Early planning

General inputs

Steve input:

- First/early 'next steps'
  - How to organise project during its creation phase?
    - Move discussion to 24/7 online where possible for faster pace than weekly calls
    - Top level phases?
    - Record and comment on work breakdown.
    - Project management: where, what, how
  - Take a breath - are we aligned on top level goals and what/who/how? Who will be active? What are their interests?
  - Implementation
    - Check state of VISS Docker support
    - Confirm project setup for upstreams, e.g VISS, DBs
      - Can be considered part of early wider socialisation of project.
    - Planning for additional DBs, e.g. Apache IoTDB, Realm
  - Determine early Use Cases to be 'guided' by
  - Look at connection to MongoDB/BMW work. What is common?
  - Feeder link

Phase Planning

Phase 1 Create Early PoC

High level content:

- Playground
  - WAII VISS
  - Connect Realm DB to WAII
  - Connect Apache IoTDB DB to WAII
- Docker deployment of Playground
  - Scope: starts container chain containing above
- Documentation website
  - Scope: online instance (prove website framework has what we need at high level), with some representative (possibly placeholder documents)

Concept drawing in DrawIO: https://drive.google.com/file/d/1hxxDVgqianoMoXapPsYfjKgQVdc6Uume/view?usp=sharing

Best Practices

List of best practices that may become a best practices document for the Playground

- Docker
  - Official Docker project best practices:
    - https://docs.docker.com/develop/dev-best-practices/
    - https://docs.docker.com/develop/develop-images/dockerfile_best-practices/
    - https://docs.docker.com/develop/security-best-practices/

Project Management

In 25th Oct 2023 meeting (see minutes for details) following decisions were made:

- PM task tool
  - **Decision:** Start with github kanban for project management
- Comms
  - **Decision:** Covesa Community Slack. Create a new data-architecture channel.

- Phases
  - Discussion favours doing simple early PoC code to help decide on source and project organisation.
  - **Decision:** Needs refinement in the kanban board but first phase is 'create Playground PoC' along lines above.

Oustanding questions:

1. How to handle Epics (dev themes) which is not a feature of Github Projects? e.g. use undated Milestones.
2. How to handle tasks in a ticket that are not directly development such as operational tasks? Could use enhancement label then have problem of filtering those from feature enhancements.

Use cases to be explored

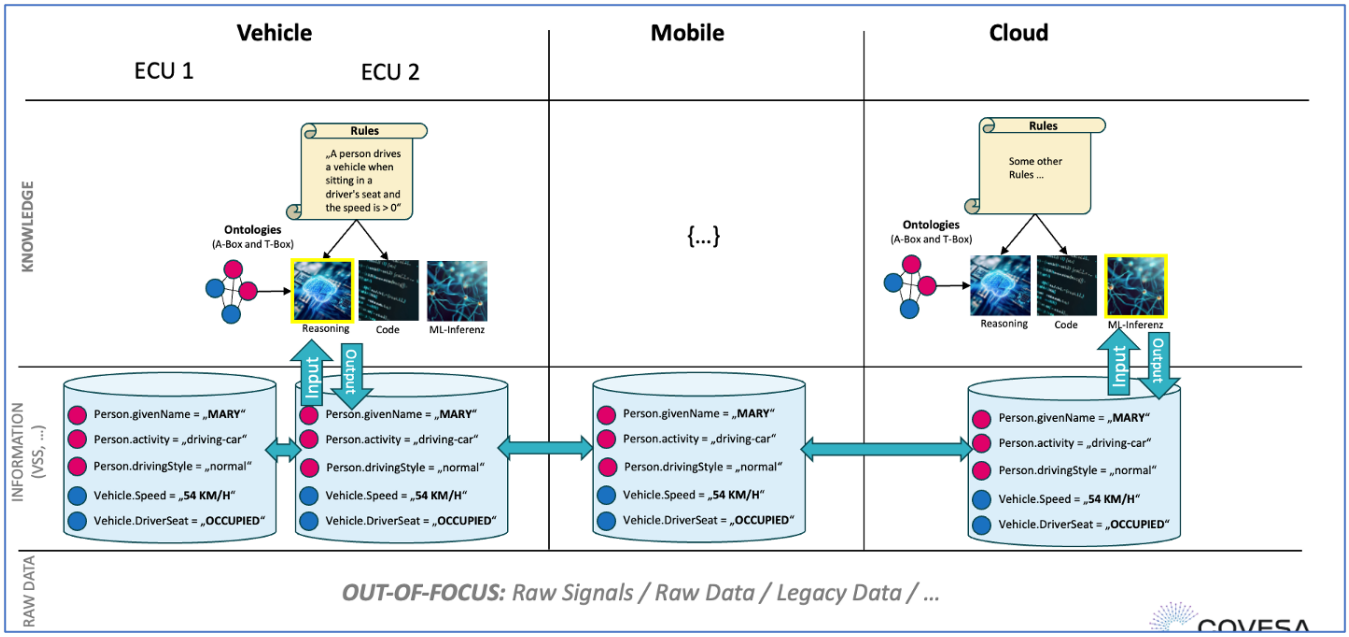
Christian Muehlbauer Input (Proposal):

