Theme Article: Special Issue on Tech Predictions 2025

Navigating the Future: Essential Considerations for the Engineering of Software-Defined Vehicles

Rodi Jolak, *RISE Research Institutes of Sweden, Sweden*
Vard Antinyan, *Volvo Trucks, Sweden*
Alexander Åström, *Volvo Technology AB, Sweden*
Darko Durisic, *Volvo Car Corporation, Sweden*
Oliver Kopp, *University of Stuttgart, Germany*
Stefan Kriebel, *FEV.io, Germany*
Daniel Krippner, *ETAS GmbH, Germany*
Mazen Mohamad, *RISE Research Institutes of Sweden, Sweden.*
Jérôme Pfeiffer, *University of Stuttgart, Germany*
Chris Seiler, *Mercedes-Benz AG, Germany*
Pontus Svenson, *RISE Research Institutes of Sweden, Sweden.*
Andreas Wortmann, *University of Stuttgart, Germany*
Jan Bosch, *Chalmers University of Technology, Sweden*

Abstract—Software Defined Vehicles (SDVs) represent a transformative shift in the automotive industry, moving from traditional vehicle development to software-driven mobility solutions. The engineering of SDVs is inherently complex and involves numerous technological and organizational challenges. Future challenges include increased software complexity, managing software updates, cultural gaps, and complying with international standards and regulations. Additionally, the rapid pace of technological change demands an efficient and effective engineering process. To remain competitive and achieve their business goals, automotive organizations must adopt new strategies and undergo significant changes in the following software development concerns: architecture, process, and organization. These changes are necessary to boost productivity and improve the efficiency of the engineering process of SDVs.

Software Defined Vehicles (SDVs) [1] represent a significant evolution in the automotive industry, transitioning from traditional hardware-centric vehicles to software-driven mobility solutions. This shift is driven by the growing demand for (i) advanced features such as advanced driver-assistance systems (ADAS) and predictive maintenance, (ii) connectivity such as over-the-air (OTA) updates, cloud services integration, and vehicle-to-everything (V2X) communication, and (iii) automation capabilities such as self-driving and self-parking in modern vehicles. SDVs leverage software (Figure 1) to manage and enhance vehicle operations, allowing for continuous updates and improvements throughout the vehicle's lifecycle. This enhances the functionality and user experience. Moreover, it opens new avenues for innovation and business models within the automotive sector.

The engineering of SDVs is inherently complex and presents numerous technological, organizational, and cultural challenges. One of the primary challenges is

Digital Object Identifier 10.1109/XXX.0000.0000000

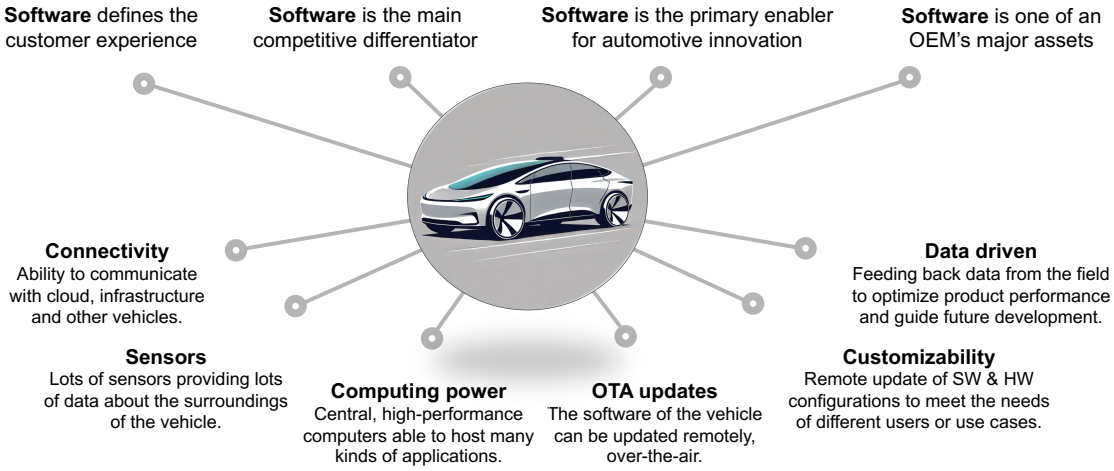

FIGURE 1. The Concept of Software-Defined Vehicles (SDVs)

