MathWorks
AUTOMOTIVE
CONFERENCE 2024
India

Driving the Future - Integrating ADAS in Software-Defined Vehicles through Model-Based Design

Vamshi Kumbham, MathWorks

MathWorks

Automotive Industry Transformation:

Our Panel of Industry experts:

Nukul Sehgal

Application Engineering Team,
Software-Defined Vehicles,
Virtualization & DevOps

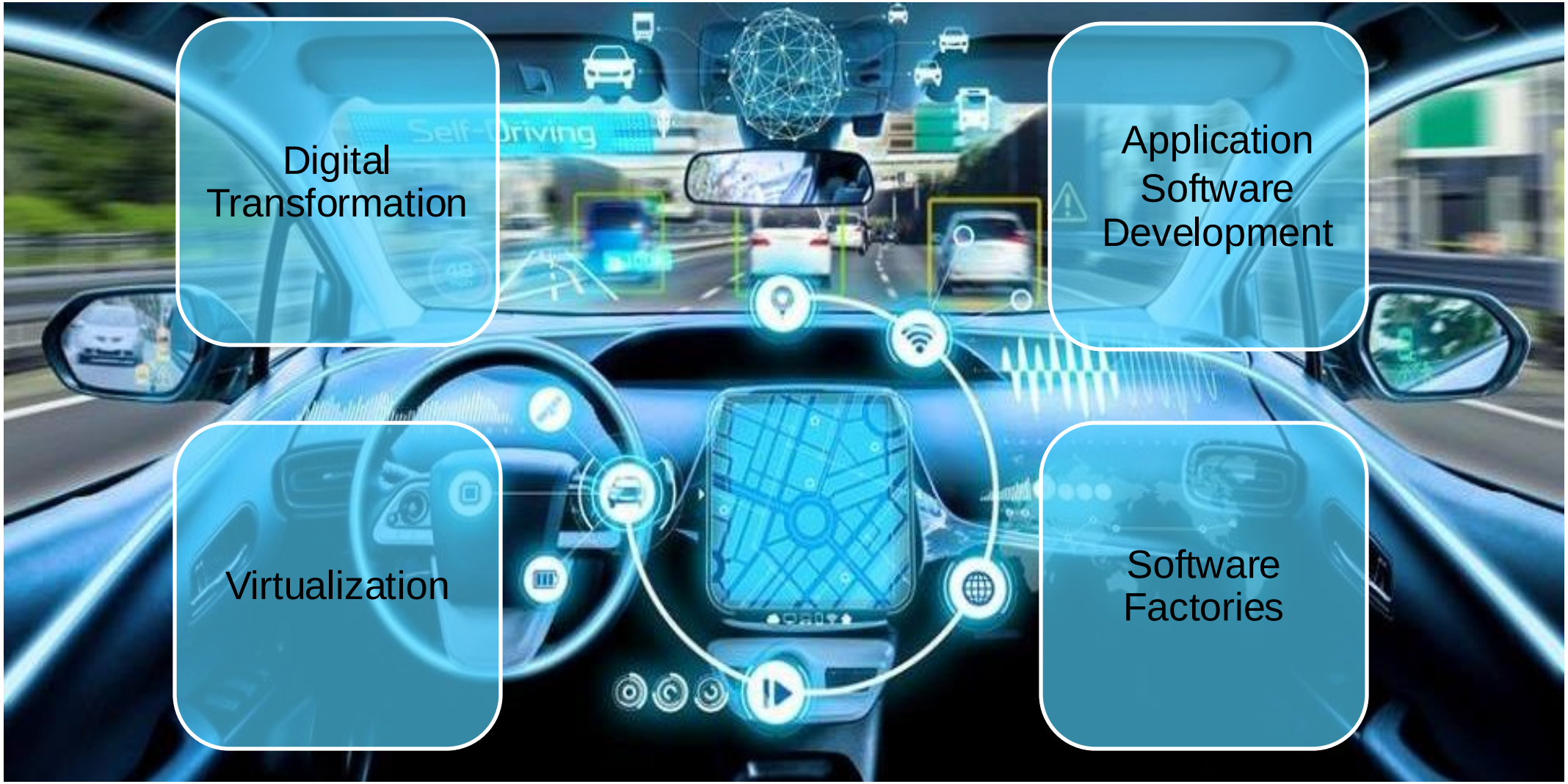
Dr. Rishu Gupta

Application Engineering Team,
ADAS & Artificial Intelligence

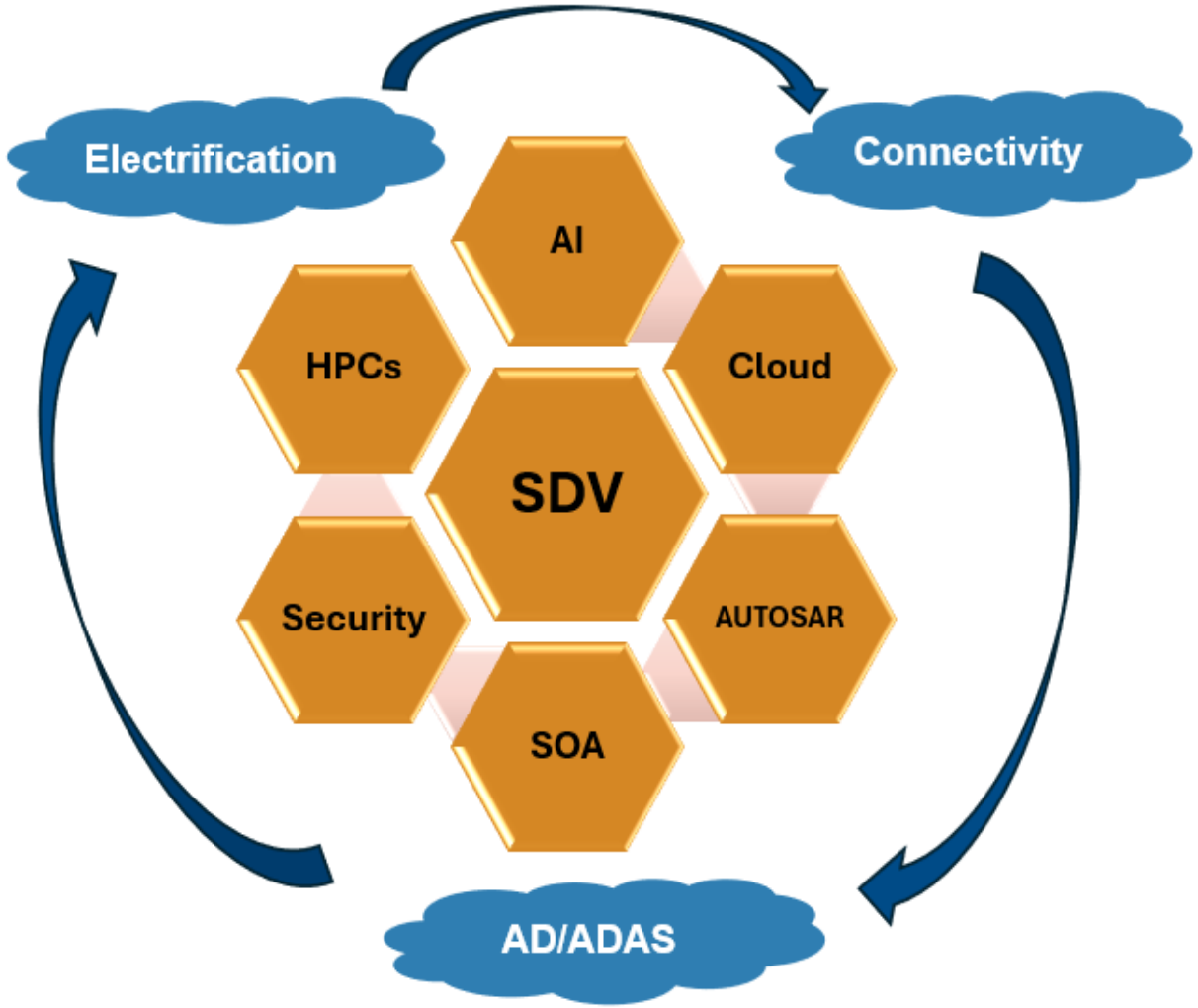
Kiran K Kulkarni

Industry Manager

What we are going to discuss:



Mega Trends Driving the Evolution of Software-Defined Vehicles

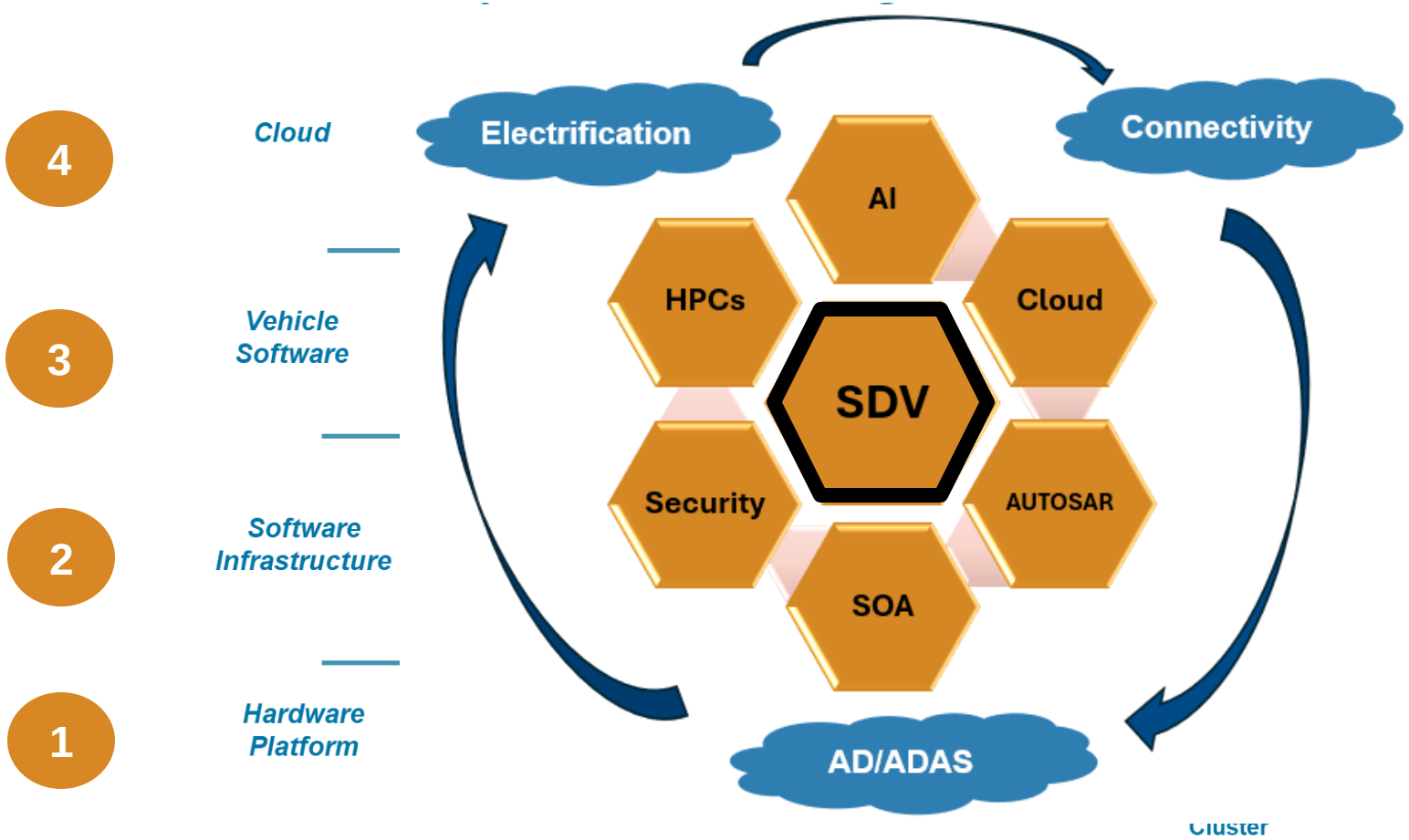


Automotive Industry is undergoing a profound transformation driven by convergence of various trends:

- **Electric Vehicles (EVs) and Electrification**
  - Fundamental transformation with advancements in battery technology.
- **Connected Cars and the IoT Revolution**
  - Safety, Predictive maintenance, and OTA updates.
  - 5G technology
- **Autonomous Driving and ADAS: Driving Smarter, Safer**
  - Investments in advanced sensor technology, AI and ML to improve safety and reliability.
  - Enormous data generated from sensors and camera
  - Sophisticated SW architecture.

Under the hood of a Software Defined Vehicle

Why Software Defined Transformation?
What are the Distinguishing Features of a Software Defined Vehicle



1: Changing architectures: Consolidation of ECUs and high-performance compute

2: Hardware and software decoupling: middleware

3: Changing application software: Signal to service orientation

4: Vehicle can communicate to the cloud

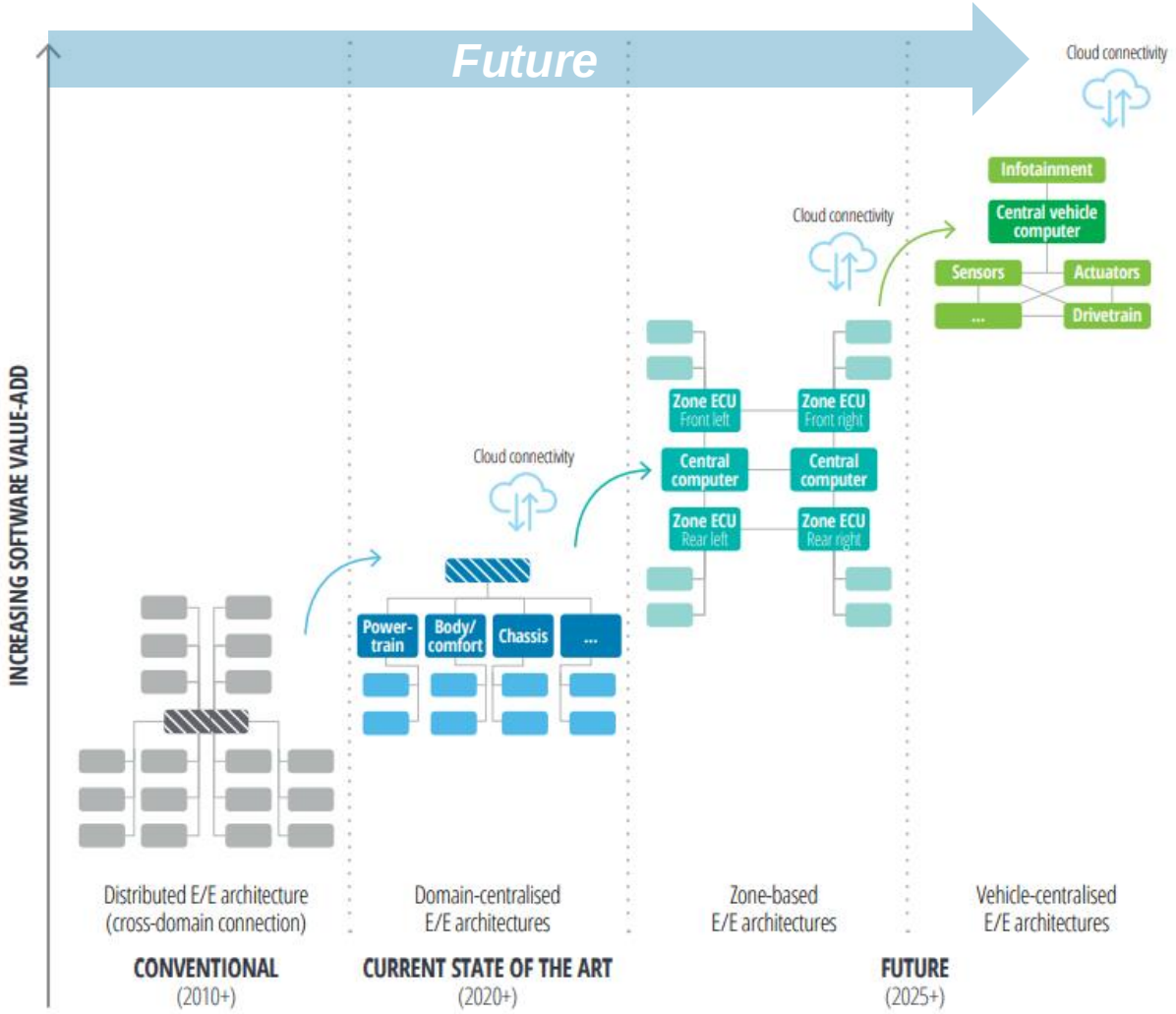
"OEMs lose US$900m annually in the US and Europe from physical recalls, and the number of vehicle recalls for software fixes has doubled in the last two years."

