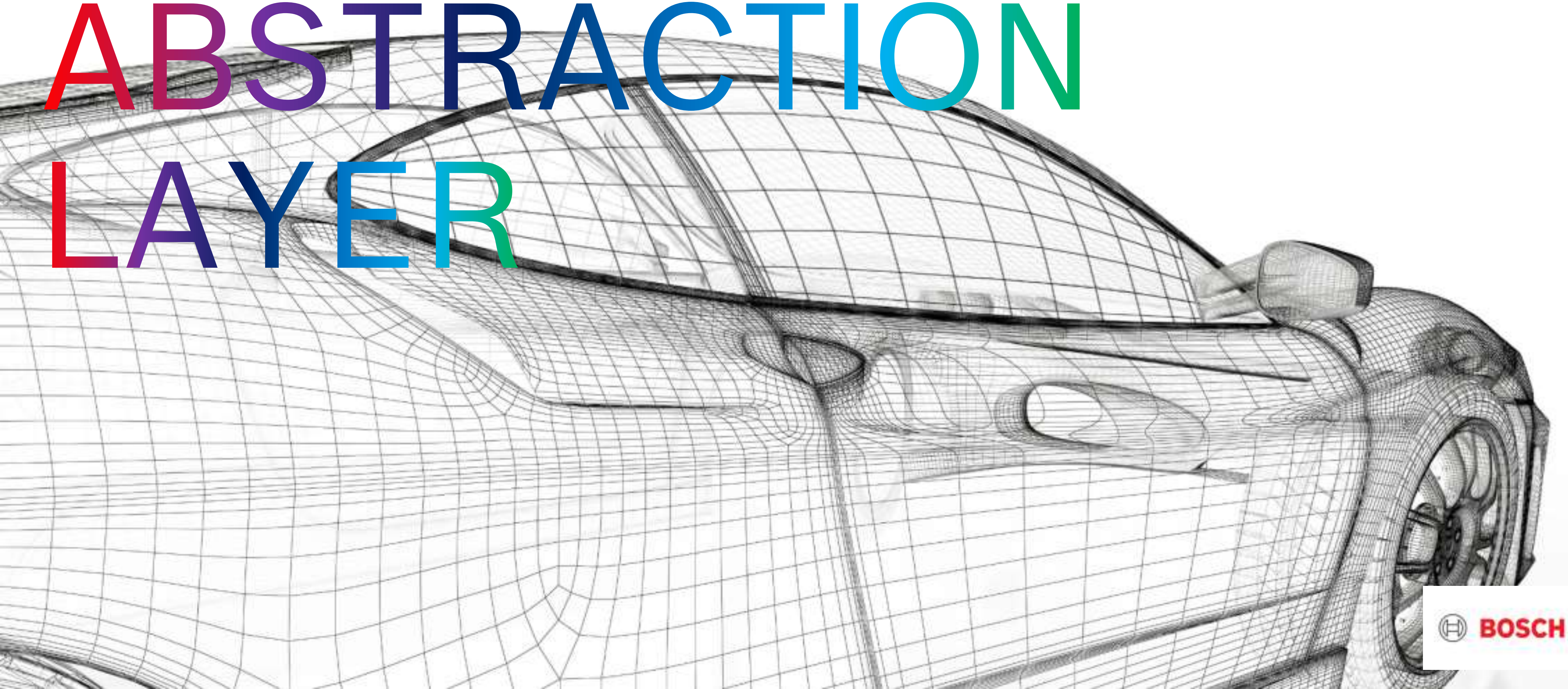
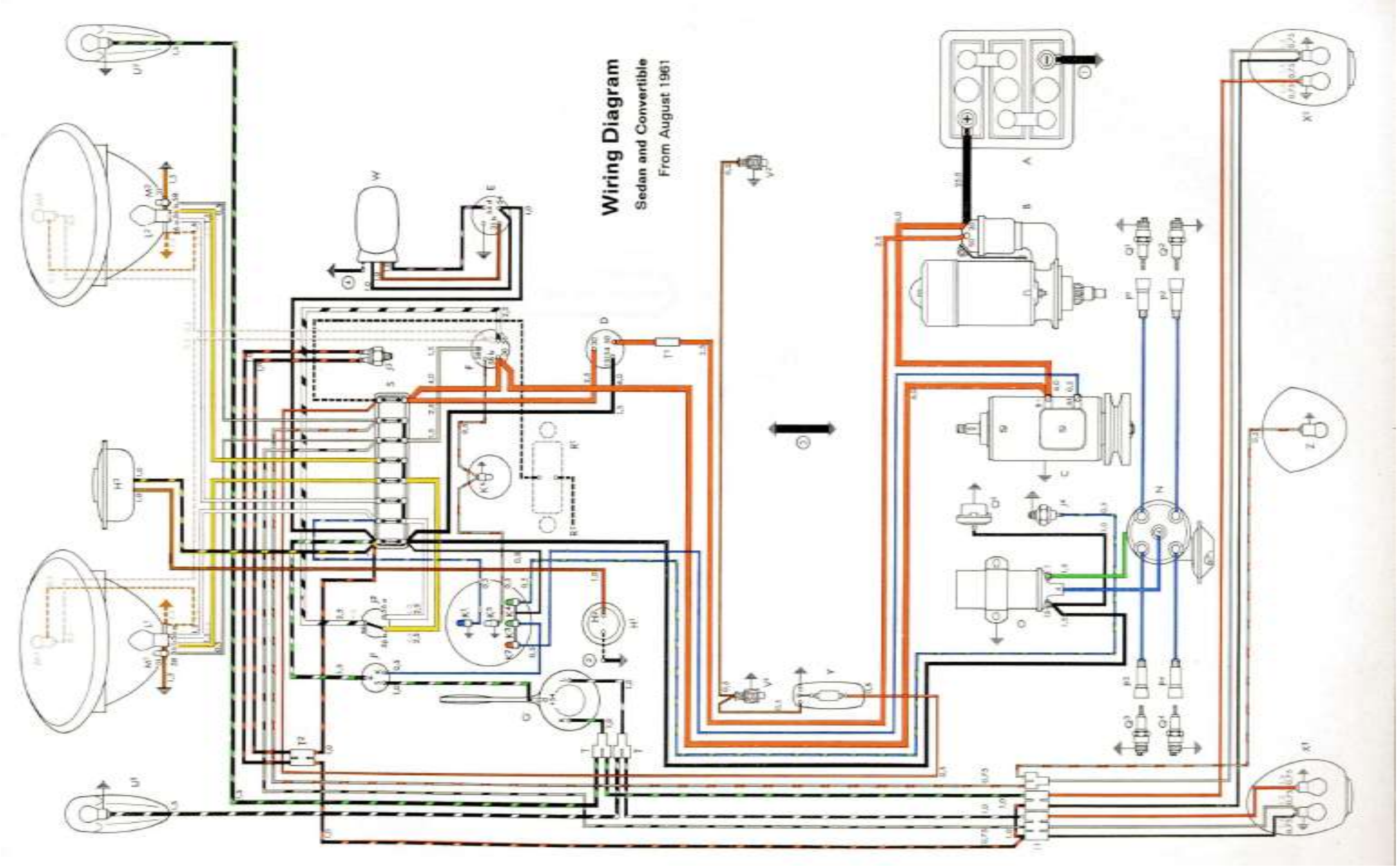
VEHICLE ABSTRACTION LAYER



