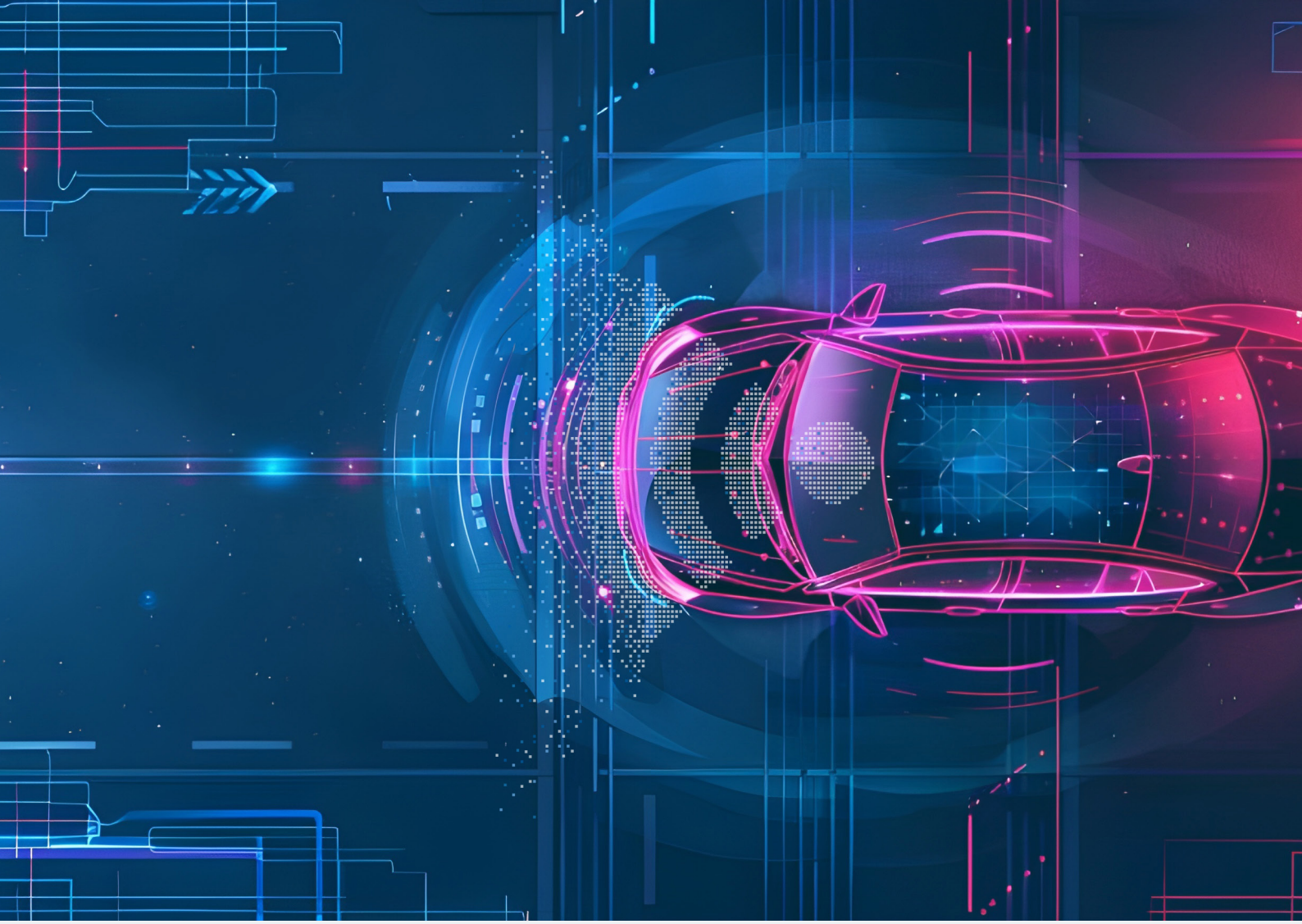
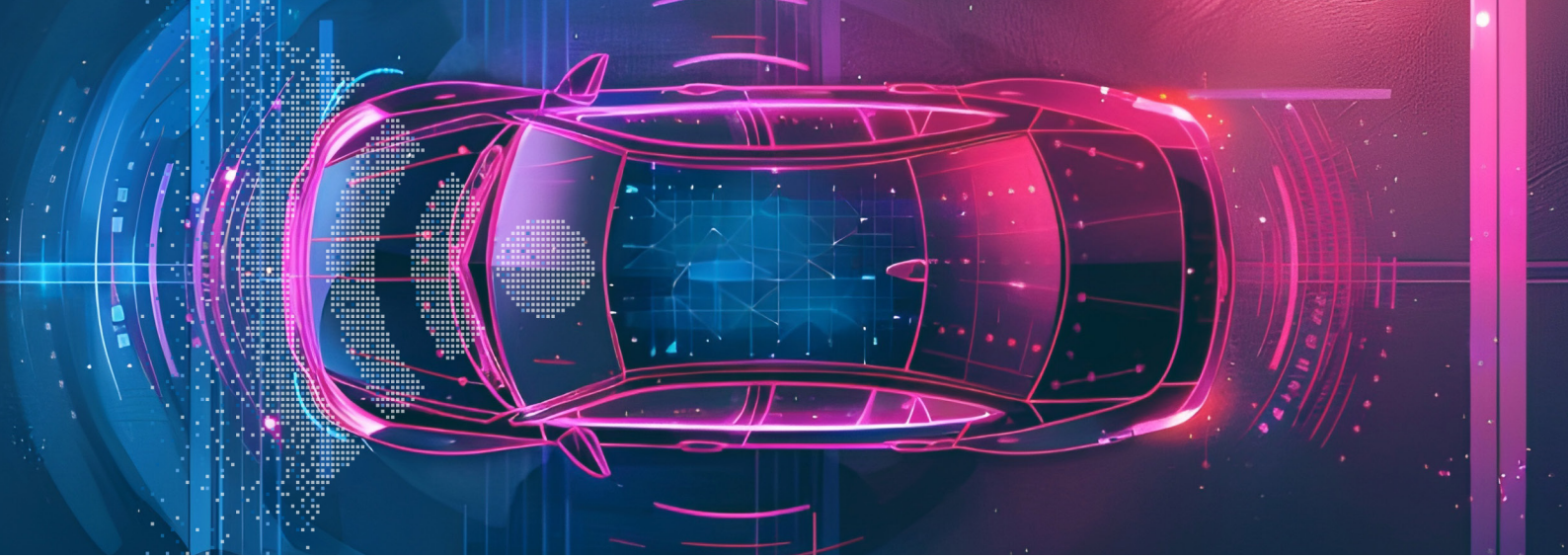


SDV Pulse

The technology driving the future

thoughtworks | aws

2024



Foreword

The automotive industry is entering a new era. The convergence of automotive engineering and digital technology has given rise to a new paradigm of mobility, where software plays a central role in defining the capabilities, performance and user experience of vehicles. This is the age of the software-defined vehicle (SDV).

