

The Vehicle OS will be open — if the industry converges now

Yoshinari Shimazaki



Content



01

The Vehicle OS

02

The Vehicle OS puzzle

03

The missing pieces

The Vehicle OS in the age of SDV

From distributed ECUs to one centralized, software-defined platform

What it is

One integrated software platform across central and zonal compute – abstracting hardware so features are developed, updated and scaled independently of the ECUs.

A leading example

Mercedes-Benz MB.OS – chip-to-cloud, spanning all four vehicle domains.

Peers: VW.OS, BMW, Toyota Arene, Stellantis STLA Brain, Tesla

Key ingredients

- Mixed-criticality kernels (Linux, QNX, AUTOSAR)
- Hardware abstraction and service-oriented middleware
- Safety and security by design (ISO 26262, ISO 21434)
- OTA, DevOps and cloud toolchains

What does the market ask for

- Faster time-to-market and reuse across model lines
- Lower cost through ECU consolidation
- New revenue from software and data services
- Less supplier lock-in via open standards

Why open-source powers the Vehicle OS

Open foundations are what make a modern automotive software platform buildable, shareable and scalable

01

A shared foundation

Linux, real-time kernels and service-oriented middleware give every Vehicle OS a proven base layer no single carmaker builds alone.

02

Open collaboration

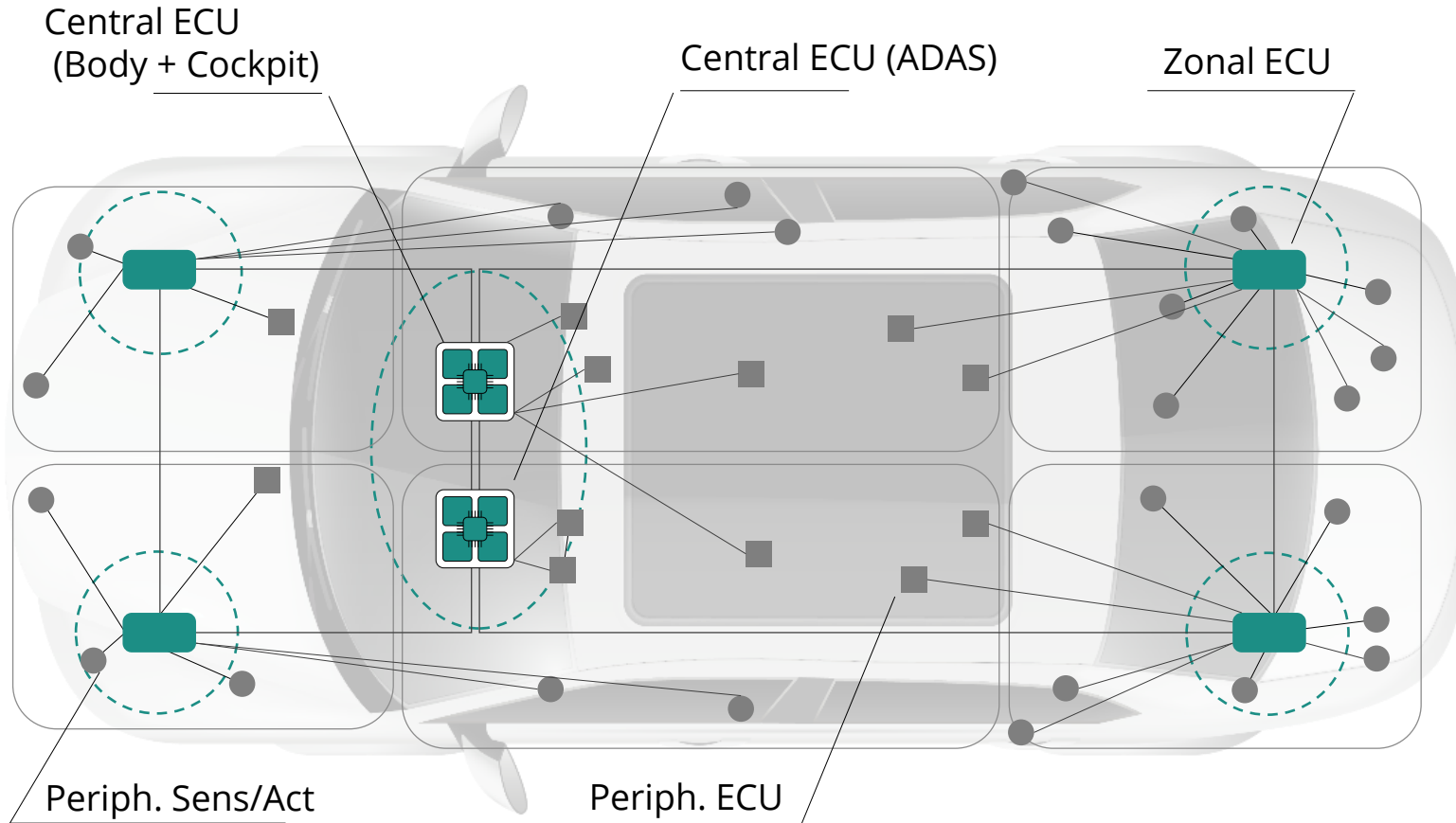
Common standards and projects such as Eclipse SDV, JASPAR, AGL, AUTOSAR Adaptive and COVESA let carmakers and suppliers co-develop instead of reinventing.