increased software complexity. As vehicles transition from hardware-centric to software-driven systems, the amount of code and software components grow steadily. In addition, managing software updates, cultural gaps, and maintaining compliance with international standards and regulations are critical concerns that require robust and efficient processes. The rapid pace of technological change further exacerbates these challenges, which demands an adaptable and effective engineering process to keep up with evolving requirements and market expectations.

Several studies investigate the challenges associated with the engineering of SDVs. Goswami [2] assesses the state of SDVs and explores the current trajectory and execution challenges of SDVs, highlighting the need for a comprehensive approach to software development and integration. Keßler et al. [3] discuss technical and organizational challenges to the engineering of SDVs. They emphasize the need of careful selection and implementation of software and electronic/electrical (E/E) architectures, with a focus on virtualization and containerization to ensure encapsulation, independence, and security. In addition, they highlight the need for agile and accelerated software development processes, highlighting the role of DevOps in improving development speed and quality. Zhang et al. [4] review the technological advancements of SDVs and their impact on the automotive industry's value ecology. They state that the outdated research and development (R&D) model, backward enterprise capability, and the combination industrial ecological resources from different companies are three major challenges that hinder the transformation of the enterprise capability and the realization of Software-Defined Vehicle (SDV).

According to the BAPO model [5] (Figure 2), there are four software engineering concerns that help organizations ensure their structure and operations are aligned with their strategic objectives, leading to more effective and innovative outcomes. **B**usinees is the most critical concern, and business goals must be established correctly from the start. The next concern is to establish the **A**rchitecture, including the technology choices and structure to develop systems. The architecture then serves as the foundation for the **P**rocess i.e., ways of working and development practices. Finally, the **O**rganization, including the teams and departments, accomodate and follow the process.

To remain competitive and achieve the business goals of their SDV endeavor, we argue that automotive OEMs must adopt new strategies and undergo significant changes in the interdependent software engineering concerns: architecture, process, and organization. In this article, we outline the business goals associated with SDVs and highlight their implications on the other engineering concerns by offering our insights into the challenges, opportunities, and future of SDV engineering.

Business Goals

As in many other business domains, there is a need for the automotive industry to adapt to a changing world. Of particular importance for automotive companies are the shift from internal combustion engines to electric power trains and the increased focus on software and

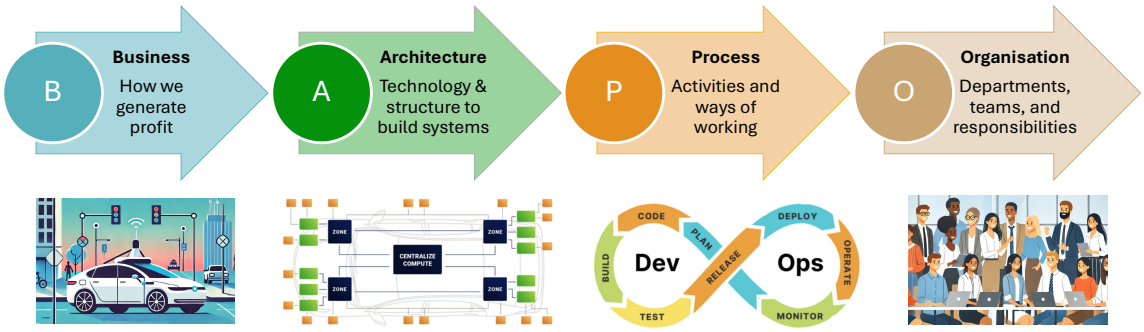

FIGURE 2. The BAPO model: Structuring organizations to ensure that business goals drive all the other engineering concerns.

connectivity.