– Esync Alliance

1

Under the hood of a Software Defined Vehicle

Changing architectures: Consolidation of ECUs and high-performance compute



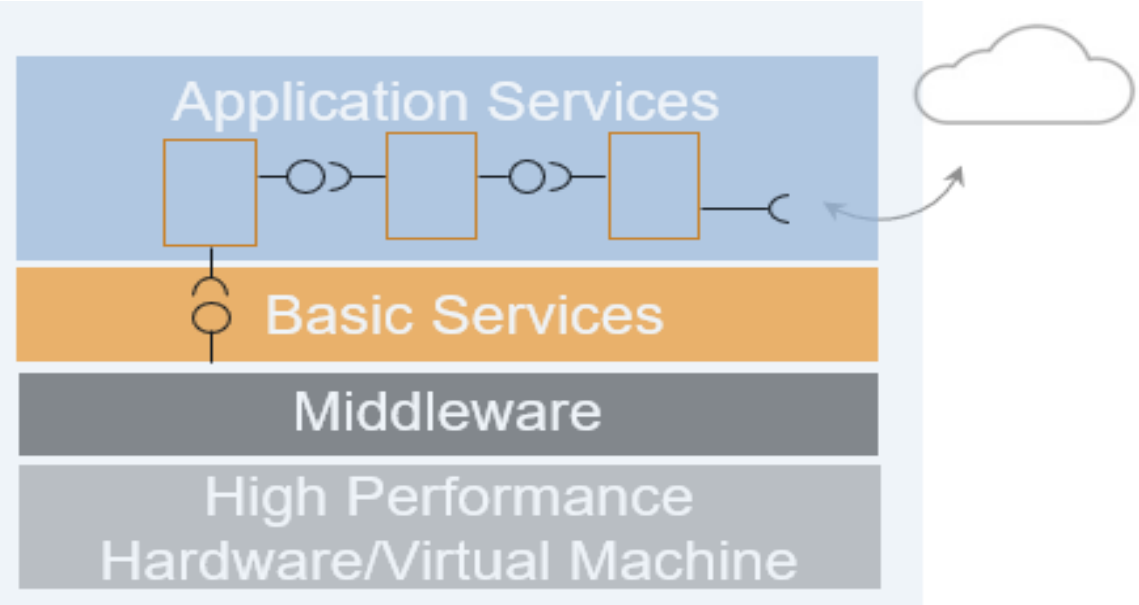
- Conventional: Constrained by memory, Low speed communication, High development effort, Lack of scalability and reusability
- Software Defined Vehicle: Vehicle computer (high compute) and zonal computers, Combines domains, Scalable, reusable, High-speed ethernet

Source: *Deloitte*

Under the hood of a Software Defined Vehicle

2

Hardware and software decoupling: middleware

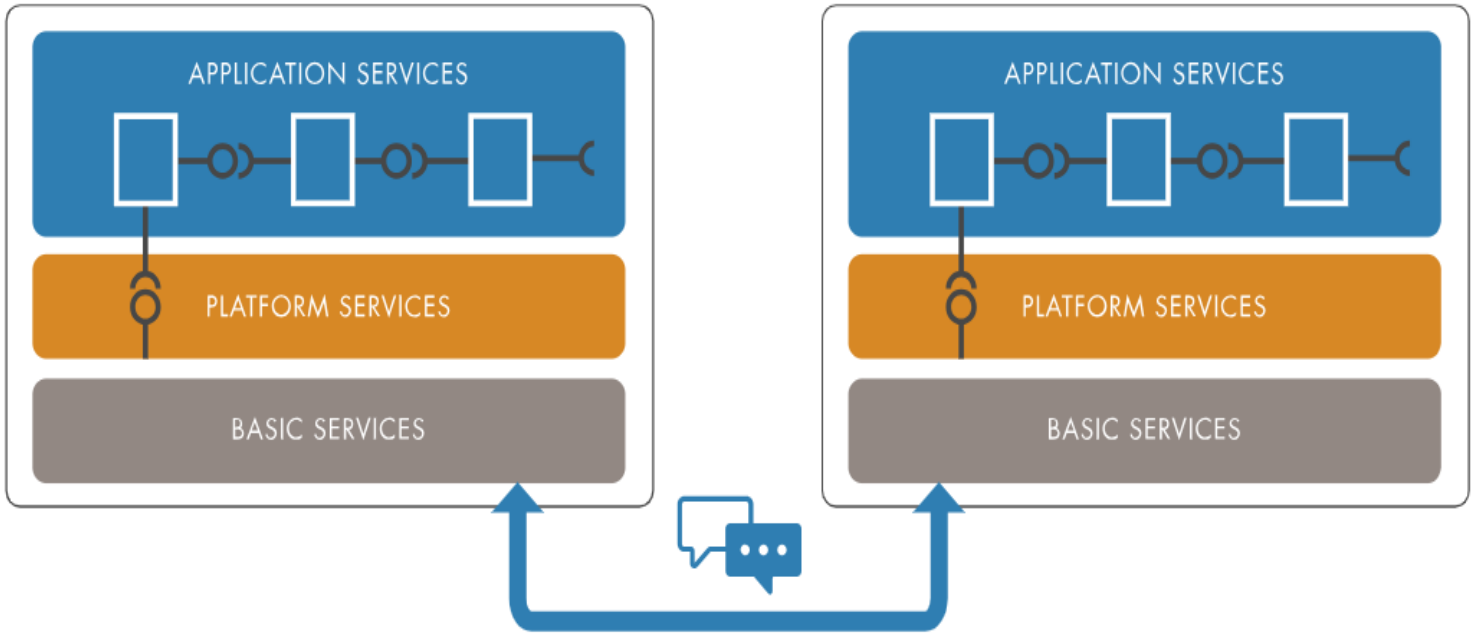


- Conventional: Hardware and software coupled
- Software Defined Vehicle: Hardware completely abstracted. Efficient communication from software functions

3

Under the hood of a Software Defined Vehicle

Changing application software: Signal to service orientation



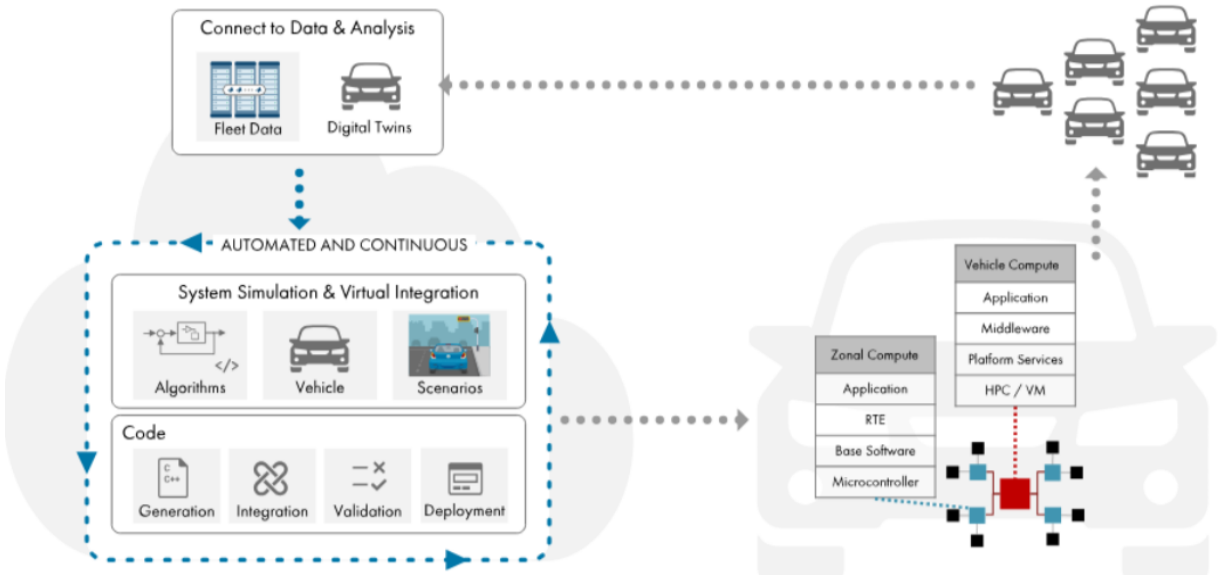
Service-oriented communication using messages.

- Conventional: Static architecture, signal based communication
- Software Defined Vehicle: Service-Oriented Architecture is a software design principle that promotes modular, loosely coupled, and interoperable services.

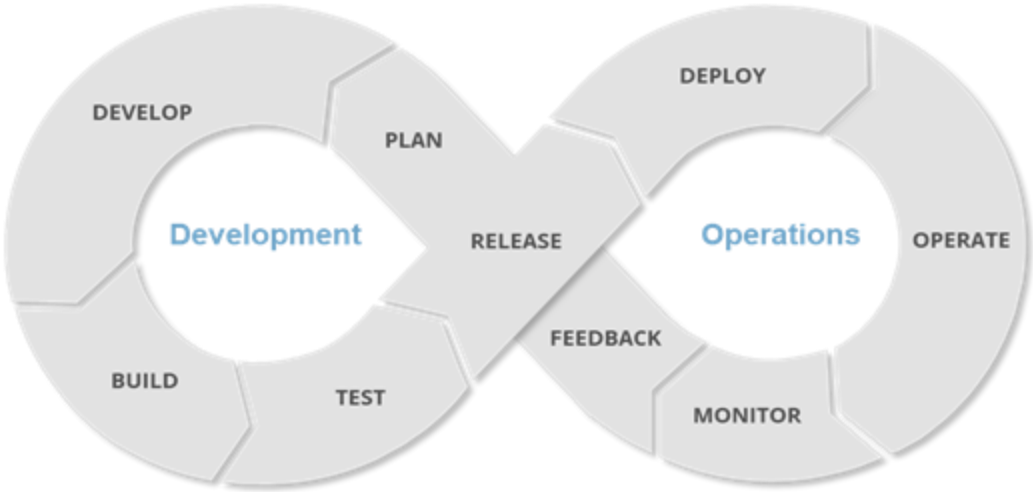
4

Under the hood of a Software Defined Vehicle

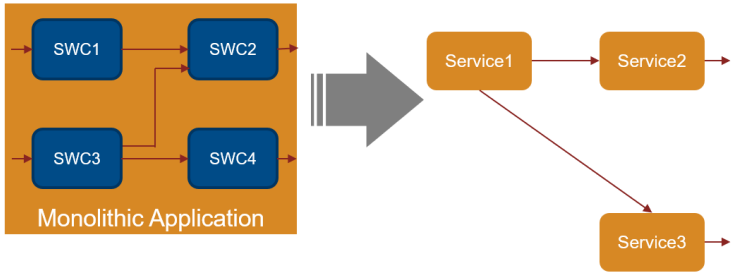
Vehicle can communicate to the cloud



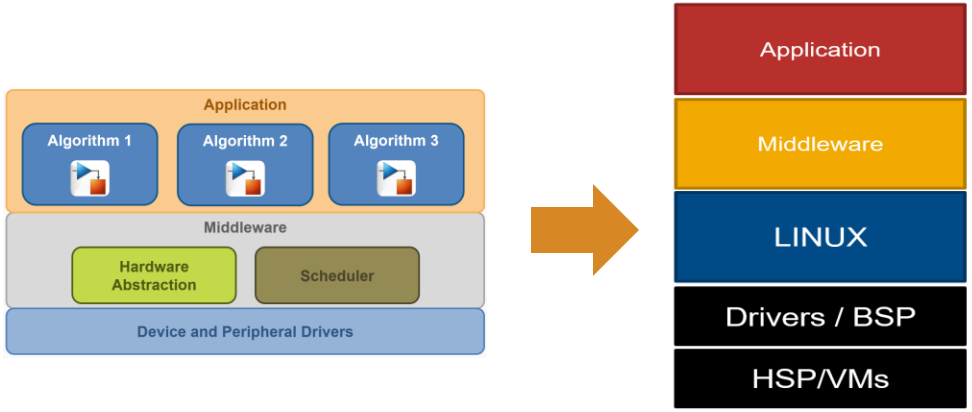
- Conventional: V-Cycle Development
- Software Defined Vehicle: Faster cycles of development using DevOps. Features on demand.



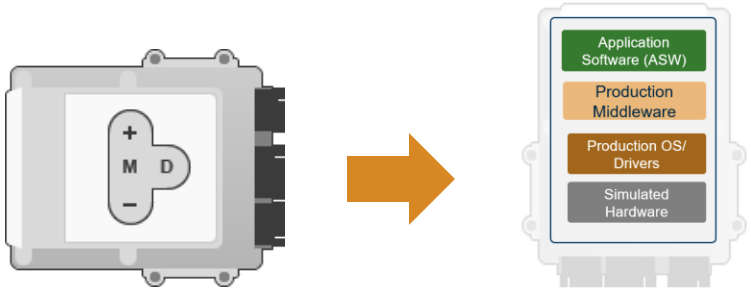
Developing SW for Modern Vehicles i.e., SDVs



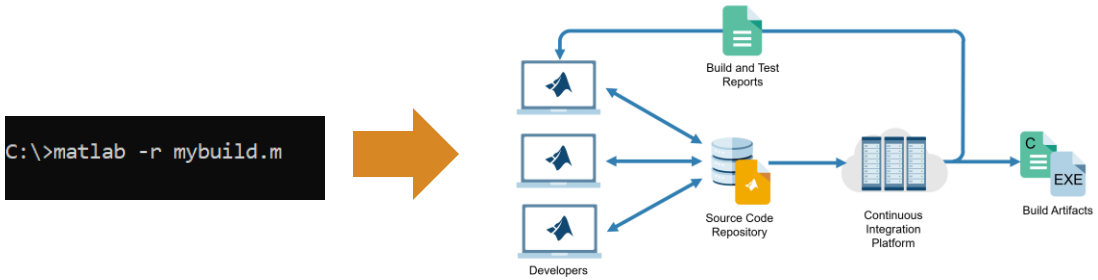
Monolithic vs. **Service Oriented App (SOA)**



Bare Metal vs. **Linux** Target

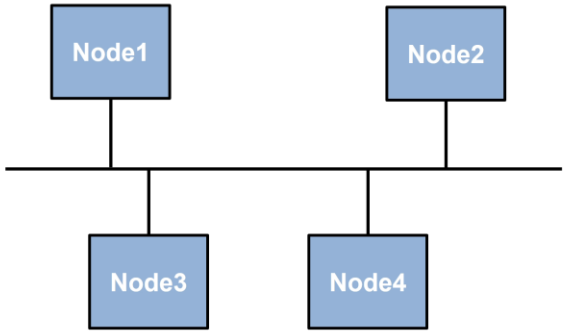


Traditional vs. **Virtual** Validation



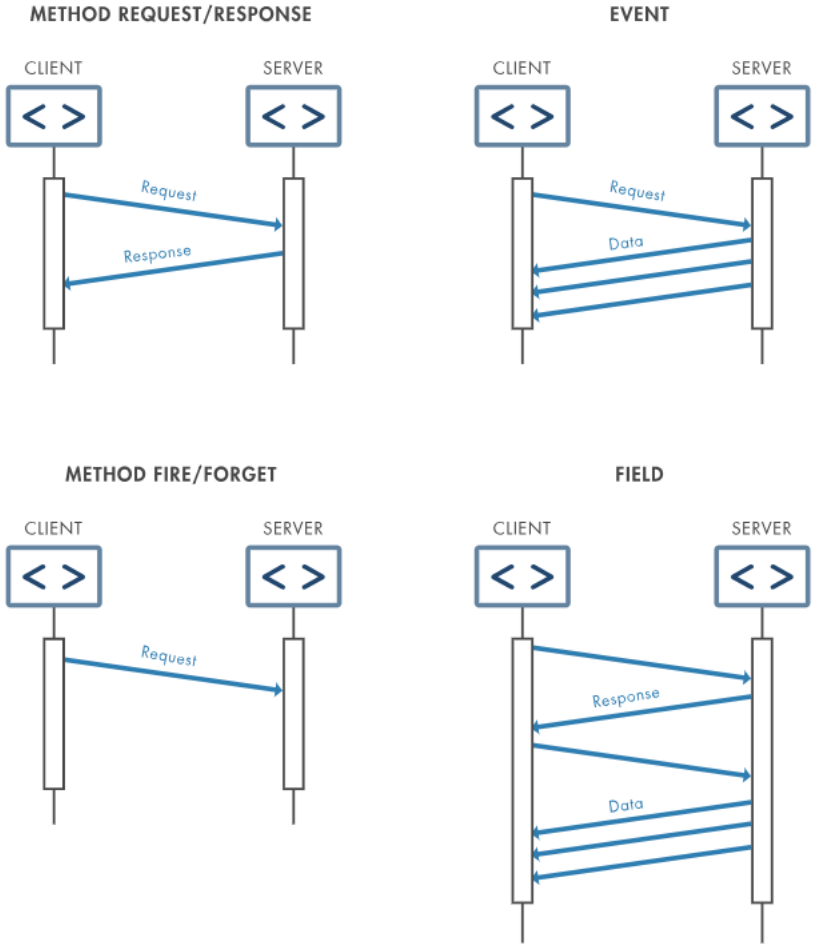
Batch vs. **CI/CD** Automation

Service Oriented Communication (SOC)



signal-oriented communication

- send data independent of needs
- high bus load
- not efficient

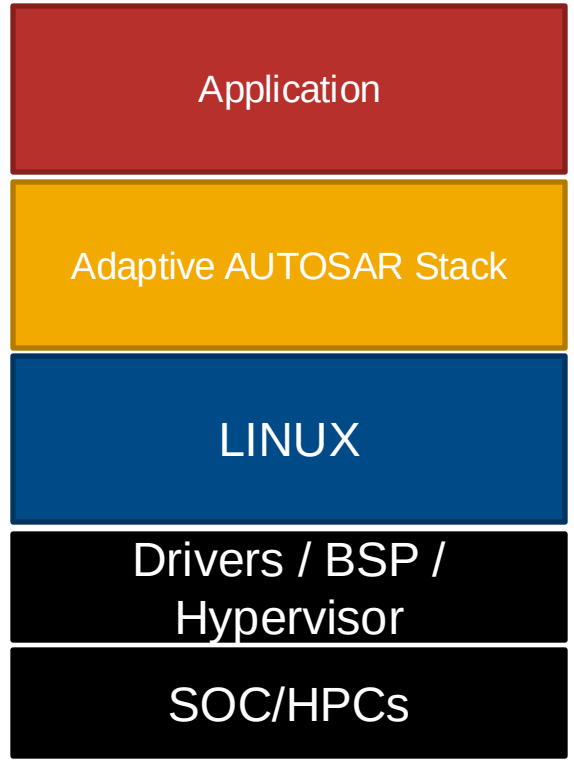


SOA Application Interface Patterns

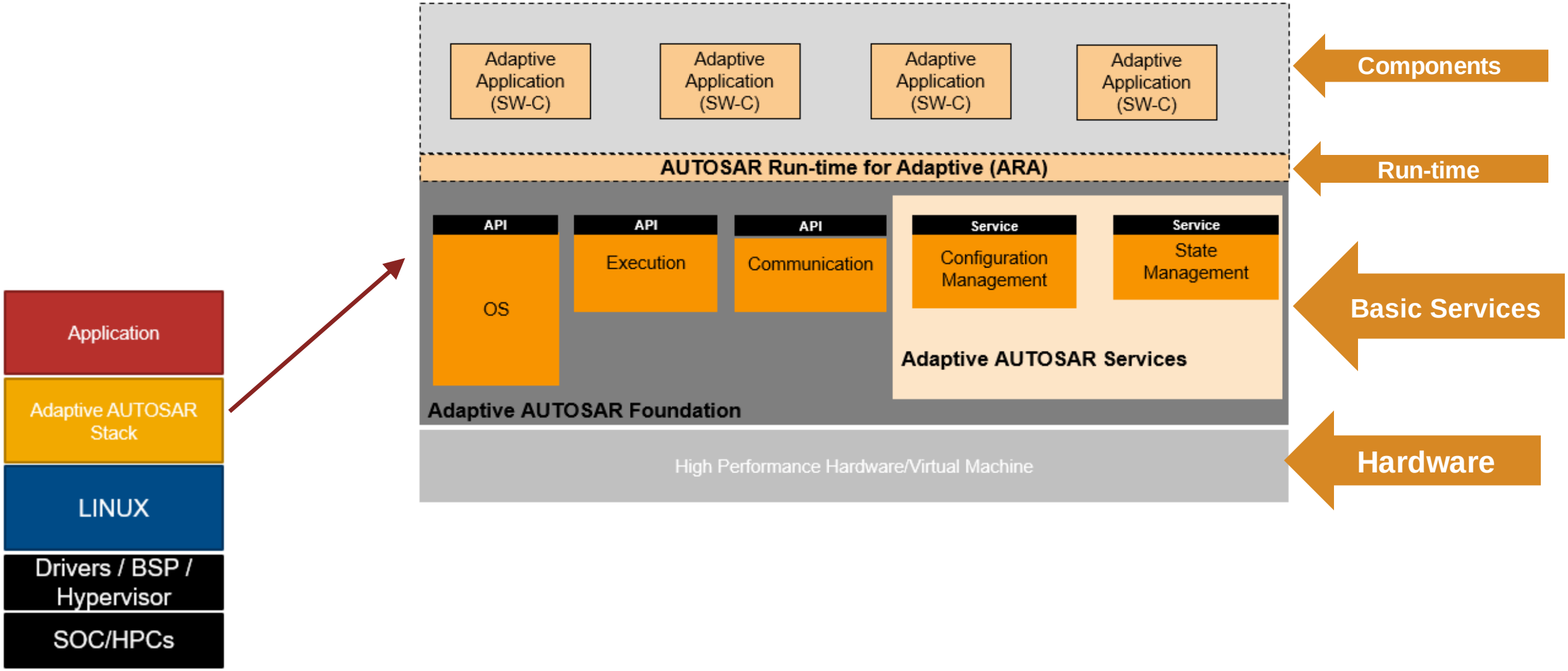
SOA Application Communication

Service Oriented Architecture (SOA) – How?