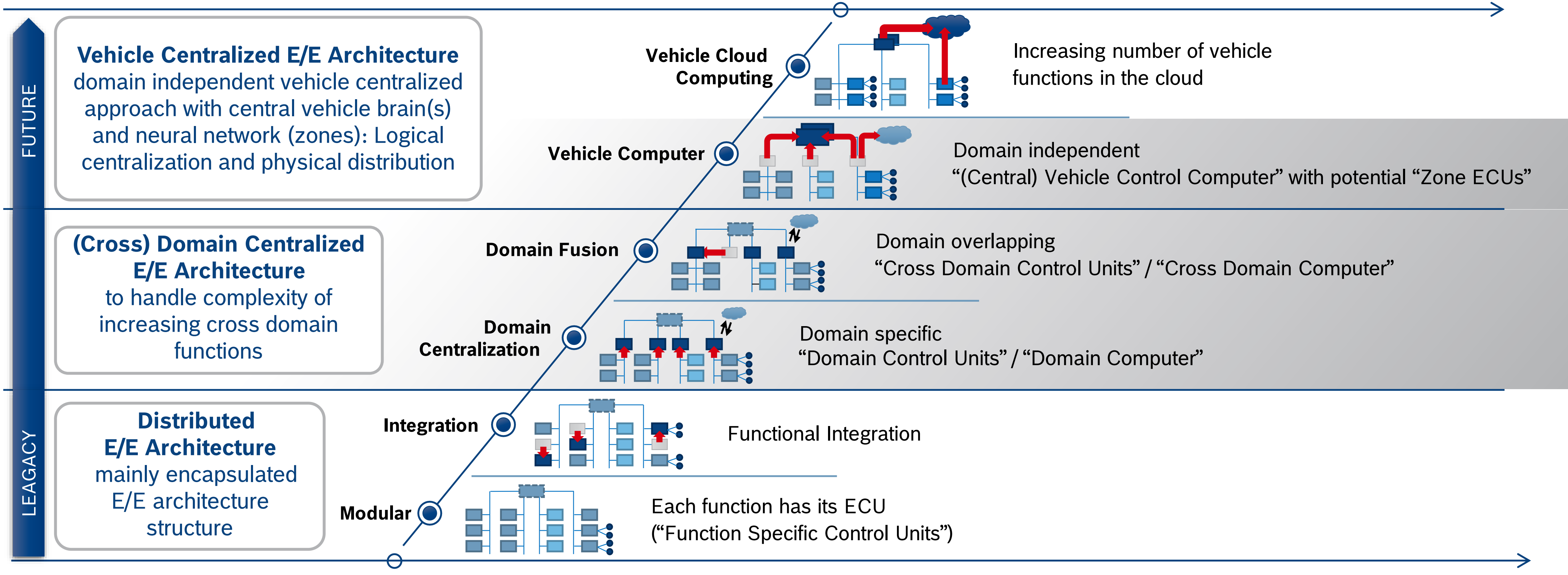
BOSCH

Vehicle Abstraction Layer
Automotive: 1962



Source: VW-Käferclub

Trends for Future Mobility Systems
E/E Architecture Roadmap



Vehicle Centralized E/E Architecture
domain independent vehicle centralized approach with central vehicle brain(s) and neural network (zones): Logical centralization and physical distribution

