FEDERATE

Software-Defined Vehicle Support and Coordination Project

D2.7 Second Technology Forecast Report

Authors

Ella Peltonen, Heidi Hietala, Yueqiang Xu, Olli Timonen, Sarthak Acharya, Benjamin Kämä, Vishaka Basnayake, Alireza Bakhshi Zadi Mahmoodi, Tero Päivärinta

University of Oulu, Finland

Bastian Lampe, Lucas Hegerath, Alexandru Kampmann

RWTH Aachen University, Germany

Mario Driussi

Virtual Vehicle Research GmbH, Austria

Peter Priller

AVL List GmbH, Austria

Karol Kobiela, Rutger van Beusekom, Bert de Jonge

Verum Software Tools B.V.

Alexander Viehl

FZI Research Center for Information Technology, Germany

October 2025

Deliverable	D2.7 – Technology Forecast Report
Work Package(s)	WP2 – WP Technology and high-level requirements solicitation
Dissemination Level	
Due Date	30-09-2025
Actual Submission Date	02-10-2025
WP Leader	VIF
Deliverable Leader	University of Oulu (UOULU)
Contact Person	Ella Peltonen
Email	ella.peltonen@oulu.fi

Document History			
Revision No.	**Date of the review**	**Name of the reviewer**	**Status of the document** ***(in progress, ready for review, released)***
V0.1	14-09-2025	Ella Peltonen	Ready for review
V0.2	22-09-2025	Mohamed Bejijou	Reviewed
V0.3	01-10-2025	Ella Peltonen	Released
V1.0	02-10-2025	Keinrath Claudia	Final Version

The project has been accepted for funding within the Chips Joint Undertaking (CHIPS JU), a public-private partnership in collaboration with the Horizon Europe (HORIZON) Framework Programme under Grant Agreement No. **101139749**

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or European Commission. Neither the European Union nor the granting authority can be held responsible for them.

Table of Contents

Executive Summary

The European Commission's Directorate-General for Communications Networks, Content, and Technology initiated a consultation process in late 2022, establishing the "Software-Defined Vehicle of the Future (SDVoF) initiative". The initiative launched its roadmap and vision document under the Chips Joint Undertaking (CHIPS JU)-funded FEDERATE Coordination and Support Action (CSA) in April 2024 [27]. The document addressed the European perspective on the rapidly changing automotive software industry market, highlighting key technical challenges such as the need for successful abstraction of vehicles' hardware components for software development, and the requirement for novel toolchains, middleware, and API solutions. The initiative has collaborative Research, Development, and Innovation (RDI) projects under the European Commission funding frameworks that focus on creating essential building blocks for the future software-defined vehicle.

In this Gap Analysis and Technology Forecast Report, Version 2, we highlight the academic and industrial perspectives on the key building blocks required to define, implement, and evaluate the software-defined vehicle concept in practice. We focus primarily on areas where significant gaps are identified. First, we discuss proposed SDV architectures and their challenges in terms of interoperability and paradigm shift from monoliths to microservices. Second, we analyse vehicles as a part of a broader continuum with other vehicles, roadside infrastructure, and edge-cloud computing capabilities, as future API developments will also require changes in the supporting systems and resources. Thirdly, we address validation and verification as integral parts of the SDV development pipeline, as vehicular software must be compatible with real-world complexities.

The automotive industry is undergoing a significant shift from monolithic to microservices, necessitating new architectures, interoperability solutions, and collaborations among various stakeholders. This shift requires flexible, service-oriented architectures, real-time data processing, and robust cybersecurity frameworks to support the increasingly complex and connected nature of traffic and vehicular systems. We emphasise that overcoming technical challenges, such as ensuring seamless interoperability between various components, platforms, and services, is crucial for the widespread adoption and economic success of SDVs. Creation, management, and governance of engaged ecosystems and collaborative efforts in building non-differentiating building blocks will enable collaboration and foster innovation in the future of autonomous and connected vehicles.

Overview

Purpose of this document

This document is the official Deliverable “D2.7 Technology Forecast Report” as promised in the proposal.

Due to the importance of the topic “Software Defined Vehicle of the Future”, the associated challenges, and earlier “Vision and Roadmap” publication (April 2024), this technology forecast and gap analysis document was made.

The published document is available under the link:

WP2 partner list

Name
VIF
UOULU
BMW
MBAG
RENAULT
CARIAD
CONTI
CONTI-FR
EB
RQTH
TUM
FZI
VECTOR

1 Introduction

1.1 Software-Defined Vehicle: Towards Programmable Interfaces

Modern cars incorporate technologies such as automatic braking, Cooperative Adaptive Cruise Control, lane departure prevention, and suggestions for charging stations, among others, to support drivers' cognitive abilities and reduce the risk of accidents. Despite the technological advancement, completely software-defined vehicles with accessible toolchains and APIs are still under development. Most in-vehicular sensors and interfaces are brand-specific or closed, limiting access to data, computing, and networking capabilities and thereby hindering, for example, the development of vehicular machine learning and artificial intelligence (ML/AI) applications. To enable connected vehicles to utilise all the available data sources, AI/ML computing resources, and networking capabilities, general interfaces and software platforms must be defined [27,87]. To reach this goal, vehicles must enable real-time computing, communication, and data resources for programmable interfaces. The current vehicular computing environment is still vendor-fragmented. It lacks practical and general interfaces and software systems that enable connected vehicles to utilise all the available data sources, AI/ML computing resources, and networking capabilities.

Connected vehicles and vehicular computing offer a potential solution for providing services and applications to drivers and in various traffic situations [7,56]. With increased networking and computing capabilities, vehicles can perform challenging inference and learning tasks to support drivers' cognition and provide additional information for route planning, intelligent charging, and driving safety. Such intelligent systems require training data, which can be provided by in-vehicle sensors and external databases. However, how to make this information available, processed, and utilised in a challenging real-time and mobile environment is still an open question. The development of vehicular edge [7,18] is foreseen to enable real-time, mission-critical, context-aware, and efficient intelligent applications. Such applications will support fully autonomous driving, which is known to be hazardous in imperfect conditions, and enable the driver to interact with the automation and regain control of the vehicle when required. In addition, many non-safety-critical applications will benefit from novel software interfaces.

Intelligent driving support and autopilot-driver interaction require machine learning (ML) and artificial intelligence (AI) solutions. The existing systems usually utilise data from in-vehicle sensors [18], such as cameras, LiDARs, radars, and speed meters [37, 77]. This information can be used to, for example, improve lane [66] and road pothole [38] recognition. Solutions for detecting drivers' behaviour while using smartphones during driving [120] and drunk driving [70] have been explored. However, the results underline that human drivers' perception and reasoning still maintain an advantage compared to fully automatic vehicles [101]. Different services and data sources are needed to understand the whole picture of driving performance and safety. Local real time computing can enhance drivers' situational awareness by allowing them to see "around the corner" and detect hazardous situations. To enable connected vehicles to utilise all the available data sources, AI/ML computing resources, and networking capabilities, general and open interfaces and software platforms must be defined [27, 87].

A software-defined approach to configuration, orchestration, and maintenance of the required sensors and actuators, data processing, storage, and network resources, and the resulting dynamic and complex systems of interacting vehicles (with their sub-components), edge nodes, and cloud services is required to make the vehicle-edge-cloud continuum possible in the first place. Recent developments in software-defined solutions and reference architectures for vehicular and edge-cloud computing have been proposed [87]. The concept of the software-defined vehicle is required to manage the numerous electronic control units, sensors, and their

connections through in-vehicle networks alone [28, 43]. Furthermore, the software-defined networking approach is a prerequisite for managing vehicular ad-hoc networks of connected vehicles, which are necessary for the evolving smart traffic and transport systems [20]. The offloading and external data services then require software-defined, multi-access edge-cloud solutions [8,119] to be dynamically and scalably orchestrated with the vehicular environment in a secure manner.

1.2 Project Setup and Initiatives

Initially, we identify and define the key initiatives behind the results discussed in this article. First, the European Software Defined Vehicle of the Future (SDVoF) Initiative is led by the European Commission's Directorate-General for Communications Networks, Content, and Technology. It emphasises collaboration among European Original Equipment Manufacturers (OEMs) and suppliers, including those in the semiconductor industry. FEDERATE, funded under the Chips Joint Undertaking (CHIPS JU) framework, is a Coordination and Support Action (CSA) that supports the SDVoF initiative by promoting the decision-making process, fostering the collaboration between projects, and ensuring alignment with European actors' strategic objectives. Both initiatives engage in collaborative Research and Innovation Actions (RIA) to achieve their goals, funded by either the Horizon EU framework or national funding resources. The current list of consortium partners and associated members of different projects and initiatives can be found on the FEDERATE website (https://federate-sdv.eu/). This article reflects the SDVoF and FEDERATE's set Vision and Roadmap [27] and extends the Gap Analysis and Technology Forecast Reports published in September 2024 [85].

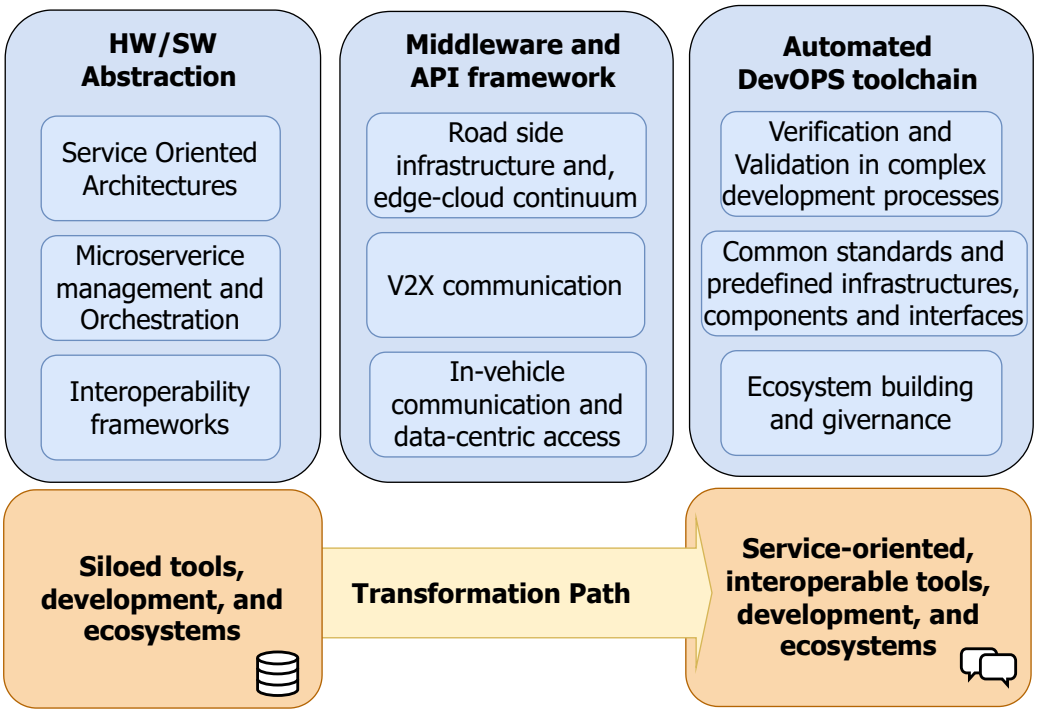

Figure 1: An overview of the FEDERATE SDV building block categories (Hardware/Software Abstraction, Middleware and API Framework, and Automated DevOps Toolchain) and corresponding themes discussed in this paper.

1.3 Data Collection Methods

This article aims to define and present key challenges in approaching the SDV era within the automotive industry. The SDVoF initiative and FEDERATE project consortiums include the leading European automotive OEMs (e.g. Mercedes-Bentz, BMW, Renault, VW, Cariad, Stellantis, Volvo Trucks, Renault, and Ford Otosan), semiconductor companies (e.g. ARM, NXP, Infenion, and ST Microelectronics), automotive software businesses (e.g. Bosch, ETAS, Accenture, Alkalee, Continental, Electrobit, Valeo, ZF, and TTTECH), industry associations and foundations (e.g. Eclipse Foundation, AUTOSAR, COVESA, VDA, and EUCAR), and academic institutions around the continent. As the projects were being drafted, a series of workshops came together to define the common goals, establish a shared understanding, and articulate a vision for the future. Based on these discussions and documentation, a series of deliverables has been published. For this research article, these published documents serve as the primary data source.

Second, we address the identified SDV Building Blocks (see below) from the FEDERATE results. The building blocks are publicly documented in the FEDERATE Github and any of the project partners or associated members can participate in defining them, either from an industrial or academic perspective. Considering the broad representation of the automotive industry and OEMs, we can cautiously claim that the identified building blocks represent the European perspective on SDV development. The building blocks are regularly updated and discussed in the telco series organised by the FEDERATE project for the partners and associated members.

Third, the FEDERATE scientific working group has organised a series of talks and presentations for the partners and associated members. These presentations and their follow-up discussions formed a group of researchers from the academy and R&D operations who were involved in providing their perspectives and experiences for defining the key challenges. In this article, we present the results of these discussions, combined with the public documentation and building block definitions from the FEDERATE project. The building blocks and themes they represent are also surveyed from the perspective of the latest research literature.

1.4 Building Blocks for Software-Defined Vehicles

Building blocks (BB) are non-differentiating reusable components that enhance the development of the automotive software stack and complementary parts on the edge-cloud continuum [27]. Agreeing on these building blocks and maintaining them continuously are the key objectives of the SDVoF initiative and related research, innovation, and development projects. By building this shared knowledge and agreement with "atomic" services, the critical building blocks, we expect to have commonly agreed concepts, components, and interfaces that are highly dependable, robust, secure, and well-tested. Initially, three main categories of building blocks were considered. However, these will be defined more robustly and specifically when the RIA projects progress. The main building block categories are:

- **Hardware/Software Abstraction:** These building blocks separate hardware components from software, allowing the software to function independently of the underlying hardware. They facilitate interoperability and enable efficient integration across various hardware platforms**.**
- **Middleware and API Framework:** These building blocks provide software tools that connect the hardware and application layers, providing essential services like communication, security, and data management. They ensure seamless interaction between different software modules.
- **Automated DevOps Toolchain:** These building blocks provide tools that automate the software development lifecycle, including continuous integration, testing, deployment, and monitoring. The goal is to speed up development and ensure software quality.

