



Joining the Forces - The Future of Adaptive Platform – CAPI, S-CORE, and Proprietary Software

Masaki Gondo
CEO/CTO/President
eSOL

Masaki Gondo, CEO/CTO/President, eSOL

- I was supposed to be a music player but graduated from SUNY ESF in Biochemistry and joined eSOL in 1996
- Development of original (RT) OSeS and applying them to automotive/industrial/consumer/aerospace projects with system design consultations
- Developed RTOS for Arm MPCore in 2006 as the first commercial OS. Released another first manycore-capable OS in 2012 (eT-Kernel Multi-Core Edition and eMCOS)
- Joined AUTOSAR to develop AUTOSAR Adaptive Platform in 2016 as one of the original architects.
- Chaired IEEE2804 SHIM (Software-Hardware Interface for Multi-manycore) and published in 2019
- Certified our OSeS for ISO26262 ASIL-D and IEC61508 by SGS-TUV
- Introduced Scrum in 2015 and integrated it with FuSa
- Continuing R&D on a behavioral driver model technology
- Introduced Product Management to the company in 2003
- Current position since 2025

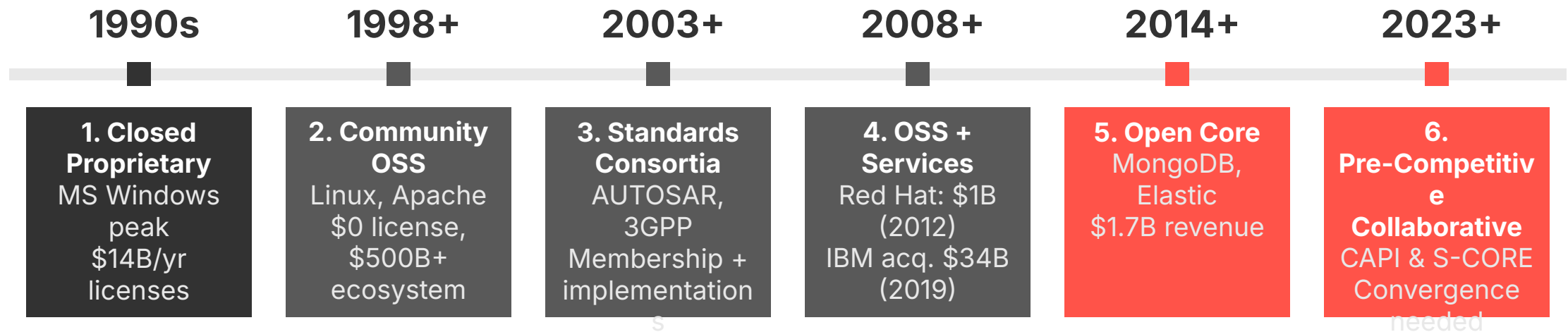
Agenda

- 30 Years of Software Business Models
- The Framework for selecting the right model
- One Team — Joining the Forces

30 Years of Software Business Models

From Closed Proprietary to Pre-Competitive Collaboration

A 30-Year Survey: How We Got Here – 6 Business Models



- Pattern: As software commoditizes, collaboration models emerge. As differentiation matters, proprietary models persist.

Models 1 & 2: Closed Proprietary & Community OSS

MODEL 1: CLOSED PROPRIETARY		MODEL 2: COMMUNITY / FOUNDATION OSS	
Mechanism	Source hidden; perpetual/subscription licenses; updates sold	Mechanism	Shared commons; no direct revenue; indirect value via ecosystem use
Examples	Microsoft Windows NT/XP: OS licensing ~\$14B/yr peak Oracle DB licensing: ~\$15B/yr today QNX Neutrino: auto RTOS proprietary	Examples	Linux: 23M+ lines, 4,300+ contributors, 96% of top web servers Apache HTTP Server: 67% of web servers at peak (2005) Eclipse Foundation: \$7.8B contributed code value est
Strengths	Maximum value capture; IP control; clear accountability	Strengths	Maximum adoption; no license friction; community innovation velocity
Weaknesses	Vendor lock-in; industry fragmentation; high cost for ecosystems needing interop	Weaknesses	Tragedy of commons risk; no funding for safety certification; governance gaps
Auto Parallel	Pre-AUTOSAR ECU software era — full fragmentation, no interop	Auto Parallel	GENIVI Alliance (2009); early Eclipse Automotive WG — good ideas, slow adoption

Model 3: Standards-Based Consortia — The AUTOSAR Way

- Mechanism: Membership fees & FTEs fund specification; members compete on implementation; standard defines interface, not code.

