Last Exit Open Source

From Taboo to Foundation of the Software-Defined Vehicle Platform

Michael Pöhnl

2026-08-06

Executive Summary

For today's vehicle manufacturers, the software-defined vehicle (SDV) platform is strategically critical. It determines how fast new software-driven applications can be developed, brought into production, and updated in the field.

At the core of this platform is the foundational software layer on top of which customer-differentiating applications run. This layer includes capabilities that are not unique to one manufacturer, but common across the industry.

Over the past more than 10 years, several approaches have been tried to create this SDV platform foundation: standardization, proprietary supplier products, and in-house development. Each addressed a real need, but it became clear that this layer is too strategic to be delegated as a black box, and too common and complex for every OEM to reinvent in isolation.

Industrial-grade open source is now emerging as the model that can combine OEM architectural ownership with mature shared building blocks developed and maintained by subject-matter experts. The strategic question for OEMs is no longer whether open source is allowed, but which parts of the SDV platform foundation should be built on industrial-grade open source.

The SDV Platform: A Deciding Factor for Today's OEMs

The architectural shift behind the SDV is well known: for decades, vehicle software was distributed across many so-called ECUs (electronic control units) and tightly coupled with the underlying hardware, including sensors and actuators. Together, hardware and software provided one dedicated function, such as engine control or anti-lock braking.

With software-defined vehicles, this model changes. New and innovative parts of the software are increasingly moving to centralized vehicle computers. The applications running on these machines are no longer software components tightly coupled to one piece of hardware, built once and left untouched for the vehicle's lifetime. They are customer-differentiating applications, such as automated driving functions or personalized cockpit experiences, that integrate data from many sources and evolve through over-the-air updates.

In this whitepaper, we use the term SDV platform for the software environment that enables vehicle applications to be developed, integrated, validated, deployed, executed, and updated in the field. This platform consists of several layers. Some live outside the vehicle and deal with aspects such as development tooling, cloud-based data management, simulation, and validation infrastructure. Others are part of the software stack running directly on the vehicle computers.

The main focus of this whitepaper is the foundational software layer of the in-vehicle stack, often broadly referred to in automotive as "middleware." This SDV platform foundation includes aspects such as communication, execution management, and lifecycle management. It decides whether applications can run efficiently, deterministically, and reliably on real embedded hardware.

The SDV platform has become a deciding factor because it determines the length of development and update cycles and, ultimately, how fast new software-enabled features can be brought to customers. OEMs that master their SDV platform can turn this into a structural advantage. Those that do not will struggle with inefficiencies across its layers, slowing down innovation and, in severe cases, even shifting start of production.

The SDV platform is an asset of strategic importance and one of the key pillars for future success. It is not just an engineering detail. And at the core of every SDV platform is its foundation.

Lessons from Past Approaches to the SDV Platform Foundation

As early as 2013, long before the term software-defined vehicle was coined, it became clear that a next generation of high-performance computers would enter automotive architectures and that a new foundational software layer would be needed. These computers would be based on POSIX operating systems rather than AUTOSAR Classic OS, which was the de facto standard at the time. As traditional OEMs were used to outsourcing most software development to suppliers, the following approaches were the most obvious paths toward an SDV platform foundation.

- **AUTOSAR Adaptive:** After AUTOSAR Classic had become the cross-OEM and supplier standard for microcontroller-based ECUs, the consortium started specifying the Adaptive Platform in 2016 as the next generation targeting vehicle computers with POSIX operating systems. Compared to the previous Classic Platform, this work included, for the first time, a members-only reference implementation. Otherwise, it can still be considered a specification-first approach, followed by the development of proprietary implementations sold by suppliers based on license models. With AUTOSAR's design-by-committee process trying to find compromises across many members with different interests, while also preserving compatibility with the existing Classic world, the initiative struggled to deliver the speed, innovation, and functionality many had expected for the SDV era.
- **Proprietary products:** A faster and, from an architectural perspective, more coherent path seemed to be proprietary solutions designed and implemented by a single team. Many of these solutions started from existing products or included lessons learned from adjacent industries such as robotics. While there are capabilities that address common needs across vehicle manufacturers, most manufacturers still have individual needs and preferences shaped by vehicle platforms that evolved over decades. These needs made it hard to provide an off-the-shelf SDV platform foundation and pushed vendors of commercial

products toward addressing many different use cases with one solution. But the biggest challenge of commercial solutions turned out to be lock-in to a non-standardized proprietary ecosystem, which vehicle manufacturers want to avoid for the strategically important foundation of their SDV platform.

- **In-house development:** Since some disadvantages of the approaches discussed above were foreseeable, in-house development of the SDV platform foundation seemed reasonable. Owning it internally promised independence from suppliers, faster decisions, and tighter integration with vehicle programs. Some newer manufacturers have shown that strong internal software ownership can become a competitive advantage. For most traditional OEMs, however, this meant hiring large new teams because the required software expertise was not yet available internally at the needed scale. These teams were then confronted with high expectations from many stakeholders and a legacy world that still had to be integrated. Together with limited experience in the development of performance- and safety-critical embedded software and the ambition to build every detail from scratch, many of these initiatives struggled, were reset, or resulted in an SDV platform foundation that did not last longer than one production program.