This paper analyses the state of the art of the mentioned building block categories from a software engineering perspective, considering gaps in knowledge and technical implementations that should be developed soon (see Figure 1). We cover Software-Defined Vehicle architectures and the interoperability aspects of hardware/software abstraction. We cover the Edge-Cloud Continuum, roadside infrastructure readiness, and vehicle-to-vehicle and in-vehicle communication aspects as the critical building blocks for seamless integration into vehicular computing space. Finally, we discuss validating and verifying software using automated DevOps toolchains and building healthy software ecosystems to ensure the vehicular software life cycle and the quality of end products. As the SDV development will require transformation from monolithic software components and silo providers to microservice-based architectures and agile ecosystems, we discuss the transformation path, aiming to provide not only technical capabilities but also integrity, accountability, and dedication towards the shared goal.

2 Software-Defined Vehicle Architectures

This section provides an overview of the SDV architectures that should be considered in the future. Software-defined vehicles represent a paradigm-shifting evolution in electrical and electronic (E/E) architectures, offering a significant increase in flexibility compared to traditional system architectures. This flexibility is crucial for supporting the rapid development cycles demanded by autonomous driving technologies, where the separation of software and hardware functions allows for more dynamic and adaptable vehicle systems [51, 57].

Traditional distributed E/E architectures typically feature tightly coupled hardware and software optimised for specific feature sets. As such, the industry is increasingly moving towards SDVs, which utilise new E/E architectures with high-performance computing capabilities [98, 134]. Containerisation and virtualisation are essential technologies within the SDV framework, facilitating rapid software deployment and updates crucial for maintaining the performance of intelligent vehicles [128]. Moreover, Model-Based Systems Engineering (MBSE) is increasingly employed in the automotive industry to manage the complex design processes of SDVs. MBSE addresses the resource allocation challenges by formally describing vehicle resources, safety requirements, and optimisation objectives [14]. This shift brings opportunities and challenges, particularly in vehicle architecture, cybersecurity, and system integration.

2.1 Service-Oriented Architectures (SOA) in Automotive

SOAs are emerging as a promising solution to the challenges posed by the increasing complexity of automotive software systems. SOA enables the dynamic integration of software components, allowing for more flexible and scalable vehicle architectures. Unlike traditional tightly coupled systems, SOA decouples software from hardware dependencies, emphasising software-driven design and modularity. This separation facilitates easier updates, component reuse, and integration of third-party services without significant system overhauls, contributing to faster development cycles and reduced maintenance costs [62].

With software becoming the central focus, SOA supports the modular development of functionalities, allowing manufacturers to dynamically introduce new features and updates, improving adaptability throughout a vehicle's lifecycle. Research has shown that SOA can significantly enhance the functional suitability and scalability of automotive software systems. However, implementing SOA in the automotive domain also presents challenges, particularly in ensuring the system's security, safety, and reliability. Despite these challenges, SOA is gaining traction as a critical architectural approach for future automotive systems, offering a way to manage the complexity of SDVs while maintaining high performance and security standards.

2.2 Mainstream Classic Architecture

AUTOSAR Classic Platform: The AUTOSAR Classic Platform has been the cornerstone of automotive software development for over a decade. It provides a standardised framework for the development of deeply embedded systems, with an emphasis on safety, security, and predictability. The layered architecture facilitates modular development, allowing for the seamless integration of various hardware and software components, essential for meeting stringent real-time requirements [16].

2.3 Popularized Architectures

Adaptive AUTOSAR: This builds upon the classic platform by introducing flexibility and adaptability, critical for modern vehicles that require dynamic software updates and re-configurations. This architecture supports the complex software demands of autonomous driving and connected car technologies. By providing standardised services and APIs, Adaptive AUTOSAR facilitates the integration of new functionalities as vehicle software continues to evolve [15]. However, Adaptive AUTOSAR faces challenges related to achieving determinism [76].

Automotive Grade Linux (AGL): This open-source initiative seeks to create a unified software platform for connected cars. Supported by significant manufacturers, AGL is noted for its flexibility and rapid development cycle. Its influence extends beyond IVI systems to cover telematics and instrument clusters, making it a versatile and integral part of the automotive ecosystem [70].

Android Automotive OS: Google's Android Automotive OS is gaining traction due to its robust ecosystem and extensive integration capabilities. It supports extensive customisation and third-party app development, making it a versatile platform for infotainment and broader vehicle system integration. Its user-friendly interface and the leverage of the expansive Android developer community enhance its appeal across the automotive industry [93].

Robot Operating System 2 (ROS 2): ROS 2 [75] is an open-source framework used for developing robotic applications, with growing adoption in automotive systems, particularly in autonomous driving [10,97,115]. Compared to its predecessor, ROS 2 offers an improved communication infrastructure, enhanced security, and better support for real-time operations [75]. Its modularity, flexibility, and middleware abstraction facilitate the integration of sensor fusion, path planning, and control algorithms in self-driving vehicles. However, despite its advancements, ROS 2 still faces challenges in achieving reliable real-time performance and deterministic behaviour [21, 22], which are essential for safety-critical automotive systems.

Automotive Service-Oriented Architecture (ASOA): ASOA [54] addresses the limitations of traditional monolithic automotive architectures. It employs a runtime-integrated service-oriented approach, enabling software components to operate as platform-agnostic services that are dynamically connected through a central orchestrator. This design supports updates, replacements, and reuse of services, promoting adaptability and modularity. ASOA has been applied in a full-scale automated vehicle [119], highlighting its potential for use in networked systems for connected and automated vehicles (CAVs).

It is worth mentioning that the Android Open Source Project (AOSP) and Android Automotive OS (AAOS) are part of Google's SDV Ecosystem. The AOSP forms the open-source foundation for Android, offering a flexible and modifiable platform that manufacturers and developers can leverage to build custom systems. AOSP cannot use Google Services, but it can use Android Runtime for APK execution. Android Automotive OS (AAOS) extends AOSP by integrating vehicle-specific features, such as the Vehicle Hardware Abstraction Layer (HAL). This enables direct communication between the operating system and car hardware components such as infotainment, HVAC, and other in-car functionalities. Unlike AOSP, AAOS is specifically tailored for automotive applications, offering a comprehensive framework for in-car system development.

On the other hand, AAOS does not include Google Services unless Car OEMs sign a separate contract. However, manufacturers can enhance AAOS further by incorporating Google Automotive Services (GAS), a proprietary suite of services that includes Google Maps, Google Assistant, and the Google Play Store. While AAOS functions independently, providing automakers the flexibility to customise and develop unique in-vehicle experiences, integrating GAS introduces Google's services ecosystem, enriching user experience. This integration, however,

requires a licensing agreement, making GAS optional but highly valuable for delivering seamless connectivity, app availability, and advanced navigation capabilities, enhancing AAOS beyond its native feature set [88].

Safe Open Vehicle Core (S-CORE) an Open-Source Ecosystem: Eclipse S-CORE: Establishing an Open Source Ecosystem for Software-Defined Vehicles. The growing complexity of Software-Defined Vehicles (SDVs) necessitates a shift toward modular, scalable, and standardised software architectures. The E/E architectures are also undergoing a change towards centralised E/E architectures that provide so-called high-performance computing units. The Eclipse S-CORE (Safe Open Vehicle Core) project is in its early stages of development. It aims to address this challenge by developing an open-source core software stack that bridges the gap between the new operating system used, which supports multi-threaded schedulers with memory and resource separation in processes, and the application layers that require a common interface with functions and features for the different application domains. By following a collaborative, non-differentiating development approach aligned with ASPICE, ISO 26262, ISO 21434 and ISO PAS 8926, S-CORE seeks to establish a high-quality, safety-compliant runtime solution that is both cost-efficient and scalable. The overarching goal is to "Build the Best Automotive Runtime Solution Only Once and as an Open-Source Solution", allowing industry stakeholders to converge on a shared, optimised software stack instead of maintaining multiple proprietary alternatives.

To achieve this, Eclipse S-CORE is actively working on creating an open-source ecosystem that extends beyond code-sharing. The project focuses on collaborative governance, standardised interfaces, and best practices to ensure interoperability and cross-industry adoption. By fostering an environment where OEMs, Tier 1 suppliers, and technology providers can co-develop a common SDV runtime stack, the initiative aims to drive efficiency, reduce costs, and accelerate innovation. As the project progresses, its success will depend on broad industry participation, the refinement of development methodologies, and the adoption of open innovation principles that can redefine how automotive software is developed and maintained.

2.4 Latest Developments and State-of-the-Art Architectures

Microservice-Based Architectures: A major trend in SDV architectures is the shift towards microservices, where the vehicle's software is broken down into small, independent services that can be updated and scaled independently. This architecture supports continuous integration and deployment (CI/CD), enabling manufacturers to roll out updates and new features more frequently with reduced risk of disrupting existing functions [41].

Containerised Solutions: To manage these microservices effectively, containerisation platforms like Kubernetes are employed. These platforms provide the necessary infrastructure for deploying, managing, and scaling services across various environments—on the vehicle, at the edge, or in the cloud. This approach aligns with the broader industry trend towards cloud-native technologies, enhancing the resilience and scalability of automotive software architectures [65].

AI-Driven Architectures: Artificial Intelligence (AI) is becoming increasingly integral to SDV architectures, particularly in autonomous driving systems and advanced driver-assistance systems (ADAS). AI-driven architectures enable real-time decision-making and predictive maintenance, improving the vehicle's adaptability to changing conditions and enhancing safety [13, 50].

2.5 Vision and Roadmap Towards Microservices

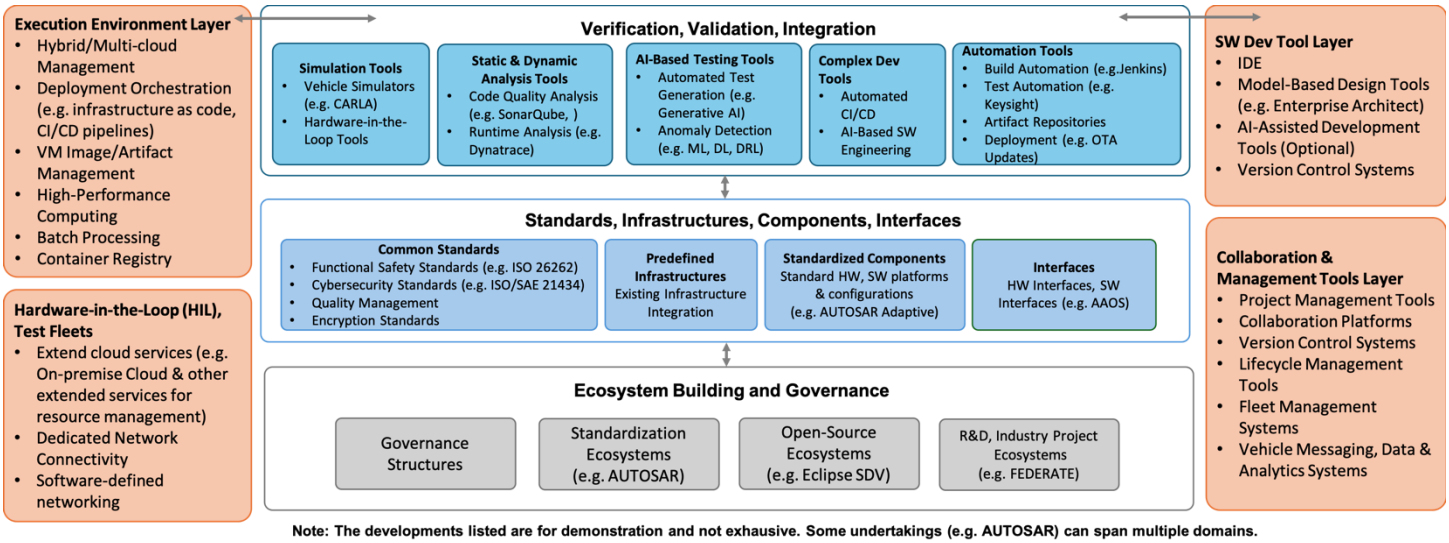

Figure 2: An example of the Layered Tooling Reference Architecture for SDVs.

To this end, the FEDERATE project and SDVoFs initiative advance these state-of-the-art principles by integrating a holistic approach that builds upon the foundational concepts of microservices and modular architecture and extends them to address the specific challenges and opportunities presented by the next generation of vehicles. For instance, the project's approach extends beyond traditional microservices by considering the potential of dynamic service orchestration, which enables the vehicle's software stack to adapt in real-time to varying conditions, such as changes in network connectivity, cybersecurity threats, or evolving user preferences. This capability is particularly critical in autonomous and connected vehicles, where the ability to respond to and recover from unforeseen events rapidly is a crucial determinant of safety and reliability.

Thus, this transition to microservices and modularity is well-aligned with the Layered Tooling Reference Architecture for SDVs, promoting a structured software development approach (see Figure 2). This architecture typically includes the Hardware Abstraction Layer, Operating System Layer, Middleware Layer, Application Layer, and User Interface Layer. Each layer has specific tools and frameworks designed to support the development, testing, and deployment of microservices, ensuring optimised operations across the entire vehicle software stack [40]. Some reference architectures, such as RobotKube [67], already exist for orchestrating containerised microservices in large-scale multi-robot systems using Kubernetes and ROS. RobotKube is built on an event-driven architecture and can automate software deployment, configuration, and data collection across SDVs and other connected entities in Cooperative Intelligent Transport Systems (C-ITS). Key components include an event detector and an application manager that orchestrates software based on real-time data. Open challenges include reducing orchestration latency, ensuring scalability and compatibility in heterogeneous systems involving diverse hardware and software platforms, and efficiently managing resources in environments with limited computational power, memory, and network bandwidth.

