

feel evolution

Pune, 05th Oct 2023

Akshay Bujone, Sr. Solution Architect
FEV.io India

Prepared for
MathWorks AUTOMOTIVE CONFERENCE 2023
India

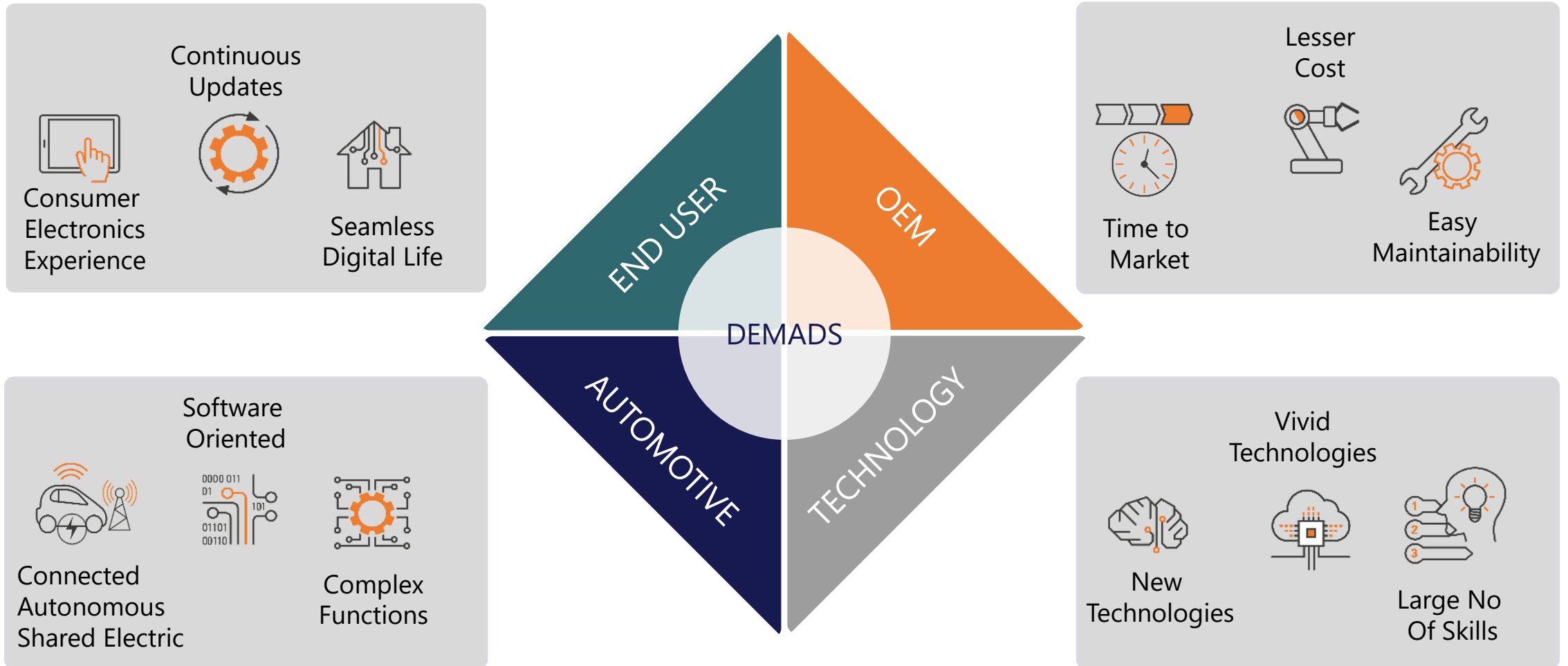


**Developing Vehicle Features for SDV
with Cross-Domain Computing**

Next-Gen E/E Architecture

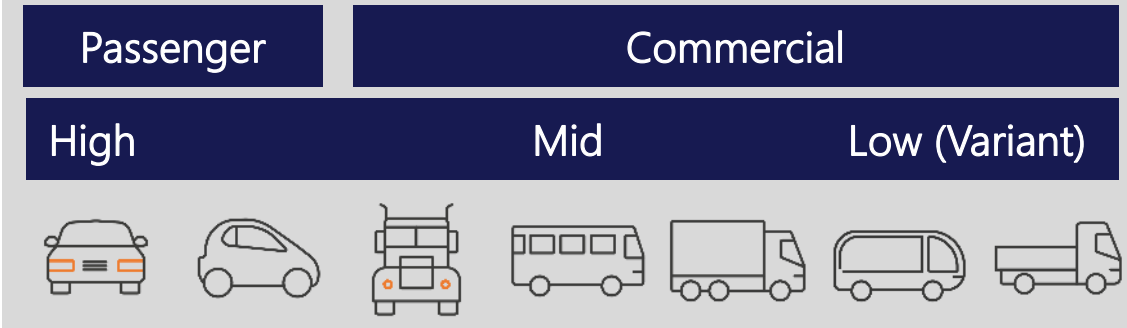
Automotive industry is currently driven by CAUSE

