DIGITAL INDUSTRIES SOFTWARE

Bringing automotive software development in-house: a shifting landscape for OEMs and suppliers

siemens.com/autosar

SIEMENS

Contents

Executive summary

For automakers, the race is on to develop and bring to market competitive feature sets for highly connected electrical and electronic (E/E) architectures, to meet consumer demand for exceptional driving experiences and to create market differentiation. New vehicles are characterized by unprecedented complexity requiring electronic control units (ECUs) with more processing power than ever.

To that end, OEMs are reversing the historical trends for outsourcing ECU development to Tier 1 suppliers and bringing development in-house, particularly for development associated with proprietary intellectual property (IP) and competitive feature sets. However, this shift comes with a new set of challenges. Historically, OEMs employed teams of specialized engineers using highly specific tools, often working in isolation. Each department had its unique, optimized workflow, which made effective collaboration difficult. Even if OEMs bring embedded development in-house, the miscommunication that can result from working in silos can lead to missed or misinterpreted requirements. As vehicles have become more complex, OEMs will lose time and money and sacrifice quality if they continue with their traditional operating methods.

The good news is that modern design methods and solutions support and enable new ways of working, helping to eliminate data silos, reduce the risk of errors and streamline embedded software development. In this whitepaper, we will explore:

1. How work split between OEM and Tier 1 suppliers is changing for embedded software development.
2. Which systems are leading this change, and how development processes and the skills required will be impacted.
3. How generative tooling supports development within new teams and reduces the reliance on specialists.

Trends in automotive systems design necessitate a new approach

Multiple trends are impacting the automotive industry today. Not only is the amount of electrification and electronic complexity within vehicles increasing, automakers must also grapple with increasing software complexity and stringent governmental regulations. Complex software-intensive features such as driver assistance and autonomous drive, high-quality media and infotainment, along with the demand for an advanced user interface and experience, put tremendous pressure on automakers to innovate rapidly.

Adding to the complexity, multiple concurrent technical transformations are under way with respect to the software-defined vehicle. New terminology is appearing to signify these changes, both to customers and the industry at large, to show how OEMs are becoming increasingly software focused. Examples include "Automotive cloud," "CAR.OS," and "Subscriptions."

> "80% of product innovation and differentiation is now electrical, electronics and software."
>
> Siegmar Haasis
> Daimler R&D CIO

Optional content, delivered as "Apps," is available through over-the-air updates as vehicles connect to an automotive cloud. For example, the option to enjoy heated seats may become a subscription-based service. This means two owners of the same car can use the vehicle differently without mechanical modification. This model provides both opportunity and obligation to provide better cybersecurity, enhanced software features and the ability to respond to and accommodate consumer choice.

As vehicles become more complex, the traditional way in which OEMs have operated is unsustainable. Rather than employing specialist engineers that work within specialist teams using specialty tools, OEMs must find ways to work collaboratively and eliminate data silos that can lead to miscommunication and errors. Often, technical information is still communicated using manual processes, and the exchange of data between teams is inefficient or, in some cases, nonexistent, resulting in misinterpreted requirements and cumbersome change management processes. All of this impacts product quality, drives up costs and affects revenue.

To achieve the efficiency, speed, accuracy and quality necessary to develop these highly sophisticated systems, data silos must be eliminated – both between specialist teams as well as functional development teams such as the departments responsible for the powertrain, chassis and vehicle body.

> "The Automotive industry is wasting millions in investment due to operating in silos."
>
> Elevenci Automotive Consulting
> March 2020

Meeting the challenges of modern automotive E/E systems development

The traditional approach to adding a new feature is to add a new ECU that enables that feature, which requires additional harnessing, capability, connections and software, further driving up vehicle weight, complexity and costs.

What's more, complex specialist toolsets often lack automation and connectivity, impacting productivity. Making changes is easiest if a problem is caught early. The cost and complexity of refining and improving the product increases through the development process. A heavy dependence on physical testing further increases costs and can delay time-to-market. The software development process usually needs to support the full vehicle development and validation plan with increasing functionality at specified phases, however using only those phases to validate the software can result in costly late changes. Software development usually follows an agile development process which includes continuous integration and testing.

Traditionally, the only way to update vehicles once sold was through Recall or Technical Service Bulletins at the dealers, at high cost to the OEM and inconvenience to the customer. Over-The-Air (OTA) updates enable software fixes directly to customers, reducing this cost, but places even more emphasis on a robust validation phase, as the vehicle is now updated away from the workshop and its trained technicians. OTA updates also require an onboard software update process, driving memory and processing resource considerations, and requiring robust Cybersecurity, such as R155 and R156 frameworks (UNECE WP29 legislation). The increasing feature content of modern vehicles means that this is also aligned with expanding functional safety considerations, and more powerful and sophisticated computing power.

Thankfully technologies such as AUTOSAR Classic and AUTOSAR Adaptive are available to support transitional automotive functionality and high-functional safety use cases, as well as service-oriented, high-compute functionality. Powerful multicore microcontrollers and more sophisticated microprocesses with cores optimized for different types of computing tasks are also available. These innovations are enabling the evolution of electronic architectures in vehicles.

