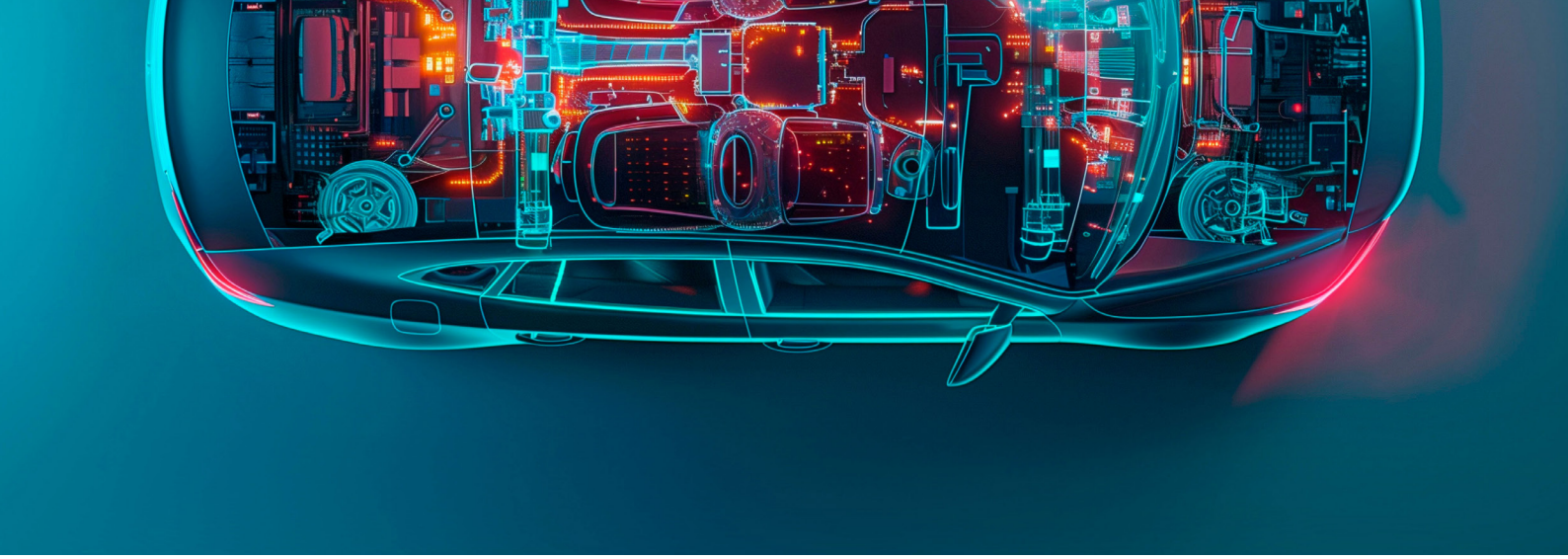
Volume 2

SDV Pulse

The technology driving the future

thoughtworks

2026



Foreword

The automotive industry stands at the threshold of its most profound transformation. The convergence of advanced software engineering, cloud-native technologies and evolving user expectations has given rise to a new class of intelligent, adaptive machines: **software-defined vehicles (SDVs).**

In this fast-evolving world, the vehicle is no longer a static product, but a dynamic platform — continuously updated, increasingly personalized and deeply integrated into the digital fabric of daily life.

With this shift, the role of software is no longer supportive; it's foundational. It defines performance, user experience, safety, monetization and even brand identity. OEMs, Tier 1s, startups and established technology companies are now constantly looking for new ways to harness the power of software to create tomorrow's mobility solutions.

At Thoughtworks, we collaborate with many of these automotive and tech leaders to solve some of the most complex SDV challenges. These close partnerships allow us to continuously learn about what works, what doesn't, and what might come next — lessons that we've compiled in this publication.

We intend this guide to be an annual compilation shaped by your feedback, ideas and continued contributions, helping us all stay ahead in the ever-shifting landscape of software-defined mobility.

Together, we can build the future of software-defined mobility.

Rachel Laycock
Chief Technology Officer
Thoughtworks

Industry snapshot

Over the past year, the automotive industry has seen four major shifts that mark 2024–25 as a defining period in the maturation of SDVs.

Global markets

The past year has highlighted diverging approaches between Western incumbents, startups and Chinese challengers. In the West, Polestar secured new funding to reinforce its premium EV growth, and Rivian streamlined operations while preparing its mass market R2. Traditional OEMs doubled down on their SDV investments and pivoted toward platform-based software development and alliances to spread R&D risk.

Meanwhile, Chinese OEMs accelerated their global push. BYD began localized production in Europe and launched dedicated shipping fleets, and Xiaomi rapidly expanded hiring and factories. The result is a market where Western OEMs are pursuing disciplined, software-focused growth while Chinese firms drive scale and speed.

Regulatory landscape

The US finalized its Connected Vehicle Rule, banning designated Chinese and Russian software from Model Year 2027 and extending restrictions to hardware by 2030. Europe is also tightening AI and cybersecurity oversight through instruments such as the EU AI Act and NIS2. China is expected to respond with reciprocal measures, underscoring the growing role of geopolitics in automotive software supply chains.

Talent

Automotive software engineering increasingly requires the fusion of mechanical, electronics and software skills. Organizations are investing in upskilling and re-skilling programs while shifting engineering talent to nearshore and offshore hubs to meet scaling needs. This reflects the demand for cross-domain expertise and the strategic rebalancing of global R&D footprints.

Safety and certification

The industry is advancing its safety toolchain, with Ferrocene securing ISO 26262 certification for Rust and proving the language viable for safety-critical systems. Frameworks such as the Trustable Software Framework are also gaining traction as practical ways to provide continuous evidence for safety and cybersecurity compliance. These moves reflect a shift from ad hoc testing to structured, auditable and AI-aware certification regimes.

/thoughtworks

Design. Engineering. AI.

About Thoughtworks

Thoughtworks is a leading global technology consultancy that blends design, engineering and AI expertise. For over 30 years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology, and today we're at the forefront of AI-powered software and data engineering.

Our unique combination of expertise in digital strategy, software engineering, design and organizational transformation has made Thoughtworks a leading partner for clients including BMW, Ford, Mercedes-Benz and Porsche.

About SDV Pulse, volume 2

SDV Pulse is our annual reflection on the evolving world of software-defined vehicles — a landscape that continues to shift rapidly as innovation, regulation and real-world adoption mature throughout the ecosystem.

The goal of SDV Pulse is simple yet deliberate: to track key ideas and topics not by their hype, but by their **real merit.** Each year, we revisit the Pulse points through fresh eyes, evaluating what's gaining traction, what's maturing and what's entering the spotlight for the first time.

In this volume, you'll find that **some Pulse points have held their ground** and others have **progressed to newer stages of adoption,** reflecting shifts from experimentation to scale. And, as always, there are **new entrants** — emerging ideas and innovations that have captured the industry's imagination and demand attention in the future.

By grouping these Pulse points under five macro themes, we offer a structured yet dynamic view of SDV evolution.

Our intent is not to predict the future, but to offer clarity and perspective, helping practitioners, strategists and technologists look beyond the hype and focus on what truly matters.

We hope this volume of SDV Pulse sparks conversation, challenges assumptions and supports decision-making across the software-defined mobility value chain. And we look forward to continuing this journey with you, year after year.

SDV Pulse at a glance

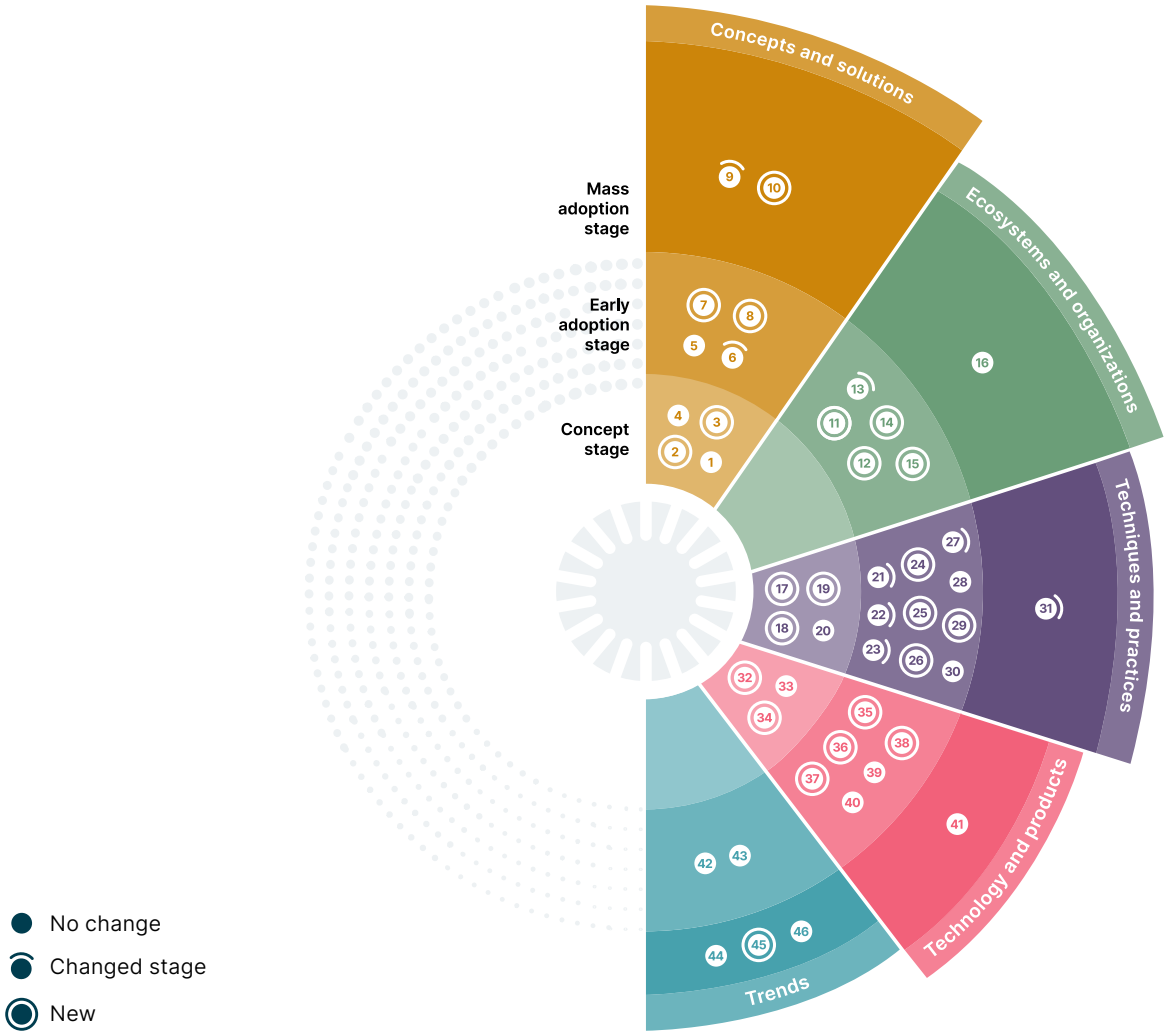
We've designed the Pulse chart to help automotive organizations keep track of relevant SDV trends, developments and opportunities.

