Data Acquisition beyond VSS

Dr. James J. Hunt | CTO & CEO aicas | COVESA AMM, Gothenburg | April 18, 2024

Aspects of Data Collection

Architecture

- Data Model—VSS
- Data Transport—VISS
- Data Prepossesing
- Data Collection

Software Defined Vehicle Concerns

- Combining Robustness and Dynamic Update
- Complete Data Types: Quantities vs. Raw Numbers
- Software Selfawareness
- Dynamic Test and Deployment

Data Transport between Vehicle and Cloud

Requirements

- Efficient Transport
- Topic and Message Structure Based on VSS
- Multiple Devices in Vehicle
- Fast Fail-Over
- Interoperability

Candidate: Sparkplug B

- Explicit, fast fail-over using multiple (hundreds) of redundant message brokers
- Fast life-cycle updates, with inherent invalidation of stale signals
- Unlimited metadata, only transmitted on connection start (same goes for immutable properties)
- Very efficient transmission scheme based on protobuf
- Topic structure supports edge-of-network (gateway) + one layer deep devices behind gateway
- Message structure supports unlimited depth of signal (metric) organization
- Support for historical (non-realtime) signal data

Data Acquisition Plan

REQUIRE

Data Acquisition Plan

- Dynamically Select Data
- Triggers and Synthetic Data
- Customizable Data Preprocessing

SIMULATE

Rapid Turnaround

- Off-board testing
- Same tools for data generation
- Generate VSS defined signals