2.6 Key Challenges

- **Complexity Challenge:** The complexity of transitioning from traditional, monolithic architectures to dynamic, service-oriented architectures provides integration and transformation challenges, especially given the reliance on tightly coupled hardware and software systems.
- **Cybersecurity Challenge:** The architecture should provide security already on the design level, as more interconnected, software-based platforms make vehicles more vulnerable to cybersecurity threats.
- **Safety and Security Challenges in Automotive Software Systems:** the SOA Perspective. One of the challenges in applying a dynamically orchestrated SOA in SDVs lies in managing the inherent trade-offs between flexibility and strict real-time performance in safety-critical functions. While SOA provides modularity and scalability, allowing services to be composed and reconfigured dynamically, these benefits introduce risk factors with the deterministic execution required in hard real-time systems. In the context of functions such as brake control, where the latency of any operation could make a critical difference, the orchestration of services introduces variability in execution times and communication latencies, making it challenging to guarantee the fixed response times necessary for critical automotive operations. In this vein, academia and industry are investigating hybrid architectures that combine the benefits of SOA with tightly controlled, deterministic pathways for critical functions. These hybrid systems aim to preserve the modularity of SOA while isolating real-time, safety-critical processes to ensure they meet the stringent timing and reliability requirements essential for automotive safety [100].
- **Resource Allocation Challenge:** Efficient resource allocation presents a significant challenge for service-oriented architectures in SDVs. Quick adaptability requires the optimal assignment of computing resources, including scheduling tasks and mapping services to CPU and GPU cores, especially when utilising AI component deployment for AI-based functionalities, and managing communication priorities between services. However, the limited availability of resources necessitates careful optimisation to ensure both performance and reliability. This requires a software framework that supports dynamic resource orchestration while making resource constraints and dependencies visible. Solutions like optimisation-based resource allocation [55] offer strategies to minimise power consumption and execution times of critical effect chains by integrating scheduling and resource mapping into a unified optimisation model.
- **Architectural Challenge:** Integration of real-time information from digital transportation infrastructure in the overall architecture is currently limited to predefined vehicle-to-everything (V2X) communication standards. The holistic development and deployment of novel functions, including those for vehicles and infrastructure, are not considered in current architectures.
- **AI/ML Challenge:** SDVs rely on AI/ML for various functionalities. However, deploying AI/ML in safety-critical automotive environments and systems requires significant work and development in terms of data management and quality aspects of AI/ML, such as high-quality data labelling and annotation for model training, data bias and mitigation strategies, addressing continuous learning and adaptation AI systems from new data collected in the real-life environment while maintaining safety and stability [116]. Some of these aspects, particularly those related to AI/ML-based perception and decision-making, are already addressed in ISO 21448 (Safety of the Intended Functionality – SOTIF), which provides a useful framework for mitigating risks beyond hardware/software failures. However, regulation-related aspects are becoming increasingly in demand as AI/ML models are being regulated by stringent policy requirements and as the trustworthiness and explainability of AI, which are still relatively nascent domains, gain attention [11].
- **Interoperability Challenges:** As SDVs are complex ecosystems involving diverse components (hardware, software) from many vendors, interoperability is critical for seamless integration and functionality across these components. For instance, heterogeneous architectures with varying

processing capabilities and communication protocols pose significant challenges for software and hardware integration. Furthermore, although open platforms and open-source architectures can provide benefits such as fostering interoperability and reducing vendor lock-in, security and quality concerns are inherent challenges of the open-source model. The harmonisation of open-source and commercial components introduces additional architectural complexity [56].

3 Interoperability Aspect in SDVoFs

Interoperability is a cornerstone of modern industrial and automotive systems [4], particularly in the context of SDVs. As the automotive industry shifts toward increasingly complex and interconnected digital environments, ensuring that different systems, components, and software solutions work seamlessly together is critical for innovation and efficiency. For example, the Eclipse Arrowhead framework, known for its Service-Oriented Architecture-Based interoperability and microservices, provides a robust architecture that facilitates seamless interaction by enabling scalable and interoperable automation solutions. Such frameworks can enhance the integration of diverse systems across the automotive value chain within the ecosystem, ensuring that different stakeholders—from OEMs to software developers—can collaborate effectively.

Seamless exchange of data between two or more systems or components and the use of them as meaningful information is what is defined as interoperability [99]. The definition of interoperability has evolved considerably over the last two decades, with the development of numerous technologies, application systems, and multidisciplinary approaches to engineering. Recent works [79] related to context-aware software systems (CASS) have discussed an interoperability theoretical framework (ITF), which examines interoperability from two aspects: structural and behavioural. The structural one considers context, perspective, levels, purpose, and attributes, which compose interoperability. The behavioural aspect deals with its evaluation methods, challenges, issues, and advantages. SDVs have their basic vehicle controls and advanced autonomous driving functions, mainly depending on the software that must interoperate efficiently.

In the current era, SDVs incorporate diverse hardware and software from suppliers that follow numerous communication protocols to interact with the external world, i.e. environment and infrastructure. Therefore, studying interoperability in this context is a technical necessity, laying the foundation for innovative research and development. The need for open and interoperable platforms has necessitated that classical architectures in SDVs adopt SOA and microservice architectures. To ensure dynamic interactions between in-vehicle and external systems, monolithic software is broken down into independent services, developed and deployed without disrupting the overall vehicle system [94].

3.1 Six-Tier Interoperability Framework for SDVs

Different components, data sources, and interfaces in Software-Defined Vehicles are often sourced from various vendors, leading to compatibility issues. Therefore, interoperability is crucial in ensuring the integration of heterogeneous systems, adapting legacy systems, facilitating real-time communication, enabling third-party integration, ensuring scalability, promoting cross-system compatibility, and allowing for modular software updates, among other benefits [3]. Cohesive interaction among different systems, components, and platforms requires a multi-layered concept of interoperability. The FEDERATE project and SDVoF initiative consider multi-dimensional facets of interoperability in its SDV architecture. Each level of interoperability contributes to seamless interactions with cloud services and V2X networks, facilitating efficient and secure communication.

The analysis of different levels of interoperability for SDVs is provided in Table 1. This Six-Tier Interoperability Framework for Software-Defined Vehicles provides a comprehensive approach to ensure seamless communication and coordination across systems [2]. It begins with technical interoperability, which focuses on the communication between hardware and software, and progresses through syntactic and semantic interoperability to ensure that data is correctly formatted and interpreted. Pragmatic interoperability

addresses the contextual use of data, while dynamic interoperability ensures real-time adaptability and flexibility. Finally, organisational interoperability emphasises collaboration between stakeholders, such as manufacturers and regulators, to promote cohesive development and compliance across the SDV ecosystem.

Interoperability Level	Importance in SDVs	Communication Protocols	Standards & Frameworks	Critical Considerations
Technical Interoperability (Level 1)	Ensures that hardware and software components (sensors, control units, actuators) can communicate with each other.	CAN, FlexRay, MOST, Ethernet, SOME/IP, DDS (data Distribution Service), TSN (time-sensitive networking), LIN (local interconnect network).	AUTOSAR, AGL (Automotive Grade Linux), ROS (Robot Operating System), ISO 26262 (functional safety), IEEE 802.3 (Ethernet), SAE J1939.	- Low-level compatibility. - Real-time data acquisition & processing. - Integration of diverse components from various vendors.
Syntactic Interoperability (Level 2)	Focuses on defining data formats, communication syntax for system-wide communication and ensures that data transmitted between systems is readable.	XML, JSON, SOME/IP, DDS (RTPS), Protocol Buffers, ASN.1, MQTT, OPC-UA.	ISO-22900, ISO-14229, OBD-II, SAE J1939, VSS (vehicle signal specification)	- Data representation consistency for in-vehicle & external communication (cloud). - Data parsing. - Compatible syntax
Semantic Interoperability (Level 3)	Ensures exchanged data between systems is interpreted with the same meaning.	V2X (DSRC, C-V2X), SOME/IP, DDS (contracts), GPS, CoAP, ROS2, 5G-NR	IEEE 802.11p (DSRC), IEEE 1609.x (WAVE), 5G-NR, ISO/IEC 15531-31, ETSI TS 103 301 (C-V2X), SAE J2735 (V2X message sets), ISO 26262.	- Misinterpretation can lead to critical failures in decision-making scenarios.
Pragmatic Interoperability (Level 4)	Focuses on context-aware use of data. E.g. For executing correct actions based on data inputs (interpreting road signs and adjusting vehicle behavior)	ROS2, ZeroMQ, MQTT, Apache Kafka.	SAE Level 3-5(SAE J3016), ISO 21448 (SOTIF), ISO 24089.	- Real-time operational decisions. - Accurate interpretation of the context.
Dynamic Interoperability (Level 5)	Ensures that systems can adjust and react to changes instantaneously, such as new software updates, traffic situations, or sensor inputs. It is essential for Over-the-Air (OTA) updates and handling dynamic environments.	5G-NR, TSN, DDS-XRCE, MQTT.	AUTOSAR Adaptive, ISO 21434 (cybersecurity), IEEE 1609.2 (V2X security), IEEE 802.1.	- Adaptive systems can react and compromise compatibility with updated or evolving software (futureproofing SDVs)
Organizational Interoperability (Level 6)	Focuses on alignment between organizations, ensuring collaboration and consistency between different stakeholders (manufacturers, developers, regulators).	API-based systems, RESTful APIs, Open Platforms	UN Regulation No. 157 (Automated Lane Keeping Systems), ISO 26262 (functional safety), UNECE-R155 (Security), UNECE-R156 (Software Update), COVESA, GDPR.	- Coordination across sectors (OEMs, Tier-1 Suppliers, regulators). - Cohesive development and Compliance.

Table 1: Six-Tier interoperability Level Framework for SDVoF.

3.2 Key Challenges

- **Managing the Complexity:** Vehicles consist of diverse components and systems, which often come from various vendors and follow different protocols, making it difficult to ensure compatibility.
- **Longevity and Legacy:** At a certain level, legacy systems must be adapted to new service-oriented architectures to maintain backward compatibility.
- **Real-time communication** between in-vehicle systems, external infrastructure, and cloud platforms is necessary to enable SDVs to function effectively in real-world environments.

4 In-vehicle Communication

4.1 Ethernet and Service-oriented Architectures

In software-defined vehicles, Ethernet plays a central role as the backbone of service-oriented architectures, effectively replacing legacy communication buses like CAN in core network functions [118]. It is increasingly adopted in automotive applications due to its high data rate and flexibility. It is well-suited for modern, data-intensive systems, such as over-the-air (OTA) updates and camera systems in autonomous vehicles. Its scalability and ability to handle various traffic classes, including real-time and best-effort communication, enable the integration of diverse subsystems into a unified network. This reduces the complexity of current heterogeneous in-car networks and supports the growing demand for reliable, high-bandwidth communication, which is essential for advanced driver assistance and infotainment systems [111].

Technologies such as TSN and middleware protocols like SOME/IP and DDS enable deterministic, real-time communication and dynamic service discovery, which are critical for the scalability and flexibility of SDV platforms [74,85,135]. These protocols support publish-subscribe and request-response communication patterns, making them ideal for dynamically reconfigurable and updatable automotive systems. Integrating Ethernet with SDN enhances the network's adaptability, allowing dynamic reconfiguration and centralised management of services at runtime [48]. As vehicles continue to evolve into increasingly complex, distributed computing platforms, Ethernet-based SOA enables the deployment, update, and scaling of software functions independently across ECUs, paving the way for continuous delivery models in automotive software development [136].

4.2 CAN Protocols

The Controller Area Network (CAN) protocol has been the vehicle bus standard since the 80's, enabling communication between microcontrollers and devices without requiring a central host computer. Initially designed for automotive use by Bosch [60], it has since expanded into various industrial applications. The protocol operates on a multi-master, multi-drop network, where any node can initiate communication with another node. Messages are assigned unique priorities via their identifiers, and when two nodes transmit simultaneously, the message with the higher priority (i.e., the lower identifier number) is transmitted first, while the other waits and retries later. This mechanism ensures reliable communication, making CAN suitable for real-time, safety-critical vehicle systems [103]. The physical layer of the CAN bus transmits data over twisted-pair cables, utilising differential signalling to reduce electromagnetic interference and enhance reliability. Each CAN message consists of several fields: the identifier (ID), which determines priority; a control field that specifies data length; the actual data payload; a CRC (Cyclic Redundancy Check) field for error detection; and an acknowledgement field. This frame structure ensures data integrity during transmission [83].

Several key CAN protocols are crucial to the vehicular industry. The CAN 2.0 standard, which includes versions 2.0A and 2.0B, remains foundational in automotive communication, facilitating data exchange between Electronic Control Units (ECUs). CAN 2.0A uses 11-bit identifiers, while CAN 2.0B allows for 29-bit identifiers, ensuring flexibility and compatibility across various systems. These can be used on the same bus as long as no extended frames are sent by controllers using CAN 2.0B [1]. As vehicle systems became increasingly complex, CAN FD (Flexible Data Rate) emerged as a solution, offering higher data transmission speeds and larger payload capacities. This advanced protocol reduces wiring complexity and supports more sophisticated in-vehicle functions, making it a preferred choice for modern automotive systems [124]. CAN FD data rate can reach up to 8 Mbps, and its payload can be extended up to 64 bytes, compared to the traditional 8 bytes in CAN 2.0.

This improvement significantly enhances data handling for applications like advanced driver assistance systems (ADAS), which require fast and reliable communication [29].

4.3 Other In-vehicle Communication Protocols

Other in-vehicle communication protocols have emerged to meet specific needs:

- LIN (Local Interconnect Network) is a low-cost, single-wire protocol designed for simple, low-speed communication tasks such as controlling seat adjustments or window lifts [109].
- FlexRay is a high-speed, time-triggered protocol used in safety-critical applications such as electronic stability control (ESC) and adaptive cruise control [109].
- MOST (Media Oriented Systems Transport) is designed explicitly for infotainment systems, providing high-bandwidth communication for audio, video, and multimedia data [109].

CAN XL represents a further advancement, increasing data capacity to 2048 bytes and offering higher bandwidth to handle the growing data demands of modern automotive applications, such as autonomous driving [66]. The ISO 11898 standards also govern essential aspects of CAN implementation. ISO 11898-2 supports high-speed communication, while ISO 11898-3 ensures fault tolerance in low-speed environments, both of which are important for reliable in-vehicle networking [63]. Another significant development is ISO-TP (ISO 15765-2), which extends CAN's data capabilities to handle larger-scale messages, primarily used in vehicle diagnostics [109]. In heavy-duty vehicles such as trucks and buses, SAE J1939 is widely used for communication and diagnostics, providing a standardised framework for heavy-duty vehicle networks [114].