AUTOSAR Classic vs. AUTOSAR Adaptive

The continuous development of AUTOSAR has made it the preferred design methodology for system-level application design. AUTOSAR provides the ability to:

1. Define the overarching functions and extract scheduling of a task or a runnable in the core.
2. Transport diagnostics data and at the ECU level.
3. Interoperability then both on the bus level between ECU, as well as on the application level on top of the RT.

There are three main parts of AUTOSAR: AUTOSAR Classic Platform, AUTOSAR Adaptive Platform, and AUTOSAR Foundation. While they may seem to be competing for attention, in reality, they complement each other.

1. **AUTOSAR Classic** is designed for the Classic control ECU, which controls legacy features for the vehicle. AUTOSAR Classic has a statically defined function that is updateable through software updates.
2. **AUTOSAR Adaptive** is used for computation and the fusion of data, and it enables the legacy POSIX and Linux ECUs such as infotainment clusters and telematics to be described at a function system level, then derived down to the implementation and applications.
3. **AUTOSAR Foundation** is underneath both of these platforms and contains the foundational requirements. These requirements capture the interoperability between the two platforms.

What Is AUTOSAR?
AUTOSAR is based on the Open Systems and Interfaces for Distributed Electronics in Motor Vehicles (OSEK), a German standard developed in the late 1990s. The AUTOSAR partnership was founded in 2003 as an international standard to enable the corporation on standard software for automotive vehicles. Today, it is the de facto standard for ECU platform software.

Apart from the specification for embedded software implementation, AUTOSAR provides a base format on data exchange: the digital thread. It also provides for multiple handoffs between different tools throughout development, so that each tool can enrich the data model and provide more granular information.

The evolution of automotive software architectures

As recently as 10-15 years ago, E/E architectures commonly consisted of two or three main CAN buses, with some function-specific LIN bus extensions. A gateway would connect the CAN buses and often, other ECUs would connect to pairs of CAN buses where data needs were high or highly time critical. Initially, adding more features to vehicles would require additional ECUs for hosting each new function, and as the number of ECUs increased, so did the number of CAN buses. Each CAN bus, or network, typically expands with ECUs of related functions, until bus load becomes impractical when it can be split, for example, Powertrain and Chassis ECUs, with potentially an ABS/Braking ECU connected to both, as well as a gateway. OEMs must always consider cost, and this is one of the factors constantly driving consolidation of functions into fewer ECUs. For example, cruise control was originally an optional ECU when the feature was new and novel and later became a software function in an engine controller or a braking module as option take-up became common.

The architecture has evolved to having several functionally oriented networks, some of which may have been "upgraded" from CAN to CAN FD or FlexRay to support higher data rates, and with functional consolidation driving the use of ECUs

with more powerful compute. Initially, such consolidation could be managed with Tier 1 suppliers developing equivalent functions, but this has become costly to OEMs, and is an inflexible approach. Increasingly, OEMs are looking to focus consolidation on a single Domain Controller ECU per functional domain, often taking more involvement in the engineering of these ECUs. The E/E architecture is organized with these ECUs at the head of each domain; however, now some OEMs break the network connection to the gateway and instead connect the ECUs to a backbone network, which is often now switched at the gateway – thanks to the adoption of high baud rate networks, such as Ethernet, which reduced latency between domain controllers.

In service-oriented architectures, the goal is to be able to move functions around easily, leaving headroom in the ECUs to accommodate new functionality, as needed. This should be accommodated in the center of the vehicle so that future updates don't have to be distributed across all ECUs within a vehicle. Typically, ECUs have optimized amounts of processing power and memory, and changing that model to accommodate additional functionality in the future is challenging. Centralizing the architecture and leaving headroom in a few ECUs enables the majority of the ECUs to run in an optimized manner (see figure 1).

As depicted in figure 1, the evolution towards centralized and zonal architectures is increasingly common today. Concentrating the maximum functions into the central and sometimes zonal ECUs enables the outer ECUs to remain relatively simple and lower-cost. Often, these will leverage the AUTOSAR Classic Platform and be quite similar to the ECUs used in vehicles for many years now. Zonal ECUs can be used as I/O concentrators, mostly serving as gateways, which could be an AUTOSAR Classic or AUTOSAR Adaptive function. Multiple OEMs are also hosting some functions, often driving a mix of AUTOSAR platform types across multiple cores. Central compute commonly includes some high-power compute and potentially sophisticated SoC type processing, enabling a combination of AUTOSAR platform types as well as other platforms, such as Linux, to optimize the hosting of different types of processing.