Demands are being raised in 4 dimensions

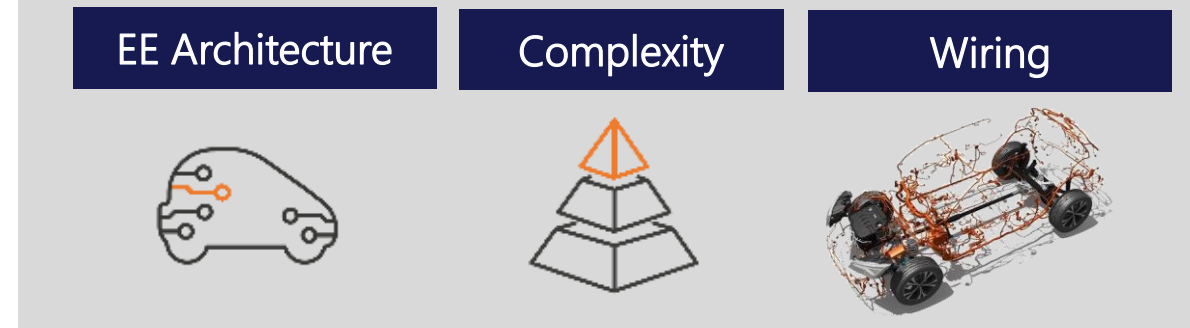


While addressing 4-dimensional demands solution must include important growing challenges within Automotive industry

