Trends towards Software-defined Vehicles

Dr. Ismet Aktaş, July 2023

Empowering Tomorrow's Automotive Software
www.etas.com

Abstract

The Software-defined Vehicle (SDV) is no longer a vision. The transition is in full swing, making the maturity level one of the strongest differentiating factors for OEMs in a highly competitive market. Each manufacturer must go through three "eras" to reach the desired state. In the first era, hardware and software are still strongly coupled. Era 2.0 marks the transitional state with an exponential growth of use cases and software applications. Era 3.0 is the finale, where vehicles are mainly defined by their software setup, making them a fully programmable vehicle [1]. What may sound like a natural evolution yet implies several challenges. After all, we are not talking about a cellphone, but a potentially dangerous and technologically highly complex moving object with a lifecycle that can span over decades.

While the goal for SDVs is set, the exact path is not [2]: OEMs are at different stages of the journey and have their own "historical" E/E architecture and software stack embedded into hardware as a starting point. To set them up for success, it is crucial to understand the major trends involved in the transition from hardware to software-defined vehicles. This knowledge makes it possible to rethink how vehicles are manufactured – and why traditional approaches are doomed to fail.

To reach the final era, shifts in three major trend areas can be observed: from distributed to centralized, from manual to tool-driven, from closed-source to open-source. All innovative approaches within the automotive sector have one thing in common: they contribute to one or more of these three transitions. ETAS has decades of experience in the mobility sector as well as close relationships with OEMs. Hence, the trends described in this paper are based on first-hand knowledge and information. Knowledge about the steps that need to be taken to reach the final era is crucial for all players within the industry if they want to stay competitive and benefit from these shifts, instead of being left behind.

In this paper, we focus on general software-related trends. Cybersecurity-related challenges and principles are addressed in our paper "Cybersecurity for the Software-defined Vehicle" [3].

Table of contents

Introduction

To move from traditional to software-driven vehicles, manufacturers go through three eras. In the first, the link between software and hardware is still strong. Many vehicles that are on the road today were produced in this first era. The code explosion marks the transition to the second era and is now an essential part of each manufacturer's offer. Software and digital services are highlighted as prominent features besides the hardware components. We are currently in a race where the goal – the third era of the fully developed SDV – is clearly in sight. However, the right way of dealing with many of the challenges still needs to be found. Looking at the numbers, the amount of code – the DNA of all the novel functions – is constantly rising. And we are not talking about a single vehicle unit only; instead, the number of produced vehicle models including its variants will rise, and each has a life expectancy of decades and needs ongoing maintenance.

For OEMs, this means handling a diverse set of components for each vehicle model (e.g., sensors, control units, communication modules, etc.), each with its own specifications and configurations provided by different suppliers. Keeping track of device versions, firmware updates, and securing compatibility with the overall system setup will, therefore, become a major concern. In a nutshell: The amount of code required for bug fixing, maintenance, and updates to keep up with customer needs and wishes is rising exponentially. Today's vehicles have around 100 million lines of code. The size of this number only becomes clear when comparing it to a passenger airplane, which has around 15 million lines of code. In other words: a modern passenger car has six and a half times more lines code than a plane. This is set to increase even more with SDVs, where the number is expected to be around 300 million by 2030 [4]. The industry urgently needs strategies to manage this code complexity.

As a means of comparison, we could refer to the smartphone journey, i.e., the way smartphone manufacturers went through the eras to reach an item which is nearly entirely defined by software. While the expectations of users to have a flexible, app-based, and personalized experience across the entire lifecycle is very similar, the nature of the two objects is very different. On the one hand, a vehicle is still very much a mechanical object with significantly more complex devices, compositions, and dependencies that can cause potential harm even through the smallest software malfunction. On the other hand, it has a much longer life span with potentially different usage scenarios.

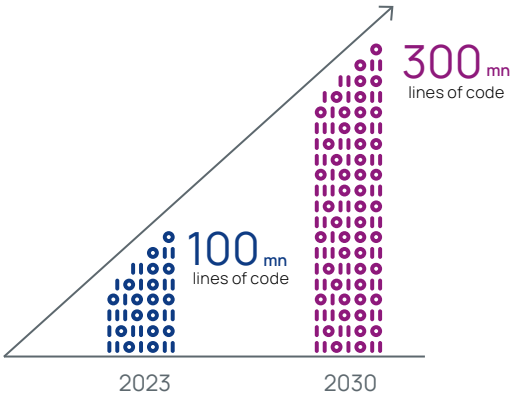
It is important to understand the uniqueness of the journey and the accompanying challenges on the road towards the software-defined vehicle. As an experienced hardware and software provider for the automotive industry, ETAS not only observes trends worldwide, but tries to anticipate their further development. Comparing global and regional requirements reveal several interesting differences. They should be observed during the journey – and used to better understand the big picture **(Fig. 1)**.

Number of connected cars

2023: 192 million vehicles in service
2027: 367 million vehicles in service [5].

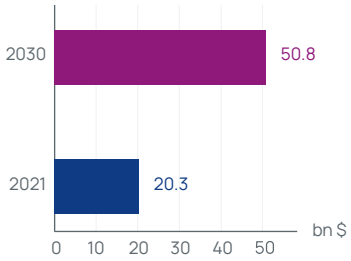
Lines of code

100 million lines of code today to
300 million in 2030 [6].



The global embedded in-vehicle infotainment market size

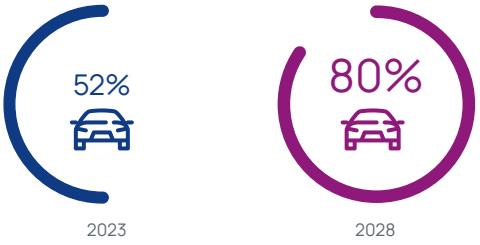
The Global Embedded In-Vehicle Infotainment Market Size grows from **USD 20.30 billion in 2021** to expected **USD 50.80 billion by 2030**



10.72%
Compund annual growth rate (CAGR) during 2021 – 2030 [7].

Connected light vehicles sold globally

Approximately **52%** of all new light vehicles sold globally are connected currently, and this number is expected to grow to **around 80%** by 2028 [8].



Centralized software architecture

The shift to a centralized software architecture trend means higher computing power, but a less number of ECUs.

The number of ECUs in a high-end car, for example, could come down from **over 120 to less than 10** [9].

Many ECUs with major SW functionality with less/moderate computational power

Centralized vehicle computers with high performance and lesser ECU SW functionality

The global software-defined vehicles market size

The global software-defined vehicles market size was accounted at **USD 35.6 billion in 2022** and it is expected to reach around **USD 210.88 billion by 2032** [10].

Fig. 1: Various forecasts related to software-defined vehicles.