3GPP (Cellular)	AUTOSAR (Automotive)	W3C / IETF (Internet)
• Mobile ecosystem: >\$4T built on shared standard • Qualcomm patent royalties on LTE/5G: ~\$7-9B/yr • Every phone manufacturer implements the same spec • Competition on chips, devices — not on radio standards	• ~330 members; annual spec releases • Classic Platform: ~100 software components specified • Vendors compete on implementation • Problem: spec ≠ code — implementation inconsistency persists	• HTML, CSS, HTTP, TCP/IP: zero direct revenue • World Wide Web: ~\$30T annual economic contribution est. • Anyone implements; browser competition is fierce • Proves: standards create massive ecosystem value at no cost

Critical gap for automotive: Specification ≠ Code. Slow standard releases cannot keep pace with SDV development velocity.

This gap is exactly what CAPI is designed to close — connecting spec to quarterly working code.

Models 4 & 5: OSS Commercial Overlay & Open Core

MODEL 4: OSS + COMMERCIAL OVERLAY

Mechanism	Core open; revenue from support subscriptions / certified distro
Examples: Red Hat	First OSS company to \$1B revenue (2012) Acquired by IBM for \$34B (2019) Subscription rev: 3x services revenue
Examples: HashiCorp	Support/subscriptions: \$165M (2021) Professional services: \$5M (2021) Acquired by IBM for \$6.4B Proves: subscriptions dominate at scale
Critical Limit	<u>Pure services model: uninvestable.</u> <u>Cloud hyperscalers re-hosting OSS</u> threatened the model (AWS/Azure hosting MongoDB, Elastic → license backlash 2018-2020)

MODEL 5: OPEN CORE (DUAL-TIER)

Mechanism	Free/open community core; paid enterprise editions with advanced features
Examples: MongoDB	Open source 2009; SSPL license 2018 IPO 2017 at \$1.6B; 2024 revenue: ~\$1.7B Open core is the most financially viable OSS model today
Examples: GitLab	IPO 2021 at ~\$11B market cap CE free; EE adds compliance, security, advanced CI/CD
Automotive Relevance	S-CORE is building toward this model: Open non-differentiating core + proprietary value-add by vendors Key question: who certifies the open core for ASiL use?

NOTE: Android is seemingly similar to the Open Core model, but it is not since the GMS license is free and their revenue is data and advertising. It is a Single-vendor controlled open-source platform. A mixture of Model1/2/6.

Model 6: Pre-Competitive Collaborative

MODEL 6: PRE-COMPETITIVE COLLABORATIVE

Mechanism: Competitors co-invest in shared non-differentiating infrastructure; capture value on top/below.

CAPI — AUTOSAR Common AP

- Source-level reference implementation of AUTOSAR AP
- Code-first: possibly quarterly releases, implementation informs spec
- Governed by AUTOSAR consortium — strict AP conformance
- Scope: strictly bounded to the AP layer
- AUTOSAR R25-11 (December 2025)

S-CORE — Eclipse Safe Open Vehicle

- Open-source automotive middleware (Eclipse Foundation, Apache 2.0)
- Founded 2024: BMW, Mercedes-Benz, ETAS, Accenture, Qorix
- Code-first, iterative, community-driven development model
- Scope: AP-layer middleware + above-AP services + tooling
- v0.6.0 (2025) — ISO 26262 + ISO/SAE 21434 aligned

⚠ The Central Tension: functional overlap at the AP middleware layer

-> I will get to this later!