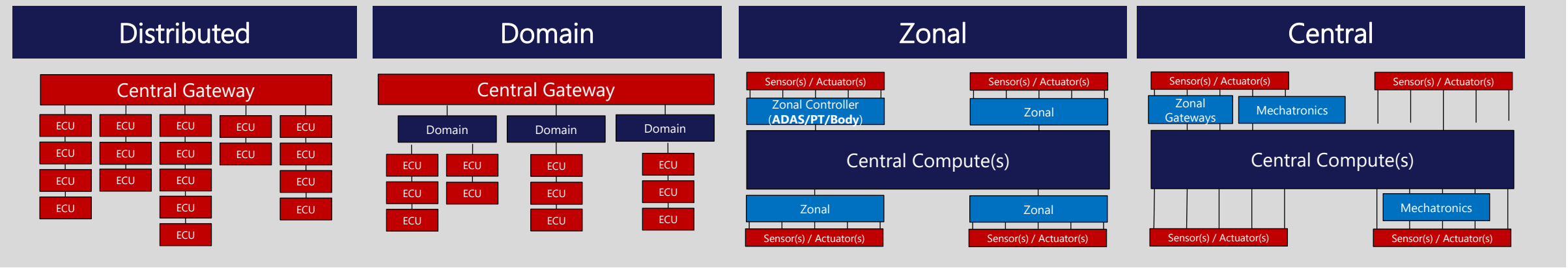
» WIDE RANGE OF MOBILITY



» GROWING INFLUENCERS



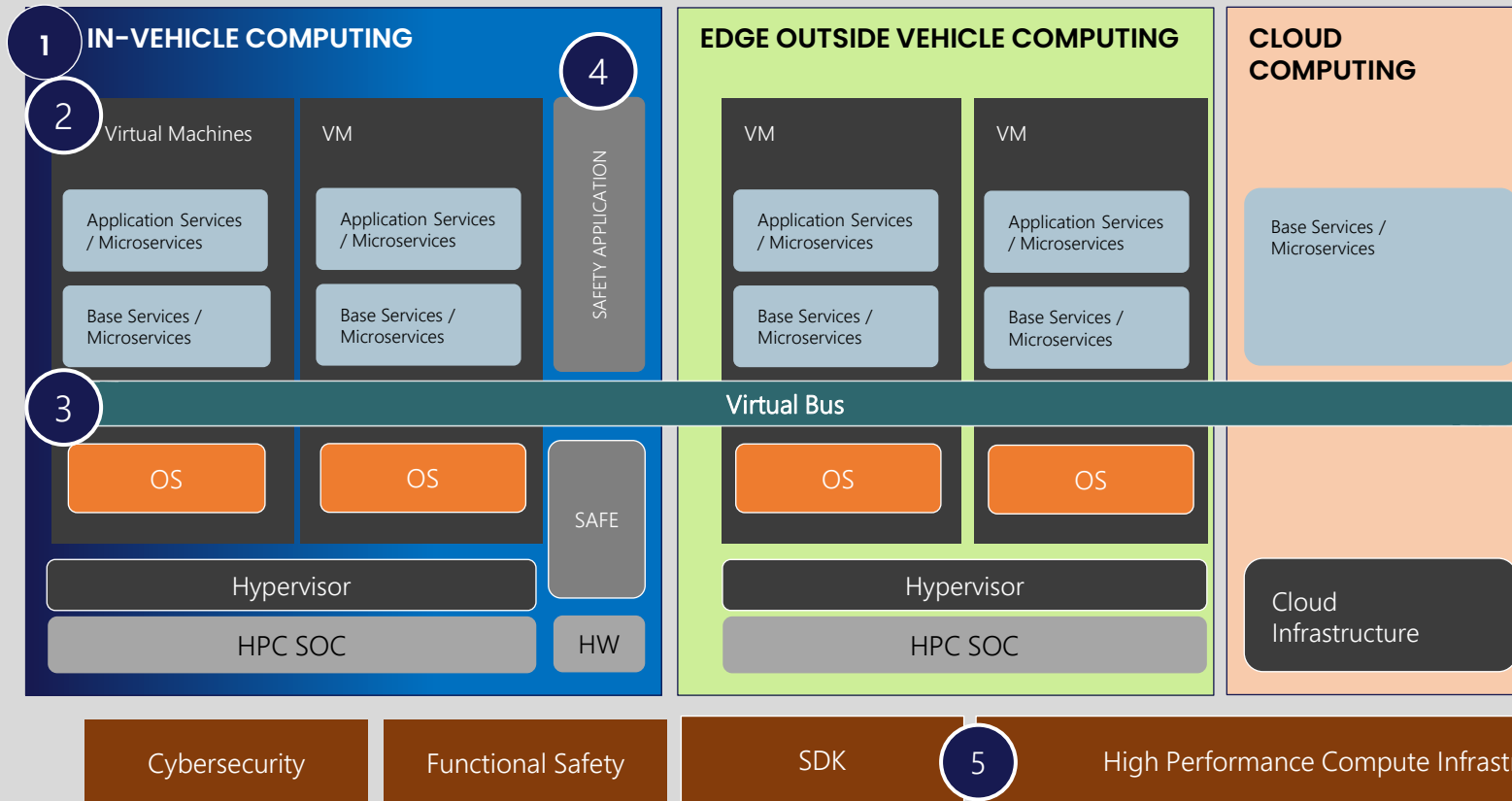
» EVOLVING EE ARCHITECTURE



One SOLUTION for all multi-dimensional requirements » SOFTWARE DEFINED VEHICLE

Realizing SDV Architecture for future automotive requirements

COMMON CONSIDERATIONS FOR SOFTWARE DEFINED VEHICLE



1. Various Computing options
2. Cloud native Platform
3. Virtual Bus (Signal & Service base) & extensible to Edge and Cloud computing
4. Platform & Product level Safety & Security
5. Additional infrastructural support for OTA, HPC management and development SDK

FEV Software Defined Vehicle Architecture and Development Methodology extends to Edge and Cloud as a scalable platform