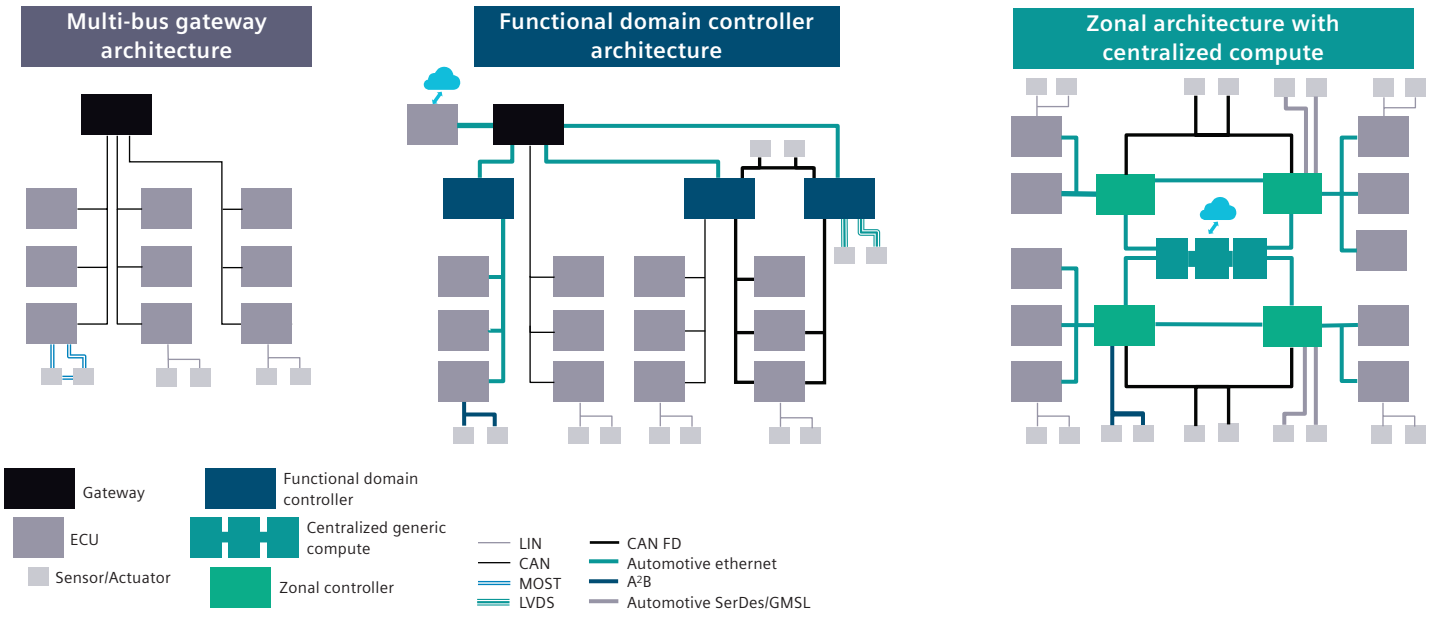
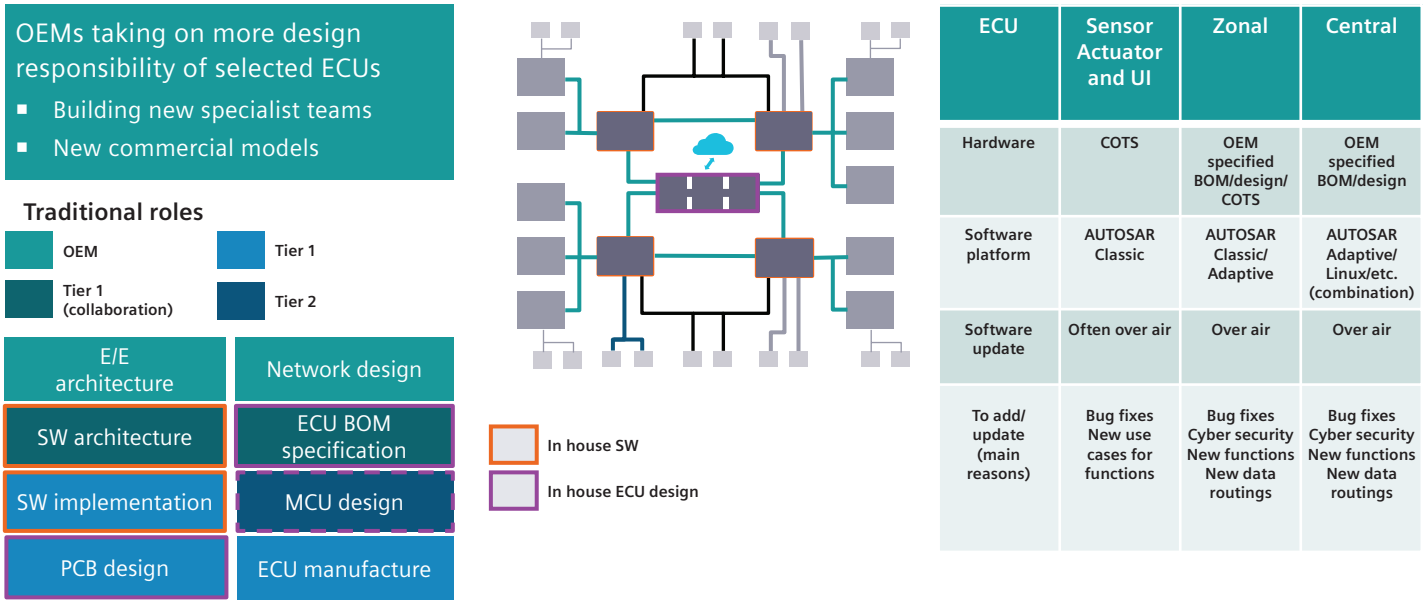


Figure 1. This diagram illustrates the evolution of E/E architecture types.