SDV Pulse is divided into five themes: concepts and solutions, ecosystem and organization, techniques and practices, technology and products, and trends. The Pulse points within these themes are organized by their adoption stage.

Each theme contains various technologies, practices or trends. The adoption stages, defined by our experts, show where we believe each technology, practice or trend currently falls.

- **Concept stage:** Pulse points at this stage are currently emerging and their potential remains largely unproven. In most cases, the vast majority of organizations won't have adopted these concepts yet.
- **Early adoption stage:** These Pulse points are early on in their adoption journey and offer a significant differentiation opportunity for early adopters.
- **Mass adoption stage:** These Pulse points are becoming widespread across the industry and the window to translate them into differentiated value is closing.

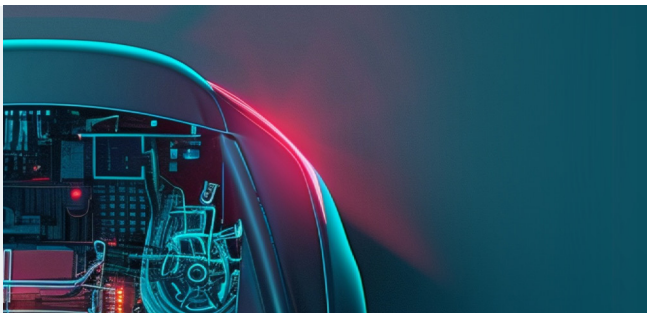
In future editions, we'll fade concepts that we see becoming less relevant, helping you track only the most relevant trends.



SDV Pulse themes

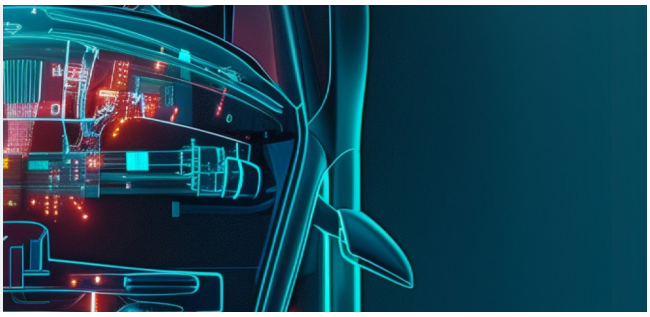
Concepts and solutions

These Pulse points indicate capabilities that are available on the market as packaged solutions that manufacturers can purchase and apply within their vehicles or use to transform their internal operations.



Ecosystems and organizations

The Pulse chart also includes emerging organizations and ecosystems that should be on every manufacturer's radar. These include ecosystems that you may wish to integrate with and become part of, as well as organizations such as regulatory bodies that could impact your operations and engineering decisions.

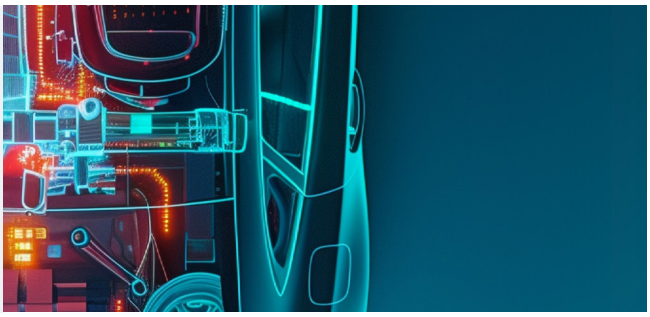


Techniques and practices

This category includes new ways of working that can help SDV manufacturers evolve the way their teams operate and enable them to deliver better results and driver outcomes.

Technology and products

Pulse points that fall into this theme represent leading technologies that mobility ecosystem stakeholders can incorporate or apply within their engineering organizations to transform experiences and deliver new value.



Trends

This category covers all other relevant trends that don't fall into the four areas above. Many of these are general evolutions taking place within software engineering organizations that OEMs and 1st tier suppliers should be aware of.

Contributors

Michael Fait
Developer, Technical Director, Head of Technology SDV

Dr.-Ing. Sebastian Werner, Global SME Automotive & Manufacturing Solutions, Europe

Sriram Jayaraman
Client Partner, Auto and Manufacturing, Business strategist SDV and SDX

Vignesh Radhakrishnan
Developer, Lead Consultant

Vivek Poovalingam
Principal Solution Architect SDV

Reviewers

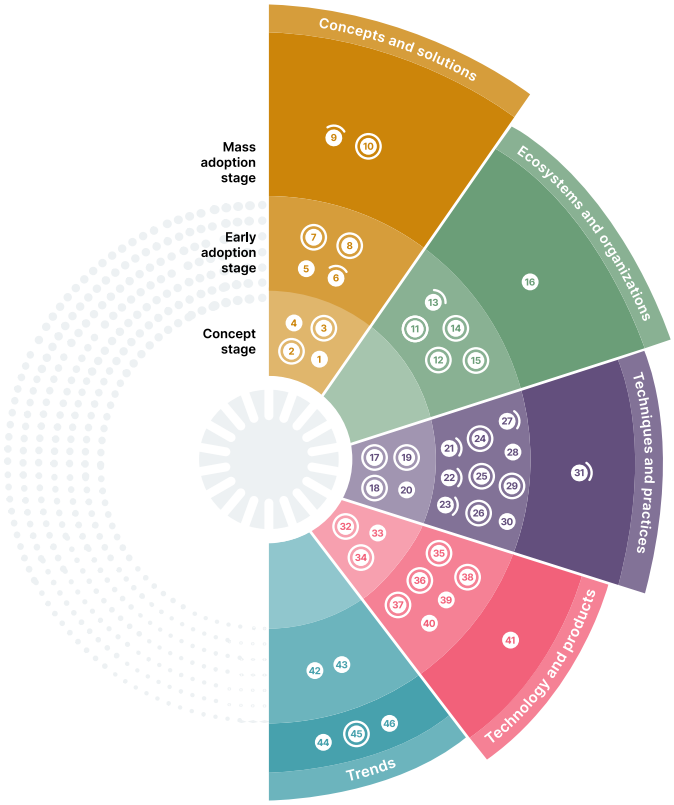
Daniel Schleicher
Senior Global Solutions Architect Automotive and Manufacturing Amazon Web Services

Mariella Minutolo
Executive Vice President ETAS

Markus Rettstatt
Vice President Software Defined Car Mercedes-Benz Tech Innovation GmbH

Patrik Andersson
Tech Lead Volvo Cars

The Pulse



Concepts and solutions

1. In-vehicle zero-trust architecture
2. Managed post training (MPT)
3. Microcontroller parallel processing unit (PPU)
4. Mixed critical orchestrator
5. Developer portals for vehicle APIs
6. Integration of vECUs on virtual network
7. Multi-tenant compute in vehicles
8. Vehicle abstraction layer
9. Virtual electronic control units (vECUs)
10. Zonal architecture

Ecosystems and organizations

11. Automotive open source
12. Custom car OS without common industry standards
13. digital.auto
14. Infotainment-limited SDV strategy
15. The in-sourcing imperative
16. EU Data Act

Techniques and practices

17. AI-led modernization of in-vehicle legacy systems
18. Centralized vehicle software repository
19. GenAI in automotive software development lifecycle
20. Software-first ECU development
21. Automated continuous end-to-end testing
22. Continuous compliance
23. Cross-project VEW dashboard
24. Golden path and sensible defaults
25. Lift-and-shift from legacy software to SDV
26. Machine learning model deployment to the car
27. Shadow mode validation
28. Shift-left security for in-vehicle software
29. The automotive test pyramid
30. Vehicle data-driven engineering
31. Incremental software delivery

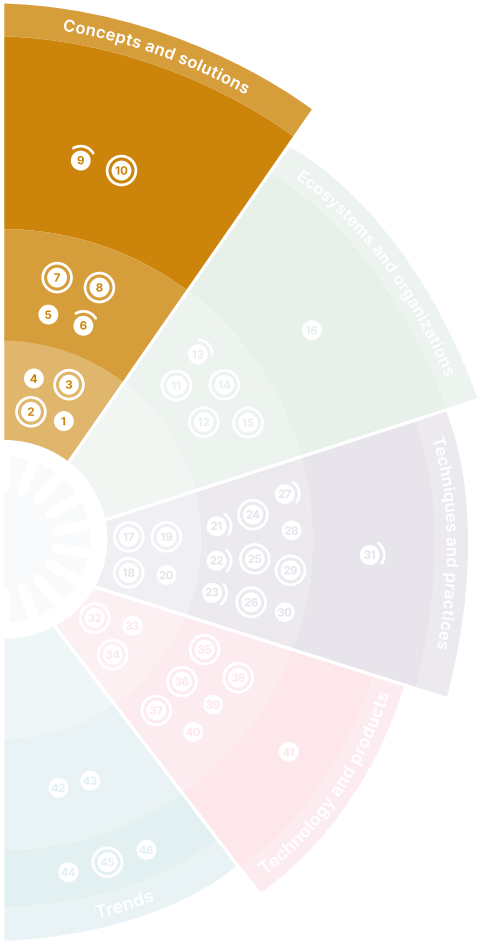
Technology and products

32. Agentic AI in cars
33. RISC-V
34. Wasm runtimes for embedded systems
35. Car-ready mobile apps
36. Google Android Build File System (ABFS)
37. Modern build systems for car software
38. NuttX in automotive
39. ROS 2
40. Rust
41. Android Automotive OS