If you want to stay ahead of the curve in this dynamic and rapidly evolving ecosystem, you need a deep understanding of the trends, challenges and opportunities at the intersection of automotive and software engineering.

This SDV Pulse report will give you a comprehensive overview of 40 of the latest developments, emerging technologies and best practices driving the evolution of software-defined vehicles. At the heart of this transformation lies an emerging spirit of collaboration and partnership, with OEMs, automotive companies, technology firms and ecosystem players coming together to co-create the vehicles of tomorrow. Even this report was born from a partnership — between Thoughtworks and Amazon Web Services (AWS).

By working together, amplifying one another's capabilities, fostering an open dialogue, and embracing innovation, we can unlock new opportunities, overcome challenges, and pave the way for a more sustainable, efficient and inclusive mobility ecosystem.

We extend our heartfelt gratitude to all the contributors who have generously shared their insights and expertise to make this publication possible. We sincerely hope that the SDV Pulse serves as a clear signpost for all who seek to navigate the ever-changing SDV landscape and shape the future of mobility.

Rachel Laycock
Chief Technology Officer
Thoughtworks

Andrea Ketzer
Director Technology Strategy
Automotive Manufacturing AWS

About Thoughtworks

Thoughtworks is a leading global technology consultancy that integrates strategy, design and software engineering to enable enterprises and technology disruptors across the globe to thrive as modern digital businesses. We have a reputation as thought leaders at the forefront of innovation, shaping and defining digital trends.

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

About AWS

Since 2006, Amazon Web Services has been the world's most comprehensive and broadly adopted cloud. AWS has been continually expanding its services to support virtually any workload, and it now has more than 240 fully featured services for compute, storage, databases, networking, analytics, machine learning and artificial intelligence (AI), Internet of Things (IoT), mobile, security, hybrid, media, and application development, deployment, and management from 105 Availability Zones within 33 geographic regions, with announced plans for 21 more Availability Zones and seven more AWS Regions in Malaysia, Mexico, New Zealand, the Kingdom of Saudi Arabia, Taiwan, Thailand, and the AWS European Sovereign Cloud. Millions of customers—including the fastest-growing startups, largest enterprises, and leading government agencies—trust AWS to power their infrastructure, become more agile, and lower costs.

About SDV Pulse

Today, software engineering is automotive engineering. Manufacturers that have led the automotive industry for decades are being challenged to fundamentally evolve and become expert software engineers in addition to delivering continuous mechanical and vehicle design innovation.

To help, Thoughtworks and Amazon Web Services (AWS) have created SDV Pulse. In this report, we've compiled expert perspectives from across our organizations on a broad range of technologies, practices and other influential trends to help you understand which ones are most relevant to your SDV strategy, and how to apply them.

The technologies and trends we explore are categorized under five macro themes. While some technologies and trends may intersect multiple themes, our experts have sorted them under the theme they consider most relevant.

To learn more about our SDV services, and our deep software engineering heritage and expertise, visit Thoughworks' SDV page or AWS' Software-Defined Vehicles.

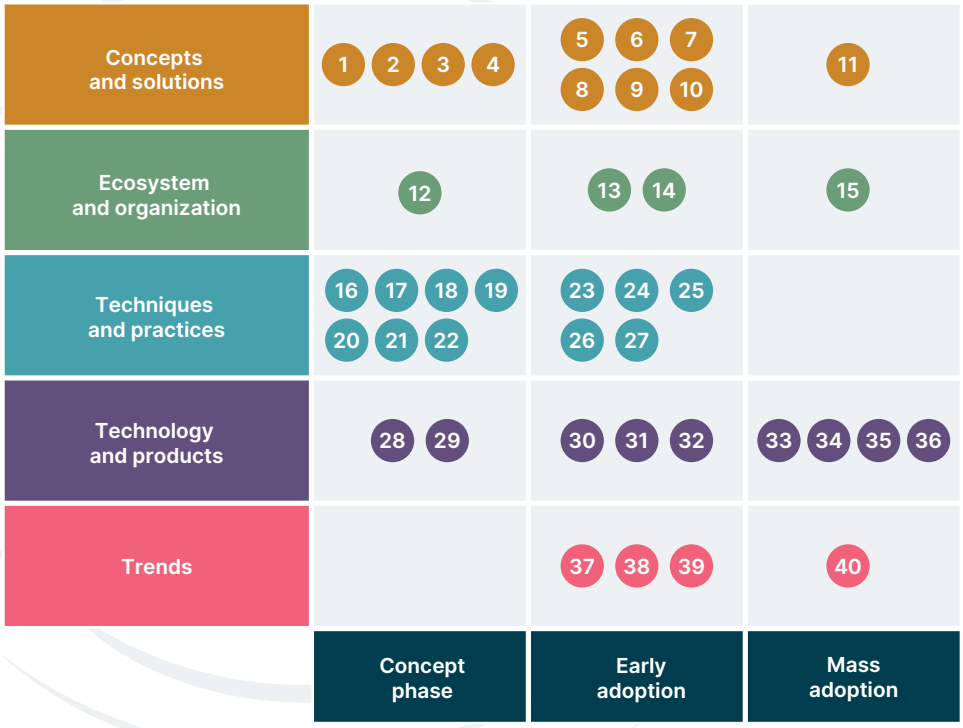
SDV Pulse at a glance

The Pulse chart has been designed to help organizations in the automotive industry keep track of relevant trends, developments and opportunities across the SDV space.

SDV Pulse is divided into five themes: concepts and solutions; ecosystem and organization; techniques and practices; technology and products; and trends, and organized by 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:** 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:** These Pulse Points are early on in their adoption journey and represent a significant differentiation opportunity for potential early adopters.
- **Mass adoption:** These Pulse Points are becoming widespread across the industry, and the window to translate them into differentiated value is closing.

SDV Pulse is forward-looking. In future editions, we will fade concepts that we see becoming less relevant, to help you track only the most relevant trends



Contributors

The perspectives shared across this report have been provided by a select group of senior technologists at Thoughtworks and AWS.

Both organizations have internal mechanisms to regularly share information creating a common body of knowledge. These teams came together and this edition of SDV Pulse is based on perspectives and knowledge shared prior to June 2024.

AWS

Aleksander Tolev
Senior Manager
Solutions
Architecture

Christian Mueller
Principal Solutions
Architect

Daniel Schleicher
Senior Global
Solutions Architect -
Automotive and
Manufacturing

Francesco Salamida
Head of Specialist
Solutions Architects -
Automotive and
Manufacturing

Sushant Dhamnekar
Senior Specialist
Solutions Architect

Thoughtworks

Erik Doernenburg
CTO
Europe

Marcos Silveira
Head of Technology
Automotive and
Manufacturing
Europe

Michael Fait
Global Head of
Technology
SDV

Sriram Jayaraman
Global Head of
SDV Business

Vivek Poovalingam
Principal Solution
Architect
SDV

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.

