The Car’s Electronic Architecture in Motion:  
The Coming Transformation

Die elektronische Architektur des Autos in Bewegung:  
Die kommende Transformation
Abstract

Our paper explores carmakers’ practical challenges in deploying new Electrical and Electronics E/E architectures to manage the ever-increasing cost and complexity of vehicle electronics. We also highlight the product and technology innovations from Vitesco Technologies, which are enabling carmakers to navigate these challenges and bring their future E/E architectures to life. We describe the family of electronic Master Controllers, which will play an important supervisory role in the vehicle’s future motion control E/E architecture. We address solutions for managing the complexity of large-scale software integration needed on such Master Controllers, via technologies such as “AUTOSAR Classic Flexibility” and “Virtual Applications”. These technologies allow software systems to be split into separately programmable Software Clusters, independently developed across many companies and eventually integrated with minimal impact to the overall system. Like Apps on a smartphone, Software Clusters are connected to the rest of the software system after the programming process. The use of AUTOSAR Classic Flexibility allows carmakers to reuse their legacy software assets in a smooth transition to new vehicle E/E architectures. Our paper describes Vitesco Technologies’ electronic hardware and software solutions for future E/E architectures and highlights key technical roles required to manage the complexity of future software architectures. Finally, we highlight the new business models between carmakers and their suppliers which our software technology unlocks.

Kurzfassung

In unserem Artikel werden die praktischen Herausforderungen der Automobilhersteller bei der Entwicklung neuer Elektrischer und Elektronischer (E/E) Architekturen untersucht, um die ständig steigenden Kosten und die Komplexität der Fahrzeugelektronik zu bewältigen. Wir stellen die Produkt- und Technologieinnovationen von Vitesco Technologies vor, die es den Automobilherstellern ermöglichen, diese Herausforderungen zu meistern und ihre zukünftigen E/E-Architekturen zum Leben zu erwecken.

Insbesondere das Problem der groß angelegten Software-Integration wird angesprochen und Lösungen dafür wie AUTOSAR Classic Flexibility und virtuelle Anwendungen werden näher erläutert. Diese Techniken ermöglichen eine unabhängige und agile Entwicklung von einzelnen Teilen eines Softwaresystems.


Dies trägt dazu bei, die Software-Assets der Autohersteller zu erhalten, in die sehr viel Entwicklungslaufzeit geflossen sind, und ermöglicht einen risikomindernden und reibungslosen Übergang in neue Fahrzeugarchitekturen.

Neben der reinen Softwarelösung geben wir auch einen Überblick über das erwartete Arbeitsmodell, das mit diesen Technologien einhergeht.

Abschließend werden wir die Geschäftsmodelle ansprechen, die mit diesen Softwaretechnologien ermöglicht werden.
Evolution of the Electrical and Electronics (E/E) Architecture

From their early days in basic engine control, powertrain electronics have evolved into sophisticated computers for precise combustion management, exhaust after-treatment and drivetrain control. Yet, that forty-year evolution pales in comparison to the transformation of the entire vehicle’s electronics and software architecture over the coming decade.

Over the years, new features for improved emission control, vehicle safety and comfort were incrementally introduced into the vehicle via new electronic control units (ECUs). The modern car now already contains as many as 150 ECUs which communicate with each other over in-vehicle networks such as CAN, Ethernet or Flexray. This incremental expansion of the E/E landscape in a “distributed architecture” has, over the years, become the state of the art at most carmakers. The Tier-1 ECU supply base and their silicon sub-suppliers have correspondingly evolved their product roadmaps to fit into the car’s distributed E/E architecture.

Yet, the automobile is now in the midst of an irreversible transformation from a stand-alone internal combustion engine (ICE) propelled vehicle, towards an electrified and fully connected supercomputer-on-wheels. The triffecta of electrification, connectivity and autonomous driving require a plethora of new features in human-machine interface (HMI), advanced driver assistance systems (ADAS), predictive energy management, over-the-air software updates and the all-important cyber-security protection. As a result, the amount of software in the car continues to increase exponentially, with some estimates placing the number of lines of code in the premium car at over 700 million between 2025-2030, compared to an already massive 100 million lines of code in 2015 (1).