4.4 Key Challenges

- **Limited bandwidth:** Traditional CAN can struggle to handle the increasing data demands of modern vehicle systems, such as autonomous driving [25].
- **CAN Bus overload:** Increased data transmission risks communication delays, or data loss [129].
- **Security vulnerabilities:** CAN lacks encryption and authentication, making it susceptible to attacks where false data can be injected or manipulated. CAN is also highly vulnerable to DoS (Denial of service) attacks because of its design, which allows dominant bits to override the recessive ones [82].
- **Ensuring compatibility with newer CAN protocols:** Transitioning to CAN FD or CAN XL presents complexities and additional costs when used with legacy CAN systems. The newer protocols improve performance, especially in data rate and flexibility, but their co-existence with older CAN networks may lead to challenges [23,24].

5 Intelligent Systems and Data-Centric Access

The advent of smart vehicles has sparked significant interest in developing intelligent transportation systems (ITS) [7]. In smart cities, ITS is crucial in improving transportation safety, mobility, and environmental sustainability by utilising modern technologies like connected vehicles, autonomous vehicles, and intelligent traffic signals [44]. A critical component of ITS is the road-side infrastructure and the concept of vehicular edge-cloud computing [64, 65]. This section examines the necessity and implications of these technologies, drawing from the existing literature.

5.1 Data-Centric AI/ML Development for SDV

Modern vehicles contain numerous internal sensors that generate vast amounts of data. Commonly used sensors are cameras, LiDAR, radar [49, 132], and thermal cameras [36]. Multi-sensor fusion, deep learning, and other advanced methods can be used to combine data from different sensors [132]. The onboard diagnostic (OBD) system provides information on the inner workings of the vehicle, including details about engine speed and any potential Diagnostic Trouble Codes [104]. External data sources, including but not limited to weather information, maps, and transport management systems, will enable a multitude of applications for holistic traffic situations and operations.

For ML/AI use cases, the data must be appropriate for the task and is often needed in large quantities from various different sensors and situations [35]. Data access usually involves connecting to the sensor, filtering, and cleaning the data. Accessing external data sources can be challenging, including varying levels of APIs and understanding and processing the data to a usable format. Various pre-processing, anonymisation, and cleaning procedures must be instituted [35]. Data collection may also be necessary if no suitable pre-made datasets are available, in which case a robust pipeline for data collection must be in place. Local data management must include how to store the data, which requires space, what data to store and delete, and at what point and what data is sent to the cloud. With the cloud, network latency and connectivity qualifications need to be considered. For ADAS applications, such as object recognition, datasets are already available; however, they do not always provide what is needed. For instance, the datasets focus primarily on images taken with a camera, and there is a lack of freely available data captured with other sensors. The subjects, situations, or the environment of the captured dataset might also not be what is needed, as common standards for data validation are limited.

For companies already in the vehicular business, access to data can come through their customers and vehicles. For now, many OEMs use different sensing solutions, and data sharing requires standardisation of data formats [86]. Sharing data raises security concerns about data collection and, most importantly, the need for effective privacy preservation solutions. If data collection can take considerable time because of the need to drive the test vehicle(s) [89]. These datasets may need to be revised when new sensors become available on the market [89]. It is possible to use simulations and digital twins to collect data, as they are a safe and scalable option for testing. However, it remains unsolved how representative the data they provide [89] is of real life.

Thus, using previously collected datasets or possibly finding openly available datasets can be a vast resource saver, especially for researchers or other developers who do not have access to manufacturer-collected datasets. Publicly sharing datasets could reap benefits for all automotive ML/AI developers if more effort could be spent on application development instead of repeatedly collecting the same type of datasets. The issue then becomes data (pre)processing and management. Then, it needs to be modelled and communicated so

that the data users know what the dataset contains. If the dataset is shared, an appropriate venue must be chosen. There are already several dataset-sharing sites, such as Roboflow, Hugging Face, and Kaggle. However, a dedicated data market or store is likely needed. This way, it would be more accessible to find datasets explicitly dedicated to this environment, and rules on dataset use could be established.

5.2 Model Development and Training

Model development and training can be challenging due to the size of the model. According to estimates from 2022, several models were published with more than 70 billion parameters, and the number had been growing steadily, especially for models in the vision and language categories [121]. For instance, the image recognition models provided by Keras range from 14MB to a whopping 1310MB.

Large size means the training time is longer and requires an increasing amount of computational resources. Training the model in a computing-restricted environment can thus become a challenge. Edge-cloud-continuum can be utilised to provide computing capacity for model training, thereby speeding it up. Machine Learning as-a-Service (MLaaS) and utilising model repositories or marketplaces could be another option for avoiding the high training costs. Transference learning could also be utilised, allowing models created for one task to be finetuned for another. Many MLaaS providers exist, such as Microsoft Azure or Amazon's AWS; however, incorporating the cloud introduces potential issues. Due to the connection to an external service, additional security risks must be considered. The connection itself may become an issue, for instance, in terms of data transfer times if the connectivity is slow or sporadic. The larger the data moving through the connection, the longer it would take, exacerbating the poor connection problem. The applicability of ready-made models for automotive use cases must also be tested. For instance, Amazon's AWS already has a marketplace for pre-made ML models; however, it is unclear without research how well these pre-made models would perform in the vehicular setting. For example, the currently existing and advanced language-related models can be used to understand the driver's speech and commands. In contrast, machine vision applications can differ significantly, for example, in industrial settings and driving scenarios. Vehicular-specific MLaaS and model sharing should be implemented to ensure that all vehicle features operate safely and efficiently.

On the other hand, GPU units have been integrated into vehicles, increasing the processing capacity within them. For instance, Nvidia provides several different system-on-a-chip, which combine CPU and GPU power, such as the NVIDIA DRIVE AGX Thor10. This could enable functionalities such as (re)training ML/AI models inside the vehicle and immediately utilising the data acquired while driving. However, these types of solutions require a functioning data (pre-)processing pipeline to be in place for in-vehicle processing. There are costs associated with these additional chips, which could be reflected in the vehicle's price.

Personalisation of the ML/AI models is a key selling point for driver experience. It could be further explored to make vehicular applications a better fit for their respective drivers. Personalisation of the ML model trains the model to fit a particular individual, which can be achieved by using a dataset tailored to that individual, along with a global dataset [105]. Example use cases could include personalised route recommendations that consider the driver's preferences, such as the type of scenery or driving conditions they prefer, pit stop frequency and location. Practical examples, however, are still largely missing from the literature, and more work should be done on how to personalise models.

Privacy and Security are key factors when developing vehicular ML/AI models. Privacy must be considered with data collection, for instance, when (re)training the model. An example is the personalisation of the model, where data collection must be done ethically to preserve the right to privacy. A factor here is how much data

and model information is shared outside the vehicle. Security relates to, for instance, data collection from various vehicular sensors and systems, both within and outside the vehicle, as part of the vehicle-edge-cloud continuum. Regulations and privacy rules must be followed when collecting the dataset and creating the models. These vary between countries and legislative systems. However, specific consideration on the AI development in the European market has to be given to the AI Act, the European Regulation on Artificial Intelligence, and the European Cyber Resilience Act (CRA).

5.3 Model Deployment and Inference

Due to the large data sizes and computational capacity required to effectively train ML/AI models, the training and validation are often left to the cloud back-end. On the vehicular side, models are only deployed for inference. In addition to ADAS, vehicular ML examples include user behaviour estimation for electric vehicle (EV) charging behaviour [26] and other driver behaviour evaluation such as evaluating if the driver is distracted [130], anomaly detection [84] [112], securing the in-vehicle network [110] and other cyber-security related tasks such as intrusion detection [73]. In public transport vehicles, it has been proposed to use ML for estimating passenger experience from the vehicular data [64]. Other possible use cases for ML are personalised route recommendations [113], (emotion-aware) personal assistants [123], and context-aware recommendation systems for infotainment [53].

However, some examples of ongoing learning processes on vehicles already exist. Reinforcement Learning (RL) has already been experimented on vehicles, for instance, for cyber-physical safety [34], navigation [61], and decision-making while on the road [133]. The performance of the ML models requires monitoring and possibly feedback to stay current and accurate. This is especially crucial for safety-critical systems, but non-safety-critical features can also benefit from feedback to stay current and improve over time. In general, validation and testing must be performed to monitor the models. Adapting MLOps and AIOps for ML model development for SDVs could also be beneficial. However, these developments usually go hand in hand with other software DevOps processes and should not be considered stand-alone. With a Human-in-the-Loop (HITL-ML) approach, the driver can be integrated into the validation of the learning processes of different applications [33]. Driver feedback can be utilised to improve the functioning of the ML model, for example, by double-checking that the information is timely, services are functioning properly, and the human experience is satisfactory.

5.4 Key Challenges

- **Safety and Reliability Challenge:** ML and AI models, for instance, deep neural networks in SDVs, can act as complex black boxes, making it challenging for engineers to predict or interpret the proper behaviours of these models under complex and diverse conditions, thus raising serious safety concerns [17]. Current automotive functional safety standards do not fully cover all AI-based components. As such, the regulators and manufacturers remain cautious about deploying AI/ML in safety-critical scenarios [78]. Such uncertainty in decision-making makes it challenging to guarantee reliable performance and establish common verification and validation methods for AI-driven vehicle functions [78]. Consequently, comprehensive safety assessments are required before integrating AI/ML systems into production to ensure road-user safety [96].
- **Data and Generalization Challenge:** AI/ML models and systems in SDVs are often highly demanding in terms of data quality, quantity, and diversity. High-quality data availability is another challenge, for example, in data gathering and labelling. Acquiring quality and comprehensive datasets is costly, yet still prone to errors [96]. Furthermore, AI/ML models trained on limited scenarios and datasets can

introduce bias and fail to generalise. On top of that, real-world driving is affected by complex, diverse and even unexpected factors and situations (e.g., changing weather, lighting, or sensor conditions) that can degrade an AI model's performance when over the training data [6].

- **Real-Time Computation Challenge:** Secondly, state-of-the-art AI/ML algorithms for perception and decision-making, such as advanced vision models or deep reinforcement learning, are computationally resource-demanding, which leads to another challenging context that, given the real-time and real-world driving constraints on real-time responsiveness [6]. SDVs must process sensor inputs, run AI/ML inference, and execute control under millisecond limits. Conversely, the current advanced or complex models, although promising higher accuracy potential, often come at the cost of higher latency and computational load [17]. Therefore, ensuring the AI/ML models run reliably within the embedded systems of the SDVs (limited power, constrained processing and computation resources) stands out as another significant challenge to tackle [6]

6 Beyond the Vehicle: Software-defined Mobility Platforms

The advent of smart vehicles has sparked significant interest in developing intelligent transportation systems (ITS) [9]. In smart cities, ITS is crucial in improving transportation safety, mobility, and environmental sustainability by utilising modern technologies like connected vehicles, autonomous vehicles, and intelligent traffic signals [59]. A critical component of ITS is the roadside infrastructure, as well as the concept of vehicular edge-cloud computing [91,92]. This section examines the necessity and implications of these technologies, drawing from the existing literature.

6.1 Roadside Infrastructure

Roadside infrastructure refers to the various sensors, communication devices, and computing resources installed along roadways [102]. These include roadside units (RSUs), which facilitate V2X communication, and various sensors that monitor traffic flow, environmental conditions, and infrastructure health. Intelligent roadside units, equipped with advanced sensors and communication modules, enable smart traffic control, environment perception, and vehicle-to-everything communication [5]. Roadside infrastructure and edge-cloud continuum support the seamless operation of smart vehicles by providing real-time data and connectivity [108], expanding vehicular data capabilities from single-vehicle applications to connected, resourceful applications.

Roadside infrastructure is crucial in enhancing road safety by facilitating real-time communication between vehicles and their surroundings. For example, RSUs can relay information about road hazards, traffic conditions, or weather changes to approaching vehicles, allowing them to adjust their behaviour accordingly. This infrastructure also facilitates traffic management by providing authorities with real-time data on traffic flow, which can be used to optimise traffic signals and reduce congestion. Additionally, while smart vehicles are equipped with a suite of sensors, there are limitations to what onboard sensors can detect, especially in challenging environments like urban canyons or during adverse weather conditions. Roadside sensors can augment vehicle sensing capabilities by providing a broader and more accurate picture of the surroundings, given that they are placed optimally to balance cost, redundancy, and coverage [120]. Moreover, roadside infrastructure enables cooperative driving, where multiple vehicles coordinate their actions to improve traffic flow and safety. This is particularly important for technologies such as platooning, where vehicles travel in close proximity at high speeds. RSU relaying enhances the performance of vehicle platooning by reducing communication link failures and inter-vehicle distances [39].

Despite the advantages mentioned, RSUs also have a couple of disadvantages. Firstly, deploying roadside infrastructure is capital-intensive, requiring significant investment in hardware and software. Maintenance costs are also substantial, particularly in urban areas where infrastructure is more prone to damage and wear. These costs can hinder widespread adoption, especially in developing regions [45]. Furthermore, as the number of smart vehicles increases, the demand for roadside infrastructure is expected to grow accordingly. Ensuring the infrastructure can scale to accommodate millions of connected vehicles is a significant technical and economic challenge. Furthermore, roadside infrastructure is a potential target for cyber attacks, which could disrupt ITS operations [30,122]. Ensuring robust cybersecurity measures is critical, but this adds to the complexity and cost of deployment.

6.2 Vehicular Edge-Cloud Continuum

On the other hand, vehicular edge computing can be considered as a complementary approach to roadside infrastructure. Vehicular edge computing (VEC) involves offloading computational tasks from the vehicle to nearby edge servers [77], typically located within the roadside infrastructure, such as base stations. This approach reduces latency [71] and conserves onboard computing resources, making it a key enabler of real-time applications in smart vehicles. Moreover, VEC allows vehicles to offload intensive computational tasks, such as image processing or AI-based decision-making, to edge servers. This conserves vehicle onboard resources and enables more complex applications that would otherwise be infeasible. Additionally, VEC can be scaled more easily than centralised cloud computing because it leverages distributed resources. Each edge server serves a localised area, reducing the overall burden on the network and making it easier to manage traffic spikes.