- SOA is used by multiple industrial standard *middleware* including:
  - AUTOSAR Adaptive Platform
  - DDS (Data Distribution Services)
  - ROS (Robot Operating System)



AUTOSAR Layered Software Architecture

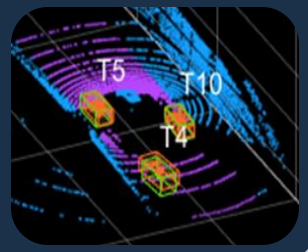


Automated Driving Algorithm Development

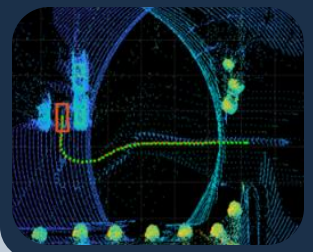
Multidisciplinary Skills

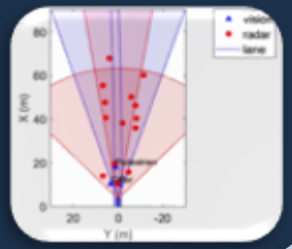
Algorithms

Perception



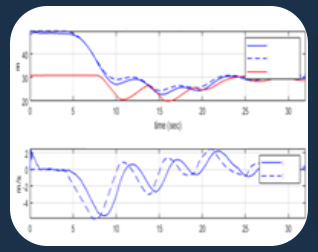
Sensor Fusion

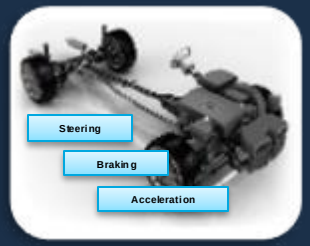




Planning

Decision & Controls





Key subsystems of an automated driving system

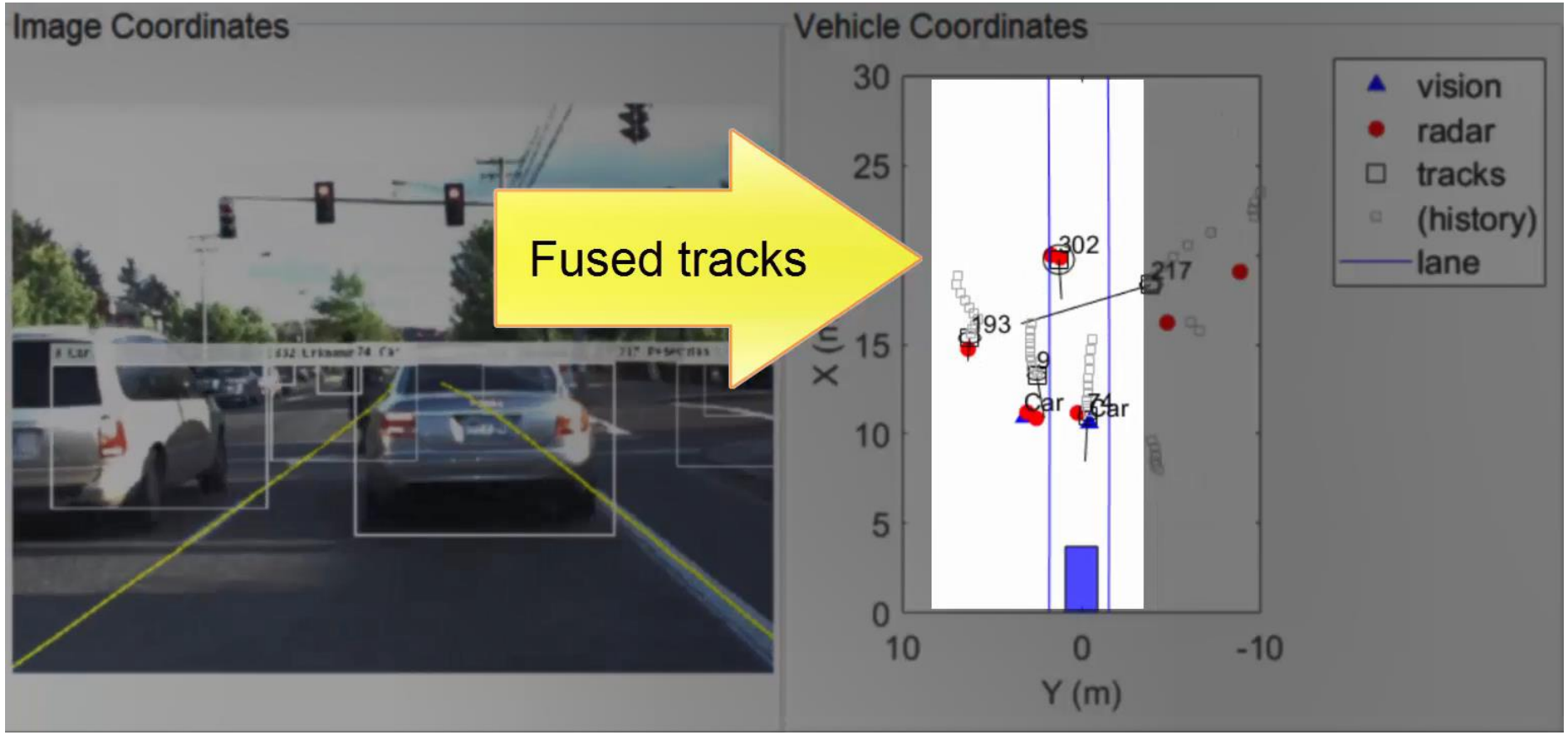
Key subsystems of an automated driving system

Perception

Key subsystems of an automated driving system

Perception

Sensor Fusion



Key subsystems of an automated driving system

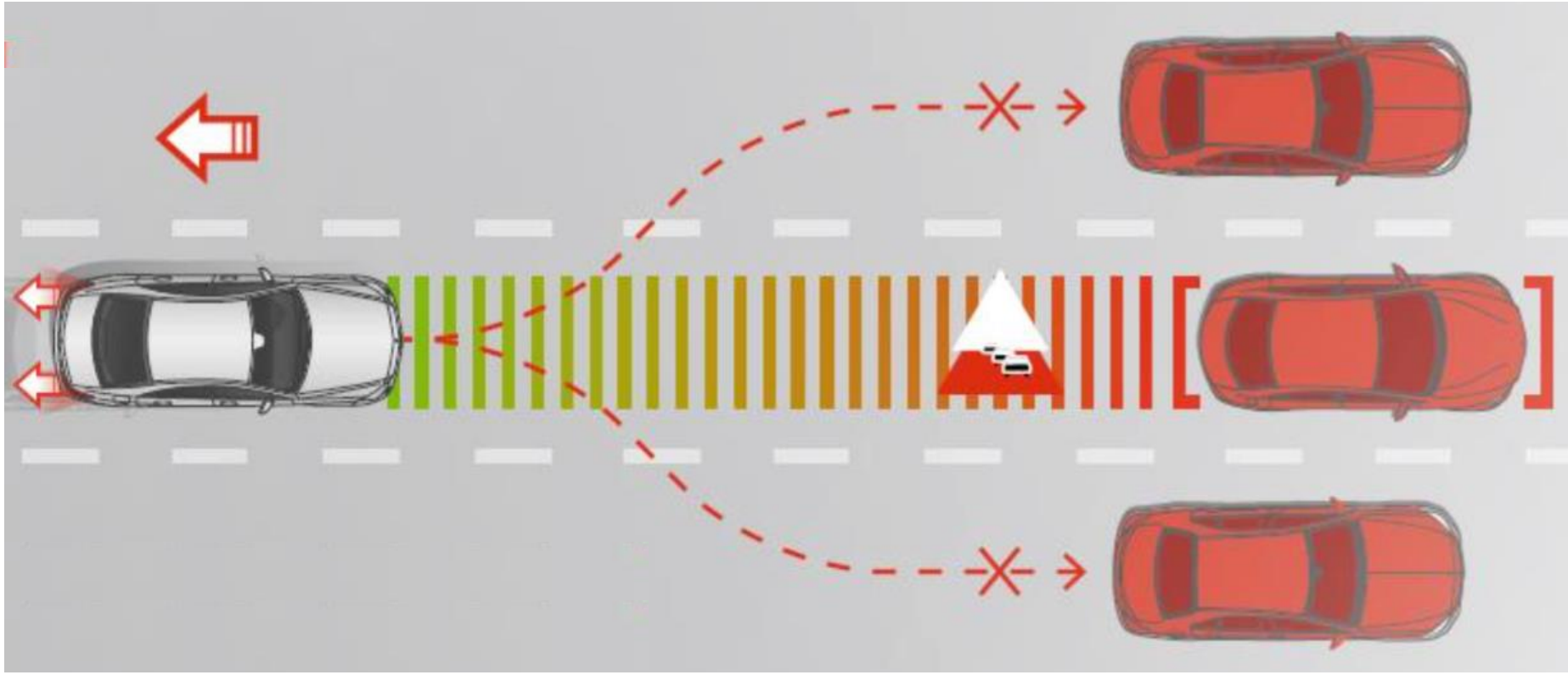
Perception

↓

Sensor Fusion

↓

Planning



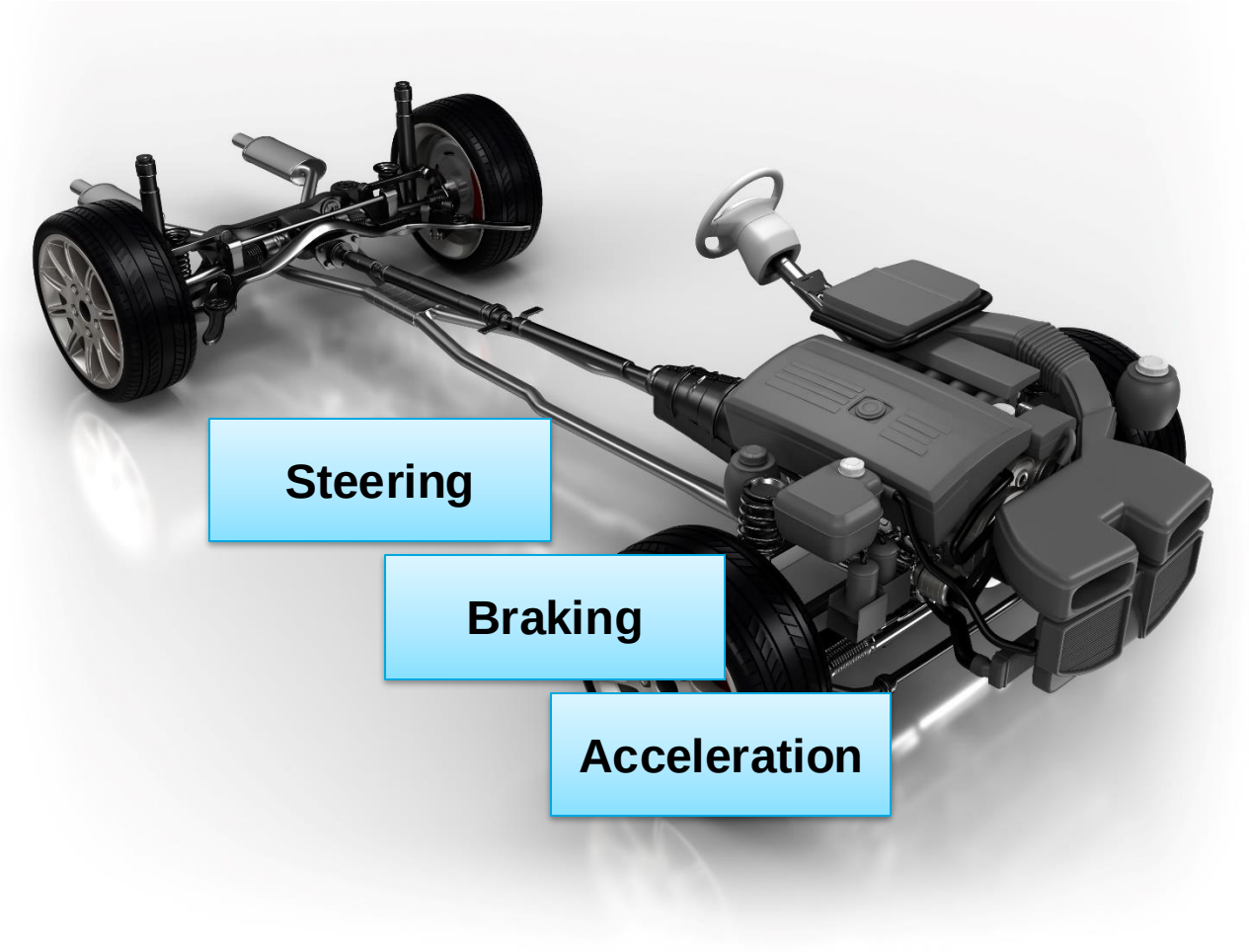
Key subsystems of an automated driving system