Trends

42. Seamless development between partners
43. Virtual engineering workbench (VEW)
44. Beyond making SOP
45. Fragmented software stack & no reusability
46. Organization shift towards SDVs

Concepts and solutions



Concept phase

1. In-vehicle zero-trust architecture
2. Managed post training (MPT)
3. Microcontroller parallel processing unit (PPU)
4. Mixed critical orchestrator

Early adoption stage

5. Developer portals for vehicle APIs
6. Integration of vECUs on virtual network
7. Multi-tenant compute in vehicles
8. Vehicle abstraction layer

Mass adoption stage

9. Virtual electronicW control units (vECUs)
10. Zonal architecture

1. In-vehicle zero-trust architecture

Concept stage

The expanding attack surface of connected vehicles, coupled with increasingly stringent data protection regulations, demands a fundamental rethink of automotive cybersecurity. As threats evolve and compliance requirements tighten, conventional security based on isolation or measures on CAN/Ethernet bus systems is no longer sufficient.

In this context, the principles of zero-trust architecture (ZTA) are essential for SDVs. ZTA's core tenet — "never trust, always verify" — allows organizations to strengthen their defenses in the evolving threat landscape. By enforcing strict access controls, employing robust encryption and implementing continuous, real-time monitoring, ZTA ensures every entity accessing the vehicle's network or resources is rigorously authenticated and authorized. This containment strategy guarantees that a compromise in one component doesn't jeopardize the entire vehicle's operation or sensitive data.

We continue to observe that ZTA should be a foundational approach for creating resilient security in SDVs. It enables compliance with industry standards such as ISO/SAE 21434, which mandate strong security measures, including continuous verification and stringent access controls, to mitigate modern cyber threats and safeguard critical systems and user data.

2. Managed post training (MPT)

Concept stage

Specialized foundational models are emerging to address distinct automotive tasks, moving beyond general-purpose large language models (LLMs). For example, vision LLMs (VLLMs) excel at object recognition, while audio LLMs (ALLMs) optimize voice recognition, each offering unique advantages for specific in-vehicle functions.

As automotive organizations look to leverage these specialized models for efficiency and differentiation, a critical challenge arises: evaluating which fine-tuned model performs best for a company's specific domain task and proprietary data. Selecting the optimal model becomes a key strategic differentiator.

Managed post training (MPT) directly addresses this need. It provides a managed service that enables automotive companies to rapidly evaluate multiple fine-tuned foundational models against their unique tasks and data. This makes it faster and easier to identify the most effective model, streamlining deployment and maximizing the value of specialized AI in the vehicle. And, once the task fit of a foundational model is established in a short iteration, the setup can be used for AI and ML loops that deliver updated models into production.

3. Microcontroller parallel processing unit (PPU)

Concept stage

The microcontroller parallel processing unit (PPU) is emerging as a significant innovation for efficient edge AI deployment. This dedicated hardware is specifically designed to offload AI inference tasks traditionally handled by power-hungry GPUs or other accelerators.

The core value of the PPU is its ability to execute real-time inferencing locally sitting beside the microcontroller. By doing so, it dramatically reduces the power consumption and hardware costs associated with more expensive accelerator solutions, while maintaining the performance levels necessary for specialized AI tasks. This shift makes scalable, cost-effective AI at the edge a tangible reality.

This efficiency is particularly critical for automotive applications, where low-latency response and minimal energy usage are paramount for functions such as intelligent sensors and real-time inferences.

4. Mixed critical orchestrator

Concept stage

Passenger vehicles today can have over 100 ECUs, from microcontroller units (MCUs) powering basic functions like headlights to complex multiprocessor units (MPUs) running infotainment systems. To keep SDVs manageable, the number of ECUs must be controlled and, where possible, consolidated.

A key enabler of ECU consolidation is advanced silicon technologies such as chiplets. Chiplets have lowered chip design costs, accelerated time-to-market, improved production yields, and integrated substantial functionality and processing power onto single chip packages.

However, consolidation removes the natural isolation between ECU software components running independently on separate processors and communicating through networks such as CAN and Ethernet. Historically, each ECU software component was designed to meet specific requirements, such as safety standards. With consolidation, heterogeneous software stacks now share common multiprocessor systems.

Mixed criticality orchestrators (MCOs) can help OEMs enable ECU consolidation by ensuring there's no interference between tasks. For example, MCOs can allow safety-critical tasks to run with sufficient memory and CPU resources as if they have sole occupancy of the system. MCOs can also manage multiple execution environments, including vehicle systems, edge computing in mobile networks with low latency, and remote cloud resources.

Critically, MCOs require rigorous certification and validation to ensure they meet the safety and reliability requirements demanded by the automotive industry.

5. Developer portals for vehicle APIs

Early adoption stage

Multiple OEMs now offer portals to help developers build applications using vehicle APIs. The portals can include vehicle status information such as location, charging status or errors. Some offer the ability to send data and commands to vehicles, enabling use cases such as drive experience personalization, customized patterns for locking and unlocking, anti-kinetosis, charging pattern optimization or sleep detection.

These developer portals are key enablers for SDV ecosystem growth, providing the foundation for third-party applications that integrate seamlessly with vehicle systems. They support innovative use cases ranging from fleet management and smart home integration to personalized vehicle experiences and connected services. The portals also create opportunities for developers to build applications that use vehicle data and control capabilities while maintaining security and safety standards.

The developer experience and functionality vary significantly across OEM portals, reflecting the early stage of this technology's evolution. Some portals offer comprehensive APIs with extensive documentation and testing tools, while others provide more limited access with basic functionality. This variation suggests the industry is still defining best practices for vehicle API design and developer engagement.

The growing availability of vehicle APIs through developer portals will accelerate innovation in the automotive ecosystem by enabling a broader range of developers to contribute to vehicle functionality. This approach will also help OEMs differentiate their vehicles through unique applications and services while maintaining control over critical vehicle systems. As the automotive industry evolves, developer portals will become essential for building vibrant software ecosystems around vehicles.

6. Integration of vECUs on virtual network

Early adoption stage

The adoption of virtual ECUs (vECUs) for development continues to advance, enabling earlier integration and collaboration. OEMs are increasingly integrating multiple vECUs early in the software development lifecycle. Architectures now also interconnect supplier-owned vECU environments, allowing independent software deployment, faster iteration and earlier error detection when fixes are less costly.

However, OEMs' expectations for vECUs must take into account that, while some cases achieve binary compatibility with hardware, the significant investment required to create exact replicas of complex ECUs often yields diminishing returns. The highest value lies in using vECUs to uncover integration issues early, analyze system behavior and run parallel simulations of vehicle instances with different software variants to accelerate validation cycles.

While vECU technology shows clear improvement and enables valuable workflows, a fully synchronized, hardware-identical virtual network — particularly concerning CPU clock speed synchronization for complex multi-vECU testing — remains a challenge. Focusing on high-value use cases, rather than exhaustive replication, offers the most pragmatic way to maximize vECU benefits.

7. Multi-tenant compute in vehicles

Early adoption stage

Multi-tenant compute is a computing architecture where multiple, independent applications, services or virtual machines share the same physical hardware resources while maintaining strict isolation and security boundaries. In automotive contexts, this means running vehicle systems — such as infotainment, Advanced Driver Assistance Systems (ADAS), telematics and third-party applications — on shared computing platforms rather than dedicated hardware for each function.

The need for multi-tenant compute has emerged from the industry's shift toward new vehicle architectures. Traditional distributed ECU architectures are being replaced by zonal and central computing architectures that consolidate multiple functions onto shared platforms. This requires sophisticated resource management and isolation mechanisms to ensure safety-critical systems, infotainment applications and third-party services can coexist securely on the same hardware.

Multi-tenant operating systems like seL4 and L4Re provide the foundation for this architecture, offering formal verification of security properties and microkernel-based isolation. These systems enable OEMs to run applications from different suppliers, third-party services and internal systems on the same hardware with guaranteed security boundaries. They also provide resource management for efficient allocation of computing resources based on real-time demands.

8. Vehicle abstraction layer (VAL)

Early adoption stage

A vehicle abstraction layer (VAL) provides a higher-level interface to access a vehicle's functionality without having detailed knowledge about the underlying implementation, sensors and actors.

Unlike the traditional approach of pre-planned, signal-based communication between components, a VAL abstracts a vehicle's capabilities, separating hardware-bound software from features software. This allows more flexibility for additional features provided by application updates, supports the porting of features to different vehicle models, and is essential for external solution providers such as ADAS and AD, infotainment systems or connected mobility services.

Many OEMs include the VAL concept in their internal architecture, but there are also more public examples that include this concept, such as Android Automotive, Veecle.io and the Vehicle Information Service Specification (VISS).

While there are clear benefits to a VAL concept, it also comes with challenges, including difficulty supporting different service requirements. Some applications using the VAL need a response or execution done in real time, while others don't, and implementing this mixed criticality is complex.

Another challenge we observe in VAL design is 'leaky' abstractions, where API layers are implicitly coupled to hardware. This results in brittle architectures, sometimes with multiple layers of APIs, so even simple feature additions require widespread 'shotgun surgery' instead of isolated changes.

