FEDERATE

Software-Defined Vehicle Support and Coordination Project

D2.6 First Technology Forecast Report

Authors

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

University of Oulu, Finland

Bastian Lampe

Institute for Automotive Engineering (ika), RWTH Aachen University, Germany

Mario Driussi

Virtual Vehicle Research GmbH, Austria

Karol Kobiela, Rutger van Beusekom, Bert de Jonge

Verum Software Tools B.V.

September 2024

Deliverable	D2.6 – Technology Forecast Report
Work Package(s)	WP2 – WP Technology and high-level requirements solicitation
Dissemination Level	
Due Date	01-10-2024
Actual Submission Date	10-10-2024
WP Leader	VIF
Deliverable Leader	University of Oulu (UOULU)
Contact Person	Ella Peltonen
Email	

Document History			
Revision No.	**Date of the review**	**Name of the reviewer**	**Status of the document** ***(in progress, ready for review, released)***
V0.1	16-09-2024	Ella Peltonen	Ready for review
V0.2	28-09-2024	Peter Priller	Reviewed
V0.3	10-10-2024	Ella Peltonen	Released

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 *6*

Overview *7*

Purpose of this document 7

WP2 partner list 7

1 Introduction *8*

1.1 Software-Defined Vehicle: Towards Programmable Interfaces 8

1.2 Building Blocks for Software-Defined Vehicles 9

2 Software-Defined Vehicle Architectures *11*

2.1 Service-Oriented Architectures (SOA) in Automotive 11

2.2 Mainstream Classic Architecture 11

2.3 Popularized Architectures 12

2.4 Latest Developments and State-of-the-Art Architectures 12

2.5 Vision and Roadmap Towards Microservices 13

2.6 Key Challenges 14

3 Interoperability Aspect in SDVoFs *15*

3.1 Six-Tier Interoperability Framework for SDVs 16

3.2 Key Challenges 16

4 Intelligent Transportation Systems and Vehicular Edge-Cloud Continuum *17*

4.1 Roadside Infrastructure 17

4.2 Vehicular Edge-Cloud Continuum 18

4.3 Future Developments 18

4.4 Key Challenges 19

5 V2X Communications *20*

5.1 Vehicular Communication Standardisation 20

5.2 Towards Large-scale V2X Deployments 21

5.3 Key Challenges 21

6 In-vehicle Communication *22*

6.1 CAN protocols 22

6.2 Other In-vehicle Communication Protocols 22

6.3 Key Challenges 23

7 Validation and Verification *24*

7.1 SOA and temporal properties 24
7.2 Engineering complexity: specifying validity and implementing verifiability 25
7.3 Current Methods and State of the Art (SOTA) Approaches for V&V in SOA 25
7.4 Key challenges 26
8 Building the Software Ecosystems *27*
8.1 Key Challenges 28
9 Transformation Path *29*
10 Conclusions *31*
References *32*

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 [20]. The document addressed the European perspective on the rapidly changing market of the automotive software industry, the key technical challenges, such as the required abstraction of vehicles' hardware components for successful software development, and the need for novel toolchain, 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, we highlight the academic and industrial perspectives on the key building blocks required to be completed to define, implement, and evaluate the software-defined vehicle concept in practice. We focus primarily on areas where significant gaps can be found. 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 an integral part of the SDV development pipeline, as vehicular software will be compatible with real-world complexities.

The automotive industry is facing a significant shift from monoliths to microservices, requiring new architectures, interoperability solutions, and collaborations across various stakeholders. This shift requires flexible, service-oriented architectures, real-time data processing, and robust cybersecurity frameworks to support traffic and vehicular systems' increasingly complex and connected nature. We stress that overcoming technical challenges, such as enabling seamless interoperability between different components, platforms, and services, is essential 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 “D2.6 Technology Forecast Report” M12 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 implement technologies for automatic braking, Cooperative Adaptive Cruise Control, preventing unwanted lane crossing, suggestions for charging stations, and so on, to supply drivers' cognition and prevent accidents. Despite the technological advancement, completely software-defined vehicles with accessible toolchains and APIs are still under development. Most of the in-vehicular sensors and interfaces are brand-specific or closed, limiting access to the data, computing, and networking capabilities and thus hindering, for example, vehicular machine learning and artificial intelligence (ML/AI) application development. 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 [20,65]. 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 are a potential solution for providing services and applications for drivers and traffic situations [5, 43]. 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 demand training data, which in-vehicle sensors and external databases can provide. 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 [5,13] 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 the driver's interaction with the automation and capabilities to take 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 [13], such as cameras, LiDARs, radars, and speed meters [27,58]. This information can be used to, for example, improve lane [50] and road pothole [29] recognition. Solutions for detecting drivers' behaviour while using smartphones during driving [87] and drunk driving [53] have been explored. However, the results underline that human drivers' perception and reasoning still maintain an advantage compared to fully automatic vehicles [76]. Different services and data sources are needed to understand the whole picture of driving performance and safety. Local real-time computing can ease drivers' senses 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 [20,65].

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. Developments on software-defined solutions and reference architectures for vehicular and edge-cloud computing have been suggested recently [65]. 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 [21,33]. Furthermore, the software-defined networking approach is a prerequisite for managing vehicular ad-hoc networks of connected vehicles

necessary for the evolving smart traffic and transport systems [15]. The offloading and external data services then require software-defined, multi-access edge-cloud solutions [6, 85] to be dynamically, scalably, and securely orchestrated with the vehicular environment.