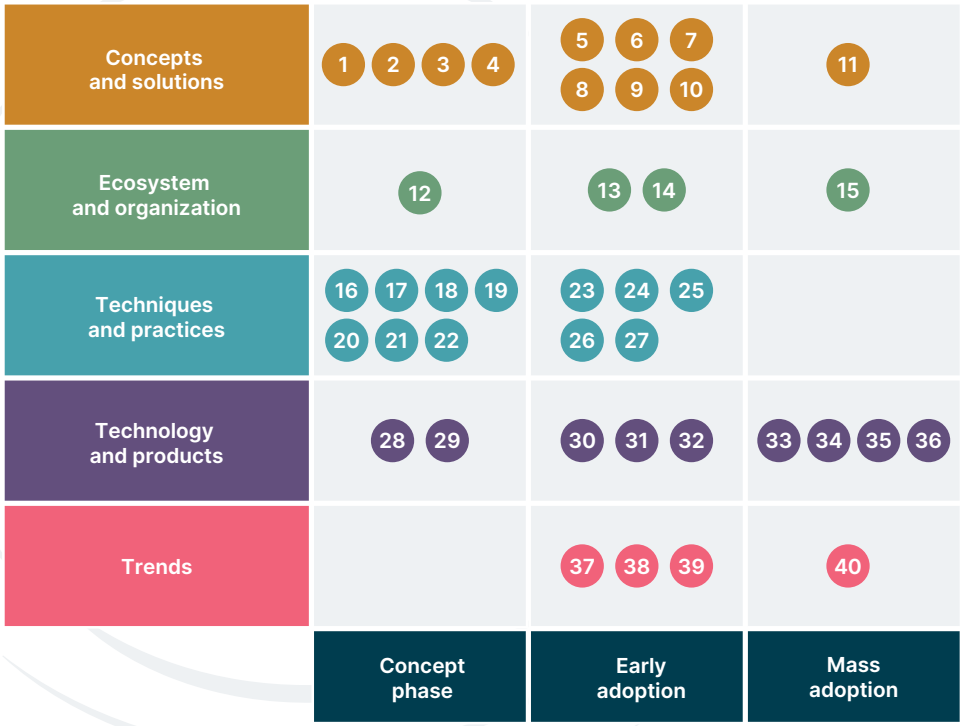
Ecosystems and organizations
The Pulse chart also includes emerging organizations and ecosystems that should be on every manufacturer's radar. These include ecosystems that you may wish to integrate with and become part of, as well as organizations such as regulatory bodies that could impact your operations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

The Pulse



Concepts and solutions

1. In-vehicle zero trust architecture
2. Integration of vECUs on virtual network
3. Liquid compute
4. Mixed critical orchestrator
5. Data mesh
6. Developer portals for vehicle APIs
7. LLM-enabled in vehicle voice assistants
8. SDV solutions on Marketplace
9. vECU
10. vECU levels
11. Centralized architecture as an enabler for SDV

Ecosystem and organization

12. digital.auto
13. Eclipse SDV
14. SOAFEE
15. EU Data Act

Techniques and practices

16. Automated continuous end-to-end testing
17. CAN over ethernet
18. Code reuse without copying
19. Continuous compliance
20. Cross-project VEW dashboard
21. Running new versions in shadow mode in-vehicle for validation
22. Software-first ECU development
23. AI-assisted software delivery for in-vehicle software
24. Incremental software delivery
25. Localized peripherals connected to virtualized ECUs
26. Shift-left security for in-vehicle software
27. Vehicle data-driven engineering

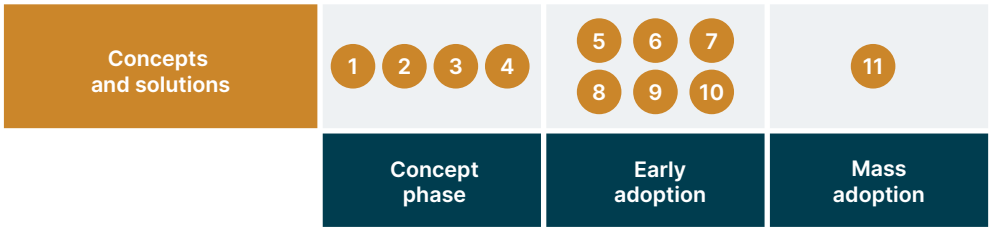
Technology and products

28. RISC V
29. VSS/VISS
30. ROS2
31. Rust
32. Time-sensitive networking
33. Android Automotive
34. Automotive Grade Linux
35. Hardware accelerators in the Cloud
36. Yocto project

Trends

37. Beyond making SOP
38. Seamless development between partners
39. Virtual engineering workbench
40. Organization shift towards SDV

Concepts and solutions



Concept phase

1. In-vehicle zero trust architecture
2. Integration of vECUs on virtual network
3. Liquid compute
4. Mixed critical orchestrator

Early adoption

5. Data mesh
6. Developer portals for vehicle APIs
7. LLM-enabled in vehicle voice assistants
8. SDV solutions on Marketplace
9. vECU
10. vECU levels

Mass adoption

11. Centralized architecture as an enabler for SDV

1. In-vehicle zero-trust architecture

Concept phase

In SDVs, the principles of zero-trust architecture (ZTA) are of paramount importance. With the ever-expanding role of software in driving functionalities, safety and user experience, conventional perimeter-based security isn't fit for purpose.

SDVs face a plethora of cyber threats. ZTA, with its fundamental principle of assuming no inherent trust, offers a robust defense mechanism against these risks. By enforcing stringent access controls, employing robust encryption protocols, and implementing real-time monitoring mechanisms, ZTA ensures continuous verification of every entity attempting to access the vehicle's network or resources. This approach, rooted in the principle of "never trust, always verify", guarantees that even if one component of the system is compromised, the overall integrity of the vehicle's operation will remain intact.

Industry-specific standards for automotive cybersecurity, such as ISO/SAE 21434, further underscore the need for robust security measures in software-defined vehicles. These regulations emphasize the importance of implementing ZTA principles, aligning with its core tenets of continuous verification and strict access controls to safeguard sensitive data and mitigate cyber threats effectively.

2. Integration of vECUs on virtual network

Concept phase

Typically, vehicle manufacturers purchase electronic control units (ECUs) from various suppliers and integrate them into their vehicles. Now, we're observing a new trend towards utilizing virtual versions of ECUs for development purposes and seeing more teams integrate multiple vECUs early in development.

Recently, we've seen architectures where supplier-owned environments hosting vECUs are interconnected with environments owned by other suppliers. This allows for the deployment of new software versions directly into the environments controlled by each supplier, giving them the ability to deploy new software at any time.

This streamlined approach facilitates faster integration of new software versions, enabling early detection of errors, when they're less costly to fix. However, an unresolved issue remains around synchronizing CPU clock speeds, which must be considered when analyzing test results from networks of multiple vECUs.

3. Liquid compute

Concept phase

Liquid Compute enables vehicles and OEMs to execute software functions wherever most suitable, based on current vehicle and environmental conditions. This could be in the vehicle, edge, cloud, or mobile network — seamlessly integrating in-vehicle and backend features.