Similar to roadside infrastructure, some cons are associated with vehicular edge computing. Deploying edge-cloud computing resources is expensive [87]. The cost includes not just the edge servers themselves but also the necessary networking equipment and power supply. Furthermore, VEC introduces additional security risks, particularly in terms of data integrity and privacy [32]. Since data is processed at multiple edge nodes, there is an increased risk of interception or tampering, making it critical to implement robust security protocols. Besides, ensuring that different edge computing nodes can work together seamlessly is a significant challenge [95]. For example, a smart city with a dense network of autonomous vehicles, all connected through edge computing nodes to ensure real-time navigation, traffic updates, and safety monitoring, etc., will suffer from scalability if the volume of the exchanged data increases exponentially, and latency will also increase if various requests from vehicles overload edge nodes. If the edge nodes are not scalable or poorly coordinated, they may struggle to handle the surge in data and consequently experience increased latency. This requires standardisation across hardware and software platforms, which can be challenging given the diversity of stakeholders involved.

6.3 Future Developments

The necessity of roadside infrastructure and vehicular edge-cloud continuum computing depends mainly on the specific applications and the scale of the ITS deployment. The benefits of roadside infrastructure and VEC are clear for high-density urban areas, where real-time traffic management and safety are paramount. They provide the necessary support for advanced applications such as cooperative driving and real-time traffic optimisation. However, in less congested or rural areas, the cost and complexity of deploying such infrastructure may outweigh the benefits, especially if vehicles are equipped with sufficiently advanced onboard systems.

Combining robust onboard vehicle systems with selective deployment of roadside infrastructure and vehicular edge-cloud continuum may offer the most practical and cost-effective solution. This would allow for the scalability and flexibility needed to accommodate a wide range of environments and use cases, ensuring that the benefits of smart vehicles are realised without imposing prohibitive costs. While roadside infrastructure and vehicular edge-cloud computing are not universally necessary, they are critical enablers of advanced ITS applications, particularly in high-density or safety-critical scenarios. Their deployment should be strategically planned, considering the immediate benefits and long-term sustainability of the ITS ecosystem.

6.4 Key Challenges

- **Architectural Challenges: Novel** functionalities and rapid technological advancement in vehicles require more modular and flexible architectures that are not currently fully supported on the roadside and edge-cloud infrastructure.
- **Real-time Challenges:** Required real-time computing presents complexities in ensuring secure and reliable systems across diverse systems and platforms.
- **Standardisation Challenges: Standardisation** across OEMs and regions in the edge-cloud continuum for vehicular computing is essential for ensuring interoperability, scalability, and security in a globally connected automotive ecosystem. It enables seamless communication between vehicles, cloud platforms, and infrastructure by establishing common protocols and data formats, reducing fragmentation and promoting compatibility.

7 V2X Communications

In the venture towards connected SDVs, reliable wireless V2X connectivity becomes useful to ensure real-time data communications in dynamic SDVs. Moreover, developing a middleware layer (see Figure 1) that can abstract the differences between the various communication standards can enable a unified software interface for V2X services and standardised message formats. Moreover, such a middleware layer can assist SDVs in connecting to the relevant network standard based on network availability and environmental conditions, thereby facilitating efficiency, interoperability, and consistency.

7.1 Vehicular Communication Standardization

The current standardisation focuses on addressing the general challenges in V2X-assisted driving, including achieving the timing and end-to-end communication latency, testing methodologies, network resource allocation, integrating network infrastructure, and maintaining transparency and trustworthiness in communications regarding safety, security, reliability, and resilience. In this regard, minimising communication latency and timing is essential, especially in time-critical scenarios. As per 6G network standards, the maximum latencies for critical applications should be under 1 ms, while those for information-sharing applications should be under 30 ms. Additionally, data rates must exceed 1 Gbps for 5G and 100 Gbps for 6G in V2X applications.

Feature	3GPP C-V2X	ETSI ITS-G5	DSRC
Communication Modes	Direct (PC5) and Network-based (Uu). Supports V2V, V2I, V2N, V2P.	Direct communication based on IEEE 802.11p. Supports V2V and V2I.	Direct communication based on IEEE 802.11p. Supports V2V and V2I.
Frequency Range	5.9 GHz (PC5) with licensed bands with flexible channel allocation.	5.9 GHz (unlicensed) with fixed 10 MHz channels.	5.9 GHz (unlicensed) with fixed 10 MHz channels.
Data Security and Privacy	Built-in encryption and authentication via LTE/5G protocols (SIM or credentials). Pseudonyms supported.	Secured via PKI-based mechanisms under the European ITS framework.	Secured via PKI-based mechanisms following IEEE 1609.2 standards.
Latency and Range	Latency: 1 ms (PC5), 10-20 ms (Uu). Range: Up to 1 km (direct).	Latency: <10 ms. Range: 300 m.	Latency: <10 ms. Range: 300 m.
Message Types	BSM, CAM, DENM, advanced cooperative perception messages in Rel-16/17.	BSM, CAM, and DENM. Limited support for advanced services.	BSM, CAM, and DENM. Limited support for advanced services.
Edge and Cloud Integration	Continuous integration with edge computing (5G MEC) and cloud. Dynamic updates for SDVs.	Limited edge integration. Requires external cloud connectivity.	Limited edge integration. Requires external cloud connectivity.
Scalability	High due to leveraging cellular infrastructure.	Moderate with fixed ITS infrastructure.	Moderate with fixed ITS infrastructure.
Adoption Regions	Global adoption with rapid advancements in 5G NR-V2X.	Mainly in Europe and Asia.	Primarily in the U.S. and Japan.

Table 2: Comparison of Vehicular Communication Standards (3GPP C-V2X, ETSI ITS-G5, and DSRC).

Vehicular communication standards, such as 3GPP C-V2X, ETSI ITS-G5, and DSRC, enable OEMs to adopt uniform standards for their communication interfaces. A comparison between the major V2X communication standards in terms of data security, encryption, authentication, communication modes, and frequency ranges is given in Table 2. Concerning the V2X communication standardisation, 3GPP C-V2X supports both direct

communication via the PC5 interface and network-based communication via the Uu interface. Further, C-V2X is considered the most adaptable standard for SDV due to its features that support integration with cloud and edge platforms. On the other hand, DSRC and ETSI ITS-G5 are primarily suited for localised safety applications using V2V and V2I, but their scalability and integration potential with SDVs are currently limited compared to C-V2X. Further, C-V2X offers greater flexibility in frequency allocation: C-V2X operates in the 5.9 GHz band (PC5) but utilises licensed bands and flexible channel allocation. On the other hand, both ETSI ITS-G5 and DSRC use the 5.9 GHz band with fixed 10 MHz channels, limiting their flexibility.

In terms of security, C-V2X includes built-in encryption and authentication through LTE/5G protocols, offering data security and privacy. ETSI ITS-G5 and DSRC rely on Public Key Infrastructure (PKI)-based mechanisms for security. Moreover, C-V2X exhibits a higher latency and range than ITS-G5 or DSRC. C-V2X achieves a latency of 1 ms (PC5) and 10-20 ms (Uu), with a range of up to 1 km (direct). ETSI ITS-G5 and DSRC have similar latencies of less than 10 ms, but are limited to a range of 300 m. C-V2X can be integrated with edge computing (5G MEC) and cloud platforms, supporting dynamic updates for SDVs. ETSI ITS-G5 and DSRC have limited edge integration and require external cloud connectivity. In terms of scalability, C-V2X leverages cellular infrastructure, resulting in higher scalability. ETSI ITS-G5 and DSRC have moderate scalability with fixed ITS infrastructure.

C-V2X supports a wider range of messages, including basic safety messages (BSM), cooperative awareness messages (CAM), decentralised environmental notification messages (DENM), and advanced cooperative perception messages in Rel-16/17. ETSI ITS-G5 and DSRC primarily support BSM, CAM, and DENM, with limited support for advanced services. In summary, C-V2X can be considered the communication standard of choice to support the needs of connected SDVs due to its features related to range, latency, security, and scalability, compared to DSRC and ETSI ITS-G5. These standardised message definitions are critical for interoperability, yet real-world deployment continues to face hurdles such as message security, transmission delays, and cross-technology compatibility [19].

7.2 Towards Large-scale V2X Deployments

Europe is at an early stage of commercial V2X deployment, and a significant challenge toward large-scale deployments has been the lack of consensus among industry players regarding the usage of a specific standard for communication. Nonetheless, V2X hardware providers, software providers, and cybersecurity companies have developed solutions compatible with several protocols, allowing deployment without compatibility concerns. While this hybrid approach can temporarily address interoperability issues, it is not a sustainable long-term solution, and many experts predict that one protocol will eventually prevail. For example, North America and China are expected to primarily converge on C-V2X by 2023, phasing out DSRC. In contrast, in Europe, OEMs such as Volkswagen use DSRC, while BMW and Daimler support C-V2X [12].

7.3 Key Challenges

- **Standardisation and Development Challenges:** There is a lack of consensus on the V2X communication protocols, leading to fragmentation that may slow progress towards agreement on architectures and hinder application development.
- **Compatibility and Hardware Challenges:** Achieving compatibility between multiple communication standards at the middleware layer is a significant technical challenge, which requires advanced hardware and further development. Large-scale deployments will require substantial investments in roadside infrastructure, V2X-capable transmitters and receivers, and compatibility with existing radio networks.

8 Validation and Verification

8.1 SOA and Temporal Properties

A service-oriented architecture (including microservices) is desirable for building flexible, modular systems where independent services interact to perform various functions. For software-defined vehicles, a prime example of service-oriented architectures would start with the data capture of each sensor, followed by the fusion of this data into a comprehensive representation of the vehicle and its environment. Depending on the conclusions drawn, the available information is transformed into actuator inputs, primarily steering angle and vehicle velocity. To implement the example above using a service-oriented architecture, the data (streams) must be carefully defined to reflect the nature of the underlying phenomena; furthermore, the transforming functions (services) must be composed of manageably sized subroutines. In this process, complex logic and interactions emerge due to subroutine invocations that direct data downstream while maintaining the required temporal system properties.

Temporal properties are critical when verifying and validating vehicular and transportation systems. A vehicle's response depends significantly on its intent, combined with the state of its environment. An example could be stopping a moving vehicle to let a passenger get out. Depending on whether the vehicle is currently travelling on the highway or on a rural road, we expect the vehicle's behaviour to correspond to its environmental state.

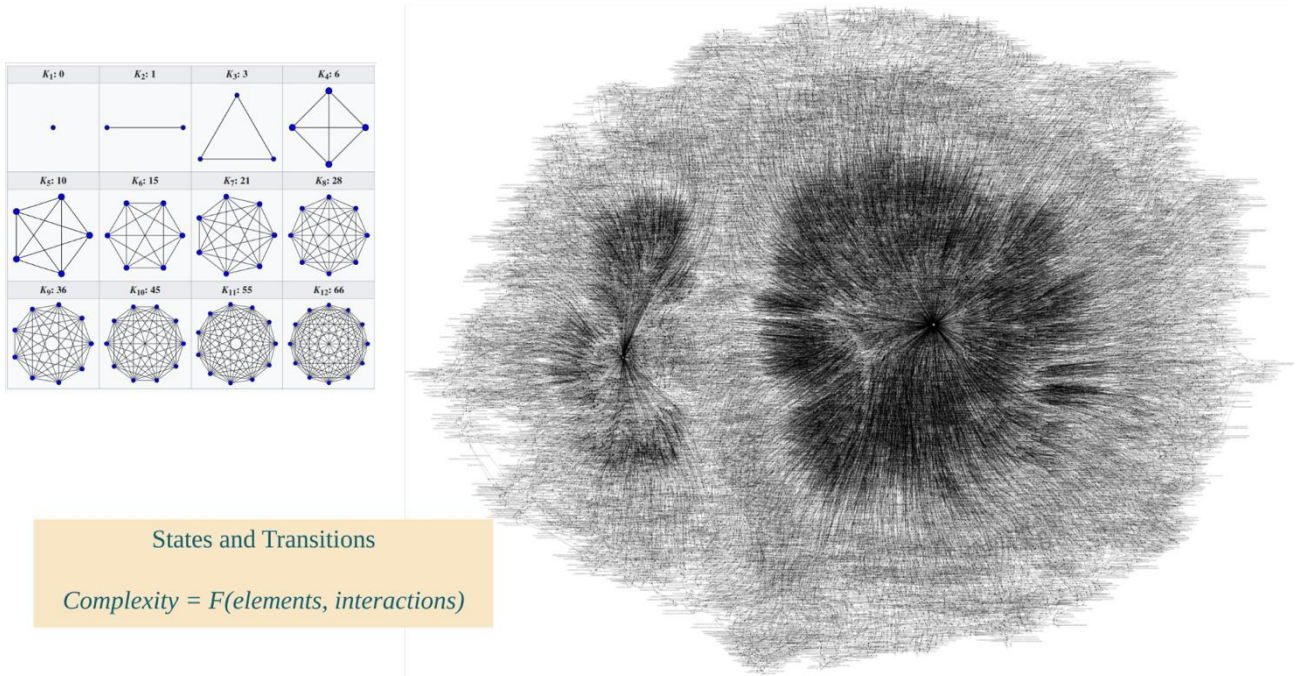

Figure 3: Visualising complexity in validation and verification processes. The image on the right represents a complex state-transition graph where each node corresponds to a system state, and each line represents a transition between states. The dense structure highlights the intricate interactions between the states, illustrating how the complexity increases as the number of elements and interactions grows.

8.2 Engineering Complexity: Specifying Validity and Implementing Verifiability

If we ignore the fact that enumerating all these possible environmental conditions is next to impossible (see Figure 3), using conventional requirement engineering, it is even less likely that one can assert whether the goal for each condition is met using traditional development methods. Where the former relates to requirements gathering and specification, the latter refers to the system engineering aspect of Validation and Verification (V&V). The status quo concerning V&V is primarily based on reviewing requirements documents (validation) and testing (verification). This is insufficient to achieve an acceptable level of functional safety and security for the SDV.

8.3 Current Methods and State of the Art (SOTA) Approaches for V&V in SOA

This chapter explores current methods and state-of-the-art approaches for validation and verification in the context of Service-Oriented Architecture (SOA). These methods, individually or in combination, provide a foundation for automating V&V processes in complex, interdependent, distributed systems while addressing functional, safety, and timing requirements.