Being able to enhance the vehicle users' experience by updating the software represents a major shift in business goals. Instead of developing new vehicle models every few years, the shift to SDV's give the possibilities to add new features continuously during an individual vehicle's lifetime. Coupled with connectivity it also provides the opportunity for data-driven feature development. Here data from cars in the field can be leveraged, e.g., to perform A/B testing to investigate different new features, as well as to evaluate new features with data from real-world scenarios.

SDVs also enable a software-driven aftermarket, where the OEM, as well as third parties, could sell software components that enable new functionality. It is likely that much of this new functionality will be driven by new ways of increasing user satisfaction and safety, e.g., by taking advantage of connected infrastructure and vehicles.

These shifts, however, bring with them an increased cost for software development and maintenance. There is a need to have common software baselines and components, with continuous updating, and that are reused across vehicle models and even types, and over a long period of time. To enable continuous updating, it is also important to future-proof the computer hardware in the vehicle: it is no longer possible to look for the simplest hardware that can meet the requirements. The hardware must be able to accommodate the needs of software that will be developed a few years into the future.

Predictive maintenance and monitoring of vehicle status is another important business driver for SDV. This is of particular importance for trucks and heavy-duty vehicles, where the fleet owner faces significant losses if the vehicle is unable to perform.

Software-defined vehicles enable continuous software updates, enabling new features, data-driven development, and a software-driven aftermarket where OEMs and third parties can sell software-based functionalities. These business benefits come with associated increases in costs for software development and maintenance, underscoring the necessity for revenue models to offset these additional expenses.

Architecture

Long-Living Complex Architectures

One challenge is the uncertain future requirements for SDVs during their complete lifetime that can span up to 30 years including design, production, and usage. Currently, it is unclear what the customer requirements for SDVs will be in 10 years from now, although this affects infrastructure, architecture, and supply chains. For instance, the computational power of the hardware needs to be over-provisioned such that it can support future functionality. Furthermore, stricter legal requirements, e.g., by the EU, like "over the air update" management regulation [6] or cybersecurity regulations [7], oblige automotive manufacturers to redefine entire concepts. Additional important challenges are cultural: Automotive manufacturers need to overcome the grown structures and silos of their heritage and aim for generic software solutions that serve for more than one use case and are viable for long-term use. To this end, efficient cross discipline collaboration and management (system, software, electric/electronic, mechanical) is required. This leads to additional technical challenges. Today, there are many dependencies, that may not be documented, between software components that complicates decoupling them for improved reuse and independent development. Since hardware components

have a finite lifetime they need to be exchanged at some point in time. Therefore, the software needs to be decoupled from the hardware that it is running on. This may lead to compatibility issues, signal interface changes, and novel software allocations that create disruptions in software architectures.

Important Qualities of SDV Architecture

These long-living and complex architectures demand to shift the focus of SDV architectures towards novel qualities. One important quality for the development of SDVs is maintainability, which also relates to evolvability and updateability of all software parts. This should also be supported over "30 years of the lifetime of a vehicle" from its initial design until its retirement. Another important concern is cyber-security which is especially hard because automotive software is complex and should be updated regularly. For instance, to meet the demands of the evolving threat landscape, the vehicle's security controls need to address new and evolving threats or vulnerabilities. Moreover, SDVs availability has become a sophisticated issue. For instance, because of ECU wakeups for some random applications, which come due to complex architectural design, battery drainage has become a common problem. In addition, safety becomes significantly important because of the increased interaction between software and hardware. Safety can go both ways – enhancement or disaster. Furthermore, software performance needs to be optimized, for instance, optimizing the scheduling, allocation, inter-process, dependencies, communication load, and to avoid bugs in the software through more testing, validation, and coverage of corner cases.

Data-Driven Safety Functions

One challenge is that safety requirements engineering needs to become compatible with a software supply chain that is at least partially open-source. Open-source has the advantage that it can be a culture change driver and can improve the code quality. A technical challenge is that redundancy of the solutions need to be assured that involves different types of hardware and software developed by different vendors. Active safety functions become increasingly more sophisticated every year. Yet, there is an opportunity, because the conditions and prerequisites of safety development can be integrated in the holistic data-driven engineering approach, thus safety engineering will become more efficient and faster. Additionally, more data for safety is becoming available in real time. Another opportunity is that the decoupling of software from hardware has the effect that the required safety functions can be designed and developed more efficiently. Furthermore, applying a clear architectural approach (system, software, e/e, hardware) enables a more efficient approach for the design of the respective safety functions.