The lesson from these approaches is not that they were irrational. Each of them addressed a real need like standardization, supplier leverage, or internal control. But for traditional automotive OEMs, none of them became the answer for building the SDV platform foundation at the required speed, functionality, and level of strategic control.

Often, shortcomings within the foundation layer led to inefficient development workflows and performance issues, ultimately contributing to delays in bringing functionality to the road and updating it in the field.

The Rise of Open Source as a True Alternative

At the end of the last century and the beginning of the current one, automotive software was purpose-built for specific problems and tied to specific hardware. The use of open-source software was not really a question to consider, as there were simply no open-source solutions available for many of the dedicated problems the industry had. When this started to change, open source was initially seen as a threat. This was based on two common misconceptions.

The first misconception was that using open-source software would automatically introduce license complications, especially due to copyleft obligations from licenses such as the GPL. The second was that open-source software was assumed to be written by developers unfamiliar with automotive and safety requirements, following informal “happy hacking” practices rather than disciplined engineering aligned with standards such as ISO 26262. As a result, internal development guidelines often explicitly prohibited the use of open-source software for code running in the vehicle.

What happened in the meantime can be described as both a change in reality and a complete turnaround in thinking. First, the amount of software in vehicles has increased significantly. Most new functionality is software-driven today, and the problem space increasingly overlaps with other domains such as robotics and IoT. This means that there are now common problems in the automotive industry for which open-source solutions exist. In addition, organizations such as the Eclipse Foundation played an important role in educating the automotive industry about open-source governance, licensing, collaboration models, and the practical use of open source in production software.

If we combine this development with the experience and limitations of the previously discussed approaches, the use of open source in the foundation of SDV platforms is no longer a thought experiment. It is a true alternative.

- **Code First, Not Committee First:** Open-source projects are typically driven by a core team of experts following a code-first approach, not a specification-first or design-by-committee model that can make standardization activities slow and overly broad.
- **Open Ecosystem, Not Closed Lock-In:** Open-source projects with permissive licenses provide an open and unrestricted ecosystem. They do not lock OEMs into a black-box solution with a closed ecosystem.

- **Proven Foundations, Not Reinvention:** Open-source projects can provide mature and proven solutions for common problems and help avoid costly and time-consuming in-house development of non-differentiating but technically demanding infrastructure.

Open source is not automatically faster or better. It only becomes a viable strategy when it is industrial-grade open source, developed professionally according to industry-wide best practices. Mature open-source projects in this category go far beyond public code. They come with clear maintainers, release management, CI/CD pipelines, transparent issue handling, security processes, SBOMs, and a visible quality culture. For automotive, additional aspects such as suitability for embedded constraints, predictable behavior, and a path toward safety artifacts matter as well.

Eclipse S-CORE: Flagship Initiative with Structural Tensions?

Against this backdrop, the Eclipse S-CORE initiative seems to be a logical consequence and, at the same time, a bold move by the involved OEMs and suppliers. On paper, it reflects many of the right ideas: collaboration across OEMs and suppliers, a shared SDV platform foundation for non-differentiating infrastructure, and a move away from proprietary lock-in toward transparent and community-driven development. What seems logical and economically sound from a vehicle manufacturer's perspective is, for suppliers that expected a full SDV platform to become a major future revenue stream, less a planned evolution and more an adaptation to changing realities.

With OEMs involved, S-CORE offers a path toward production use that earlier initiatives often could not provide. The goal is to avoid repeated development on the supplier side and repeated integration efforts on the OEM side. The initiative aims for a functional-safety-compliant development process, but explicitly states that S-CORE is not a ready-to-integrate series product and that compliance with standards such as ISO 26262 remains the responsibility of the series project.

A foundational software layer as targeted by Eclipse S-CORE needs a strong technical vision, a coherent architecture, strict scope control, and excellent building blocks to become a solid base for performance- and safety-critical applications running on top of it. While developing such a production-ready software stack is hard, the biggest challenges for S-CORE may be on the organizational and business side.

- **The organizational challenge:** Eclipse S-CORE follows a shared governance model under the Eclipse Foundation, with core members jointly shaping the project's direction. As many participating organizations have been involved in AUTOSAR in the past and have non-aligned commercial interests, there is a risk of slow progress. More importantly, the strong technical ownership and architectural coherence needed for a production-grade platform foundation may be difficult to preserve.
- **The business challenge:** One envisioned commercialization path around Eclipse S-CORE is a distributor model in which suppliers provide commercial distributions, downstream services, production-specific adaptations, and extensions for OEM programs using the jointly developed SDV platform foundation. This is a reasonable part of the wider ecosystem. But if there is no monetization around the platform foundation itself, incentives can become misaligned. The ecosystem may then reward downstream gap-closing more than contributing the best possible building blocks to the shared core, weakening incentives to compete on its completeness, quality, and innovation.