IN-CAR SERVICES

**BUSINESS SERVICES
DIGITAL TWIN**

**VEHICLE TO VEHICLE
VEHICLE TO INFRA**

SDK

Cyber Security

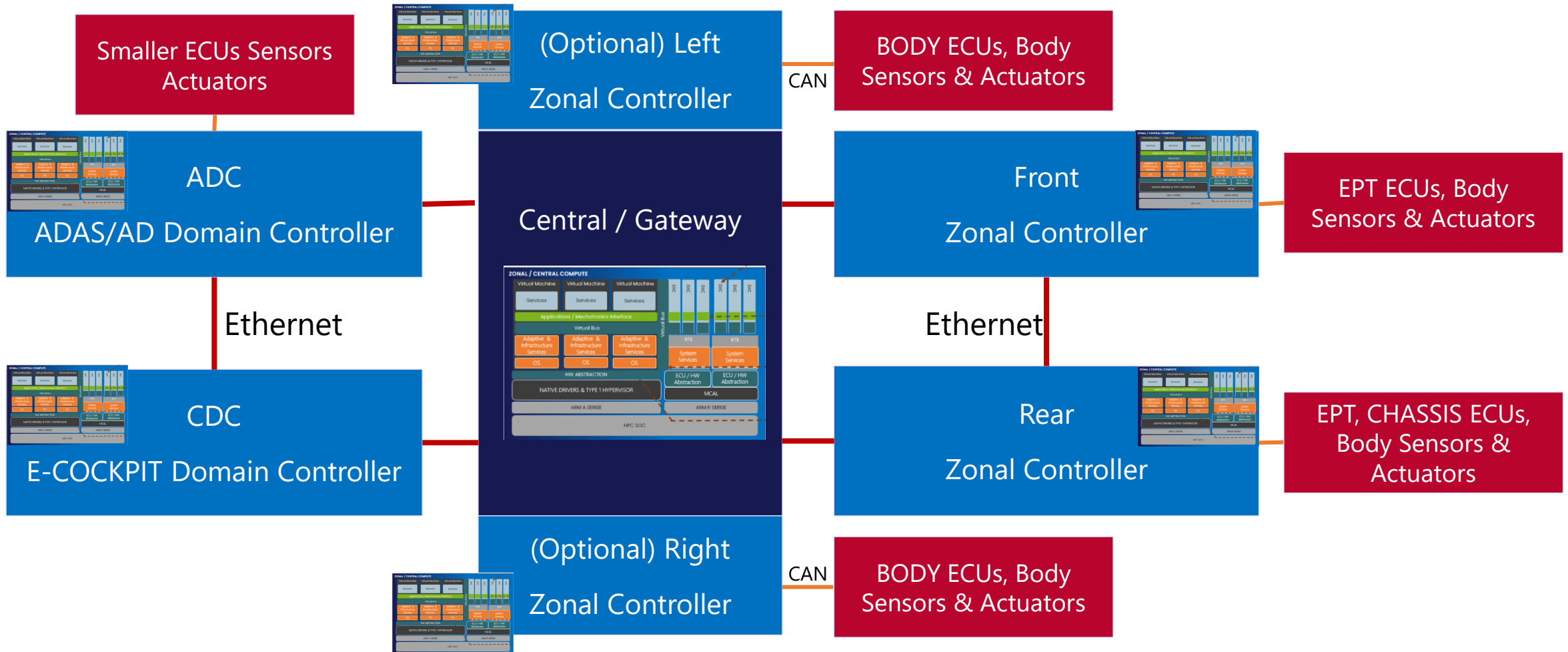
Functional Safety

HPC INFRA MGMT

OTA

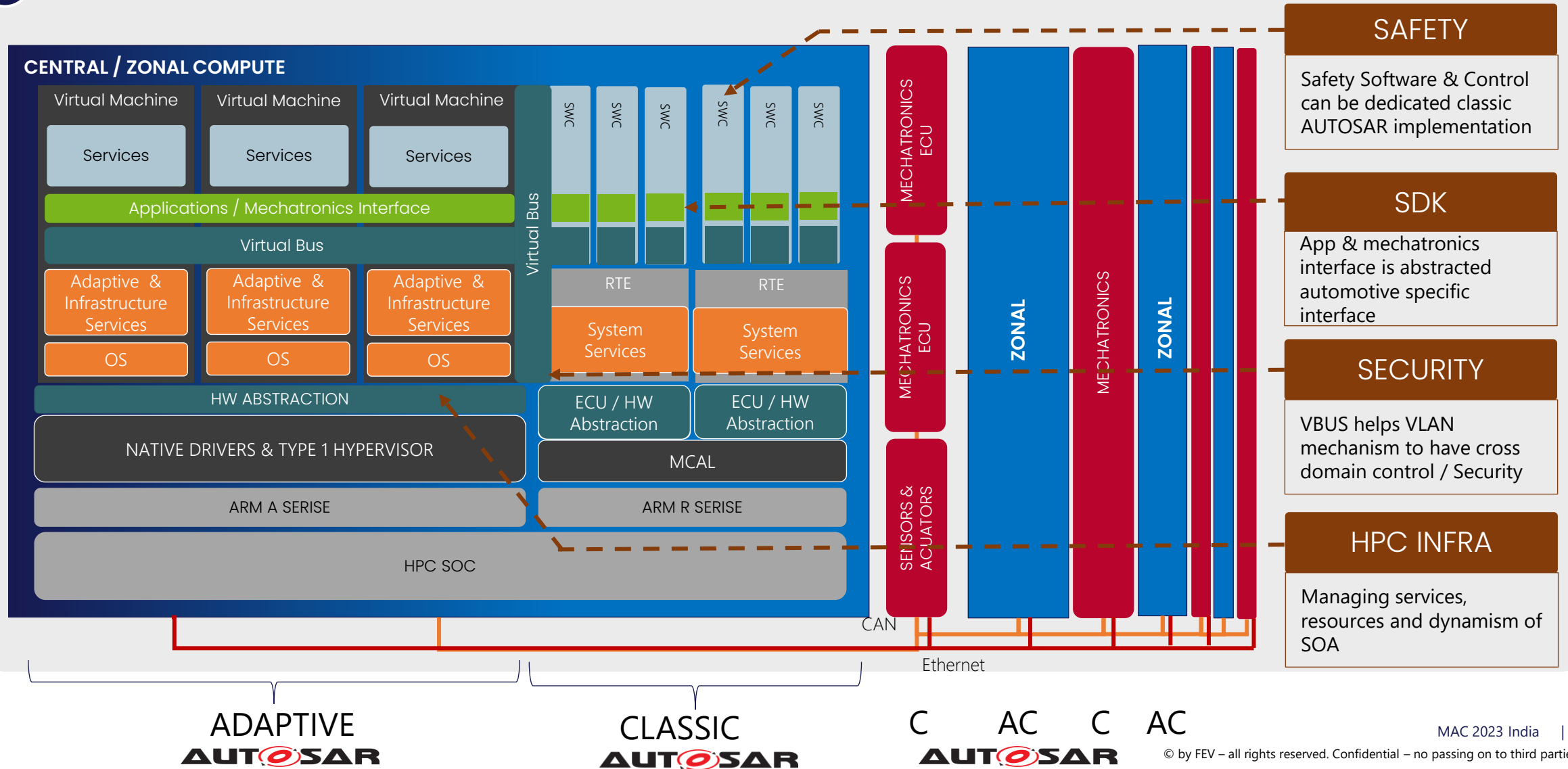
MOVING TO CENTRAL & ZONAL ARCHITECTURE

Typical EE Architecture by various OEMs for 2024 to 2026



System & Software Architecture to be design as reusable services

UNIFIED SYSTEM & SOFTWARE ARCHITECTURE



Hierarchy levels in modern E/E architecture

How functions can be distributed across computational options

	Function	Computation	Communication	Updating / Change
Cloud	complex / fleet-oriented functions	fully scalable, cloud-native	Internet Protocols	frequent and continuous update
Central Compute*	in-vehicle centralized high-level functions	limited scalability. high performance	Internet Protocols	high update / change
Zone Controller	spatially boxed multi-domain functions, I/O concentration	Restricted/medium performance	Internet Protocols, classic automotive (CAN (FD), LIN, ...)	low update / change
Local Controllers	specific & local single task control	restricted performance	classic automotive	No/low update / change
Sensors & Actuators	highly specific	low-level	classic automotive	no update / change

*) not necessarily means one single computing instance ("vehicle computer")

Next Generation In-Vehicle EE Architecture combines Central, Zonal, Domain and Legacy Controls

- Features/functions with high demand for
 - Computing power, on-chip accelerators
 - Updateability / Upgradeability
 - Data exchange
- Virtualized cross-domain Application SW independent from HW