Opportunities of cloud computing in the SDV

The cloud should be considered as part of the extended vehicle system where any non-real time application can in principle move into the cloud. Data storage and analysis of real-time data for predictive maintenance could be offloaded to the cloud. This leads to a highly accurate digital shadow that could also include the deployed software and hardware in the car. Furthermore, it could offer customer-specific personalization, e.g., by having the same driver profile in all owned cars, rentals, or trucks in a large fleet. The cloud could be leveraged to control and optimize the behavior of the fleet of vehicles or enable V2X. Furthermore, functionalities that are of interest for a broad customer group like traffic-aware routing can be shifted to the cloud. However, since most cars, in the field, are not even completely updatable over the air today, function offloading and other related functionalities are deeply hypothetical. Furthermore, accessing the cloud requires continuous network connectivity that may not always be available, e.g., for vehicles operating in mines, tunnels or areas with poor connectivity.

> The SDV architecture needs to anticipate future requirements, legal mandates, as well as managing dependencies and ensuring compatibility over a long lifespan of vehicles. The emphasis on qualities such as maintainability, cybersecurity, and performance optimization underlines the need for evolving architectural practices and better software governance. Safety engineering can benefit from real-time data integration. Decoupling of software from hardware, and holistic approaches improve efficiency. Lastly, cloud computing offers capabilities like predictive maintenance, fleet optimization, and customer personalization but is hindered by current limitations in vehicle connectivity and updateability.

Process

Changes in the software architecture management

These novel qualities of SDVs require changes in the software architecture management. First of all, "the mainline software branch" should be the overarching guideline and north star for any architecture manage-

ment decisions. Besides, governance of the complete system architecture becomes extremely important to avoid local optimizations affecting the overall system quality. An example scenario that needs to be avoided is teams that make local optimizations and quick solutions without considering the effects on the CPU load, bus bandwidth, dependencies, performance. Lastly, the vehicle system architecture is moving away from over hundred of ECUs towards a zonal architecture which also implies decoupling the software from system and hardware.

Stricter Decoupling of Software and Hardware Lifecycles

The complexity of software-defined vehicles (SDVs) surpasses that of traditional cars by orders of magnitude. While a car consists of 8,000 to 12,000 physical components, its software counterpart consists of millions of interdependent functions. Engineering complexity is expected to decrease with the shift from CAN signals to commodity protocols such as HTTPS and IP. However, this requires rigorous lifecycle management and well-maintained semantic APIs for both software interfaces and hardware modules. However, the challenge grows as more software components and dependencies must be managed in parallel, especially with increasing customization. A critical difficulty lies in balancing subsystem decomposition for independent lifecycles while maintaining overall system integrity to prevent local optimizations from compromising global quality. Decoupling the lifecycle of software and hardware requires an engineering framework that supports their independent evolution while ensuring seamless integration. Frequent hardware changes intensify this issue, making robust version and configuration management essential. A holistic versioning approach could help bridge the gap between hardware and software development.

A major challenge is designing generic hardware adaptable across multiple car lines, enabling software to be reused in current and future vehicles. Although zonal architectures enhance decoupling, evolving sensors—such as high-G acceleration sensors [8]—bring challenges to software reuse: The software must be able to cope with different hardware configurations (the number of axis in the example of the acceleration sensors).

Views on business requirements and the SDV development concept

Strictly managed APIs enabling full updateability allow continuous feature evolution, shifting organizational structures from hardware-centric units to system capabilities. Clear interfaces facilitate software reuse across car lines, increasing test complexity but expanding market potential. Software-driven functionality enables OEMs to accelerate feature development and release vehicles with incomplete feature sets, later refined via updates.