9. Virtual electronic control units (vECUs)

Mass adoption stage

Virtual electronic control units (vECUs) have become foundational for modern automotive software development, enabling the design, testing and validation of embedded systems without waiting for physical hardware. This approach enables teams to rapidly develop and test new features and validate complex functionalities such as ADAS, significantly reducing development time and cost.

vECUs reduce reliance on physical prototypes, enable the early detection of software problems and provide enhanced simulation capabilities for comprehensive system testing. They're now integrated seamlessly into enterprise-grade CI/CD pipelines and cloud-based virtual engineering workbenches.

The simulation fidelity of vECUs has also significantly improved, and they're now used deeper into the validation cycle, not just for early-stage testing. The explosion of data generated by vECU simulations is being tackled with unified logging, traceability and data lake architectures, enabling search, comparison and compliance reporting.

10. Zonal architecture

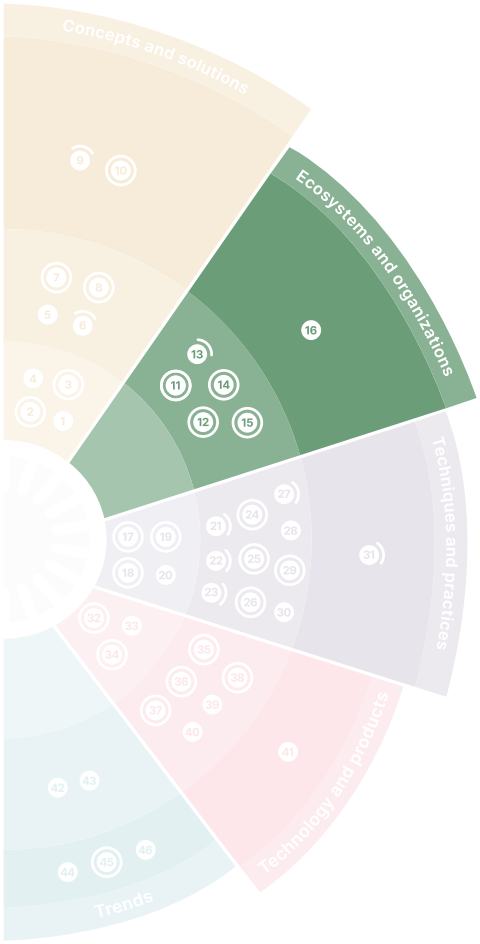
Mass adoption stage

In recent years, OEMs have moved from distributed to domain-centric to centralized architectures containing a single or few high-performance computers (HPCs). The next phase of this evolution of electrical/electronic (EE) architectures is a hybrid approach combining centralization with an additional layer focused on the physical distribution of components.

In this architecture, zonal control units (ZCUs) are strategically positioned throughout the vehicle, such as in the left, right, rear and cockpit zones. Within each designated zone, all local sensors, actuators and power consumers establish direct connections to their nearest ZCU. In turn, these ZCUs are connected to one or more central HPCs.

This approach should lead to a substantial reduction in physical wiring, lowering the cost and weight of vehicles. Further potential benefits or downsides rely heavily on the amount of logic put into the ZCUs. For example, I/O aggregation, filtering and localized processing within the ZCUs allows for further abstracted communication to the HPCs. This can separate feature logic from the underlying hardware, but it could also lead to familiar challenges inherent in developing and managing complex, distributed systems.

Ecosystems and organizations



Early adoption stage

11. Automotive open source
12. Custom car OS without common industry standards
13. digital.auto
14. Infotainment-limited SDV strategy
15. The in-sourcing imperative

Mass adoption stage

16. EU Data Act

11. Automotive open source

Early adoption stage

Historically, automotive open source involved differentiating and integrating generic community software such as Linux kernels. Automotive organizations rarely led projects targeting the industry's unique needs for functional safety, regulatory compliance and long-term maintainability.

Open source releases from OEMs and Tier1s such asHalo OS, RHIVOS, Corbo Linux and CTRL OS, which attempt to provide safety-compliant alternatives, are also significantly changing the landscape of solutions available to OEMs.

Apart from open source releases, a strategic pivot appears on the horizon, with OEMs and Tier 1s now collaborating within foundations through a governed framework (Eclipse S-CORE). New initiatives like this are engineered from inception for in-vehicle safety, security and hardware integration, marking a shift from passive consumption to automotive-driven open source creation that reflects a deeper ecosystem maturity. The Eclipse S-CORE project is showing early traction, attracting collaborations across the value chain with code contributions from various organizations.

We are truly excited about these trends, but we also see challenges for these projects to thrive:

- **A lack of focus:** There appears to be a desire to tackle many topics at the same time. While there's nothing wrong with having a long-term vision, there's a risk of getting stuck with too many loose ends. Practicing the Unix philosophy of "Do one thing and do it well" might help to create traction.

- **Build an active community:** Unlike widely adopted open source projects, automotive software has a niche user base of in-vehicle software developers. This makes it harder to cultivate an active, engaged community to support the project's success.
- **Creating 'automotive-grade' software:** Adhering to industry-specific standards is a hurdle for open source automotive projects, and they must establish an approach for creating software that's easily certifiable. Interesting approaches can be found from Ferrocene and the Trustable Software Framework promoted by Codethink.

Addressing these challenges will be essential for the success of any automotive open source project.

12. Custom car OS without common industry standards

Early adoption stage

Many automotive manufacturers are vertically integrating their development processes and creating everything in-house, including custom operating systems, without establishing common industry standards or collaborating with partners. This approach leads to fragmentation where each OEM develops its own proprietary OS, reinventing the wheel for basic car functions that could be standardized across the industry.

This custom OS approach forces core component manufacturers such as Infineon, Qualcomm and NVIDIA to develop generic hardware solutions instead of creating tailored, optimized solutions for automotive applications. When each OEM implements different operating systems and interfaces, semiconductor companies must design chips that can support multiple OS variants and communication protocols, leading to over-engineered, more expensive hardware that lacks optimization. This fragmentation increases costs throughout the supply chain while reducing the potential for specialized automotive features and performance optimizations.

Open-source approaches and industry collaboration — such as COVESA, Eclipse SDV, the AUTOSAR Rust working group, ASAM eV and RISCV international — offer better alternatives for standardization. By working together on common platforms and interfaces, organizations across the industry can reduce development costs, improve software quality and enable faster innovation.

13. digital.auto

Early adoption stage

digital.auto is a community-driven initiative that connects OEMs, suppliers, startups and developers to accelerate SDV innovation. It provides a digital-native, agile alternative for automotive software development, offering a cloud-based, use case-driven platform that aligns closely with modern developer expectations.

With a safe, open and fast environment to simulate, prototype and validate new mobility services and SDV features, developers and automotive organizations benefit from:

- **Interoperability:** Standards and open APIs to connect diverse components in the SDV stack.
- **Use-case-driven co-creation:** Rapid experimentation with real user validation, not just engineering assumptions.
- **A cloud-based playground:** A virtual space to model, prototype and test SDV use cases instantly, without waiting for hardware.
- **dreamKIT hardware:** Plug-and-play hardware for SDV prototyping, bridging the gap between cloud simulation and in-car testing.
- **Bring your own device:** The ability to add connected hardware using standard APIs for validation in real-world scenarios.

14. Infotainment-limited SDV strategy

Early adoption stage

Some automotive organizations focus their SDV strategy primarily on user experience, screens and app stores while neglecting the underlying real-time software infrastructure, testing capabilities, and overall product lifecycle management. This infotainment-limited approach constrains the full potential of SDV transformation.

When SDV strategy is limited to infotainment, organizations miss opportunities to use software to improve vehicle performance, safety systems and operational efficiency. They also fail to build the robust testing infrastructure and development processes needed for reliable vehicle software. This narrow focus can lead to fragmented development efforts and missed opportunities for comprehensive vehicle optimization across all systems.

A successful SDV strategy demands a comprehensive approach spanning the entire vehicle software ecosystem. OEMs must invest in real-time software capabilities for safety-critical systems, strong testing frameworks for vehicle-wide validation, and lifecycle management tools that support continuous software evolution. This holistic investment will enable OEMs to unlock the full potential of SDVs beyond infotainment features.

15. The in-sourcing imperative

Early adoption stage

Automotive manufacturers are increasingly taking control of software development. This change is directly linked to more centralized electrical/electronic (EE) architectures which consolidate functionality into a few high-performance computers (HPCs).

By owning the software of the core ECUs, OEMs have more control over the functionality where lower-level ECUs expose the capabilities and data of sensors and actors. The goal is to accelerate feature development, reduce the communication overhead of working with external suppliers, and support the creation of hardware-agnostic software.

The path to in-house software development is different for new and traditional car makers. Digital-native OEMs have the advantage of a greenfield approach, building their entire strategy around in-house development. In contrast, traditional OEMs are disrupting their longstanding relationships with tier one suppliers, causing friction. In addition, the strategy of some OEMs to concentrate software development in new organizational units has shown mixed results and led to significant course corrections.

16. EU Data Act

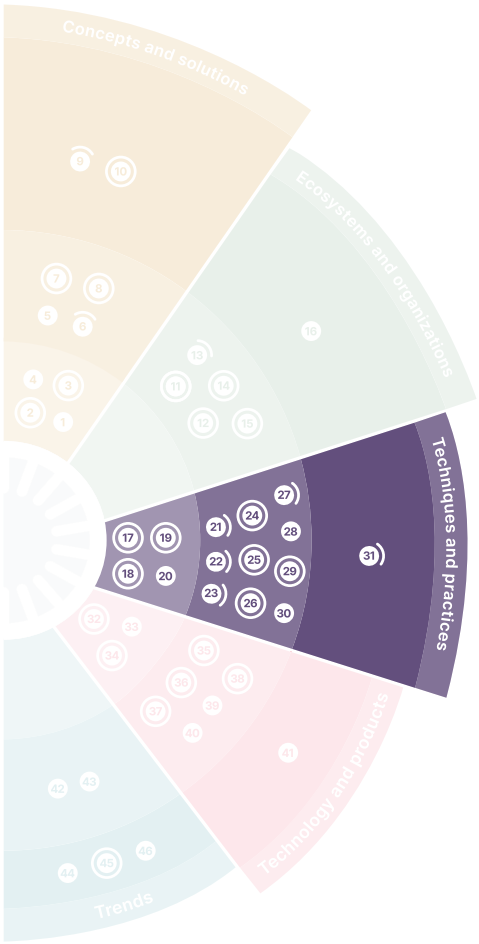
Mass adoption stage

The European Union Data Act presents challenges and opportunities for automotive organizations. The legislation is not just a compliance hurdle; it's a chance to unlock new value and foster innovation through easier and fairer access to data from SDVs — and a growing number of organizations are moving in this direction.