Perception

Sensor Fusion

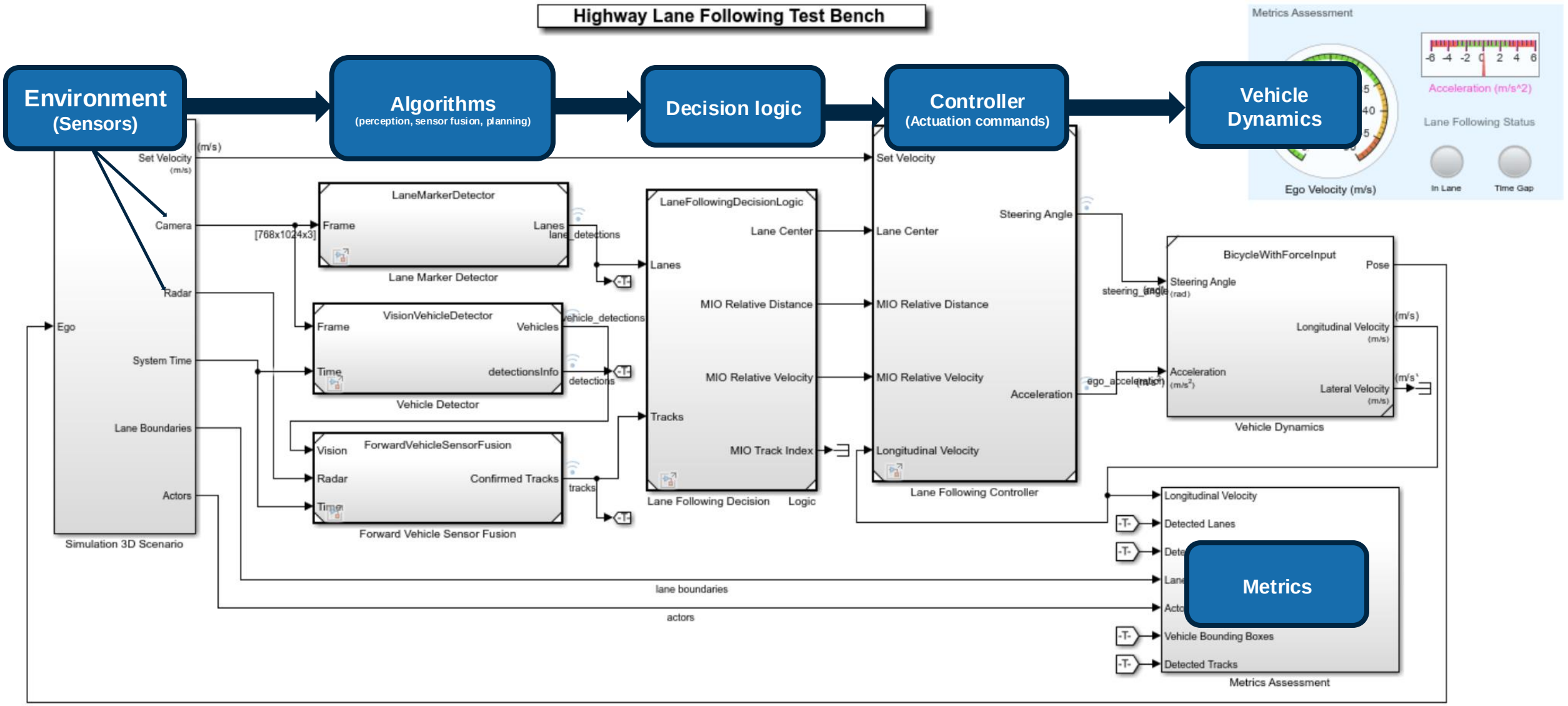
Planning

Controls

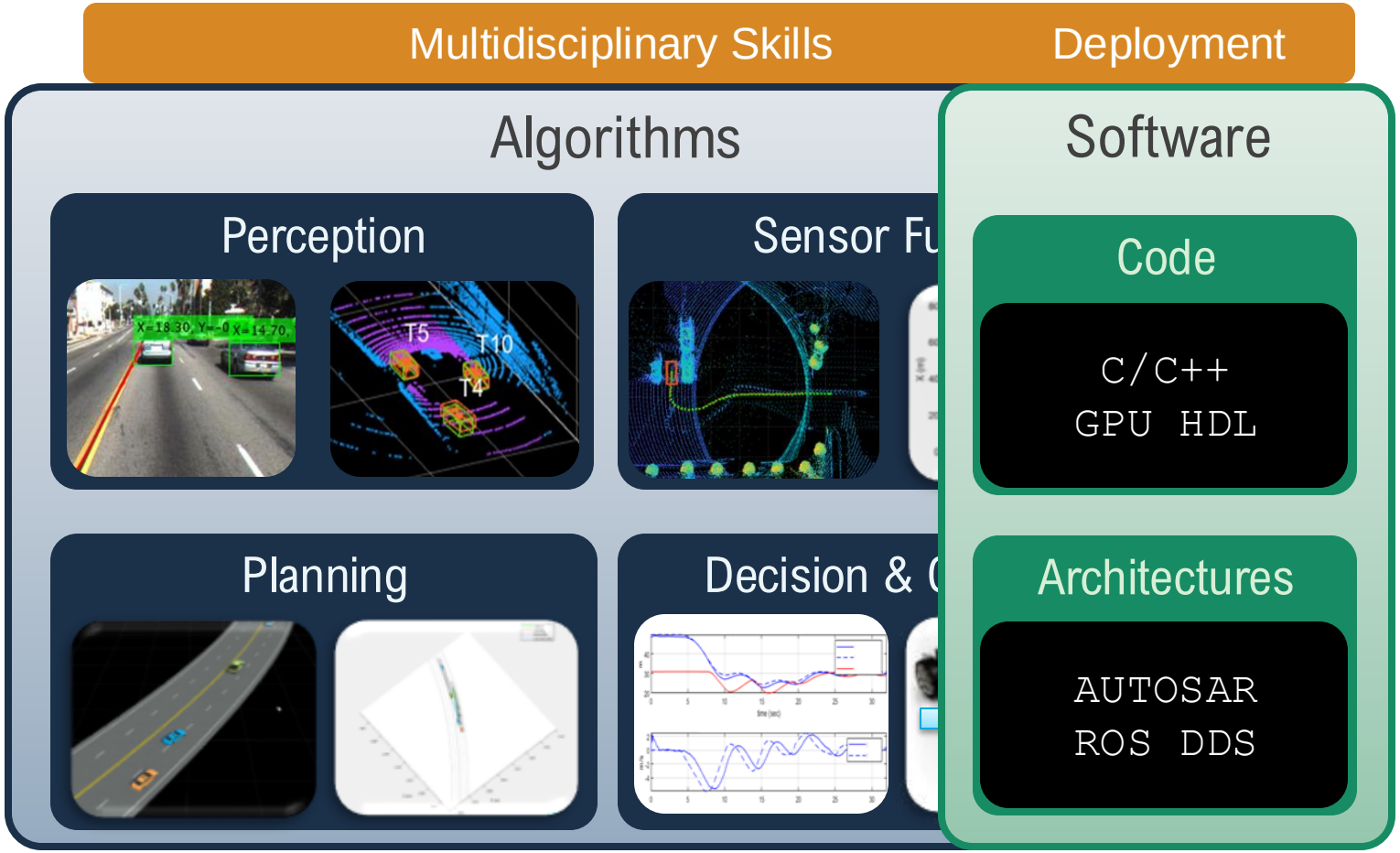


Key subsystems of an automated driving system

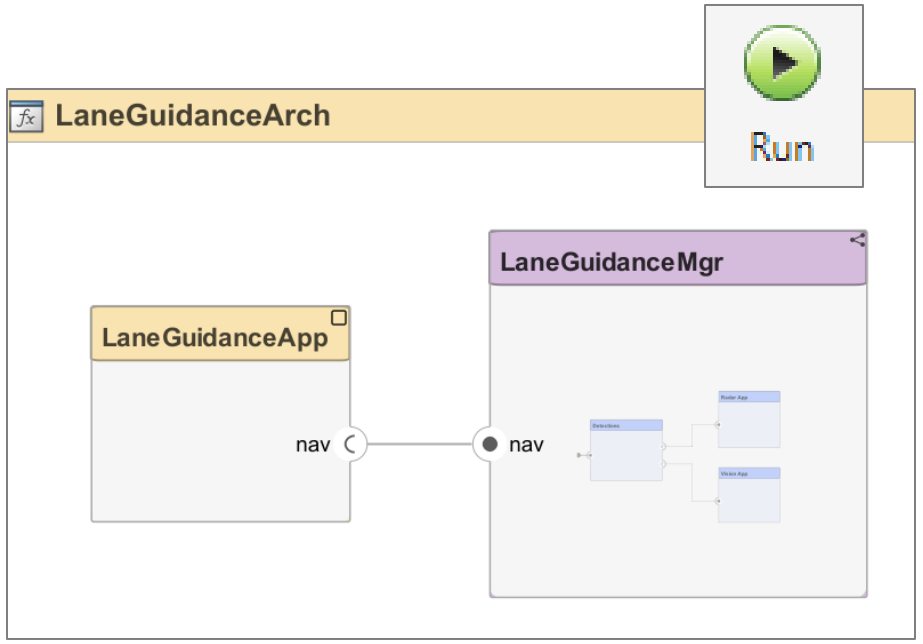
Highway Lane Following



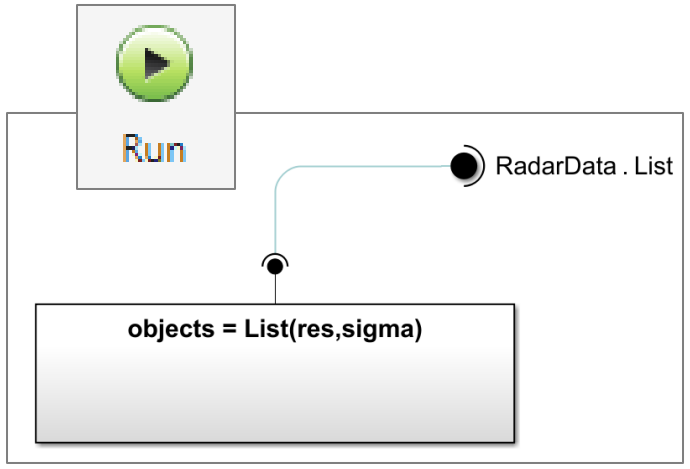
Automated Driving Algorithm Development



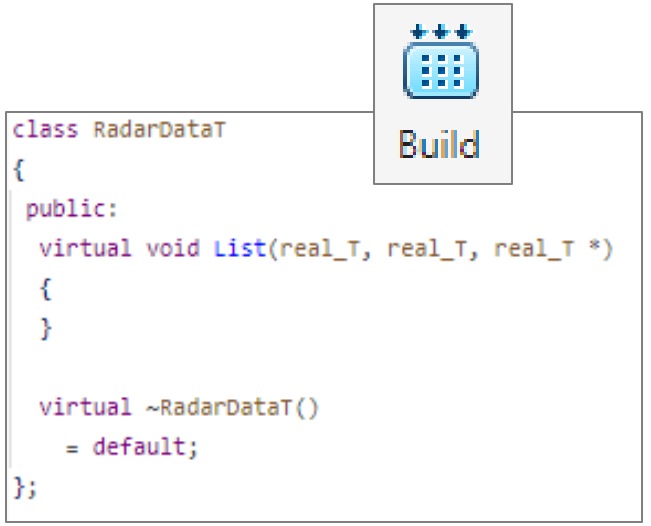
Service-Oriented Architecture (SOA) Design



Describe SOA with
System Composer



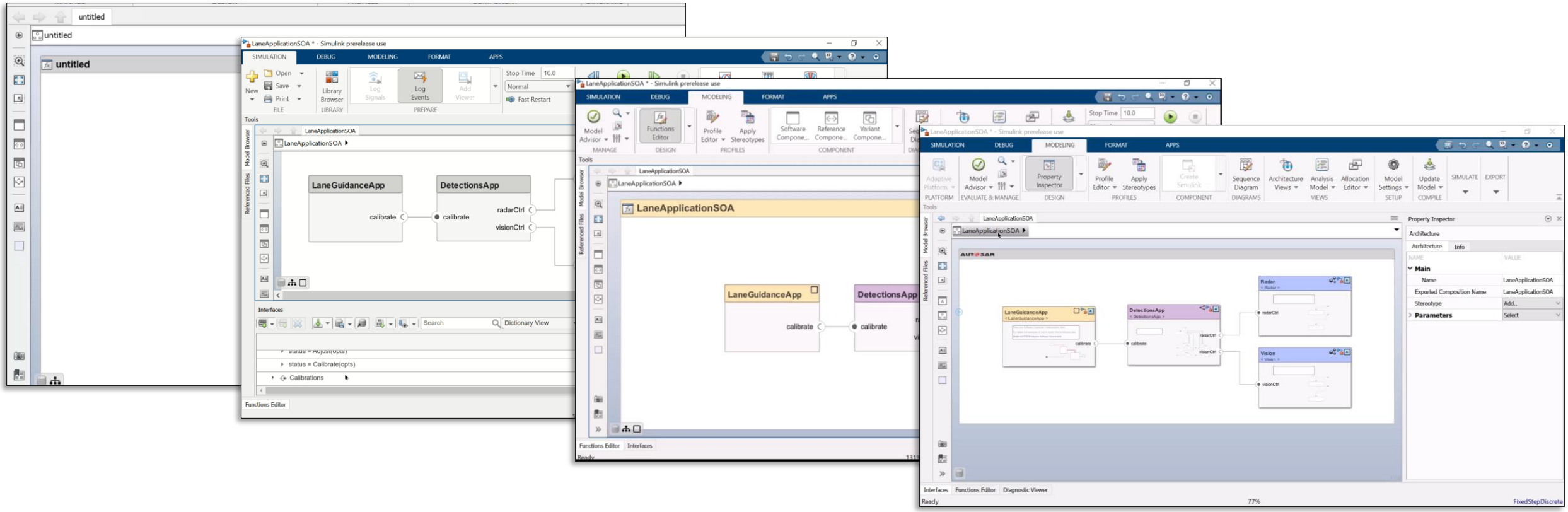
Implement detailed components with
Simulink



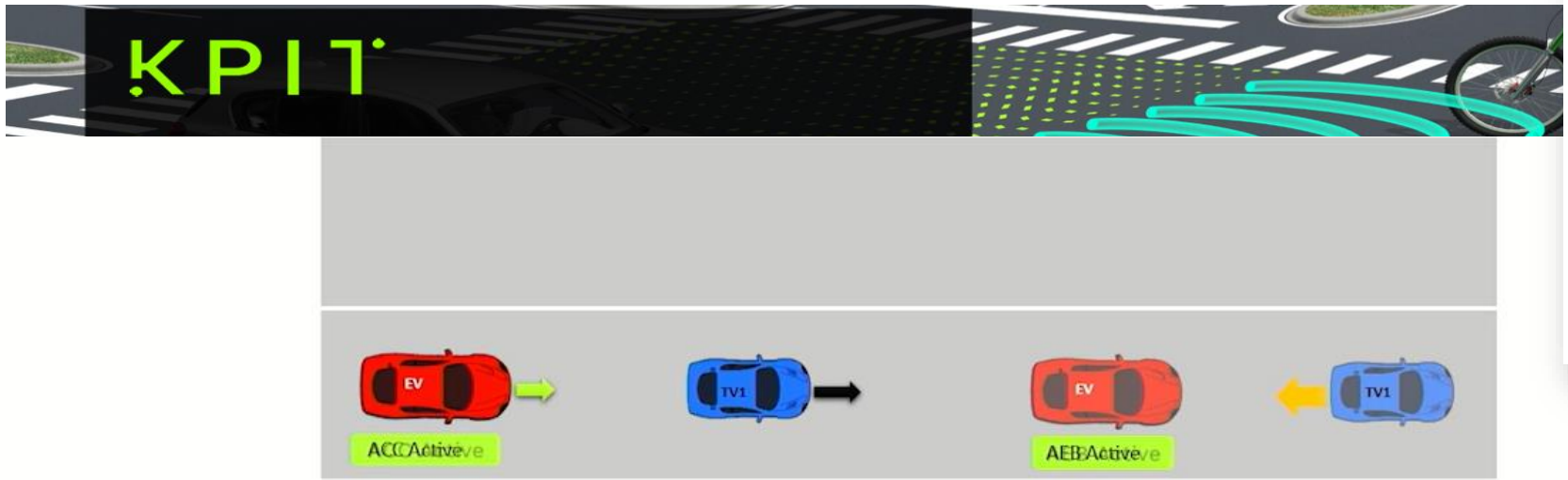
Generate code with
Embedded Coder

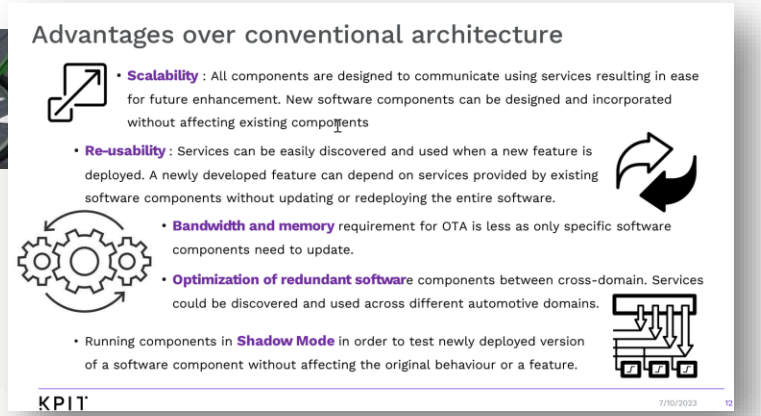
How to decompose traditional application software compositions into services for Software Defined Vehicles applications?

Identify and Analyze Services



KPIT- Service-oriented arbitration of ADAS features with Model-Based Design





[Link to the talk]

Description: -

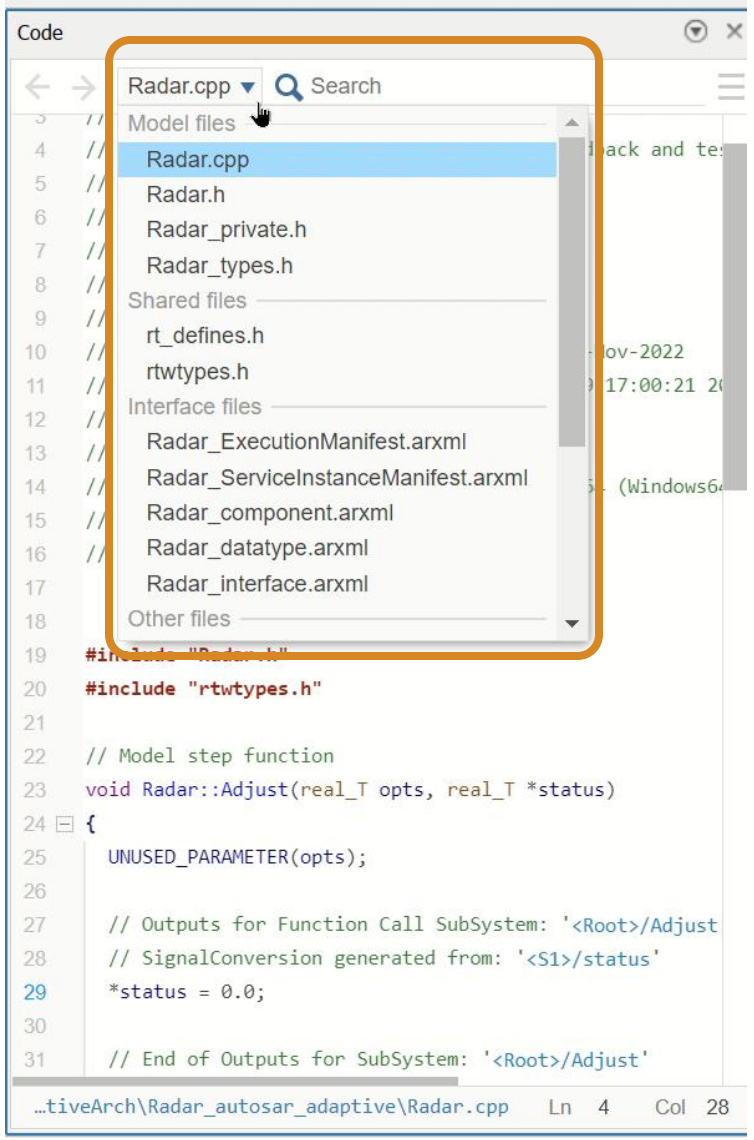
- Traffic Vehicle (TV1) is cruising on the road with a little lower speed than ego vehicle(EV)(lower relative speed)
- Ego Vehicle enters follow mode and decelerates to match TV1 speed
- After a while, TV1 performs sudden deceleration. Current TTC is less than the threshold TTC for activation of emergency feature. Arbitration accepts the maneuver request of AEB.

