Software-Defined Vehicle (SDV): Architecture, Technologies, and Practical Implementation

A comprehensive technical reference for automotive engineers, system architects, and SDV practitioners

Authored by

Ms. Kavitha Sasikumar
Senior Embedded & SDV Trainer, ANCIT

Technical review by

Dr. Kaarthick Balakrishnan
Research Director & Head of Engineering, ANCIT

In this whitepaper, you will understand the complete architecture of the Software-Defined Vehicle — from E/E architecture evolution and service-oriented communication to high-performance computing, software design, artificial intelligence, and CI/CD pipelines. This is a practitioner's reference for engineers, architects, and technical leaders building vehicles that improve over their lifetime.

1. What is a Software-Defined Vehicle?

In a conventional vehicle, every capability is locked into hardware at manufacture. A new feature means a new ECU, new wiring, and months of integration work. The Software-Defined Vehicle (SDV) eliminates this constraint. The hardware provides a fixed, standardised compute and communication platform. Software — deployed over the air throughout the vehicle's lifetime — defines what the vehicle can do and continuously redefines it.

An SDV treats the vehicle as a software platform: the hardware is fixed and standardised; software defines its capabilities and evolves continuously throughout its lifetime.

1.1 The Core Principle: Hardware-Software Decoupling

A hardware abstraction layer sits between application software and physical hardware. Applications communicate with this layer, not with hardware directly. The consequences:

- Software updates change vehicle behaviour without any hardware modification
- The same application runs across different hardware generations without rewriting
- New features are deployed to vehicles already in the field, after sale
- Multiple applications safely share the same physical compute resources

1.2 Conventional Vehicle vs. SDV

Characteristic	Conventional Vehicle	Software-Defined Vehicle
Functionality	Set at design time; cannot change without hardware replacement	Defined in software; extended via OTA throughout vehicle life
Architecture	150+ isolated ECUs, point-to-point wiring	Centralised compute nodes with standardised software interfaces
Software updates	Workshop visit; individual ECU reflash	OTA to any software layer; no workshop visit required
Feature lifecycle	Ages with vehicle; no post-sale expansion	Improves over time; new features deployed to vehicles in field

2. Why SDV is Needed

The SDV transition is driven by five converging forces. No single one could be ignored; together they make the transition a strategic and engineering imperative.

2.1 Consumer Expectation

Tesla demonstrated that a 2018 vehicle could have meaningfully improved autopilot, range, and features in 2023 — no hardware change. This set a new benchmark: vehicle quality now includes how the vehicle performs two years after purchase. OEMs whose vehicles cannot improve after sale face a structural competitive disadvantage.

2.2 Architectural Scaling Limit

A modern premium vehicle has 150+ ECUs, 5+ km of wiring, and up to 300 million lines of code across incompatible environments. Adding one new function typically takes 18–24 months and costs millions. The distributed ECU architecture has reached its physical and economic scaling limit.

2.3 Post-Sale Software Revenue

Features like enhanced autopilot, performance upgrades, and premium services can be provisioned post-sale via software. The hardware is installed at manufacture; the feature is activated through an authenticated OTA update. Without SDV architecture, this model is not technically achievable.

2.5 Competitive Pressure

New entrants — Tesla, Rivian, NIO — were built as software platforms from day one. Volkswagen Group created CARIAD with a multi-billion euro mandate. BMW and Toyota are building proprietary vehicle operating systems. The direction of the industry is clear.

2.4 Regulatory Requirements

- **UNECE R155 (Cybersecurity)** — Requires a certified Cybersecurity Management System for new type approvals from July 2024. OEMs must patch fielded vehicles remotely — only achievable with OTA.
- **UNECE R156 (Software Updates)** — Mandates a regulated Software Update Management System. Remote OTA capability is a legal requirement for new type approvals.
- **ISO/SAE 21434** — Requires cybersecurity maintenance across the full vehicle lifetime, including remote remediation of fielded vehicles.

> ***Key Takeaway:*** *The SDV transition is driven by consumer expectations of post-sale improvement, the scaling limits of distributed ECU architecture, post-sale software revenue models, binding regulatory mandates for OTA capability, and competitive pressure from software-native entrants.*

3. Clearing the Confusion: SDV, ADAS, Connected Vehicle, and OTA

SDV is frequently used interchangeably with ADAS, Connected Vehicle, and OTA updates. These are related but distinct concepts.