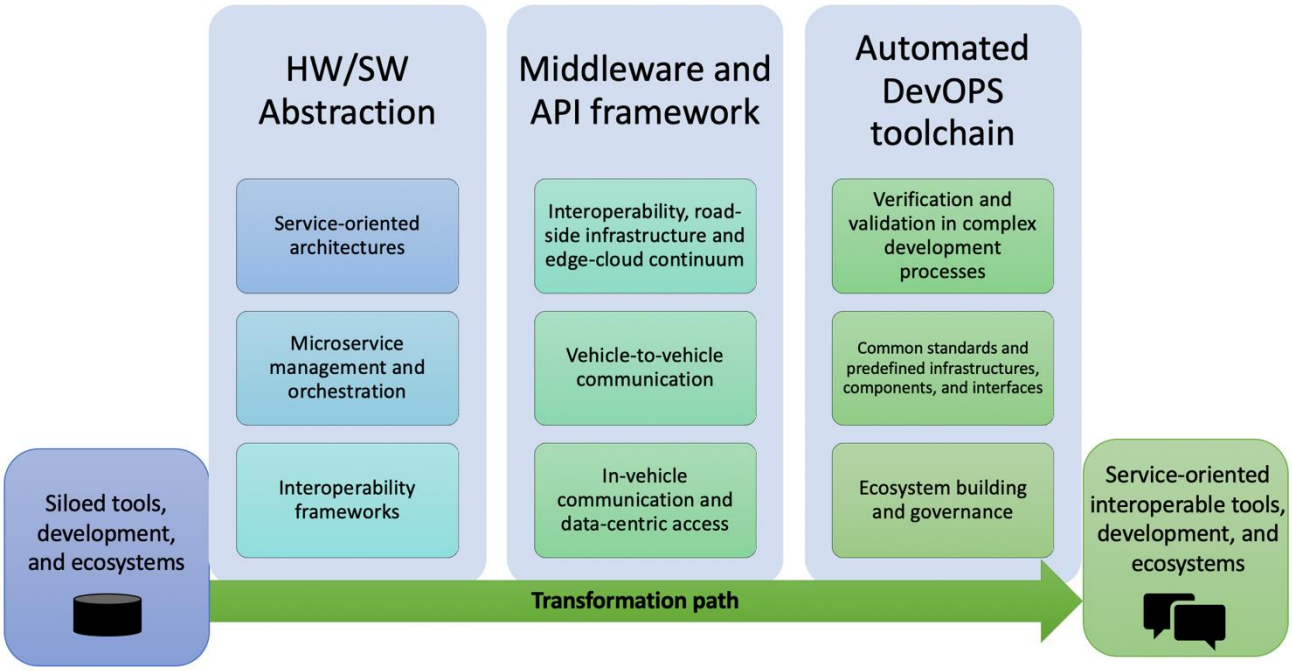

Figure 1: An overview of the FEDERATE SDV building block categories (Hardware/Software Abstraction, Middleware and API Framework, and Automated DevOps Toolchain) and corresponding themes discussed in this paper.

1.2 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 [20]. 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. In the beginning, three main categories of building blocks were considered. However, these will be defined more robustly and specifically when the 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 the 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 interoperability aspects of HW/SW 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 by automated DevOps toolchains and building healthy software ecosystems to ensure vehicular software life cycle and 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 Software-Defined Vehicles (SDVs) 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 [38,42].

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 [69,92]. Containerisation and virtualisation are essential technologies within the SDV framework, facilitating rapid software deployment and updates crucial for maintaining the performance of intelligent vehicles [89]. 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 [10]. This shift brings opportunities and challenges, particularly in vehicle architecture, cybersecurity, and system integration [55,63].

2.1 Service-Oriented Architectures (SOA) in Automotive

Service-oriented architectures (SOA) are emerging as a promising solution to the challenges posed by the increasing complexity of automotive software systems. SOA allows for the dynamic integration of software components, enabling more flexible and scalable vehicle architectures. This approach is particularly well-suited to the needs of SDVs, which require the ability to adapt to changing software requirements and system configurations [26].

Research has shown that SOA can significantly improve automotive software systems' functional suitability and scalability. However, implementing SOA in the automotive domain also presents challenges, particularly in ensuring the system's security, safety, and reliability [14,72]. 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 standards of performance and security [19].

2.2 Mainstream Classic Architecture

AUTOSAR Classic Platform: The AUTOSAR (AUTomotive Open System ARchitecture) 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 [12].

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 integrating new functionalities as vehicle software continues to evolve [11].

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 [51].

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 [66].

It is worth mentioning the Android Open Source Project (AOSP) and Android Automotive OS (AAOS) 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 like 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 customize and develop unique in-vehicle experiences, integrating GAS introduces Google's ecosystem of services, enriching the 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 [62].

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 [32].

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 [47].

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 [9,37].