Challenges on legal requirements on SDVs

Legal challenges for SDVs will intensify, particularly regarding IP regulations, software patents, and the complexities introduced by generative AI. Efficient software supply chain management is crucial for frequent updates while ensuring compliance with safety and product regulations. A unified, continuously refreshed software baseline can help manage complexity. The data-driven approach of SDVs will accelerate homologation, necessitating a fully digital and seamless process. Certification remains a major hurdle, especially achieving ASIL compliance for “commodity” operating systems, though examples such as ThreadX[1] show feasibility. Market-specific certification further complicates development, requiring early planning and thorough documentation. Cybersecurity will be critical, as expanding functionality increases the attack surface, demanding continuous threat management.

On streamlining the SDV process

Automating SDV processes improves speed and quality while upskilling related roles such as project and product management. Open-source methodologies: Transparent communication, shared code, and rigorous IP checks—serve as key inspirations. A major challenge is frequent updates to hardware and middleware, requiring constant adaptation at the application layer. Achieving a balance between decoupling and system-wide governance hinges on solid system design principles and tools. For verification of the deployed software configuration “traditional” software verification techniques need to be adapted for automotive settings, which have different requirements. Furthermore, through the continuous updates with new functionalities that may effect old functionalities these require re-verification.

When a new change is made by a programmer, the system under test needs to be compiled, made available, and tested. This process is first applied to the software component, then to the complete hardware component, and ultimately extends to a fleet of vehicles. The challenge is in selecting the variants to test [9] as well as how to smoothly deploy on different target systems [10]. All in all, the main challenge is to have a fully

[1]ThreadX: https://threadx.io

automated end-to-end CI/CD pipeline available. This especially means well-defined and commonly available interfaces instead of manual data transfer. Adhering to evidence-based software engineering, establishing lightweight yet rigorous processes, and avoiding superficial trends are essential for long-term success.

SDVs require strict decoupling of software and hardware, robust versioning, and semantic APIs to manage complexity. Legal and regulatory challenges necessitate seamless compliance and cybersecurity measures. Automation and evidence-based engineering streamline development while ensuring adaptability.

Organization

Moving Software Engineering to Qualified and Integrated In-House Teams

According to Conway's law [11], *"organizations design systems which look like their organization charts"*. For example, if an organization has different teams for front-end and back-end development that have problems in communication, the end product developed by this organization will lack smooth integration between its user interfaces and the back-end system. This emphasizes the importance of the organizational aspect even on systems and products' levels. Hence, in order to shift the industry towards software-centric products, the management needs to develop an understanding of software-defined development and a data-centric approach to it. However, change management is crucial, time-consuming, and highly dependent on the existing culture. A seamless system engineering approach must be aligned with organizational structures to enable collaboration across disciplines and allow efficient (automated) integration. It is important during the transition phase to focus on transparent definition of interfaces, responsibilities, and decision making processes. On the workforce side, there is a lack of software engineers in the market, particularly senior software engineers and architects. Companies transitioning their focus from hardware (both mechanics and electronics) to software have faced challenges in quickly adapting their workforce. Since replacing a large number of employees was not feasible, some internal employees from traditional mechatronic teams underwent quick software development training. However, these efforts rarely produce fully capable software engineers, as the knowledge required (both in depth and breadth) as well as decision-making skills cannot be achieved in short-term training programs. This issue has been improving in recent years, but it remains a challenge, particularly when OEMs move towards more in-house software development rather than relying on suppliers.

Promoting effective cross-functional collaboration between hardware, software, and data teams

The major challenge in cross-functional collaboration is the inability of hardware engineers to understand how software behaves. This requires training on main software principles like abstraction, separation of concerns, management of complexity, and the use of version control systems such as Git. Additionally, hardware engineers need to dimension the hardware based on software needs that might come years later, which in turn requires a close collaboration with other teams, especially software teams. Besides, the teams and individuals also need to discuss on higher levels. For example, moving from low-level specifics like CAN-message Bytes to higher-order software topics aimed at long-term maintainable, capable, and stable architectures.

