Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Software-Defined Vehicle Platforms for Safety-Critical Control: Architecture, Updates, and Validation

Ajit Gajre
Michigan Technological University, USA
Corresponding Author: Ajit Gajre, **E-mail**: ajitgajreglobal@gmail.com

| ABSTRACT

The current automobile technology is no longer hardware-oriented in designing vehicles but has become Software-Defined Vehicle platforms that have redefined the way cars are designed, built, and supported during their operational life. This change is no longer tied to feature development and inflexible hardware requirements, which allows dynamic over-the-air updates and seamless cross-domain functionality integration. The evolution is to deal with the increasing software complexity and cut down the cost of development, as well as accelerate the time-to-market delivery by having a single common platform across a variety of vehicle types and types of powertrain. Critical application of safety controls requires rigorous compliance with automotive safety standards, especially ISO standards of functional safety and cybersecurity, and it also includes systematized hazard identification and component-based development approaches. Time-Sensitive Networking protocols apply deterministic networking infrastructure offering tight timing control of safety-critical communications with the support of sophisticated encryption and network segmentation schemes. Higher over-the-air update functionalities offer revolutionary advantages, as they make available sophisticated orchestration and validation strategies, the use of digital twins technologies, and rollout strategies, to ensure system integrity and safety of operations throughout the vehicle lifecycle.

| KEYWORDS

Software-Defined Vehicles, Automotive Safety Standards, Time-Sensitive Networking, Over-The-Air Updates, and Vehicle Cybersecurity

| ARTICLE INFORMATION

ACCEPTED: 01 October 2025 **PUBLISHED:** 26 October 2025 **DOI:** 10.32996/jcsts.2025.7.11.10

1. Introduction

Today's car industry faces a massive change. For decades, vehicles relied on fixed hardware systems where each function needed its own dedicated electronic box. Now, everything is shifting toward software-controlled platforms that can update and improve after the car leaves the factory. This isn't just a minor upgrade - it's completely rewriting how cars get built and maintained.

Think about smartphones. When Apple releases iOS updates, phones gain new features without buying new hardware. Cars are heading in the same direction. Software-Defined Vehicles break the old rules where adding a new feature meant installing new electronic parts. Instead, manufacturers can push updates that unlock capabilities, fix problems, or even add entirely new functions through wireless connections.

This transformation creates serious technical challenges. Managing software complexity in safety-critical automotive environments requires completely new approaches compared to traditional methods [1]. The old way of building cars - where each model needed its own unique electronic systems - simply doesn't work anymore. Modern vehicles need unified platforms that work across different car types and engine configurations.

Standards become crucial here. AUTOSAR has become the foundation that lets different car systems talk to each other properly, cutting down development time while making software more reliable and secure [2]. Without these standards, the complexity would be unmanageable. Car companies can now build more efficiently while still meeting tough safety requirements. The business impact is enormous - analysts predict massive growth in software-focused vehicle technology, especially for wireless updates and connected services that go far beyond what traditional cars offered.

2. Industry Context and Platform Architecture

Car manufacturers are completely restructuring how vehicles are designed. The old approach used dozens of separate computer boxes scattered throughout the car, each handling specific functions. Modern cars are moving toward centralized computing unit that manages multiple systems from fewer, more powerful processors. This change isn't just about technology - it's about survival in a market where software updates happen monthly rather than never.

Building robust software foundations for cars requires thinking decades ahead while staying compatible with current technology [3]. The challenge is immense. The current luxury cars have more computing power than the initial space mission, but all have to be reliable to last 15-20 years under extreme conditions, such as the desert heat and the arctic cold. The processing power continues to increase, but the requirements of autonomous driving facilities, entertainment, and safety services continue to rise.

Network processor design in cars shows the constant tension between wanting maximum performance and dealing with limited power, space, and heat dissipation [4]. Car software architectures use multiple layers - hardware interfaces, operating systems, middleware services, and applications - each serving specific purposes. Central computers handle the heavy computational work and coordinate between systems, while smaller zone controllers manage local functions and provide backup for critical safety operations.