Although the powertrain constitutes a fraction of the total amount of software in the future car, it is nevertheless a showcase for ever-increasing complexity. Electrification will be achieved via several flavors of hybrid and pure-electric architectures, which multiply the number of powertrain variants that most large carmakers must manage. This increase in powertrain variance not only drives a more diverse E/E landscape across the car fleet but also drives more software content for predictive energy management via trajectory planning, thermal management for optimal use of battery energy, and ‘fail-operational’ capability to cope with ASIL-D safety requirements.
The cost of managing the explosive increase of E/E and software content is uneconomical within existing architectures and is quickly bringing carmakers’ E/E architectures to their breaking point. Carmakers globally have now embarked on the journey to simplify their E/E landscape to cope with the new paradigm. They pursue a single direction – reduce the number of ECUs in the car and manage ever-growing complexity with smarter software architectures. In this sense, the “North Star” E/E architecture is a small number of high-performance “servers” for computation, supported by several smart actuators for I/O, completely dissolving the traditional domain boundaries between powertrain, chassis and car-body. While a few automotive newcomers with a small portfolio of pure-electric cars can immediately apply such a “North Star” concept, most established carmakers have extensive legacy in their car fleets and organizational structures which prevent them from making a direct jump to such a server based architecture.

This paper delves into the practical architectural approaches that established carmakers can use to manage the new electronics and software paradigm, and the solutions Vitesco Technologies offers, to facilitate carmakers’ journey to bring their future cars to life.

Trends in E/E Architecture

 Architects imagine a simple E/E constellation borrowed from the IT-world as a “North Star”: a few centralized servers for computation and distributed smart actuators for real-time mechatronic control. Centralized servers, consisting of a few high-performance computers, enable the simplest electronics landscape in a car, without restriction from legacy electronics, software and company organizational structures. The servers enable delivery of attractive, yet computationally intensive user experiences such as those found on mobile phones, into the car via the carmaker’s “App store”. However, most established car companies have a long legacy of car platforms which must be maintained in parallel with
the desire to introduce new features via simple E/E architectures. This legacy includes millions of lines of software code, deployed across several distributed electronic control units, evolved over decades as a way to deploy an extensive vehicle fleet across multiple global markets. A single leap from the current distributed architecture of as many as 150 ECUs, towards a server-based architecture centered around a handful of high-performance computers is impractical for most established carmakers.

A more practical solution emerges when one considers a more targeted E/E simplification, by unifying functionally related “domains”. Powertrain and Chassis form such a pair of related domains to integrate functions for managing “vehicle motion”. This cross-domain integration not only provides for a logical grouping of all software related to vehicle motion control, but also enables a reduction of ECUs in the car. Although this cross-domain integration results in a collision between longstanding organizational structures for Powertrain and Chassis, the benefits are significant enough that many carmakers are pursuing this option for their next generation of E/E architectures.

Integration of the Powertrain and Chassis domains requires a special purpose ECU, which we henceforth refer to as a “Master Controller”, to play a supervisory role over the combined domains. The Master Controller provides the incremental integration step over the currently separate Powertrain and Chassis domains, and allows for the reuse of existing knowhow, legacy software, and release methods to minimize risk of change, while still making a tangible step towards the “North Star”.

As shown in Figure 3 Vitesco Technologies has developed a scalable family of ECUs to cover a range of future Master Controllers, from which carmakers select the appropriate device-type based on their specific E/E architecture migration needs:
> **Powertrain Domain Controller:** The basic Master Controller variant is based on single microcontroller technology and offers high-speed communication datalink for gateway functions and I/O circuits for sensors and motor control.