Taking location search as an example, the function would normally be executed in the cloud, where more compute power is available to run more sophisticated algorithms to find the best direction, based on real-time traffic information, driving style and personal preferences. However, in areas where there is poor or no connectivity, you can run the function locally in the vehicle itself. Once connectivity is reestablished, Liquid Compute will offload those functions again to the cloud, ensuring seamless experiences while optimizing performance.

The idea is to package vehicle functions like that in containers so that you can run the same binary artifact inside and outside the vehicle, reusing the same code for efficiency in developing and maintaining the code.

4. Mixed critical orchestrator

Concept phase

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, this number must be controlled and, where possible, consolidated.

A key enabler of ECU consolidation is advanced silicon technologies like chiplets. Chiplets have lowered chip design costs through efficient intellectual property reuse, 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 via networks like CAN and Ethernet.

Previously, each ECU software component was designed to meet specific requirements, such as safety standards. With consolidation, these heterogeneous software stacks now share common multiprocessor systems.

Mixed criticality orchestrators (MCOs) can help OEMs ensure there's no interference between tasks. Safety-critical tasks, each with its own level, must run with sufficient resources — like memory and CPU — as if solely occupying the system. The MCO can manage multiple execution environments, including vehicle systems, edge computing in mobile operator networks with low latency, and more 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. Data mesh

Early adoption phase

The automotive industry's shift towards software-defined vehicles is accelerating growth in the volume of data organizations must gather and manage, the complexity of data use cases, and the potential to derive value from vehicle data.

In scenarios like this, a centralized data organization approach can become a barrier to innovation and effectiveness. First, data storage management often becomes disconnected from the specific data expertise, making effective data utilization challenging. Second, centralized data teams can become bottlenecks, impeding the fluid exchange of information between data providers and consumers. Plus, aspects like data quality, transformation and serviceability all suffer when they are not managed by consumers who directly depend on them.

Data mesh is an analytical data architecture and operating model in which data is treated as a product and owned by the teams that most intimately work with and consume that data. It can benefit organizations by granting autonomy to business units over their data domains.

This approach facilitates democratic data processing, where domain experts control data creation within a decentralized governance framework, leading to faster access and enhanced business agility. Additionally, by reorganizing the technical implementation around business domains, the architecture enhances flexibility, reduces operational bottlenecks, and promotes cost efficiency through real-time data streaming.

In the SDV world, we can visualize a data mesh by thinking of three main components:

1. Teams, organized by domain, such as vehicle subsystems or vehicle lifecycle, that own the production and consumption of their data and build value-driven data products.
2. A self-service ingestion platform that provides services for transferring vehicle data to the cloud.
3. A self-service data infrastructure platform that enables the publication of data products, governance and security aspects.

To learn more, explore AWS' deep-dive article on data mesh.

6. Developer portals for vehicle APIs

Early adoption phase

Multiple OEMs now offer portals to help developers build applications using vehicle APIs. The functionality can include vehicle status information like location, charging status, or errors. Some brands offer the ability to send data and commands to vehicles to allow use cases such as configuration of the navigation system or locking and unlocking the doors.

We see these developer portals as a key enabler of SDV ecosystem growth. They enable the creation of vehicle applications that are seamlessly integrated with digital services or other physical devices, like orchestrated fleets of vehicles or implementing flow with smart home products.

The developer experience and functionality of the portals available today varies, suggesting that we are still in the early stages of evolution and adoption.

7. LLM-enabled in vehicle voice assistants

Early adoption phase

As we enter a new era of human-machine interaction empowered by AI capabilities, natural language processing (NLP) has been leading the way in enabling machines to engage in interactions that closely resemble human-to-human communication.

However, precise algorithms tailored to specific domains are essential for achieving superior results. Combining the precision of NLP with the extensive contextual knowledge of large language models (LLMs) will enable a major step forward in the development of these experiences and interactions.

Within the automotive sector, OEMs have begun enhancing their voice assistants with LLMs to provide more human-like conversational ability. As consumers make purchasing decisions, refining user interactions with a brand using LLM technology could significantly improve their experiences, increase brand distinctiveness and drive loyalty.

8. SDV solutions on marketplace

Early adoption phase

The decoupling of hardware and software in SDVs creates two new opportunities for software development. Firstly, it enables developers to create off-the-shelf automotive tooling and data, and deliver them via marketplaces as Software-as-a-Service (SaaS) offerings. This innovative approach not only streamlines the development process but also opens up new revenue streams for automotive vendors and suppliers, eliminating the complexities of traditional licensing models.

Secondly, decoupling presents an opportunity for new entrants into the automotive software development ecosystem to provide services across the SDV technology stack. By offering plug-and-play solutions, new developers can cater to various automotive functions, ranging from infotainment systems to autonomous driving algorithms. This model not only fosters innovation within the industry but also creates additional revenue streams for emerging technology firms, tapping into the growing demand for advanced automotive software solutions.

9. vECU

Early adoption phase

Virtual Electronic Control Units (vECUs) facilitate the development, testing and validation of embedded automotive software and systems without the need for actual hardware. This approach enables engineers to rapidly develop and test new features and validate complex functionalities like advanced driver-assistance systems, 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. The use of vECUs in embedded software development and testing brings a paradigm shift towards simulation-based testing — also referred to as “shift-left” — enabling early and continuous testing of software before physical hardware is available. This approach aligns with agile development methodologies, fostering rapid iterations, seamless collaboration and continuous integration.

However, there are currently a few challenges with using vECUs. It can be difficult to ensure high simulation accuracy and faithfully replicate real-world conditions, address the complexities of hardware-software integration, and efficiently manage the vast amounts of data vECUs generate.

10. vECU levels

Early adoption phase

Virtual Electronic Control Unit (vECU) levels refer to the degree of simulation or abstraction used in the virtualization process. The vECU levels can be understood as follows:

0: Controller model: This most basic type of vECU consists solely of the controller model — or the C code derived from this model — and is limited to testing the control algorithm exclusively.

1: Application level: vECUs at this level feature production application software, supplemented with additional non-production functionalities from lower software layers for effective simulation. This setup includes the run-time environment and operating system, plus capabilities for data transmission. The vECU may contain all, part, or a distributed set of application functions, but it excludes direct hardware access.

2: Simulation middleware: At this level, vECUs are enhanced by adding simulated basic software functionalities. Unlike Level 1 vECUs that communicate only on the signal level, Level 2 vECUs can also simulate bus or network communications like CAN or Ethernet.

3: Production middleware: This level combines production application software with production basic software (BSW), focusing on hardware-independent components. These can be supplemented with simulated BSW, but the goal is to include as much production code as possible.

4: Target binary: Here, vECUs incorporate the actual production code compiled for the target physical ECU and may also include hardware dependencies. This integration allows for the inclusion of all software layers. To run this type of vECU on a simulation platform, an instruction set simulation is essential.