Software-defined vehicles – a global transition

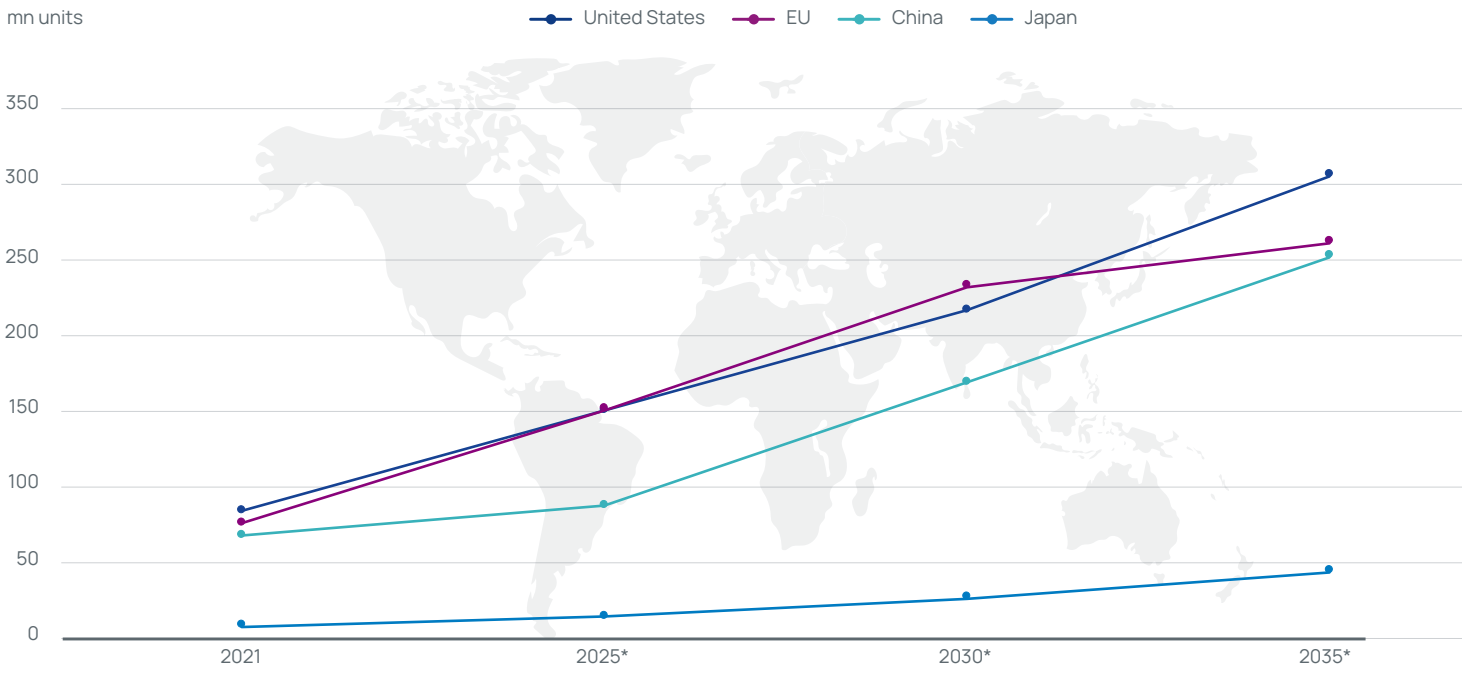


Fig. 2: Size of the global connected car fleet in 2021, with a forecast for 2025, 2030, and 2035 by region (in million units) [11].

statista

The race towards the third era has already begun all over the world, even if it is currently overshadowed by another major change: the parallel transition to electric vehicles. In both cases, traditional differentiating factors (including brand loyalty) are fading into the background; the cards are being reshuffled. Established manufacturers cannot rest on their good reputation or mechanical excellence. If they did so, they would be overtaken by new innovative players.

For example, China is exerting pressure on European, North American, and Japanese manufacturers **(Fig. 2)**. The times when Chinese cars were only marketed locally in the low-price sector and associated with lower quality are definitely over. Now they are on a par with established brands on the world market, especially when it comes to electric vehicles, combining falling prices with more and more innovative features. Western OEMs, on the other hand, are currently raising the prices for such state-of-the-art vehicles from a global perspective (e.g. in the EU and the US), as they spend large sums on increasingly complex developments [12].

Frank Sieren, expert on Chinese economies, defines three major points to explain why traditional manufacturers seem to be falling behind: firstly, the lengthy go-to-market time of around 48 months, to be seen, among others, with German OEMs, where Chinese manufacturers need only 30 months. Secondly, the Chinese market wants SDVs to be cheaper than a traditional car, forcing regional manufacturers to optimize their development and production processes. And finally, Chinese buyers are generally more focused on digitalization than on looks [13].

According to the EU, the vehicles sold should be emission-free by 2035. It can therefore be assumed that vehicles with combustion engines (aka. Internal Combustion Engine Vehicles, ICEV) will probably no longer be sold. When talking about the fully software-defined vehicle of tomorrow, it will likely be electric at some point in time. Moreover, an electric engine alone will no longer be a selling argument – which brings us back to the importance of a fast transition to the SDV.

As we can see, Chinese manufacturers have become major innovation drivers in the Electric Vehicle (EV) sector. But does that mean they are also ahead in terms of SDV development? A forecast [14] predicts that the Asia Pacific SDV market will be the most encouraging market from 2023 to 2032. But still, North America, and Europe are seen as the key SDV regions in this timespan. Nevertheless, there are trends within the Chinese automotive industry worth exploring, as the dynamics and speed of their R&D is baffling.

At the Shanghai Car Show 2023, regional manufacturers presented their vision of the software-defined vehicle with a wide range of in-house SDV solutions like central or cross-domain computing platforms to benefit vehicle development in terms of cost and time. Another focus was on user experience like hardware and software, enabling manufacturers to integrate high-demand technologies (e.g. entertainment systems) into vehicles [15]. Asia Pacific is anticipated to have the highest share in the embedded in-vehicle infotainment market by 2030, fueled by the explosion of car sales [16].

While forecasts predict that European and North American manufacturers will hold their market dominance in the SDV sector over the next few years, it is not predictable what will happen afterwards. The Asia Pacific customers seem to increasingly consider buying vehicles from regional vehicle manufacturers – with a strong focus on an electrified, sophisticated, connected, flexible, and digitalized vehicle. Moreover, Chinese manufacturers increasingly sell their vehicles in other regions, thus slowly increasing their global market share. Independent of the OEM and its originating region, an expansion into more global markets will require manufacturers to create their solution (hardware and software) in an adaptive manner to tackle these different regional needs.