> **Cross-Domain Controller:** More computational power to enable integration of Powertrain and Chassis domains, or as a Zone Controller. Contains two microcontrollers, high speed datalink, I/O and special purpose circuits such as for battery charging supervision. Such a device would typically serve as a Vehicle Motion Master Controller.

> **In-Vehicle Servers:** Highest computational power with up to two microcontrollers and an additional microprocessor. Typically, does not contain I/O circuitry but provides a large-scale software integration platform.

### Managing Software Complexity

While cross-domain integration via a Vehicle Motion Master Controller reduces the number of ECUs in the car, it greatly increases the complexity of software inside the device. The Master Controller is not only the home of legacy energy management and vehicle stability functions, but also contains a slew of new functions to manage electrification, connectivity, and the impact of autonomous driving. Figure 4, which compares estimated relative complexities of selected electronic control systems, shows that a Vehicle Motion Master Controller is expected to contain 8 million lines of C-code by 2025.
The Vehicle Motion Master Controller is 3 to 5 times larger than Euro-6 Gasoline Direct Injection Engine Management Applications today. (Source own data and (1))

To cope with the high computational load, Master Controllers can feature multiple high-end multicore microcontrollers and can be further scaled up with microprocessors and AUTOSAR Adaptive operating environments. Furthermore, as the software content in a Master Controller extends beyond traditional organizational boundaries at carmakers and Tier-1s, an increasing number of parties are involved in software development, making multi-party software integration a critical factor in the robust performance and update-ability of the device over its lifetime. Figure 5 shows the expected dimensions associated with a typical Vehicle Motion Master Controller development.

One of the challenges of managing large software content on a single ECU is the required re-integration and re-qualification of the entire software set, if even small changes or bugfixes made to one specific function. For a Vehicle Motion Master Controller, which contains legislative homologation and safety critical functions, such extensive re-qualifications are time-consuming and expensive. Therefore, a key success factor for ECUs with large software content is the ability to independently modify and reprogram each individual function, without affecting any of the other software. Just as the update of a single App on a mobile phone does not require a rebuild and re-qualification of the entire phone’s software, individual functions of a Vehicle Motion Master Controller must be independently update-able in the field without requiring requalification of the system.
Carmakers therefore need a software architecture and supporting technologies which reconcile seemingly contradictory goals:

- Enable large-scale integration of software from many different organizations or suppliers.
- Allow App-like over-the-air updates of individual functions.
- While still maintaining the qualities of a robust hard real-time control system.

To answer these objectives, Vitesco Technologies has enhanced the AUTOSAR Classic Software mechanisms for virtualization on microcontrollers. This enhancement, called “AUTOSAR Classic Flexibility”, now enables carmakers to cope with the above challenges, on legislation and safety critical Vehicle Motion Master Controllers.

“AUTOSAR Classic Flexibility” Concept and Virtual Applications

This chapter explains the AUTOSAR Classic Flexibility (7) concept - henceforth abbreviated as “AR-Flexibility” - its advantages and the consequences on project technical roles and development workflow. We also explain the concept of Virtual Applications, which further extend the AR-Flexibility principle.

Basic Principles

Figure 6 shows the differences between AUTOSAR Classic (AR-Classic), which is the current state of the art, and the extensions with AR-Flexibility and Virtual Applications. The AR-Classic approach is that of one monolithic software system where the application software is connected via the Run Time Environment (RTE) to the basic software and finally the microcontroller hardware. In AR-Classic, the application software is divided to software components (SWCs) representing individual C-file modules. SWCs can be aggregated into structures of the software system. AR-Classic has long been the state of the art for powertrain related software and while it serves the purpose of architectural decomposition of a complex software structure, it has limitations when the size of the software is significantly scaled up, such as in a Vehicle Motion Master Controller.
The hierarchical AUTOSAR model defines internal and external “ports” via which modules communicate, which need to be connected when the software monolith is compiled and linked. This approach, while robust, carries the disadvantage that a change in just one of thousands of software components could require regeneration of the entire software image. For smaller systems (~2 MB, ~400 SWCs) this approach is manageable (4). As the systems become bigger (e.g. 16 to 20 MB, ~4000 SWCs) the time for software builds and debugging increases exponentially. A monolithic approach would hinder development of a Vehicle Motion Master Controller, which is expected to have dimensions as shown in Figure 4.