Software Defined Data Collection

Evolution of Data Collection

Static Aquisition

- Predefined Signals
- Limited control

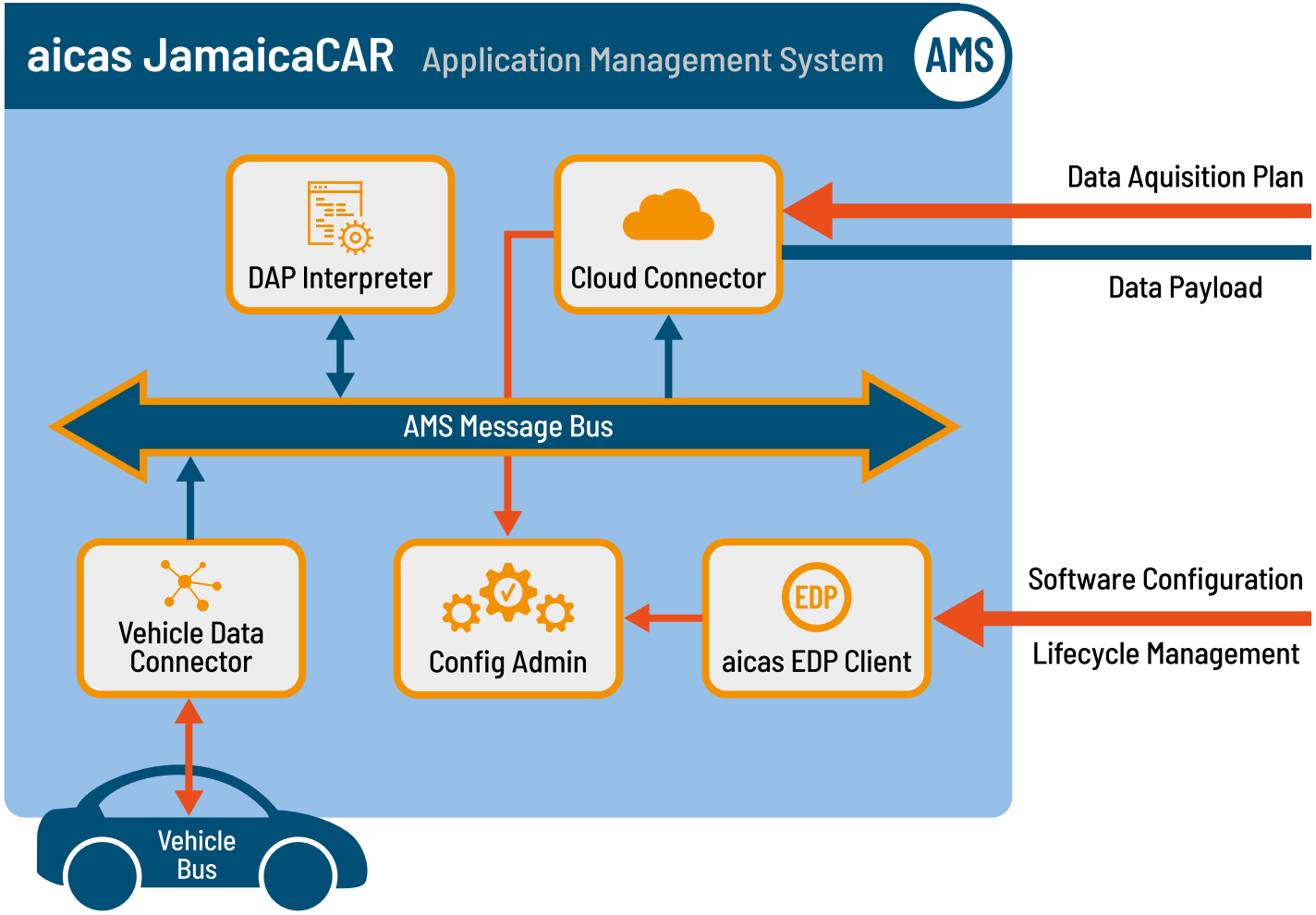
Rules w/ Fixed Operations

- Downloadable Rules
- Predefined Signals
- Predefined Operations
- Frequency Control
- Event Triggers

Dynamic Network & Nodes

- Network of Nodes
- Dynamically Updatable
- Dynamically Defined Nodes
- Based on Application Framework
- Eases Merging Multiple Data Aquisition Plans
- VSS selfawareness