(Cross) Domain Centralized E/E Architecture
to handle complexity of increasing cross domain functions

Distributed E/E Architecture
mainly encapsulated E/E architecture structure

Increasing number of vehicle functions in the cloud

Domain independent
“(Central) Vehicle Control Computer” with potential “Zone ECUs”

Domain overlapping
“Cross Domain Control Units” / “Cross Domain Computer”

Domain specific
“Domain Control Units” / “Domain Computer”

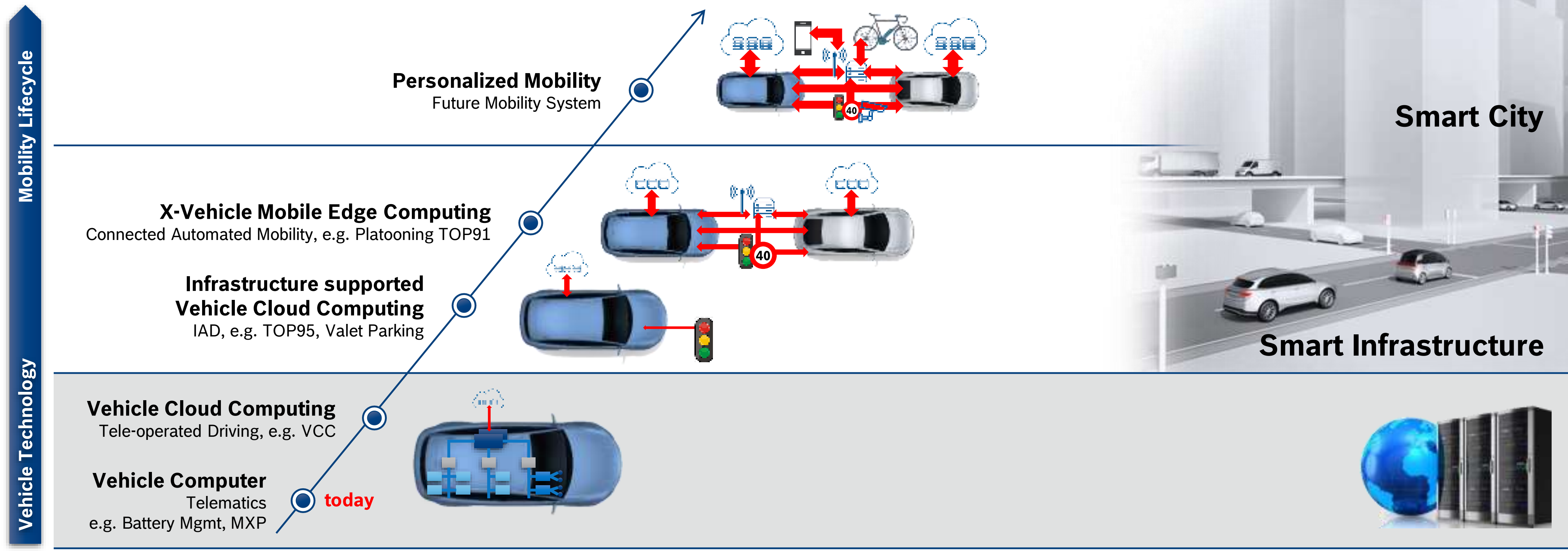
Functional Integration

Each function has its ECU
(“Function Specific Control Units”)

increasing SW amount

- ▭ typ. state of the art automotive ECUs (function specific)
- ■ Performance ECUs e.g. (Cross-)Domain Control Unit, (Cross-)Domain Computer, Vehicle Control Computer
- ▭ Optional ECUs (e.g. Central Gateway)
- ■ Domain independent Zone ECUs
- Sensors/Actuators
- ECU = Electronic Control Unit
- ▭ Domain specific Zone ECUs (e.g. todays Door ECU)

Trends for Future Mobility Systems
E/E Architecture Extension to Cloud Connectivity



Mobility Lifecycle

Vehicle Technology

Personalized Mobility
Future Mobility System

X-Vehicle Mobile Edge Computing
Connected Automated Mobility, e.g. Platooning TOP91