 - Mixed criticality: safety and non-safety functions
 - Service-Oriented Architecture

- Dedicated domain controller with specific requirements (e.g. AD, VMC)
- Local cross-domain features/functions on zonal controller
- Simplified / reduced wiring harness
- Transfer zone between service- and signal-orientation

- Legacy sub-systems & ECU (cost, availability, eco-system)
- Smart mechatronics with high demand for
 - Safety
 - Hard real-time, start-up time
- Signal-based

> 50ms



10 ~ 50ms



< 10ms



Ethernet Backbone

Ethernet
CAN
LIN

SOFTWARE DEFINED VEHICLE FEV DEMONSTRATION

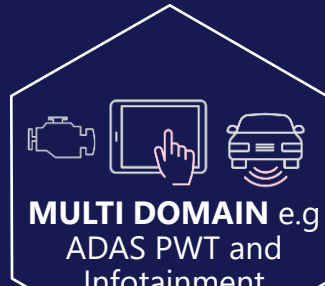
FEV runs a technology initiative for core platform development meeting the SDV targets

KEY VALUES

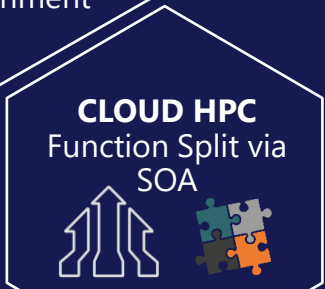
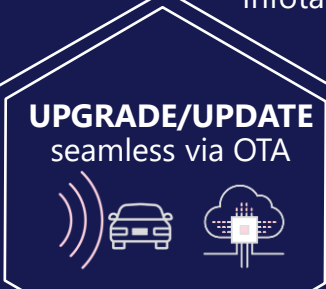
CORE TECHNOLOGIES

PROJECT TARGETS

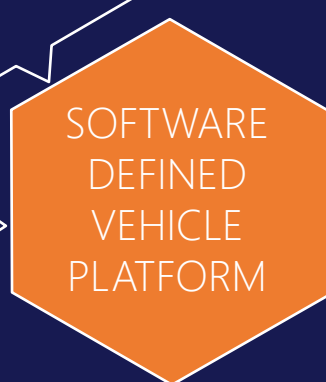
E/E Consolidation



Feature Increase



Automotive Fit



Manage System Complexity
Via consistent SE for SW & EE design



Increase Software excellence
Usage of Cloud native design principles, app abstraction, virtualization and new SW frameworks to run cross domain applications
(Re)- Engineer of (legacy) features from different domains into SOA architecture



Create E2E understanding of the dataflow
Buildup of cloud backend, middleware (on-/offboard) and virtual bus



Extend skills in automotive grade development
Analysis and Implementation of safety & security mechanism

SDV Demonstrator



Scalable, modular demonstrator to test, improve and showcase the project objectives