Reaching the final era of SDVs – three major trends

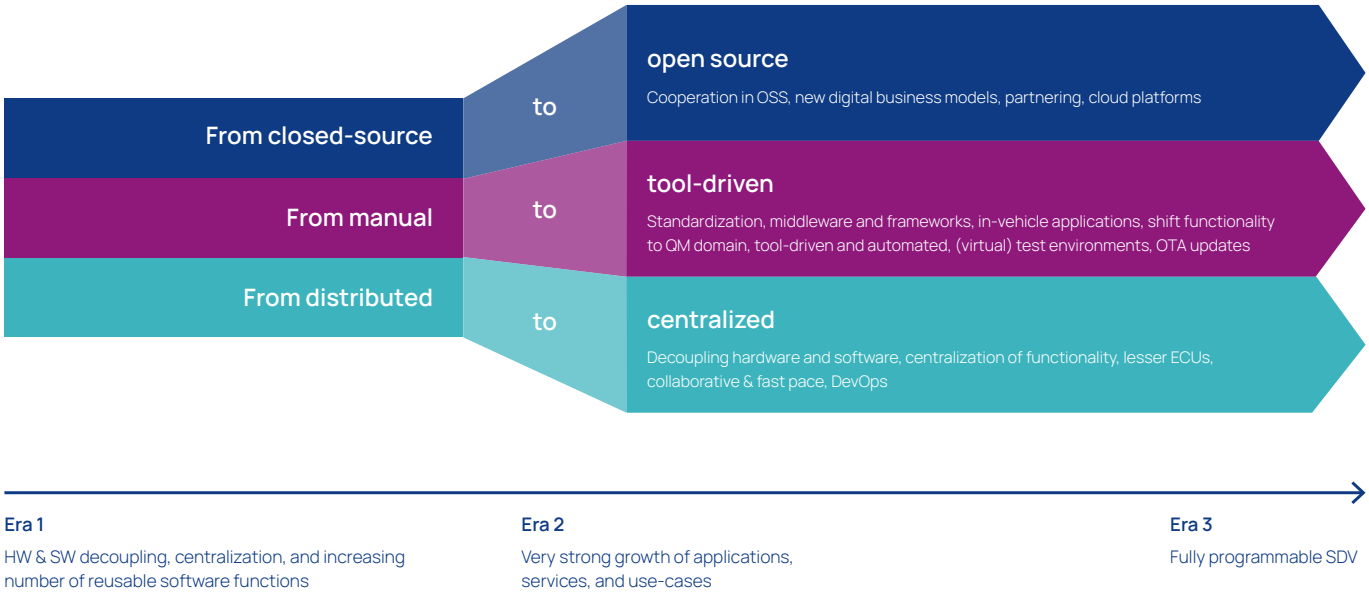


Fig. 3: Three major trends are dominanting the journey towards the final era of a fully programmable software-defined vehicle.

The central question is: which trends are currently driving the evolution to the SDV and, most importantly, which ones should be considered when planning one's own change process? This is not only a major concern for OEMs, but also for suppliers, third-party developers, and further partnering companies. After all, we are all in the same boat, and more collaboration and a shared vision will be the key along the entire lifecycle of a vehicle.

The shift to a full SDV can be clustered into three major trend areas: from distributed to centralized, from manual to tool-driven, from closed-source to open-source **(Fig. 3)**. Before we take a deep dive, let's look at the order. In some way, the trend areas roughly match the three eras towards the desired state of the SDV, with the aspect of centralization being a prerequisite for a more tool-driven and automated vehicle development process. Also, the further we proceed along the journey, the more the importance of collaboration and synergies within the whole industry rises. But the implementation of open-source approaches needs to be addressed early in the process to unlock their full potential in the long-term.

So, the trend areas are strongly connected, they rely on and enable each other. To reach the third era, all three need to be brought to a high level of maturity.

From distributed to centralized: Re-defining vehicle software

In the past decades of vehicle development, software functions were tightly integrated into specific electronic control units (ECUs) and came as a software-hardware bundle from different suppliers – which was a good solution for vehicles with a manageable number of features. With a rising demand of functionalities, the number of ECUs and sub-systems in need for integration skyrocketed: a typical passenger car currently has a set-up of about 150 individual ECUs, making it a highly complex system [17].

Customers expect shorter development times and a fast pace when it comes to new features. But due to the growing complexity, the current reality in the automotive industry looks different: developers spend more than 60% of their time with maintenance, testing, documentation, and formalities, instead of developing new functionalities or fulfilling latest customer demands. And with each new bundle of ECUs in upcoming models, it starts all over again. This project-centered approach makes it hard to transfer implemented function code from one vehicle to another, which limits reusability.

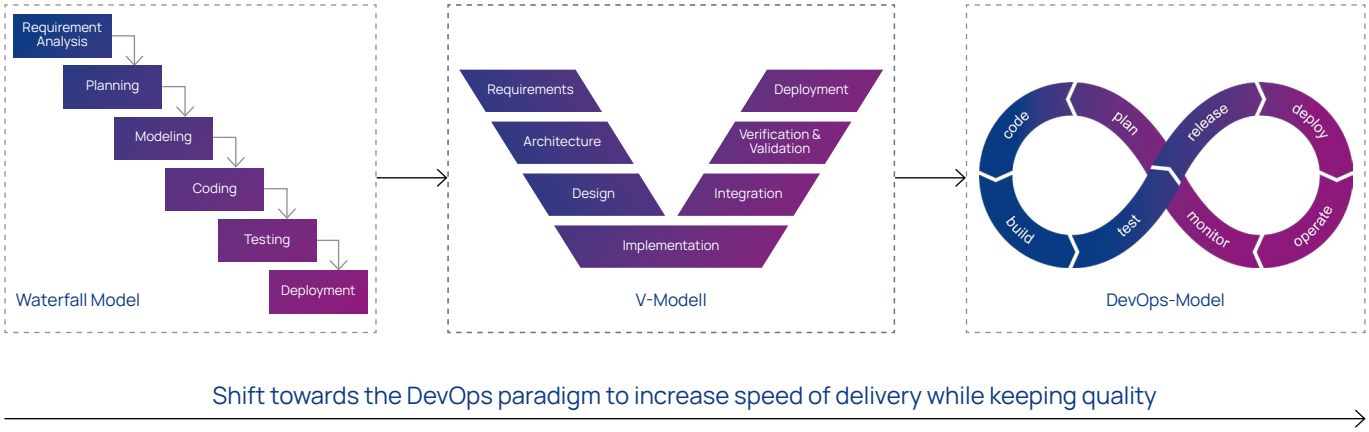
Even if the vehicle is put on the road, the work hasn't finished. It needs frequent software delivery and updates (including safety-related updates) that can barely be met under the circumstances described above. According to McKinsey, software complexity grew by a factor of 4.0 over the past ten years, while development productivity increased by only a factor of 1.0 to 1.5 [18]. To start the journey to the fully software-defined vehicle, this problem needs to be tackled first.

The trend here is already set: **de-coupling of hardware and software**, as well as centralization. Instead of integrating hundreds of individual CPUs with their function-specific code and tooling, developers want a harmonized and uniform approach. **Reducing the number of ECUs and bundling functionalities** and computing power onto few zonal ECUs and vehicle computers is a key trend – and one of the major challenges OEMs all over the world are facing if they want to stay competitive.

