Hardware Abstraction


Off-board
Vehicle Computer
Deeply Embedded
Vehicle Signal Specification
Vehicle Signal Specification
App
App
SwC
KUKSA.val
Vehicle Signal Specification

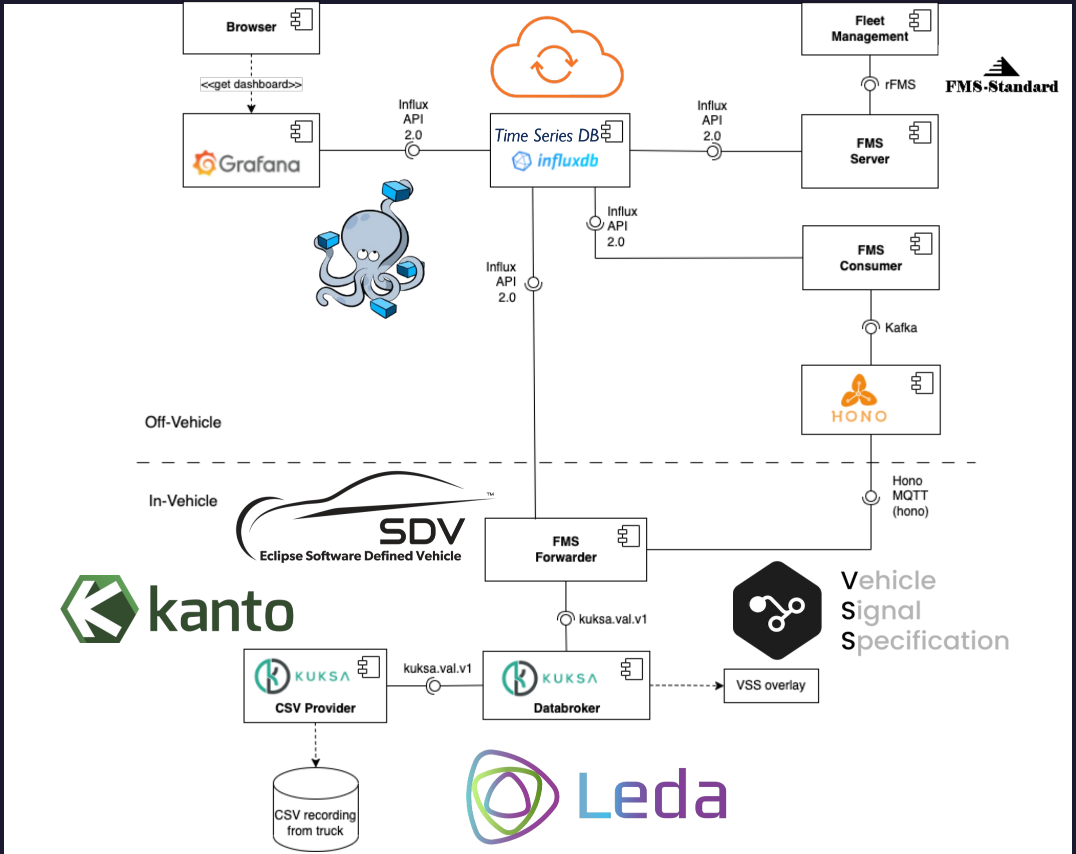
Planned Initial Architecture

- What's missing

  - Service Discovery
  - (FOSS) cloud environment
  - more advanced edge client
  - Simulating Vehicle behavior

- Topics for community discussions

  - Identify relevant and missing VSS signals
  - Map VSS signals to other standards
  - Vehicle Application SDK + data-driven broker client
  - Authentication & Authorization
  - an onboarding point for new devs (tutorial etc)
  - Something to use for demos