KPIT

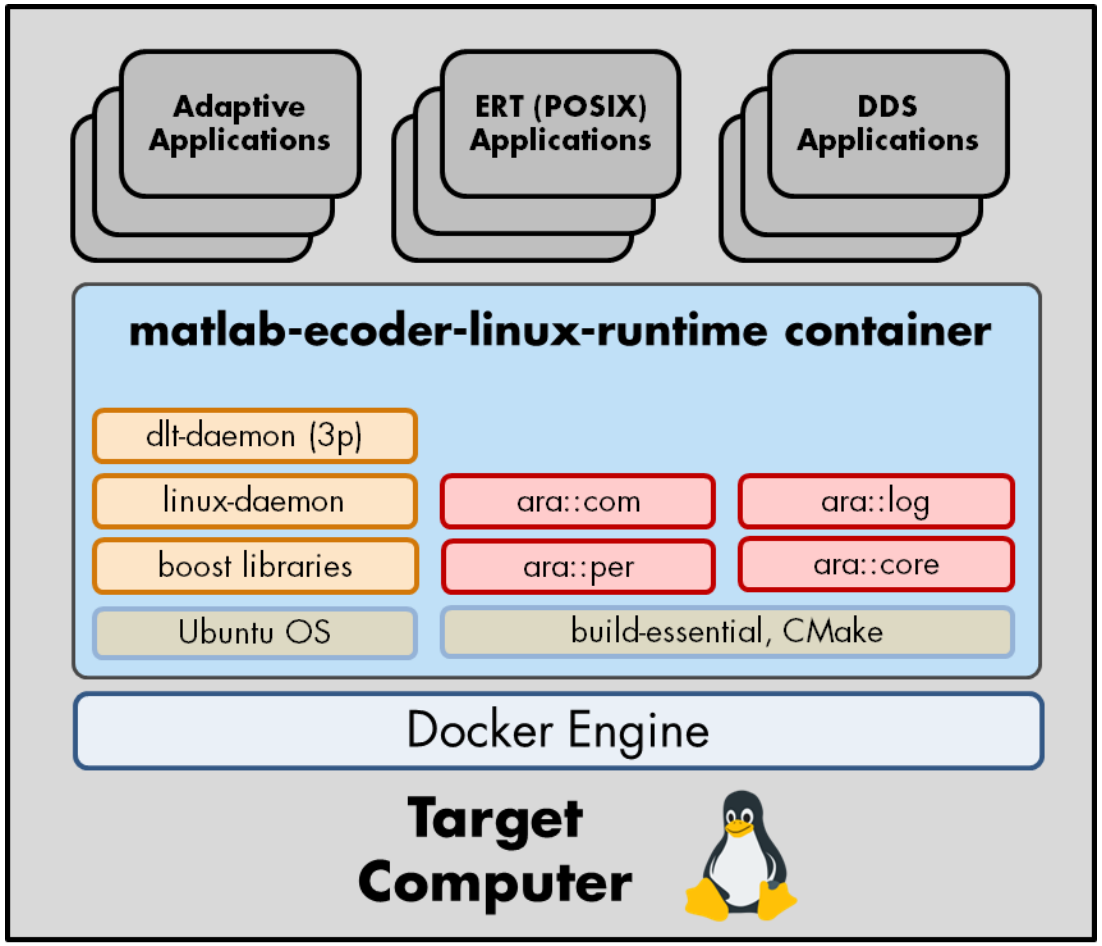
7/12/2023 10

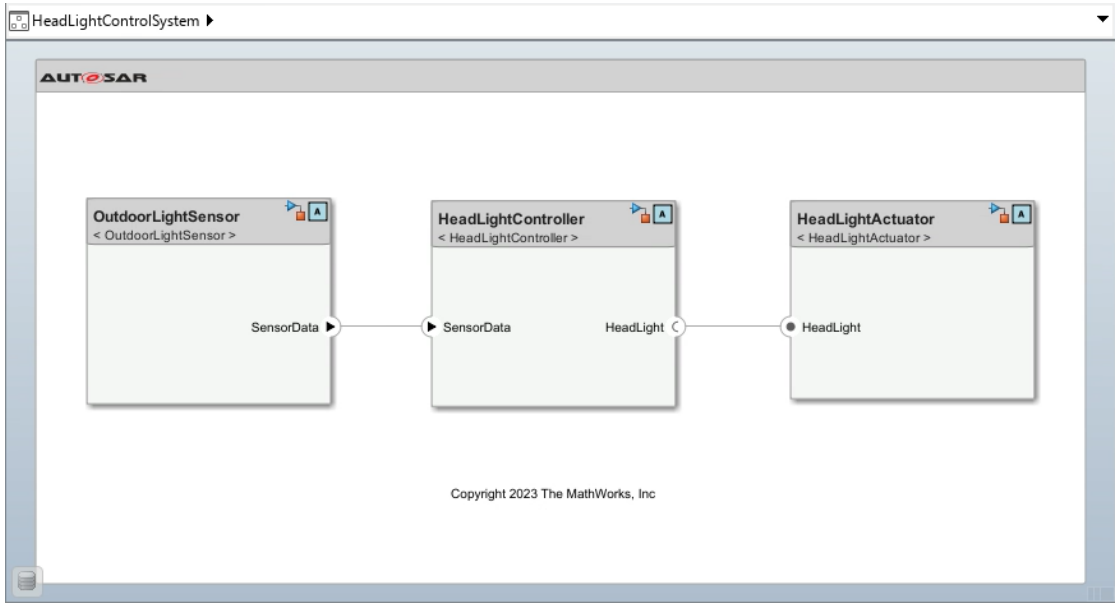
Implement and Deploy Services

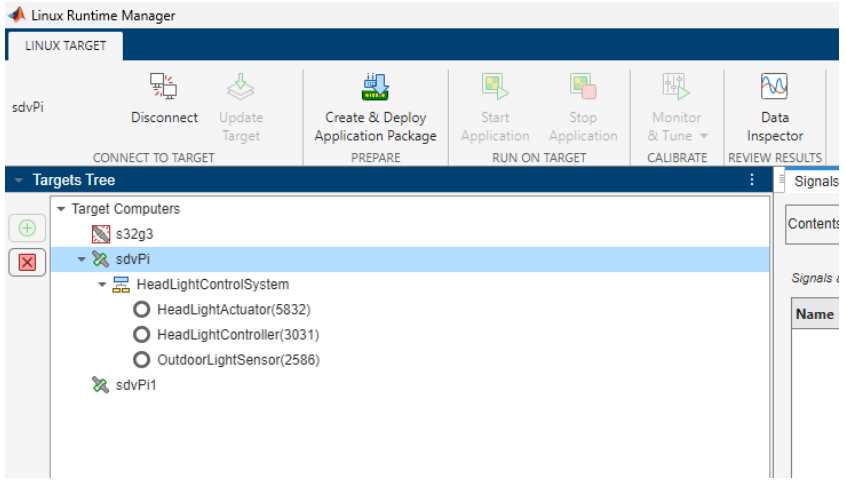
- Each service need to be deployed as a standalone application, with its own artifacts including
  - Code
    - C++ Code
    - ARA Stub
  - AUTOSAR interface descriptions
    - Machine Manifest
    - Execution Manifest
    - Service Instance Manifest



Deploy, Control, and Instrument Software Applications on Linux Platform (Run-Time Environment)



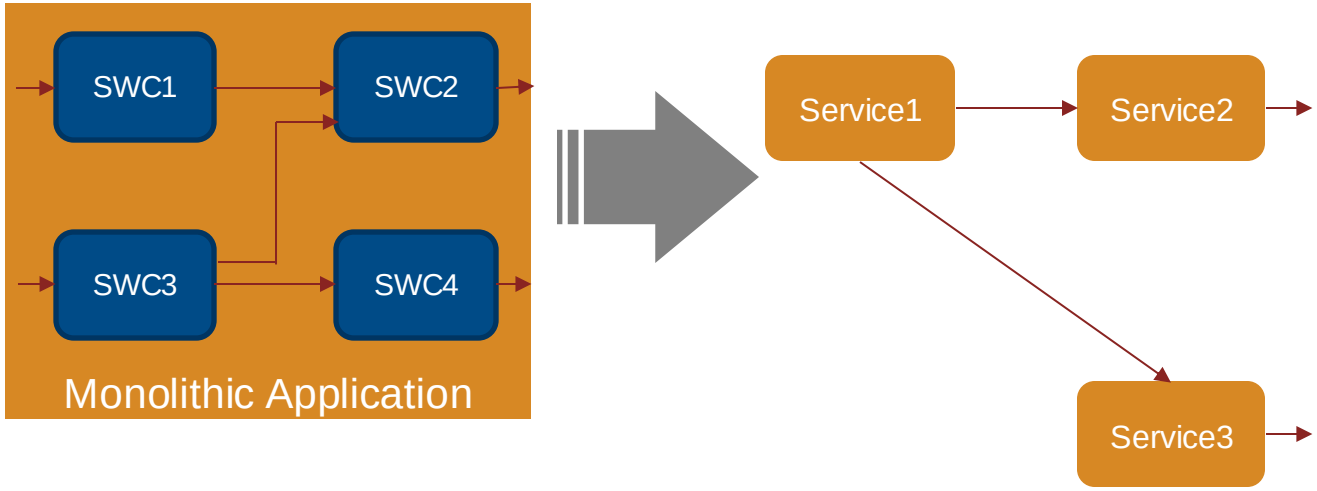




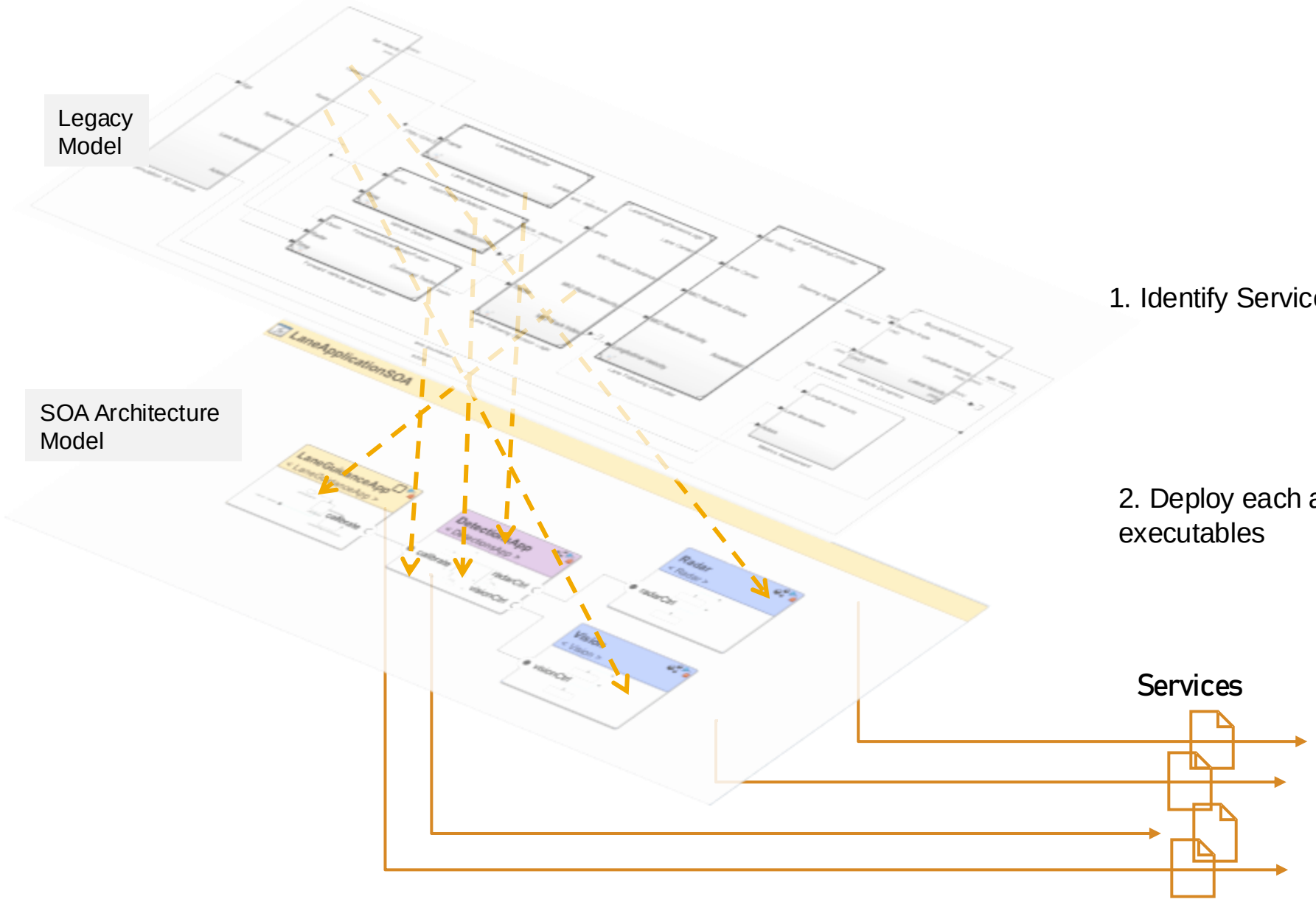
Decompose the Monolithic Applications to SOA

To decompose monolithic application components to services we need to :

- Identify the different components, functionalities, and dependencies
- Understand component interactions and execution order of the components



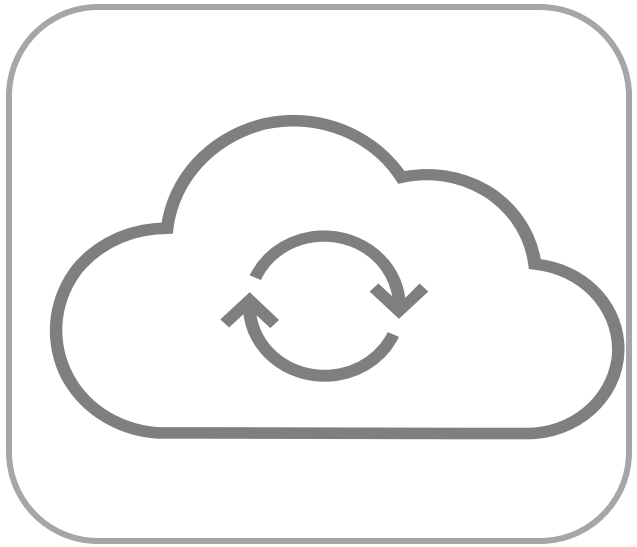
Decompose the Monolithic Application



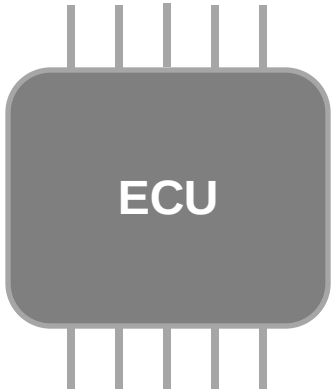
1. Identify Service Components

2. Deploy each as service executables

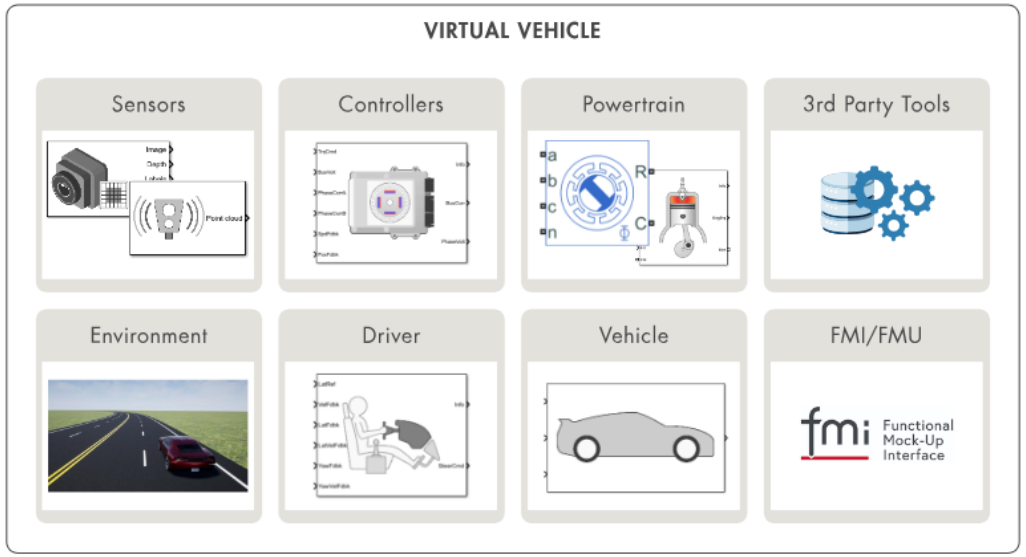
Exploring Virtualization: The MathWorks Perspective



Virtualization in the cloud



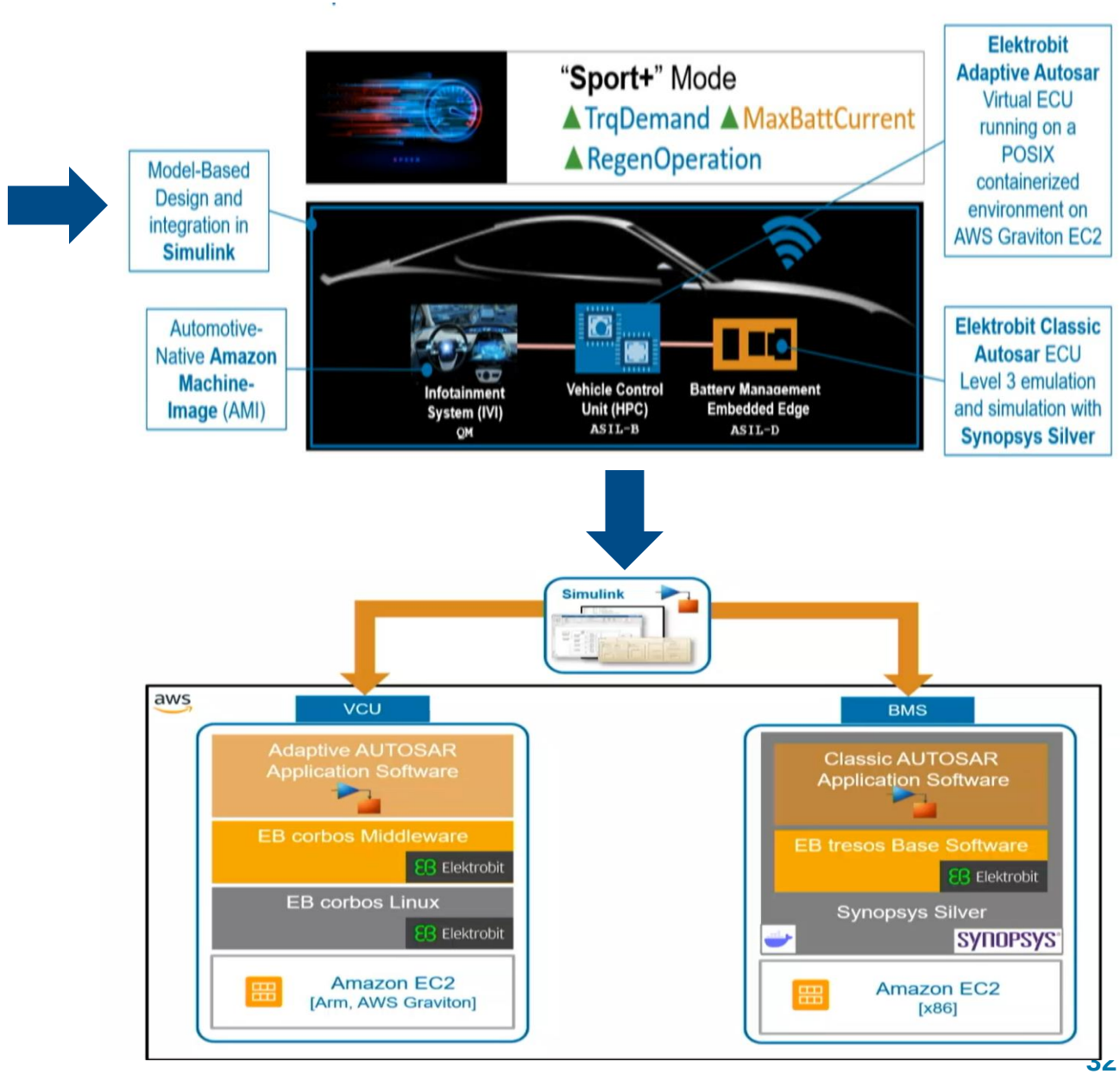
Virtualization of ECU



Virtualization of complex Scenarios

Use case of virtualization on the cloud

Use case: *Enable a new feature—Sport + mode—that reduces the time taken by vehicle to accelerate from 0 to 60 mph (100 kph) per hour by 10 percent.*