Data Collection Architecture in Vehicle

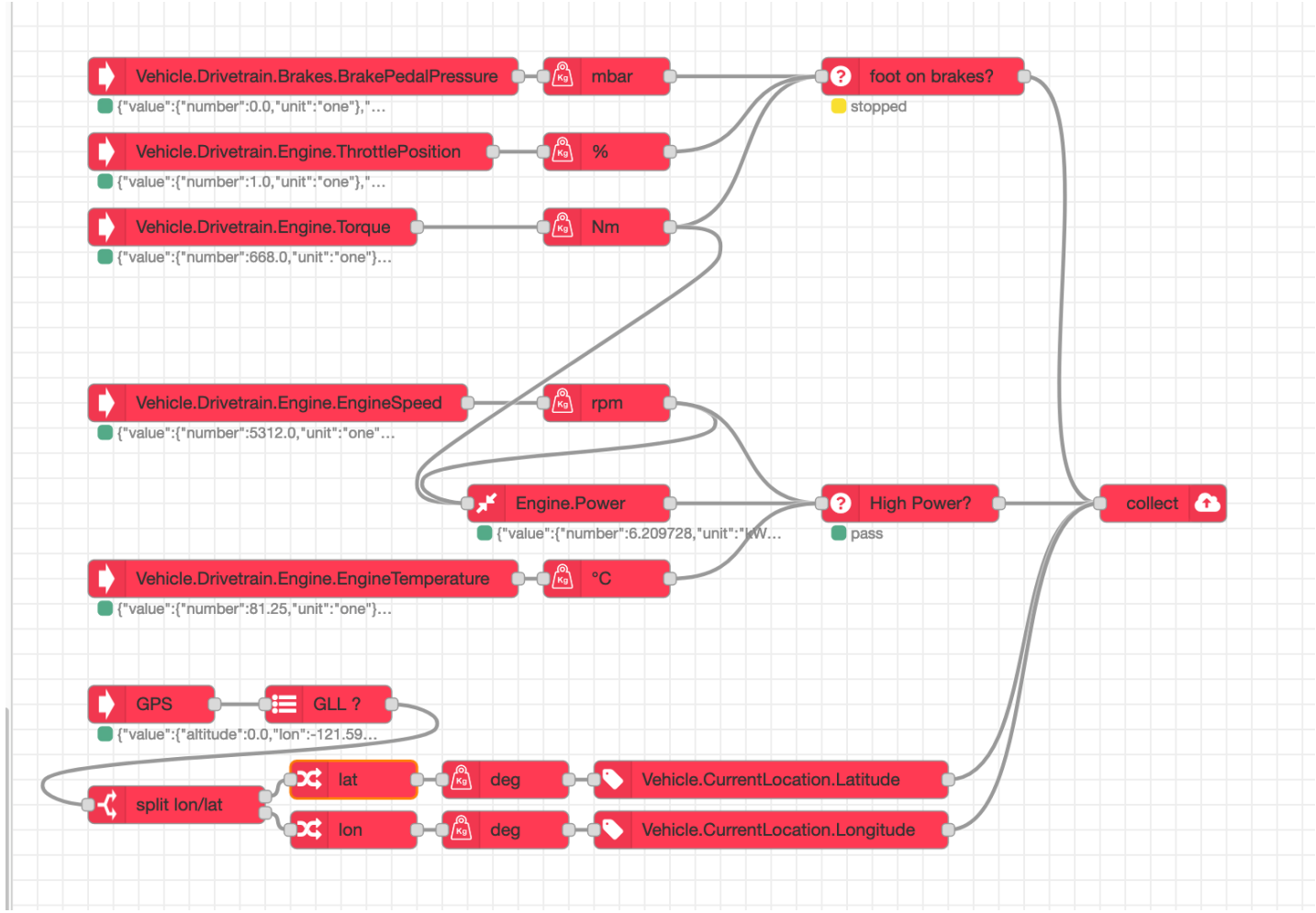


Data Acquisition Plan Examples

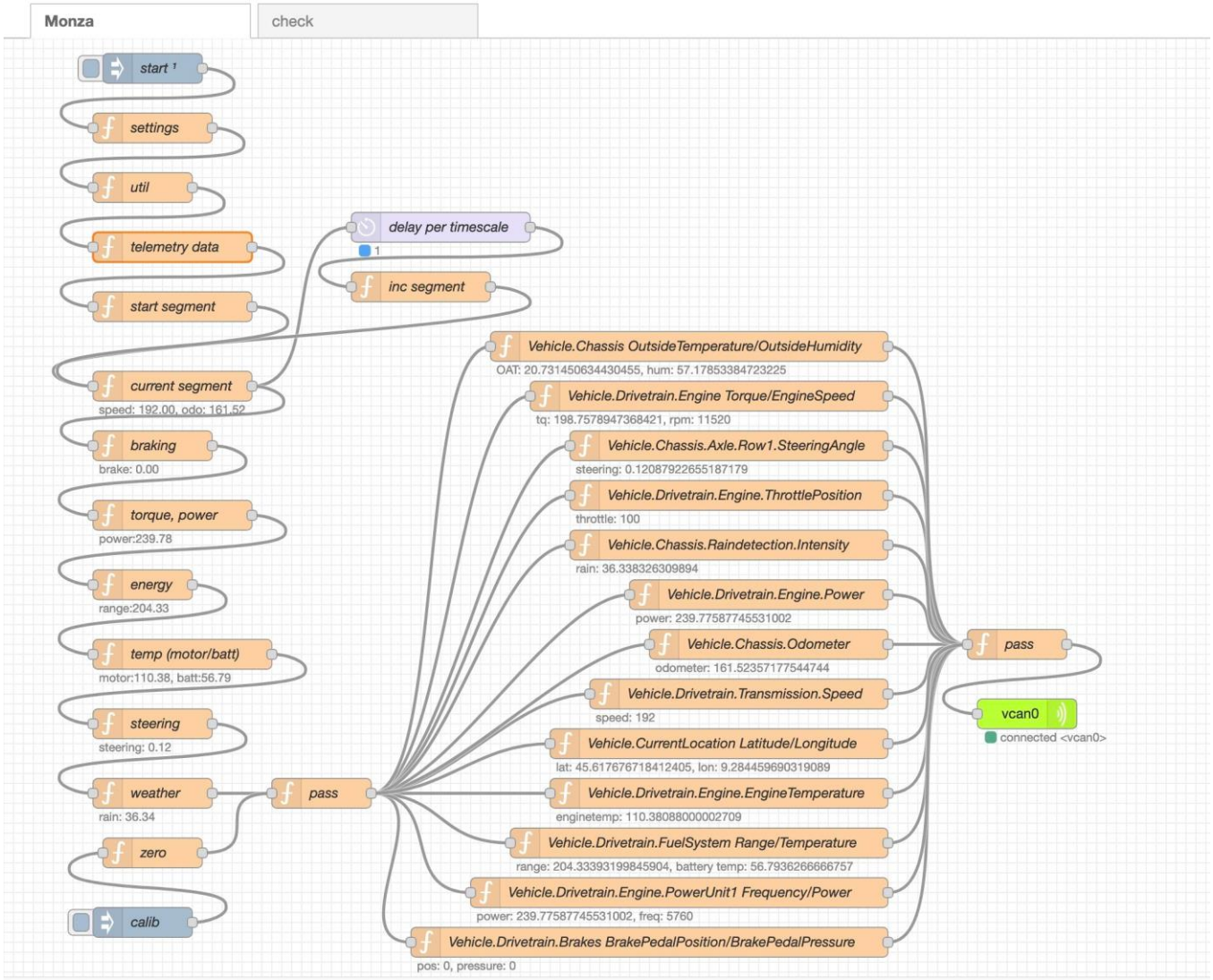
1. **Conditional**
   Pressure on
   Brake & Throttle

2. **Synthesis**
   Power from
   RPMs & Torque

3. **Conversion**
   Device binary or
   text data stream
   to VSS signal



Track Data for Simulation



Software Selfawareness: Tracing Software Providence

UN Global Regulations

Cyber Security and Software Updating

- UNECE r155—vehicle cybersecurity and cybersecurity management systems
- UNECE r156—vehicle software updates and software update management systems

ISO Guidance and Best Practice

Road Vehicles

- ISO/SAE 21434:2021
  Cybersecurity engineering
- ISO 24089:2023
  Software update engineering

US Government: EO 14028—SECURING THE SOFTWARE SUPPLY CHAIN

Memory Safety

Why is it important?

- ~70% of Security vulnerabilities are due to lack of memory safety.
- Found in iOS, Android, and Microsoft Products

What does it mean?

- Buffer overrun or out of bounds array reference
- Reference object with wrong type
- Use number as pointer
- Reference object that has been freed

References

- https://www.memorysafety.org/docs/memorysafety/
- https://media.defense.gov/2022/Nov/10/2003112742/-1/-1/0/CSI_SOFTWARE_MEMORY_SAFETY.PDF
- https://www.whitehouse.gov/wp-content/uploads/2024/02/Final-ONCD-Technical-Report.pdf

DO-332 Memory Safety

Objectives							
Technique	Unambiguous Reference	Fragment Avoidance	Timely Deallocation	Reference Consistency	Deterministic Allocation	Atomic Move	Sufficient Memory
Object Pooling	AC	AC	AC	AC	MMI	N/A	AC
Stack Allocation	AC	MMI	MMI	AC	MMI	N/A	AC
Scope Allocation	MMI	MMI	MMI	AC	MMI	N/A	AC
Manual Heap Allocation	AC	?	AC	AC	MMI	MMI	AC
Garbage Collection	MMI	MMI	MMI	MMI	MMI	MMI	AC

AC = application code, MMI = memory management infrastructure, N/A = not applicable, and ? = difficult to ensure by either AC or MMI.