11. Centralized architecture as an enabler for SDV

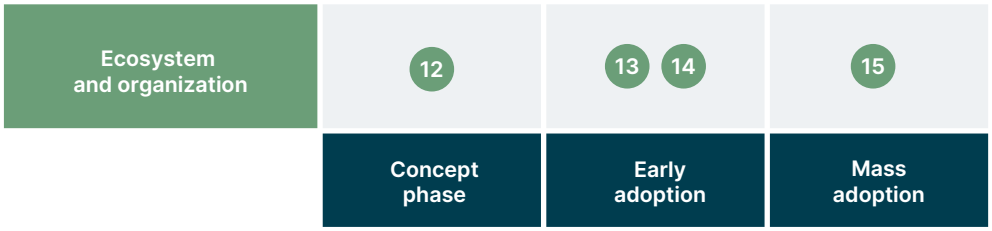
Mass adoption phase

Centralized vehicle architecture is a key enabler of SDVs, marking an evolution in the automotive industry's approach to vehicle design and functionality. Vehicles traditionally operate on multiple, distinct electronic control units (ECUs) that manage different functions separately, from engine control to infotainment systems. This distributed approach can lead to complexity in system integration, updates and maintenance, as well as limitations in scalability and flexibility.

A centralized vehicle architecture consolidates these distinct systems into a unified computing platform, streamlining the control and operation of vehicle functions. This shift simplifies the vehicle's electronic and software infrastructure, making the update process less risky.

The main component of a centralized architecture is the vehicle high-performance computer. It offers a robust foundation for SDVs, providing the computational power and flexibility required to support advanced software applications and updates.

Ecosystem and organization



Concept phase
12. digital.auto

Early adoption
13. Eclipse SDV
14. SOAFEE

Mass adoption
15. EU Data Act

12. digital.auto

Concept phase

The digital.auto initiative is an open community collaboration platform involving car manufacturers, suppliers, start-ups and end-users. It's a digital-first approach for the creation of next-generation customer experiences and data-driven mobility services. The open platform can be used to prototype use cases quickly and validate them on any connected target.

Key aspects of digital.auto include:

- Fostering interoperability in the SDV ecosystem.
- Offering a use-case-driven co-creation approach for fast, tangible results validated by real end-users.
- Providing a cloud-based, rapid prototyping environment for SDV features, known as the digital.auto playground.
- The digital.auto dreamKIT for automotive-grade hardware for rapid prototyping and validation of SDV applications.
- The ability to add any target to the playground with standard APIs, for testing use cases.

13. Eclipse SDV

Early adoption phase

Eclipse SDV is a working group within the Eclipse Foundation that aims to accelerate innovation in automotive software stacks by creating an open technology platform for the software-defined vehicle of the future. Its mission is to provide a platform for individuals and organizations to build and promote open-source solutions for the global automotive market.

Eclipse SDV:

- Encompasses various projects and initiatives related to in-vehicle software, communication, runtime, orchestration and more.
- Focuses on open-source development, collaboration, and sharing best practices.

- Provides a scalable, modular and extensible open-source licensed vehicle software platform.
- Adheres to high standards for quality management, security and safety across all vehicle domains.

The working group hosts a variety of projects aimed at advancing in-vehicle software development and promoting community participation. Notable examples include:

- **SommR:** Eclipse SommR is a Rust-based implementation of the Some/IP specification tailored for embedded POSIX systems in the automotive sector, complemented by essential developer tools for support.
- **uProtocol:** Eclipse uProtocol offers a communication protocol that operates independently of underlying transport protocols and can work with Eclipse Zenoh, SOME/IP, MQTT 5, Android (Binder), etc). It is adaptable across various deployment devices (such as vehicles, cloud systems, mobile phones, charging stations, etc.), regardless of the operating system in use.
- **Velocitas:** Provides an end-to-end, scalable, modular, and open-source development toolchain for creating both containerized and non-containerized in-vehicle applications.
- **BlueChi:** A deterministic multi-node service controller with a focus on highly regulated ecosystems. It integrates seamlessly with systemd via its D-Bus API.
- **Heimlig:** A Hardware Security Module (HSM) firmware for embedded platforms written in Rust. It typically runs on dedicated hardware and provides cryptographic services.

14. SOAFEE

Early adoption phase

Scalable Open Architecture for Embedded Edge (SOAFEE) is an industry-led collaboration that aims to create an open framework for collaborative development, testing and software deployment, based on cloud-native architecture.

It enables various automotive entities, including OEMs and their suppliers, to collaborate at every stage of the vehicle software lifecycle, ensuring compatibility across different vehicles and hardware configurations.

In the short time since SOAFEE was created, the consortium has gained tremendous traction and now has more than 50 technical voting members.

The SOAFEE project focuses on the development of four key SDV pillars:

- **Standards:** Developing new software architecture and methodologies that enable cloud-native development.
- **Industry Collaboration:** Working with various stakeholders for a unified approach towards creating a shared platform for vehicles using cloud-native architecture that accommodates multiple hardware configurations..
- **Vehicle Simulation:** To test and validate the software architecture.
- **Core Principles:** Ensuring safety, security and real-time capabilities are inherent in each pillar.

15. EU Data Act

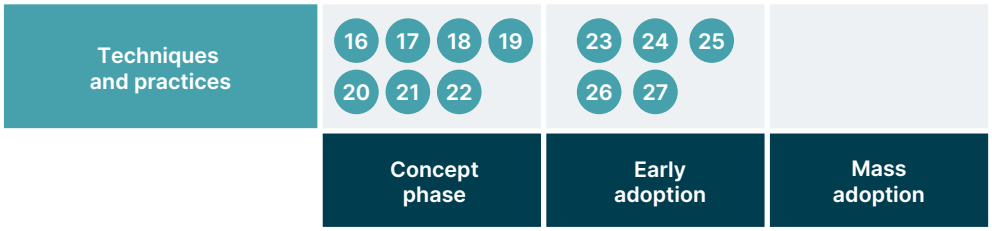
Mass adoption phase

The EU Data Act is a new piece of EU legislation that attempts to facilitate access to and use of data. It establishes rules for customers who wish to access data generated by connected devices and machines, allowing them to, among other things, switch data processing service providers more easily and share IoT device data with third parties. It applies across sectors and lays out principles and guidelines that must be followed by future sector-specific rules.

The set of measures in the Data Act directly impacts vehicle manufacturers, especially those further along in their Software-Defined Vehicle (SDV) journey. For example, it obligates data holders to make data available to users and third parties of the user's choice in certain circumstances, which has obvious implications for how SDV OEMs manage the data generated by vehicles.

While achieving compliance with the legislation may require OEMs to make significant process and operational changes, there are some benefits to complying with the legislation. Companies that deliver convenient access to data can enhance customer experiences and improve loyalty, in addition to ensuring they're acting responsibly.

Techniques and practices



Concept phase

16. Automated continuous end-to-end testing
17. CAN over ethernet
18. Code reuse without copying
19. Continuous compliance
20. Cross-project VEW dashboard
21. Running new versions in shadow mode in-vehicle for validation
22. Software-first ECU development

Early adoption

23. AI-assisted software delivery for in-vehicle software
24. Incremental software delivery
25. Localized peripherals connected to virtualized ECUs
26. Shift-left security for in-vehicle software
27. Vehicle data-driven engineering

16. Automated continuous end-to-end testing

Concept phase

To build robust vehicle software and bring exceptional SDVs to market, OEMs must continuously test software at every stage of its lifecycle. So, many OEMs are leveraging end-to-end development and testing platforms and Virtual Engineering Workbenches (VEWs).