- **A Software-Defined Vehicle** — is an architectural concept — a vehicle whose functional software is decoupled from hardware through a standardised abstraction layer, making capabilities extensible via software throughout the vehicle's lifetime.
- **ADAS** — refers specifically to the perception, decision, and control software functions that assist or automate driving tasks. A vehicle can have ADAS on fixed ECUs that cannot be updated — that vehicle is not software-defined.
- **A Connected Vehicle** — has wireless interfaces enabling data exchange with the cloud. Connectivity is necessary for OTA delivery in an SDV, but a vehicle can send cloud telemetry while its core architecture remains hardware-bound.
- **OTA updates** — are an operational capability — delivering software without a workshop visit. In a true SDV, OTA covers all software layers including safety-relevant systems.

The distinction: SDV is the architecture. ADAS is one application built on it. Connectivity is the communication channel. OTA is the operational outcome. All must work together — but none of the three individually makes a

vehicle software-defined.

4. Realising the SDV: The Five Engineering Pillars

Building an SDV requires five interdependent engineering disciplines. Each is a prerequisite for the others. A gap in any pillar limits the capability of all the rest.

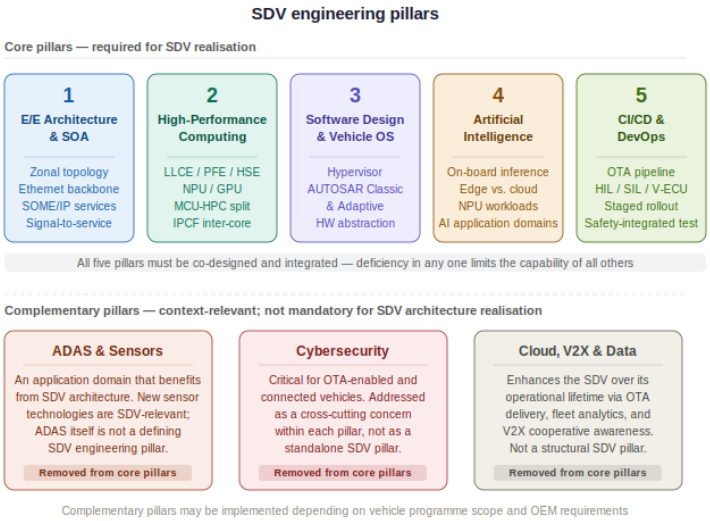

Figure 1: Engineering Pillars for SDV Realisation

5. Pillar 1 — E/E Architecture & Service-Oriented Architecture

The E/E architecture defines how computing, communication, and power are organised across the vehicle. It is the structural foundation: every other pillar depends on it for the physical infrastructure it provides.

5.1 Architecture Evolution

Distributed Architecture

Each function has its own dedicated ECU. Separate ECU clusters for Powertrain, Chassis, Body, Infotainment, and ADAS are connected by domain-specific bus protocols — CAN, LIN, FlexRay, MOST. Software is written for a specific ECU and cannot move. The wiring harness in a premium vehicle exceeds 60 kg. The core drawback: software is inseparable from hardware. The vehicle cannot evolve after it leaves the factory.