A key provision of the Act mandates easier switching between data processing services, such as cloud and edge providers. By defining standardized interfaces for their data, automotive organizations can ensure interoperability and reduce the complexity of switching, helping them avoid vendor lock-in and increase business agility.

We observe organizations adopting a 'compliance by design' approach to navigate this new regulatory landscape. Instead of retrofitting systems, automotive organizations are baking data governance and standardized interfaces into their processes from the outset. This proactive stance not only ensures compliance but also helps organizations monetize high-value data through new services in areas such as after-sales and insurance. Acting now is crucial, as waiting could mean missing out on the substantial economic benefits and competitive advantages the EU Data Act offers.

Techniques and practices



Concept stage

17. AI-led modernization of in-vehicle legacy systems
18. Centralized vehicle software repository
19. GenAI in automotive software development lifecycle
20. Software-first ECU development

Early adoption stage

21. Automated continuous end-to-end testing
22. Continuous compliance
23. Cross-project VEW dashboard
24. Golden path and sensible defaults
25. Lift-and-shift from legacy software to SDV
26. Machine learning model deployment to the car
27. Shadow mode validation
28. Shift-left security for in-vehicle software
29. The automotive test pyramid
30. Vehicle data-driven engineering

Mass adoption stage

31. Incremental software delivery

17. AI-led modernization of in-vehicle legacy systems

Concept stage

We observe a growing need to re-architect or rewrite in-vehicle software, driven by several factors. Changing electrical/electronic (EE) architectures demand changes to the distribution of functionality across ECUs. OEMs' in-sourcing strategies are pushing them to take greater control over components previously owned by suppliers. And new technology stacks are emerging, requiring rewrites of a significant proportion of vehicle software.

OEMs' traditional outsourcing strategies for development have created various legacy software scenarios. These range from white-box situations, where there's full access to the source code, to black-box situations, where components must be reverse-engineered based on the behavior of the software. Legacy software may also have inconsistent documentation, poor test automation coverage, or a large code base that has grown over time.

Although the technology is still in its early stages, the use of GenAI tools appears to hold promise for various use cases, including onboarding, post documentation, test generation, refactoring and even code porting. However, since these tools are not specifically designed for the automotive domain, OEMs should consider developing their own solutions based on open source models and tools.

18. Centralized vehicle software repository

Concept stage

A centralized vehicle software repository is a fundamental shift in how automotive manufacturers manage their software components and portfolios. Instead of maintaining separate codebases for different vehicle models and variants, OEMs are consolidating their software into a single virtual monorepo with the main/master branch serving as the primary development line.

This approach eliminates the complexity of managing multiple software stacks and enables faster feature delivery throughout the vehicle portfolio. Teams can develop features once and deploy them across multiple vehicle variants, significantly reducing development time and costs. The centralized repository model also improves code quality through enhanced reuse and reduces the overhead of maintaining multiple codebases.

Leading OEMs, including BMW, Mercedes-Benz and Volkswagen Group, are planning to adopt unified software platforms, taking a centralized approach to streamline development and accelerate time-to-market for new features. And OEMs such as Volvo Cars are implementing this strategy through a superset tech stack approach, consolidating software development across their vehicle portfolio. As more automotive manufacturers adopt this pattern, it will become a key enabler of software-defined vehicle transformation.

19. GenAI in automotive software development lifecycle

Concept stage

The automotive software development lifecycle (SDLC) presents unique scale and context challenges. It's characterized by significantly longer timelines, extensive upfront simulation and design phases, rigorous multi-stage verification and validation, and stringent safety standards. This generates a vast, interconnected context far exceeding typical software projects. Point-solution GenAI tools fail to capture this overarching context, hampering their ability to provide meaningful assistance.

The true value of GenAI can only be realized through its holistic integration throughout the SDLC, providing context spanning the entire V-cycle. Organizations that merely use GenAI as a coding assistant miss valuable opportunities to reduce friction, accelerate innovation and enable smarter workflows throughout the complex development process.

The key to success is holistic, yet pragmatic, GenAI integration, embedding capabilities across the SDLC to understand the extended development continuum and enable richer automation and insight. Specialized AI agents could emerge for every phase of the SDLC. Crucially, GenAI integration must meet developers where their needs are with an ecosystem of interoperable tools — avoiding monolithic 'one platform' solutions.

20. Software-first ECU development

Concept stage

The unprecedented complexity of SDVs demands a fundamental change from the 'hardware first, software later' development model to hardware and software co-development. Consolidating diverse, mixed-criticality applications onto domain or zonal controllers requires deep integration of hardware and software from the very beginning. Sequential approaches simply cannot achieve the necessary optimization and guarantee safety and security properties effectively.

Hardware and software co-design addresses this by enabling concurrent architecture trade-offs. It breaks down silos, allowing teams to simultaneously optimize silicon for specific workloads (such as

dedicated neural network accelerators), embed safety mechanisms directly into hardware (such as lockstep cores or hardware watchdogs), and build in security (such as hardware Roots of Trust). This intrinsic integration is far superior to bolting on solutions later, which is costly and often ineffective.

Collaborative innovation across the automotive value chain is essential to unlock the full potential of hardware and software co-design. Traditional siloed workflows between OEMs, tier ones, semiconductor vendors and software providers must evolve into integrated partnerships.

21. Automated continuous end-to-end testing

Early adoption stage

The automotive industry is witnessing a fundamental shift in software validation as organizations move toward automated, continuous end-to-end testing throughout the development lifecycle. This approach mirrors successful patterns from IoT and other phygital industries, where automated testing has become essential for managing complex, interconnected systems. While validation has always been a standout in the automotive industry, organizations are now adopting testing automation to ensure software quality across increasingly sophisticated vehicle architectures.

To achieve comprehensive testing coverage, organizations are using end-to-end development platforms and virtual engineering workbenches (VEWs) that integrate testing capabilities into mobile devices, vehicle back-ends and in-vehicle systems.

This approach goes hand-in-hand with the test pyramid methodology, where automation is implemented across multiple testing levels. At the base, continuous integration (CI) pipelines enable automated unit and integration testing in virtual environments. VEWs provide simulation-based testing, while automated boxcars and labcars enable hardware-in-the-loop testing for real-time validation. At the top of the pyramid, vehicle-in-loop systems combine virtual and physical testing to validate complete vehicle behavior.

The shift toward automated testing requires significant investment in testing infrastructure, simulation tools and CI pipelines. With vehicle software complexity increasing through multiple subsystems and interdependencies, automated, continuous end-to-end testing is becoming essential for maintaining quality and safety while enabling rapid development.

22. Continuous compliance

Early adoption stage

Continuous compliance is the process of continually assessing automotive software components to ensure they always meet regulatory requirements and industry standards. This approach integrates compliance checks into the development process rather than treating them as separate milestone activities.

In automotive software development, compliance audits — including functional safety, cybersecurity and ASPICE — play a crucial role in the safety assessment of vehicle systems before homologation. We are now seeing the emergence of knowledge graphs that capture entire development processes, enabling scalable safety arguments from individual components to complete code bases. Augmented by GenAI, frameworks such as the Trustable Software Framework from Codethink are unlocking innovative approaches to compliance by automating evidence synthesis and transforming audit workflows. Continuous compliance tools such as Kosli and ChainLoop complement this evolution, providing real-time monitoring of software changes against regulatory requirements. These tools also ensure traceability across the development lifecycle, allowing teams to identify and resolve compliance gaps proactively.

The continuous approach reduces the risk of compliance failures during certification and enables faster adaptation to evolving regulatory requirements. It also supports over-the-air updates by ensuring that software changes maintain compliance with safety standards. As vehicles become more software-defined, continuous compliance will be essential for maintaining safety while enabling rapid software evolution.

23. Cross-project VEW dashboard

Early adoption stage

SDV OEMs need deep visibility into software development processes and their external dependencies to monitor progress, manage risks and avoid production delays. These constant assessments apply to individual software artifacts and systems — and to entire vehicle software projects.

The virtual engineering workbench (VEW) integrates engineering, testing and virtual validation tools, allowing users to trace successfully executed tests and validations back to their requirements. As this process can be fully automated, the VEW dashboard will always reflect the current state of development progress, providing OEMs with real-time visibility.

VEWs also offer integration with other VEW environments. By securely connecting these systems, OEMs can gain insights into the development progress of dependent software artifacts spanning multiple suppliers and accounts, giving them a comprehensive view of the end-to-end software development lifecycle.

This approach enables more efficient resource allocation and better coordination of complex automotive software development programs. However, the shift toward cross-project visibility requires the integration of various development tools and the standardization of metrics and reporting. Organizations need to balance transparency with security and ensure teams can focus on their work while maintaining awareness of broader project context.

24. Golden path and sensible defaults

Early adoption stage

Historically, the way of working for automotive embedded software has been defined by the OEM's 'process, methods and tools' (PMT). The goal was to fix the software development lifecycle to ensure quality and compliance. However, the strict PMT approach is now often seen as a burden by developers. The growing complexity of in-vehicle software and the adoption of new tools and technologies means developers need greater flexibility; the established way of working can seem inefficient or even counter-productive.

Companies like Netflix, Spotify and Etsy have established a 'Golden Path' (sometimes referred to as a 'Paved Road') approach. This provides a recommended and supported way to complete essential tasks in building, testing and deploying software, but allows teams to deviate from the described path as long as they fulfill defined outcomes.