Re-shaping the distributed E/E architecture to a **centralized** one is not only a matter of IT knowledge, it also changes the way we work together, especially when cooperating with suppliers and internal/external developers. This new **collaborative and fast-paced** way of working can be mapped only to a limited extent into old, established processes. In parallel, there is a trend towards iterative development processes as available within the **DevOps principle**. Such an approach makes it possible to continuously work on optimizations throughout the entire lifecycle of a vehicle **(Fig. 4).**

Both the cultural aspects and the re-arrangement of the E/E architecture are first steps and prerequisites for the SDV. However, they do not make a difference to customers as such. A user will not notice whether a functionality is centrally controlled or distributed to different ECUs. Hence, this is no sales argument. What they will notice is faster continuous updates of software and inherently the availability of new functionalities. The task now is to make optimal use of these new development processes and principles to generate added value.

The DevOps circle shows a new, interative way of developing and operating software, with a strong focus on permanent optimalization



Shift towards the DevOps paradigm to increase speed of delivery while keeping quality

Fig. 4: Shift towards the DevOps paradigm

From manual to tool-driven: Conveniency and efficiency as future core factors

In 2022, the value of software in a vehicle compared to the entire vehicle value was about 20%. By 2030, it is expected to reach over 40% [19]. Looking at such numbers, it becomes clear that the future differentiating factor will be the grade of vehicle digitalization, with end-customers expecting no less than the state-of-the-art configuration in their vehicle. The decoupling of hardware and software alone does not magically speed up the development process to tackle the rising demands. Manual effort needs to be minimalized to reduce time spent for creating and testing software. Developers must be enabled to focus on the actual functionality rather than time-consuming tasks.

With regards to **standardization**, AUTOSAR Classic is already widely accepted in the industry as a comprehensive solution for safety and hard real-time ECUs. Yet with the demand to move major ECU software capabilities towards centralized vehicle computers, AUTOSAR Adaptive has become very popular within several OEMs because of its service-oriented architecture including functional safety and soft real-time features. Both AUTOSAR standards aim at reducing integrational and management efforts of vehicle functions distributed among various (often safety-related) actuators within the vehicle. This means developers do not need to start from scratch every time; instead, they have standardized formats and interfaces, which are accepted throughout the industry.

Still, it is an ECU-centric approach with a focus on the safety-related domain. Each developer needs to dive deep into the specifics of the E/E architecture, even when he or she just wants to program a simple new functionality, e.g., regarding driver comfort. A new trend can be observed here: shifting functionality to the non-safety domain, i.e., basically the utilization of a tool-supported environment (with an SDK that allows to implement against a vehicle API) for app development, testing, and release management. It allows developers to fully concentrate on the actual feature and user experience without caring about underlying architecture, safety, and security issues.

In this context, different **middleware and frameworks** have been released to run these features as so-called **in-vehicle applications**. While safety-critical applications can be deployed on AUTOSAR Adaptive, non-safety-critical applications have a few of more options. We see a strong popularity increase of infotainment applications based on Android Automotive Operating System (AAOS). Furthermore, the utilization of containers (inspired from cloud-native approaches) running on light-weighted container frameworks for vehicles have gained significant attraction. A special case is available with Advanced Driver Assistance Systems (ADAS) und Autonomous Driving (AD) middleware [20] and applications which are typically deployed on a safety-critical OS such as QNX. However, in cases of development machines and testing infrastructure a deployment in non-safety domains is possible. Moreover, we have recently seen an increasing demand from OEMs to **shift functionality to the non-safety critical domain**, referred to as quality management (QM) domain. This allows them a high degree of flexibility and speed in development. This trend has been strengthened with announcements such as safety-critical Linux [21] that, for example, enables to deploy ADAS/AD middleware [22] in series, or with a safeguarded actuator interface [23] that restricts and handles access to the safety-domain. With these enablers, developers have the flexibility to execute safety-critical features from the QM domain while retaining the required safety level. With this burden off their shoulders, developers do not necessarily have to work for the manufacturer, nor must they program explicitly for a particular vehicle. Standardized interfaces make it possible to roll out the software to different models. This not only keeps development times short, but also optimally exploits scaling effects. OEMs can "reuse" software throughout their vehicle portfolio.

With approaches such as standardization and domain-specific programming, we get closer to our goal of a convenient and efficient development process. Still, manufacturers underestimate the complexity of the process where many steps need to be executed manually – from setting up the development environment to testing, deploying, and updating. Therefore, **tool-driven** ways of dealing with various steps of the DevOps cycle are a rising trend. This basically means to **automate** as much as possible to provide developers with an intuitive interface of one-click solutions for previously complex programming tasks. Pre-defined building blocks, ready-to-use software environments, templates for recurring tasks, step-by-step manuals, and vehicle-specific add-ons for existing programming platforms make the toolbox larger every day, without being limited to just the development process.

One of the key challenges that OEMs face is the fast integration of software towards the end of the software development process, where several issues are typically uncovered leading to delays in the start-of-production of the vehicles. One challenge lies in setting up **test environments** that are accurate enough to mimic real behavior, to provide detailed feedback to understand the root cause, and that can be run

and repeated automatically and flexibly (regression tests). With regards to testing and integration, a trend of cloud-based tools and environments is unfolding. For example, virtual testing from the cloud has gained increasing popularity and allows simulating ECUs with the accuracy required, while minimizing the need for costly real-life tests and significantly accelerating the delivery [24].

Another challenge to increase the speed in software delivery is cloud-based **over-the-air (OTA) updates** to distribute software including new features to vehicles. According to market predictions, the volume of connected vehicles in service will rise significantly from around 192 million to 367 million in 2027 [25]. The key challenge though is not only to update such a high number of vehicles, but also to support different types of updates (e.g., AUTOSAR Adaptive, classic ECUs, infotainment updates, self-updates of connectivity units, vehicle computer updates, container updates, etc.) for different models and their variants with changing dependencies between software components. Campaigns need to be dynamically created when the vehicle connects to distribute the software required in matter of few hours instead of several days. Being able to perform OTA updates alone may soon become a commodity use case, as many OEMs and software suppliers strongly focus on this topic. So, the question will not be whether an OEM offers OTA updates at all, but what their specific differentiating factors will look like – resulting in being faster and more flexible, e.g., through automated dependency handling, comprehensive testing features, and/or other value-adding end customer features.

Thus, the key for success will be to significantly increase the speed needed from software creation, for example for ECUs implemented by various suppliers, to its series availability within the OEMs vehicle.

The automotive industry is competing with many domains for talented developers; young talents are rare. However, software and tool-driven development attracts young professionals to the industry, as this way of working (fast, iterative, function-oriented, and innovative) matches their requirements and allows them to use state-of-the-art multi-purpose programming languages they are familiar with, rather than learning complex and old-fashioned automotive-related ones. They see the shift towards software-defined vehicles as a chance to significantly change the industry. The latest Edge approach is also very much to their taste, as they are familiar with LINUX-based non-safety environments and containerized apps through cloud and edge computing. A much more heterogeneous development team, including young professionals from other industries that are more advanced in digitalization, can bring in new principles and surely speed up the journey towards the SDV.