This is where the AR-Flexibility helps to divide the Application Software into smaller “Software Clusters” which can be managed independently from each other and are only connected to the rest of the system by an embedded linking process based on a binary manifest technology, explained further in the next chapter. For additional independence of software functions, it is also possible to decouple parts of Basic Software from each other and allocate parts of the system to independent execution units in the ECU called Virtual Applications which are explained further in this paper. A Virtual Application can consist of several Software Clusters and is executed on either a given microcontroller on the ECU, or across the various computing cores of a single microcontroller. As Software Clusters in AR-Flexibility separate software units in computer memory, Virtual Applications extend that to a separation in time. This means that if one Virtual Application in the system malfunctions, it will not influence or halt execution of all the other Virtual Applications, within the limits of the underlying micro controller architecture.

Connecting Multiple Software Clusters

With AR-Flexibility, the software system is divided into independent build units resulting in one executable binary per Software Cluster. But how are the connections between the software clusters established?

![Figure 7: Basic principle of the connection mechanism between software clusters, using the binary manifest.](image)

Every Software Cluster has a so-called binary manifest in a predefined memory section. The binary manifest shows all input and output ports of that Software Cluster, as shown in Figure 7. After the software is built, the output port is complete and holds references to the data or C-functions and a unique identifier for the port. At the time of the software build, the input port remains incomplete, containing only the unique identifier of the input port. The connection process starts after successful programming of one or more Software Clusters into the ECU. A special software component in the system boot loader scans all
binary manifests and maps all provided ports to the manifests that need them, based on the stated unique identifiers. This means that the incomplete input ports will be enriched with reference to the port of the provider and stored persistently in the micro controller’s flash memory. At system start, all connections are already established, resulting in a similar startup behavior as in today’s AR-Class systems.

Hence, the traditional linking process in a software build is replaced by a programming and connect process on the device. But what happens if an interface is not available? In such a case, the AR-Flexibility concept foresees that a required port needs to be resolved by a default handle which is a calibratable constant that delivers a neutral functional behavior.

**Types of Software Clusters**

An AR-Flexibility software system contains two types of Software Clusters – an arbitrary number of Application Software clusters and a single “Host Software Cluster”, as illustrated in Figure 8. The Host Software Cluster contains the operating environment for the ECU, including basic software, the operating system (5), service functions like error management, non-volatile memory management and the inter-ECU communication stacks (6). The Host Software Cluster contains all the pre-defined operating environment services for all the Application Software Clusters.

Figure 8: Software Clusters communicate via the Binary Manifest and Proxy Modules in the Software Cluster Connect layer. The Host Software Cluster provides the Services including the Communication Stacks for the inter ECU communication.

An Application Software Cluster doesn’t contain any basic software, but does have its own RTE. The RTE accesses the service functions in the Host Software Cluster indirectly, via the binary manifest. RTE generates its operating system tasks, which are invoked from the Host Software Cluster via an automated mechanism built into the AR-Flexibility supporting toolchain.
Advantages of AUTOSAR-Flexibility

1. **Independent Targeted Update of Individual Functions**
   A given function can be updated independently from any other. The update will have no collateral effects on the other functions present on the device. AR-Flexibility can therefore be used for elementary feature migration from legacy to new vehicle architectures, as well as for efficient variant management.

2. **Reduced Re-Validation Effort**
   Software re-validation effort upon making a change, can be limited to test cases for interfaces to the other functions. Each Software Cluster can be pre-validated on its own.

3. **Separation of Functions (e.g. OBD/non-OBD)**
   One can physically separate OBD relevant homologation functions from all other parts of a Vehicle Motion Master Controller.