This requires close collaboration between hardware and software teams to avoid any misalignment in design decisions, keeping both points of view in consideration. Moreover, to break down silos in thinking, there is a need to build an open culture that values abstractions and general solutions with management supporting scalable approaches and not the local fixes that just solve the particular problem

The impact of continuous software updates of SDV on OEM culture and organization

On the one hand, the shift towards SDV comes along with the demand for continuous or frequent software updates with shorter and faster design, implementation, and testing cycles putting an increased time pressure on software teams. On the other hand, the tolerance for failures and errors might increase as correction would be possible with Over the Air (OTA) updates allowing the products to evolve after production. However, the number of read/write operations on existing non-volatile memory solutions could induce wear-out that again might accelerate the rate of device failure.

A significant change is the move towards in-house development opposed to the current common practices where the software is usually developed by suppliers. This allows for faster development and more re-usability of code and development artifacts. Moreover, there is a need to move towards general-purpose hardware especially for the core system, keeping in mind that

specialized hardware would still be necessary for components such as cameras and radars.

How pioneers in the SDV space are redefining organizational structures

Traditionally, OEMs were structured around hardware and mechanical engineering. The pioneers in the SDV are structured around software engineering built on top of their legacy hardware and mechanical engineering. They have the flexibility to structure their organizations in a software-centric way, putting the development teams at the center. An organization of an SDV company adopts a best-of-breed approach, integrating all disciplines (system, software, electrical & electronic, and mechanics) while focusing on measurable deliverables. On the management side for pioneers, the focus is orchestrating the collaboration between the different teams, e.g., software, hardware, and data, to ensure alignment and breaking up silo-thinking.

A main difference between traditional OEMs and the pioneers in the area can be seen in their abilities for continuous updateability. This ability requires shifting towards capable system-level implementation in the product rather than focusing on narrowly-scoped sub-functions.

Strategies to effectively manage change across organizations transforming to SDVs

Transforming towards SDV can be time-consuming and there is a market pressure to make it fast. It is crucial that OEMs are patient to avoid making wrong moves that can have significant negative consequences. The decisions made during this transformation have to be transparent and data-driven based on proof of concepts and study results. The process should involve experts to empower them to drive the transformation, which reduces hierarchy and ensures that the right, interdisciplinary, and collaborative people are onboard at all hierarchical levels within the organization.

Transformation to SDV also requires financial investments. The budget should prioritize development and as well as hiring and training to build a competent workforce and establish a software-centric culture through systematic approaches. This includes increasing and empowering software-capable managers on all levels of the organization and strengthening collaborations with universities and research institutes especially with automotive-related areas.

> The transition to SDVs requires organizational changes that support a cultural shift towards software-centric development. This involves emphasizing cross-functional collaboration, shared understanding, and long-term, scalable solutions. It is very important to break down silos between different teams (hardware, software and data), and adopt a system-level mindset. The process is long and demands patience, transparency, and data-driven decision-making, along with financial investments in development and workforce training.

Future Perspectives

Software-Defined Vehicle is a term highly charged with many hopes, expectations, and fears. First and foremost, OEMs and suppliers will define that SDV means to them to ensure their business goals, processes, architecture, and culture can be aligned with this vision. Based on the experts' feedback, outline above, we predict that the design of software will change dramatically, that SDV-related decisions will be increasingly based on data-driven observations, that organizations will reshape to reduce frictions in systems engineering, and that shifting towards a software culture will become vital to keep up with and surpass competition.

Lean Software First

Recognizing and addressing the increasing importance of software is another key differentiator. Software will be the main driver of value-added SDV functions (either on-board or in the cloud). Creating and maintaining it will require expanding in-house software engineering capabilities through training, restructuring, hiring, strategic partnerships, or adoption of open-source solutions as well as adopting a software competency and culture across all layers of a company. The essential competencies will be (1) reducing and managing variability of the SDV software to ensure its maintainability and extensibility, and (2) regulating changes to it properly to prevent misalignment between software (and hardware) parts. This might entail moving from vast product lines with hundreds of features to monoliths with fewer interacting parts or identifying means to manage large product lines of software for mechatronic systems efficiently.