Demonstrate mix criticality load on HPC

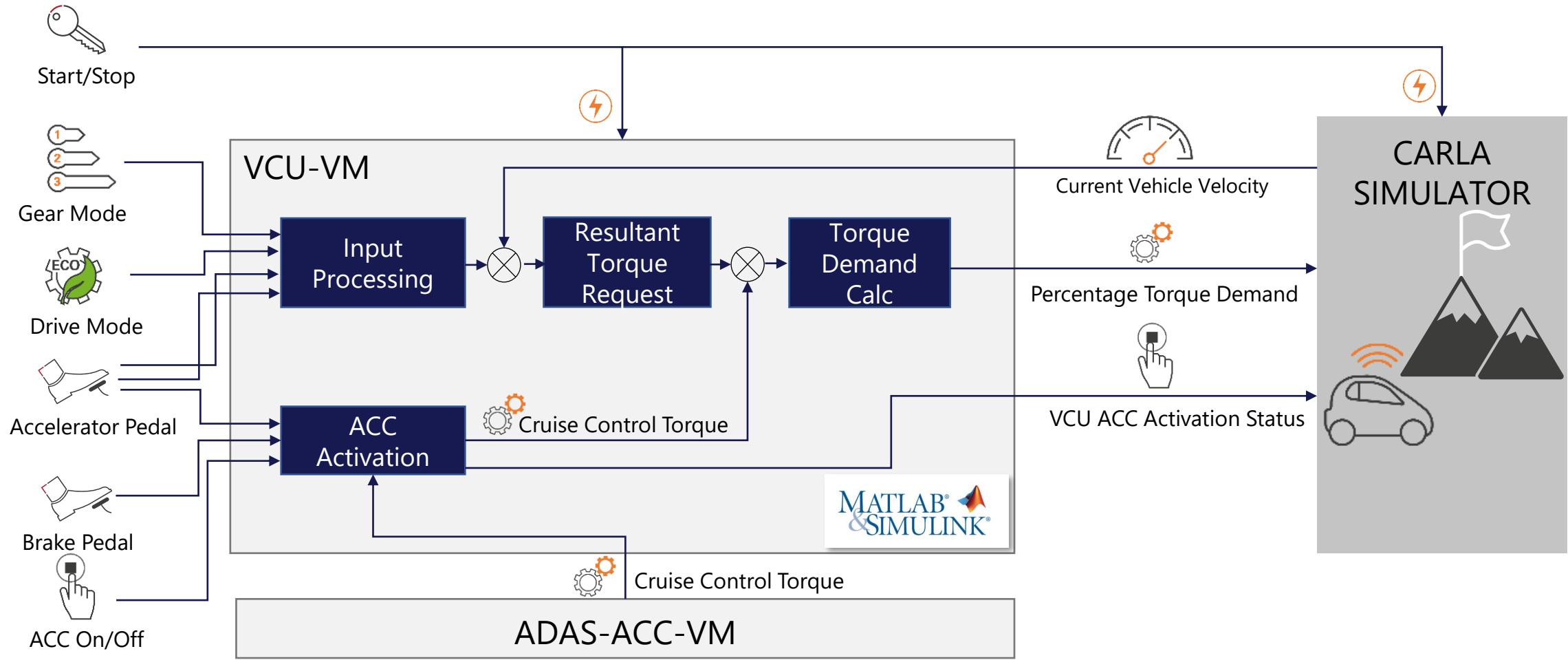


Redesign legacy functions into SOA Architecture



Create virtual bus for maximal abstraction of SW

Migration of Legacy Function to SoA: *E-Drive Torque Path Arbitration*



Integration of vehicle function in VCU Cluster on HPC with Autosar Adaptive for SOA architectural design



Legacy to SOA architecture approach:

SYSTEM DESIGN

- Use Cases elaboration
- Features definition
- Requirements derivation
- System design