What the Numbers Tell Us

- No model dominates across all dimensions
- The winning formula is selecting the right model for the right layer

Model	Value Creation	Value Capture	R&D	Automotive Fit
1. Closed Proprietary	High (exclusive)	\$14B+/yr peak	Very High (all fees)	Fragmented pre-AUTOSAR era
2. Community OSS	Very High (ecosystem)	Near zero directly	Low (donation-based)	Poor alone (no cert)
3. Standards Consortia	Massive (\$4T+ mobile)	Indirect (membership)	Medium (membership)	Strong — AUTOSAR proven
4. OSS + Services	High	Medium (Red Hat \$34B exit)	Medium (thin margins)	Emerging — vendor distros
5. Open Core	High	High (MongoDB \$1.7B)	High (subscription)	S-CORE direction
6. Pre-Competitive	Very High (shared cost)	Varies by layer	Depends on governance	CAPI + S-CORE now?

The Framework for selecting the right model

Why Different Models Exist — and What Determines Which One Fits

Software Engineering Principles Are Universal

- What varies is who pays, who governs, and who captures value

INVARIANT — Engineering Principles

Modularity: decouple what changes from what stays stable

Abstraction: define interfaces, hide implementation

Separation of concerns: safety / function / performance

Backward compatibility: protect upstream investments

Quality by design: testability, traceability, observability

Safety evidence: FMEA, DFA, safety manual, ASIL analysis

Developer experience: The ease of use

These principles appear in Windows, Linux, AUTOSAR, Android, S-CORE and CAPI — regardless of business model.

VARIABLE — Business Model Parameters

Who funds?

License fees / membership dues / donations /

Who governs?

Single vendor / consortium / foundation / community

Who captures?

Licensor / service provider / platform layer / nobody

Who certifies?

Vendor / TUV / community / nobody (gap!)

Who owns IP?

Company / foundation / public domain / shared

How fast?

Annual releases / quarterly / continuous / committee pace

5 Characteristics That Determine Model Fitness

- 1 Commodity vs. Differentiator — and at What Scale?**
Differentiator → Proprietary. Commodity + large scale (cost beyond any single entity, e.g. Linux ~30M LOC) → pre-competitive collaborative. Commodity + small scale → scale alone does not justify collaboration; governance and interoperability objectives must be the primary driver.
- 2 Safety Criticality & Certification Burden**
Higher safety levels (ASIL-D, SIL-3) require auditable development. | Increases cost of pure community models → creates commercial certification opportunity
- 3 Market Structure: Concentration vs. Fragmentation**
Few dominant players → proprietary feasible (Oracle DB) | Many interdependent OEMs + Tier1s → standards or OSS necessary
- 4 Network Effect & Ecosystem Value**
Does wider adoption increase value for all? | Android: Google chose OSS because Android's value scales with number of devices
- 5 Innovation Velocity Required**
Slow-moving infrastructure → standards body pace OK | Fast-moving domain → code-first collaborative or proprietary

Validation: The Framework Applied Across Industries

Linux Kernel

Model: Community OSS

**Commodity=MAX, Safety=None,
Market=Fragmented, Network=MAX,
Speed=Moderate**

\$0 license, \$500B+ ecosystem. Pure community model works because all 5 characteristics align perfectly.

Android (AOSP)

Model: OSS Platform + Proprietary Services

Commodity=HIGH (OS), Differentiator=HIGH (Search/Maps), Network=MAX

Google makes Android OSS (commodity OS layer) but keeps Search/Maps proprietary (differentiator). 3B+ devices; \$31B/yr mobile revenue enabled by free OS.

Microsoft Windows (1990s)

Model: Closed Proprietary

**Commodity=LOW (novel), Safety=None,
Market=Concentrated, Network=LOW initially**

Proprietary worked: low competition, no interop needed, single platform. Peak \$14B/yr license revenue.

