Paving the way for the "Software Defined Vehicle"

Enabling future mobility by mastering software

Martin Schleicher, Continental, Frankfurt

Abstract
The “Software Defined Vehicle” enables new business models, software-hardware separation, shorter development cycles, data-driven development as well as virtual development. Key building blocks for the Software Defined Vehicle are a) Scalable Computing Platforms, b) Operating System and Middleware, c) Cloud, d) Toolchain and DevOps Workbench. The Software Defined Vehicle requires close collaboration between the automotive and software industry.

Overview, Introduction
The automotive mega trends “CASE” (Connected, Automated, Shared, Electrification) have one thing in common: they are dependent on electronics and even more on software. The "Software Defined Vehicle" enables new consumer functions and user experiences through software. The increased importance of software has a significant impact for the automotive industry.

Vehicle Architecture Transformation
The current vehicle electric/electronic (E/E) architectures were useful to realize specific functions implemented in single ECUs (Electronic Control Units). With the growing complexity of functions, and the extra-ordinary growth of software and connectivity this approach has been reaching its limits and new approaches for EE architectures with High-Performance Computer (HPC) are introduced by many OEMs:
EE architectures are going to evolve in several steps from current distributed systems to domain-centralized networks with a central HPC per domain to zonal (zone-based) architectures:

Fig. 2: Example for Server / Zone (zonal) architecture

While transforming the vehicle architecture, OEMs and suppliers need to continue building automotive-grade systems: reliable, safe and usable over a long lifetime. At the same time consumers expect at least the same level of quality and availability for software as they are familiar with from their mobile phones and computers. To be able to continuously update...
software over the complete vehicle lifetime, automotive companies have to develop software like IT-companies: delivered in short release cycles by applying modern agile and lean methods.

**The Software Defined Vehicle**

To successfully master the "software defined vehicle", the automotive industry needs to adopt thinking and acting from software companies and master several dimensions of software: architecture, development methods, culture, and business models. New central architectures are the technological foundation for the Software Defined Vehicle, an

**Goals of the Software Defined Vehicle:**

- New business models such as “Software as a Product”, and “Software as a Service” as well as “Function on Demand”
- Software separation and consolidation from hardware
- Shorten development, test, and integration cycles
- Data-driven development
- Virtual development

To support these goals, on the one hand the software architecture has to fulfil key characteristics which are usually part of the Operating System and middleware of the vehicle. On the other hand, it is also important how the software is being built and deployed to a vehicle.

**Key characteristics of the Software Defined Vehicle:**

- Service Oriented Architecture enabling SW-HW separation
- Containerization to support isolation and density of software
- Cloud enablement
- (Over-the-Air) Updates for continuous software deployment to vehicles
- Agile and lean Methods used in development
- DevOps, Continuous Integration and Automated Testing

**Service Oriented Architecture**

The decoupling of functions from ECU hardware calls out to use modern software concepts like “Services”.

**Definition Services** (Source: Wikipedia)

- It logically represents a repeatable business activity with a specified outcome
- It is self-contained
• It is a black box for its consumers, meaning the consumer does not have to be aware of the service's inner workings. (API, REST, Stateless)
• It may be composed of other services.

According to this definition “Service” is a generic concept. This has shortcomings, such as Services can be relatively “big”, e.g. think of a Navigation Service or relatively “small”, e.g. Get Current GPS Position. Hence more advanced concepts are being introduced, the so called “Microservices”.

Definition Microservices (Source: Wikipedia)
• Services in a microservice architecture are often processes that communicate over a network to fulfill a goal using technology-agnostic protocols such as HTTP
• Services are organized around business capabilities
• Services can be implemented using different programming languages, databases, hardware and software environment, depending on what fits best
• Services are small in size, messaging-enabled, bounded by contexts, autonomously developed, independently deployable, decentralized and built and released with automated processes

By implementing a Service-oriented architecture using Microservices, Functions and Applications can be separated from ECUs and HPCs. Thus, Functions and Services can be distributed in a vehicle network, which is typically based on TCP/IP (Ethernet).

Fig. 3: Service Oriented Architecture

Containers as a means to organize Services
Containers are a concept to separate software applications and functions.
Description Containers:
(Source: State of the Union of Microservices and Containers – Stackify)
• Containers are the tools and methodology used to organize and develop microservices.
• Think of a container as the packaging of software, including everything it needs to run: code, runtime, system tools, system libraries, and settings.
This container completely isolates the software from any other software in the same environment.

This allows for different teams to work on different microservices simultaneously.

Fig. 4: Density and isolation of processes, containers, virtual machines for software

By using Containers applications can be packaged with the respective services and deployed to a distributed architecture.

Fig. 5: Container in a Service-oriented Architecture

Flexible deployment of application and services, and access to data

Microservices and containers enable decoupling of software functions from ECU hardware, as well as consolidation of software in HPCs. Redundancy and fail-safe concepts can be implemented on top of a Microservices and container architecture. Operating Systems with Platform services enable relocation and deployment of applications and services “horizontally” within a system, depending on needs for computing power and determinism (real-time) needs as well as bandwidth in the network.