4. **Reduced Integration Complexity**
   Enables the large software image of the Vehicle Motion Master Controller to be cut into Software Clusters so as to minimize dependency across clusters, hence reducing the complexity of system integration. Debugging is limited to the software components within the Software Cluster, hence reducing the problem space.

5. **Reduced Build Time**
   Due to the minimal dependence across Software Clusters, even a full software image build can be done for several Clusters in parallel, hence reducing the total software build time.

6. **Enables Agile Development**
   A full software build of a Euro-6 gasoline direct injection engine management system (8MB code including calibration data) based on AR-Classic needs several hours, which doesn’t suit agile development with continuous integration and test (CI/CT). A system which is divided into e.g. 8 software clusters each ~1 MB flash size correspondingly reduces build time into a range which is practical for CI/CT.

7. **Clean Work Distribution Across Teams**
   AR-Flexibility enables a split of responsibility across different development teams with minimal cross-dependency and the following clear deliverables:
   > An executable binary of the Software Cluster built against a clear contract,
   > at the predefined memory location,
   > with the binary manifest as connector to the rest of the system.
Working Model

Figure 9 shows the key roles on a development team that works with AR-Flexibility. The Software System Architect is a principal role which ensures that the content and the relationships between Software Clusters always remain coherent.

![Role model for working with AUTOSAR Classic Flexibility having the SW System Architect as central role.](image)

The Software System Architect formally describes the structure of the software system and the relationships between Software Clusters. Software Clusters are designed by the respective Software Cluster Architects. Because the Host Software Cluster has a special role in the system, the interaction between the Software System Architect and the Host Software Cluster Architect must be especially strong. The Software System Architect provides unique identifiers to each output port of a Software Cluster. Once the integration frame is defined, the Host Software Cluster Architect releases the intended memory layout and the global resource identifiers for each Application Software Cluster. From this point onwards, every Application Software Cluster team can develop, build and release his own Software Cluster independently from all other Software Clusters, as shown in Figure 10.

![Independent work on SW-Clusters against a common defined integration frame.](image)

Extension with Virtual Applications

Software Clusters share the same basic software and hardware resources of the microcontroller, including its cores. If two Software Clusters are allocated to the same core and one of them were to violate timing or malfunction such as a floating point exception, or a memory access violation, the relevant computer core, and with it the other Software
Cluster would undergo a system reset. In some cases, such a system reset is not appropriate – for example, a qualified and released OBD or safety monitoring relevant function in one Software Cluster cannot be disabled by a malfunction in another Software Cluster. In this case, the software architecture needs to provide a further strong level of separation between Software Clusters.

![Figure 11: Software system for virtual applications including the virtual application manager and the mapping of cores and peripherals.](image)

The level of additional separation is achieved via Virtual Applications (vApp). A vApp consists of one Host Software Cluster and one or more Application Software Clusters. Each vApp is, to the extent allowed by underlying hardware architecture, exclusively mapped to microcontroller cores, peripherals (e.g. CAN, FlexRay) and hardware elements (ASICs, Transceivers) as shown in Figure 11. In cases such as microcontroller peripherals, where exclusive mapping to a vApp is not possible (such as for hardware security module or hardware accelerators) a software virtualization mechanism, called Virtual Application Manager provides the needed vApp separation.

Vitesco Technologies has already demonstrated the Virtual Application approach even on the current generation of microcontrollers such as the Infineon AURIX 2nd Gen. As a result of successful demonstration of AR-Flexibility and vApp concepts, the next generation of microcontrollers intended for production in 2025 and beyond will include superior hardware support for vApps.

**Selecting the Right Concept for Your Application**

While the AR-Flexibility and vApp concepts do utilize additional ECU memory to realize the separation concepts, they are still the more economical solution to other alternatives - such as separate microcontrollers - for realizing such separation of software.
Figure 12 shows levels of separation offered by the Vitesco Technologies’ software platforms, and the types of applications that they are suitable for.