To make testing an always-on process, they also need to embed testing capabilities at the system level. One example of a testing capability that can be embedded at the system level is automated regression testing. Automated regression testing involves rerunning a series of previously completed tests to ensure that new changes or additions to the software do not negatively impact existing functionality. By integrating testing capabilities into mobile devices, vehicle back-ends and in-vehicle systems, OEMs can automate continuous end-to-end testing.

17. CAN over ethernet

Concept phase

Controller Area Networks (CAN) have traditionally served as the primary communication bus for network communication within vehicles. However, with the introduction of High Performance Compute (HPCs) and Zonal ECUs and the increasing variety and complexity of use cases emerging from vehicle-generated data, Automotive Ethernet adoption is rising.

We're seeing many automotive organizations maintain the use of CAN-based communication between smaller ECUs and domain/zonal ones. The primary rationale behind this decision appears to be cost savings and the continuation of well-established ECUs from older vehicle models into newer ones. However, many now choose to send and receive CAN format data and messages over Ethernet.

Maintaining two different communication formats results in significant cost overheads and increases the complexity of software engineering. As software increasingly becomes the main driver of value in vehicles, and as more use cases depend on updates and service discovery capabilities even within deeply embedded ECUs, adaptive layers that can convert CAN format data and messages to Ethernet for communication will deliver significant value for automotive OEMs.

18. Code reuse without copying

Concept phase

In many automotive companies, we often see monolithic software components tightly coupled to the underlying hardware they were developed for. This often leads to a habit known as “clone and own” where the software of a previous hardware generation is fully copied and then further developed.

Having different versions of entire software systems usually leads to cluttered code, increased maintenance costs and poor quality. For many in-vehicle software components, this technique is no longer feasible.

A more sustainable approach is to decouple hardware-bound software from feature software and establish a modular architecture with loose coupling between components. Creating a set of libraries and modules that can be used company-wide across programs — or even in an open-source context — will result in the ability to reuse code without copying it.

Adopting this approach requires organizations to invest in refactoring existing software and establishing a different way of working within the company. However, we believe that the widely used “clone and own” approach has reached the end of its viable life across a lot of use cases.

19. Continuous compliance

Concept phase

Continuous compliance is the process of continually assessing automotive software components to ensure they always meet regulatory requirements and industry standards.

In the automotive software development process, compliance audits — including functional safety (ISO 26262, SOTIF), cybersecurity, and ASPICE — play a crucial role in the safety assessment of vehicle systems before homologation. But these audits can become a bottleneck to vehicle homologation, due to a lack of data and documentation to validate the functional safety of the system.

Continuous compliance ensures that software development processes and technologies comply with industry regulations and security standards on an ongoing basis. Organizations can automate compliance checks and audits and integrate tools into software development pipelines, allowing teams to detect and address compliance issues early in the development process.

Codifying compliance rules and best practices helps enforce policies and standards consistently across teams. It enables you to scan code changes for vulnerabilities, enforce coding standards, and track infrastructure configuration changes to ensure they meet compliance requirements. Lastly, automated reporting simplifies audits and provides clear evidence of compliance.

20. Cross-project VEW dashboard

Concept phase

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 not only to individual software artifacts or systems but also 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 respective requirements. As this process can be fully automated, dashboards will always reflect the current state of the development progress, providing OEMs with real-time visibility.

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

21. Running new versions in shadow mode in-vehicle for validation

Concept phase

Shadow mode refers to running two versions of a software component in parallel. One is live and impacts other components with its results, while the other is running in shadow mode, where its results don't affect other components. This allows teams to gather data relating to software performance and compare it with current production versions.

Taking adaptive cruise control (ACC) as an example; a vehicle sensor detects the distance to the vehicle in front frequently. The respective electronic control unit (ECU) continuously processes the data and generates output. When we deploy a new ACC variant into the vehicle in shadow mode, we can compare how our new version performs against the current one.

When enough data is collected from the field to prove that the new ACC variant outperforms the current one, the new ACC variant can be deployed to the entire fleet, becoming the new acting ACC variant. For features like ADAS, it's impossible to develop test cases for all imaginable scenarios. Running vehicle functions in shadow mode provides a safe method of testing new vehicle functions at scale, based on their behavior under real-world conditions.

With shadow mode capabilities in on-board architectures, you can also safely train new functions in the field while processing data in the vehicle and comparing results with expected outcomes.

This technique promotes the need for increased software quality in the automotive industry and is also known as "parallel run with reconciliation" in general software development.

22. Software-first ECU development

Concept phase

As use of virtual Electronic Control Units (vECUs) increases, vehicle manufacturers and suppliers are starting to create ECUs in innovative new ways. Traditionally, ECUs have been developed with a hardware-first mentality. In this mindset, technical details — such as CPU performance and memory capacity — have limited which software functions can be implemented on new ECUs.

We're seeing teams flip that model and prioritize software implementation throughout the ECU development process. With the ability to test software on virtual ECUs that simulate the limitations of multiple hardware platforms and configurations, software development teams can now demonstrate what's possible within these constraints.

Tailoring virtual resources is straightforward, allowing software development teams to showcase the potential for software functions in a new ECU. Having these demos available makes it much easier for vehicle manufacturers to assess customer value and tailor the necessary hardware resources for the respective ECU.

23. AI-assisted software delivery for in-vehicle software

Early adoption phase

The integration of AI into software delivery processes has the potential to boost productivity significantly and enhance the overall developer experience. And we're already seeing companies realize many of its potential benefits.

By automating some of the more tedious software development and delivery tasks, AI allows developers to concentrate on areas where they can truly make a difference and add value. Consequently, this leads to more productive engineering organizations that can deliver superior, high-quality software at speed.

Similarly, AI has the potential to accelerate the development of SDVs. Many AI-driven methodologies employed in design and analysis are transferable to embedded systems. However, code generation for embedded systems currently lacks variety and domain-specific knowledge. This can be addressed by expanding model capacity to include proprietary embedded software and platforms like digital.auto which facilitate experimentation in general SDV environments.

24. Incremental software delivery

Early adoption phase

Within SDVs, software updates can be continuously delivered directly to vehicles over the air. OEMs can deploy or activate new features, enhancements and fixes rapidly and frequently, without the need for physical access to the vehicle. It transforms the car into a dynamic platform that evolves to ensure ongoing optimization, the introduction of new services, and enhanced user experiences, mirroring the continuous delivery model prevalent across the software industry.

Developing the ability to deliver software updates over the air can 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.