From closed-source to open-source: personalized, cloud-based, and cooperative

When talking about development tools and centralized platforms, one major question arises: where do they come from? The classical way of obtaining software in the industry is by building it in-house or buying it from a third-party provider (make or buy). Both have their pros and cons **(Fig. 5)**. In-house development (make) allows for a high degree of flexibility and customization of the software itself but is costly in terms of resources. Expertise needs to be built up and maintained within the company in the long term. When the whole process is outsourced (buy), manufacturers have a great flexibility to choose the most suitable system providers available on the market. They are, however, also dependent on a specific product and price strategy, and hence prone to vendor lock-in. In both cases, the result is most likely a proprietary solution with limited interoperability and difficulties in exchanging software components.

Furthermore, in both cases manufacturers remain limited to a small number of external or in-house experts.
Using open-source software (OSS) is an interesting trend that makes it possible to benefit from the synergies of a community and its drive to push innovation. OEMs have started collaborating in open-source working groups that cover various challenges within the industry: from general standardization to platforms and entire development toolchains. The results of their work are free for everybody to use; smaller companies and software developers can participate in the industry much more easily. The general idea is to enable more and more players along the entire lifecycle of a vehicle to offer their services by making open specifications and solutions available – without them ever setting an eye on the actual vehicle they are programming for.

Make

⊕
- Well-suited for the overall solution
- High degree of flexibility and customization
- Optimized for a specific need
- Adaptable for changing business conditions
- Short cycle-times for changes in own control
- Retention of intellectual property
- Lesser dependency on others
- Can be open sourced by own decision and responsibility

⊖
- Resource demanding and costly for entire product-life-time
- Experts required and bound for a long period
- "Do-everything-yourself"-approach (support, documentation, maintenance, etc.)
- High initial, operations, and maintenance costs
- Low scaling effects
- Could lead to proprietary solution with limited interoperability and difficulties in exchanging software components

Buy

⊕
- Flexibility to choose the most suitable providers
- Mature documentation, training, and support
- Challenged by many OEMs and competitors driving feature richness
- Predictable regular costs (license fees)
- Low initial capital expenses

⊖
- Dependency on supplier decisions and roadmap
- Prone to vendor lock-in, no intellectual property
- Longer cycle-times due to generic features to satisfy several OEMs
- Too rich functionality may compromise usability or performance
- Expensive custom change requests
- Interoperability and openness depends on supplier strategy

OSS

⊕
- Well-suited for commodity core technology components
- Innovative community of talents
- Reduced time to market
- High transparency & trust
- No vendor lock-in
- Benefits from a bigger wisdom and vision of the community
- Setting high industry standards, e.g. for security, interoperability, safety
- Low cost enables all players to participate
- Fast and iterative working culture (quick fixes & features)
- High security & reliability due to a more eyes principle
- Allows to efficiently focus on differentiating features & contribute/use commodity features

⊖
- Requires a community which may or may not grow
- In smaller communities, support might not always be reliable
- No roadmap
- No support could cause long-term costs to maintain with all associated obligations
- Contributing team needed if specific features should be realized/prioritized
- Additional invest could be required to be an influential member in the community

Fig. 5: Comparison of pros and cons for make, buy, and OSS.

Open-source communities are already the standard in other areas, e.g., for smartphones and computers. Since vendor lock-in is a big concern in the automotive industry, there was initially some hesitance to collaborate in OSS initiatives, but this has changed. Initiatives such as the SDV working group on Eclipse [26], the COVESA alliance [27], or the SOAFEE project [28] has gained strong attention and contributors across the automotive industry. For example, in Eclipse SDV the number of members increased in 2022 from 11 to 44 in 2023 and similarly the number of OSS projects increased from 0 to 22 within a single year.

Furthermore, we currently see a close collaboration between the more traditionally composed automotive alliance AUTOSAR and COVESA. In October 2022, both alliances announced an intent to align on several software-defined vehicle topics. While COVESA will focus on vehicle data and cloud interaction, AUTOSAR will offer an open interface for the overall system architecture and the in-vehicle network [29].

