Architecting Automotive Software for the Emerging Compute-in-the-Network Paradigm

Saad Mubeen
Department of Computer Science and Engineering
Mälardalen University, Sweden
Västerås, Sweden
saad.mubeen@mdu.se

Mohammad Ashjaei
Department of Computer Science and Engineering
Mälardalen University, Sweden
Västerås, Sweden
mohammad.ashjaei@mdu.se

John Lundbäck
Arcticus Systems, Sweden
Stockholm, Sweden
john.lundback@arcticus-systems.com

Abstract

Emerging compute-capable automotive network switches now allow parts of application functionality to be hosted and executed within the communication fabric, complementing the primary application software execution on electronic control units. This compute-in-the-network paradigm offers opportunities for reducing latency and improving responsiveness in time-critical automotive systems. However, existing automotive architecture description languages and component models lack abstractions for representing application software residing in network elements, limiting accurate design and analysis. This paper motivates the need to treat the models of onboard network switches as the first-class software architectural entities and outlines research opportunities including unified abstractions, end-to-end timing models extraction, alignment with standards and toolchain interoperability, and addressing new cybersecurity challenges.

CCS Concepts

• **Computer systems organization → Real-time systems**.

ACM Reference Format:
Saad Mubeen, Mohammad Ashjaei, and John Lundbäck. 2026. Architecting Automotive Software for the Emerging Compute-in-the-Network Paradigm. In *2026 IEEE/ACM 48th International Conference on Software Engineering (ICSE-Companion '26), April 12–18, 2026, Rio de Janeiro, Brazil.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3774748.3795872

1 Introduction

Automotive software in a modern car is distributed over several tens of computational nodes, also called the Electronic Control Units (ECUs). The nodes, where the automotive application software resides, may communicate with each other over five or more different types of onboard networks [7]. The software architectures of these systems are modeled using the principles of model-based development and component-based software engineering [6, 10, 11]. Examples of automotive software Architecture Description Languages (ADLs) and component models include EAST-ADL [3, 9], AUTOSAR [1, 4], and Rubus Component Model (RCM) [5, 8].

ICSE-Companion '26, Rio de Janeiro, Brazil

ACM ISBN 979-8-4007-2296-7/26/04
https://doi.org/10.1145/3774748.3795872

To provide perspective, consider examples of software architecture modeling using the automotive industrial component model Rubus, shown in Fig. 1. The figure depicts the models of (a) software component, (b) node and its internal software architecture, (c) network and messages that are sent by software components, signals and signal database providing information of signal-to-message mapping, (d) network element, e.g., Time Sensitive Networking (TSN) [12] switch, and (e) software architecture of a two-node distributed automotive embedded system. Similar modeling examples can be drawn using other languages and models like EAST-ADL and AUTOSAR. The software architecture of the application within a node consists of software components (yellow boxes) with well-defined interfaces and behaviors (both residing within the software components). The interfaces consist of various types of ports including trigger or activation ports, data ports for intra-node communication and network ports for inter-node communication. The network model represents the physical network that defines communication elements, including messages and their properties, and a set of signals mapped to each message according to protocol-specific encoding. The switch models are used in the case of switched Ethernet-based protocols like TSN to encode the switching and routing logic. Note that the models of network and underlying elements do not include any automotive application software.

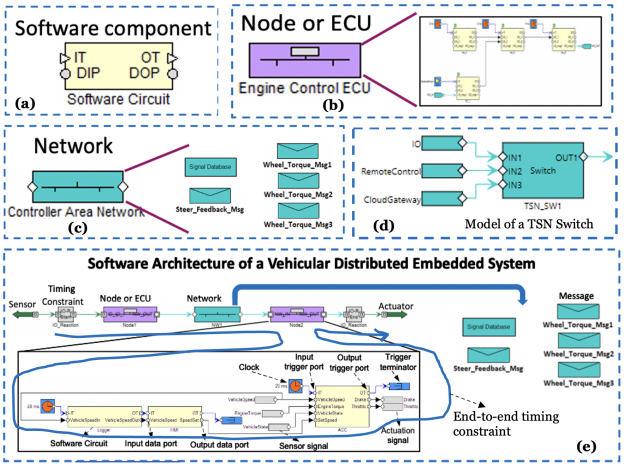

Figure 1: Examples of software architectural modeling with an automotive industrial component model (Rubus).