Cars don't operate in isolation anymore. V2E communications transform individual vehicles into active participants in a larger, connected transportation system. Traffic lights, road signs, other cars, and internet services exchange information with cars. This opens possibilities of such functions as the avoidance of traffic jams, the anticipation of the time when some parts are to be maintained, and the communication with emergency forces in the case of an accident. It is incredible how technical it is, the processing of sensor data in the cameras, radar, LIDAR, as well as managing the communications, entertainment, and safety systems, all at the same time. Everything must happen in real-time without compromising safety or user experience.

Aspect	Traditional Architecture	Software-Defined Platform
Hardware Design	Fixed, model-specific	Unified, scalable platform
ECU Distribution	Distributed across the vehicle	Centralized domain controllers
Computing Power	Limited, function-specific	High-performance, multi-purpose
Update Capability	Hardware replacement required	Over-the-air software updates
Development Cost	High per model variation	Reduced through platform reuse
Time-to-Market	Extended development cycles	Accelerated deployment
Feature Integration	Hardware-dependent	Software-based cross-domain
Maintenance	Physical component replacement	Remote software maintenance
Scalability	Limited by hardware constraints	Flexible software expansion
Resource Utilization	Fixed allocation	Dynamic load balancing

Table 1: Industry Context and Platform Architecture Comparison [3, 4]

3. Safety-Critical Control Systems and Standards

Car safety standards exist for good reason - vehicle malfunctions can kill people. Software-defined vehicles must follow strict international safety rules, particularly ISO 26262 for functional safety and ISO/SAE 21434 for cybersecurity. These aren't just

guidelines - they're mandatory frameworks that define how software-controlled safety systems must work. Breaking these rules means cars can't be sold.

Systematic hazard analysis in object-oriented automotive development provides the framework for finding and preventing potential failures in safety-critical applications [5]. This means examining every possible way software could fail and building in protections. Safety-critical functions must maintain their integrity throughout the car's entire life, with comprehensive monitoring and fault detection running continuously in the background.

Component-based development has become essential for managing automotive software complexity while keeping safety intact [6]. Modern cars trace millions of software pieces simultaneously, and version control software forms an update and checks compatibility. Tasks that are safety critical have precedence to processing power and memory so that non-critical tasks, such as entertainment systems, do not disrupt the braking or steering.

The demands of real-time operating systems in vehicles and automobiles are extremely stringent in terms of timelines. The software that controls the anti-lock braking system has microseconds to react when the driver depresses the brakes. Any kind of delay might spell out the difference between a safe stop and a crash. Safety architectures use multiple backup systems - if one computer fails, others take over instantly. Triple redundancy means three separate systems constantly check each other. If one disagrees with the others, it gets overruled. Automatic failover is quite a rapid process, and the drivers are not even aware that such systems make the car secure even when the parts fail. There are continuous monitoring eyes on issues, and they can be used to close non-essential systems to be able to save some of the critical functions in case of need.

Component	Standard/Framework	Key Features	Safety Level
Functional Safety	ISO 26262	Hazard analysis, risk assessment	ASIL A-D
Cybersecurity	ISO/SAE 21434	Threat analysis, security measures	Security levels
Service Registry	Component-based	Version control, traceability	High integrity
Mixed-Criticality	Real-time scheduling	Priority management, isolation	Time-deterministic
Redundancy Systems	Triple Modular Redundancy	Fault tolerance, failover	Critical availability
Boot Process	Secure/Measured boot	Cryptographic verification	System integrity
Diagnostic Coverage	Safety monitoring	Anomaly detection, response	Comprehensive
Fault Tolerance	Multi-level redundancy	Hardware/software backup	Continuous operation
Version Management	Software lifecycle	Deployment control, audit	Complete traceability
Safety Architecture	Layered protection	Graceful degradation, shutdown	Mission critical

1) Table 2: Safety-Critical Control Systems Standards and Components [5, 6]

4. Networking and Communication Infrastructure

Car networks need split-second timing for safety functions. Unlike home internet, where a few milliseconds delay doesn't matter, automotive networks control brakes, steering, and stability systems, where timing means everything. Time-Sensitive Networking protocols guarantee that safety-critical messages get delivered on time, every time. Missing a deadline isn't just inconvenient - it could be fatal.

Modern automotive networking research shows how TSN implementations meet the demanding requirements of vehicle control systems [7]. Automotive Ethernet offers the speedy data highways that are required for advanced capabilities, with a high degree of accuracy in timing demanded by the safety system. Future specifications target even higher speeds to handle the massive data streams from autonomous driving sensors and real-time processing requirements.