'Sensible defaults' offer a similar approach, giving teams a defined set of practices and methods that are considered to be standard. They should be the starting point for every project and only changed in special circumstances. For software development, this might include test-driven development, pairing and build automation. Companies might also define sensible defaults for data, security or AI. Every employee should be proficient in their use and able to apply them in their role.

We currently see automotive organizations attempting to apply these approaches for their software development. While many have met challenges in finding the right categories and achieving alignment, we consider these approaches to offer powerful ways to foster discussions and define the target way of working.

25. Lift-and-shift from legacy software to SDV

Early adoption stage

The automotive industry's electrical/electronic (E/E) architecture and systems are rapidly evolving toward having a powerful, centralized compute system for most core car functions. Many OEMs adopting new E/E architectures are also modernizing their software.

However, in many cases automotive manufacturers moving functionality from legacy software to modern architecture are simply translating existing software to the new architecture.

This lift-and-shift approach fails to leverage the fundamental benefits of SDV. Instead of carefully planning features for the new architecture, OEMs are porting their static, ECU-centric software designs to work on the new E/E architecture, missing opportunities for improved scalability, performance and maintainability. To take full advantage of the capabilities of modern SDV architectures and compute systems, OEMs must properly redesign their software components.

26. Machine learning model deployment to the car

Early adoption stage

Machine learning (ML) models are fundamental to modern vehicles, powering Advanced Driver-Assistance Systems (ADAS) and enabling features like lane-keeping assist and object detection by processing real-time sensor data with ML models to enhance safety.

Self-driving and ADAS applications in cars require rapid decision-making. Therefore, processing information directly within the vehicle is preferable to transmitting large volumes of data to the cloud and processing for information there. While the cloud can assist with highly complex tasks, cars can pre-process data to reduce information transfer. However, integrating increasingly sophisticated AI models into compact, power-efficient in-car computers presents a significant challenge.

AI is also transforming automotive interactions through voice commands and personalized entertainment system recommendations. To enable continuous improvements and new features, frequent vehicle updates are necessary. This necessitates proficiency in wirelessly deploying AI models to cars. This capability will foster the development of the next generation of safer, smarter and continuously evolving vehicles. Automotive manufacturers should also consider managing all ML models within a vehicle fleet to ensure optimal performance and feature delivery throughout the car's lifespan.

27. Shadow mode validation

Early adoption stage

Shadow mode refers to running two versions of a software component in parallel. One is live and impacts other components with its results; the other runs simultaneously but its results are logged for comparison and analysis and don't affect other components. This allows teams to gather software performance data and compare it with current production versions.

For example, in adaptive cruise control (ACC), a vehicle sensor detects the distance to the vehicle in front frequently, with an ECU continuously processing the data and generating output. When a new ACC variant is deployed into the vehicle in shadow mode, teams can compare how the new version performs against the current one. When enough data is collected from the field to prove the new ACC variant outperforms the current one, the new variant can be deployed to the entire fleet.

For features such as ADAS, it's impossible to develop test cases for every imaginable scenario. Running vehicle functions in shadow mode provides a safe way to test new vehicle functions at scale, based on their behavior under real-world conditions.

With shadow mode capabilities in on-board architectures, organizations can also safely train new functions in the field while processing data in the vehicle and comparing results with expected outcomes. This technique supports the drive toward increased software quality in the automotive industry.

28. Shift-left security for in-vehicle software

Early adoption stage

Shift-left security is a set of practices for addressing security questions early in the software development process, rather than dealing with them at the end. As vehicles become increasingly software-driven they also become more attractive targets for cyber attackers. For example, in 2024, the Chaos Computer Club showed how it retrieved the movement information of 800,000 vehicles, including user details. This growing threat landscape requires a fundamental change in how OEMs approach software security.

Development teams are evolving their practices to embed security into CI/CD pipelines. They're also conducting automated security testing to identify vulnerabilities in code and dependencies, and carrying out constant threat model assessments for their applications.

Teams are increasingly using AI and ML models to perform real-time code analysis, detect anomalies and assist with automated threat modeling. These tools are being integrated into IDEs and CI/CD pipelines, reducing manual overhead and enabling proactive vulnerability detection before code even hits test environments.

There's also a greater focus on end-to-end trust, including mutual attestation between in-vehicle ECUs and cloud services, and secure OTA workflows that preserve cryptographic integrity and traceability across domains.

29. The automotive test pyramid

Early adoption stage

Testing in automotive software development involves various environments, from local unit tests on virtual environments to more complex physical hardware setups. Environments early in the testing process are typically faster, but can't reliably detect certain types of errors.

As automotive software development moves to a more iterative approach, testing must 'shift left', catching issues sooner and reducing development time and costs. However, it's true that different levels of physical test environments are still needed.

The test pyramid is a common metaphor in general software development. It categorizes tests by their scope and granularity, suggesting a pyramid-shaped ratio between the test types. While commonly used test categories must be changed to reflect the challenges and requirements of embedded automotive development, the concept of a test pyramid remains valid.

Simply aiming to shift left does not define a desired target state, making it difficult to measure progress. By defining their own test pyramid and making it an essential part of testing strategy, automotive organizations can determine testing goals more clearly to make their initiatives more efficient.

30. Vehicle data-driven engineering

Early adoption stage

Traditionally, vehicle data has been instrumental during product development phases, primarily for debugging car components and fine-tuning them to meet quality, security and user experience standards. However, with the advent of SDVs, there's been a significant increase in the volume of data vehicles generate.

One of OEMs' biggest challenges is integrating this data into product engineering so engineers can use real-world insights to refine components more efficiently and deliver enhanced vehicle iterations.

Another challenge is managing data at scale. Selecting which data to extract and analyze from the vehicle is a complex process, as is building the tooling to translate it into actionable insight.

Vehicle data-driven engineering enables more informed decision-making throughout the development process. Teams can use data to validate design assumptions, prioritize development efforts and measure the impact of software changes. This approach also supports predictive maintenance and helps identify potential issues before they affect vehicle performance or safety. If organizations can use data effectively, they'll engineer better vehicles and better vehicle experiences.

31. Incremental software delivery

Mass adoption stage

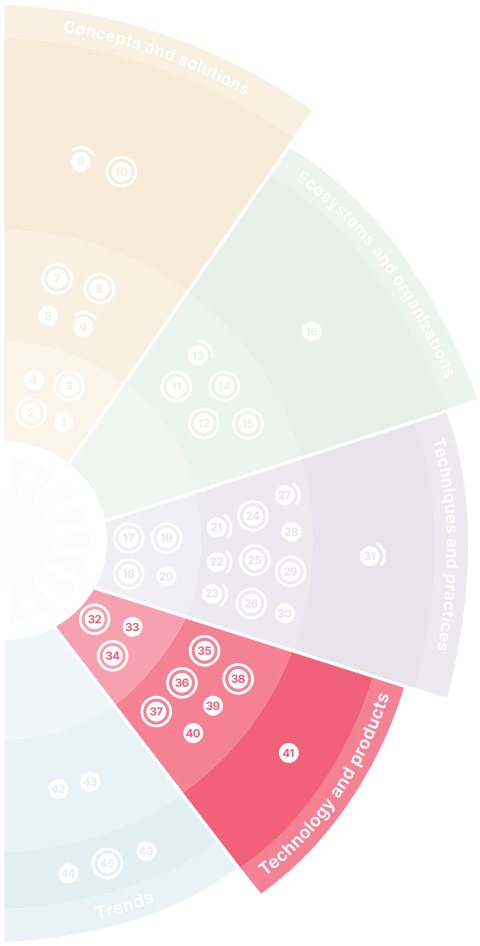
Over-the-air (OTA) software updates can be continuously delivered directly to SDVs, allowing OEMs to deploy or activate new features, enhancements and fixes rapidly and frequently, without the need for physical access to the vehicle.

This transforms the car into a dynamic platform that evolves to deliver ongoing optimization, new services and enhanced user experiences, mirroring the continuous delivery model prevalent across the software industry. OTA software updates can also reduce the need for recalls and service center visits for software-related issues, leading to a more convenient and cost-effective ownership experience for customers.

The OTA approach requires teams to develop software in smaller, manageable increments that can be tested and deployed independently. Incremental software delivery supports feature updates by enabling teams to add new capabilities without disrupting existing functionality. It also improves development efficiency by allowing teams to get feedback on features earlier and make adjustments based on real-world usage.

However, the shift toward incremental delivery requires changes in how teams structure their work, plan releases and coordinate across different vehicle systems. Organizations need to establish processes for managing dependencies, testing incremental changes and ensuring updates maintain vehicle safety and performance. With the right processes in place, incremental delivery enables OEMs to respond faster to customer needs and market changes, while maintaining software quality and reliability.

Technology and products



Concept stage

32. Agentic AI in cars
33. RISC-V
34. Wasm runtimes for embedded systems

Early adoption stage

35. Car-ready mobile apps
36. Google Android Build File System (ABFS)
37. Modern build systems for car software
38. NuttX in automotive
39. ROS 2
40. Rust

Mass adoption stage

41. Android Automotive OS

32. Agentic AI in cars

Concept stage

An early trend of agentic AI is emerging in the automotive sector, shifting in-car intelligence from being simply reactive to proactively helpful. Unlike a standard voice assistant that only responds to commands, an AI agent understands context, anticipates needs and independently executes tasks to achieve a goal. It can act as a nascent digital copilot, but this capability is in early product states.

This trend is already visible in the most advanced in-car experiences. Some new vehicle assistants learn a driver's habits to make proactive suggestions, such as activating the heated steering wheel on a cold morning or recommending a less congested route to a destination in the driver's calendar. By analyzing patterns, the AI takes initiative, adjusting climate, media and navigation to create a seamless experience without needing explicit instruction. Several OEMs have announced efforts to integrate large language models (LLMs) directly into their cars' computers to make these interactions even more fluid and powerful.

Modern Advanced Driver-Assistance Systems (ADAS) can be seen as a foundational form for future use of agentic AI. Features like adaptive cruise control with lane-centering have a clear goal: maintain a safe distance and position within the lane. The system autonomously uses steering, braking and acceleration to achieve this objective.