NOTE: Model broke as cloud commoditized and shifted to the Open Core model – **Business model fitness is not static.**

3GPP / LTE Cellular

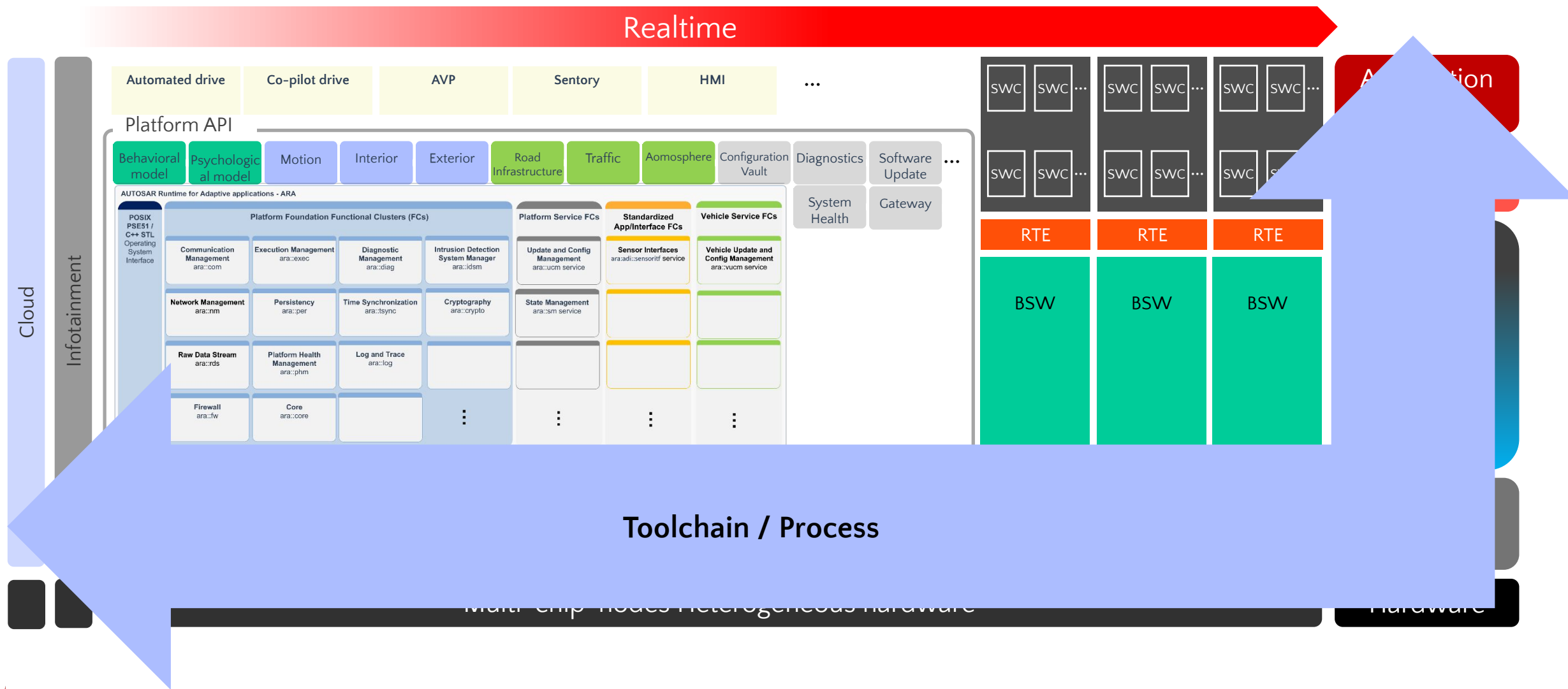
Model: Standards Consortium + Proprietary Implementation

**Commodity=HIGH (radio), Safety=Adjacent,
Market=FRAGMENTED, Network=EXTREME**

\$4T+ economy on shared standard. Qualcomm captures \$7-9B/yr in royalties on top. Proves: when market fragmentation + network effects align, standards win.

Applying the Framework: The SDV Software Stack

Referring to the DVE+U model presented in the Four Principles of SDV at AOC2024



Applying the Framework: The SDV Software Stack

Each layer has a different characteristic profile — and fits a different business model.
CAPI and S-CORE sit at the same layer. This overlap is the central challenge.