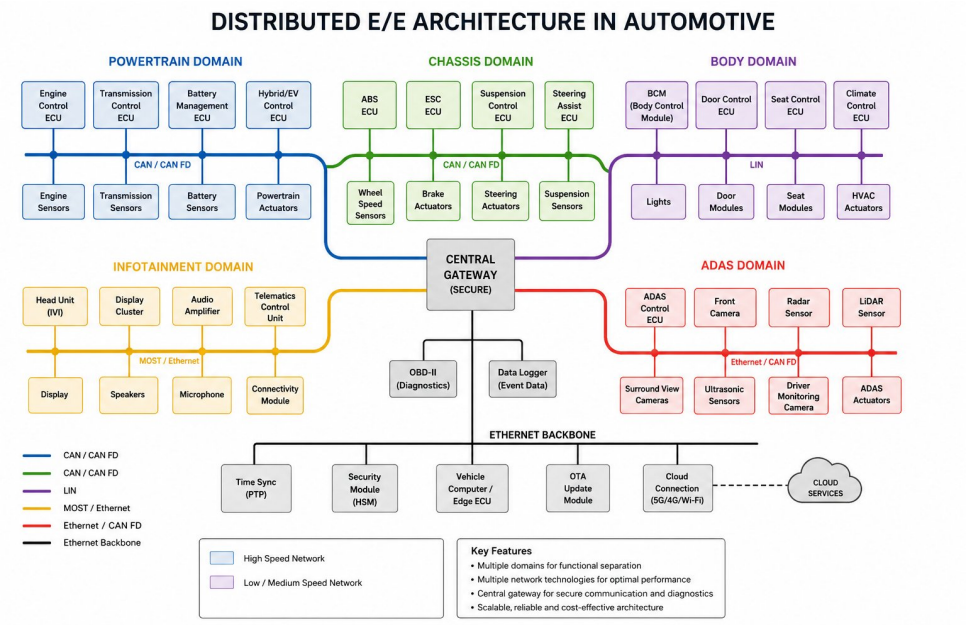

Figure 2: Distributed E/E Architecture — Five isolated domain clusters with dedicated ECUs and domain-specific bus protocols connected through a central gateway

Domain-Centralised Architecture

A more powerful Domain Controller aggregates multiple ECUs within each functional domain — Powertrain, Chassis, ADAS, Infotainment. This reduces wiring within each domain and consolidates compute. However, the domains themselves remain isolated silos. Software still cannot cross domain boundaries.

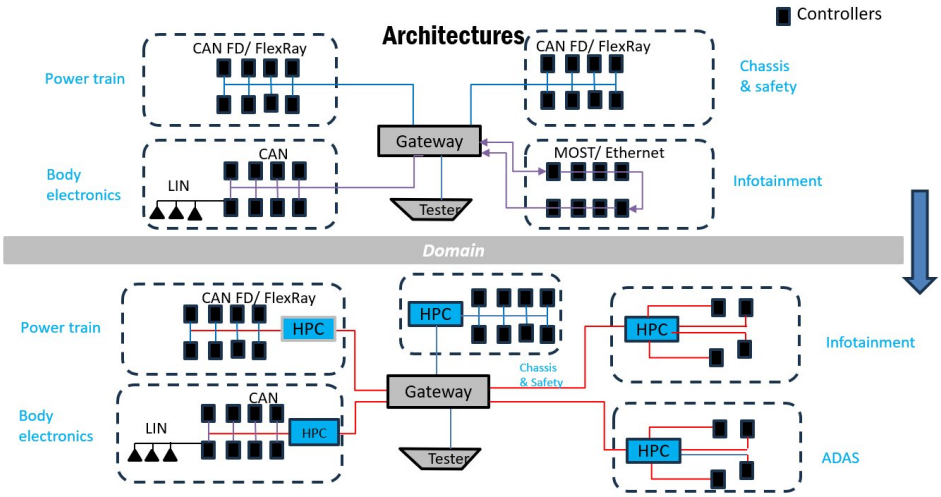

Figure 3: Architecture Transition — Distributed (top) evolving to Domain-Centralised (bottom) with per-domain HPCs, leading to Zonal architecture

Zonal Architecture — The SDV Foundation

The vehicle is divided by physical location (Front, Rear, Left, Right). A Zonal Controller (ZC) serves each zone, handling all functions regardless of domain. Central HPCs connect to all ZCs over Ethernet.

- Wiring reduced by up to 50% — each zone connects via one Ethernet cable instead of multiple domain buses

- Functions migrate from fixed ECUs to software services on the HPC, removing hardware-software coupling
- Applications address sensors and actuators through a vehicle API layer, not through hardware directly
- New functions are software deployments, not hardware additions