- **Formal Modelling Languages:** An example is Verum Dezyne which develops a language for specifying, designing, and implementing requirements, with automated verification for completeness and correctness. Its key feature is stateful compositionality, ideal for systems like software-defined vehicles. Dezyne defines protocols and interactions through interfaces, ensuring safety and security. Components are formally verified using the mCRL2 model checker. Dezyne comes with built-in code generators. Ongoing work focuses on module specifications and integrating data for verification. Dezyne enhances validation through executable specifications, enabling system simulations and bug detection.
- **Model-Based Design (MBD)**: Model-Based Design is widely used in the development of embedded and control systems, offering visual modelling tools such as MATLAB Simulink and IBM Rational Rhapsody. MBD supports simulation and automatic code generation, allowing for iterative testing of the system's functionality before implementation. In the context of SOA, MBD helps validate the interaction between services and the timing constraints that are critical for the system's real-time operations.
- **Timed Automata and Formal Verification of Temporal Logic:** For SOA systems, particularly in safety-critical applications like autonomous vehicles, ensuring that services meet strict timing constraints is vital. Timed automata are used to model real-time systems, and tools such as UPPAAL enable formal verification of temporal properties. These methods provide guarantees that services not only function correctly but also adhere to timing requirements, a critical factor in real-time SOA applications.
- **Contract-Based Design and Verification:** Contract-based design is a method used to specify formal agreements between different services in an SOA. These contracts define the expected inputs, outputs, and behaviours for each service. Using formal contracts ensures that services behave as expected and fulfil their roles in the system's overall functionality. Tools such as OCRA (Othello Contract Refinement Analysis) allow developers to specify and verify the compliance of service contracts, ensuring consistency and correctness throughout the system. This approach is particularly useful in verifying that independently developed services meet predefined specifications when integrated into a larger SOA, supporting modularity and reducing integration risks.

8.4 Virtual Simulation and Digital Twins

Virtual simulation is a key technology that should contribute to reducing the complexity of verifying and validating SDV products. Through virtualised driving environments, risk-critical safety simulations can be performed in a realistic manner without the need for extensive and expensive mechanical testing. Various open-source and commercial simulations for different purposes have been developed [69]; some of them focus extensively on sensor data flow, traffic conditions, vehicle dynamics, or control policies, decision-making, and control logic of autonomous vehicles. Probably the best-known and widely used, comprehensive simulator is CARLA [31], an open-source toolbox that provides urban layouts, buildings, vehicles, and maps.

However, such simulation environments still lack support for integrating real-life data into the simulation pipeline, which is necessary to address data fidelity in diverse real-world scenarios [69]. To enable effective real-life data modelling, simulators should integrate the intake of recorded data from various fleets, data management pipelines, and accessible data formats for representation. In addition, in-depth testing of software components, particularly different ECUs, requires consideration of both hardware and software viability. One such initiative to bring hardware and software testing into the same loop is Eclipse openDuT, which aims particularly at grey-box tests for single ECUs or clusters of ECUs, as well as the adaptation and full functional integration of third-party components (proprietary and private source). These features are critical for even more realistic simulations in industrial use.

Considerations have also been taken to adopt digital twinning technology for the vehicular use case [42]. Digital twins are well-known in other industrial sectors, particularly in manufacturing, and are capable of processing different scenarios based on real-world data flows. For now, most work has focused on utilising digital twinning technologies on vehicular design, evaluation, and deployment. However, for the complex and comprehensive verification and validation purposes of vehicular systems, even the current digital twins require greater maturity and adaptability in terms of safety, accuracy, and fidelity [107].

8.5 Continuous Testing and Validation

The long lifespan of vehicles, spanning up to decades, necessitates special attention from a software engineering perspective for continuous testing, validation, and verification practices. Specific challenges identified for vehicular software development include, but are not limited to, integrating software and various hardware components, ensuring compatibility and code reusability across complex ECU architectures, and the safety-critical nature of all software components on vehicles, which necessitates reliability and security assurance [80]. In addition, covering the full software lifecycle, supported by Over-the-Air (OTA) software updates, remains one of the key challenges for SDV development [44]. As such, continuous testing of software and hardware components should be integrated into the architecture design and software engineering pipelines to comprehensively cover the full lifecycle of the vehicle.

8.6 Key challenges

- **Full Automation Requirements:** The V&V process must become fully automated by removing manual test execution and definition.
- **Early Specifications:** Complete and comprehensive early specifications should be produced from the constituent part specifications, allowing early validation and verification throughout the engineering process.

- **Integration of proprietary and open-source components** in the same simulation, validation, and verification pipeline to increase their adaptability into industrial use cases. Such a pipeline should also enable black-box and grey-box validation and verification in a reliable manner, through understanding the components as part of a real vehicle.
- **Integration of simultaneous hardware and software testing** on simulation environments to provide more realistic scenarios for failure, safety, and security testing.
- **Continuous testing, validation, and verification** through the vehicle's lifespan.

9 Building the Software Ecosystems

The SDVoF initiative emphasises the creation of a robust ecosystem, integrating various stakeholders, including OEMs, suppliers, regulators, research institutions, and open-source communities. This ecosystem is structured around non-differentiating building blocks within the vehicle software stack, designed to facilitate collaboration and innovation. By fostering an open ecosystem built on open-source software solutions, the initiative aims to reduce redundant efforts and streamline the development of essential software components, accelerating the time-to-market for new features and improving cost efficiency.

The SDVoF initiative's roadmap and vision document [27] has identified some significant challenges the European automotive industry faces. The shift to autonomous, electric, and connected vehicles leads to increased software complexity, where vehicle code is expected to grow significantly, posing challenges to both development and maintenance. Customers increasingly demand new features and frequent software updates, which in turn increases the pressure for continuous innovation. Large tech companies dominate some key areas, creating dependencies for traditional OEMs. Additionally, non-EU manufacturers, such as Tesla, with a software-first approach, are creating competitive pressure. Transitioning from proprietary software and fragmented development initiatives to an open-source ecosystem centred around non-differentiating software solutions is essential for reducing inefficiencies and eliminating duplicated efforts. Collaborative efforts are necessary to establish an open SDV ecosystem and enhance the strategic autonomy of the European automotive industry.

To support the European automotive industry's strategic autonomy, the ecosystem encourages open-source solutions and reduces dependence on non-European technology providers. By fostering collaboration and leveraging shared resources, the industry can streamline its development processes and strengthen its global competitive position. To achieve this, it is crucial to connect existing initiatives and partners to avoid duplicating efforts and ensure the utilisation of existing building blocks and design patterns [27]:

- AUTOSAR: A global partnership that develops standardised software frameworks and system architectures for intelligent mobility.
- COVESA (Connected Vehicle Systems Alliance): Focuses on accelerating the potential of connected vehicles through collaboration.
- SOAFEE (Scalable Open Architecture for Embedded Edge): An industry-led collaboration to create open-source architecture for software-defined vehicles.
- Eclipse SDV: Provides an open technology platform for the software-defined vehicle to accelerate automotive software innovation through open-source communities.
- Eclipse Arrowhead: Provides a robust framework based on Service Oriented Architecture (SOA) and microservices, which facilitates seamless interaction by enabling scalable and interoperable automation solutions.
- Eclipse S-CORE (Establishing an Open Source Ecosystem for Software-Defined Vehicles): Modular, scalable, and standardised software architecture proposed by the Eclipse Foundation.

Automotive Industry Associations:

- ANFIA (Italian Association of the Automotive Industry): Represents automotive component manufacturers and other related industries in Italy.
- PFA (Automotive Platform in France): Brings together the French automotive industry to implement sector strategies.
- VDA (German Association of the Automotive Industry): Works on establishing the right conditions for automotive companies in Germany.

- EUCAR (European Council for Automotive R&D): Coordinates precompetitive research and development projects for major European automotive manufacturers.
- CLEPA (European Association of Automotive Suppliers): Represents companies supplying components and technology for safe, smart, and sustainable mobility.

However, transitioning from proprietary, siloed value chains to a robust ecosystem, connecting multiple partners with various interests and goals, is challenging. In traditional value chains, interactions and data flows are linear and controlled within predefined organisational boundaries. However, complexity increases as the scope shifts to an ecosystem with multiple actors contributing in a more open and interconnected environment. Standardising data acquisition and exchange across diverse participants becomes critical. [125] Additionally, governance becomes more complicated, extending beyond organisational boundaries [45]. Issues such as intellectual property rights (IPRs) related to data analytics and processing become increasingly pronounced, necessitating clear agreements and robust governance structures [125].

Engaging stakeholders in an ecosystem is crucial, as their commitment is necessary to achieve successful outcomes. If their internal priorities and needs are not addressed, there is a risk that critical actors may withdraw from the collaborative effort, endangering the entire initiative [46]. Partners must be open to collaborating and integrating into the ecosystem in a way that aligns with their own goals [112]. The ecosystem must be attractive enough to engage a critical mass of partners for effective outcomes [46]. Effective coordination of interrelated organisations with significant autonomy is central to ecosystems [45]. This coordination is achieved through processes, rules, and standards that help resolve issues and ensure alignment among participants [50]. Governance is necessary to manage the balance between fostering innovation and maintaining the ecosystem's overall health, guiding development, and ensuring that collective efforts contribute positively to market demands [121].

9.1 Key Challenges

- **Diverse Stakeholders**: Achieving seamless collaboration between diverse stakeholders, including manufacturers and regulators, is essential to ensure cohesive development and integration of systems. This includes understanding the diverse needs and perspectives of various stakeholders and fostering effective collaboration towards shared goals.
- **Long-term Engagement:** Engaging stakeholders in an ecosystem is crucial, as their commitment is necessary to achieve successful outcomes. This includes ensuring the longevity of publicly financed initiatives and projects, as well as integrating collaboration activities into everyday work and business operations for the long term.
- **Ecosystem Governance** is necessary to manage the balance between fostering innovation and maintaining the ecosystem's overall health. Without a reliable governance strategy, the ecosystem will again become siloed and fragmented, hindering the progress of established collaboration and innovation.

10 Transformation Path

Transforming towards an SDV ecosystem requires an approach involving various stakeholders, developers, and others in the development pipeline [112,125]. This transformation shifts from monolithic software stacks to dynamic systems using microservices and service-oriented architectures, impacting the entire product life cycle. The focus must be on both technical and societal aspects.

The automotive industry is currently siloed, with large manufacturers managing both mechanical and software components. The transformation path must encompass vehicular software development, life cycle, and developer communities across all vehicles and manufacturers. The classic V model of software engineering process [96] is insufficient, necessitating changes in processes, tools, business models, value chains, and working methods. Responsibilities are shifting, requiring cooperation among development, operation, maintenance, and service providers. An SDV is a system of systems that involves parallel work and constant dialogue, requiring new skills and approaches.

To implement these changes more efficiently, it makes sense to form communities and work together. Through a joint approach, whether in classic software development or by sharing new or existing software, many aspects can be simplified, become more cost-effective, and achieve higher quality. By involving multiple development partners, diverse perspectives and insights can be brought to projects, enhancing the overall outcome. This approach requires the participation of a variety of individuals involved in the value chain, from CEOs, who make a corresponding commitment and ensure the support and provision of resources, to the developers, who no longer work in isolated offices on individual solutions, but instead collaborate with partners from other departments, companies, or even industries to develop open, shared solutions. However, this transformation cannot happen overnight and must be integrated harmoniously into existing workflows, building on and supplementing current solutions. Successful transformation leads to healthy ecosystems, fostering innovation and new market entrants [121, 125].

Key aspects of the transformation path include identifying the defining characteristics of automotive software development pipelines and communities, learning from digital transformations in other industries, and involving diverse stakeholders in creating a roadmap. Activities may include training, community-building, and stakeholder engagement, with success measured by community engagement, software adaptation, product quality, and interoperability.

Effective governance is essential, starting with identifying and engaging ecosystem actors and aligning their goals with the ecosystem vision [45, 46]. A governance structure integrating centralised, decentralised, and group-level elements will manage interactions, enhance collaboration, and promote innovation [45]. Evaluating performance across boundaries supports collaboration and integration, ensuring collective efforts align with common goals while respecting individual business interests. This governance aims to minimise duplicated efforts, optimise development, and enhance cost efficiency.

11 Conclusions

In this gap analysis and technology forecast report, we have highlighted the key challenges that must be addressed to ensure the successful outcomes and continuity of the European Software-Defined Vehicle Initiative's future endeavours. In addition to technical gaps, there are those related to ecosystem maintenance, governance, and health that are especially critical for long-term success. This report will be updated during the FEDERATE project as outcomes from the research, innovation, and development actions come through.