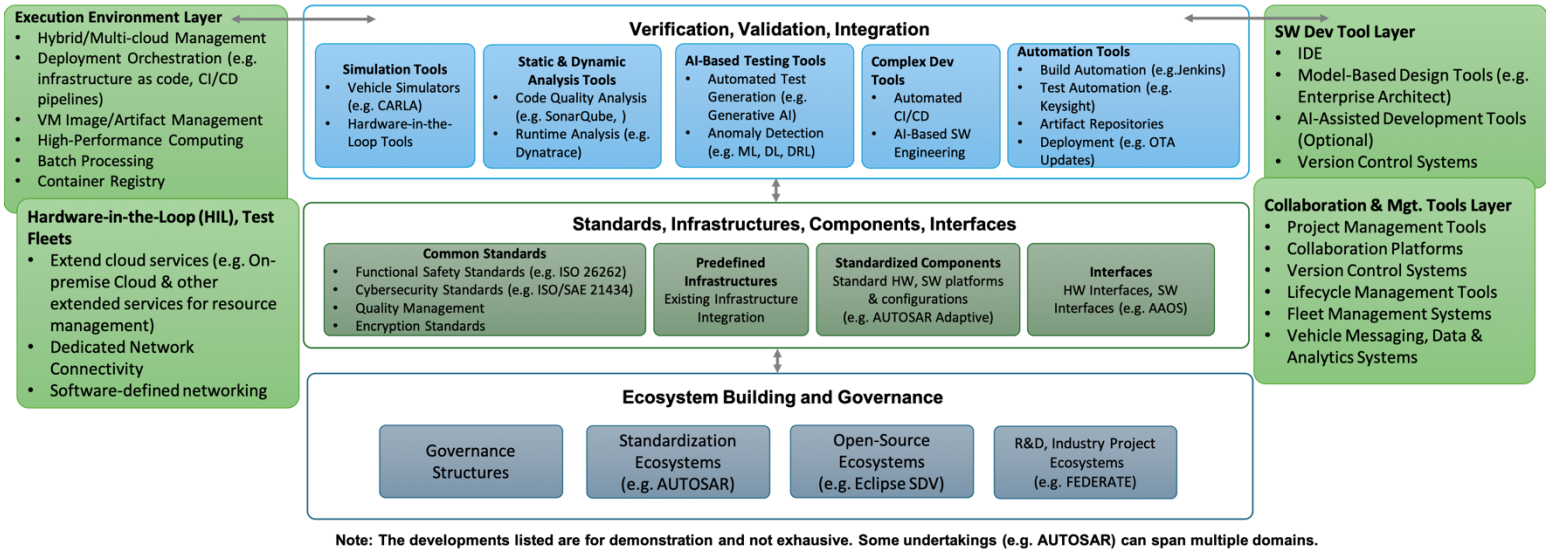

Figure 2: An example of the Layered Tooling Reference Architecture for SDVs.

2.5 Vision and Roadmap Towards Microservices

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 goes beyond traditional microservices by considering the potential of dynamic service orchestration, which allows 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 microservices' development, testing, and deployment, ensuring optimised operations across the entire vehicle software stack [31]. Some reference architectures, such as RobotKube [49], 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 software-defined vehicles 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 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.
- The architecture should provide security already on the design level as more interconnected, software-based platforms make vehicles more vulnerable to cybersecurity threads.
- The architecture should provide seamless interoperability between different components, platforms, and services, as SDVs require highly coordinated and efficient communication across various systems and networks to function effectively.
- 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 SOA's benefits 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. [73]

3 Interoperability Aspect in SDVoFs

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 [70]. The definition of interoperability has evolved considerably over the last two decades with the development of countless technologies, application systems and multi-disciplinary approaches to engineering. One of the recent works [57] related to the context-aware software systems (CASS) has discussed interoperability theoretical framework (ITF), which observed interoperability in two aspects, i.e. 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, majorly 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 and lays the foundation for innovative research and development. The need for open and interoperable platforms has necessitated the classical architectures in SDVs to adopt SOA and microservice architecture. To ensure the dynamic interactions between in-vehicle and external systems, monolithic software breaks down into independent services developed and deployed without disrupting the overall vehicle system [67].

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, 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), 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), GDPR.	- Coordination across sectors (OEMs, Tier-1 Suppliers, regulators). - Cohesive development and Compliance.

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

3.1 Six-Tier Interoperability Framework for SDVs

Different components, data sources, and interfaces in Software-Defined Vehicles are often procured from various vendors, causing compatibility issues. Therefore, interoperability is crucial in ensuring heterogeneous system integration, an adaptation of legacy systems, real-time communication, third-party integration, scalability, cross-system compatibility, modular software updates, and so on [3]. Cohesive interaction among different systems, components, and platforms requires a multi-layered concept of interoperability. The FEDERATE project SDVoFs initiative considers multi-dimensional facets of interoperability in its SDV architecture. Each level of interoperability contributes to the seamless interactions with cloud services and V2X networks.

Analysis of different levels of interoperability for SDVoFs is provided in Table 1. This Six-Tier Interoperability Framework for Software-Defined Vehicles (SDVs) provides a comprehensive approach to ensure seamless communication and coordination across systems. It begins with technical interoperability, which focuses on hardware and software communication and progresses through syntactic and semantic interoperability to ensure data is correctly formatted and interpreted. Pragmatic interoperability addresses the contextual use of data, while dynamic interoperability ensures real-time adaptability. Finally, organisational interoperability emphasises collaboration between stakeholders, such as manufacturers and regulators, to promote cohesive development and compliance across the SDV ecosystem.

3.2 Key Challenges

- Vehicles consist of diverse components and systems, which often come from various vendors and follow different protocols, making it difficult to ensure compatibility.
- 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 should be achieved to have SDVs function effectively in real-world environments.

4 Intelligent Transportation Systems and Vehicular Edge-Cloud Continuum

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.

4.1 Roadside Infrastructure

Roadside infrastructure refers to the various sensors, communication devices, and computing resources installed along roadways [74]. These include roadside units (RSUs), which facilitate Vehicle-to-Everything (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 [4]. Roadside infrastructure and edge-cloud continuum support the seamless operation of smart vehicles by providing real-time data and connectivity [77], expanding vehicular data capabilities from single-vehicle applications to connected, resourceful applications.

Roadside infrastructure is critical in improving road safety by enabling real-time communication between vehicles and the environment. 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 [82]. 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 [30].

Despite the advantages mentioned, there are a couple of disadvantages to RSUs as well. 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 [34]. Besides, as the number of smart vehicles increases, the demand for roadside infrastructure will also grow. 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 [23,83]. Ensuring robust cybersecurity measures is critical, but this adds to the complexity and cost of deployment.

4.2 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 [56], typically located within the roadside infrastructure, such as base stations. This approach reduces latency [52] 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. Besides, VEC can be more easily scaled 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 [61]. 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 around data integrity and privacy [25]. 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 [68]. 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 exchanging 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 reduce latency. This requires standardisation across hardware and software platforms, which can be challenging given the diversity of stakeholders involved.