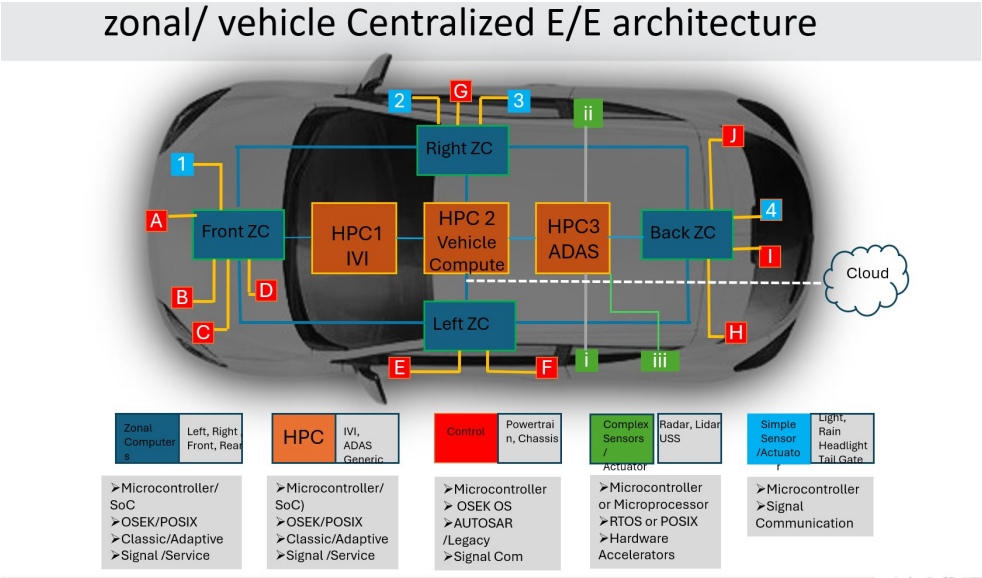

Figure 4: Zonal/Vehicle-Centralised Architecture — Zonal Controllers connected to central HPCs via Ethernet backbone

5.2 Service-Oriented Architecture

In a traditional vehicle, communication uses a static signal matrix defined at build time. In a Service-Oriented Architecture, components publish named services with versioned interfaces. Any consumer discovers and subscribes at runtime, without knowing which hardware provides the service. When hardware changes, only the service implementation changes — all consumers continue through the unchanged interface.

SOME/IP — The Primary SDV Protocol

SOME/IP (Scalable service-Oriented MiddlEware over IP) is the standard in-vehicle SOA protocol in AUTOSAR Adaptive. It runs over Ethernet and supports RPC, Events (publish-subscribe), and Fields (stateful attribute). SOME/IP-SD allows any HPC or ZC to find services dynamically at runtime.

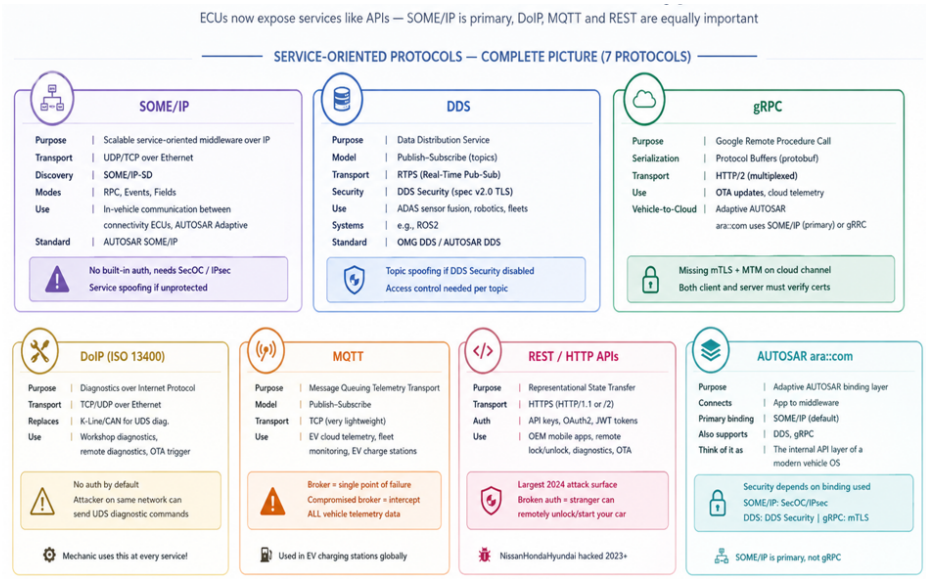

Figure 5: Service-Oriented Protocols — SOME/IP as the primary in-vehicle protocol with context-specific alternatives

SOME/IP is the default for all in-vehicle service communication in Adaptive AUTOSAR deployments; DDS is preferred for high-frequency sensor data pipelines; gRPC for vehicle-to-cloud communication; DoIP for diagnostic sessions; MQTT for lightweight fleet telemetry; REST/HTTP for OEM mobile and web API integration; and ara::com is the internal Adaptive AUTOSAR binding layer.

> ***Key Takeaway:*** *Zonal architecture provides the physical infrastructure. SOA provides the communication model. Without both, the SDV cannot be realised. Every other pillar runs on top of what this one provides.*

6. Pillar 2 — High-Performance Computing

The HPC is the central compute platform where SDV services run, where AI models execute, and where OTA updates are managed. Without it, the zonal architecture is wiring without intelligence.