References

[1] Kvaser AB. The can bus protocol tutorial, 2023. Accessed: 2024-07-30.
[2] Sarthak Acharya, Arif Ali Khan, and Tero Päivärinta. Interoperability levels and challenges of digital twins in cyber-physical systems. Journal of Industrial Information Integration, page 100714, 2024.
[3] Sarthak Acharya, Aparajita Tripathy, Juho Alatalo, Pekka Seppänen, Aki Lamponen, Jukka Säkkinen, and Tero Päivärinta. Interoperability challenges and opportunities in vehicle-in-the-loop testings: Insights from nuve lab's hybrid setup. Scandinavian Simulation Society, pages 339–347, 2025.
[4] Sarthak Acharya, Oskar Wintercorn, Aparajta Tripathy, Muhammad Hanif, Jan Van Deventer, and Tero Päivärinta. Twins interoperability through service oriented architecture: A use-case of industry 4.0. In Proceedings of the Annual Symposium of Computer Science 2023 co-located with The International Conference on Evaluation and Assessment in Software Engineering (EASE 2023). R. Piskac c/o Redaktion Sun SITE, Informatik V, RWTH Aachen, 2023.
[5] Shiva Agrawal, Rui Song, Kristina Doycheva, Alois Knoll, and Gordon Elger. Intelligent roadside infrastructure for connected mobility. In SMARTGREENS/VEHITS, 2022.
[6] M. Ahmed, R. Iqbal, S. Amin, O. Alhabshneh, and A. Garba. Autonomous vehicle and its adoption: Challenges, opportunities, and future implications. In 2022 International Conference on Emerging Trends in Computing and Engineering Applications (ETCEA), pages 1–6, Karak, Jordan, 2022.
[7] Ahmad Alhilal, Tristan Braud, and Pan Hui. Distributed vehicular computing at the dawn of 5g: a survey. arXiv:2001.07077, 2020.
[8] Belal Ali, Mark A. Gregory, and Shuo Li. Multi-access edge computing architecture, data security and privacy: A review. IEEE Access, 9:18706–18721, 2021.
[9] Insha Altaf and Ajay Kaul. A survey on autonomous vehicles in the field of intelligent transport system. Studies in Autonomic, Data-driven and Industrial Computing, 2021.
[10] Apex.AI. Autosar and ros 2 for software-defined vehicle, 2022.
[11] Shahin Atakishiyev, Mohammad Salameh, Hengshuai Yao, and Randy Goebel. Explainable artificial intelligence for autonomous driving: A comprehensive overview and field guide for future research directions, 2024.
[12] Autocrypt. The v2x deployment roadmap in europe: What to expect by 2024. https://autocrypt.io/v2x-deployment-roadmap-europe-2024/. (Accessed on 08/22/2024).
[13] SBD Automotive. How will the software-defined vehicle impact the automotive industry?, Unknown.
[14] AUTOSAR. Autosar classic platform, 2023. Accessed: 2024-08-22.
[15] AUTOSAR. Adaptive autosar, Unknown. Accessed: 2024-08-22.
[16] AUTOSAR. Autosar classic platform, Unknown. Accessed: 2024-08-22.
[17] D. Bacciu, A. Carta, C. Gallicchio, and C. Schmittner. Safety and robustness for deep neural networks: An automotive use case. In Computer Safety, Reliability, and Security. SAFECOMP 2023 Workshops, volume 14182 of Lecture Notes in Computer Science. Springer, Cham, 2023.
[18] S. Baidya, Y. Ku, H. Zhao, J. Zhao, and S. Dey. Vehicular and edge computing for emerging connected and autonomous vehicle applications. In Proc. of the 57th Design Automation Conference (DAC), 2020.
[19] Vishaka Basnayake, Prabhash Rathnayake, and Ella Peltonen. Vehicle- to-everything services in 3gpp 5g networks: An empirical analysis. In Proceedings of the 8th International Workshop on Edge Systems, Analytics and Networking, pages 25–30, 2025.
[20] Jitendra Bhatia, Yash Modi, Sudeep Tanwar, and Madhuri Bhavsar. Software defined vehicular networks: A comprehensive review. International Journal of Communication Systems, 32(12):e4005, 2019.
[21] Tobias Blaß, Daniel Casini, Sergey Bozhko, and Björn B Brandenburg. A ros 2 response-time analysis exploiting starvation freedom and executiontime variance. In 2021 IEEE Real-Time Systems Symposium (RTSS), pages 41–53. IEEE, 2021.
[22] Daniel Casini, Tobias Blaß, Ingo Lütkebohle, and Björn Brandenburg. Response-time analysis of ros 2 processing chains under reservation-based scheduling. In 31st Euromicro Conference on Real-Time Systems, pages 1–23. Schloss Dagstuhl, 2019.
[23] G. Cena, I. Bertolotti, T. Hu, and A. Valenzano. Improving compatibility between can fd and legacy can devices. 2015 IEEE 1st International Forum on Research and Technologies for Society and Industry Leveraging
a better tomorrow (RTSI), pages 419–426, 2015.
[24] G. Cena, S. Scanzio, and A. Valenzano. Composite can xl-ethernet netorks for next-gen automotive and automation systems. 2023 IEEE 19th International Conference on Factory Communication Systems (WFCS),
pages 1–8, 2023.
[25] Raghu Changalvala, Brandon Fedoruk, and Hafiz Malik. Radar data integrity verification using 2d qim-based data hiding. Sensors (Basel, Switzerland), 20, 2020.
[26] Yu-Wei Chung, Behnam Khaki, Tianyi Li, Chicheng Chu, and Rajit Gadh. Ensemble machine learning-based algorithm for electric vehicle user behavior prediction. Applied Energy, 254:113732, 2019.
[27] FEDERATE Consortium and SDVoF Sherpa Governance Team. European software-defined vehicle of the future (sdvof) initiative – vision and roadmap, April 2024. Accessed: 2024-05-16.
[28] Nada Cvijetic and Tom Tomazin. Developing a centralized compute architecture for autonomous vehicles. ATZ electronics worldwide, 16:10–15,2021.

[29] Ricardo de Andrade, Kleber N. Hodel, J. F. Justo, A. Laganá, M. M. Santos, and Zonghua Gu. Analytical and experimental performance evaluations of can-fd bus. IEEE Access, 6:21287–21295, 2018.
[30] Kheelesh Kumar Dewangan, Vibek Panda, Sunil Ojha, Anjali Shahapure, and Shweta Rajesh Jahagirdar. Cyber threats and its mitigation to intelligent transportation system. SAE Technical Paper Series, 2024.
[31] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Proceedings of the 1st Annual Conference on Robot Learning, pages 1–16, 2017.
[32] Mehmet Ali Eken and Pelin Angin. Vehicular edge computing security.Secure Edge Computing, 2021.
[33] Yousef Emami, Luis Almeida, Kai Li, Wei Ni, and Zhu Han. Human-in-the-loop machine learning for safe and ethical autonomous vehicles: Principles, challenges, and opportunities, 2024.
[34] Aidin Ferdowsi, Ursula Challita, Walid Saad, and Narayan B. Mandayam. Robust deep reinforcement learning for security and safety in autonomous vehicle systems. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pages 307–312, 2018.
[35] Andrea Fieschi, Pascal Hirmer, Rose Sturm, Martin Eisele, and Bernhard Mitschang. Anonymization use cases for data transfer in the automotive domain. In 2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops), pages 98–103, 2023.
[36] D. Forslund and J. Bjärkefur. Night vision animal detection. In 2014 IEEE Intelligent Vehicles Symposium Proceedings, pages 737–742. IEEE, 2014.
[37] Fernando Garcia, David Martin, Arturo De La Escalera, and Jose Maria Armingol. Sensor fusion methodology for vehicle detection. IEEE Int Transportation Systems, 9(1), 2017.
[38] Avik Ghose, Provat Biswas, Chirabrata Bhaumik, Monika Sharma, Arpan Pal, and Abhinav Jha. Road condition monitoring and alert application. In IEEE International Conference on Pervasive Computing and Communications Workshops, pages 489–491, Lugano, Switzerland, 2012. IEEE.
[39] Tiago Rocha Gonçalves, Vineeth Satheeskumar Varma, and Salah-Eddine Elayoubi. Performance of vehicle platooning under different v2x relaying methods. 2021 IEEE 32nd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pages 1018–1023, 2021.
[40] Anna Grimm and Rainer Walz. Current and future roles of the automotive and ict sectoral systems in autonomous driving - using the innovation system approach to assess value chain transformation. Technological Forecasting and Social Change, 188:122990, 2023.
[41] NCC Group. Microservices-based architectures, Unknown.
[42] Jiajie Guo, Muhammad Bilal, Yuying Qiu, Cheng Qian, Xiaolong Xu, and Kim-Kwang Raymond Choo. Survey on digital twins for internet of vehicles: Fundamentals, challenges, and opportunities. Digital Communications and Networks, 10(2):237–247, 2024.
[43] Marco Haeberle, Florian Heimgaertner, Hans Loehr, Naresh Nayak, Dennis Grewe, Sebastian Schildt, and Michael Menth. Softwarization of automotive e/e architectures: A software-defined networking approach. In IEEE Vehicular Networking Conf. (VNC), pages 1–8. IEEE, 2020.
[44] Subir Halder, Amrita Ghosal, and Mauro Conti. Secure over-the-air software updates in connected vehicles: A survey. Computer Networks, 178:107343, 2020.
[45] Jeong-Seok Heo, Byung-Joon Kang, Jin Mo Yang, Jeongyeup Paek, and Saewoong Bahk. Performance-cost tradeoff of using mobile roadside units for v2x communication. IEEE Transactions on Vehicular Technology, 68:9049–9059, 2019.
[46] Heidi Hietala and Tero Päivärinta. Governing collective ambidexterity: Antecedents, mechanisms, and outcomes in digital service ecosystems. Government Information Quarterly, 42(1):102001, 2025.
[47] Heidi Hietala, Tero Päivärinta, Elina Annanperä, and Kari Liukkunen. Collectively ambidextrous digital service ecosystems: a case of bureaucracy of death. ECIS 2023 research papers, 2023.
[48] Timo Häckel, Philipp Meyer, Mehmet Mueller, Jan Schmitt-Solbrig, Franz Korf, and Thomas C. Schmidt. Dynamic service-orientation for software-defined in-vehicle networks. In 2023 IEEE 97th Vehicular Technology Conference (VTC2023-Spring), pages 1–5, 2023.
[49] Henry Alexander Ignatious, Hesham-El-Sayed, and Manzoor Khan. An overview of sensors in autonomous vehicles. Procedia Computer Science, 198:736–741, 2022. 12th International Conference on Emerging Ubiquitous Systems and Pervasive Networks / 11th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare.
[50] Deloitte Insights. Software-defined cars: Industrial revolution on the arrow, Unknown.
[51] Md. Mahmudul Islam, Muhammad Toaha Raza Khan, Malik Muhammad Saad, and Dongkyun Kim. Software-defined vehicular network (sdvn): A survey on architecture and routing. Computer Networks, 185:1–12, 2021.
[52] Michael G Jacobides, Carmelo Cennamo, and Annabelle Gawer. Towards a theory of ecosystems. Strategic management journal, 39(8):2255–2276, 2018.
[53] Jangkyu Ju, Eunju Lee, and SangJun Park. Comparative analysis of ensemble machine learning models for personalized in-vehicle infotainment recommendation systems. In Adjunct Proceedings of the 16th International Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI '24 Adjunct, page 45–50, New York, NY, USA, 2024. ACM
[54] Alexandru Kampmann, Bassam Alrifaee, Markus Kohout, Andreas Wüstenberg, Timo Woopen, Marcus Nolte, Lutz Eckstein, and Stefan Kowalewski. A Dynamic Service-Oriented Software Architecture for Highly Automated Vehicles. In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), pages 2101–2108, 2019.

[55] Alexandru Kampmann, Maximilian Lüer, Stefan Kowalewski, and Bassam Alrifaee. Optimization-based resource allocation for an automotive service-oriented software architecture. In 2022 IEEE Intelligent Vehicles Symposium (IV), pages 678–687. IEEE, 2022.
[56] Ankush Keshar. Enhancing software-defined vehicles with service-oriented architecture: A framework for scalable and modular mobility solutions. IRE Journals of Automotive Systems Engineering, Volume 4 Issue 5, 2020.
[57] G. Keßler, D. Sieben, A. Bhange, and E. Börner. The software defined vehicle – technical and organizational challenges and opportunities. In A.C. Kulzer, H.C. Reuss, and A. Wagner, editors, 23. Internationales Stuttgarter Symposium. ISSYM 2023. Proceedings, pages 414–426. Springer Vieweg, 2023.
[58] Yacine Khaled, Manabu Tsukada, José Santa, and Thierry Ernst. On the design of efficient vehicular applications. In VTC Spring 2009-IEEE 69th Vehicular Technology Conference, pages 1–5. IEEE, 2009.
[59] Samaneh Khazraeian and Mohammed Hadi. Intelligent transportation systems in future smart cities. Studies in Systems, Decision and Control, 2018.
[60] B. Kirk. Using software protocols to mask can bus insecurities. In IEE Colloquium On Electomagnetic Compatibility Of Software, pages 5/1–5/5, 1998.
[61] Songsang Koh, Bo Zhou, Hui Fang, Po Yang, Zaili Yang, Qiang Yang, Lin Guan, and Zhigang Ji. Real-time deep reinforcement learning based vehicle navigation. Applied Soft Computing, 96:106694, 2020.
[62] Stefan Kugele, Philipp Obergfell, Manfred Broy, Oliver Creighton, Matthias Traub, and Wolfgang Hopfensitz. On service-orientation for automotive software. In 2017 IEEE International Conference on Software Architecture (ICSA), pages 193–202. IEEE, 2017.
[63] Kvaser. Can standards. https://kvaser.com/about-can/can-standards/, 2024. Accessed: 24.7.2024.
[64] Timo Laakkoa, Juho Kostiainena, and Olli Pihlajamaaa. Estimating passenger experience from vehicle data: Preconditions for using machine learning. Proceedings of TRA2020, the 8th Transport Research Arena, Liikenne-ja viestintävirasto Traficom, Finland, 2020.
[65] Aurora Labs. The role of ai in software-defined vehicles, Unknown.
[66] Farag Mohamed E. Lagnf and Subramaniam Ganesan. The improved implementation of the message freshness on can xl using fpga. 2022 IEEE International Conference on Electro Information Technology (eIT), pages 215–220, 2022.
[67] Bastian Lampe, Lennart Reiher, Lukas Zanger, Timo Woopen, Raphael van Kempen, and Lutz Eckstein. Robotkube: Orchestrating large-scale co-operative multi-robot systems with kubernetes and ros. In 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), pages 2719–2725, 2023.
[68] Qingquan Li, Long Chen, Ming Li, Shih Lung Shaw, and Andreas Nüchter. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. IEEE Transactions on Vehicular Technology, 63(2):540–555, 2014.
[69] Yueyuan Li, Wei Yuan, Songan Zhang, Weihao Yan, Qiyuan Shen, Chunxiang Wang, and Ming Yang. Choose your simulator wisely: A review on open-source simulators for autonomous driving. IEEE Transactions on Intelligent Vehicles, 9(5):4861–4876, 2024.
[70] Automotive Grade Linux. Automotive grade linux, Unknown.
[71] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. Mobile Networks and Applications, 2020.
[72] Jonas Ljungblad, Bertil Hök, Amin Allalou, and Håkan Pettersson. Passive in-vehicle driver breath alcohol detection using advanced sensor signal acquisition and fusion. Traffic injury prevention, 18, 2017.
[73] George Loukas, Tuan Vuong, Ryan Heartfield, Georgia Sakellari, Yongpil Yoon, and Diane Gan. Cloud-based cyber-physical intrusion detection for vehicles using deep learning. IEEE Access, 6:3491–3508, 2018.
[74] Feng Luo, Yi Ren, Yanhua Yu, Yunpeng Li, and Zitong Wang. A centralized discovery-based method for integrating data distribution service and time-sensitive networking in in-vehicle networks. ArXiv, abs/2409.05904,2024.
[75] Steven Macenski, Tully Foote, Brian Gerkey, Chris Lalancette, and William Woodall. Robot operating system 2: Design, architecture, and uses in the wild. Science Robotics, 7(66):eabm6074, 2022.
[76] Christian Menard, Andrés Goens, Marten Lohstroh, and Jeronimo Castrillon. Achieving determinism in adaptive autosar. In 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), pages 822–827.IEEE, 2020.
[77] Rodolfo Ipolito Meneguette, Robson Eduardo De Grande, Jó Ueyama, Geraldo P. Rocha Filho, and Edmundo Roberto Mauro Madeira. Vehicular edge computing: Architecture, resource management, security, and challenges. ACM Computing Surveys (CSUR), 55:1 – 46, 2021.
[78] Farshad Mirzarazi, Sebelan Danishvar, and Alireza Mousavi. The safety risks of ai-driven solutions in autonomous road vehicles. World Electric Vehicle Journal, 15(10):438, 2024.
[79] Rebeca C Motta, Káthia M de Oliveira, and Guilherme H Travassos. A conceptual perspective on interoperability in context-aware software systems. Information and Software Technology, 114:231–257, 2019.
[80] Lama J Moukahal, Marwa A Elsayed, and Mohammad Zulkernine. Vehicle software engineering (vse): Research and practice. IEEE Internet of Things Journal, 7(10):10137–10149, 2020.
[81] Michael Munz, Mirko Mahlisch, and Klaus Dietmayer. Generic centralized multi sensor data fusion based on probabilistic sensor and environment models for driver assistance systems. IEEE Intelligent Transportation Systems, 2(1), 2010.
[82] Pal-Stefan Murvay and B. Groza. Dos attacks on controller area networks by fault injections from the software layer. Proceedings of the 12th International Conference on Availability, Reliability and Security, 2017.