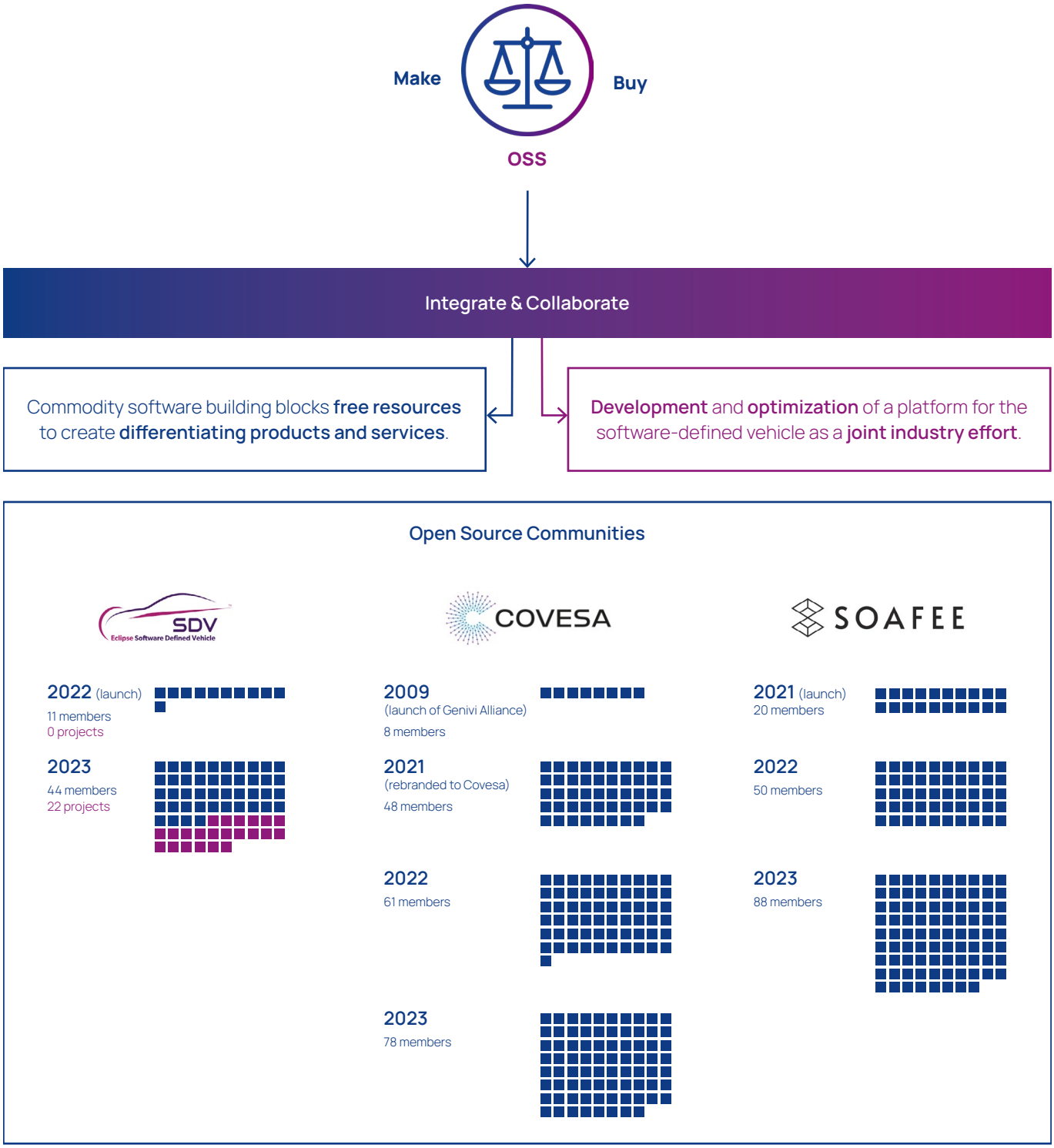


Fig. 6: Growing OSS contributions from multiple type of members (OEM, suppliers, hyperscalers, chip vendors, etc.) in different open source communities.

Generally, creating after-sale revenue becomes an important factor for OEMs. OEMs already take advantage of this up-selling potential with **new digital business models and use-cases**. This is a logical conclusion of an expensive development process and the need for permanent updates to bring in new software into vehicles. New features are particularly important as consumers are reluctant to pay for keeping a vehicle "digitally functional". They simply expect the software to work. Few will appreciate or understand that software needs constant maintenance just like a mechanical part.

OEMs and other service providers therefore need to come up with innovative apps and upgrades or new innovative functionalities for an optimized user experience. Software not only defines the vehicle, but is also responsible for its behavior, e.g., when it comes to sudden obstacles or a dangerous situation. To provide such functionalities, the next challenge must be tackled: the traditional way of software or functionality deployment is no longer suitable. The SDV no longer communicates solely with its manufacturer or service provider through error messages which are read out manually after a problem has occurred. The modern vehicle is connected over-the-air, not only to receive updates, but also to send useful telemetry and diagnostics information during its entire lifecycle up to the cloud. Here, the options for service providers are limitless: vehicle health information over interactive and remote diagnostics, function execution, predictive maintenance with individual maintenance schedules, or road condition analyses for picking the perfect routes. Customers can fully personalize their vehicle with apps and functions, creating continuous revenue streams for OEMs and other suppliers. The longer the lifespan of a vehicle is, the more services can be sold.

Many IT companies have entered new domains. It is not surprising that OEMs have started **to partner** with well-known players in the IT sector from outside the automotive industry. They not only offer long-standing experience in software development, but they are also known brands from the smartphone or computer market that may attract customers. Many such collaborations have recently been announced. To co-build a vehicle operating system, Renault Group has chosen Google as a key partner in their SDV project [30]. BMW has partnered with multiple companies like Microsoft Azure, AWS, and IBM [31]. General Motors started a collaboration with Red Hat's open-source operating system, basically Linux [32]. Mercedes announced a strategic partnership with Google to integrate new in-vehicle data and navigation capabilities [33]. The walls of the once closed-up automotive sector are falling one by one, which is set to have a significant influence on OEM working culture. Apart from becoming more iterative and agile in their development processes, OEMs need more contemporary ways of bringing all players together. Here, **cloud-based platforms** come into play.

Cloud-based platforms as central accelerators

Cloud-based platforms provide a common ground to work on software development by using various tools and typically a user-friendly interface – which makes them important for all three trend areas. Developers can focus on value creation and domain-specific topics instead of generic non-differentiating features. The platforms function as accelerators and are hosts for many of the trends and use-cases mentioned above. They provide long-term cost benefits and master increasing software development costs.

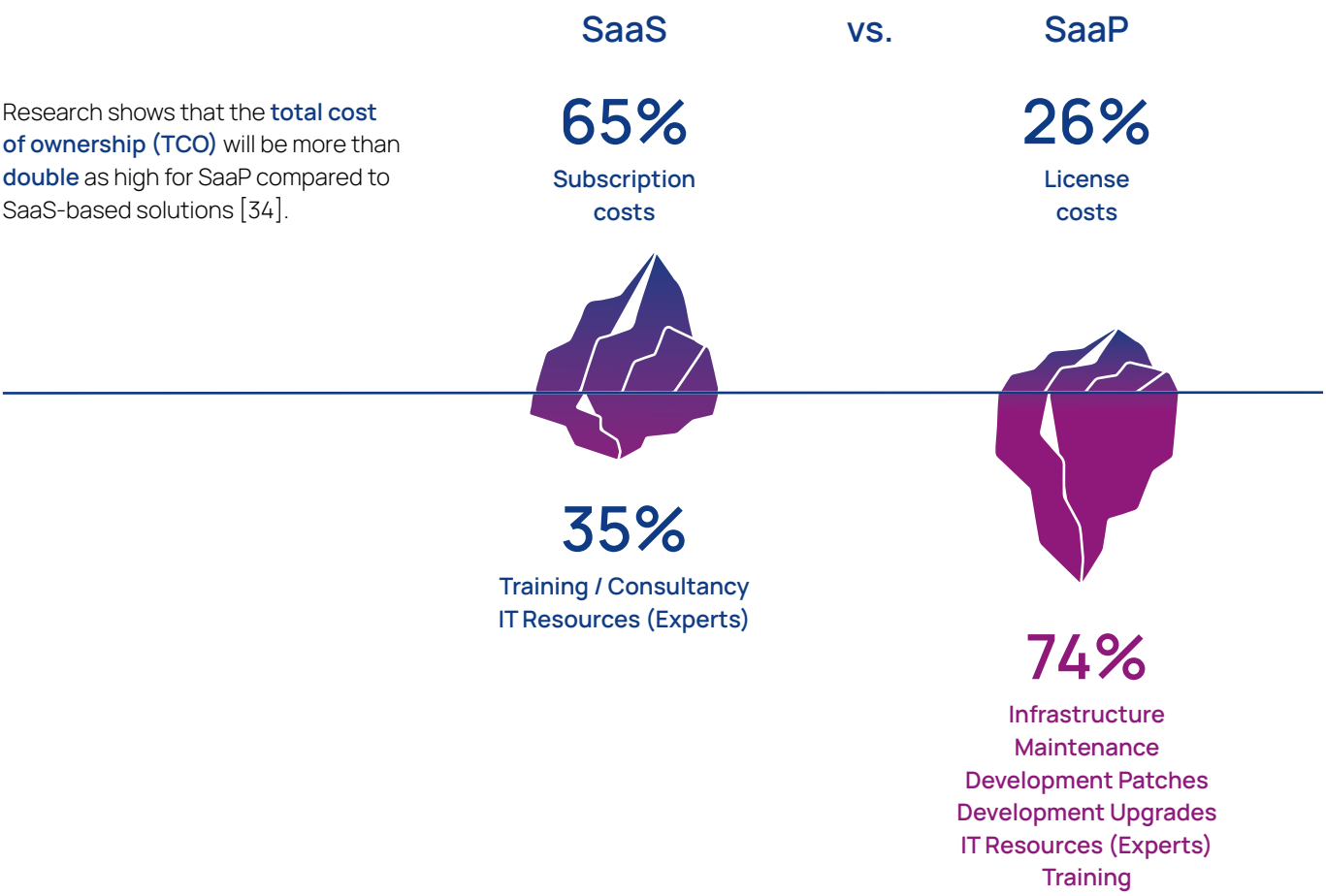
Starting at the beginning, the development and testing of software is simplified, it becomes faster and automated, resulting in shorter time to market. Working pipelines go through a series of automated steps, which enable, e.g., fast rollout of a fleet update without the need to set up a campaign for each model manually.