Layer	Business Model	Implementation
Vehicle Applications (OEM Features)	<i>Proprietary</i>	<i>OEM differentiating SW</i>
Domain Services & Vehicle Functions	<i>Standards + Open Core</i>	<i>AUTOSAR Adaptive Services layer</i>
S-CORE: Above-AP Services & Tooling	<i>Community OSS (S-CORE unique scope)</i>	<i>Middleware for AP services + DevOps toolchain</i>
AP Implementation Layer — CONVERGENCE ZONE	<i>Pre-Competitive Collab (CAPI) ↔ Community OSS (S-CORE AP components)</i>	<i>~80% overlap: must align on CAPI</i>
OS / Hypervisor	<i>Mixed OSS + Proprietary</i>	<i>Linux, QNX, eSOL eMCOS, etc.</i>
Hardware / SoC	<i>Proprietary</i>	<i>Qualcomm, NXP, Renesas</i>

Proof: The Internet Did This First: Coopetition at Scale

- The internet succeeded not because everyone agreed on everything, but because **each layer had clear governance and stable interfaces to the layers above and below**
- The internet is **built on exactly this multi-model layered architecture** — and nobody calls it fragmented

Internet Layer	Governing Model	Automotive Equivalent	Business Model
TCP/IP, HTTP, TLS	IETF Standards Consortium	AUTOSAR Specification	→ Proven Model
Linux Kernel	Community OSS (Foundation)	Eclipse S-CORE	→ Proven Model
RHEL / Ubuntu	Commercial OSS Overlay	Vendor-certified RTOS + BSW	→ Proven Model
AWS / Azure	Proprietary Cloud Platform	OEM + Tier1 Integration Stack	→ Proven Model
SaaS Apps	Proprietary + OSS mix	Vehicle Applications (OEM features)	→ Proven Model

One Team — Joining the Forces

C-API, S-CORE, and Proprietary Software Working Together

CAPI: From Specification to Living Code

- CAPI: Common AUTOSAR Platform Implementation – closing the spec-to-code gap – **the natural evolution from architectural blueprint to pre-fabricated foundation.**

What CAPI Is

- Source-level reference implementation of AUTOSAR Adaptive Platform
- Automotive-grade, certification-ready C++ code base
- Quarterly releases (vs. AUTOSAR's annual spec releases)
- "Code-first": implementation informs specification through real-world validation
- Governed by AUTOSAR – legally and technically robust
- Not a replacement for AP standard — its living complement

Why CAPI Matters for the Ecosystem

- Eliminates implementation inconsistency between vendors
- Enables S-CORE components to target a defined implementation baseline
- Possibly provide test suites that distributions can run for AP conformance
- Proprietary vendors add certification, tooling, optimization on top
- Continuous feedback loop: code → spec → code
- Governance: AUTOSAR consortium ensures legal clarity and IP protection

Characteristic Profile Fit:

Commodity middleware ✓ | High safety cert burden ✓ | Large multi-stakeholder market ✓ | Strong network effects ✓ | Quarterly velocity needed ✓ → **Pre-competitive Collaborative Model** ✓

Eclipse S-CORE: Assessment and Path to Collaboration

Eclipse S-CORE — Three Distinct Zones of Scope

ZONE 1: AP Implementation

~80% overlap with CAPI

Execution management, IPC, logging, persistency, OS abstraction, communication management

This is where CAPI and S-CORE directly conflict.

Resolution: S-CORE contributors align these components with CAPI conformance. The business model for contributors is fully preserved — they continue to build certified distributions and value-add on top.

The path forward: Zone 1 aligns with CAPI — contributors keep their business model. Zones 2 and 3 continue under S-CORE — no compromise needed.

This is not S-CORE losing ground. It is S-CORE focusing on what it uniquely offers.

ZONE 2: Above-AP Services

Unique to S-CORE — no conflict

Higher-level vehicle service abstractions, application orchestration frameworks, domain-specific service libraries

This scope is explicitly out of CAPI's boundary.

S-CORE owns this space fully. No convergence needed. This is where S-CORE's unique contribution to the SDV ecosystem lives.

ZONE 3: Tooling & DevOps

Unique to S-CORE — no conflict

Automotive-grade CI/CD pipelines, integration toolchains, safety process tooling, ASPICE 4.0 compatible DevOps infrastructure

Also, explicitly out of CAPI's scope.