Infrastructure supported Vehicle Cloud Computing
IAD, e.g. TOP95, Valet Parking

Vehicle Cloud Computing
Tele-operated Driving, e.g. VCC

Vehicle Computer
Telematics
e.g. Battery Mgmt, MXP

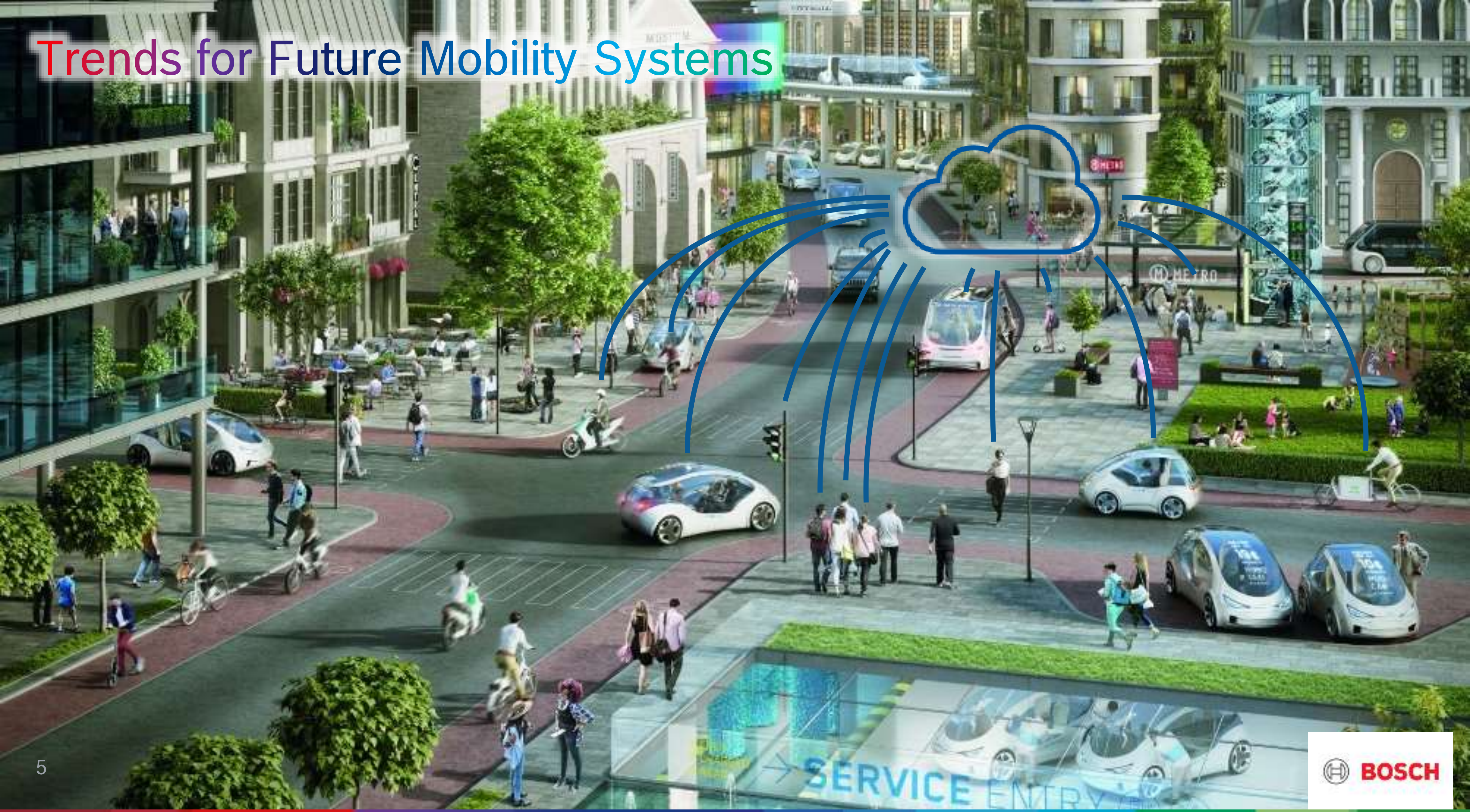
today

Smart City

Smart Infrastructure

▭	State of the art ECU	▭	Vehicle Computer	☁	Cloud	📡	Edge
▭	Zone ECUs w/ sensors/actuators	▭	(Cross-)Domain ECU	🚦	Infrastructure	📱	Personal µMobility

