Eclipse SDV Community Meetup Japan

™ Open SDV Initiative

Defining an Open Vehicle Service API: Activities of the Open SDV Initiative

December 11, 2025

Hiroaki Takada

Center for Embedded Computing Systems (NCES) /
Global Research Institute for Mobility in Society (GREMO)
Nagoya University
Email: hiro@ertl.jp URL: http://www.ertl.jp/hiro/

Self-Introduction

Primary Positions

- Executive Director & Professor, Center for Embedded Computing Systems (NCES), Graduate School of Informatics, Nagoya Univ.
- Executive Director & Professor, Global Research Institute for Mobility in Society (GREMO), Institutes of Innovation for Future Society, Nagoya Univ.

Research Areas

- Real-Time Operating Systems, Software Platforms
- Real-Time Analysis and Scheduling Theory
- Functional Safety, Cybersecurity of Embedded Systems
- Automotive Embedded Systems, In-Vehicle Networks
- Dynamic Map, Cooperative Autonomous Driving, SDV

Key Messages of this Presentation

- The automotive industry needs to standardize a Vehicle Service API.
  - Without it, an open SDV ecosystem cannot grow, and the *industry risks being subsumed into the smartphone ecosystem*.
- The API must work across diverse vehicles while still enabling OEM-specific differentiation.
- We are defining such an API ahead of OEM-led standardization through the Open SDV Initiative.
- We welcome collaboration with those who share this vision.

What is an SDV? – Our Working Definition

Software-defined Vehicle (broad sense)

- A vehicle whose behavior, functionality, and value are defined by software.

More practical definition for this talk

- A vehicle whose functions can be extended or modified *after sale* by adding or updating software via OTA – beyond IVI.

Why SDV matters: the DevOps analogy

- Software can be updated continuously.
- SDV applies DevOps to vehicles.
  - Continuous improvement of the product after delivery, based on real-world usage.

Evolution Steps of SDV and Open SDV

Step 0: Technical SDV

- Vehicle supports OTA updates for ECUs beyond IVI.

Step 1: Value Creation through SDV

- OEM continuously expands functionality and generates recurring revenue through software updates.

  OEM-controlled SDV

Step 2: Open SDV (Our term)

- Allows third-party developers to install applications into the vehicle.

  Ecosystem-enabled SDV (true openness)

Impact of Open SDV

Why Open SDV matters

- In closed SDV, new features require OEM involvement.
- In Open SDV, third parties can deploy apps without OEM intervention.

Vehicles become platforms, not just products, similar to smartphones.

Value shift enabled by Open SDV

- Much innovation will come from the ecosystem, not only from OEMs.
- New recurring revenue opportunities for OEMs *and* developers.
- Vehicles gain functions far beyond what OEMs can supply alone.

Challenges Toward Realizing Open SDV

Handling Safety-Related Applications

- Should third parties be allowed to develop safety-related apps?
- If NO → ecosystem shrinks.
- If YES → responsibilities and regulations must be clearly defined.

Sustainable Business Model for Third-Party Developers

- There is a concern that SDV apps may not be profitable, because drivers spend far less daily time in vehicles than in smartphones.
  - Business models can rely on services, partnerships, or data – not necessarily on selling apps.
- Safety restrictions directly shape what kinds of business models are possible.

Standardization of Vehicle API

Third-party developers need a *common* vehicle API

- Without a common API, the ecosystem cannot exist.
- The API should be easy-to-use for app developers outside automotive industry.
  - A *Vehicle Service API* is needed.
  - Conceptually similar to COVESA VSC (Vehicle Service Catalog)

If the automotive industry does not define this API…

- Smartphone ecosystems will define their own “vehicle APIs.”
- There is a risk that vehicles could become mere *peripherals* of the smartphone platforms.
- External APIs and platforms may constrain vehicle hardware design choices.

Desired Features of the Vehicle Service API

Applicable across diverse vehicles and E/E architectures

- Abstracts differences in vehicle types, models, and internal architectures.
  - Works regardless of OEM-specific design choices.
- Provides a stable logical API above hardware and signal differences.

Allows OEM-specific differentiation in implementation

- The API defines *what* is requested; OEMs determine *how* it is realized.
  - OEMs remain free to implement their own control logic, safety strategies, and tuning.
- To realize this separation, the API must be defined at a higher level of abstraction.

Open SDV Initiative – Motivation and Activities

Why we launched Open SDV Initiative

- Momentum for defining a common Vehicle Service API has *not yet emerged* among OEMs.
  - The slow progress of VSC reflects this situation.
- We decided to define it ahead of OEM-led standardization, launching the Open SDV Initiative in Oct. 2024.

Current activities

- Definition of the Vehicle Service API (Logical Level)
  - Early drafts released in Mar. and Sep. 2025 (under ongoing refinement).
- Prototype implementations
  - Simulation-based testing and initial in-vehicle trials.
  - Full-scale implementation needs more resources.

API Design Principles

Easy-to-use for application developers

- Simple and consistent Vehicle Service API.

Compatible and extensible across diverse vehicles

- Works across different vehicles and E/E architectures.

Provide application arbitration and locking mechanisms

- Multiple apps may request overlapping functions.
- The API must define conflict resolution rules and locking/priority mechanisms.

Introduce a mechanism to restrict which apps may use safety-related functions

- Which application is allowed to use which API is determined by the *OEM* and the *user*.
- This restriction mechanism is called *risk control*.

Risk Control – Handling Safety-Related Functions

- Risk-involving APIs are still defined
  - Safety-related functions remain part of the API.
- Vehicle describes its residual risks
  - Residual risks (risks the vehicle cannot fully mitigate) differ from vehicle to vehicle.
- App describes which risks it mitigates
  - Each app declares the risks it can handle. The OEM assesses whether the app’s risk mitigation is sufficient.
- API call is permitted only if the app covers the vehicle’s residual risks and user consent is given
- This risk-control mechanism is defined in the API spec
  - The spec defines how risks are described and how the platform verifies them.

Concluding Remarks

A standardized Vehicle Service API is essential for the automotive industry

- It must work across diverse vehicles while still enabling OEM-specific differentiation.
- It must provide application arbitration and clear responsibility partitioning for safety.

We are defining such an API ahead of OEM-led standardization

- Through the Open SDV Initiative, we are prototyping and refining this approach.

We seek active collaboration with the Eclipse SDV community

- We would be happy to work together – either by joining our initiative or by collaborating within Eclipse SDV activities.