Cloud technology is also a good basis for collaboration between teams within OEMs and their suppliers. It makes working together much easier and cheaper with its common working ground that is accessible from anywhere at any time.

The software-as-a-service (SaaS) delivery model provides further benefits compared to software-as-a-product (SaaP) for OEMs on track for era three **(Fig. 7).** Typically, SaaS-based services and platforms are designed from the beginning to support a growing number of customers (aka. tenants) and vehicles. It utilizes well-established and proven services and methodologies from Cloud Native Computing Foundation (CNCF) for infrastructure, development, operations, and monitoring – basically, pre-defined building blocks for fast and vast value creation. This enables the limited set of developers/talents within a company to focus on differentiating features, rather than wasting time on basic programming of commodity features. SaaS platforms also relieve OEMs from having to develop their own security functionalities and ongoing improvements to match regulations. SaaS-based platforms are not only challenged security-wise by many OEMs, but also capability-wise where each OEM requests more features. That is, individual requests for features can turn into an overall benefit for all platform users, consequently, leading to an overall reliable and feature-rich system. SaaS platform typically offer several use-cases. Thus, an OEM can first focus on a particular use-case such as on OTA updates, and later extend its use-case portfolio with additional use-cases such as vehicle health, vehicle data, or function calls.

OEMs also benefit in terms of cost. SaaS typically comes along with pay-as-you go pricing models that lead to fair and flexible pricing, utilization, and growth. The development costs are spread over many OEMs who require the same functionalities. Deployment and operations costs are shared amongst tenants. Moreover, the number of tenants and hence high vehicle volumes on a platform allow negotiating lower prices with the infrastructure providers than a typical OEM, if approached alone, could ever achieve. In addition, service level contracts allow downtimes and problems to be solved quickly without having to use one's own resources. In this case, collaboration provides OEMs more room for individualization and the focus on differentiating features. In contrast, in the SaaP-based model the user only get the right to use the service for a relatively low subscription fee and support contracts. All other mentioned points have to be self-manged/-organized.

Research shows that the **total cost of ownership (TCO)** will be more than **double** as high for SaaP compared to SaaS-based solutions [34].



Upfront Costs	Low	High
Subscription/License Costs	Moderate	Low
Time-to-Use	Immediate	Self-managed** & lead time for development
Operations & Maintenance	Included	Self-organized**
P/IaaS Costs	Included	Excluded**
Cost Effective Deployments	Shared and optimized	Exclusive and expensive
Service Level Agreements (SLAs)	Available	Contract-based** with 3rd Party Ops
Scalability	Available	Self-managed**
Support	Available (typically 24x7)	Contract-based**
Privacy & Security Maturity	Mature & challenged by many customers	Self-managed**
Compliance	Available	Self-managed**

** To be managed/organized by OEM

Fig. 7: SaaS vs. SaaP comparison.

The fully software-defined vehicle: Where are we heading?

There is no doubt about it: the vehicle of the future will be software-defined and the third era will eventually become reality. All trends presented in this paper will accelerate OEMs to enter the final stage. But when is a vehicle truly "fully software-defined"? To date, there is no final milestone to hit in terms of apps, functions, lines of code, or any other measurable entity. Also, no matter how far we proceed on this journey, a vehicle will still be a mechanical object for the physical transportation of people or goods. Motor functions or battery capacity can certainly be optimized further with intelligent apps, and the entire interior can be individualized with light, seat settings, temperature, and entertainment. However, there are some physical boundaries that cannot be crossed since they are set by the physical parts of the vehicle.

Within these restrictions, however, we can image a futuristic SDV: we use a fully adaptive SDV for working purposes in the morning, which has all the required settings right through to an individual insurance for this specific timespan. It is connected to a fleet cloud service, providing all information needed to move flawlessly through traffic to the various destinations, and switches to self-driving mode if a meeting is scheduled. In the afternoon, another family member uses the same vehicle in leisure mode, equipped with a sport setting and entertainment package. Later, the teenager of the family has a digital driving lesson during the charging process in the garage, where a simulation is projected onto the screens. All consumer-operated parts of the vehicle including the steering wheel are of course decoupled from the actual mechanics for this purpose. This also makes it possible to switch to a mode for people with mobility issues, assigning random buttons to brake and gas.

Vehicle manufacturers have minimal development times, as the whole vehicle functions with plug & play parts connected to the cloud via a central computing platform. Differentiation takes place within the app store, where consumers can pick and download every functionality or service they need, designed by global development teams that might not ever see each other in real life, working entirely over online platforms to bring together the best experts on a topic.

Even at this point, new visions will emerge as to how a vehicle can be made even more software defined. The "race" to the third era is therefore not about who reaches the non-existing goal first. It is rather about remaining fast, flexible, and extensible to be competitive in rapidly fulfilling ever-increasing customer and other external demands. And not only the ones rising in a specific region – thanks to the new flexibility, it becomes easier to sell vehicles worldwide. Adapting them to local customs - be it legal requirements, user preferences, specific characteristics regarding weather, temperature, road conditions, and usage scenarios - will then be largely a matter of software updates. A vehicle model can become "universal", and function in any circumstances when provided with the optimal set of apps and functions.

Nevertheless, we should not look at the SDV as just a very sophisticated commercial product. Its flexibility allows manufacturers to react extremely fast to sudden or unexpected events. This can be new regulations for climate protection, social trends such as a rise in vehicle sharing movements, but also disasters like global pandemics, or resource shortage due to political conflicts. The possibilities to adapt vehicles become endless and can help to overcome today´s and future challenges.

Summary

There are many challenges ahead towards software-defined vehicles. Even if the focus and strategies may vary for each OEM and supplier, we observe patterns and trends to address these challenges which we emphasized in this paper.