Signal Data Safety

Why is it important?

- Prevents false interpretation of data
- Provided additional type checking on data transformations
- Ensures robust software defined systems
- Can prevent catastrophic failure when data is used for control

What does it mean?

- Every value is accompanied by its unit of measurement
- This includes boolen values, which should be tristate: on, off, or unknow

Reference

- https://jcp.org/aboutJava/communityprocess/mrel/jsr385/index2.html

Framework Requirements

SOA	Flexibly Combine Capabilities
Memory Safety	Exact Garbage Collection
Realtime Scheduling	Properly Prioritize Tasks
Life Cycle Management	Full Control of Software
Software Signature Verification	Only Run Validated Software
Versioned Service Resolution	Prevent Service Mismatching
Low Latency	Minimize Context Switch and Serialization

When is Java not Java?

Fairness — Conventional Java

- Base Language
- Undefined Scheduling: assumes fair scheduling
- Relies on JiT for performance
- Little support for interacting w/ hardware

Priority — Realtime Java

- Refined semantics & additional APIs (RTSJ)
- Defined Scheduling: preemptive priority & time-sharing (fair)
- Uses AoT for performance
- Support for events, device access, & interrupts

Advantages of OSGi

- Nanoservices
- Service-Oriented Architecture
- Messaged Based

- Very small Modules (< Containers)
- Remote controlled deployment and update
- Life-cycle management w/ version tracking
- Service Registry
- Standard Typed Data Bus

OSGi Challenge: Robustness and Realtime Scheduling

Problem

System Lockup

- CPU Exhaustion
- Memory Exhaustion
- RT Scheduling makes this worst
  - Priority Preemption

    A high priority thread can block all lower priority threads

Solution

Resource Enforcement

- Resource Enforcement (limits)
  - CPU Use
  - Memory Use
  - Thread Creation
- Safe bundle force termination
  - Thread.stop is unsafe
  - Ensure that finally clauses can run
  - RTSJ: Asynchronous Task Termination
- RTSJ 2.0 provides infrastructure
  - javax.realtime.control
  - javax.realtime.enforce

Framework Comparison

	Memory Safety	Realtime Scheduling	Software Validation	Service Oriented Architecture	Life Cycle Management	Versioned Service Resolution	Low Latency
AUTOSAR Adaptive							
Macchina.io							
Container							
Conventional OSGi							
Realtime OSGi							