Additionally, for several years, OEMs have been reevaluating their make-or-buy decisions – vacillating from the idea of spinning out the electronics businesses to buying and working with Tier 1 suppliers (see figure 2). Competitive tension led to OEMs focusing on electrical architecture, network design, and specific ECU components. In recent years, they've become involved in developing the software architectures and writing some of the applications themselves, as well as integration. In some cases they've moved from just specifying certain components of an ECU to specifying the full bill of materials, and even designing the PCB layout. A few also design the microcontrollers, as is the case for Tesla's self-driving ECU. Still, many ECUs are conventionally sourced against a set of requirements.

OEMs have started to become more strategic with respect to ECU development, and bringing brand-defining software functions in-house. This provides more control and enables them to work in a strategic way with long-term partners. In-house development has implications for the CAR.OS and its ability to receive over-the-air updates. Over the life of the vehicle, updates will be necessary to fix bugs, add new functions, and improve cybersecurity. Whereas many of the I/O functions, such as switches and actuators, may require new functionality to be enabled at a higher level, base actuators and sensors won't change much, so traditionally sourced ECUs will continue to perform well over time.



ECU	Sensor Actuator and UI	Zonal	Central
Hardware	COTS	OEM specified BOM/design/ COTS	OEM specified BOM/design
Software platform	AUTOSAR Classic	AUTOSAR Classic/ Adaptive	AUTOSAR Adaptive/ Linux/etc. (combination)
Software update	Often over air	Over air	Over air
To add/ update (main reasons)	Bug fixes New use cases for functions	Bug fixes Cyber security New functions New data routings	Bug fixes Cyber security New functions New data routings

Figure 2. ECU development collaboration requires centralized and zonal architectures.

Table 1 shows how collaboration on ECU design, manufacturing, and integration has changed over the years. To remain competitive, OEMs are realizing that it's faster and more efficient to design high compute ECUs in-house.

Trends	ECU manufacture	ECU Hardware design	Platform software e.g. AUTOSAR	Application software	Software integration	Notes
15+ years ago	OEM Tier 1	OEM Tier 1	OEM Tier 1	OEM Tier 1	OEM Tier 1	Few ECUS, largely independent functions
5 to 15 years ago	Tier 1	Tier 1	Tier 1 Tier 2	OEM: library functions Tier 1	Tier 1	Reducing OEM vertical integration, Tier 1 business units spun out
Today	Tier 1	OEM Tier 1	Tier 1 Tier 2	OEM: software/ controls models	OEM: integrating software for specialist ECU's Tier 1	OEMs insourcing software (IP) development for brand experience related functions
Longer term trends	Tier 1	OEM Tier 1	Tier 1 Tier 2	OEM: brand defining functions Tier 1: commodity	OEM Tier 1	OEMs insourcing design of high compute ECUs, mostly

Table 1: ECU Development collaboration requires varying levels of vertical integration.

Following are definitions of new terminology used to describe the shift toward in-house embedded software development:

1. **Apps:** Features on vehicles are delivered as applications, enabled by vehicle functions. Important to this trend is the ability to add apps over the life of the vehicle. Examples include BMW's Digital Key and High beam assist or Tesla's Dog Mode.
2. **Subscriptions/Features-as-a-Service:** Similar to apps, features are delivered as a service via paid subscriptions.
3. **OTA updates:** The ability to perform software updates without visiting the physical service center. These may include updates to improve cybersecurity, functional updates, additional features, or media.
4. **Automotive cloud:** The Automotive Cloud is a cloud for series production cars in service and is used for software updates and to perform compute for some onboard functions, when connectivity allows. An example is Volkswagen's VW.AC.
5. **OEM.OS:** The OEM software platform base is used by an OEM group, and consists of a combination of OS and middleware, organized strategically according to architecture and functions. Examples include Volkswagen's VW.OS and Volvo's VolvoCars.OS.
6. **Software factories:** OEMs are forming specific software development sites, often separate from traditional product development and sometimes set up as separate businesses. This enables software-specific processes and methods, new supplier collaboration models, and the ability to recruit software developers. Examples include BMW's Car IT and Volkswagen's CARIAD.

The future of software-defined vehicles

Several considerations contribute to the successful development of automotive software:

1. **The digital thread:** A digital thread is a single set of data that is accurate and consistent throughout the design lifecycle, giving all stakeholders insight into decisions made across domains. Using the digital thread, issues can be caught and addressed early in the design process, accelerating design while reducing rework and associated costs.
2. **Collaboration with IP protection:** Several teams will contribute their IP to the software design, so we must ensure IP protection for the organization's IP as well as Supplier and Partner IP.
3. **Developer centric development environment:** The software developer environment must be developer-centric and attractive, enabling developers to utilize their skill sets effectively.
4. **Structured software architecture:** A structured software architecture is required to control resource usage over the system: MCU, CPU, load balancing between ECUs and multi-cores, and so on. This requires a higher level of control over the application.
5. **Configuration:** For configuration, future personal transportation needs will require a different set of configuration abilities. We will see service-oriented architectures, because functions will be traveling over ECUs and possibly to the cloud over the versions of the vehicle and the lifetime of the software-defined vehicle.