2 Need for Extending Software Architecture into the Network

The automotive application software architecture resides within the models of nodes or ECUs as shown in Fig. 1. **Historically, modeling the software architecture of automotive applications within the network and its underlying elements, such as network switches, was not required.** Traditional switches included configuration software and operated as passive communication devices that simply forwarded frames or packets based on protocol logic, without a support for hosting and executing any application software. Application-level functionality was confined to only nodes. As a result, automotive ADLs and component models focused exclusively on nodes, while models of networks and switches were limited to capturing, network configuration, switching and routing, timing, bandwidth, and mapping information for messages and signals that are sent/received by the automotive application software components residing in the nodes.

This paradigm has recently started to change across distributed embedded software systems in the automotive and automation domains. In many applications, these domains demand low-latency computation and communication. For instance, autonomous vehicles operating in confined environments such as autonomous mines or autonomous recycling sites require real-time decision-making that exceeds the capabilities of traditional ECUs with limited computational power. Even when a private edge or on-site cloud infrastructure is available, offloading such computation can still introduce unacceptable latency for time-critical applications. Similarly, industrial control loops and other safety-critical functions also depend on minimizing the physical and logical distance between data acquired from sensors and computation to meet stringent real-time requirements.

To address these requirements, manufacturers are introducing network switches with sufficient capability to host and execute automotive application software traditionally deployed on ECUs. This trend is exemplified by Time-Sensitive Networking (TSN) standards, in which modern switches not only provide deterministic switching and scheduling but are also becoming capable of hosting and executing application software [13, 14]. The increasing computational capacity of these industrial-grade switches now makes it feasible to run application-level software directly inside network elements. Notable benefits of supporting application software and its execution in the network switches include: (i) latency reduction, i.e., co-locating processing with forwarding shortens paths and avoids external offloading, and (ii) security and resilience, i.e., localized, isolated execution can shrink the attack surface and reduce remote infrastructure dependency.

To fully realize the benefits of these technological advancements in the automotive networks, software architectural abstractions must be extended into network elements. To the best of our knowledge, the existing automotive ADLs and software component models do not support architectural abstractions for the application software residing within network switches or related network infrastructure. This limitation hampers the modeling and analysis of distributed automotive embedded software systems that utilize the advanced onboard networking devices where computation is integrated with the communication.

3 Discussion and Research Opportunities

We advocate that network switches be treated as first-class modeling elements on par with the models of nodes (ECUs). This perspective makes it possible to model their internal application software architectures, including software components, interfaces, interactions, and resource or timing semantics. These aspects have been missing from the existing automotive ADLs and component models. In addition to this, the automotive ADLs would continue to support the legacy software models of switches that already capture switching behavior, configuration, and scheduling. Such an extension is needed for modeling the software architectures of the next-generation automotive distributed embedded systems in which computation is integrated into the communication fabric.

We identify the following research opportunities for extending existing automotive ADLs and software component models.

- **Unified architectural abstractions for switches as first-class elements.** Develop modeling constructs for compute-capable switches that capture both network-protocol-dependent concerns (such as switching, configuration, and scheduling) and the internal application software architectures executed within the switch, including explicit software components, interfaces, and interconnections. These constructs should provide an abstraction that elevates the switch models to first-class architectural elements without overfitting them to traditional node models.
- **End-to-end timing model extraction with in-switch computation.** To perform timing analysis of distributed automotive software architectures, their end-to-end timing models must first be extracted and then provided as input to the analysis tools [2]. These timing models also include network-level linking and timing information that constructs the distributed software component chains. For example, one such chain is identified with the blue arrow that traverses the models of two nodes in Fig. 1(e). An end-to-end reaction-time constraint is also specified on the chain according to the AUTOSAR standard. An interesting challenge is to enable systematic extraction of end-to-end timing models from distributed automotive software architectures that incorporate both node-level software components and software components within compute-capable switches. Such timing models must capture the full software component chains, including network-level linking and communication timing information. With the introduction of in-switch computation, traditional timing models extraction methods that only consider node software must be extended to account for software architecture of the part of application embedded in the network fabric. A key research opportunity lies in developing methods and tooling for extracting consistent, analyzable end-to-end timing models that span nodes, switches, and their interactions.
- **Alignment with standards and toolchain interoperability.** Define extensions to network and switch models that maintain backward compatibility while enabling automated transformations into existing configuration, code-generation, and analysis workflows. A key challenge is to identify which extension mechanisms best align with standards such as AUTOSAR, ADLs like EAST-ADL, and industrial component models like Rubus, ensuring that enhanced switch abstractions integrate smoothly into current toolchains without introducing fragmentation.