Network architecture design in cars must work within tight constraints - limited power, space, and harsh environmental conditions [8]. Modern automotive networks support multiple data rates, with some connections running at gigabit speeds for high-bandwidth applications like camera feeds and sensor fusion. Network switches in cars handle millions of data packets per second while managing traffic bursts during peak usage periods.

Many legacy car functions continue to be served by older versions of the CAN bus protocol, and more recent versions of the CAN-FD and Ethernet protocol in SDV protocol allow higher-rate diagnostic systems. This is a hybrid strategy that allows manufacturers to slowly move away and use older technologies without necessarily having incompatibility with the current components. Security layers are provided to prevent cyberattacks by providing advanced encryption and authentication. Segmentation of the network ensures that important control systems cannot be connected to other less important processes, minimizing vulnerability and limiting damage in case security breaches are detected. All these types of networks are hard to manage and need seamless functioning and real-time performance, which poses great engineering challenges. Quality control mechanisms prioritize safety messages while ensuring entertainment and communication systems still get adequate bandwidth.

Technology	Protocol/Standard	Data Rate	Latency	Application
Time-Sensitive Networking	TSN	Variable	Microseconds to milliseconds	Safety-critical control
Automotive Ethernet	IEEE 802.3	Mbps to Gbps	Low deterministic	High-bandwidth data
Controller Area Network	CAN	kbps to Mbps	Moderate	Traditional functions
CAN Flexible Data-rate	CAN-FD	Enhanced Mbps	Improved	Enhanced diagnostics
Vehicle-to-Everything	V2X	Mbps	Real-time	External communication
Network Segmentation	VLAN/Firewall	Protocol dependent	Minimal impact	Security isolation
Encryption	AES-256	Data rate dependent	Minimal overhead	Data protection
Authentication	RSA	Session-based	Initial handshake	Identity verification
Quality of Service	Traffic prioritization	Full bandwidth	Prioritized	Service management
Legacy Integration	Hybrid protocols	Mixed rates	Backward compatible	Migration support

Table 3: Networking and Communication Infrastructure Technologies [7, 8]

5. Update Management and Validation Strategy

One of the largest benefits of software-defined vehicles is wireless software updates, but new risks are associated with this technology that should be managed carefully. The conventional car recalls are expensive, costing billions, and require months to be finished. Wireless updates can fix problems instantly across entire vehicle fleets, saving enormous amounts of money while improving cars continuously throughout their lifetimes.

Research on vehicular hardware security modules shows why robust security frameworks are essential for protecting update processes and maintaining system integrity during software deployments [9]. Update systems must coordinate changes across multiple vehicle computers containing hundreds of software components while keeping the car safe and functional during potentially long update procedures.

Advanced testing methods use digital twin technologies and resource-constrained sensor networks to create virtual copies of vehicle systems for comprehensive testing before releasing updates to real cars [10]. Feature flags allow controlled rollouts where new capabilities get activated gradually, with immediate rollback options if problems appear. Testing frameworks deploy updates to small vehicle populations first, monitoring performance across thousands of cars over weeks before broader releases.

Digital simulations run thousands of test scenarios simultaneously, cutting validation time from months to weeks while testing across diverse conditions that cars encounter in real-world use. Shadow mode testing runs new software alongside production systems, collecting data and validation without affecting vehicle operation, though this requires extra processing power. Staged deployment strategies start with carefully monitored small groups before expanding to larger fleets over extended periods. Rollback systems must respond quickly and reliably, reverting cars to known-good software when problems exceed acceptable failure rates. Testing must account for every possible driving condition - extreme weather, different road surfaces, varying traffic patterns - ensuring updates work reliably across the complete range of scenarios that vehicles face during their operational lifetime.