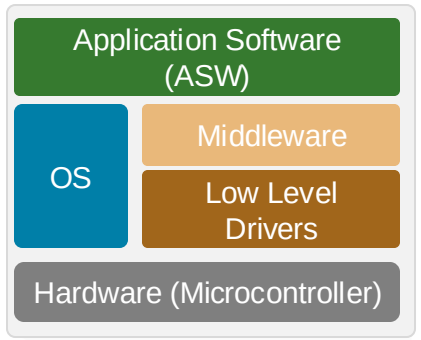


MathWorks aws FOR AUTOMOTIVE

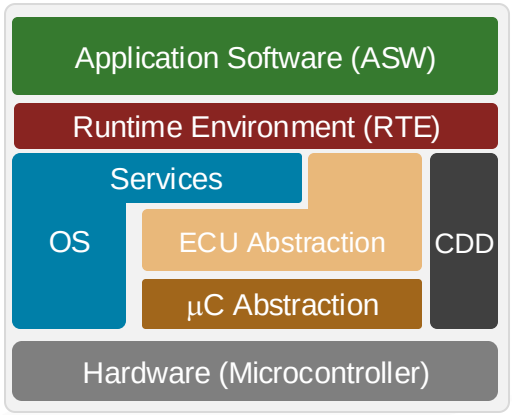
Elektrobit SYNOPSYS

Virtualization

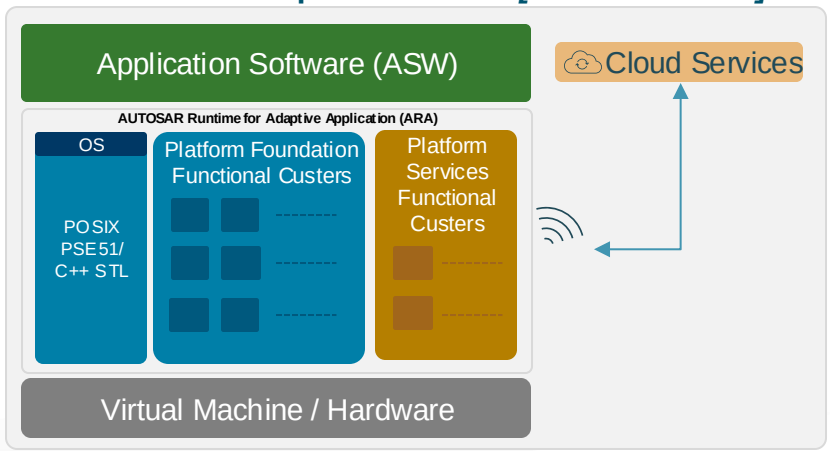
Classic Embedded Controller

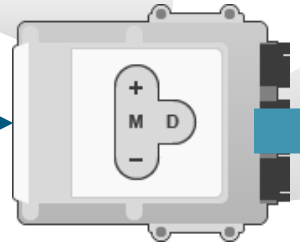


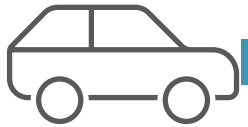
AUTOSAR Classic Platform



AUTOSAR Adaptive Platform *[HPC Controllers]*



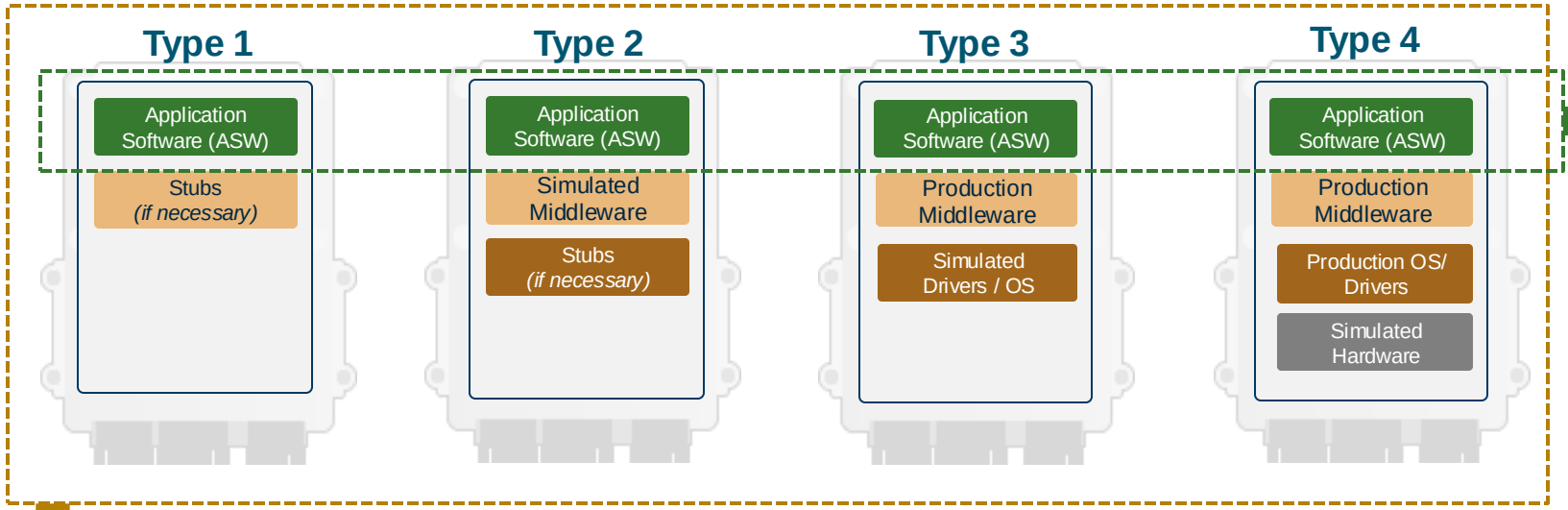




Vehicle/ Component Models
[OR]
Real Components/Vehicle

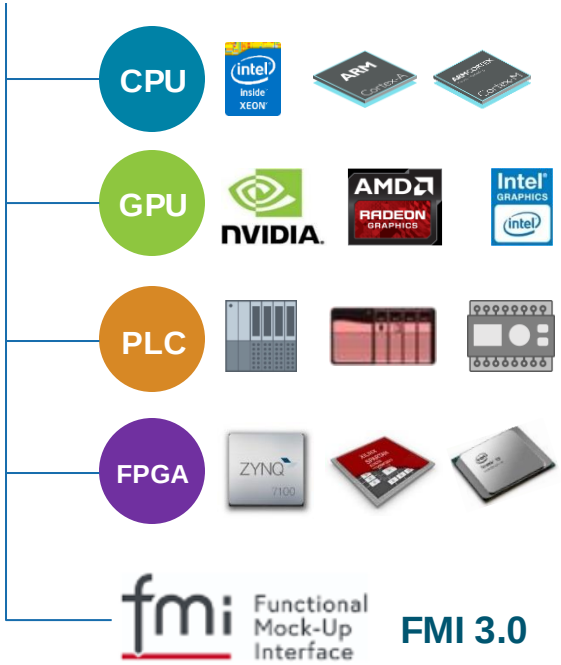
1. Virtualizing the ECU – **VirtualECU**
2. Virtualizing complex models - **Virtual Vehicle & Components**
3. Supporting **Cloud Workflows** for Virtualization

Virtualization – Virtual ECU [vECU]



Model-Based Design for **traditional embedded development to SOA workflow for SDV,** with **Automated Code Generation**

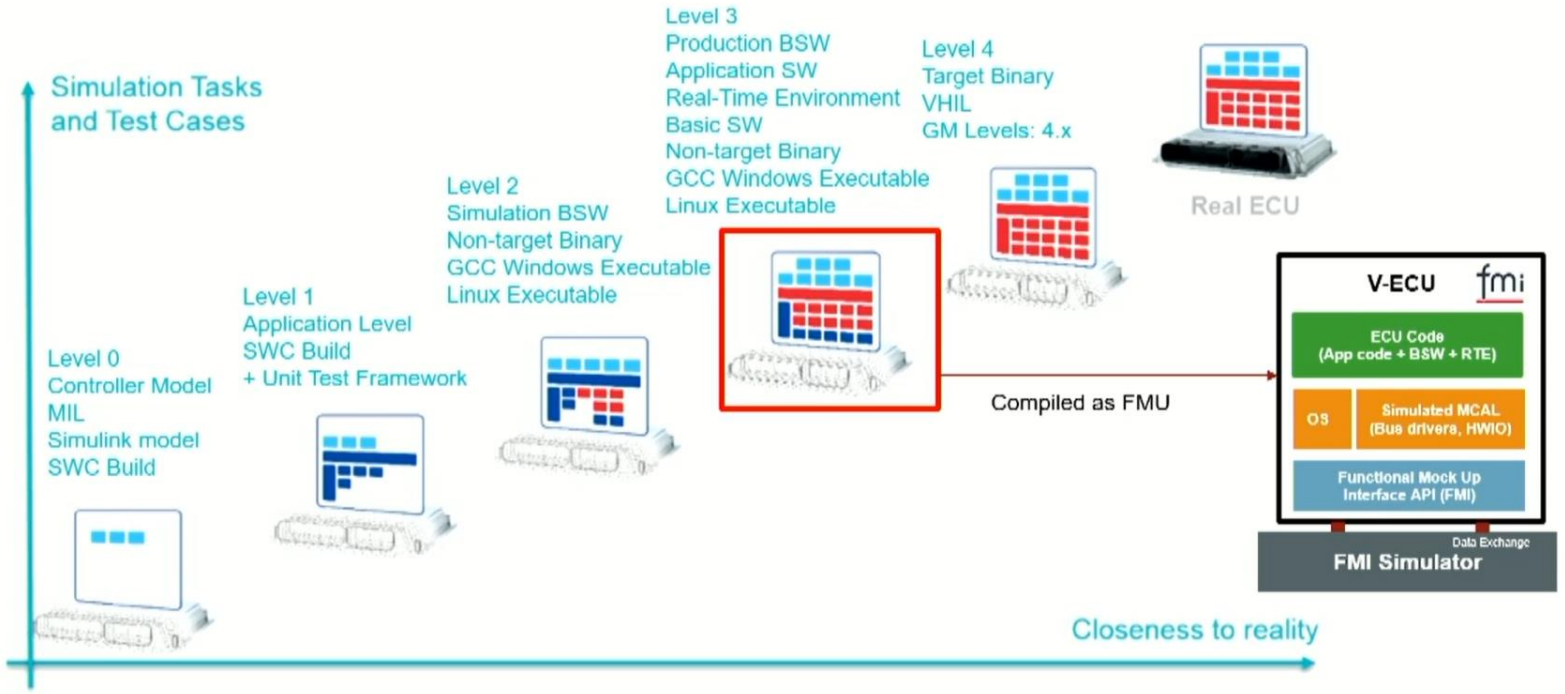
- Simulink® as an **integration platform**
- **Co-Exist with multiple integration platforms**
- bring multiple virtual ECUs as **FMUs (or) Model (or) C/C++ Code**.



General Motors Cuts Testing Time in Half by Simulating E-Drive System

Approach Achieves 95% of Performance Targets Before Hardware Availability

Virtual ECU Overview



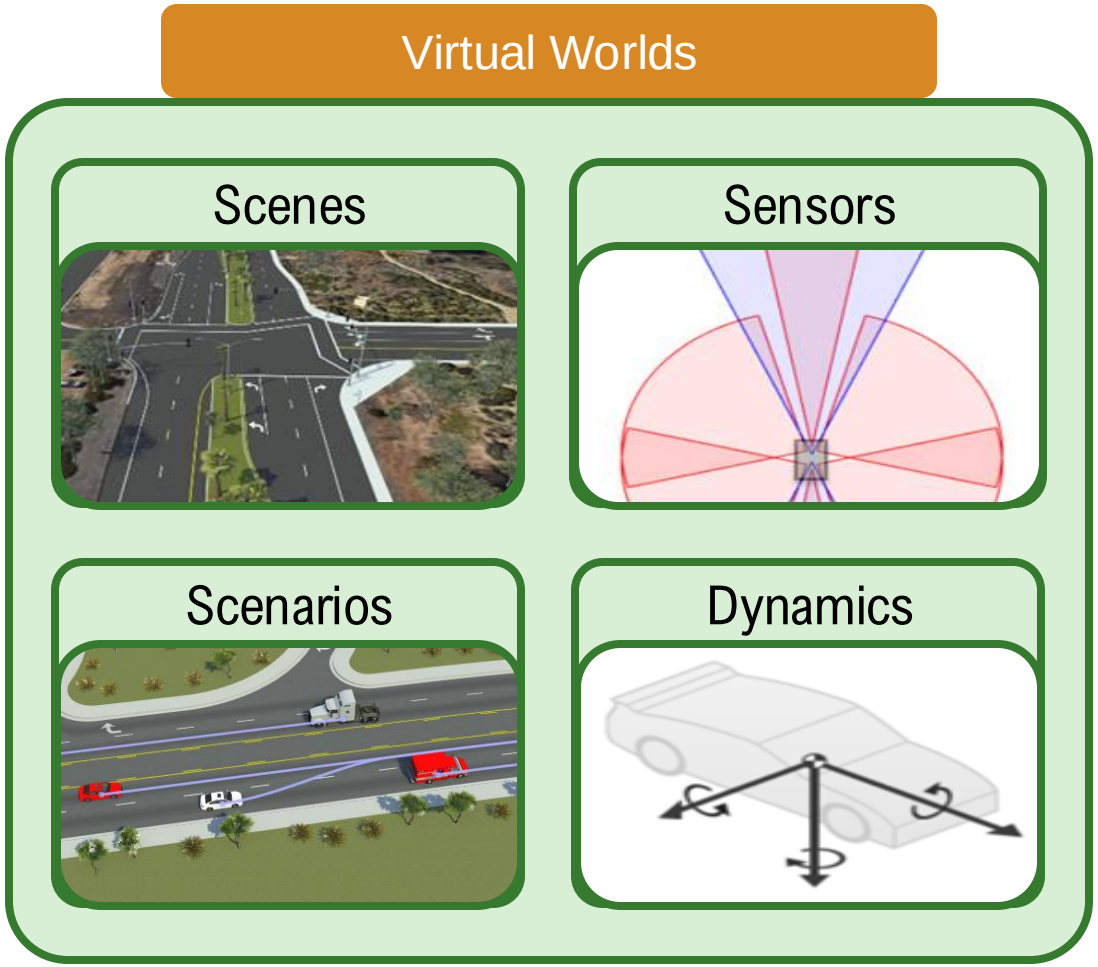
> "The...major win for us is having Simscape plant models give us the right mix of the fidelity that we need to do the calibration work."
>
> — *Srinivas Naveen, GM*

Key Outcomes

- Simulink enables running high-fidelity models for software and controls
- Simscape allows engineers to calibrate models before applying them to physical hardware
- Models developed with MATLAB® and Simulink enable researchers to cut physical dynamometer testing time in half while running near-real-time simulations using a standard CPU

Link

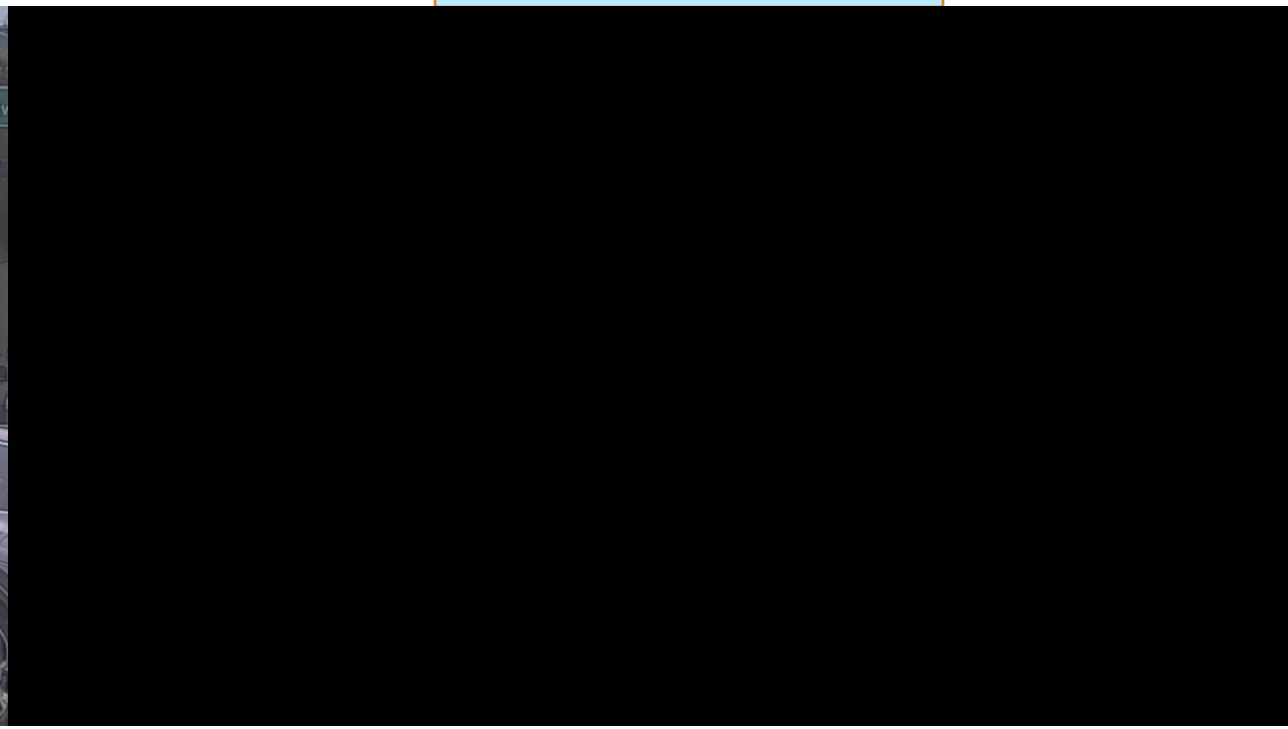
Virtual Validation- Requirements



Creating scenarios from real world data

Sensor data

Virtual Scenario



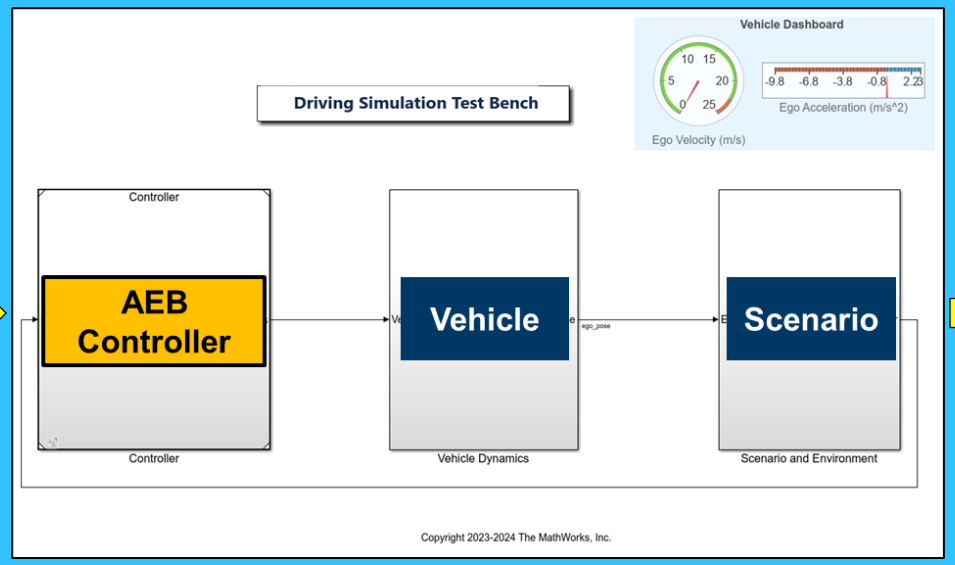
Standard based testing given certification requirements