S-CORE's tooling contributions are a distinct and genuinely valuable addition to the ecosystem — independent of the AP implementation debate.

Proprietary Software: Where It Belongs and Why

- Proprietary Software Is NOT the Enemy – Proprietary at the wrong layer is the problem. **Proprietary at the right layer is essential.**

Where Proprietary IS Appropriate

- ✓ **Toolchains:** compilers, debuggers, model-based dev tools
- ✓ **Certified RTOS + BSW distributions**
- ✓ **SoC-specific BSP and hardware abstraction optimization**
- ✓ **OEM-specific vehicle functions and feature differentiators**
- ✓ **System integration and long-term maintenance contracts**
- ✓ **Liability-bearing product delivery** (certified by TUV/SGS etc.)

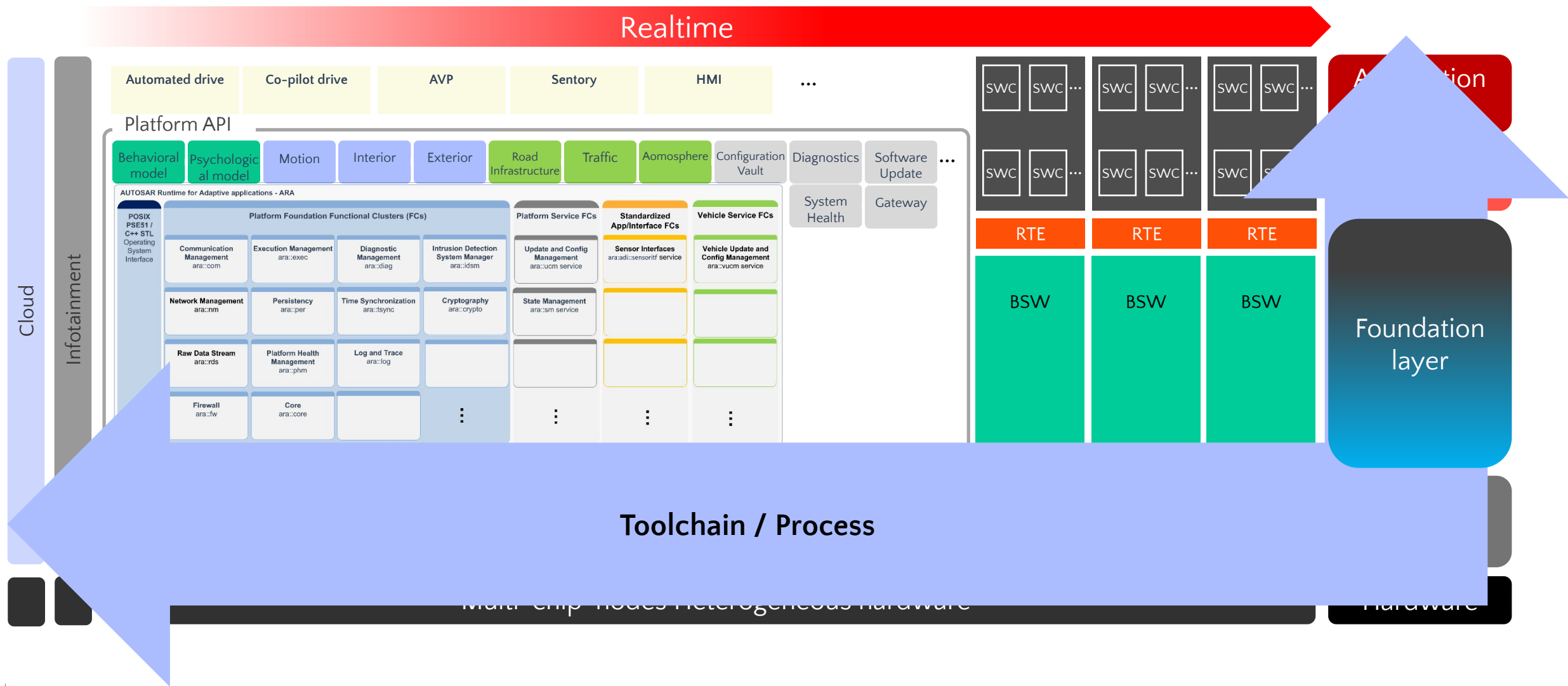
Where Proprietary Creates Problems

- × **Commodity middleware** (communication, OS abstraction) → duplication
- × Basic software startup sequences → inconsistent, costly
- × **API definitions for standard services** → interop fragmentation
- × **Test suites for AP conformance** → every vendor tests differently
- × **Safety certification process templates** → wheel reinvented 330x
- × **Documentation formats for safety evidence** → each vendor uses different formats, making system-level safety case assembly difficult and costly