On this way, we highlighted very different speeds of development at the regions and their expansions to others, which may change the landscape of OEMs and suppliers. In this context, we presented the varying focus, requirements, speed, growth, and numbers in different regions. We addressed the central question "which trends are currently driving the evolution of SDV and, most importantly, which ones should be considered when planning one's own change process?". For this, we clustered the different trends **into three trend areas** that are addressed by OEMs and their suppliers:

From distributed to centralized: re-defining vehicle software

We observe a trend towards decoupling hardware from software and the centralization of functionality in vehicle computers. The functionality shift and centralization will lead to lesser ECUs. Centralization will also change how developers from different parties need to collaborate to bring functionality to a common vehicle computer platform. In parallel, the collaborative way will be supported with enhanced development processes and paradigms as available through DevOps.

From manual to tool-driven: conveniency and efficiency as future core factors

The key to success for an OEM is to significantly increase its development and deployment speed while managing the ever-growing software complexity. In this regard, we see strong standardization and popularity increases of middleware and frameworks with the demand to move major ECU software capabilities towards centralized vehicle computers. Associated with frameworks, we see a significant attention on in-vehicle applications. Furthermore, an increasing demand to shift functionality to the non-safety critical QM domain has gained attraction. Enhanced tooling with DevOps is a rising trend to increase speed for essential parts of the development including automation, testing, deployment, and delivery. This is supported with virtual testing environments to allow developers to further speed up their development while facilitating focus. Another prominent trend is cloud-based over-the-air (OTA) updates to distribute software including new features into all domains in the vehicles.

From closed-source to open-source: personalized, cloud-based, and cooperative

The collaboration and synergies within the whole industry rises. We see a strong growth of members and projects in different OSS communities with a mobility focus. A logical conclusion of the ability to bring in new software features such as with in-vehicle application via OTA updates, will lead to new digital business models for the OEM and suppliers. Consequently, we see many new partnerships along the entire software value chain. Especially, the rise of cloud computing platforms further accelerates many of the above-mentioned trends, use-cases, as well as new cooperation and business models.

At ETAS, we are in daily contact with various OEMs and suppliers from all over the world, be it directly or via different standardization bodies, regulations, and OSS initiatives, or consultancy on world-wide governmental levels. As a result, we can convert these impressions early into products that help suppliers, OEMs, and end customers.

Author

Dr. Ismet Aktaş

Portfolio Management & Onboarding VCS
ismet.aktas2@etas.com

Since January 2023 Dr. Ismet Aktaş is part of the portfolio management and onboarding team in the Vehicle Cloud Services solution field at ETAS. In the past, he was a work stream lead from Bosch side (on OPS - cloud-based backend) in the project “software-defined car” jointly realized with Microsoft. Moreover, he was the technical lead at Bosch for various topics ranging from technical PreSales in different domains (e.g. automotive, industry, retail) over technical interfaces for different Bosch divisions to product/project management. He received his Ph.D. and Diploma degrees from RWTH Aachen University where he had a focus on software architectures and cross-layer optimizations for communication and distributed systems.

Sources

[1] Zhao, Fuquan & Song, Haokun & Liu, Zongwei. (2022). Identification and Analysis of Key Technical Elements and Prospects for Software-Defined Vehicles. 10.4271/2022-01-7002.

[2] LIU, Zongwei; ZHANG, Wang; ZHAO, Fuquan. Impact, Challenges and Prospect of Software-Defined Vehicles. Automotive Innovation, 2022, 5. Jg., Nr. 2, S. 180-194.

[3] www.etas.com/de/downloadcenter/42853.php

[4] www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-race-for-cybersecurity-protecting-the-connected-car-in-the-era-of-new-regulation

[5] www.juniperresearch.com/researchstore/operators-providers/connected-vehicles-research-report#:~:text=The%20number%20of%20connected%20vehicles,of%20in-vehicle%20infotainment%20systems

[6] www.sphericalinsights.com/reports/embedded-in-vehicle-infotainment-market

[7] auto.economictimes.indiatimes.com/news/auto-technology/will-2023-be-a-turning-point-for-software-defined-vehicle-trend/97996782?redirect=1

[8] www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-race-for-cybersecurity-protecting-the-connected-car-in-the-era-of-new-regulation

[9] auto.economictimes.indiatimes.com/news/auto-technology/will-2023-be-a-turning-point-for-software-defined-vehicle-trend/97996782?redirect=1

[10] www.precedenceresearch.com/software-defined-vehicles-market

[11] Statista

[12] apnews.com/article/china-electric-car-ev-technology-byd-c7fda57fb0f761c637a71f9f9e7d8b67

[13] www.chinahirn.de/2023/05/01/wirtschaft-i-im-rueck-spiegel-shanghai-motor-show/

[14] www.precedenceresearch.com/software-defined-vehicles-market

[15] www.sbdautomotive.com/post/what-did-you-miss-at-auto-shanghai-2023

[16] www.sphericalinsights.com/reports/embedded-in-vehicle-infotainment-market

[17] semiengineering.com/shifting-toward-software-defined-vehicles/

[18] O. Burkacky, D. Hepp, J. Deichmann, S. Frank, A. Rocha, "When code is king: Mastering automotive software excellence". McKinsey & Company Group, February 2021

[19] www.linkedin.com/pulse/cars-future-more-software-than-hardware-luca-de-meo/

[20] www.etas.com/en/company/press-releases-etas-a-full-service-solutions-provider-for-the-software-defined-vehicle.php

[21] www.redhat.com/en/about/press-releases/red-hat-help-transform-automotive-industry-edge-etas

[22] www.etas.com/en/company/press-releases-etas-a-full-service-solutions-provider-for-the-software-defined-vehicle.php

[23] www.etas.com/en/applications/safeguarded-actuator-interface.php

[24] www.etas.com/en/products/isolar.php

[25] www.juniperresearch.com/researchstore/operators-providers/connected-vehicles-research-report#:~:text=The%20number%20of%20connected%20vehicles,of%20in%2Dvehicle%20infotainment%20systems

[26] sdv.eclipse.org/

[27] www.covesa.global/

[28] www.soafee.io/

[29] www.covesa.global/sites/default/files/COVESA-AUTOSAR-Alignment-202301.pdf

[30] www.renaultgroup.com/en/news-on-air/news/all-about-software-defined-vehicle/

[31] www.maxval.com/blog/from-connected-to-software-defined-vehicles-8-companies-innovating-in-automotive-cloud/

[32] www.marketsandmarkets.com/Market-Reports/software-defined-vehicles-market-187205966.html

[33] group.mercedes-benz.com/company/news/mercedes-benz-and-google.html

[34] www.thesoftwarereport.com/saas/

ETAS GmbH
Borsigstraße 24, 70469 Stuttgart, Germany
T +49 711 3423-0, info@etas.com