03

Strategic advantage

Portable, reusable code speeds time-to-market, lowers cost and frees brands from vendor lock-in across every model line.

Open source in automotive today



Eclipse S-CORE

Core stack for high-performance ECUs

JASPAR API

Standardized vehicle API and development environment

Automotive Grade Linux

Linux for IVI, Body and Gateway

Android Automotive

Open IVI platform and apps

COVESA VSS

Vehicle Signal Specification

Autoware

Open autonomous driving stack

RISC-V

Open hardware ISA

Zephyr RTOS

Real-time kernel for ECUs

... and many others

Content



01

The Vehicle OS

02

The Vehicle OS puzzle

03

The missing pieces

Why is a Vehicle OS so hard for the industry?

Challenges span technology, business model and innovation speed — the hardest barriers are organizational, not just technical

1 Technology & Engineering

- **Unwinding legacy E/E** — 100+ distributed ECUs with tightly coupled hardware and software must be re-platformed onto centralized / zonal compute
- **Integration complexity** — stitching fragmented software stacks from dozens of suppliers into one reliable, maintainable platform
- **Safety & security gating** — ISO 26262 functional safety plus 500+ new global regulations slow every release and certification cycle
- **Decoupling & OTA** — separating software from hardware and retrofitting robust over-the-air update pipelines onto existing vehicles

2 Business & Commercial

- **Huge, uncertain capex** — VW's Cariad absorbed €14B+ and ~\$7.5B operating losses before being recast as an integrator of partner code
- **Tier-1 dependency** — leaning on suppliers means loss of control over the stack — a hard make-vs-buy dilemma
- **Unproven monetization** — recurring subscriptions and OTA feature upgrades still earn far less than the promised software revenue
- **Talent gap & culture** — competing with tech giants for software engineers (~150k projected shortfall) inside a hardware-first organization

3 Innovation Speed & Operating Model

- **Waterfall vs. agile** — ASPICE-gated, phase-driven programs clash with continuous DevOps delivery and fast iteration
- **Cadence mismatch** — multi-year vehicle development cycles cannot match tech-industry release speed
- **Software-native rivals** — Tesla, BYD, NIO and Xpeng iterate years ahead; China is scaling SDVs fastest
- **Becoming a software company** — the blocker is restructuring the operating model and culture, not just adopting new tooling

Distributed IP ownership — the journey to open source

Shared, co-developed code makes ownership the fourth hard challenge — and pushes the platform from proprietary toward open source

4 Distributed IP ownership

- **Joint ownership** — co-developed code across OEMs, suppliers and consortia blurs who actually owns the IP
- **License compatibility** — mixing copyleft and permissive licenses in one stack creates real compliance risk
- **Patent exposure** — uneven indemnification and non-assertion leaves members open to third-party claims
- **Open vs. differentiate** — contributing to shared platforms can erode the proprietary advantage OEMs rely on