Trends for Future Mobility Systems



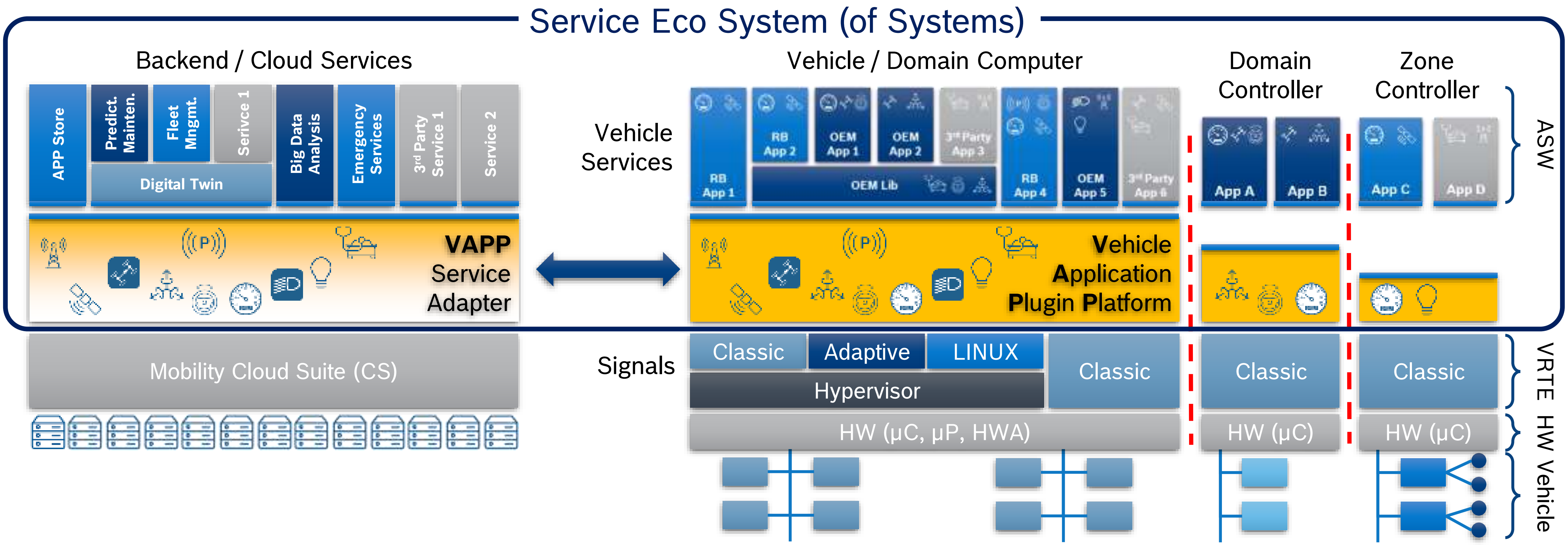
BOSCH

Vehicle Abstraction Layer

Point of View Cloud

00101
11001
WWW
CO2
AUTOMATED SHUTTLE

Vehicle Abstraction Layer
Vehicle Application Architecture

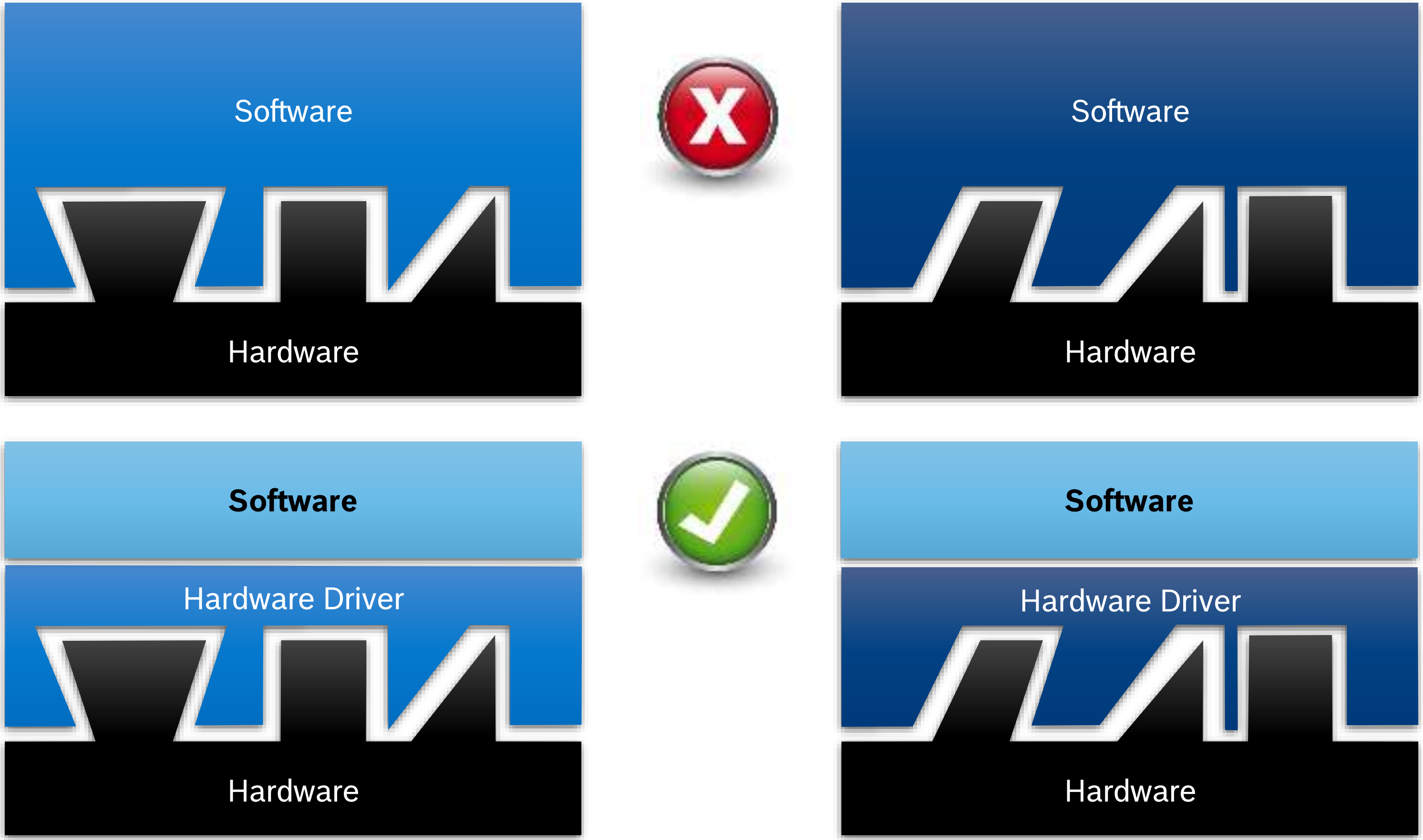


Digital Twin as virtual representation of the Vehicle
Vehicle Application and Service Interfaces are evolving as trend in automotive service area

Vehicle Abstraction Layer
Abstraction and Freedom of Interference – ECU / Hardware

Introduction of drivers allowed independent development of hardware and software

- ▶ Reduction of dependencies and complexity
- ▶ Reduction of porting effort to different hardware
- ▶ Separation of driver and software development