Strategy Component	Technology/Method	Coverage Scope	Risk Level	Implementation Phase
Digital Twin	Virtual simulation	Complete system	Low	Pre-deployment
Shadow Mode	Parallel execution	Production validation	Minimal	Testing phase
Feature Flagging	Controlled deployment	Selective activation	Managed	Gradual rollout
A-B Testing	Statistical validation	Limited population	Controlled	Pilot deployment
Staged Rollout	Phased implementation	Expanding fleets	Contained	Production release
Rollback Mechanism	Rapid reversion	System restoration	Emergency	Failure response
Update Orchestration	Multi-ECU coordination	Complete vehicle	Coordinated	System-wide
Security Framework	Protected deployment	Integrity maintenance	Secured	All phases
Validation Testing	Comprehensive scenarios	Diverse conditions	Thorough	Pre-release
Risk Containment	Failure threshold monitoring	Fleet management	Proactive	Continuous

Table 4: Update Management and Validation Strategy Framework [9, 10]

6. Conclusion

A fundamental paradigm shift of the movement to software-defined vehicle platforms is much more than technological enhancement. This revolution can allow the automotive manufacturers to no longer be confined to the traditional hardware, but can instead develop vehicles that are constantly evolving and changing during the lifespan of their operation. By combining the state-of-the-art computing platforms, deterministic network protocols, and dependable safety systems, a platform of unprecedented mobility solutions innovation is created. Economic advantages encompass high cost savings on both development and maintenance, whereas environmental benefits appear in the form of increased life cycle of the vehicles, and it has lowered electronic waste. Interoperability through software architectures, such as AUTOSAR standards, assures automotive software interoperability and is likely to accelerate the adoption of such transformative technologies throughout the industry. Safety is also of the utmost importance with strict compliance to the international standards and adding several layers of

redundancy, ensuring the availability of critical functions even in the case of component failures. The next areas of development are expected to be more integration of artificial intelligence, better human-machine interfaces, and the enlarged connectivity to smart infrastructure systems. Further cooperation between automotive companies, technology vendors, and governmental agencies is necessary to implement the software-defined vehicle platforms successfully without hindering the innovation process at the expense of safety, security, or reliability. The essence of this evolution is that vehicles and their users are becoming redefined; cars no longer remain as inert mechanical systems but are dynamic and intelligent platforms that can learn continuously and adjust to the shifting needs and circumstances.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Manfred Broy, "Challenges in automotive software engineering," ICSE '06: Proceedings of the 28th international conference on Software engineering, 2006, pp. 33-42. Available: https://dl.acm.org/doi/10.1145/1134285.1134292

[2] SRM Technologies, "AUTOSAR in Automotive Systems: An In-Depth Exploration," 2024. Available: https://www.srmtech.com/knowledge-base/blogs/autosar-in-automotive-systems-an-in-depth-exploration/

[3] Alexander Pretschner et al., "Software engineering for automotive systems: A roadmap," Future of Software Engineering (FOSE 07), 2007. Available: https://ieeexplore.ieee.org/document/4221612

[4] Lothar Thiele et al., "Design Space Exploration of Network Processor Architectures," ResearchGate, 2003. Available: https://www.researchgate.net/publication/315387952_Design_Space_Exploration_of_Network_Processor_Architectures

[5] P. Johannessen et al., "Hazard analysis in object-oriented design of dependable systems," 2001 International Conference on Dependable Systems and Networks, 2001, pp. 507-512. Available: https://ieeexplore.ieee.org/document/941436

[6] Mikael Åkerholm et al., "The SAVE approach to component-based development of vehicular systems," Journal of Systems and Software, 2007. Available: https://www.sciencedirect.com/science/article/abs/pii/S0164121206002226

[7] Mohammad Ashjaei et al., "Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities," Journal of Systems and Software, 2021. Available: https://www.sciencedirect.com/science/article/pii/S1383762121001028

[8] Wilfried Steiner, "An Evaluation of SMT-Based Schedule Synthesis for Time-Triggered Multi-hop Networks," 2010 31st IEEE Real-Time Systems Symposium, 2010. Available: https://ieeexplore.ieee.org/document/5702246

[9] Marko Wolf and Timo Gendrullis, "Design, Implementation, and Evaluation of a Vehicular Hardware Security Module," International Conference on Information Security and Cryptology, 2012. Available: https://www.researchgate.net/publication/264946050_Design_Implementation_and_Evaluation_of_a_Vehicular_Hardware_Security_Module

[10] Christian Buckl et al., "Services to the Field: An Approach for Resource Constrained Sensor/Actor Networks," 23rd International Conference on Advanced Information Networking and Applications, AINA 2009, Workshops Proceedings, Bradford, United Kingdom, 2009. Available: https://www.researchgate.net/publication/221191009_Services_to_the_Field_An_Approach_for_Resource_Constrained_SensorActor_Networks