Leverage Open-Source Software

Leveraging open-source software could significantly accelerate development cycles, improve collaboration, and reduce costs. This will enable SDV engineers to take advantage of network effects, such as collaborative development of jointly used software parts, to improve

interoperability across industry, which can lead to faster innovation [12]. Benefiting from open-source might be essential to prevent "established industry from steamrolled" by competitors shaping their business models, processes, architectures, and organization to focus on software first. Open-source also introduces important challenges. These include managing license compliance, ensuring the security and reliability of third-party code, and aligning open-source components with safety-critical standards. Despite these challenges, open-source initiatives like Eclipse SDV[2] and ThreadX demonstrate how collaborative development can drive standardization and improve interoperability, which ultimately enables a more inclusive SDV ecosystem.

Adopt Software Culture

This transition to software-driven will conflict with a noticeable gap in software culture, mindset, and skill set within the current workforce. To mitigate this, "train[ing] hardware engineers in software principles," such as abstraction, generalization, ISO 25010 qualities of software, as well as generally lifting the discussion to software topics ("We're not talking CAN-message bytes anymore, but long-term maintainable and stable capability architectures") might be vital. However, a SDV still is a vehicle and, opposed to various kinds of software, subject to physical laws, legal constraints, and other influences that need to be considered properly: an SDV is not purely software. Overall, all of the suggested chances and opportunities demands shifts in automotive systems engineering culture–and "culture eats strategy for breakfast" (Peter Drucker).

Make Decisions Data-Driven

Another challenge lies in determining how AI and data-driven decisions will shape and influence the development of SDVs. AI-driven automation, predictive analytics, and adaptive learning models might play a crucial role in improving SDVs and their development processes. This will demand capturing as much data of the SDV development process and their operations to enable learning from and improving both. Consequently, OEMs and suppliers will assign resources to develop and maintain individual, specific AI capabilities (e.g., co-pilots for design-space exploration, platform-specific code generation, improving test coverage, documentation, or explainability). This will ultimately create additional software infrastructure and might shift them to become more software-focused companies in the process. Moreover, this will also demand training all kinds of SDV engineers in the efficient and reliable use of AI to make decisions based on its suggestions.

Engineer Systems Holistically

Another vital challenge will be integrating systems engineering holistically into the SDV development process to address the problems of "silo thinking", in which departments optimize their solution locally without considering negative effects on the solutions of other departments. Only by aligning hardware and software engineering departments more closely—organizationally, culturally, and technologically—a seamless collaboration, enabling a more efficient and cohesive approach to SDV development can be obtained.

SDV Ecosystem

The challenges discussed in this article are highly relevant across the broader SDV ecosystem. Tier 1 suppliers, cloud infrastructure providers, regulatory bodies, and mobility service operators all play critical roles in shaping the SDV landscape. For instance, aspects such as cybersecurity, life-cycle management, and compliance with evolving standards are shared concerns that require coordinated efforts across organizational boundaries. Moreover, the transition toward SDVs demands new forms of collaboration, data sharing, and co-development practices that impact the entire value chain.

[2]Eclipse SDV: https://sdv.eclipse.org/

ACKNOWLEDGMENTS

The work of Jolak and Svenson was supported by the TWIN-LOOP Project (Grant Agreement No. 101192649) which is funded by the European Union.

REFERENCES

1. Slama, D., Nonnenmacher, A., Irawan, T., "The Software-Defined Vehicle", O'Reilly Media, 2023.
2. Goswami, P. "The Software-defined Vehicle: Its Current Trajectory and Execution Challenges." SAE International, 2024.
3. Keßler, G., et al. "The Software Defined Vehicle–Technical and Organizational Challenges and Opportunities." International Stuttgart Symposium. Springer, 2023.
4. Liu, Z., Zhang, W., Zhao, F. "Impact, challenges and prospect of software-defined vehicles." Automotive Innovation 5.2 (2022): 180-194.