25. Localized peripherals connected to virtualized ECUs

Early adoption phase

Vehicle manufacturers and suppliers are starting to use virtual versions of Electronic Control Units (ECUs) hosted in the cloud. The advantages are manifold. Unlike hardware ECUs, their virtual representations can be copied and instantiated anywhere in the world. This gives vehicle manufacturers the flexibility to speed up software development by having the ability to source software developers around the world and give each of them a separate instance of a virtual ECU. For infotainment ECUs, software developers are able to have the graphical interface streamed from the cloud to their development machine. Using the same technology, they can send back input events, like touch events originating from a touch screen or data coming from a local USB device, to the cloud. This gives software developers the feeling of having a physical representation of the hardware they are developing for on their desk while they are interacting with a virtual representation in the cloud.

26. Shift-left security for in-vehicle software

Early adoption phase

Shift-left security refers to 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 — marked by digitalization, separation of hardware and software, system expansions into the cloud, and growing software volumes — they become more attractive targets

for cyber attackers. This requires a fundamental change in how vehicle manufacturers approach software security.

We see companies developing capabilities in security and operations within teams, as well as adapting processes to automate security and compliance checks. Development teams are evolving their practices to embed security into continuous integration and deployment pipelines, conduct automated security testing to identify vulnerabilities in code and dependencies, and carry out constant threat model assessments for their applications.

27. Vehicle data-driven engineering

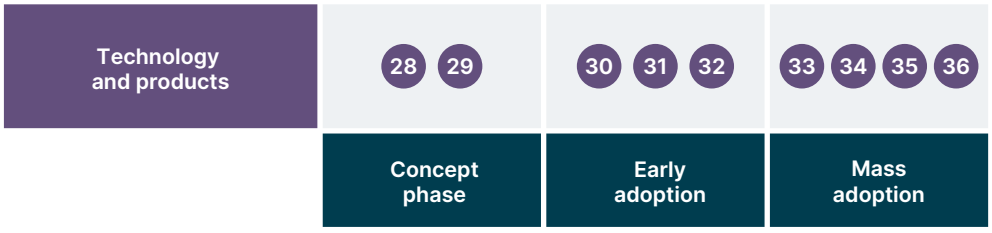
Early adoption phase

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 that vehicles generate.

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

The other challenge they face is managing that 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. But if organizations can do it effectively, they will engineer better vehicles — and stronger vehicle experiences — than ever before.

Technology and products



Concept phase
28. RISC V
29. VSS/VISS

Early adoption
30. ROS2
31. Rust
32. Time-sensitive networking

Mass adoption
33. Android Automotive
34. Automotive Grade Linux
35. Hardware accelerators in the Cloud
36. Yocto project

28. RISC V

Concept phase

RISC-V is an open standard instruction set architecture (ISA) — based on reduced instruction set computer (RISC) principles — which forms the foundation for compute devices such as microprocessors and microcontrollers. Unlike most other ISA designs, RISC-V offers an open-source ISA to which software can be ported, hardware can be developed, and processors can be built, all without licensing fees or royalties.

Also, because of its open-source nature and highly modular design, RISC-V enables the development of highly specialized automotive-specific processors.

As we look ahead, RISC-V is poised to drive a new era of processor innovation, offering extensible software and hardware freedom. Its architectural neutrality and collaborative development pave the way for the next wave of computing design and innovation.

29. VSS/VISS

Concept phase

The Vehicle Signal Specification (VSS) is a project by COVESA that aims to define a syntax and a catalog for vehicle signals, covering static attributes like brand, model and fuel type, as well as constantly changing telemetry data.

VSS has a hierarchical structure that organizes signals into categories and subcategories, providing a framework for accessing and managing vehicle data. VSS is accompanied by the Vehicle Information Service Specification (VISS), which defines a service for accessing the vehicle information. Together, they support connected services with vehicle data, but can also be applied to in-vehicle communication.

In the automotive industry, there is a clear need to make vehicle information more accessible through easy-to-use APIs. VSS and VISS provide a sensible blueprint for this. However, it remains to be seen whether they will become de facto standards that enable application building across models.

It's important to remember that every vehicle will have only a subset of the defined information available, and likely will have extra signals that are not defined in the standard catalog. Plus, VISS only defines authentication on an architectural level, which means it will likely work slightly differently for every brand. So, full standardization may still be some way away.

30. ROS2

Early adoption phase

ROS2 is an open-source middleware framework originally designed for the development of robotic systems. It provides a set of libraries and tools that enable the modular implementation of applications, covering functions like inter-process communication, multi-threaded execution, and quality of service. ROS2 builds on its predecessor by providing improved real-time capabilities, better modularity, and increased support for diverse platforms.

ROS2 is gaining traction in the automotive industry. Its node-based architecture and topic-based communication model are especially attractive for OEMs with more complex, evolving in-vehicle applications, such as autonomous driving functionality.

31. Rust

Early adoption phase

Rust is a programming language designed for performance and safety — especially safe concurrency. In most cases, its performance and power consumption are on par with C++. Moreover, Rust always guarantees memory safety without garbage collection.

The language eliminates many potential issues at compile time, making most automotive coding guidelines — such as Motor Industry Software Reliability Association (MISRA) rules — largely obsolete. Rust programs simply will not compile if it would cause potential memory issues. Rust also comes with a modern and mature ecosystem of tools and libraries that support the adoption of modern software practices, especially around testing and documentation.

This makes Rust a viable alternative to C++ for embedded automotive development, and many players in the industry are already experimenting with it today. The Ferrocene project qualified the Rust compiler against ISO 26262 (ASIL D) and IEC 61508 (SIL 4). The source code and qualification documents are open-sourced. We see this as a fundamental step towards using Rust in functional safety-relevant cases.

Rust's large-scale adoption appears to be imminent.

32. Time-sensitive networking

Early adoption phase

Ethernet has been the standard for inter-computer connection since the 1980s. However, its "best effort" approach doesn't guarantee messages reach their destination within a certain time frame — posing challenges for real-time automotive communications.

Timely and reliable communication between a vehicle's ECUs is critical. For example, if a braking system's sensors detect a pedestrian, the signal to activate the brakes must reach the braking ECU within milliseconds to avoid an accident.

To enable Ethernet for in-vehicle use, standards are evolving to prioritize messages by traffic class, but without fundamentally changing its best effort nature. Time-Sensitive Networking (TSN) — an Ethernet extension — now provides the synchronization and deterministic, low-latency communication required for real-time automotive applications.

By guaranteeing delivery times for critical messages, TSN makes Ethernet a viable alternative to technologies like CAN for in-vehicle communications. The automotive industry's adoption of Ethernet TSN is essential to connect the growing number of bandwidth-hungry ECUs for autonomous driving and advanced driver assistance systems.

33. Android Automotive

Mass adoption phase

Android Automotive (AAOS) is a variant of the Android operating system developed to serve as a foundation for in-vehicle infotainment systems. It's not to be confused with Android Auto, a mobile application to mirror smartphone features onto car displays.

AAOS can be used alongside Google's Automotive Services (GAS) to provide access to maps, an app store, and Google's assistant within vehicles. Since its launch in 2017, AAOS has seen relatively rapid adoption across the market.