Model-based systems development enables a holistic assessment of the E/E architecture developed in the context of the target platform. It absorbs the system models and all of the requirements impacting development, while enabling developers to optimize design for downstream implementation of Electrical Distribution System (EDS, wiring harness), hardware, and software. Using a digital thread, developers can dynamically assess the design against key criteria and in the context of the mechanical targets. This leads to the ability to refine architectural proposals for implementation teams who are developing the final product, while keeping costs in check.

> "Software will account for 30% of total vehicle content by 2030, up from about 10% today."
>
> McKinsey

Part of the Siemens Xcelerator portfolio, Siemens Capital ™ VSTAR is Siemens' implementation of the AUTOSAR standard, and a complete offering with all the tools and features required for rapid embedded software development using a model-based systems development approach. Capital VSTAR reduces hardware dependency and enables automotive software developers to work within an AUTOSAR-aware environment as they integrate, test, and analyze their next-generation automotive software. Capital VSTAR adheres closely to the AUTOSAR methodology and includes native support for the AUTOSAR Layered Software Architecture.

Figure 3 illustrates the Capital VSTAR design and development flow.

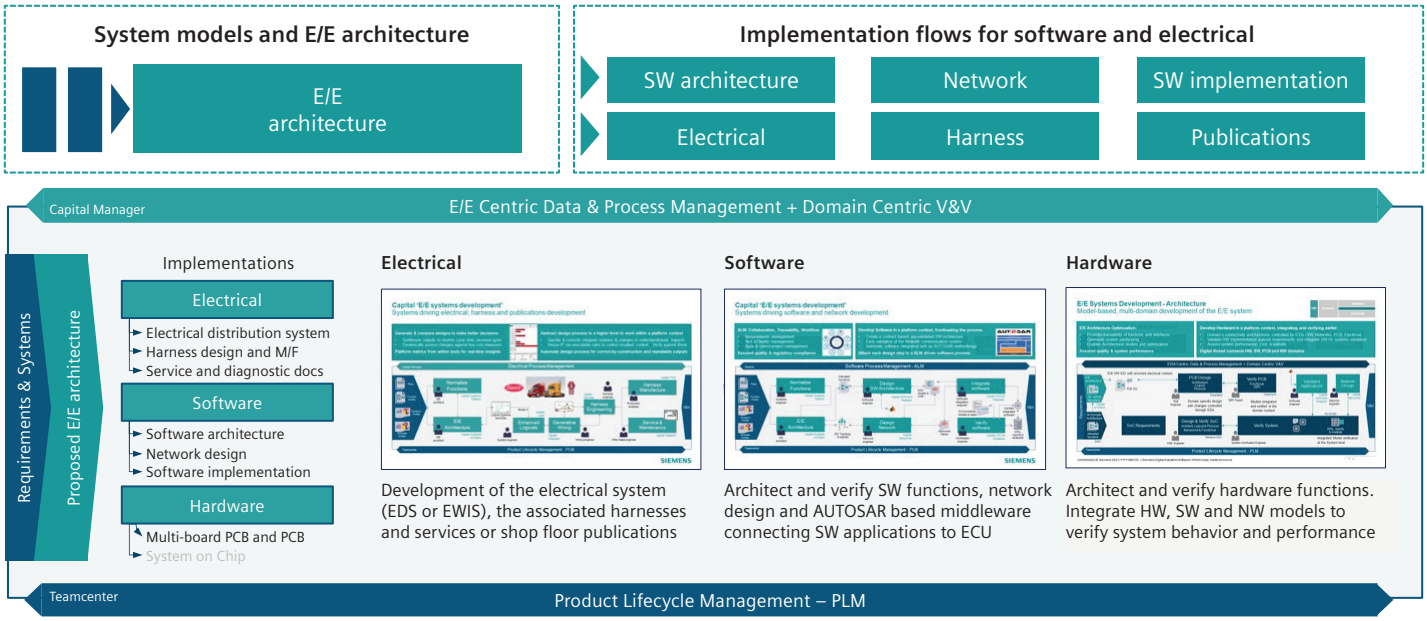


Figure 3: E/E Systems development using the Capital flow.

Capital's development flow can be used as a blueprint of the digital thread for the E/E systems development. In the flow, the mechatronics – the sensors and actuators – are isolated early on to the physical positions of the vehicle. The electrical and the hardware flows are used to connect these mechatronics and provide the execution platform for the software.

To address the increasing amounts and complexity of the software, a software-centric data and process management system is essential (see figure 4).