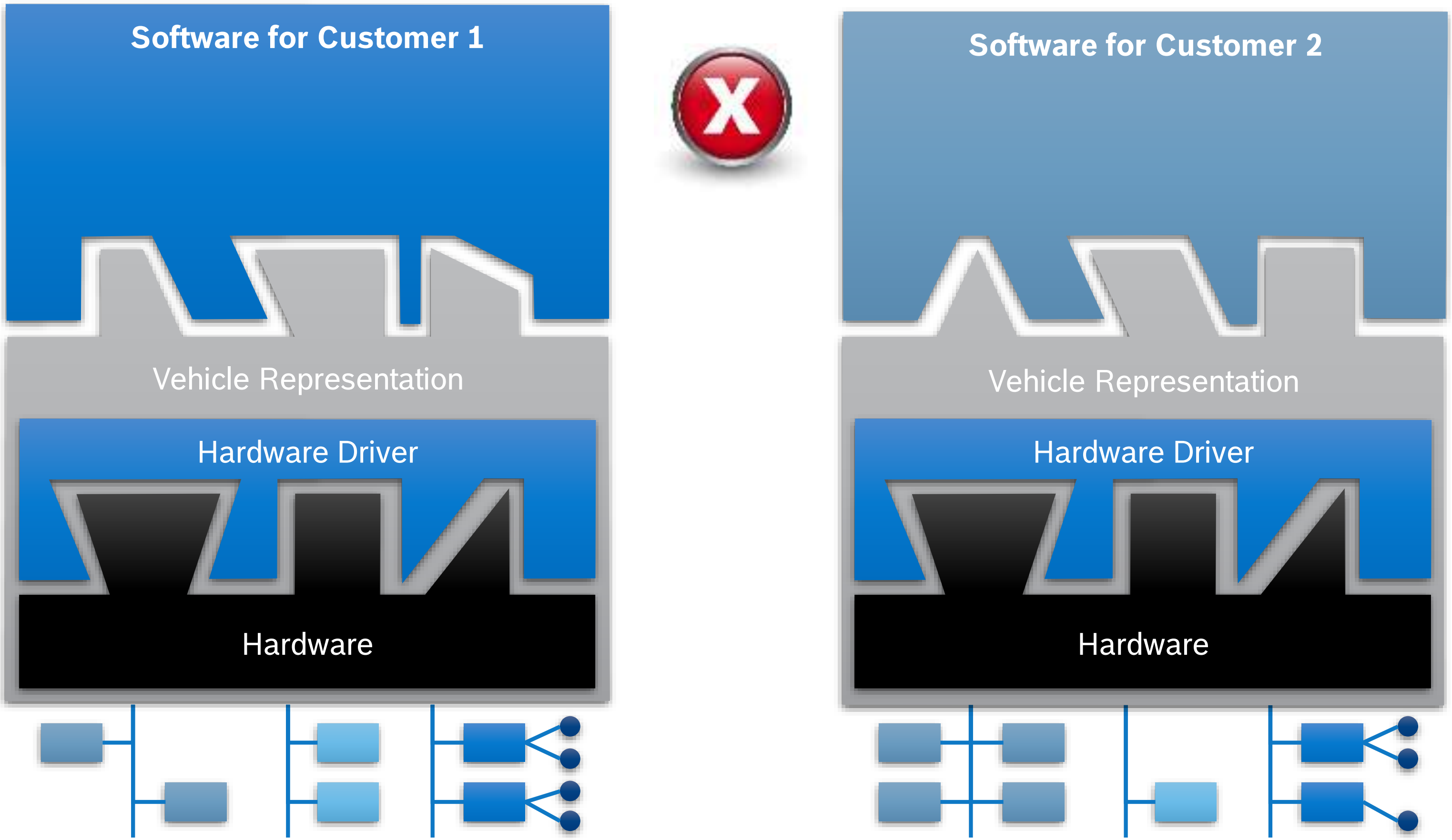


Source: based on AUTOSAR Guided Tour

Vehicle Abstraction Layer
Abstraction and Freedom of Interference – E/E Architecture

Each Embedded System is reflected on implementation level due to communication, resources and specific component selection