Modern cloud-enabled Operating Systems and appropriate Middleware enable orchestration and consumption of microservices also from the cloud, as well as deploying containers flexibly between the vehicle and cloud. Such an architecture also enables transfer and sharing of data between vehicle and cloud.
Over-the-Air (OTA) Updates enabling continuous deployment of software

Consequently, Software Defined Vehicles need to be updatable. Fundamentally there are two types of software updates:

- Bug fixes, Security, and safety updates
- Deployment of new features

In Automotive Systems for Over-the-Air updates there are specific requirements according to UNECE. A real Software Defined Vehicle requires a comprehensive solution (cloud and vehicle) and corresponding strategies to efficiently update all ECUs and HPCs within a vehicle.
Agile and Lean Methods, DevOps

Projects for new vehicle architectures are software-focused with a high complexity, huge number of requirements and continuous stream of changes.

Fig. 8: Example of one of the latest High Performance Computer projects at Continental

Such projects can be managed applying Agile and Lean concepts, in particular for large projects, such as LeSS (Large Scale Scrum) or SAFe (Scale Agile Framework). At Continental we apply SAFe in complex projects.

Software systems with millions of Lines of Code, many developers and continuous long-term development of new software releases, requires the consequent application of Continuous Integration (CI) and Continuous Deployment (CD) in combination with Automated Testing.

Fig. 9: CI/CD toolchain for High Performance Computer

Collaborative Development and Continuous Integration

The above example indicates how many parties and engineers are contributing to a software defined vehicle. In addition to CI/CD this also requires special focus on managing the
collaboration of the various parties that deliver software for a Software Defined Vehicle. At Continental we have created the “Continental Collaboration Portal” to enable systematic approaches for delivery, test and integration of various software suppliers.

**Fig. 10: Collaborative Development in High Performance Computer projects**

**Continental Automotive Edge Framework**

With the "Continental Automotive Edge Framework" we are creating the foundation of this evolution, consisting of the following building blocks:

- Scalable high-performance computer hardware
- Vehicle operating system and Middleware
- Cloud software
- Toolchain and DevOps Workbench

These building blocks are closely linked, but need to be combined modularly and flexibly, as we are supporting different types of HPCs and ECUs that will be applied to different vehicle domains.
To support the needs of future vehicle architectures, with central, high-performance vehicle computers and zonal power & signal distribution architectures, we are able to build flexible and scalable HPC hardware systems, serving demanding applications for ADAS and Automated driving as well as Body and Cockpit systems.

The middleware is based on Elektrobit’s Automotive SW platform, covering non-differentiating elements. The middleware is using industry standard such as Linux, AUTOSAR Classic and Adaptive. In addition, with “Edge Enabler” the middleware is seamlessly and transparently connected to the cloud. A cloud based DevOps Workbench that offers seamlessly integrated toolchain covering the complete development cycle is also available. This enables an integrated full-stack solution from the cloud to the sensor.

The ”Continental Automotive Edge Framework“ is designed to be flexible, scalable, and ready for future needs. We are continuously going to evolve the system. With this approach we enable OEMs, and partners to start development of innovative functions and systems for future architectures now. Development kits allow the development of software applications and system functions independent of the availability of the target hardware.
Summary

By applying the concepts of the Software Defined Vehicle both the transformation of the vehicle architecture and the growth of software can be turned from a challenge into an opportunity.

**Vehicle Architecture Transformation is happening... Now!**

- New vehicle architectures enable the Software Defined Vehicle
- The Software Defined Vehicle establishes new business and collaboration models as well as new value streams within the Mobility Industry
- The Software Defined Vehicle can support the management of complexity with shorter cycles for development, integration, and testing
- Data driven development enabling continuous improvement and enhancement of functions
- Virtual Development and Development Kits allow software and system development before the availability of target hardware. The goal for virtual development is to enable software and systems development happening in a virtual environment (“Environmental Parity”)
Consequences

- Complexity and functional growth for software is continuing. CASE will further increase the demand for computing power for mobility applications. The Software Defined Vehicle provides means to manage complexity, but it does not reduce the inherent complexity of vehicle software.

- The cloud is an essential part of the Software Defined Vehicle. Platforms and tools should be inherently cloud enabled. Companies need to embrace cloud technology and apply the concepts for the future system architectures and vehicle related software and tooling.

- Platform approaches for the Software Defined Vehicle are helpful to optimize re-use, time-to-market and costs. At Continental with “CAEdge” we are continuously evolving a flexible, modular and scalable architecture for the Software Defined Vehicle.

- Cross industry collaboration avoids fragmentation of software architectures and increases maturity and quality of software building blocks across all levels of the stack. This allows us to focus on innovative and differentiating function. Reuse of software depends on proven and stable interfaces between building blocks.

- Let’s apply best practices from the software industry on how to define and manage interfaces as basis for re-use and continuous improvement. The software and IT industry has applied this strategy successfully for both proprietary and open-source software.


[5] Deloitte “Software is transforming the automotive world”, 2020