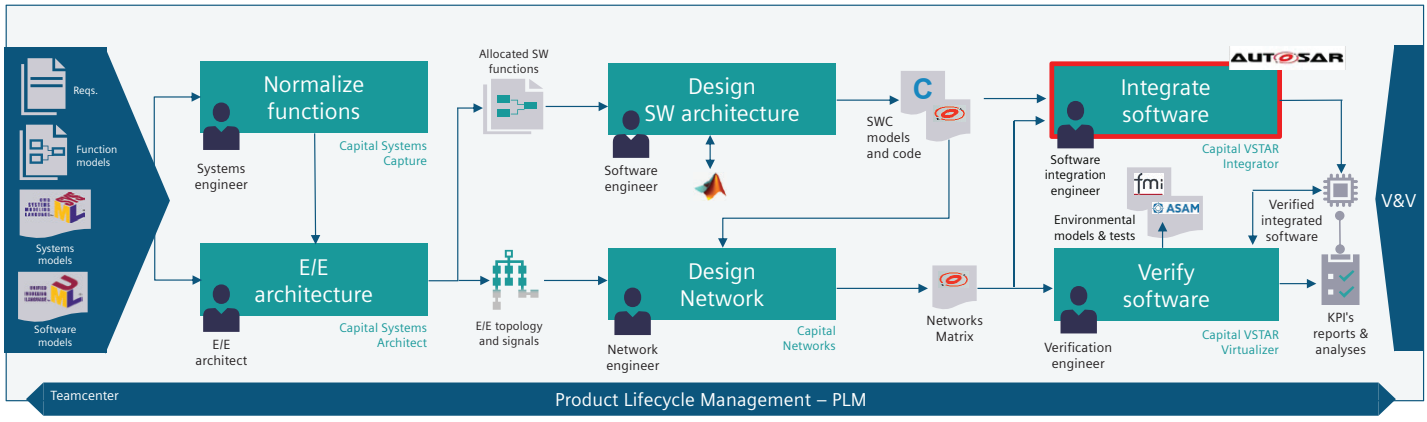


Figure 4: Software-centric data and process management system.

By developing software in a platform context, you can frontload the process and create a contract-based, pre-validated software architecture. Early validation of the network communication system is possible, and software integration is automated using the AUTOSAR methodology, enabling you to attach each design step to an ALM-driven software process.

For example, the Siemens Polarion tool can be used for requirements traceability and also to ensure that the compliance evidence is in place. Refining the software architecture at the vehicle level and connecting application communication to the network design is the next step.

Software integration and verification

Siemens' Capital VSTAR is a highly compliant AUTOSAR implementation and tooling enables streamlined ECU integration with a guided user interface, and automates software integration with an AUTOSAR methodology. Sophisticated configurations are managed by a generative development flow, enabling the development of high-performance embedded platform software that is scalable, safe and secure.

In figure 5, the AUTOSAR Classic platform is used for ECU basic software configuration and the application integration to target a scalable solution that is safe and secure.

Let's take a look at the various components:

1. **ECU extract.** The ECU extract describes the functional connectivity used for configuration of the middleware. The application and diagnostics connections are defined, in a file or files, supporting exchanges between tools and technical partners.
2. **ECU software.** The ECU software in this context is the functional application software that is hosted in this ECU. The integration tool uses this to generate application interfaces, network bus communication, etc.

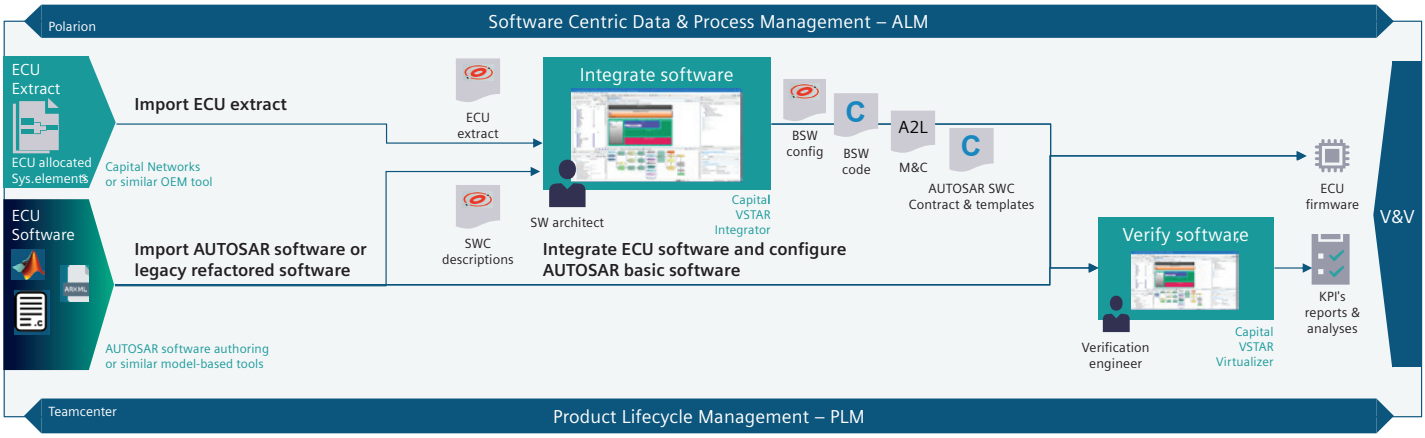


Figure 5. Integration of ECU allocated software on top of virtual or physical hardware.