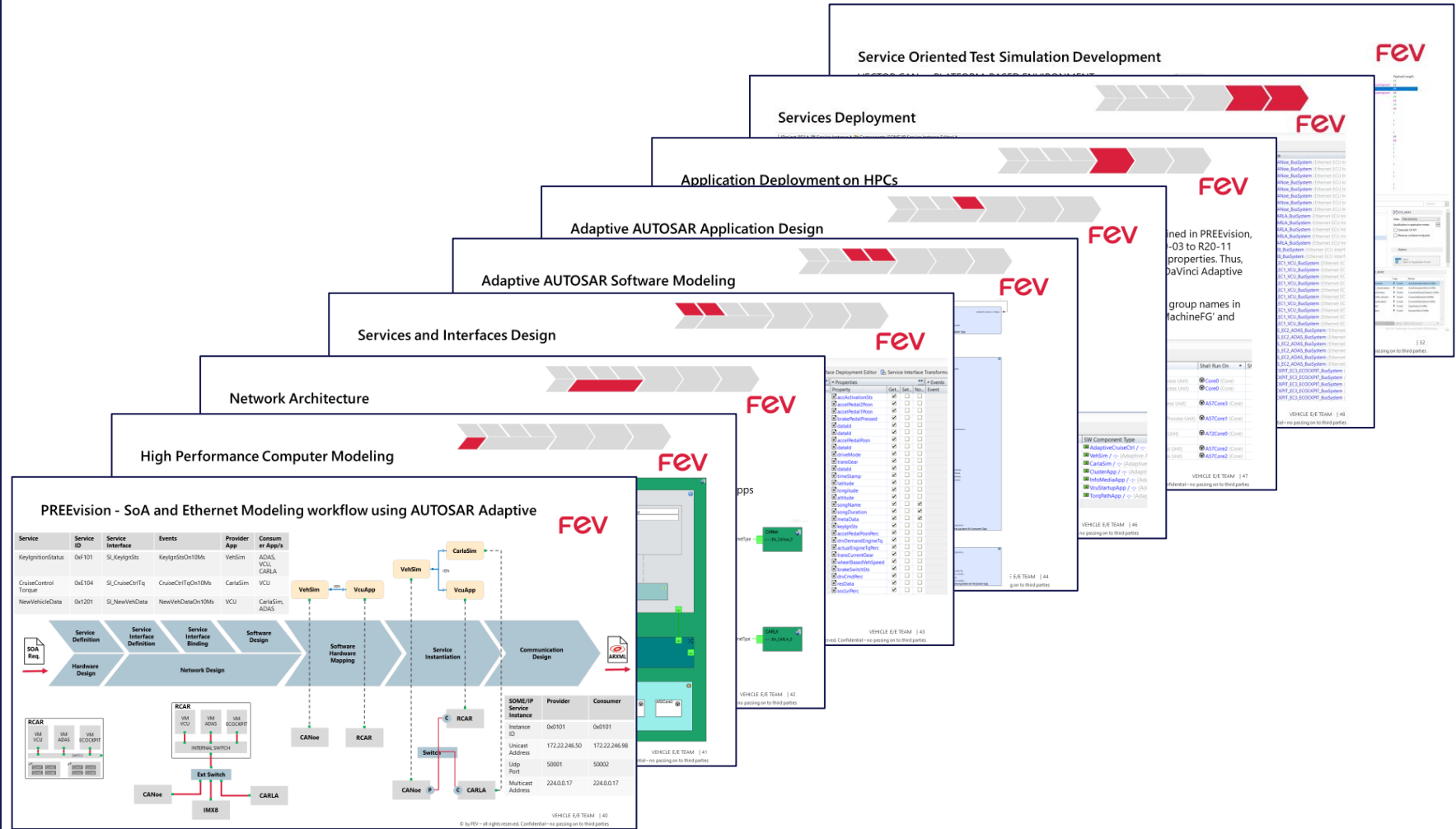
SOFTWARE DESIGN

- Service Architecture Definition
- Application Design in MATLAB/Simulink
- Signal Modeling including timing req. analysis
- Deployment configuration of Autosar Adaptive stack

INTEGRATION:

- Application implementation
- Integration with Vector Adaptive Stack
- Deployment of QNX OS and Hypervisor
- Set up of HIL Demo

Continuous improvement along the entire workflow via consequent monitoring of best practices and lessons learned