5. Van der Linden, F., Bosch, J., Kamsties, E., Känsälä, K., and Obbink, H. “Software product family evaluation”. In Software Product Lines: Third International Conference, SPLC 2004, Boston, MA, USA, 2004.
6. ECE/TRANS/WP.29 United Nations. Addendum 155 – UN Regulation No.156 - Uniform provisions concerning the approval of vehicles with regards to software update and software updates management system. 2021.
7. ECE/TRANS/WP.29 United Nations. Addendum 154 – UN Regulation No.155 - Uniform provisions concerning the approval of vehicles with regard to cyber security and cyber security management system. 2021.
8. Robert Bosch GmbH. “Bosch MEMS sensors now enable faster airbag deployment”, 2018. https://www.bosch-presse.de/pressportal/de/en/bosch-mems-sensors-now-enable-faster-airbag-deployment-174592.html
9. Hettich, H. “Automated Test Process for Highly Configurable Software-Defined Mobility Systems in CI/CD”, SAE International, 2025.
10. RedHat, “Ford’s community-first approach to CI/CD ”, 2023. https://www.redhat.com/en/blog/fords-community-first-approach-cicd
11. Conway, M. E. “How do committees invent.” Datamation 14.4, 1968.
12. Eclipse Foundation, “Driving Efficiency and Sustainability: The Business Value of Open Source Software in the Automotive Industry”, 2024.

Rodi Jolak is a senior researcher at RISE, the Research Institutes of Sweden. Rodi holds a PhD in Computer Science and Engineering from the University of Gothenburg. His research interests include software engineering (SE), secure SE, AI for SE, and SE for AI. https://www.rodijolak.com

Vard Antinyan received his PhD from the University of Gothenburg. He works as a technology and strategy leader of software engineering at Volvo Trucks. His research interests are software complexity, software quality, software metrics, software architecture, and software development processes.

Alexander Åström is a cybersecurity specialist working as a Consultant in the Automotive domain, focusing on both development and research. He currently works as cybersecurity architect within Volvo Group Trucks Technology (GTT).

Darko Durisic is a senior program manager leading the development of the software and electronics architecture and system design for one of the newest vehicle programs at Volvo Cars.

Oliver Kopp is a senior researcher at the Institute for Software Engineering (ISTE) of the University of Stuttgart. He conducts research on universal metamodels for SDVs and advanced workflow languages.

Daniel Krippner is a Tech and Open Source Strategist within the Bosch/ETAS organization, engaged in open source communities like Eclipse SDV. He has 18 years of experience in the automotive industry and a technical background in IT and software development, vehicle connectivity, and protocols.

Stefan Kriebel is Technical Director of FEV.io GmbH. Previously he held at BMW AG various R&D positions in E/E, Driving Dynamics, Driver Assistance and E-Mobility. He is affiliated to the RWTH University in Aachen as lecturer for Software engineering.

Mazen Mohamad is a researcher at RISE, the Research Institutes of Sweden and a lecturer at Chalmers University of Technology. He holds a PhD in software engineering from the University of Gothenburg. His research focuses on security assurance, combined safety and security analysis, AI in software engineering, AI for security, and security for AI.

Jérôme Pfeiffer is a research assistant and PhD candidate at the Institute for Control Engineering of Machine Tools and Manufacturing Units (ISW) of the University of Stuttgart. His research interests are software language engineering, digital twins, and artificial intelligence. https://www.isw.uni-stuttgart.de/en/institute/team/Pfeiffer-00005/.

Chris Seiler studied information technology at the Stuttgart University of Cooperative Education. He has filled various roles in software projects in the telecommunications, airline and automotive industries. Since 2016 he has been with Mercedes-Benz Passenger Cars and he is product owner configuration management within MB.OS.

Pontus Svenson is a senior scientist at RISE, the Research Institutes of Sweden, where his research is focused on systems of systems. He has a PhD in Theoretical Physics from Chalmers University of Technology and previously worked on AI and decision support for defence and security applications.

Andreas Wortmann leads the chair “Model-Driven Engineering for Manufacturing Automation” at the University of Stuttgart. He conducts research on model-

driven engineering, software language engineering, and systems engineering with a focus on Industry 4.0 and digital twins. www.wortmann.ac.

Jan Bosch is professor at Chalmers University of Technology in Sweden and director of the Software Center (www.software-center.se). His research activities include digitalisation, evidence-based development, business ecosystems, artificial intelligence and machine/deep learning, software architecture, software product families and software variability management. www.janbosch.com.