4.3 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 ITS ecosystem's immediate benefits and long-term sustainability.

4.4 Key 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.
- Required real-time computing presents complexities in ensuring secure and reliable systems across diverse systems and platforms.
- Standardization 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.

5 V2X Communications

Maintaining real-time situational awareness of the environment for safe autonomous driving is challenging to achieve solely through the vehicle's perceptual capabilities due to the limited sensing, computing, and communication resources at the vehicle level. To overcome the limitations of autonomous driving in level four and beyond, a collaborative perception (CP) system can be implemented to facilitate vehicle-to-network and cloud connectivity. In this venture, V2X technology enables vehicles to connect with their surroundings—neighbouring vehicles, cyclists, pedestrians, and road infrastructure, via wireless links for improved driver awareness and automotive safety.

In 1999, the U.S. Federal Communications Commission (FCC) allocated 75 MHz spectrum in the 5.9 GHz band for Intelligent Transportation Services (ITS). This allocation led to the initiation of research activity to develop and deploy V2X communications. The research resulted in the introduction of the first set of radio standards for V2X in 2010. These standards were based on the IEEE 802.11p technology and developed in the US as Dedicated Short-Range Communications (DSRC) [41]. Radio standards were developed by the definition of higher layer standards, message formats, protocols, and applications in Europe and the U.S. [28]. Furthermore, direct communications via sideline between vehicle to vehicle or infrastructure have been enabled with the introduction of Proximity Services (ProSe) in 3GPP Long Term Evolution (LTE) Release 14 and evolved in Release 15, which functions without or with any involvement of the Base station or the gNB. As a step forward, an evolved version of V2X known as cellular V2X (C-V2X) enables both direct and network connections to emerge.

In C-V2X, direct communications are enabled via the PC5 interface and network communications via the Uu interface. The PC5 interface allows vehicles to communicate directly with other road users and roadside infrastructure using sidelink channels. Thereby, the direct communication mode that allows vehicle-to-vehicle (V2V), vehicle-to-infrastructure (V2I), and vehicle-to-pedestrian (V2P) communications, etc. does not necessarily require any network infrastructure. For this direct communication, a specialised WiFi mode in the 5.9 GHz frequency band, which enables short-range (< 1 km) ad hoc communication with local vehicles, pedestrians and road infrastructure, is utilised. On the other hand, vehicle-to-network (V2N) communication is enabled via the Uu interface, which operates in the traditional mobile broadband spectrum. V2N connects conventional mobile networks to vehicles to receive real-time information via edge-cloud continuum services, such as local road traffic and road and weather conditions. Thereby, V2N plays a pivotal role in enabling services such as over-the-air updates, information sharing, and task offloading. Collision avoidance actions are expected to be executed using onboard vehicle sensors in critical situations. At the same time, emergency alerts will be communicated through V2V and V2I systems, and enhanced risk avoidance will be facilitated via V2N information sharing.

5.1 Vehicular Communication Standardisation

Concerning V2X communication standardisation, regulatory bodies such as IEEE [36], 3GPP [1,71], and Intelligent Transportation Systems (ITS) [39] have proposed standard network requirements for autonomous driving. 3GPP has defined V2X communication modes 3 and 4, which allocate network resources based on the vehicle's coverage status, whether within or outside network coverage. 3GPP has also identified and proposed network service requirements for prominent use cases, including vehicle platooning, advanced driving assistance, extended sensing, and remote driving [86].

The standardisation focuses on addressing key challenges in V2X-assisted automated driving, including achieving the timing and end-to-end communication latency, testing methodologies, network resource

allocation, integrating network infrastructure and maintaining trustworthiness in the 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. Further, experimental validation of the proposed V2X standards is ongoing and works in [24,54] present the impact of hardware limitations, interoperability, and network resource unavailability on the network performance. However, many challenges still exist regarding V2X deployment.

5.2 Towards Large-scale V2X Deployments

Europe is at an early stage of commercial V2X deployment. A significant challenge has been the lack of consensus among industry players regarding the communication protocol. The ongoing debate between DSRC, based on wireless local-area networks, and C-V2X, utilising LTE and 5G, has hindered progress. Each player has preferences, prolonging discussions around these technologies. Nonetheless, V2X hardware providers, software providers, and cybersecurity companies have developed solutions compatible with both protocols, allowing deployment without compatibility concerns. However, since DSRC and C-V2X are fundamentally incompatible at the access layer, achieving dual compatibility requires advanced hardware and further development. While this hybrid approach can temporarily address interoperability issues, it is not a sustainable long-term solution; many experts predict that one protocol will eventually prevail. Additionally, expanding V2X services on a large scale will necessitate deploying more on-board units (OBUs) and road-side units (RSUs), which consumes considerable time and cost. As of 2023, North America and China have primarily converged on C-V2X, phasing out DSRC. In contrast, Europe remains divided, with Volkswagen using DSRC while BMW and Daimler support C-V2X.

A significant challenge is the public's limited understanding of V2X technology and its potential, which creates uncertainty in market demand and makes it difficult to predict consumer interest. Given consumers' strong desires for safety and convenience—both of which V2X can provide—there is significant potential demand. The issue is not whether demand exists but whether consumers are informed enough to realise how V2X can meet their needs. Consequently, industry players must invest not only in the technology itself but also in educating consumers about its benefits and creating innovative, appealing services [8].

5.3 Key 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.
- Achieving compatibility between incompatible systems is a significant technical challenge, which requires advanced hardware and further development to support dual compatibility in the interim.
- Expanding V2X services on a large scale presents practical and financial difficulties, particularly in deploying additional onboard and roadside units, which is time-consuming and costly.