6.1 Why MCUs Are Not Sufficient

MCUs are excellent for single-function deterministic control — braking, airbag, EPS. But they cannot host Linux, support virtual machines, run containerised services, or sustain neural network inference. The HPC addresses all of this through multicore ARM processors, gigabytes of RAM, POSIX OS support, and domain-specific hardware accelerators.

6.2 Hardware Accelerators

- **LLCE** — Handles all CAN/LIN/FlexRay processing in dedicated hardware. Without it, every CAN frame triggers a CPU interrupt. With it, CAN processing is fully offloaded.
- **PFE** — Routes Ethernet packets between zones at wire speed in hardware. Without it, inter-zone routing consumes CPU cycles proportional to traffic.
- **HSE** — Executes AES, ECDSA, and SHA operations in tamper-resistant hardware. Keys provisioned into the HSE cannot be extracted by software at any privilege level.
- **NPU/GPU** — In ADAS-class HPCs, dedicated neural processors run AI inference at TOPS scale. A pedestrian detection network that takes 2+ seconds on a general-purpose CPU runs in under 10ms on a dedicated NPU.

6.3 MCU or HPC: The Decision

Criterion	MCU (Zonal Controller)	HPC
Compute requirement	Low — control loops, signal conditioning	High — AI inference, service orchestration
Safety certification	ASIL D — braking, steering, airbag	QM to ASIL B via hypervisor partition
Update frequency	Rare — full re-certification per change	Frequent — service-level OTA, AI model refresh

The IPCF (Inter-Platform Communications Framework) connects MCU real-time cores and HPC application cores on the same SoC via shared SRAM and hardware interrupt signalling — with sub-millisecond bounded latency.

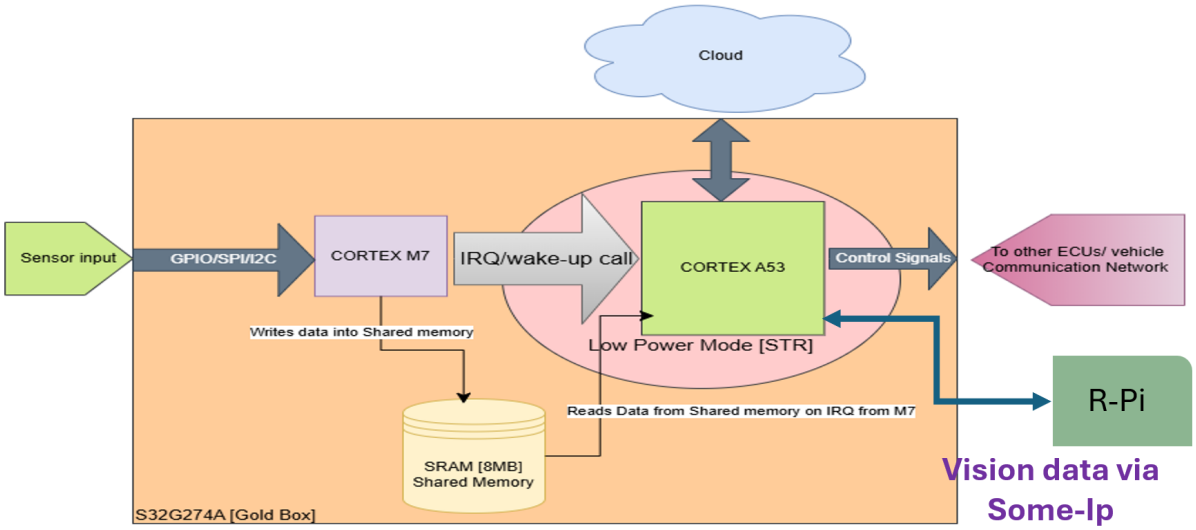

Figure 6: IPCF Inter-Core Communication — Cortex-M7 writes sensor data to shared SRAM and signals Cortex-A53 to wake and process

> ***Key Takeaway:*** *HPC hardware accelerators are not optional enhancements — they are the mechanisms through which CAN processing, cryptographic security, and AI inference meet real-time automotive requirements.*

7. Pillar 3 — Software Design & Vehicle Operating System

The right hardware is necessary but not sufficient. The software architecture — OS, virtualisation, middleware, and application framework — determines whether the vehicle evolves over its lifetime or stagnates.