As these systems evolve to handle more complex scenarios, such as navigating urban traffic, they'll demonstrate increasingly sophisticated agent-like behavior, paving the way for a truly autonomous future. However, the consistency in quality of results for the driver experience remains a key challenge.

33. RISC-V

Concept stage

RISC-V, the open-standard instruction set architecture (ISA), continues to gain significant traction as a foundation for automotive computing. Its royalty-free, open-source nature and highly modular design uniquely enable the development of specialized, automotive-optimized processors. RISC-V also fosters crucial hardware-software co-design — a key advantage over proprietary ISAs.

The momentum behind RISC-V adoption is growing, with major semiconductor companies announcing automotive-qualified RISC-V product lines. Development boards are also becoming available for experimentation, accelerating ecosystem maturity.

These developments signal RISC-V's move beyond promise into practical deployment. As software is ported to this open ISA, we anticipate a fundamental disruption to the traditional ECU supply chain and unit economics, offering OEMs greater flexibility and potential cost savings. RISC-V's architectural neutrality and collaborative model position it to drive a new wave of innovation, offering extensible freedom for the next generation of SDV compute platforms.

34. Wasm runtimes for embedded systems

Concept stage

In centralized automotive computing, virtualization is vital for isolating mixed-criticality apps across security domains, establishing it as a foundational component of the SDV stack. While hypervisors dominate virtualization today, emerging runtimes signal an ongoing evolution in this layer.

WebAssembly (Wasm) is evolving beyond its web origins into a compelling runtime for embedded systems, particularly for high-level and cloud-connected applications. Its core advantage remains 'compile once, run anywhere' portability. This allows developers to leverage familiar web APIs and toolchains for initial development and testing before deploying apps onto target vehicle hardware.

Critically, Wasm provides robust sandboxing, making it an excellent instrument for enforcing strict, limited APIs and enhancing security isolation. That makes it ideal for embedded use cases in areas such as infotainment, telematics and connectivity features. It also has the potential to offer lightweight 'containers' for third-party software.

35. Car-ready mobile apps

Early adoption stage

Car-ready mobile apps can seamlessly leverage automotive infotainment systems and provide enhanced user experiences inside vehicles.

The barrier for getting mobile applications into vehicles is becoming lower as OEMs open up their vehicle APIs and software interfaces to third-party developers. In addition, Android Auto and Apple CarPlay are constantly improving compatibility modes, allowing developers to adapt existing mobile apps for automotive environments with minimal additional development work.

As vehicles become more connected and equipped with feature-rich infotainment systems, car-ready mobile apps will use multiple screens, vehicle sensors and contextual information to create immersive user experiences. Future cars will offer even more sophisticated platforms, enabling mobile apps to provide personalized navigation, entertainment, productivity and lifestyle services tailored to the in-vehicle environment.

36. Google Android Build File System (ABFS)

Early adoption stage

Android Build File System (ABFS) is emerging as a significant solution in the Android ecosystem, and is particularly relevant for complex automotive software projects. It provides a virtual file system layer specifically engineered to accelerate Android builds by optimizing file access and caching mechanisms.

ABFS directly targets build time reduction, a critical challenge as projects scale. By minimizing I/O overhead, it dramatically speeds up incremental builds — the frequent, smaller compilations developers perform during coding. It achieves this efficiency through intelligent caching and avoiding traditional file system bottlenecks.

Adopting ABFS improves developer productivity by streamlining workflows and reducing delays. For SDV development that relies on large Android code bases, ABFS is becoming a key enabler for maintaining quick, efficient build pipelines to support the faster iteration cycles essential in modern automotive development. Its growing traction demonstrates the automotive industry's focus on optimizing foundational developer tools, and although it's being implemented for Android builds, the fundamental principles appear to be extendable to any ECU.

37. Modern build systems for car software

Early adoption stage

Modern build systems are becoming essential for automotive software development as projects grow in complexity and scale across diverse application types. These build tools offer faster, more reliable compilation processes that can handle the large, centralized software repositories typical of SDVs while supporting multiple target platforms and architectures.

Build systems such as Bazel and Project Horizon provide powerful caching, incremental builds, parallel execution and reproducible builds that significantly reduce development time. They also integrate seamlessly with various CI/CD pipelines, enabling automated testing and deployment workflows that span from low-level vehicle control systems to high-level user interface applications.

The versatility of modern build systems allows automotive teams to manage complex dependencies and build processes for different vehicle subsystems simultaneously. Whether building real-time operating systems for vehicle control units, embedded software for sensor networks, or sophisticated infotainment applications, these tools provide the scalability and reliability needed. This unified approach to building diverse automotive software components streamlines development workflows and ensures consistency across the entire vehicle software stack.

38. NuttX in automotive

Early adoption stage

NuttX is an open-source real-time operating system (RTOS) gaining attention in the automotive industry as an alternative to proprietary RTOS solutions. It provides a POSIX-compliant environment that makes it easier for developers to port existing software and use familiar development tools.

The RTOS offers the deterministic, real-time performance required for safety-critical automotive applications while offering the flexibility of open-source software. NuttX can run on various microcontrollers and supports multiple architectures commonly used in vehicle systems. Its modular design allows OEMs to include only the components they need, reducing memory footprint and complexity.

Several OEMs are exploring NuttX for specific use cases. Li Auto's Halo OS leverages NuttX for in-vehicle systems, while Xiaomi's Vela platform uses NuttX for intelligent cabin devices and sensor fusion in its SU7 electric car. Sony's Aitrios edge devices employ NuttX for driver monitoring systems and Advanced Driver Assistance System (ADAS) edge sensors.

As the automotive industry moves toward more software-defined architectures, having reliable, open-source RTOS options will be vital for reducing dependency on single vendors and enabling faster innovation. NuttX represents a shift toward more transparent and customizable operating system solutions for embedded automotive applications, particularly in body control modules, sensor nodes and safety islands.

39. ROS 2

Early adoption stage

ROS 2 is an open source middleware framework originally designed for the development of robotic systems. Its adoption is accelerating throughout the automotive sector, moving beyond research into pilot programs and production-intent development for passenger cars and heavy vehicles. Automotive organizations are using its libraries, tools and modular middleware to manage the growing complexity of SDV architectures, particularly for advanced autonomy and ADAS features.

This momentum is bolstered by the traction of open source solutions such as Autoware.Auto and Automate.Universe, which are built natively on ROS 2. The increasing use of Autoware for automated driving development and vehicle experimentation provides a proven use case, accelerating ecosystem maturity and demonstrating ROS 2's viability for critical, real-world automotive applications.

The automotive industry's growing commitment to open source frameworks aims to foster collaboration, reduce development silos and accelerate innovation cycles. This strategic shift enhances ROS 2's position as a foundational enabler of scalable SDV development.

40. Rust

Early adoption stage

Adoption of the Rust programming language in the automotive industry is accelerating, with OEMs establishing dedicated Rust teams, signaling serious commitment. In addition, AUTOSAR has formed working groups to formally recognize Rust as a viable language, paving the way for standardization. The ecosystem is maturing rapidly, with Rust-based open-source projects emerging in foundations such as Eclipse.

This momentum is fueled by active cross-industry collaboration. OEMs and tier ones are jointly developing Rust-based prototypes for critical components, while hardware suppliers collaborate to create Rust support for drivers and lower-layer software. Rust's core strengths — guaranteed memory safety without garbage collection, performance parity with C++ and compile-time enforcement of safety rules — make it a compelling alternative for embedded systems. Crucially, the Ferrocene project has achieved ISO 26262 (ASIL D) and IEC 61508 (SIL 4) qualification for the Rust compiler, enabling safety-critical deployment.

The alignment of organizational investment, standardization efforts, open source growth and proven safety certification gives Rust a distinct advantage for large-scale adoption in automotive software development. It directly addresses critical industry imperatives for safety, security and modern development practices.

41. Android Automotive OS

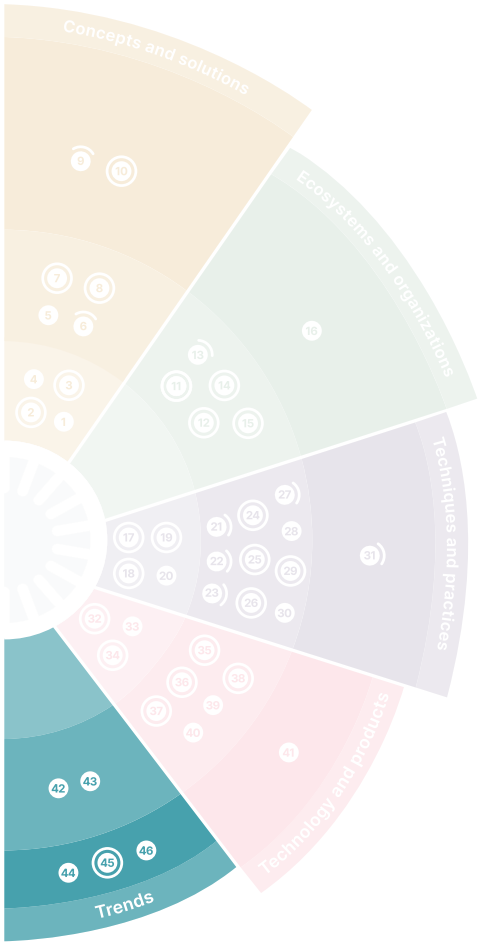
Mass adoption stage

Android Automotive (AAOS) is a variant of the Android operating system that serves as a foundation for in-vehicle infotainment systems. It's not to be confused with Android Auto, a mobile application that mirrors smartphone features onto car displays. AAOS can be used alongside Google's Automotive Services (GAS) to provide in-vehicle access to maps, an app store and Google Assistant. Since its launch in 2017, AAOS has seen relatively rapid success in the market, and today is broadly adopted by OEMs seeking proven infotainment solutions. Some OEMs use AAOS, while others build their own version based on the Android Open Source Project (AOSP) to gain greater control and customization over the user experience.