6 In-vehicle Communication

6.1 CAN protocols

The CAN (Controller Area Network) protocol is a vehicle bus standard that enables communication between microcontrollers and devices without needing a central host computer. Initially designed for automotive use by Bosch [45], it has since expanded into various industrial applications. The protocol works on a multi-master, multi-drop network, where any node can initiate communication with another. Messages are assigned unique priorities via their identifiers, and when two nodes transmit simultaneously, the message with the higher priority (lower identifier number) gets transmitted first, while the other waits and retries later. This mechanism ensures reliable communication, making CAN suitable for real-time, safety-critical vehicle systems [75]. 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 [60].

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 2.0B allows for 29-bit identifiers, ensuring flexibility and compatibility across various systems; these can be used in the same bus as long as no extended frames are sent by controllers using 2.0B [2]. As vehicle systems became more complex, CAN FD (Flexible Data Rate) emerged as a solution, offering higher data transmission speeds and larger payloads. This advanced protocol reduces wiring complexity and supports more sophisticated in-vehicle functions, making it a preferred choice for modern automotive systems [84]. 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 [22].

6.2 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 [78].
- FlexRay is a high-speed, time-triggered protocol used in safety-critical applications such as electronic stability control (ESC) and adaptive cruise control [78].
- MOST (Media Oriented Systems Transport) is designed explicitly for infotainment systems, providing high-bandwidth communication for audio, video, and multimedia data [78].
- Ethernet is increasingly adopted in automotive applications due to its high data rate and flexibility. It is well-suited for modern, data-intensive systems like 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, allow for integrating 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 essential for advanced driver assistance and infotainment systems. [79].

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 like autonomous driving [48]. 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 important for reliable in-vehicle networking [46]. 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 [78]. 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 [80].

6.3 Key Challenges

- Limited bandwidth: Traditional CAN can struggle to handle the increasing data demands of modern vehicle systems, such as autonomous driving [18].
- CAN Bus overload: Increased data transmission risks communication delays or data loss [90].
- 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 [59].
- 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 [16,17].

7 Validation and Verification

7.1 SOA and temporal properties

A service-oriented architecture (including microservices) is a desirable approach for building flexible, modular systems where independent services interact to perform a wide range of functions. For autonomous vehicles, a prime example area of service-oriented architectures would start at data capture of each sensor, then the fusion of this data into a comprehensive representation of the vehicle and its environment. Depending on different conclusions drawn, the available information transforms into actuator inputs, primarily steering angle and vehicle velocity. To realise an implementation of 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 out 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 considering the verification and validation of vehicular and transportation systems. A vehicle's response depends very much on the intent of the vehicle 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 is on a rural road, we expect the behaviour of the vehicle to correspond to the environmental state.

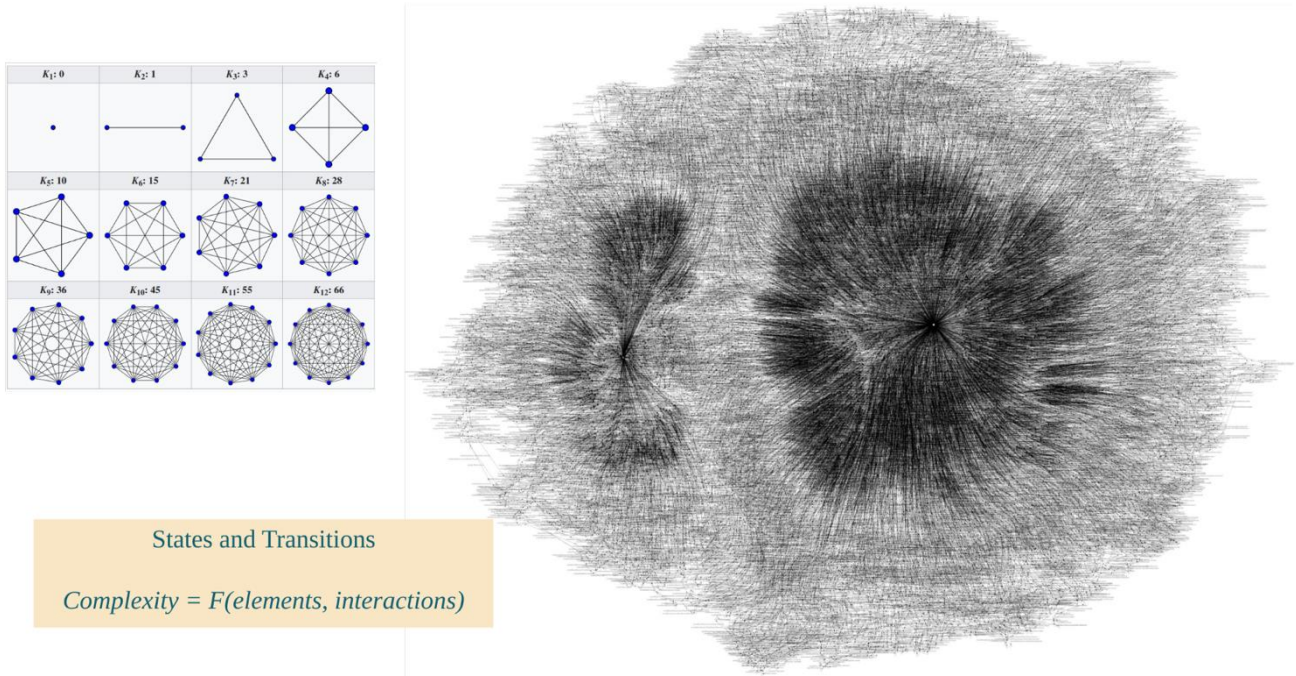

Figure 3: Visualising complexity in validation and verification processes. The image on the right represents a complex state-transition graph where each node corresponds to a system state, and each line represents a transition between states. The dense structure highlights the intricate interactions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

7.2 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 far 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 inadequate to achieve acceptable functional safety and security for the software-defined vehicle.