7.1 Software Abstraction: A Concrete Example

Consider a radar sensor used for speed regulation. Without abstraction, the application is written to that specific radar's hardware interface. When the OEM upgrades to a newer radar, the application must be rewritten and re-certified. With abstraction, the application is written to a standardised 'radar service' interface. Only the service driver changes.

7.2 Mixed-Criticality: Hypervisor Isolation

An SDV HPC must simultaneously run ASIL-D certified braking and steering software alongside Quality-Managed infotainment and navigation applications — on the same physical chip. A Type-1 hypervisor resolves this through hardware-enforced partitioning. Each partition receives its own dedicated OS — QNX Neutrino or AUTOSAR OS for the safety domain, Linux or Android Automotive for the non-safety domain. A fault or security breach in the infotainment partition is physically prevented from propagating to the braking partition by the hypervisor's hardware boundary.

7.3 AUTOSAR Classic vs. Adaptive

Aspect	AUTOSAR Classic	AUTOSAR Adaptive
Target hardware	Microcontrollers (MCU)	High-Performance Computers (HPC)
Communication	Static — full schedule defined at build time	Dynamic — services discovered at runtime via SOME/IP-SD

Aspect	AUTOSAR Classic	AUTOSAR Adaptive
OS requirement	AUTOSAR OS (OSEK-based)	POSIX OS (Linux, QNX)
Update model	Full ECU reflash; requires re-certification	Service-level OTA; independent update per service
Typical use	Braking, EPS, airbag, body control (ZC)	Gateway, telematics, OTA management, AI inference (HPC)

> ***Key Takeaway:*** *Software abstraction removes hardware dependency. The hypervisor enables mixed-criticality on one chip. Classic AUTOSAR handles safety ECUs; Adaptive AUTOSAR handles HPC services. Together they form the software foundation of the SDV.*

8. Pillar 4 — Artificial Intelligence in SDV

AI enables the vehicle to perform tasks that rule-based software cannot handle: recognising objects in complex scenes, predicting road user behaviour, detecting early component failure, and adapting to individual drivers.

8.1 Hardware Determines AI Capability

AI Task	On ARM CPU	On Dedicated NPU/GPU
Pedestrian detection (CNN)	>2,000 ms — not deployable	<10 ms — meets safety budget
Sensor fusion at 20 Hz	2–3 Hz achievable — insufficient	20 Hz sustained — highway safe
Voice LLM inference	Several seconds — unusable in-car	Sub-second — usable real-time

8.2 AI Application Domains

- **Obstacle perception** — CNNs and BEV transformer models on the ADAS HPC NPU provide real-time environment understanding. This is the primary safety-critical AI application in the vehicle.
- **Driver monitoring** — Facial landmark detection and gaze tracking detect drowsiness and distraction. Mandatory under EU General Safety Regulation from 2024.
- **Predictive maintenance** — Anomaly detection models predict component failures 200–500 hours ahead, reducing unscheduled downtime and safety incidents.
- **Personalisation** — Preference learning models configure seat, climate, route, and media settings per individual driver without manual input.
- **Voice interaction** — Quantised LLM inference on the HPC NPU enables natural language vehicle control without a touch interface.

8.3 On-Board vs. Cloud AI

On-board execution on the HPC with hardware accelerators is the only viable context for real-time vehicle functions. Cloud round-trip latency is 50–200 ms; real-time vehicle decisions require responses in under 50 ms. Cloud AI has genuine value for non-real-time functions: AI model retraining from fleet data, predictive maintenance analysis, and OTA scheduling.

> ***Key Takeaway:*** *AI pillar value is hardware-determined. On-board, hardware-accelerated AI is the only viable path for real-time functions. The AI pillar must be co-designed with the HPC pillar from the start.*

9. Pillar 5 — CI/CD and Automotive DevOps

The SDV's ability to continuously improve depends on the engineering process used to create, validate, and deploy software. Automotive CI/CD differs from consumer software CI/CD in one critical respect: a failed deployment can have safety consequences. Every stage must generate evidence of correctness and provide a clear rollback path.