- **Cybersecurity challenges introduced by in-switch computation.** Integrating automotive application software into network switches expands the system's attack surface, as the software has traditionally been confined to nodes only, and introduces security risks that traditional switch models do not capture. A research opportunity lies in defining modeling abstractions and analysis methods that represent these new vulnerabilities, support threat assessment for compute-capable switches, and guide the design of appropriate protection and isolation mechanisms within automotive ADLs and software component models.

Acknowledgments: This work is supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA) via the INTERCONNECT and FLEXATION projects, and by the Swedish Knowledge Foundation (KKS) via the SEINE project. The authors thank the industrial partners: HIAB, Arcticus, ABB, and Westermo.

References

[1] AUTOSAR Standards. 2025. AUTOSAR - Aautomotive Open System Architecture. http://autosar.org

[2] B. Houtan, M. Onuer Aybek, M. Ashjaei, M. Daneshtalab, M. Sjödin, J. Lundbäck, S. Mubeen. 2023. End-to-end Timing Modeling and Analysis of TSN in Component-Based Vehicular Software. In *26th IEEE International Symposium on Real-Time Distributed Computing (ISORC).*

[3] EAST-ADL Association. 2013. EAST-ADL Domain Model Specification. In *Version V2.1.12.* http://www.east-adl.info/Specification/V2.1.12/EAST-ADL-Specification_V2.1.12.pdf

[4] S. Fürst and M. Bechter. 2016. AUTOSAR for Connected and Autonomous Vehicles: The AUTOSAR Adaptive Platform. In *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshop (DSN-W).*

[5] K. Hänninen et al. 2008. The Rubus Component Model for Resource Constrained Real-Time Systems. In *IEEE Symposium on Industrial Embedded Systems.*

[6] K. Petersen, D. Badampudi, S. M. A. Shah et al. 2018. Choosing Component Origins for Software Intensive Systems: In-House, COTS, OSS or Outsourcing?—A Case Survey. *IEEE Transactions on Software Engineering* 44, 3 (March 2018), 237–261.

[7] M. Ashjaei, L Lo Bello, M. Daneshtalab, G. Patti, S. Saponara, S. Mubeen. 2021. Time-Sensitive Networking in automotive embedded systems: State of the art and research opportunities. *Journal of Systems Architecture* 117 (2021), 102–137.

[8] S. Mubeen, H. Lawson, J. Lundbäck, M. Gålnander, and K. L. Lundbäck. 2017. Provisioning of Predictable Embedded Software in the Vehicle Industry: The Rubus Approach. In *IEEE/ACM 4th International Workshop on Software Engineering Research and Industrial Practice (SER&IP).*

[9] R. T. Kolagari, D. Chen, A. Lanusse, R. Librino, H. Lönn, N. Mahmud, C. Mraidha, M. Reiser, S. Torchiaro, S. Tucci-Piergiovanni, T. Wägemann, N. Yakymets. 2015. Model-Based Analysis and Engineering of Automotive Architectures with EAST-ADL: Revisited. *Int. J. Concept. Struct. Smart Appl.* 3, 2 (2015), 25–70.

[10] S. Mubeen, M. Ashjaei. 2025. Problem-Based Learning in an Educational and Training Module on Model-Based Development of Vehicle Software *(EASE Companion '25).* Association for Computing Machinery, New York, NY, USA, 10 pages.

[11] T. Vale, I. Crnkovic and E. Santana de Almeida et al. 2016. Twenty-eight years of component-based software engineering. *Journal of Systems and Software* 111 (2016), 128 – 148.

[12] TSN Task Group. 2020. Time-Sensitive Networking (TSN) Task Group. In *IEEE 802.1 Standards.* https://1.ieee802.org/tsn.

[13] Vector. 2025. MICROSAR Switch – Scalable Software for Switch-ECUs. *Smart Solutions for Smart Switches* (2025). https://www.vector.com/se/en/products/products-a-z/embedded-software/microsar-switch/.

[14] Westermo. 2025. Application Hosting. *WeOS 5.26.1* (2025). https://docs.westermo.com/weos/weos-5/Container/App-Hosting.html.