Test Suite for Euro NCAP® Protocols

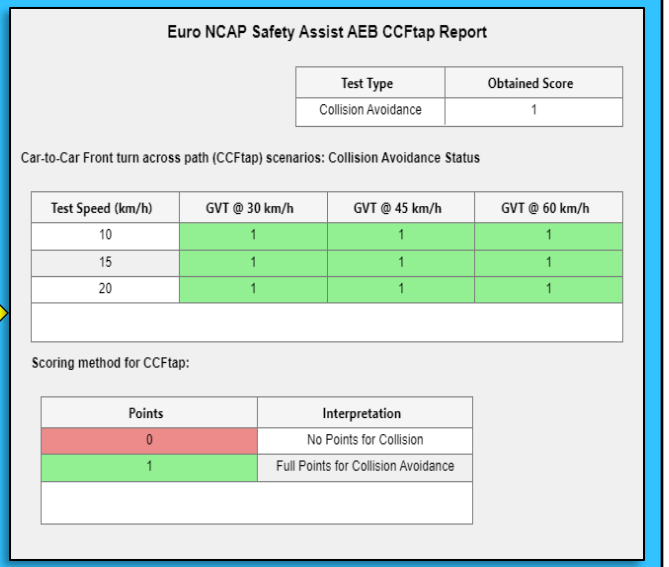
Euro NCAP Scenarios

Test Name	Road Type	#
Car-To-Car Rear Stationary (CCRs)	Straight	75
Car-To-Car Rear Moving (CCRm)	Straight	55
Car-To-Car Rear Braking (CCRb)	Straight	4
Car-to-Car Front Turn-Across-Path (CCFtap)	Junction	9
Car-to-Car Crossing Straight Crossing Path (CCCscp)	Junction	25
Car-to-Car Front Head-On Straight (CCFhos)	Straight	2
Car-to-Car Front Head-On Lane change (CCFhol)	Straight	2

AEB Test Bench



Euro NCAP Report



Euro NCAP Safety Assist AEB CCFtap Report

Test Type	Obtained Score
Collision Avoidance	1

Car-to-Car Front turn across path (CCFtap) scenarios: Collision Avoidance Status

Test Speed (km/h)	GVT @ 30 km/h	GVT @ 45 km/h	GVT @ 60 km/h
10	1	1	1
15	1	1	1
20	1	1	1

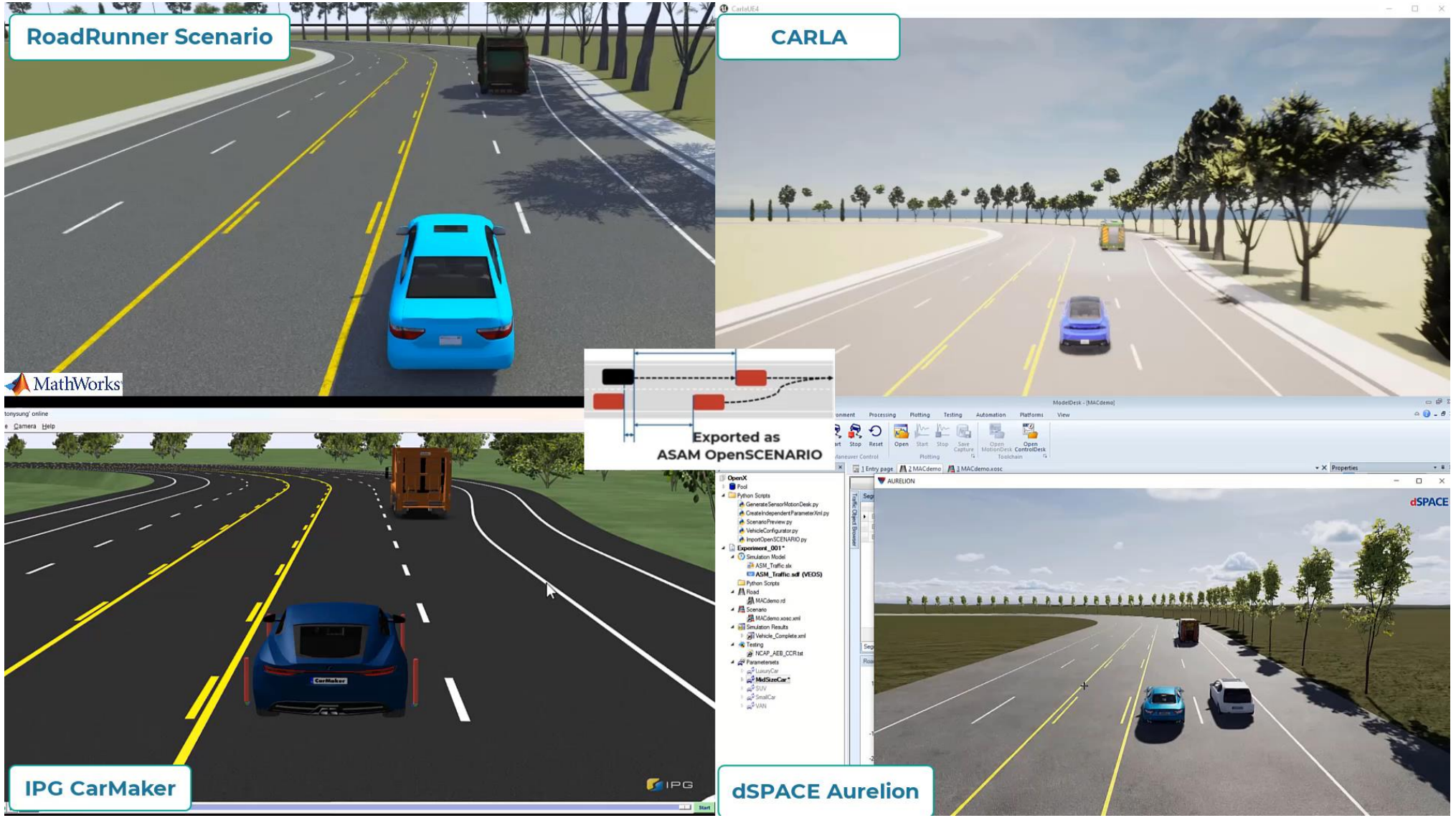
Scoring method for CCFtap:

Points	Interpretation
0	No Points for Collision
1	Full Points for Collision Avoidance

Scenario

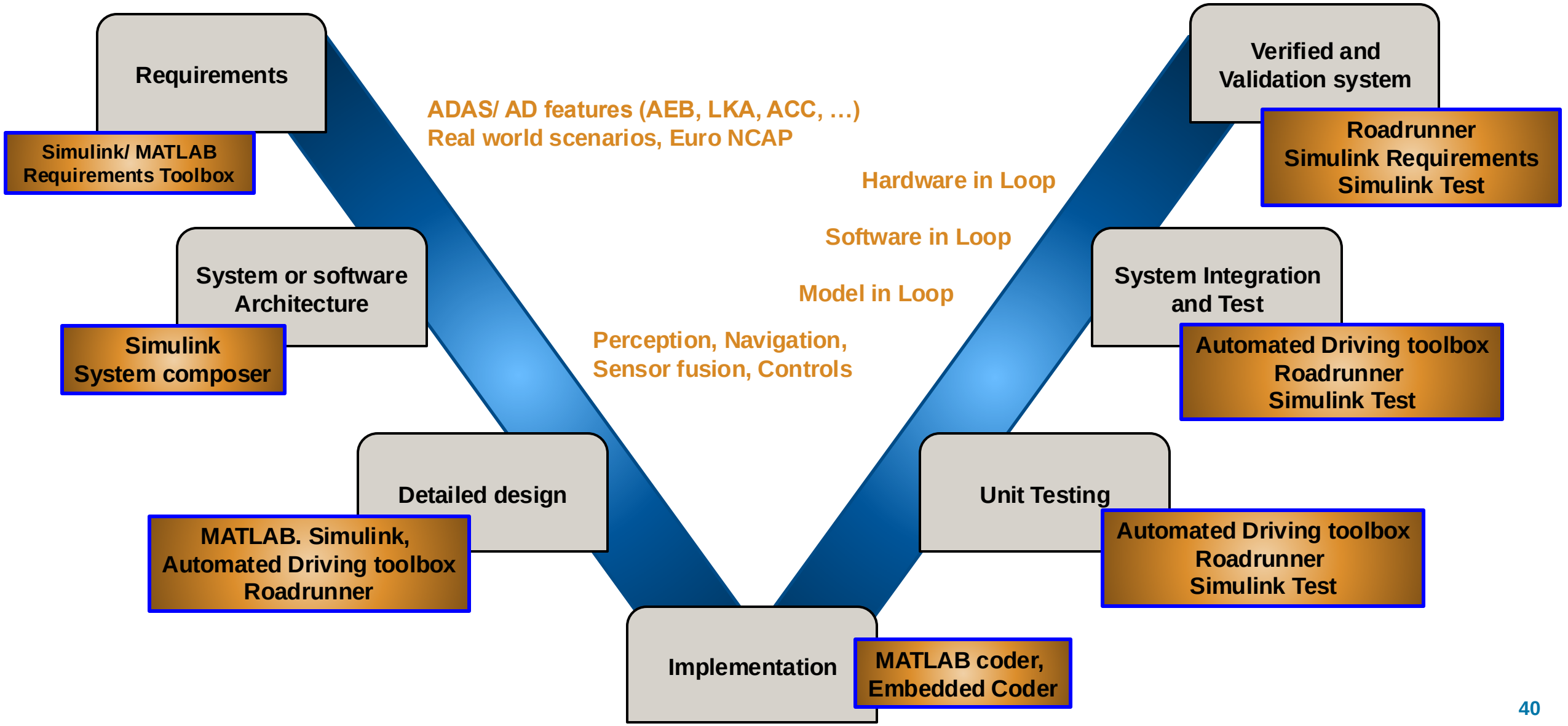
Test manager

Exporting from RoadRunner Scenarios to ASAM OpenSCENARIO for multiple driving simulators



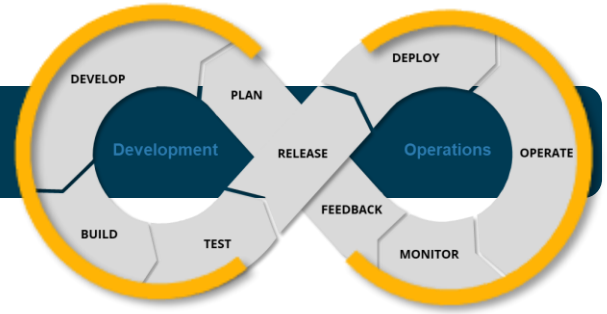
Model-Based Design for Automated Driving

Systematic use of models throughout the development process



Goals of DevOps and Software Factory

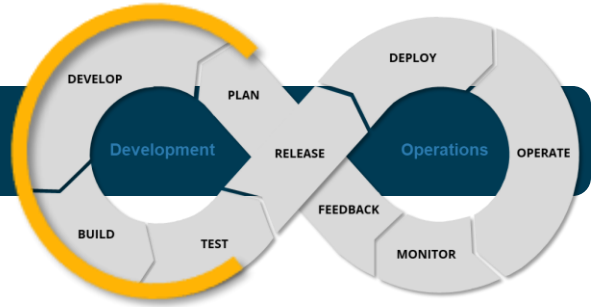
DevOps

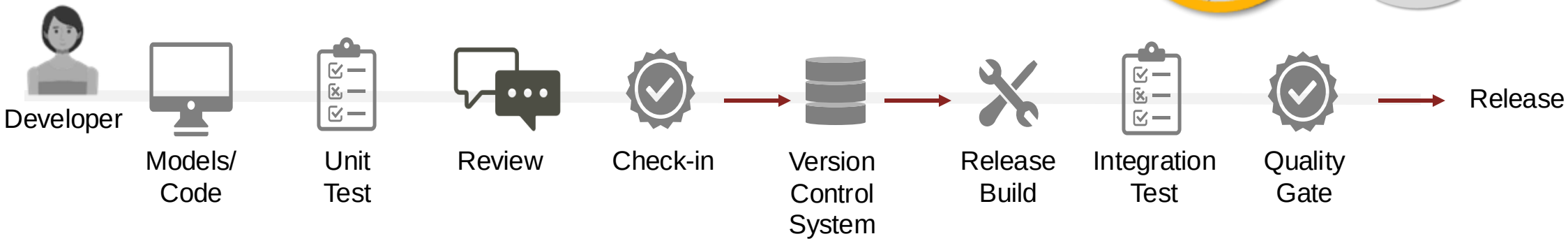


Unites agile development with reliable operations

Goal: **Reduce the time** between committing a change and placing it in production, **while ensuring high quality**

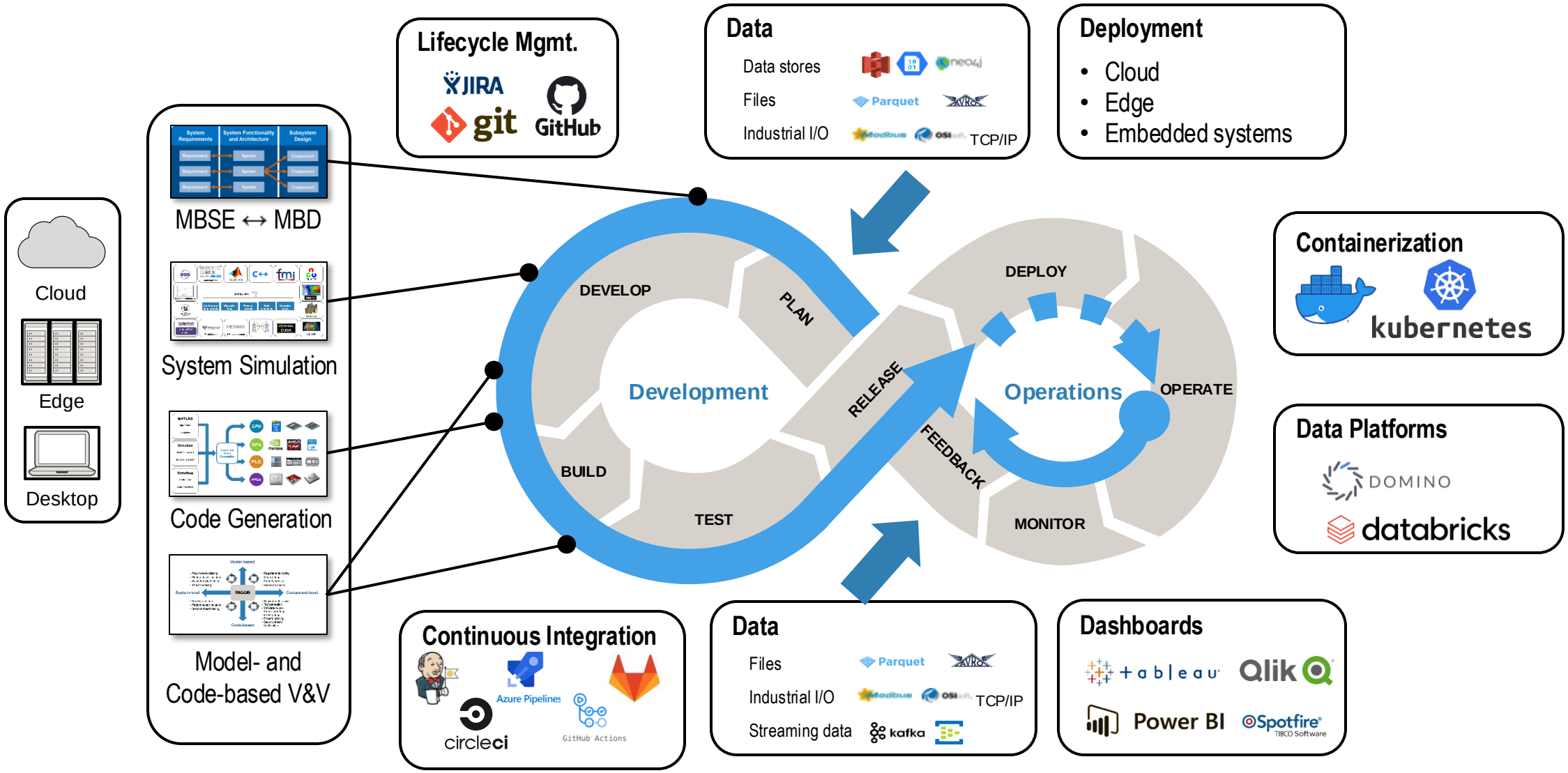
Software Factory



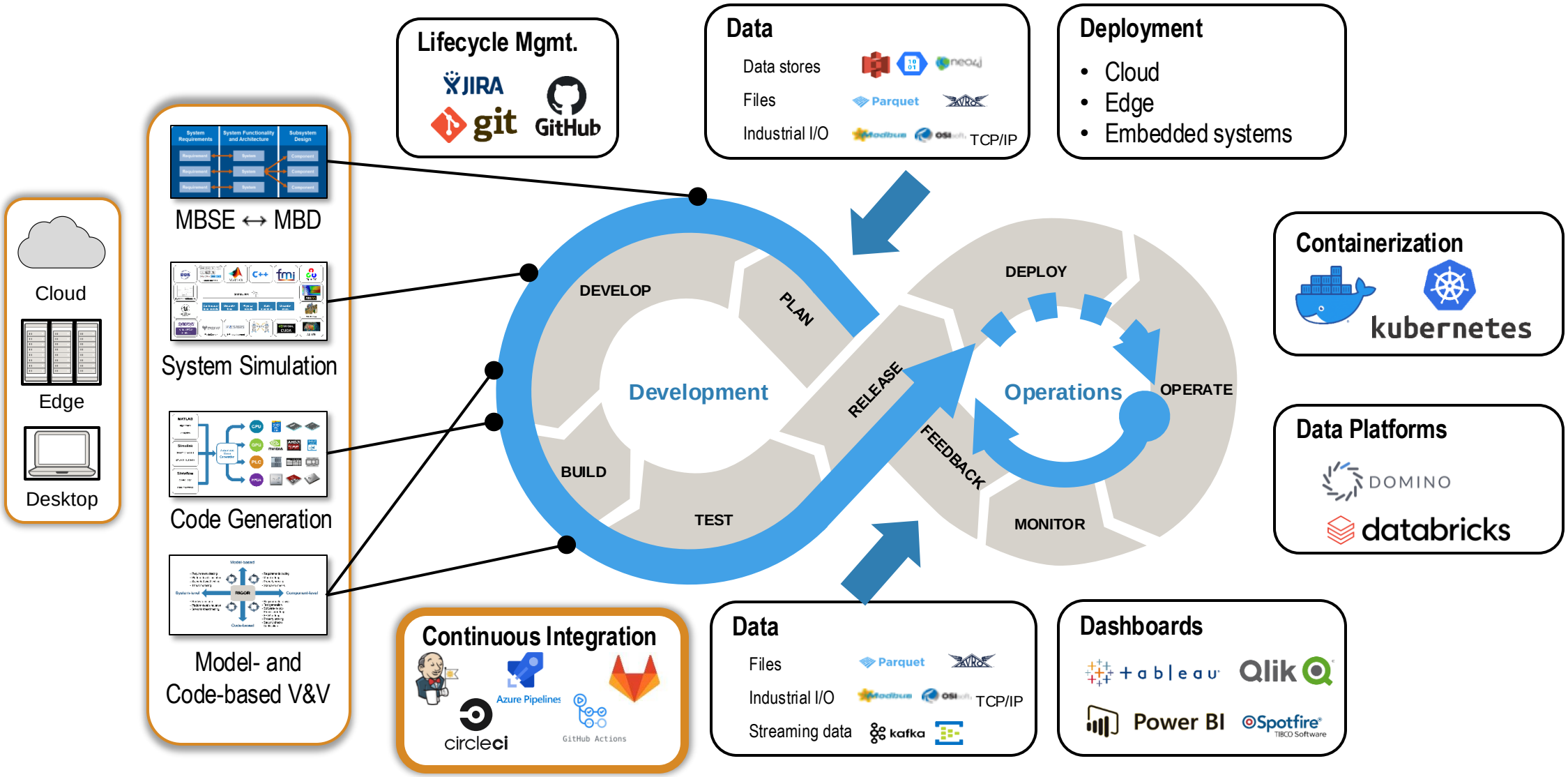


Goal**: Repeatability, Faster delivery, Higher quality**

DevOps building blocks for Embedded Production SW



Continuous Integration for embedded production SW



Automated testing and codegen via Continuous Integration

Challenge:

- Enable multiple engineers to simultaneously develop features in parallel, but can share and sync with the other colleagues
- Test application code while still in development, thereby creating a “fix-as-you-go” workflow
- Automate large scale testing and code generation when development branches are merged to main/release branch

Solution:

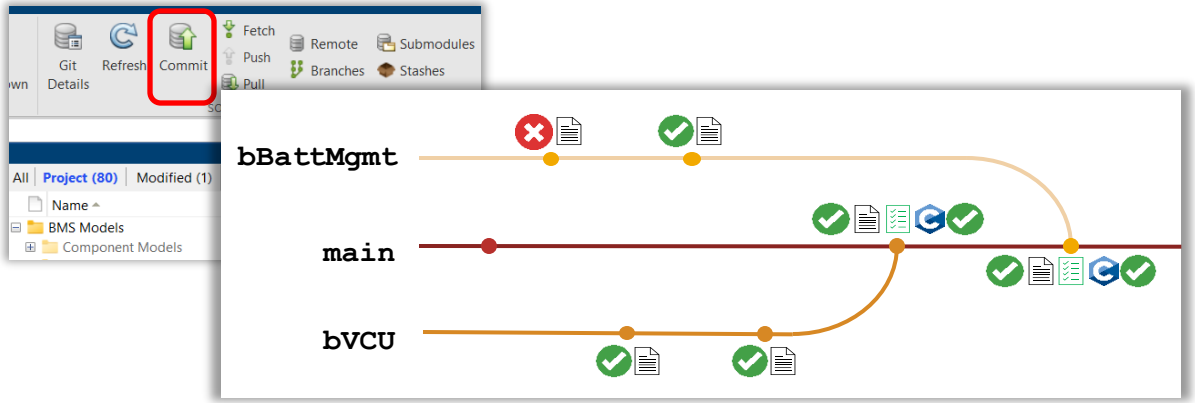
Virtualize

Componentize models and place them under source control

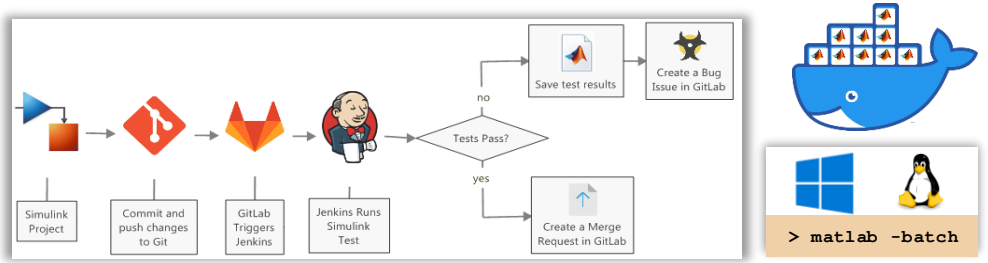
- Test at model level, application software level, and conduct MISRA C checks

Scale

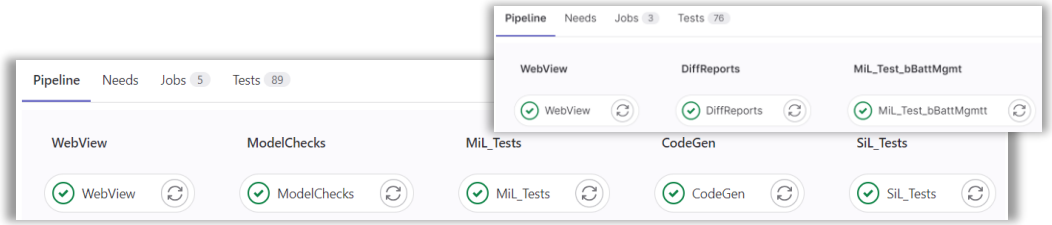
Setup non-interactive MATLAB on runner computer(s); and perform automated tests, codeGen, and report authoring tasks



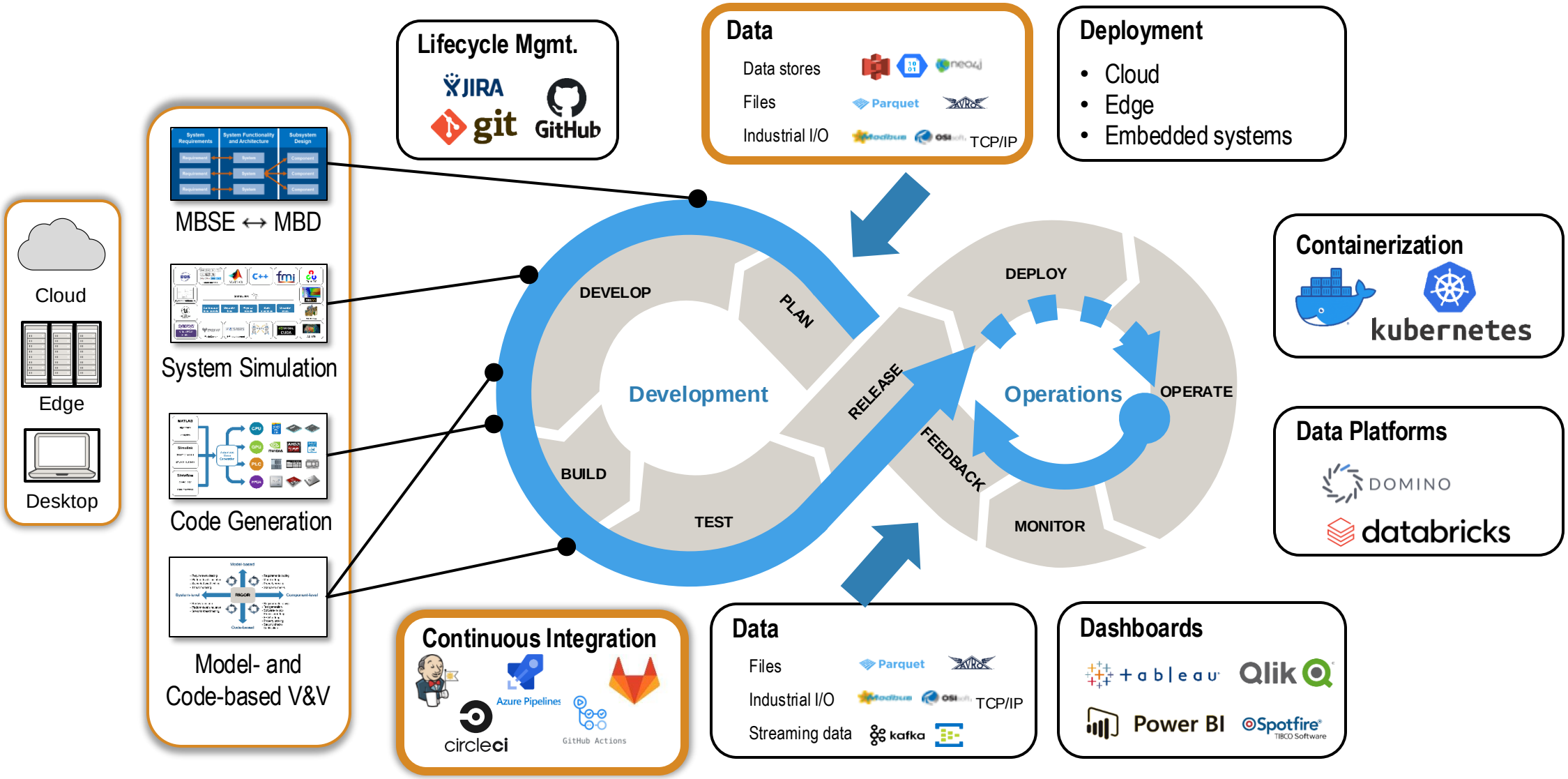
Commit changed models/tests to git using Projects



Automated pipelines can be configured to run in a variety of environments, to meet your scaling needs



DevOps building blocks for Embedded Production SW

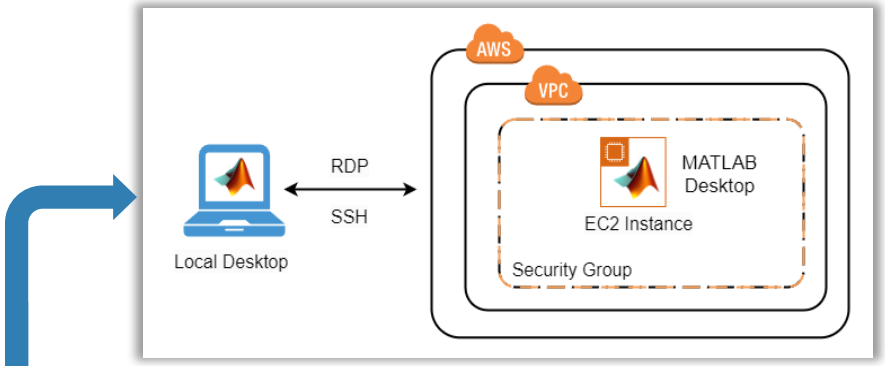


Scaling up with `parsim` on the Cloud

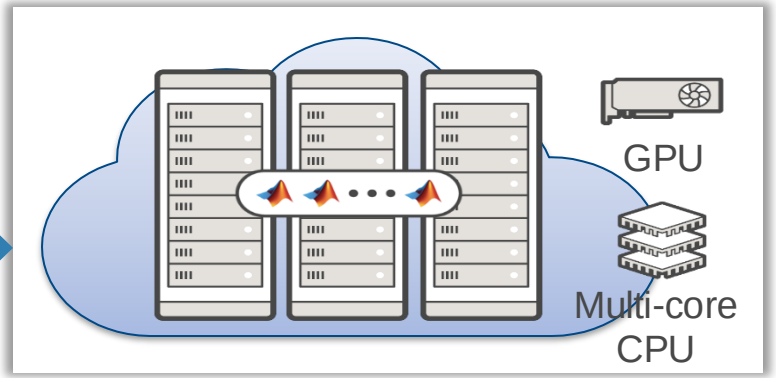
Different cloud computing resources for different jobs


```
simOut = parsim(in)
```

Parallel Computing Toolbox
MATLAB



MATLAB on AWS EC2



MATLAB Parallel Server

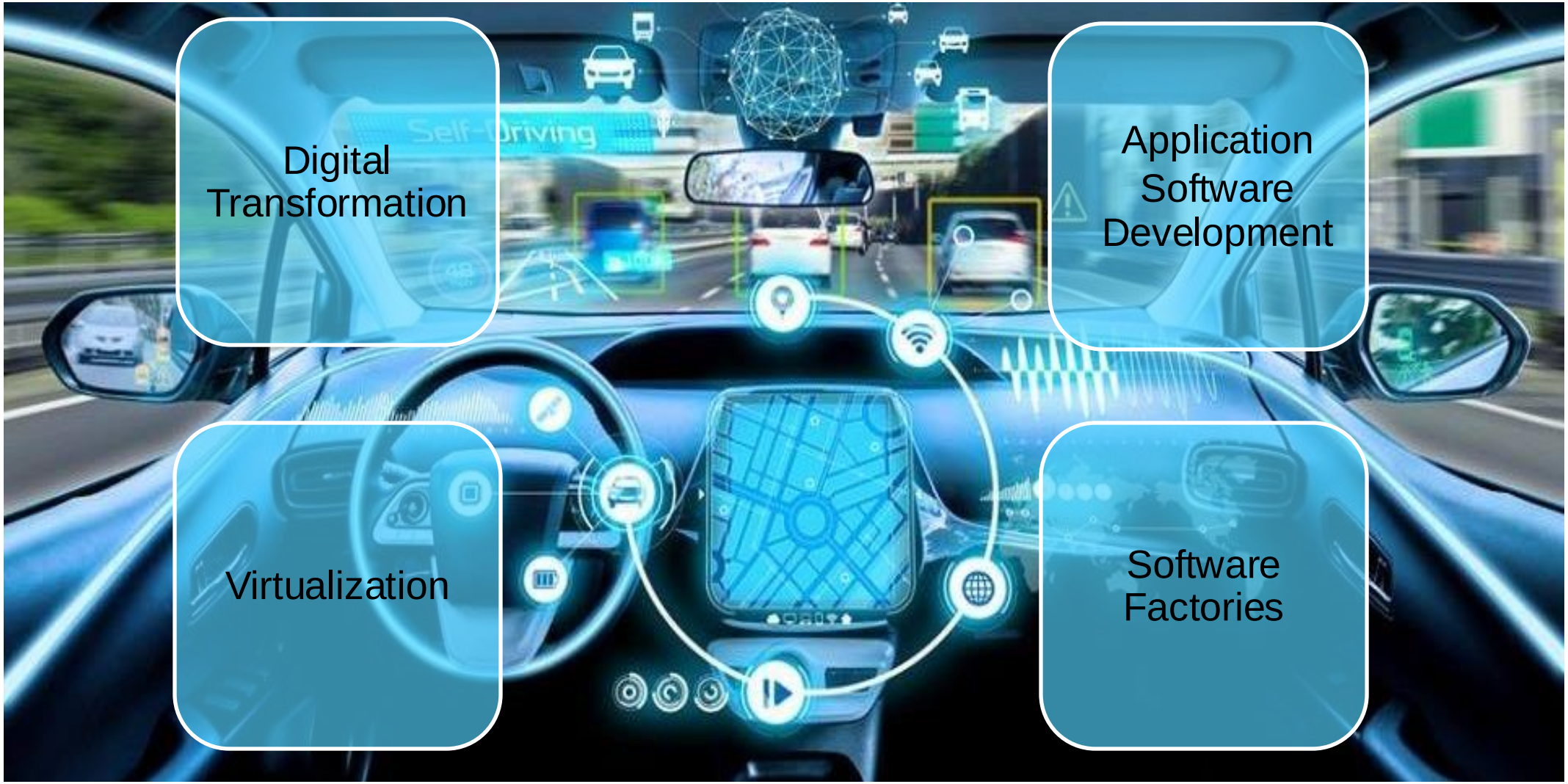
Running 1352 Simulations
~ **18 hours** in series
~ **5.2 hours** on Quadcore Laptop
~ **59 mins** on an m5.12xlarge EC2 instance, 24 core

Worker Machine = m5.12xlarge (24 cores)

Running 1352 Simulations
~ **22.7 mins** on 5 Worker machines, 120 cores
~**17 mins** on 10 Worker machines, 240 cores

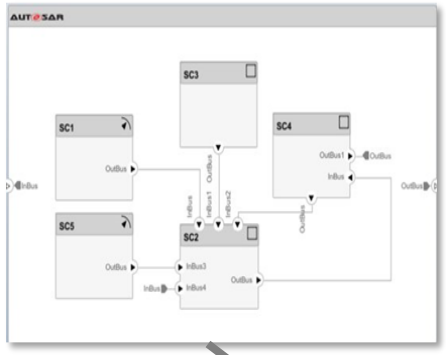
Learn more: MATLAB on AWS, MathWorks Reference Architecture, MathWorks CloudCenter

Our discussion

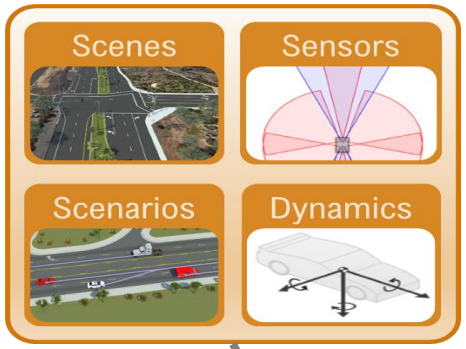


Key Takeaway:
MathWorks Solutions Accelerate Software-Defined Vehicle Development

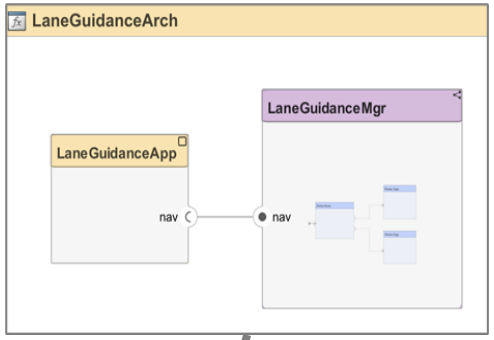
Shift-left integration and validation with simulation



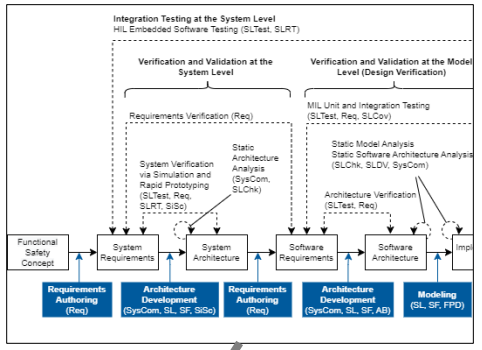
Virtual vehicle and scenario for automated driving



Flexible generation of SOA and signal-based software



Reference workflow for safety and security



Automate and integrate model-based capabilities with CI