Validation and Approval Staging Example

Data Acquisition Plan Deployment Process

Initial Development

- Select signals
- Design Preprocessing
- Test functionality with simulated data

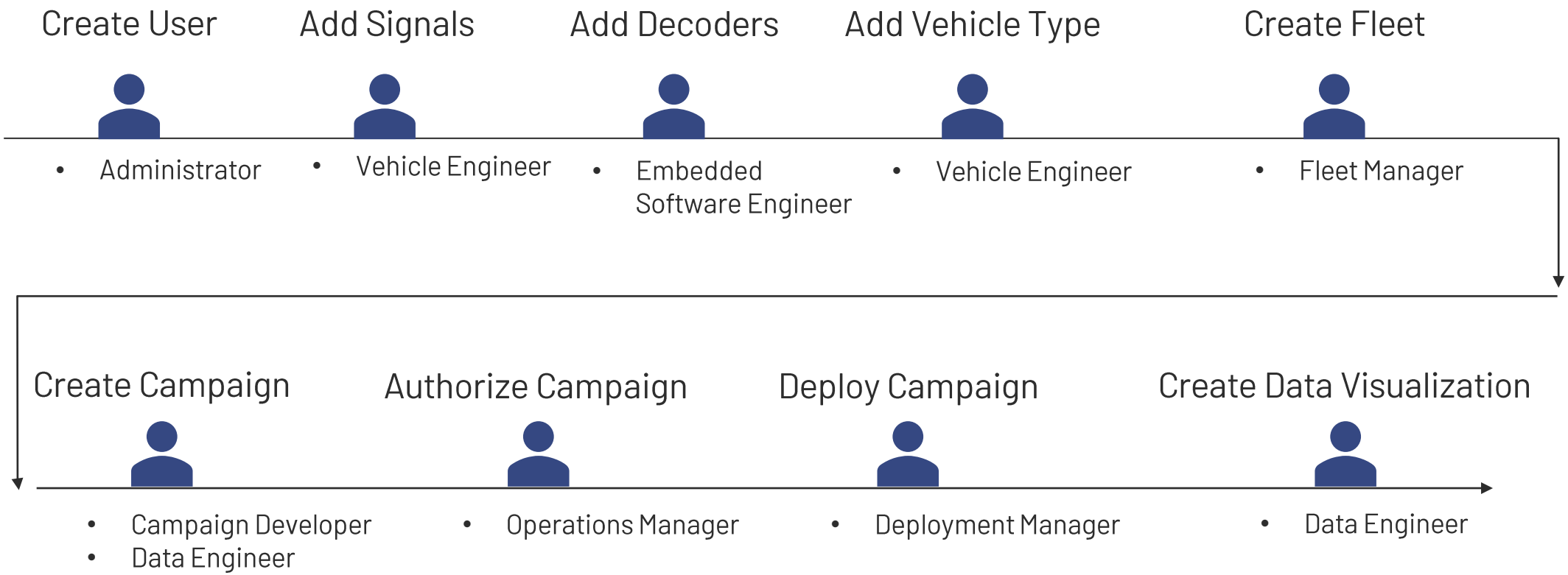
Integration Test

- Test with simulated data
- Integrate with other data collection plans
- Check performance

Deployment Test

- Test with real-world data
- Check performance
- Test usefulness

Role-Based Access UI



Who needs Data Collection? Example Roles and Permissions

Acquisition Element	Create	Read	Use	Authorize	Update	Delete
Vehicle connector	ESW Eng	ESW Eng SW Eng	ESW Eng SW Eng	QA	ESW Eng	ESW Eng Ops Mgr
Enterprise VSS catalog	Vehicle Mgr	Vehicle Mgr SW Eng	Vehicle Mgr SW Eng	QA	Vehicle Mgr	Ops Mgr
DB Schema	IT Eng	IT Eng SW Eng	IT Eng SW Eng	QA	IT Eng	Ops Mgr
Vehicle Data	Vehicle Eng	Vehicle Eng Data User	Vehicle Eng Data User	Vehicle Mgr	Vehicle Eng	Vehicle Mgr Ops Mgr
Vehicle Definition (VSS)	Vehicle Eng	Vehicle Eng Data User	Vehicle Eng Data User	Vehicle Mgr	Vehicle Eng	Vehicle Mgr
Fleet	Fleet Mgr	Fleet Mgr Data User	Fleet Mgr	Deploy Mgr	Fleet Mgr	Fleet Mgr
Data Acquisition Plan	Data User	Data User	Data User	Deploy Mgr	Data User	Data User
Campaign	Data User	Data User	Data User	Ops Mgr	Data User	Data User
Visualization Template	UI Design	UI Design Data User	UI Design Data User	Data Mgr	UI Design	UI Design

Summary

1. VSS is a great contribution to interoperability for automotive data collection
2. Use industrial standard data transmission protocols: Sparkplug
3. Vehicles should be self-aware (maintain their own VSS description)
4. A flexible means of selecting, preprocessing, and sending data is needed
5. Signal data should always carry its unit of measurement
6. Secure software defined data collection requires a memory safe framework
7. Processes, rolls, and responsibilities are necessay for dynamic data collection

Simplify Edge-to-Cloud

aicas. embedded. connected.

Dr. James J. Hunt
Cofounder, CEO, and CTO

aicas GmbH
76131 Karlsruhe, Germany
www.aicas.com
+49 721 66 39 68-0