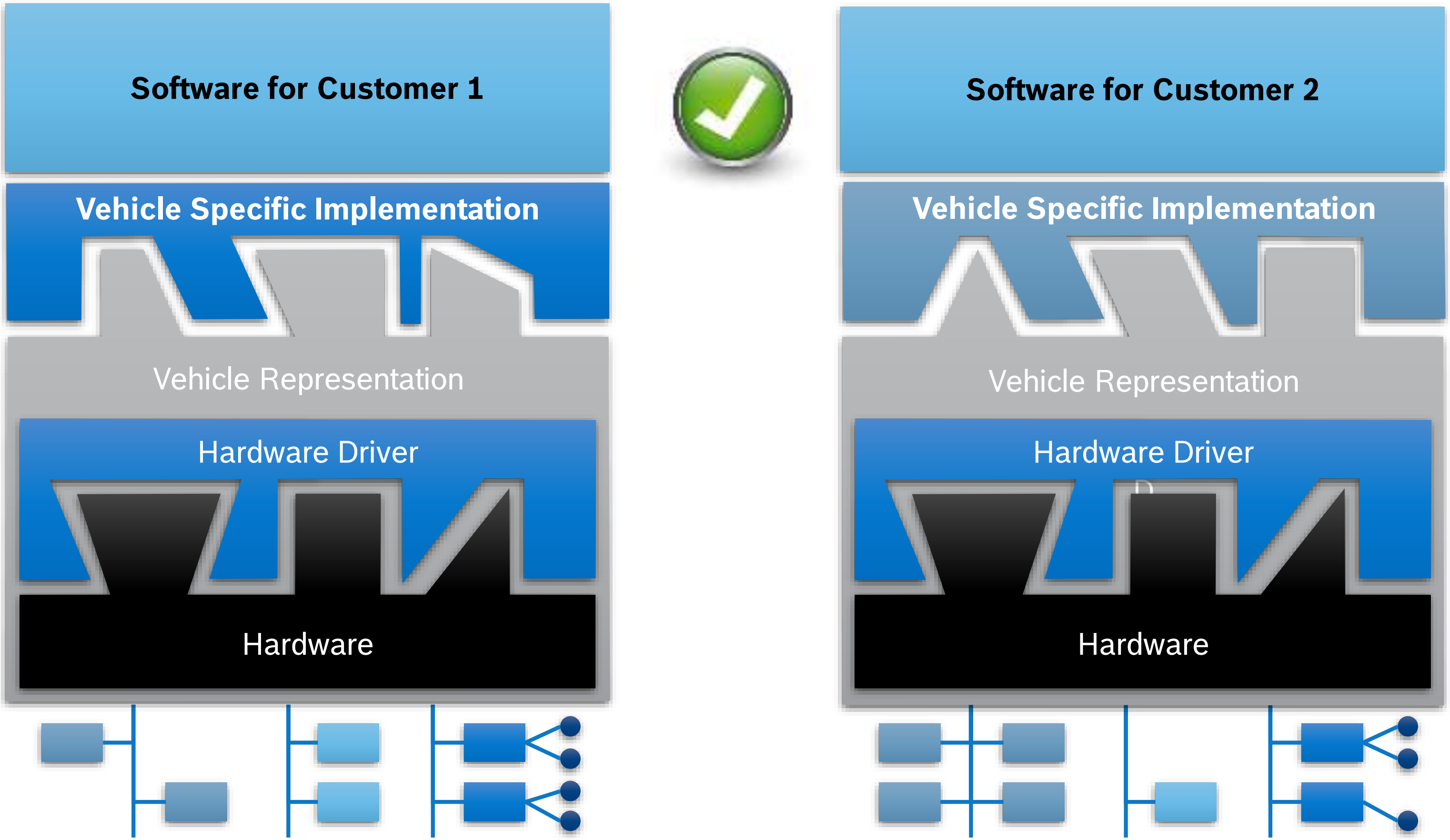
- Porting software from a device depending on one E/E architecture to another requires high adaptation efforts



Vehicle Abstraction Layer
Abstraction and Freedom of Interference – E/E Architecture

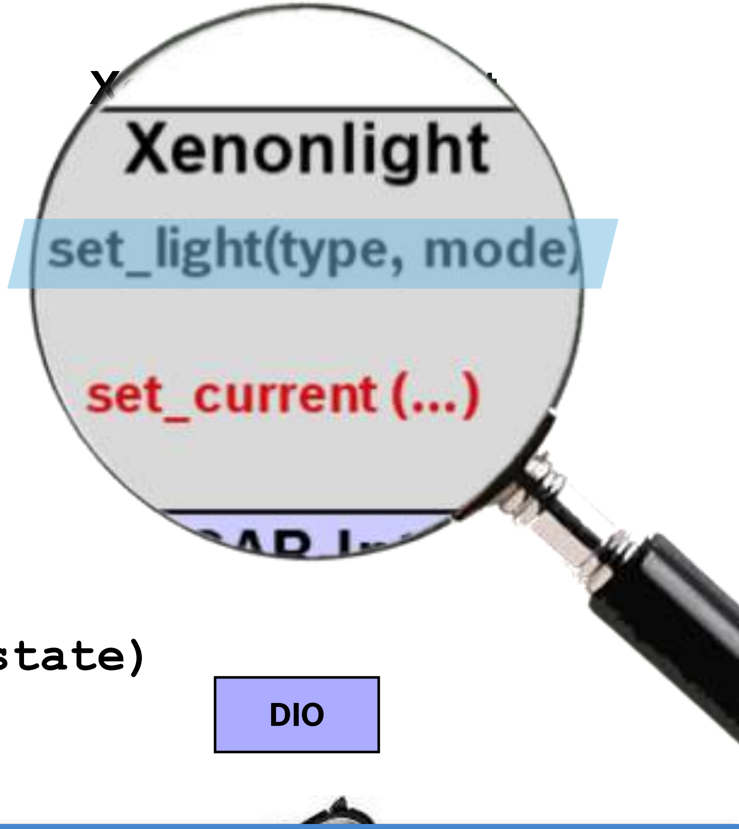
A vehicle specific software layer allows independent development of E/E architecture and software

- ▶ Reduction of dependencies and complexity
- ▶ Reduction of porting effort in case of integration into new E/E architecture
- ▶ Separation of vehicle dependent and independent software development