3. **Interface to the applications:** Following the integration process, the interface to the applications is generated by the RTE.

4. **Application implementation:** The applications implemented and connected in the C code interface are used to build the binary.

At the same time, you also have a virtual representation – a digital twin. We can use the digital twin for verification and to analyze application behavior, to make sure that it behaves correctly. Additionally, the virtual environment enables you to investigate any issues and identify their root causes. The digital twin enables you to verify, analyze and report on whether you have enough headroom for software, or if you're running tight on resources, as well.

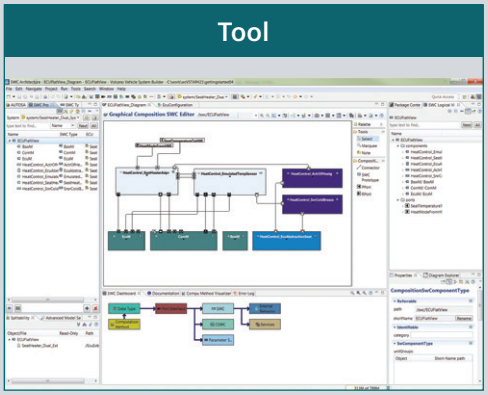
The new OEM branded operating systems describe a technology mix brought together by the OEM to serve as a technical base for their functional applications. This almost always includes both defined versions of AUTOSAR. As such, AUTOSAR will remain relevant for the future, and OEMs' future use cases will be defined and supported by these platforms.

As an example, Capital VSTAR is a complete AUTOSAR compliant solution that supports the development of innovative features for today and the future. Capital VSTAR provides:

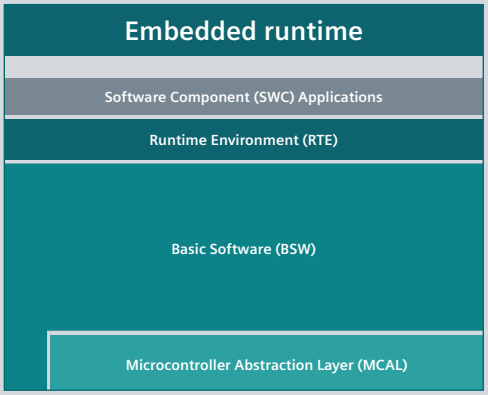
1. Performance via tooling that puts the developer first
2. Automation and virtual validation for right-the-first-time design
3. The ability to reuse software functions with future MCU and CPU families
4. Functional safety and cybersecurity
5. Fast deployment and multi-core support
6. Efficient customer enablement, with deployment and support services

Capital VSTAR consists of the Tool and the Embedded Runtime:

1. The tool connects to the digital stream and provides an automated workflow with the highest visibility of integration status and software behavior. The tool provides the configuration for the embedded software and output for further development activities, such as calibration.



2. The embedded implementation provides the required features to support the application needs, based on the provided configuration, with the right properties for efficiency, safety and security.



The software factory: developer-centric systems design

The rise in software factories requires organizations to hire and retain more software developers. Often, this means a differentiated culture for the OEM, or Tier 1 making this change. An important part of establishing these teams and retaining the talent is the use of processes and software tooling that enable developers to be innovative and add value with minimum wasted effort and repetition. Such tooling accelerates development cycles because they feature consistent, automated workflows, with assistance and checking features where needed, all supporting right-first-time design. Additionally, they support collaboration with open interfaces, enabling connections between tools within the factory and/or partner organizations.

Collaborative development requires protecting your Software IP. At the tool level, plug-ins, APIs and cloud-level services provide sufficient separation and stable interfaces. Capital VSTAR leverages the industry's established methodology for bus-level separation, where ECUs communicate over the bus interface that's defined either by the OEM's communication matrix or by using a standard such as SAE J1939.

Finally, for embedded testing, applications built using the AUTOSAR Adaptive methodology leverage the Portable Operating System Interface (POSIX), which enables you to load processes into memory during runtime and separate them by the MMU, to guarantee the interfaces and the separation and E/E system validation.

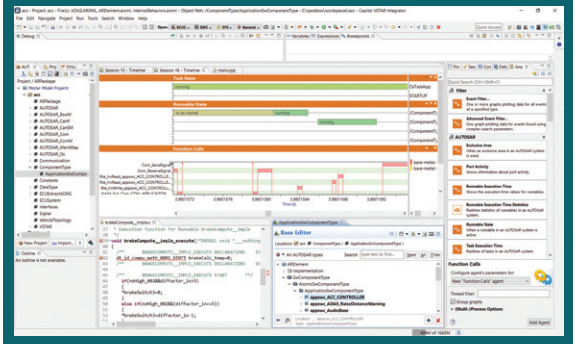
For AUTOSAR Classic applications, Capital VSTAR provides the ability to separate the objects utilizing the RTE interfaces.

In the software factory, the virtual environment is used to provide application-level interfaces for integration and verification (see figure 6).