User Story:

"As a user of the Playground, for example, I would like to be able to quickly implement the following use case - representative for a Use Case “touching all touchpoints” of a data-centric automotive ecosystem - in a fast proof of concept implementation using Lego pieces from the Playground."

Validation Use Case:

(Based on: https://wiki.covesa.global/pages/viewpage.action?pageId=71074417)



Ideally, the Playground Repo provides a selection of components for each „X“ for the corresponding target platform (SDK, or docker, etc):

	Playground Components						
	Data Level: **Data Feeder**	Information Level: **Data Models**	Information Level: **Data Access**	Information Level: **Data Storage**	Information Level: **Data Sync**	Knowledge Level: **Reasoning**	Knowledge Level: **Machine Learning**
In a Mobile Phone UI - a person must be able to enter their name		x	x	x	x		
In a Vehicle UI - it shall be shown if a vehicle is currently moving	x	x	x	x	x		
- a person can be assigned to a seat in the vehicle		x	x	x	x		
It shall be visualized in a cloud UI, - if person is driving the car at the moment		x	x	x	x	x	
- how Driving style of person is		x	x	x	x		x

Steve input (brainstorm list):

- General
  - Some aspect of the BMW / MongoDB project in the open
  - Look at MongoDB digital twin example
  - Interfaces: Common Service definitions, IFEX, Autosar VehicleAPI
  - Data Expert Group touchpoints
- Internals
  - Medium and high speed data, Latency etc. as previously discussed in the group
- Southbound
  - vSOME/IP example
  - RemotiveLabs feeder
  - Carla feeder
- Northbound
  - Android / Mobile
  - Cloud

VISS Protocol Servers

Notes on known OSS VISS Servers:

- WAII
  - W3C reference implementation for the VISS standard.
  - Hosted by W3C as part of the W3C and Covesa cooperation agreement
  - Has support for historical (persistent) data
  - Has backend abstraction for VSS data stores
  - Architecture supports connecting southbound data feeders to a data store
- Kuksa.val
  - Project hosted in Eclipse.
  - Historically inconsistent support for VISS (see Detroit AMM presentation and mentioned PR discussion)
  - Architecturally live 'last value' data only. No current support for historical or persistent data.
    - Assumption is that persistence is left to northbound clients?
- Aos Edge
  - The Aos Edge orchestration platform provides a VISS Server as part of the Aos Edge Core embedded part of the platform to provide secure, abstracted data to services orchestrated by the system. Implementation hosted at the Aos Edge Core product site.
  - VISS support equates more to VISS v1, with access control provided by the Service definition controls in Aos Edge, rather than using the access control specified in VISS v2.
- Other earlier implementations
  - During the early development of VISS (e.g. v1) various groups created implementations.
  - It appears they are no longer maintained but they are mentioned here for completeness.

VSS Data Store / Embedded DBs

WAII VISS server currently supports the following DBs as data store backends:

- SQLite
- Redis

Notes on OSS DB candidates to be added (this does not represent their complete feature lists):

- Apache IoTDB
  - Column orientated time-series DB designed for embedded IoT (high ingestation rate, low latency query etc.)
  - User Defined Functions for data processing at the Edge
  - Wide support for Apache eco-system, e.g. Flink, Hadoop, Spark in the cloud.
  - Wide range of API bindings + MQTT broker
  - Some sync functionality between instances
  - Possibility to use simplified TsFile API in low resource environment, although currently there is no C, C++ or Rust implementation.
- MongoDB Realm
  - Object DB initially developed for mobile.
  - Wide range of mobile API bindings + C#. C++ binding in Beta.
  - Advanced support for automated sync between instances. Also sync between Realm and Atlas (MongoDB Cloud DB) in the cloud.