The role of proprietary software vendors in the One Team model: add certification, integration, performance optimization, and long-term support for the shared CAPI + S-CORE foundation, in addition to differentiating lower/higher layer IPs and toolchains.

Architectural Vision: One Ecosystem, Defined Boundaries

Referring to the DVE+U model presented in the Four Principles of SDV at AOC2024



Architectural Vision: One Ecosystem, Defined Boundaries

Layer	Business Model	Implementat	Role
OEM Vehicle Applications	<i>Proprietary</i>	<i>OEM Differentiating SW</i>	<i>OEM feature delivery</i>
Domain Services & APIs	<i>Standards + Open Core</i>	<i>AUTOSAR AP Services layer</i>	<i>Function-specific middleware</i>
S-CORE: Above-AP & Tooling	<i>Community OSS (S-CORE)</i>	<i>Middleware for AP services + DevOps tooling</i>	<i>S-CORE unique scope — no conflict</i>
AP Implementation — CONVERGENCE ZONE <i>CAPI (primary) ↔ S-CORE Zone 1</i>	<i>Pre-Competitive Collaborative</i>	<i>AUTOSAR AP reference code</i>	<i>Quarterly releases; conformance; no single-partner dominance</i>
OS / Hypervisor	<i>Mixed OSS + Proprietary</i>	<i>Linux + eSOL eMCOS/QNX + Hypervisors</i>	<i>Core OS layer</i>
SoC / Hardware	<i>Proprietary</i>	<i>Qualcomm, NXP, Renesas...</i>	<i>HW-specific BSP by vendors</i>

Call to Action: Joining the Forces — Five Concrete Steps

AUTOSA R	Complete governance reform: move <u>from unanimity to majority-based decision-making</u> . Without this, the speed advantage of CAPI is structurally undermined.
AUTOSA R	Publish and enforce transparent CAPI contribution guidelines: Ensure <u>no single partner dominates the codebase</u> . CAPI must be a genuine industry commons, <u>not an open-core</u> product in disguise.
AUTOSA R + S-CORE	Formally agree the scope boundary: <u>AP implementation layer → CAPI. Above-AP services and tooling → S-CORE</u> . Publish this as a joint statement. Make it actionable for the supply chain.
S-CORE	Formally align Zone 1 (AP-layer) components with CAPI conformance — phased, not overnight. The business model is preserved: certified distributions, integration services, and value-add on top remain the commercial foundation. Aligning with CAPI strengthens, not
OEMs	Express demand, not just membership. (Publicly) commit to production roadmaps that require CAPI conformance at the AP layer by SOP 2030. <u>OEM demand signals are the most powerful forcing function for convergence.</u>

One Team, one more time – let us work together

- A slide from my talk at AOC2024 Tokyo – Four Principles of SDV

Let us work together

The automotive industry faces the question –
“To be, or not to be”

Will we work together to live, or die slowly?

Because it is a smartphone on wheels, it will
eventually come down to a few platforms
through generations.

Will we play the geopolitical game or execute
Open-closed principle and collaborate in the
open standards?

I still believe AUTOSAR as one of few grounds
we can **play together**



[Intro: Aladdin and Jasmine]

How are you doing that?

It's a magic carpet

I-it's lovely...

You don't...you don't wanna go for a ride, do you?

We could get out of the palace, see the world

Is it safe?

Sure, do you trust me?

What?

Do you trust me?

Yes...

AUTOSAR™

Gondo, eSOL - AOC 2024 Tokyo

<https://www.brockton.com/kids.com/aladdin-coloring-pages.html>

26