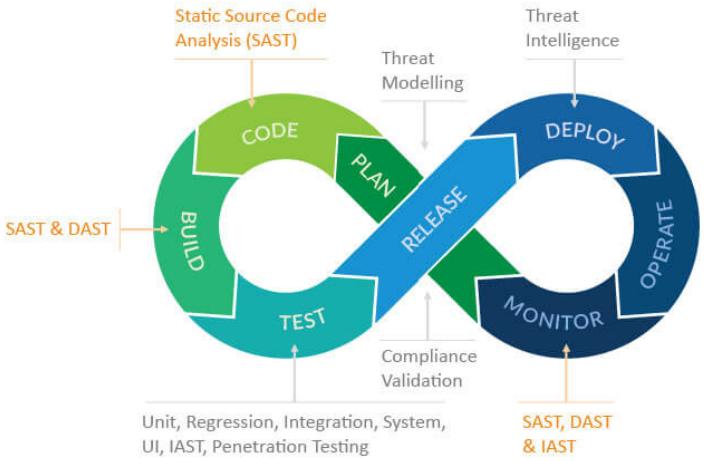

Figure 7: Automotive CI/CD Pipeline — Continuous loop from Plan through Monitor with safety-integrated testing and regulated OTA deployment

9.1 Pipeline Stages and Requirements

Stage	What Happens
Plan	Decide what to build and check if it affects safety software. Map all changes before coding starts.
Code	Write software following MISRA/AUTOSAR coding rules. Every change is peer-reviewed before acceptance.
Build	Compile the full software automatically. Run SAST/DAST security and quality scans on every submission.
Test	Validate at MIL, SIL, and HIL levels. Real hardware (HIL) testing is mandatory for safety software.
Release	Document all software components (SBOM), complete regulatory checklists, and digitally sign the package.
Deploy	Roll out to 1% of fleet, monitor, then expand to 10%, then 100%. Auto-reverse if thresholds are crossed.
Operate	Monitor all vehicles continuously for faults. Auto-rollback if a safety threshold is exceeded.
Monitor	Feed field data into the next development cycle. Fast-track security fixes. Retrain AI on edge cases.

9.2 Virtual ECU: Software Development Without Hardware

Virtual ECU (V-ECU) technology runs automotive software in a simulated hardware environment, removing the 18–24 month hardware dependency from early development.

Level	Type	Use Case
L0	Model-in-the-Loop (MIL)	Control algorithm verification in MATLAB/Simulink

Level	Type	Use Case
L1	SIL — Application Layer	AUTOSAR SWC testing with BSW stubs; no physical BSW required
L2	SIL — Simulated BSW	Full software stack integration with virtual peripherals
L3	SIL — Production SW on Virtual HW	Pre-HIL validation on QEMU-emulated target hardware
L4	Full V-ECU	System integration, SOME/IP communication, OTA pipeline testing

10. Conclusion

> *"The software-defined vehicle is not a single technology decision. It is the result of five engineering disciplines — architecture, HPC, software design, artificial intelligence, and continuous delivery — working together as one coherent system. Each pillar depends on the others. The E/E architecture provides the physical foundation; the HPC provides the processing power; software design makes that platform evolvable; AI makes it intelligent; and CI/CD makes the entire system continuously improvable after the vehicle leaves the factory. When all five are in place and integrated correctly, the vehicle stops being a product that depreciates and becomes a platform that grows. That is the SDV."*

References

[01]	Bosch Mobility Solutions — Software-Defined Vehicle: Architecture, Platform and Technology Overview. Robert Bosch GmbH, Stuttgart.
[02]	NXP Semiconductors — S32G274A Vehicle Network Processor Reference Manual (Doc. No. S32G2RM). NXP Semiconductors N.V.
[03]	AUTOSAR Consortium — Adaptive Platform Specification R22-11; Classic Platform Specification R21-11. AUTOSAR GbR.
[04]	ISO/SAE 21434:2021 — Road Vehicles — Cybersecurity Engineering. ISO / SAE International.
[05]	ISO 26262:2018 — Road Vehicles — Functional Safety, Parts 1–12. ISO.
[06]	UNECE WP.29 — UN Regulation No. 155 (Cybersecurity/CSMS) and No. 156 (Software Updates/SUMS). United Nations Economic Commission for Europe.
[07]	SAE International — SAE J3016: Taxonomy and Definitions for Driving Automation Systems. SAE International.

*This whitepaper is authored by **Ms. Kavitha Sasikumar**, Senior Embedded and SDV Trainer at ANCIT, and reviewed by **Dr. Kaarthick Balakrishnan**, Research Director and Head of Engineering, ANCIT — a leading automotive engineering training company specialising in AUTOSAR, Embedded Systems, Functional Safety, and Software-Defined Vehicles. For training enquiries, contact **info@ancitconsulting.com** or call **+91 9840378602**.*