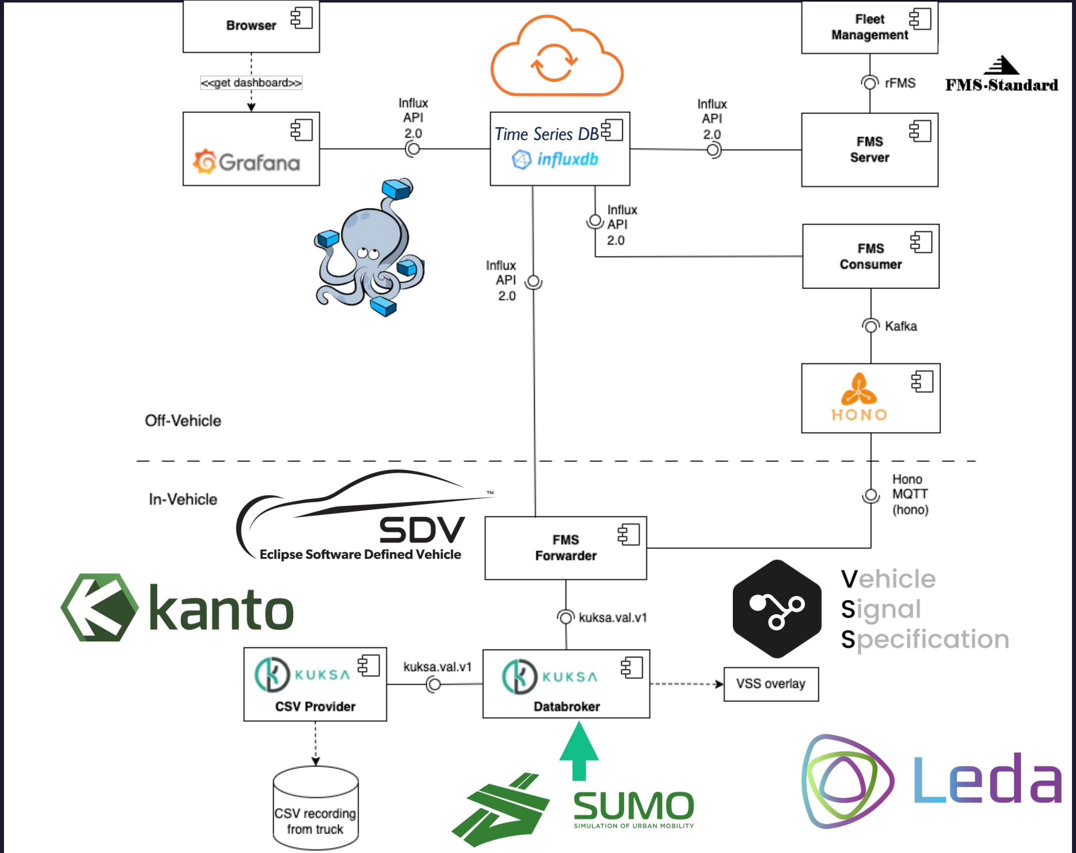
Idea: Adding Simulation

- What's missing

  - Service Discovery
  - (FOSS) cloud environment
  - more advanced edge client
  - Simulating Vehicle behavior

- Adding Eclipse SUMO

  - Run simulation scenario in Eclipse Sumo
  - Use Traci interface to get simulation data and write it to Eclipse Kuksa.val data broker as converted VSS
  - Future potential for integration with Eclipse Mosaic



Possible other Extensions

- Here's what we'd like to do…

➢ Implement FMS Feeder as Eclipse Velocitas App

- Topics for community discussions

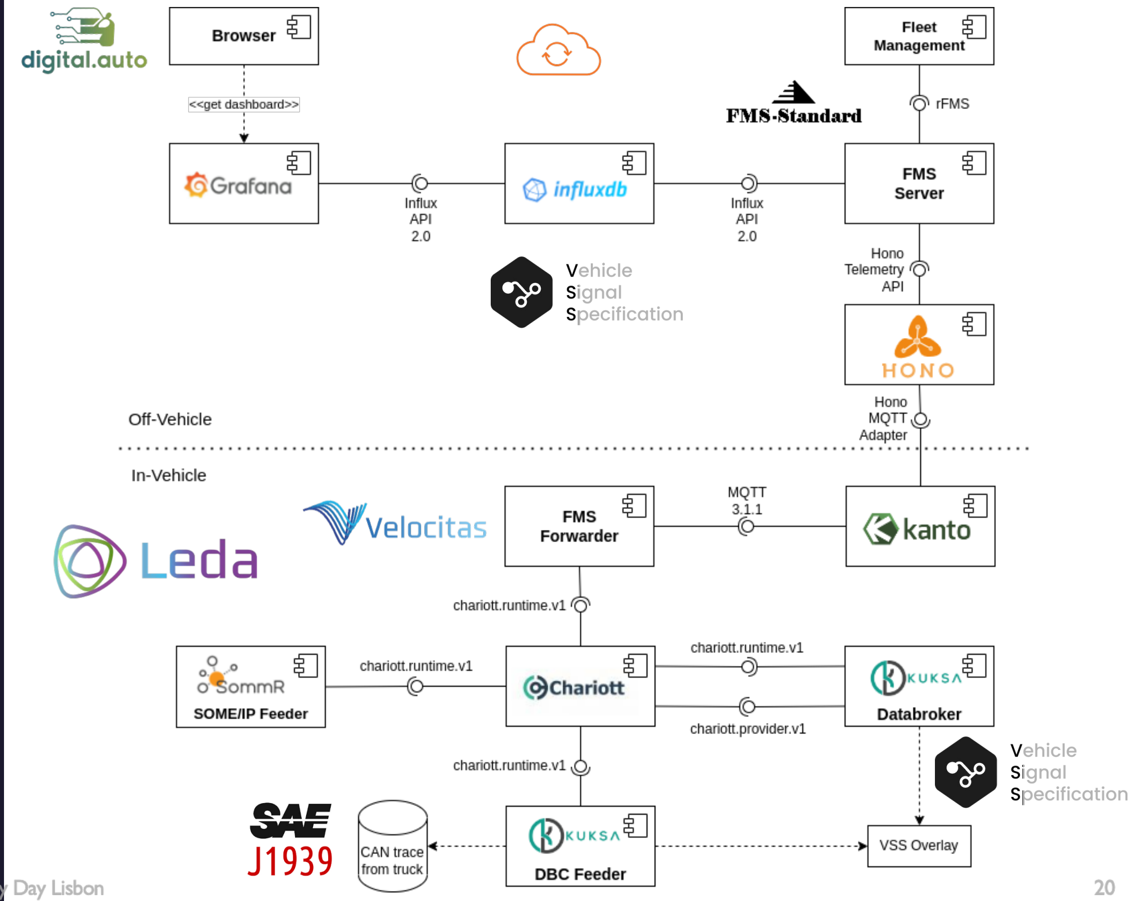
➢ Start developer journey with Eclipse Autowrx/digital.auto

➢ Auth using Eclipse Chariott

➢ Onboarding point for new devs (tutorial etc)

➢ Something to use for demos

➢ Eclipse SommR based SOME/IP feeder



Thank You

Sven Erik Jeroschewski