Why it points toward more open source

- Shared platforms spread the cost of non-differentiating software across the whole industry
- Neutral, foundation-style governance stops any single OEM or supplier from owning the IP
- Open licenses make patent, non-assertion and contribution terms explicit and predictable
- Differentiation moves up the stack — to apps, data and experience, not the base platform

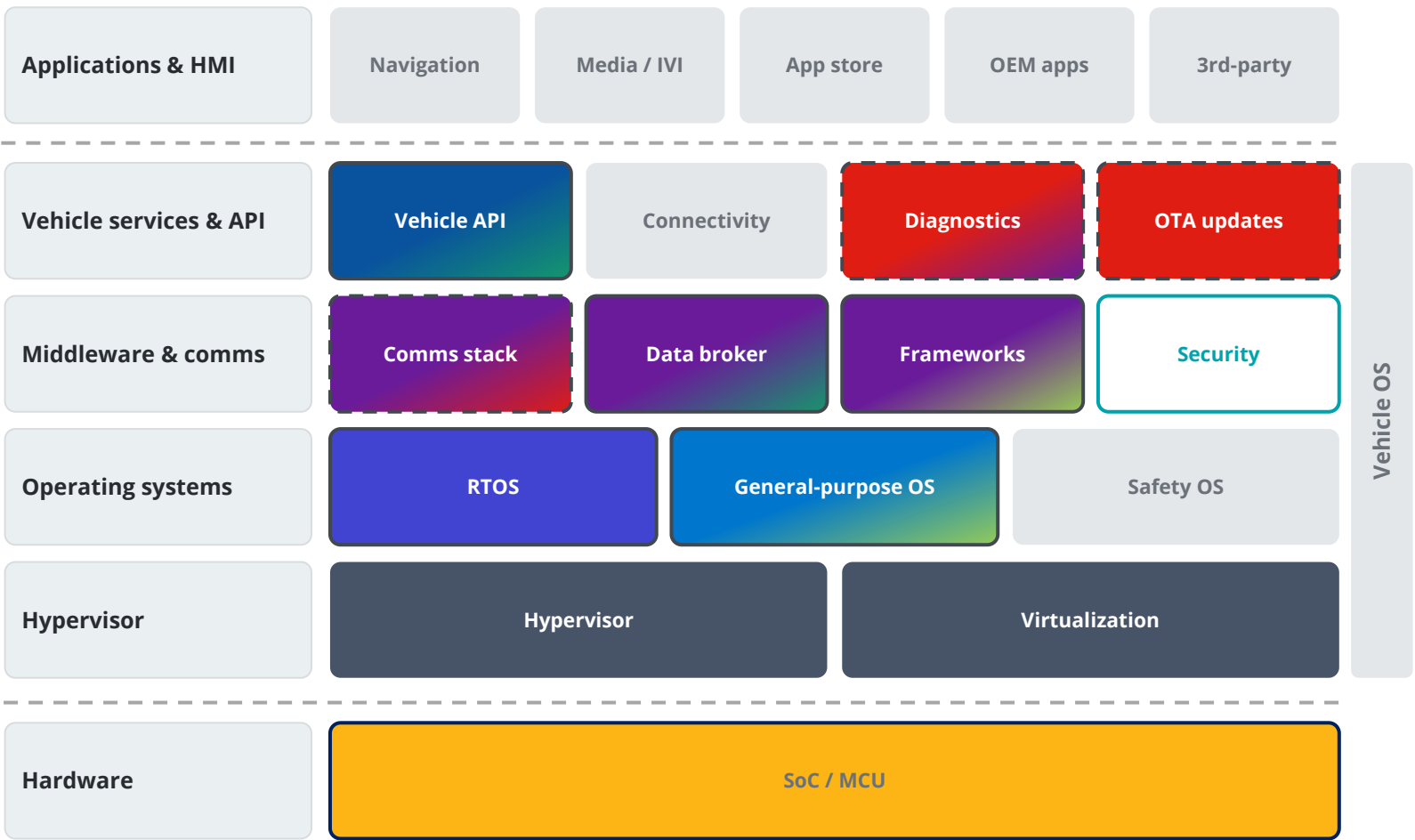
IP journey — from proprietary, member-gated terms to fully open source



What open-source covers — and what it doesn't

Many open-source projects want to provide building blocks for the vehicle OS. Significant overlap and duplication slows progress and wastes effort.