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

In this chapter, we explore current methods and state-of-the-art approaches for validation and verification in the context of Service-Oriented Architecture (SOA). These methods, either 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.

7.4 Key challenges

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

8 Building the Software Ecosystems

The Software-Defined Vehicles of Future (SDVoF) initiative emphasises the creation of a robust ecosystem, integrating various stakeholders, including OEMs, suppliers, 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, 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 [20] has identified some significant challenges the European automotive industry faces. The shift to autonomous, electric, and connected vehicles leads to greater software complexity, where vehicle code is expected to grow dramatically, challenging both development and maintenance. Customers increasingly demand new features and frequent software updates, increasing the pressure towards continuous innovation. While large tech companies dominate some key areas, creating dependencies for traditional OEMs, Non-EU manufacturers, such as Tesla, with a software-first approach, are creating competitive pressure. Moving from proprietary software and platforms and fragmented development initiatives towards an open ecosystem centred around non-differentiating software solutions can decrease inefficiencies and duplicated efforts, which is essential for a strong European automotive industry. Thus, collaborative efforts are needed to build an open SDV ecosystem and reinforce the region's strategic autonomy in this area.

Interoperability is a cornerstone of modern industrial and automotive systems, particularly in the context of software-defined vehicles (SDVs). As the automotive industry shifts towards increasingly complex and interconnected digital environments, ensuring that different systems, components, and software solutions seamlessly work together is critical for innovation and efficiency. The Eclipse Arrowhead framework[1], known for its Service Oriented Architecture-based interoperability and microservices, provides a robust architecture that facilitates this seamless interaction by enabling scalable and interoperable automation solutions. Such frameworks as the Arrowhead 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.

The ecosystem aims to support the European automotive industry's strategic autonomy by encouraging open-source solutions and reducing dependence on non-European technology providers. To avoid duplicating efforts and ensure the utilisation of existing building blocks and design patterns, it is crucial to connect existing initiatives and partners [20]:

- 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.

[1] https://arrowhead.eu/eclipse- arrowhead- 2/

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, moving from proprietary siloed value chains towards a robust ecosystem, connecting multiple partners with various interests and goals, is challenging. In value chains, interactions and data flows are typically linear and controlled within predefined organisational boundaries. As the scope shifts to an ecosystem, where multiple actors contribute together in a more open and interconnected environment, complexity increases, too. Standardising data acquisition and exchange across diverse participants in the ecosystem, thus, becomes critical. Additionally, governance becomes more complicated, extending beyond organisational boundaries. Issues such as intellectual property rights (IPRs) related to data analytics and processing become more pronounced, requiring clear agreements and governance structures. [91]

Engaging stakeholders in an ecosystem is crucial, as their commitment is necessary to achieve successful outcomes. However, 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. [35] While partners must be open to collaborating and integrating into the ecosystem that is able to align with their own goals [81], the ecosystem needs to be attractive enough to engage a critical mass of partners for effective outcomes [35].

Accordingly, their ability to coordinate interrelated organisations with significant autonomy is central to ecosystems. This necessitates structures that allow different system parts to operate independently while still adhering to common standards and predefined interfaces. This coordination is achieved through processes, rules, and standards that help resolve issues and ensure alignment among participants. [40] Here, ecosystem governance is necessary to manage the balance between fostering innovation and maintaining the ecosystem's overall health. Without effective governance, the uncontrolled growth of low-quality innovations could harm the platform, potentially destabilising or even destroying the ecosystem. Governance helps guide the development of the ecosystem by managing tensions and ensuring that the collective efforts of diverse actors contribute positively to meeting market demands. [88] Overall, the governance needs to align the interests and goals of individual partners with those of the ecosystem as a whole, allowing innovation from ecosystem partners but in a coordinated manner [35].

8.1 Key Challenges

- Achieving seamless collaboration between diverse stakeholders, including manufacturers and regulators, is essential to ensure cohesive development and integration of systems.
- Engaging stakeholders in an ecosystem is crucial, as their commitment is necessary to achieve successful outcomes.
- Ecosystem governance is necessary to manage the balance between fostering innovation and maintaining the ecosystem's overall health.

9 Transformation Path

Transformation towards a software-defined vehicle cannot be achieved without a people-centric transformation path considering different stakeholders, developers, and other integral people in the development pipeline. The technical and architectural change starts from monolithic software stacks and E/E architectures that are transformed into more dynamic software systems, where novel paradigms of microservices and service-oriented architectures are applied and addressed. The change will affect the whole product life cycle and create a software-defined vehicle as a unique novel paradigm and product. The focus on defining the transformation cannot be not only technical but also societal and community-related.

The current manufacturer landscape in the automotive industry is still siloed between large manufacturers that manage not only the mechanics of the products but also software developed and implemented in the market. The necessary transformation path encompasses and defines vehicular software development, software life cycle, and developer communities and goes across all vehicles, manufacturers, and models. The classic V model, which begins with planning and extends to the finished (validated or certified) software, does not fulfil the requirement of a connection between the development teams. This changes not only the way (processes, workflows, methodologies) and the tools that are used (entire toolchains) to develop the software but also the roles of the companies and people involved in the development (new business models, changed value chains, as well as new working methods of the development teams). Responsibilities are also shifting, and boundaries are disappearing (the cooperation of many people involved in development, operation, maintenance, and other service providers is necessary). An SDV is a system of systems in which everyone involved works in parallel and is in constant dialogue with each other. This requires new skills and approaches from those involved.

It seems clear that this transformation cannot be achieved overnight. Therefore, how this transformation takes place must be seen as one of the critical challenges on the road to SDV. This transformation should be integrated into the existing workflow as harmoniously as possible and step by step, building on existing solutions, supplementing them where necessary and, if not otherwise possible, replacing them with new approaches or technologies. This should guarantee that all those involved recognise themselves in this transformation, actively support it, are open to the new challenges and move towards SDV together. The successful transformation should lead to healthy ecosystems where shared resources and interfaces make it possible for new innovative companies to enter the market, increase innovation, and support the creation of novel products, services, and software.