S-CORE's model of providing an open-source SDV platform foundation that distributors extend and industrialize for specific OEM programs makes sense. But it can only work if the shared platform foundation is strong enough. The organizational model must preserve technical ownership and architectural coherence, while the business model must reward improvements to the shared core itself, not only services and extensions around it.

Viable Open-Source Models for the SDV Platform Foundation

As the SDV platform is key to today's competitiveness of a vehicle manufacturer, OEMs should fully own it. This includes the foundational software layer on top of which applications run. This layer is too important

for OEMs to delegate responsibility to a supplier or standardization body. But it is also too common for isolated reinvention, which would require enormous expertise and investment.

This remains true even as AI-assisted development becomes more powerful. The hard part is not only writing code, but defining, validating, maintaining, and owning a production-grade platform foundation over many vehicle programs.

The right middle ground is a strategy in which the OEM owns the architecture and uses industrial-grade open-source building blocks where differentiation is low and complexity is high. This allows OEMs to:

- lower costs compared to full in-house development
- lower risk by using proven building blocks and working with the experts behind them
- reduce the total time required compared to starting from a blank sheet

These building blocks are mature, proven open-source projects with high-quality development, an active maintainer and user ecosystem, and an architecture and implementation that fit the use case. OEMs then have a natural interest in the sustainable further development and maintenance of these projects by collaborating with the experts behind them. But these investments become shared investments into common open-source building blocks.

In an open and healthy ecosystem, traditional suppliers and new specialized startups compete on functionality, speed, quality, support, and industrialization of open-source building blocks. This can include commercial offerings for production use, such as integration support, production extensions, long-term support, and safety artifacts.

There are different possible variations of how exactly this ecosystem can work. They differ in how openly, collaboratively, and inclusively OEMs want to build the SDV platform foundation whose architecture they ultimately control. Possible realizations include:

- **Proprietary platform with open-source building blocks:** This is the most pragmatic model. The SDV platform foundation is developed in-house by the OEM, but not every infrastructure component is built from scratch. Mature open-source projects allow the OEM to benefit from existing engineering work, transparent source code, and the expertise of the project maintainers. Development teams can focus more energy on platform integration and vehicle-specific differentiation instead of rebuilding infrastructure that already exists. Some newer manufacturers have successfully followed this path, and the model can also be found in many other industries.
- **OEM-controlled open-source project:** This model goes further, as one OEM or a group of aligned OEMs creates an open-source ecosystem around the SDV platform foundation. Unlike broader shared-governance initiatives, the OEM defines the architecture, interfaces, priorities, and acceptance criteria, while suppliers and startups compete on the best building blocks. The OEM rewards the strongest building blocks by sponsoring dedicated projects and the maintainers behind them. The result is a more dynamic ecosystem that incentivizes innovation at the component level without surrendering control over the platform architecture.
- **OEM-funded organization with technical ownership:** In this model, OEMs share costs and fund an organization that controls the overall technical development of the SDV platform foundation. This is explicitly not design by committee. Large parts of the resulting stack can again be based on mature open-source projects, which the organization sponsors and integrates into the platform foundation under clear technical ownership. Other assets, such as certification artifacts or long-term maintenance packages, can be available only to members. This creates a sustainable funding model for work that is essential but difficult to finance through pure community contribution.

Conclusion: Last Exit Open Source

The SDV platform has become a strategic capability for OEMs. It determines how fast software-defined functionality comes to life and how well vehicles can evolve after start of production. Its foundational software layer determines whether vehicle computers can run complex workloads efficiently and predictably.

The industry has tried multiple approaches to create this SDV platform foundation. The AUTOSAR consortium defined a standard, suppliers offered proprietary solutions, and OEMs tried to build it internally. Most of these initiatives struggled and did not deliver the needed functionality and desired control in time.

Open source is becoming part of every viable future model. Not because software should be free, but because the non-differentiating platform foundation is too expensive, too complex, and too important for every manufacturer to rebuild in isolation. Eclipse S-CORE is the first major industry move toward a jointly developed platform foundation based on open source, while still showing that the full potential of open source depends on the right governance and incentive model.

The SDV platform needs to be owned by OEMs, which also means they have to control its foundation. That responsibility cannot be delegated away. But no matter how exactly this control looks in practice, OEMs do not need to build every non-differentiating infrastructure layer alone.

Industrial-grade open source provides the missing model for reducing time, effort, cost, and risk. For this model to work, OEMs need to finance the open-source projects they want to rely on. Suppliers and startups behind these projects compete to be selected for the platform foundation, while OEM funding keeps the shared building blocks maintained, improved, and production-ready.

The strategic question for OEMs is no longer whether open source is allowed. The question is which parts of the SDV platform foundation should be built on industrial-grade open source, which expert communities should be supported, and where proprietary differentiation actually begins.

Open source is no longer taboo. Industrial-grade open source is becoming the foundation of the SDV platform.