**AR-Flexibility and Virtual Applications Enable New Business Models**

A Vehicle Motion Master Controller is a platform for large-scale software integration. Software clusters structured as Apps on a mobile phone can be independently developed, independently downloaded and executed on their defined spaces in the ECU. AR-Flexibility and Virtual Application mechanisms provide the necessary independence from interference across Software Clusters. In doing so, they also unleash new possibilities for business models between carmakers and their suppliers. Most importantly, they enable an independence of development-cycle and procurement between hardware and software. The carmaker can define and procure ECUs from one supplier and procure software and integration services from a multitude of other suppliers, who can follow different development and integration/release schedules.

The larger the software content in an ECU, the larger is the group of contributing parties, each with specialist knowhow in certain areas, and the more difficult is the software integration and qualification. AR-Flexibility and Virtual Application mechanisms enable a large number of software suppliers to independently deliver release software for the ECU. The carmaker’s critical responsibility is in a robust system architecture definition and allocation of ECU resources to Software Clusters. Having established the architecture, the carmaker is then free to outsource software development to the most qualified parties, either internal or external to their company. Integration of the independent Software Clusters can either be done by the carmaker, or by their Tier-1 ‘Anchor Partner’ such as Vitesco Technologies, who has the skills and system knowhow to perform such a task. Table 1 shows a potential job-split scenario for important system development functions.
**SYSTEM REQUIREMENTS AND ARCHITECTURE**
- **Carmaker**: Define & release
- **Anchor Partner (Lead Integrator: Tier-1 Hardware & Software expert)**: Support
- **Tier-2 Software Specialist (internal / external)**: Use

**HARDWARE**
- **Carmaker**: Design & deliver
- **Anchor Partner (Lead Integrator: Tier-1 Hardware & Software expert)**: Use

**SOFTWARE OPERATING ENVIRONMENT**
- **Carmaker**: Create & release
- **Anchor Partner (Lead Integrator: Tier-1 Hardware & Software expert)**: Use

**FUNCTIONAL REQUIREMENTS**
- **Carmaker**: Define & release
- **Anchor Partner (Lead Integrator: Tier-1 Hardware & Software expert)**: Support
- **Tier-2 Software Specialist (internal / external)**: Comply

**FUNCTION REALIZATION**
- **Carmaker**: Create, qualify and release

**SYSTEM INTEGRATION & RELEASE**
- **Carmaker**: Validate & deploy
- **Anchor Partner (Lead Integrator: Tier-1 Hardware & Software expert)**: Integrate and release

| Table 1: Potential job-split between carmaker, Tier-1 Anchor Partner and Tier-2 Software Partners. |

**Conclusion and Outlook**

The need for simplifying the car’s E/E architecture is of paramount importance to carmakers who must, at once, introduce new technology for electrification, connectivity and autonomous driving, while also managing large diverse vehicle fleets in global markets. We expect the journey towards the E/E architecture “North Star” to be a stepwise one for most established carmakers. Most carmakers will migrate their architectures to a cross-domain structure after 2025, where Powertrain and Chassis domains will be merged. Vehicle Motion Master Controllers will play a big part of this transformation and Vitesco Technologies, a longstanding leader in powertrain electronics remains a leading supplier.

Yet, the future belongs to those who can master the software challenge. The ability to efficiently outsource and then integrate large-scale software systems into such Master Controllers will become a carmaker’s competitive advantage as the vehicle becomes a super-computer on wheels. Vitesco Technologies’ innovations such as AR-Flexibility and Virtual Applications are the critical tools that every carmaker needs to manage software complexity in Master Controllers. AR-Flexibility has now been incorporated into AUTOSAR standard 20/11 (7) and will be supported by 3rd party tooling, for programs starting production in 2025.

Vitesco Technologies continues to lead the charge in electronics and software innovations to help carmakers bring their future E/E architectures to life!
References:

2. https://www.nxp.com/design/training/zonal-architecture-for-future-cars:TIP-NXP-AUT-T4068