To achieve a successful transformation path, it is necessary to identify the key defining aspects that characterise the desired automotive software development pipelines and developer communities. As this industry comes with a long history and legacy that might even be unique in the world, it might be hard to find corresponding examples from a cross-industrial perspective. However, possible lessons learned should be identified among other industries that have undergone a digital transformation to find the best practices, avoid pitfalls, and ensure healthy growth during the process. Different stakeholders, developers, and communities should be involved in creating a roadmap for the transformation path. To move along the transformation path successfully, the necessary activities may include training and educational activities, community-building activities, and involving a wide variety of stakeholders from public and private players. The success of the transformation path should be measured actively as the acceptance inside the ecosystem and by utilising a variety of healthiness measures: community engagement and activity of its members, adaptation, and usage of the software products developed through the ecosystem, quality, and interoperability of the products and interfaces, and so on.

To successfully transition from siloed software development to an open ecosystem, careful governance of the transformation path is essential. The process begins with identifying and engaging various ecosystem actors and understanding their roles, particularly those critical to the ecosystem who may currently favour siloed business models over open collaboration. These actors must be recognised and potentially incentivised to participate, as not all will be willing to engage in co-developing non-differentiating building blocks without motivation.

Once the key ecosystem actors and their roles are recognised, it becomes essential to align the prospects and characteristics of individual organisations with the overarching ecosystem vision. This alignment ensures that all participants are dedicated to a shared goal. To foster innovation while maintaining ecosystem integrity, a governance structure that integrates centralised, decentralised, and group-level elements must be implemented. This structure will manage the interactions within the ecosystem, balancing synergies and promoting collective innovation and efficiency.

In addition, evaluating the potential for improved performance across organisational and sector boundaries is crucial. This evaluation supports the case for collaboration and integration, ensuring that the collective efforts of ecosystem partners are directed toward common goals while respecting the individual business interests of each participant. Effective governance of the open ecosystem centred around non-differentiating building blocks within the vehicle software stack seeks to minimise duplicated efforts and optimise the development process for essential software components, thereby speeding up the introduction of new features and enhancing cost efficiency.

10 Conclusions

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

References

[1] 3GPP. 3gpp – the mobile broadband standard. https://www.3gpp.org/. (Accessed on 08/19/2024).

[2] Kvaser AB. The can bus protocol tutorial, 2023. Accessed: 2024-07-30.

[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. 2024.

[4] Shiva Agrawal, Rui Song, Kristina Doycheva, Alois Knoll, and Gordon Elger. Intelligent roadside infrastructure for connected mobility. In SMART GREENS/VEHITS, 2022.

[5] Ahmad Alhilal, Tristan Braud, and Pan Hui. Distributed vehicular computing at the dawn of 5g: a survey. arXiv:2001.07077, 2020.

[6] 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.

[7] 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.

[8] 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).

[9] SBD Automotive. How will the software-defined vehicle impact the automotive industry?, Unknown.

[10] AUTOSAR. Autosar classic platform, 2023. Accessed: 2024-08-22.

[11] AUTOSAR. Adaptive autosar, Unknown. Accessed: 2024-08-22.

[12] AUTOSAR. Autosar classic platform, Unknown. Accessed: 2024-08-22.

[13] 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.

[14] M. Becker, D. Ganesan, and S. Kowalewski. Safety and security challenges in automotive software systems: A soa perspective. In 2023 IEEE International Conference on Software Architecture (ICSA), pages 1–10, 2023.

[15] 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.

[16] 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.

[17] G. Cena, S. Scanzio, and A. Valenzano. Composite can xl-ethernet networks for next-gen automotive and automation systems. 2023 IEEE 19th International Conference on Factory Communication Systems (WFCS), pages 1–8, 2023.

[18] Raghu Changalvala, Brandon Fedoruk, and Hafiz Malik. Radar data integrity verification using 2d qim-based data hiding. Sensors (Basel, Switzerland), 20, 2020.

[19] A. Chattopadhyay, M. Lukasiewycz, S. Chakraborty, and A. Knoll. Security and safety co-design for automotive systems: Challenges and emerging solutions. In 2023 IEEE International Conference on Software Architecture (ICSA), pages 1–10, 2023.

[20] 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.

[21] Nada Cvijetic and Tom Tomazin. Developing a centralized compute architecture for autonomous vehicles. ATZ electronics worldwide, 16:10–15, 2021.

[22] 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.

[23] 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.

[24] Giammarco Di Sciullo, Luca Zitella, Elena Cinque, Fortunato Santucci, Marco Pratesi, and Francesco Valentini. Experimental validation of c-v2x mode 4 sidelink pc5 interface for vehicular communications. In 2022 61st FITCE International Congress Future Telecommunications: Infrastructure and Sustainability (FITCE), pages 1–6, 2022.

[25] Mehmet Ali Eken and Pelin Angin. Vehicular edge computing security. Secure Edge Computing, 2021.

[26] G. Fortino, G. Russo, W. Russo, and A. Puliafito. Toward service-oriented architectures for the automotive domain: State of the art and future challenges. IEEE Transactions on Industrial Informatics, 18(5):2835–2846, May 2022.

[27] 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.

[28] Mario H. Castañeda Garcia, Alejandro Molina-Galan, Mate Boban, Javier Gozalvez, Baldomero Coll-Perales, Taylan Şahin, and Apostolos Kousaridas. A tutorial on 5g nr v2x communications. IEEE Communications Surveys & Tutorials, 23(3):1972–2026, 2021.

[29] 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.