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

However, while cost-effective and proven, using AAOS can limit an OEM's ability to deliver differentiated infotainment experiences. That's why — alongside the growth of AAOS — we've also seen many OEMs build their own infotainment OS based on the Android Open Source Project (AOSP). This gives them more control and greater ability to customize experiences.

34. Automotive Grade Linux

Mass adoption phase

Automotive Grade Linux (AGL) is an open-source project that OEMs and suppliers can use to kickstart the development of vehicle software. The goal of AGL is to provide a system that is 70–80% complete, giving OEMs the opportunity to adapt it to their needs by implementing the remaining 20–30%.

Initially started as a platform for building infotainment systems, AGL is the only organization that addresses all software in a vehicle. It has now reached a level of maturity where several major automotive manufacturers are adopting it for their production vehicles. The AGL team created a reference hardware specification to share between OEMs as a starting point for creating a commodity hardware platform.

35. Hardware accelerators in the Cloud

Mass adoption phase

Automotive companies send and store petabytes of sensor data to the cloud — a process that can be both time-consuming and expensive.

To turn that data into measurable value and apply it to train AI models, vehicle manufacturers need access to specialized compute accelerators in the cloud environment they use to store their model-training datasets.

Recently, physical hardware accelerators have become scarce and more expensive to acquire. Accessing hardware accelerators in the cloud enables OEMs to mitigate that challenge.

36. Yocto project

Mass adoption phase

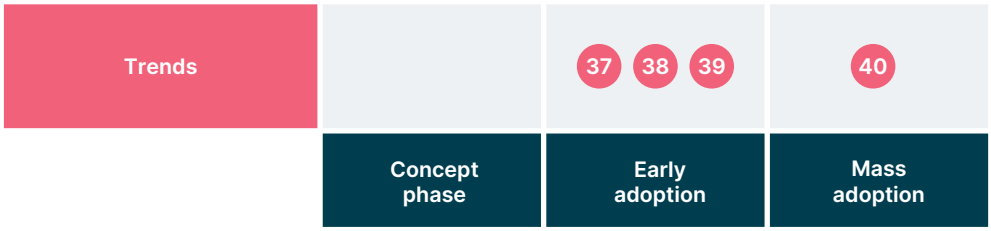
The Yocto Project is an open-source collaboration project that provides templates, tools and methods to create custom Linux-based systems for embedded products. It allows developers to create their own Linux distribution, tailored to their specific hardware and software requirements.

Yocto is not an embedded Linux distribution. Rather, it's a tool for creating a custom distribution for the user. It supports all major hardware architectures, including Intel, ARM, MIPS, AMD, PPC, and many others. The programming instructions required for Yocto are developed in the form of a 'BitBake recipe', which is designed in a highly flexible and modular manner.

Yocto is already widely adopted across automotive applications due to its ability to create customized, reliable and secure Linux-based systems for embedded devices. It's most commonly applied in five key use cases:

- **Customization:** Automotive systems often require specific configurations and features. The Yocto Project enables developers to build a tailored Linux system that meets the unique requirements of automotive applications.
- **Cross-Compilation:** Yocto supports cross-compilation, which is essential for compiling software on a development machine that can be transferred onto embedded systems with different architectures.
- **Real-Time Systems:** Yocto can be configured to support real-time systems, which are crucial for automotive applications where timing is critical.
- **Security:** With the Yocto Project, you can incorporate security features and ensure that the system is protected against vulnerabilities, which is vital for the safety-critical nature of automotive systems.
- **Environmental Parity:** Yocto can be used to achieve environmental parity, where the development, testing and production environments are aligned, which is particularly useful for cloud-native-software-defined vehicles.

Trends



Early adoption

37. Beyond making SOP
38. Seamless development between partners
39. Virtual engineering workbench

Mass adoption

40. Organization shift towards SDV

37. Beyond making SOP

Early adoption phase

Historically, automotive software development programs have had one business critical goal: to make the start of production (SOP). Features were designed upfront and deadlines were set years ahead in accordance with hardware development. As long as the software was ready when the vehicle was meant to go into series production, everything was fine.

With the complexity of in-vehicle software rising by an order of magnitude, the software development process has become a more significant element and cost factor within overall vehicle development. Increasingly, OEMs' focus is shifting away from just "making SOP" to investing in engineering effectiveness and continuously improving development processes and experiences.

In the era of the SDV, software engineers have a huge impact on the overall quality and success of a vehicle. So, evolving mindsets beyond "making SOP" and investing in building more robust and efficient development processes is crucial to every OEM's success.

38. Seamless development between partners

Early adoption phase

The development of vehicle software involves numerous organizations, each contributing components for the same Electronic Control Unit (ECU) or for multiple ECUs to interface with. This collaboration, while essential, often becomes inefficient due to disparate systems and a lack of unified 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, companies can ensure that all teams access and contribute documentation and other essential materials to the same central repository, which facilitates consistency and reduces redundancy.

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

39. Virtual engineering workbench (VEW)

Early adoption phase

VEW is a cloud-based software solution that integrates various tools and technologies to facilitate the development, testing and validation of vehicle functions within a virtual environment. This increases developer productivity, cuts costs, and drives innovation, particularly in complex projects like the development of autonomous driving technologies.

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 artifact 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.

40. Organization shift towards SDV

Mass adoption phase

The main obstacles to implementing SDVs among OEMs are organizational rather than technical. To shift toward SDVs, automotive organizations must tackle three key challenges:

- Building the capabilities and effectiveness of their software engineering organizations, to improve practices and output quality.
- Evolving team structures to ensure they’re agile and conducive to continuous improvement, development and delivery.
- Reorganizing around domains and technology, rather than discrete projects, to further support continuous delivery.

Many of the items in this publication — like automated continuous end-to-end testing, continuous compliance, shift-left security for in-vehicle software, and shifting testing left with virtualization— aim to push embedded development towards agility. Together, they will enable OEMs to rearchitect their teams and processes to continuously deliver exceptional software and experiences, and meet constantly evolving customer demands and regulatory requirements.

Conclusion

Don’t just be driven by SDV transformation. Drive it yourself.

The insights shared in this report represent an ongoing evolution across the automotive industry. As that evolution continues, every OEM, tier 1 supplier or system integrators has an important decision to make: to simply keep up with these changes or drive transformation proactively and shape the software-defined future themselves.

Thoughtworks leverages the broad and deep portfolio of AWS cloud services to help organizations modernize their infrastructure, capabilities, and practices enabling them to not just prepare for tomorrow, but actively shape their industry’s future.

If you’d like to explore the impact of any of these trends in more detail and discuss how your organization could transform them into powerful, value-creating opportunities, talk to us today.

We truly hope you’ve found the SDV Pulse report useful, and that it’s helped you gain a complete view of the diverse forces driving transformation across your industry. We intend to release new versions of this report periodically to help you stay ahead of these shifts — and new ones — as they evolve. So please look out for future editions.

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

Thoughtworks is a global technology consultancy that integrates strategy, design and engineering to drive digital innovation. We are 10,500+ people strong across 48 offices in 19 countries. Over the last 30 years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology as the differentiator.

/thoughtworks

Strategy. Design. Engineering.