[83] Pal-Stefan Murvay and B. Groza. Tidal-can: Differential timing based intrusion detection and localization for controller area network. IEEE Access, 8:68895–68912, 2020.
[84] Vishal Vijayan Nair and Boppudi Pranava Koustubh. Data analysis techniques for fault detection in hybrid/electric vehicles. In 2017 IEEE Transportation Electrification Conference (ITEC-India), pages 1–5, 2017.
[85] Teodor-Constantin Nichiţelea and Maria-Geanina Unguritu. Automotive ethernet applications using scalable service-oriented middleware over ip: Service discovery. In 2019 24th International Conference on Methods and Models in Automation and Robotics (MMAR), pages 576–581, 2019.
[86] Jeffrey Nickerson, Kalle Lyttinen, and John L King. Automated Vehicles: A Human/Machine Co-learning Perspective. SAE International, 2022.
[87] Richard Olaniyan, Olamilekan Fadahunsi, Muthucumaru Maheswaran, and Mohamed Faten Zhani. Opportunistic edge computing: Concepts, opportunities and research challenges. ArXiv, abs/1806.04311, 2018.
[88] Alex Oyler. Automotive with aosp & google services, 2022.
[89] Ashish Pandharipande, Chih-Hong Cheng, Justin Dauwels, Sevgi Z. Gurbuz, Javier Ibanez-Guzman, Guofa Li, Andrea Piazzoni, Pu Wang, and Avik Santra. Sensing and machine learning for automotive perception: A review. IEEE Sensors Journal, 23(11):11097–11115, 2023.
[90] Ella Peltonen, Vishaka Basnayake, Olli Timonen, Heidi Hietala, Yueqiang Xu, Sarthak Acharya, Alireza Bakhshi Zadi Mahmoodi, Tero Päivärinta, Bastian Lampe, Mario Driussi, et al. Software-defined vehicle support and coordination project - first technology forecast report. September 2024. Accessed: 2025-03-08.
[91] Ella Peltonen, Mehdi Bennis, Michele Capobianco, Merouane Debbah, Aaron Ding, Felipe Gil-Castiñeira, Marko Jurmu, Teemu Karvonen, Markus Kelanti, Adrian Kliks, Teemu Leppänen, Lauri Lovén, Tommi Mikkonen, Ashwin Rao, Sumudu Samarakoon, Kari Seppänen, Paweł Sroka, Sasu Tarkoma, and Tingting Yang. 6g white paper on edge intelligence. arXiv:2004.14850, 2020.
[92] Ella Peltonen, Arun Sojan, and Tero Päivärinta. Towards real-time learning for edge-cloud continuum with vehicular computing. In 2021 IEEE 7th World Forum on Internet of Things (WF-IoT), pages 921–926. IEEE, 2021.
[93] Android Open Source Project. What is android automotive?, Unknown.
[94] Dominik Püllen. Holistic Security Engineering for Software-Defined Vehicles. PhD thesis, Universität Passau, 2024.
[95] Samuel Rac and Mats Brorsson. At the edge of a seamless cloud experience. ArXiv, abs/2111.06157, 2021.
[96] Q. Rao and J. Frtunikj. Deep learning for self-driving cars: Chances and challenges. In 2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS), pages 35–38, Gothenburg, Sweden, 2018.
[97] Michael Reke, Daniel Peter, Joschua Schulte-Tigges, Stefan Schiffer, Alexander Ferrein, Thomas Walter, and Dominik Matheis. A self-driving car architecture in ros2. In 2020 International SAUPEC/RobMech/PRASA Conference, pages 1–6. IEEE, 2020.
[98] Gregor Resing. The software defined vehicle. https://www.ibm.com/blogs/digitale-perspektive/2023/06/the-software-defined-vehicle/, 2023. Accessed: 2024-08-23.
[99] Reza Rezaei, Thiam-kian Chiew, and Sai-peck Lee. A review of interoperability assessment models. Journal of Zhejiang University SCIENCE C, 14:663–681, 2013.
[100] Marcel Rumez, David Grimm, Ralf Kriesten, and Erich Sax. An overview of automotive service-oriented architectures and implications for security countermeasures. IEEE Access, 8:221852–221870, 2020.
[101] Nayan B Ruparelia. Software development lifecycle models. ACM SIGSOFT Software Engineering Notes, 35(3):8–13, 2010.
[102] Mohammad Ali Salahuddin, Ala Al-Fuqaha, and Mohsen Guizani. Software-defined networking for rsu clouds in support of the internet of vehicles. IEEE Internet of Things Journal, 2(2):133–144, April 2015.
[103] A. A. Salunkhe, Pravin P Kamble, and Rohit Jadhav. Design and implementation of can bus protocol for monitoring vehicle parameters. In 2016 IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), pages 301–304, 2016.
[104] Pooja Rajendra Sawant and Yashwant B Mane. Design and development of on-board diagnostic (obd) device for cars. In 2018 Fourth International Conference on Computing Communication Control and Automation (IC-CUBEA), pages 1–4, 2018.
[105] Johannes Schneider and Michail Vlachos. Personalization of deep learning, 2020.
[106] Brandon Schoettle. Sensor fusion: A comparison of sensing capabilities of human drivers and highly automated vehicles. University of Michigan, 2017.
[107] Chris Schwarz and Ziran Wang. The role of digital twins in connected and automated vehicles. IEEE Intelligent Transportation Systems Magazine, 14(6):41–51, 2022.
[108] Shuyao Shi, Neiwen Ling, Zhehao Jiang, Xuan Huang, Yuze He, Xiaoguang Zhao, Bufang Yang, Chen Bian, Jingfei Xia, Zhenyu Yan, Raymond W. Yeung, and Guoliang Xing. Soar: Design and deployment of a smart roadside infrastructure system for autonomous driving. Proceedings of the 30th Annual International Conference on Mobile Computing and Networking, 2024.
[109] Craig Smith. The Car Hacker's Handbook: A Guide for the Penetration Tester. No Starch Press, San Francisco, 2016.
[110] Hyun Min Song, Jiyoung Woo, and Huy Kang Kim. In-vehicle network intrusion detection using deep convolutional neural network. Vehicular Communications, 21:100198, 2020.
[111] T. Steinbach, Franz Korf, and T. Schmidt. Real-time ethernet for automotive applications: A solution for future in-car networks. 2011 IEEE International Conference on Consumer Electronics -Berlin (ICCE-Berlin), pages 216–220, 2011.
[112] Fei Su and Prashant Goteti. Improving analog functional safety using data-driven anomaly detection. In 2018 IEEE International Test Conference (ITC), pages 1–10, 2018.

[113] Bingrong Sun, Lin Gong, Jisup Shim, Kitae Jang, B. Brian Park, Hongning Wang, and Jia Hu. A human-centric machine learning based personalized route choice prediction in navigation systems. Journal of Intelligent Transportation Systems, 2022.
[114] Embien Technologies. Sae j1939 protocol: An introduction. https://www.embien.com/automotive-insights/sae-j1939-protocol-an-introduction, 2024. Accessed: 17.07.2024.
[115] The Autoware Foundation. Autoware: Open-source software for autonomous driving, 2024.
[116] Jherrod Thomas. Integrating machine learning and ai in automotive safety. International Journal of Innovative Science and Research Technology, 9(1):2302–2305, Jan 2024.
[117] Nirnaya Tripathi, Heidi Hietala, Yueqiang Xu, and Reshani Liyanage. Stakeholders collaborations, challenges and emerging concepts in digital twin ecosystems. Information and Software Technology, page 107424, 2024.
[118] S. P. Vadanan, Balasubramanian Srimukhee, A. Lingam, and D. Prasanna Vadana. Service-oriented network architecture for future automotive networks. pages 1327–1333, 2020.
[119] Raphael Van Kempen, Timo Woopen, and Lutz Eckstein. Unicaragil: Agile development of self-driving vehicles. In Proceedings of the 30th Aachen Colloquium Sustainable Mobility, Aachen, Germany, pages 4–6, 2021.
[120] Roshan Vijay, Jim Cherian, Rachid Riah, Niels de Boer, and Apratim Choudhury. Optimal placement of roadside infrastructure sensors towards safer autonomous vehicle deployments. 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pages 2589–2595, 2021.
[121] Pablo Villalobos, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, Anson Ho, and Marius Hobbhahn. Machine learning model sizes and the parameter gap, 2022.
[122] Skanda Vivek and Charles Harry. Evaluating the strategic consequences of cyber targeting strategies on road transport networks: A case study of washington dc. Int. J. Cyber Warf. Terror., 12:1–14, 2022.
[123] Hans-Jörg Vögel, Christian Süß, Thomas Hubregtsen, Viviane Ghaderi, Ronee Chadowitz, Elisabeth André, Nicholas Cummins, Björn Schuller, Jérôme Härri, Raphaël Troncy, Benoit Huet, Melek Önen, Adlen Ksentini, Jörg Conradt, Asaf Adi, Alexander Zadorojniy, Jacques Terken, Jonas Beskow, Ann Morrison, Kynan Eng, Florian Eyben, Samer Al Moubayed, and Susanne Müller. Emotion-awareness for intelligent vehicle assistants: a research agenda. SEFAIS '18, page 11–15, New York, NY, USA, 2018. ACM
[124] Tushar Waghmare, Sarvesh Tandale, and Prof. B. S. Kadam. Intelligent accidental detection and prevention system for car using can fd protocol. International Journal of Advanced Research in Science, Communication and Technology, 2022.
[125] An Wang, Zili Zha, Yang Guo, and Songqing Chen. Software-defined networking enhanced edge computing: A network-centric survey. Proceedings of the IEEE, 107(8):1500–1519, 2019.
[126] Yan Wang, Jie Yang, Hongbo Liu, Yingying Chen, Marco Gruteser, and Richard P Martin. Sensing vehicle dynamics for determining driver phone use. In Int. conf. on mobile systems, applications, and services, pages 41–54, 2013.
[127] Jonathan Wareham, Paul B Fox, and Josep Lluís Cano Giner. Technology ecosystem governance. Organization science, 25(4):1195–1215, 2014.
[128] Hans Windpassinger. The software defined vehicle - the architecture behind the next evolution of the automotive industry. https://www.ibm.com/blog/the-software-defined-vehicle-the-architecture-behind-the-next-evolution-of-the-automot 2023. Accessed: 2024-08-23.
[129] Yujing Wu and Jin-Gyun Chung. Efficient controller area network data compression for automobile applications. Frontiers of Information Technology & Electronic Engineering, 16:70–78, 2015.
[130] Yang Xing, Chen Lv, Huaji Wang, Dongpu Cao, Efstathios Velenis, and Fei-Yue Wang. Driver activity recognition for intelligent vehicles: A deep learning approach. IEEE Transactions on Vehicular Technology, 68(6):5379–5390, 2019.
[131] Yueqiang Xu, Tero Päivärinta, and Pasi Kuvaja. Digital twins as software and service development ecosystems in industry 4.0: towards a research agenda. In Big Data and Security: First International Conference, ICBDS 2019, Nanjing, China, December 20–22, 2019, Revised Selected Papers 1, pages 51–64. Springer, 2020.
[132] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. Sensors, 21(6), 2021.
[133] Mingfeng Yuan, Jinjun Shan, and Kevin Mi. Deep reinforcement learning based game-theoretic decision-making for autonomous vehicles. IEEE Robotics and Automation Letters, 7(2):818–825, 2022.
[134] Miaomiao Zhang, Yu Teng, Hui Kong, John Baugh, Yu Su, Junri Mi, and Bowen Du. Automatic modelling and verification of autosar architectures. Journal of Systems and Software, 202:111675, 2023.
[135] Zhouping Zhang, Ke Cui, Yuan Zhu, Chenming Jiang, Xiangxi Yao, and Ke Lu. A lightweight implementation of data distribution service (dds) incorporating time sensitive networking (tsn) on the autosar classic platform. In 2024 IEEE Intelligent Vehicles Symposium (IV), pages 3356–3362, 2024.
[136] Xuan Zhou, K. Wang, L. Zhu, Shuai Huang, Guangsheng Han, and Jie Yu. Development of vehicle domain controller based on ethernet. Journal of Physics: Conference Series, 1802, 2021.