[30] 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.

[31] 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.

[32] NCC Group. Microservices-based architectures, Unknown.

[33] 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.

[34] 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.

[35] 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.

[36] IEEE. Ieee - the world's largest technical professional organization dedicated to advancing technology for the benefit of humanity. https: //www.ieee.org/. (Accessed on 08/19/2024).

[37] Deloitte Insights. Software-defined cars: Industrial revolution on the arrow, Unknown.

[38] 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.

[39] Intelligent Transportation Society (ITS). Cen/tc 278 - road transport and traffic telematics. https://standards.iteh.ai/catalog/tc/cen/aa876bfe-1d4c-4f9a-8e50-2fcca8393dea/cen-tc-278?srsltid=AfmBOoo58YW552XpxVuGJTwRyPSzOWs90lAf7pTLax7UPm9fJYJTJ9ar. (Accessed on 08/19/2024).

[40] Michael G Jacobides, Carmelo Cennamo, and Annabelle Gawer. Towards a theory of ecosystems. Strategic management journal, 39(8):2255–2276, 2018.

[41] John B Kenney. Dedicated short-range communications (dsrc) standards in the united states. Proceedings of the IEEE, 99(7):1162–1182, 2011.

[42] 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.

[43] 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.

[44] Samaneh Khazraeian and Mohammed Hadi. Intelligent transportation systems in future smart cities. Studies in Systems, Decision and Control, 2018.

[45] B. Kirk. Using software protocols to mask can bus insecurities. In IEE Colloquium On Electomagnetic Compatibility Of Software, pages 5/1–5/5, 1998.

[46] Kvaser. Can standards. https://kvaser.com/about-can/ can-standards/, 2024. Accessed: 24.7.2024.

[47] Aurora Labs. The role of AI in software-defined vehicles, Unknown.

[48] 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.

[49] Bastian Lampe, Lennart Reiher, Lukas Zanger, Timo Woopen, Raphael van Kempen, and Lutz Eckstein. Robotkube: Orchestrating large-scale cooperative multi-robot systems with kubernetes and ros. In 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), pages 2719–2725, 2023.

[50] 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.

[51] Automotive Grade Linux. Automotive grade linux, Unknown.

[52] Lei Liu, Chen Chen, Qingqi Pei, Sabita Maharjan, and Yan Zhang. Vehicular edge computing and networking: A survey. Mobile Networks and Applications, 2020.

[53] 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.

[54] Meng Lu, Jaime Ferragut, Matti Kutila, and Tao Chen. Next-generation wireless networks for v2x. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pages 1–5, 2020.

[55] L. Mauser and S. Wagner. Centralization potential of automotive e/e architectures. In 2023 IEEE International Conference on Automotive Software and Systems (ICASS), pages 1–10, 2023.

[56] 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.

[57] 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.

[58] 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.

[59] 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.

[60] 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.

[61] Richard Olaniyan, Olamilekan Fadahunsi, Muthucumaru Maheswaran, and Mohamed Faten Zhani. Opportunistic edge computing: Concepts, opportunities and research challenges. ArXiv, abs/1806.04311, 2018.

[62] Alex Oyler. Automotive with aosp & google services, 2022.

[63] F. Pan, J. Lin, and M. Rickert. Automatic platform configuration and software integration for software-defined vehicles. In 2022 IEEE International Conference on Software Architecture (ICSA), pages 1–10, 2022.

[64] 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.

[65] 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.

[66] Android Open Source Project. What is android automotive?, Unknown.

[67] Dominik Püllen. Holistic Security Engineering for Software-Defined Vehicles. PhD thesis, Universität Passau, 2024.

[68] Samuel Rac and Mats Brorsson. At the edge of a seamless cloud experience. ArXiv, abs/2111.06157, 2021.

[69] Gregor Resing. The software defined vehicle. https: //www.ibm.com/blogs/digitale-perspektive/2023/06/ the-software-defined-vehicle/, 2023. Accessed: 2024-08-23.

[70] 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.

[71] rimdeolabs. V2x communications within the 3gpp standards - rimedo labs. https://rimedolabs.com/blog/ v2x-communications-within-the-3gpp-standards/, 11 2021. (Accessed on 08/20/2024).

[72] J. Ruh, M. Schörner, and F. Sagstetter. Towards a security-aware service-oriented architecture for automotive systems. In 2023 IEEE International Conference on Software Architecture (ICSA), pages 1–10, 2023.

[73] 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.

[74] 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.

[75] 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.

[76] Brandon Schoettle. Sensor fusion: A comparison of sensing capabilities of human drivers and highly automated vehicles. University of Michigan, 2017.

[77] 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.

[78] Craig Smith. The Car Hacker's Handbook: A Guide for the Penetration Tester. No Starch Press, San Francisco, 2016.

[79] 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.

[80] Embien Technologies. Sae j1939 protocol: An introduction. https://www.embien.com/automotive-insights/ sae-j1939-protocol-an-introduction, 2024. Accessed: 17.07.2024.

[81] 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.

[82] 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.

[83] 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.

[84] 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.

[85] 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.

[86] Donglin Wang, Yann Nana Nganso, and Hans D Schotten. A short overview of 6g v2x communication standards. In 2023 International Conference on Intelligent Communication and Networking (ICN), pages 20–26. IEEE, 2023.

[87] 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.

[88] Jonathan Wareham, Paul B Fox, and Josep Lluís Cano Giner. Technology ecosystem governance. Organization science, 25(4):1195–1215, 2014.

[89] Hans Windpassinger. The software defined vehicle the - the architecture behind the next evolution of automotive industry. https://www.ibm.com/blog/the-software-defined-vehicle-the-architecture-behind-the-next-evolution-of-the-automoti 2023. Accessed: 2024-08-23.

[90] 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.

[91] 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.

[92] 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.