Vehicle Abstraction Layer
Example: AUTOSAR: Exchange type of Front Light

```
setLight(enum state)
Set_Light(bool state)
switchHeadLight(enum type, enum mode)
lightOn()
setLight(bool state)
SetLight(bool state)
Set_Beam(enum range)
OP_MOD_Light_Func2(enum param1)
g_DrvReqHB(enum state)
lightSwitchEvent(enum state)
```

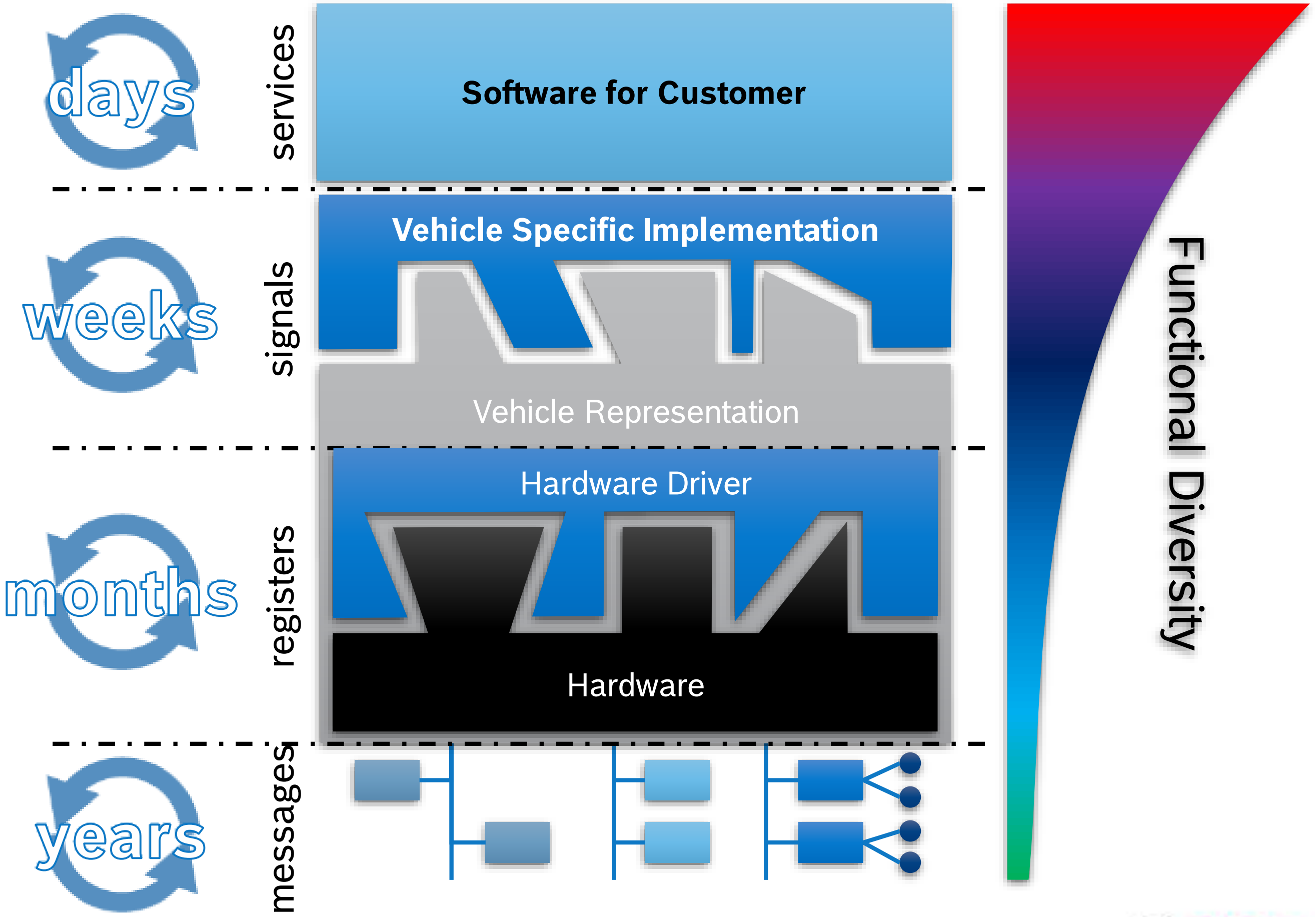


Remaining Challenge: NO Standardized Application Interface over OEMs / Project Borders

Source: based on AUTOSAR Guided Tour

Vehicle Abstraction Layer
Decoupling of Development & Deployment Cycles

- Decoupling of implementation reduces effort and complexity
- Decoupling of deployment cycles allows fast updates for high level features and well-proven processes for embedded functionality
- Service development does not require knowledge of all future functionality
- New business models possible due to independent deployment



Vehicle Abstraction Layer
“Double Shift Left” utilizing Vehicle APIs and Service IFs

Simulation and Shared Models

- Simulated Vehicle Services for early agile software development
- Increased coverage
- Accelerated test cycles
- Enable early discovery of functional gaps
- Improved cost, time to market, quality

SW Development before HW

- SiL - test bench enables regression and high coverage even before hardware is available

Source: based on Synopsys presentation 2019

Vehicle Abstraction Layer Conclusion

We need

- Standardized interfaces
- to easily develop functionalities for all kind of vehicles
- which can be distributed faster in a flexible way
- within the vehicle or in the digital twin

Let‘s define this together