Open-source coverage across the automotive stack



What open-source covers

- ☐ Open source
 - ☐ Open standard / proprietary
 - ☐ Partial / framework only
 - ☐ Not covered — supplied by others
-
- ☒ COVESA
 - ☒ JASPAR
 - ☒ AUTOSAR
 - ☒ Eclipse SDV
 - ☒ Automotive Grade Linux
 - ☒ Linux Project
 - ☒ Xen Project
 - ☒ Zephyr OS Project
 - ☒ RISC-V

Content



01

The Vehicle OS

02

The Vehicle OS puzzle

03

The missing pieces

The missing pieces

The recurring asks — whole-stack coverage, proprietary parts going open, clear IP ownership and clean interfaces — finally come together

The missing pieces:

1

Whole-stack coverage

One platform spanning every layer, not just isolated parts

2

Proprietary goes open

Today's closed blocks open up — e.g. AUTOSAR moving toward open source

3

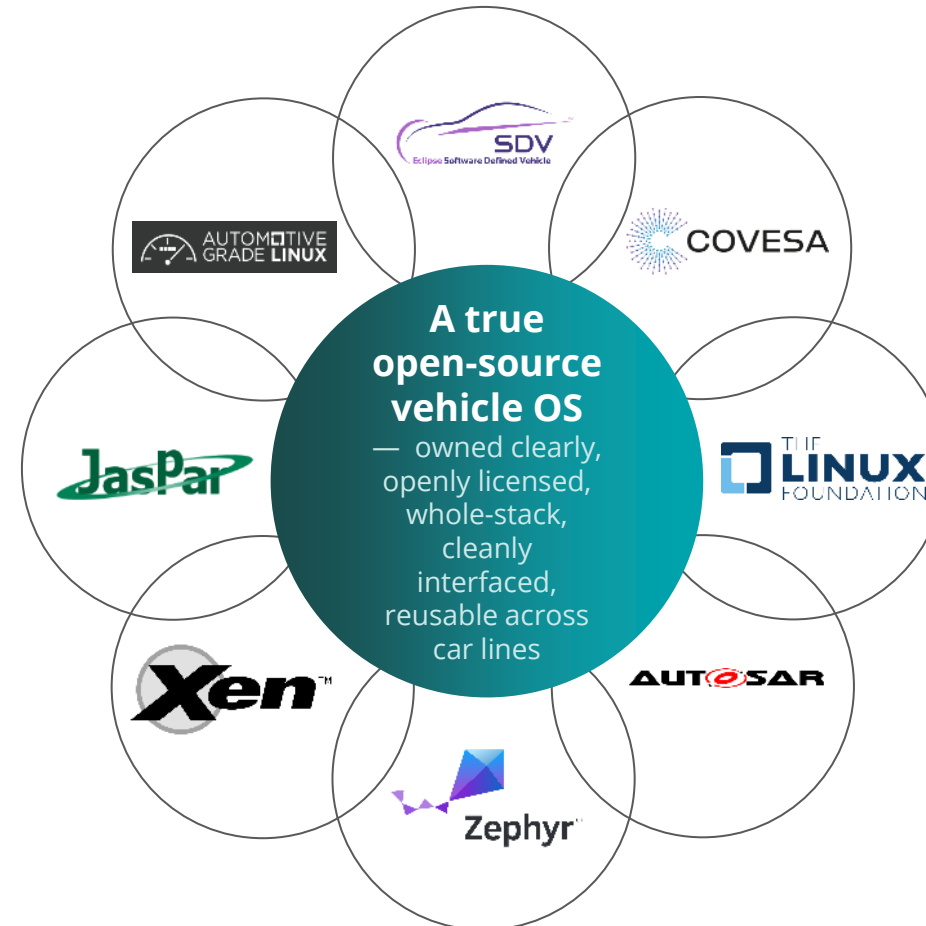
Clear IP ownership

Unambiguous rights across the entire OS, not piecemeal

4

Long term maintenance

Ecosystem and community to support throughout the lifecycle



Convergence

What it looks like – shared interfaces and a proven baseline

1 Common semantic API



What it means

Applications target one shared, meaning-based interface instead of platform-specific calls.