AAOS provides a cost-effective way for OEMs to build a rich infotainment service in their SDVs. It offers a mature development ecosystem, a large set of fundamental functionalities, familiar and intuitive user experiences and the ability to integrate existing third-party applications.

With rising customer expectations for highly integrated and personalized in-vehicle AI assistance, OEMs have to make a significant decision: to go all-in on the Google ecosystem with Gemini AI or build their own AI assistance. The first approach would result in a lock-in to the ecosystem and limit their ability to differentiate their user experience, while the second approach would likely require significant investment to keep up with the pace of AI innovation.

Trends



Early adoption stage

42. Seamless development between partners
43. Virtual engineering workbench (VEW)

Mass adoption stage

44. Beyond making SOP
45. Fragmented software stack and no reusability
46. Organization shift towards SDVs

42. Seamless development between partners

Early adoption stage

The development of vehicle software involves numerous organizations. Often, multiple teams or even companies contribute to the same stack running on an ECU. This collaboration, while essential, often becomes inefficient due to disparate systems and heterogeneous processes.

To address these issues, there's a critical need for close collaboration, more frequent integration and cross-partner testing. By adopting common artifact stores, organizations can ensure all teams access and contribute documentation and other essential materials to the same central repository, facilitating consistency and reducing redundancy.

Shared development pipelines could standardize processes across companies, streamlining development and testing activities. In addition, virtual testing environments shared among stakeholders expedite validation and integration, and improve the overall quality and reliability of software. Embracing these tools and methodologies can transform inefficient supplier relationships into highly productive partnerships.

43. Virtual engineering workbench (VEW)

Early adoption stage

A virtual engineering workbench (VEW) is a cloud-based software solution that integrates various tools and technologies to facilitate the development, testing and validation of vehicle functions in a virtual environment. A VEW consists of three main components: function-tailored development environments, CI/CD pipelines for software builds and tests and virtual target environments for software validation.

VEWs trigger CI/CD pipeline executions for every change made to software. Newly produced software artifacts are stored in a central repository and validated in virtual target environments. The VEW self-service portal provides access to these components and functions based on user roles and permissions.

Natural language interfaces now enable developers and testers to auto-generate code snippets, test cases and even entire simulation scenarios in VEWs. AI copilots assist with debugging, refactoring and impact analysis, reducing the time spent on repetitive or low-level engineering tasks.

44. Beyond making SOP

Mass adoption stage

Before the adoption of over-the-air (OTA) updates, automotive software development programs had one business-critical goal: to make the start of production (SOP). Features were designed upfront and deadlines were set years ahead in alignment with hardware development. As long as the software was ready when the vehicle was meant to go into series production, the goal was achieved.

With the huge increase in the complexity of in-vehicle software, the development process has become a more significant element and cost factor in overall vehicle development. OTA updates have become a standard feature, setting customer expectations that new features and fixes will be available after purchasing a vehicle.

In the SDV era, software engineering has a major impact on the quality and success of a vehicle. Increasingly, OEMs' focus is shifting away from just making SOP to investing in engineering effectiveness and continuously improving development processes. This new mindset, which makes in-vehicle software development more iterative and incremental, is now crucial for OEMs' success.

45. Fragmented software stack

Mass adoption stage

One of the inherent challenges of in-vehicle software development is the vast number of variants the software must support. Accommodating different electrical/electronic (EE) architectures, hardware generations, model lines and optional packages means the software must run on numerous distinct targets.

The established practices to deal with this challenge come from a time when software was significantly less complex. Software components were developed as individual projects, usually tied to specific hardware, leading to a highly fragmented software stack. While this approach is typically quicker for developing a feature or fixing an issue for a single line, it's significantly more effort to do the same for all models and variants.

Today's in-vehicle software is too large and complex for this traditional way of working. Time and costs for developing new features and supporting new hardware are rising; to remain competitive, OEMS must find a new approach.

Various approaches exist to handle this challenge more effectively, including extracting common components, creating a 'car OS', or having a single superset software package handle all setups. Some of these are described in more detail in this publication.

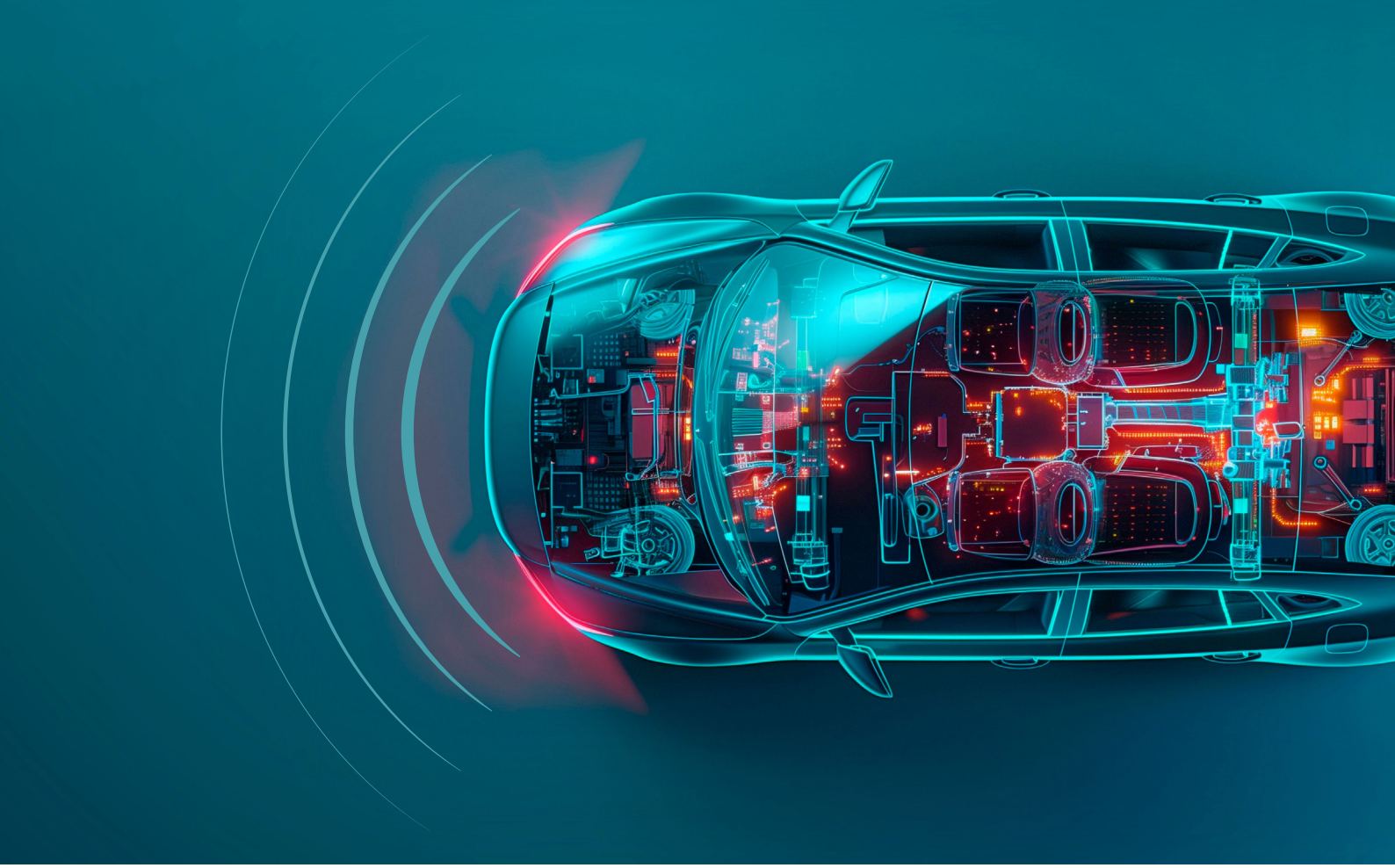
46. Organization shift towards SDVs

Mass adoption stage

Many of the obstacles to OEMs implementing SDVs are organizational rather than technical. For decades, structures, processes and mindsets have been aligned to a project-based model-focused way of working. To succeed in the SDV era, organizations must adapt to handle:

- **Iterative and incremental development:** Most development teams are still siloed based on activities in the V-Model, leading to numerous handovers and long cycle times. To iterate and innovate faster, a more cross-functional model is needed.
- **The Inverse Conway Maneuver:** With in-vehicle architecture and communication flows changing, a reasonable approach is to reflect them in the organizational setup to minimise the effects of Conway's Law.
- **Products over projects:** Internal tools and services as well as software components used across models should be treated as evolving products rather than temporary projects. Product teams should be able to handle not only development, but also operations and support.

Every organization has to accept that all structures come with downsides that must be deliberately mitigated. Fostering direct communication and collaboration across organizational structures appears to be essential for this.



Driving clarity in a software-defined world

As the automotive industry embraces the shift toward software-defined mobility, clarity amid complexity becomes more important than ever. SDV Pulse reflects our collective learning, exploration and evolving perspective on what truly matters in this transformation.

Whether you're leading strategy, building platforms or writing code, we hope this publication helps you pause, reflect and decide. The road to SDV success is neither a sprint nor a straight path; it's a layered journey of discovery, trade-offs and continuous adaptation.

We thank you for being part of this community and invite you to share your feedback, challenge our assumptions and contribute your voice to future editions. Together, we can shape the next chapter of software-defined mobility — with purpose, openness and momentum.

To get in touch with Thoughtworks please contact us here.

Stay up to date with all SDV-related news and insights

Subscribe to the SDV Pulse to receive emails every other month for tech insights from Thoughtworks and future SDV Pulse releases.

Subscribe now

We are a global technology consultancy that delivers extraordinary impact by blending design, engineering and AI expertise.

For over 30 years, our culture of innovation and technology excellence has helped clients strengthen their enterprise systems, scale with agility and create seamless digital experiences.

We're dedicated to solving our clients' most critical challenges, combining AI and human ingenuity to turn their ambitious ideas into reality.

/thoughtworks

Design. Engineering. AI.