- Virtual ECU simulation model of actual hardware
- Execute actual ECU application software
- Use for system driven development
- Connect to standard development tools
- Allows testing, profiling, tracing like real ECUs

Benefits

- Identify application problems early
- Avoid dependency to real hardware
- Share data and results globally
- Deep, non-intrusive debug
- Flexible scalability up to MCU Emulation



Virtual ECU enables rapid debug by software developer on their laptop

Figure 6: Application-level interfaces for integration and verification in the virtual environment.

The virtual ECU enables software developers to perform rapid and early debugging on a laptop, without having to have the physical chip available, using standard development tools that make it possible to move between the virtual environment and the embedded environment. Through the virtual environment, Capital VSTAR democratizes developer access to the target environment, where you can provide virtual stimuli to the applications that are developed using standard interfaces such as FMI.

Considerations for establishing a software factory

Here are some of the critical elements of an efficient software factory:

1. Accelerate the adoption of software-specific development processes using focused organizations. Using the standardized tools really helps to lower the entrance and gives you access to a much wider pool of developers.
2. Rapidly scale the software organization using developer-focused tooling and methods.
3. Leverage state-of-the-art software development practices, including Agile methodologies to improve quality, CI/CD for repeatability and feedback, digital twins throughout development and traceability for sign-off and certification.
4. Maintain abstractions through model-based practices and generative tools. This includes developing a vehicle software platform that supports adding new functionality over the vehicle's lifespan, and building services and applications that can be widely deployed and reused.
5. Use service-oriented architectures to provide a centralized, scalable compute node to support software updates throughout the life of the vehicle, to extend functionality without requiring a complete redesign.

Conclusion

In today's rapidly changing E/E embedded software development environment, OEMs must develop high-quality software with increasingly complicated interoperating functions at an unprecedented pace. Automakers face increasing complexity of in-vehicle systems and continuous pressure regarding cost and timing. Meanwhile, the increasing dependence on software has given rise to new, more stringent regulations, making collaboration critical for OEMs when building future systems. Developers need to have an efficient, automated workflow that removes repetitive tasks, eliminates manual errors, and gives additional tool assistance when required to collaborate engineering activities.

Tier 1 OEMs are developing Software Factories and leveraging strategic software platforms and proprietary operating systems. New relationships between OEMs and different tiers of suppliers are required, and standards are essential to enabling effective collaboration. AUTOSAR is designed to support a wide variety of functional use cases and standards, and the interoperability necessary to enable innovation.

Siemens' AUTOSAR platform, Capital VSTAR, offers an integrated software development flow as part of a full-system development environment to support collaboration, design validation, and interoperability with a high amount of automation, saving time, improving quality, and minimizing the need for expensive hardware validation. Capital VSTAR empowers E/E development teams with the tools, capabilities, and services necessary to meet evolving requirements for designing sophisticated modern vehicles now and into the future.

Learn more about Siemens solutions for the Automotive industry here.

Siemens Digital Industries Software

Americas: 1 800 498 5351

EMEA: 00 800 70002222

Asia-Pacific: 001 800 03061910

For additional numbers, click here.

About Siemens Digital Industries Software

Siemens Digital Industries Software is driving transformation to enable a digital enterprise where engineering, manufacturing and electronics design meet tomorrow. Xcelerator, the comprehensive and integrated portfolio of software and services from Siemens Digital Industries Software, helps companies of all sizes create and leverage a comprehensive digital twin that provides organizations with new insights, opportunities and levels of automation to drive innovation. For more information on Siemens Digital Industries Software products and services, visit siemens.com/software or follow us on LinkedIn, Twitter, Facebook and Instagram. Siemens Digital Industries Software – Where today meets tomorrow.

About the authors

Mathias Fritzson, Senior Product Manager, Siemens Digital Industries Software

Mathias has spent nearly 20 years of his career in embedded development and design of software and multiplexed communication for the automotive industry. His interest in automotive systems and technical curiosity has resulted in key assignments, scaling from software engineering on large-scale ECU production programs to research programs in functional safety, base technology development, standardization, and verification on bus technology. Mathias has spent the last decade advocating for the use of the AUTOSAR standard while conducting product development and making customers successful. Currently, he is Product Manager for Siemens Capital VSTAR, overseeing product development and marketing activities. Mathias earned his Master's degree in electrical engineering from Chalmers University in Gothenburg.

Brendan Morris - Senior Technical Marketing Engineer

Brendan spent his early career working in Tier 1 suppliers into the Automotive Industry, focusing primarily on software development in powertrain electronics. He spent some time working in a diverse range of technologies for on- and off-highway vehicles. Brendon spent a decade working for several vehicle OEMs at all stages of vehicle development programs, leading research projects at Jaguar Land Rover, introducing new network technologies into their latest and future E/E Architectures, and representing the company in the AUTOSAR WP-A2 Com Work Package. He has also led several aspects of development and launching of low-volume vehicles, including the E/E architecture, as well as start-up vehicle OEMs as diverse as McLaren Automotive and Rivian. Brendan holds a Master's degree in Automotive Engineering from Loughborough University.

siemens.com/software

84607-D4 5/22 H