Why it matters

Software written once runs across vehicle platforms and car lines without costly rewrites.

1

2 Reference integration



What it means

A canonical, working build showing how the open components fit together end to end.

Why it matters

Teams validate against a proven baseline, cutting integration risk and guesswork.

2

SPANNING THE OPEN-SOURCE ECOSYSTEM



Convergence

What it looks like – coordinated projects and certified interoperability

3 Structured bridge between projects



What it means

Formal coordination links the separate open-source efforts such as AUTOSAR, Linux and COVESA.

Why it matters

Aligned roadmaps replace duplicated, divergent work and keep interfaces compatible.

3

4 Conformance profiles



What it means

Defined compliance levels that implementations can test and certify against.

Why it matters

Buyers and partners gain a clear, verifiable guarantee of interoperability.

4

5 Joint release trains and integration labs



What it means

A synchronized release cadence backed by shared facilities for continuous testing.

Why it matters

Predictable, jointly validated releases shorten time to a production-ready stack.

5

Business model convergence

Open source doesn't eliminate business models – it shifts value from license control to integration, qualification, lifecycle and compliance.

OPEN CORE		COMMERCIAL VALUE
Reference architecture	➡	Productized integration
API mappings	➡	OEM-specific adaptation
Open CI tests	➡	Qualification evidence
Community release	➡	Long-term maintenance
Open conformance profile	➡	Safety/security lifecycle service
Reference stack	➡	Production-grade platform integration

“Open source creates the commons. The business is making the commons production-grade.”

The background of the slide features a row of cars parked in a lot, viewed from a low angle. The cars are slightly out of focus, with the closest one being a dark-colored sedan. Overlaid on the right side of the image is a white network diagram consisting of numerous small dots connected by thin lines, forming a complex web-like structure. A single white diagonal line is positioned on the left side of the slide, pointing towards the text.

Will there be an open source Vehicle OS?

How Elektrobit drives the industry to open source

Three initiatives where we lead the industry toward open, safety-critical vehicle software

01

EB corbos Linux for Safety Applications

The world's first open-source OS assessed for automotive functional safety.

ASIL B / SIL 2 certified

CES 2025 Innovation Award

Live in Mobileye Drive

02

Eclipse S-CORE — an open Safe Vehicle Core

Co-developing an open, safety-aligned vehicle core with BMW, Mercedes-Benz and Qualcomm.

Founding member

ISO 26262 aligned

Shared, reusable core

03

Eclipse Ankaaios — open vehicle orchestration

Our own Eclipse project orchestrating software workloads on in-vehicle high-performance computers.

Initiated & led by Elektrobit

Eclipse SDV project

Automotive-grade, in Rust

We don't just adopt open source – we lead it. 35 years in automotive, 630M+ vehicles powered.

Smart SDV platform

Right-sized SDV for global vehicle portfolios



How can we help?

Architecture review and consulting

- Platform variants
- SDV use-case definition
- Development model

Hands-on PoCs including vehicle functions

- Identification of first use-case
- PoC in-vehicle showcasing Smart SDV and existing E/E integration

Incremental adaption of E/E architecture

- Staged rollout into staggered projects and platforms

Implementation of connectivity and applications

- Integration into existing OTA backends or completely new setups
- Typical in-car experience apps for a complete offering

Thank you



Yoshinari Shimazaki

Product Manager
HPC solutions

Elektrobit – Our software moves the world

yoshinari.shimazaki@elektrobit.com
elektrobit.com