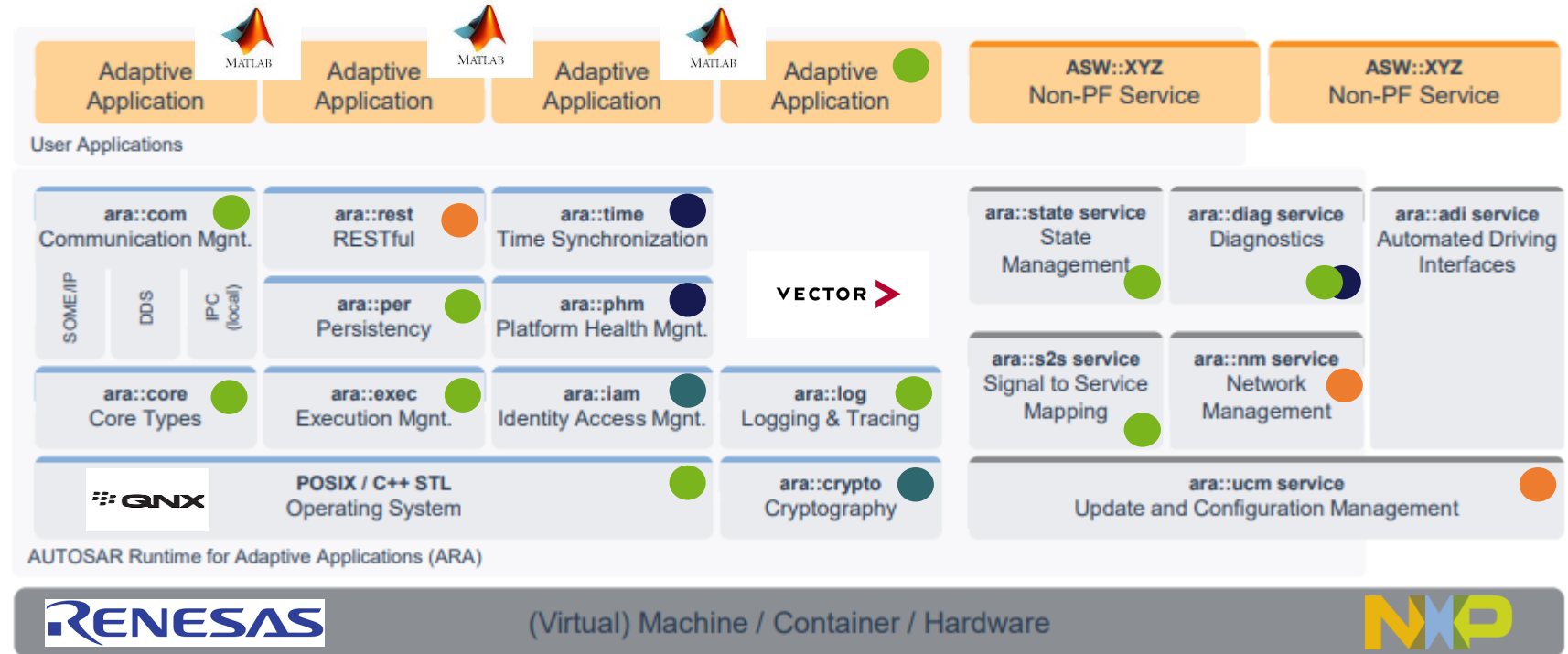
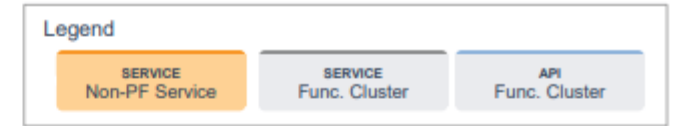
Adaptive AUTOSAR: FEV Partnership Landscape

BASE PLATFORM

SCALABLE APPROACH



AUTOSAR Adaptive Platform Logical view



EXPLOITATION ROADMAP

	2023	2024
● Base Platform	High	High
● Functional Safety	Mid	High
● Cyber Security	Low	Mid
● Over The Air Update	Mid	High

✓ SOVD (Service Oriented Vehicle Diagnostics) -> Currently creating production specification

Realizing the Legacy Functions as Adaptive SWC in Simulink

1

2

The image shows two screenshots of the Simulink environment. The first screenshot (1) shows the Simulink interface with the TorqPathApp model open. The second screenshot (2) shows the AUTOSAR Component Designer tool, which is used to generate AUTOSAR components from the Simulink model. The tool's interface includes a toolbar with options like 'Quick Start', 'C/C++ Code Advisor', and 'Code Interface'. The main workspace displays the AUTOSAR dictionary for TorqPathApp, which lists various service interfaces and data types. The 'XML Options' section on the right allows for configuring the generated XML files, including options for 'XML Options Source', 'Exported XML File Packaging', and 'Additional Packages'.

Adaptive AUTOSAR compliant C++ Code Generation in Simulink

3

PathApp - Simulink

AUTOSAR Adaptive

Code for: TorqPathApp

Generate Code

View Code

Remove Highlighting

Share

SELECT OUTPUT

AUTOSAR

AUTOSAR-compliant code (autosar.tlc)

AUTOSAR Adaptive

AUTOSAR-compliant C++ code (autosar_adaptive.tlc)

SIMULATION

Simulation Only

Model intended for simulation only.

Select System Target File...

Code Mappings - Component Interface

4

PathApp - Simulink

AUTOSAR Adaptive

Code for: TorqPathApp

Generate Code

View Code

Remove Highlighting

Share

Configuration Parameters: TorqPathApp/QuickStart_50023_1_31_20_48_37302 (Active)

Search

Solver

Data Import/Export

Math and Data Types

Diagnostics

Hardware Implementation

Model Referencing

Simulation Target

Code Generation

Optimization

Report

Comments

Identifiers

Custom Code

Interface

Code Style

Verification

Templates

Code Placement

Data Type Replacement

AUTOSAR Code Generat...

Coverage

Target selection

System target file: autosar_adaptive.tlc

Description: AUTOSAR Adaptive

Shared coder dictionary: <empty>

Language: C++

Language standard: C++11 (ISO)

Build process

Generate code only

Package code and artifacts

Toolchain: AUTOSAR Adaptive | CMake

Build configuration: Faster Builds

Code generation objectives

Prioritized objectives: Execution efficiency, RAM efficiency

Check model before generating code: Off

Advanced parameters

Built-in FFTW library callback

Code Mappings - Component Interface

THANK YOU

feel evolution