KPIT

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

MathWorks
Automotive Conference
2023

Rajat Shetty, Jayprakash Dubey

Independent software integration partner bringing scale and dependability

to build and integrate software features to accelerate the journey from prototype to production

10+

Mn vehicles on road with KPIT software

500+

Production programs experience

25+

OEM/Tier-1's count us as strategic partners for next gen mobility

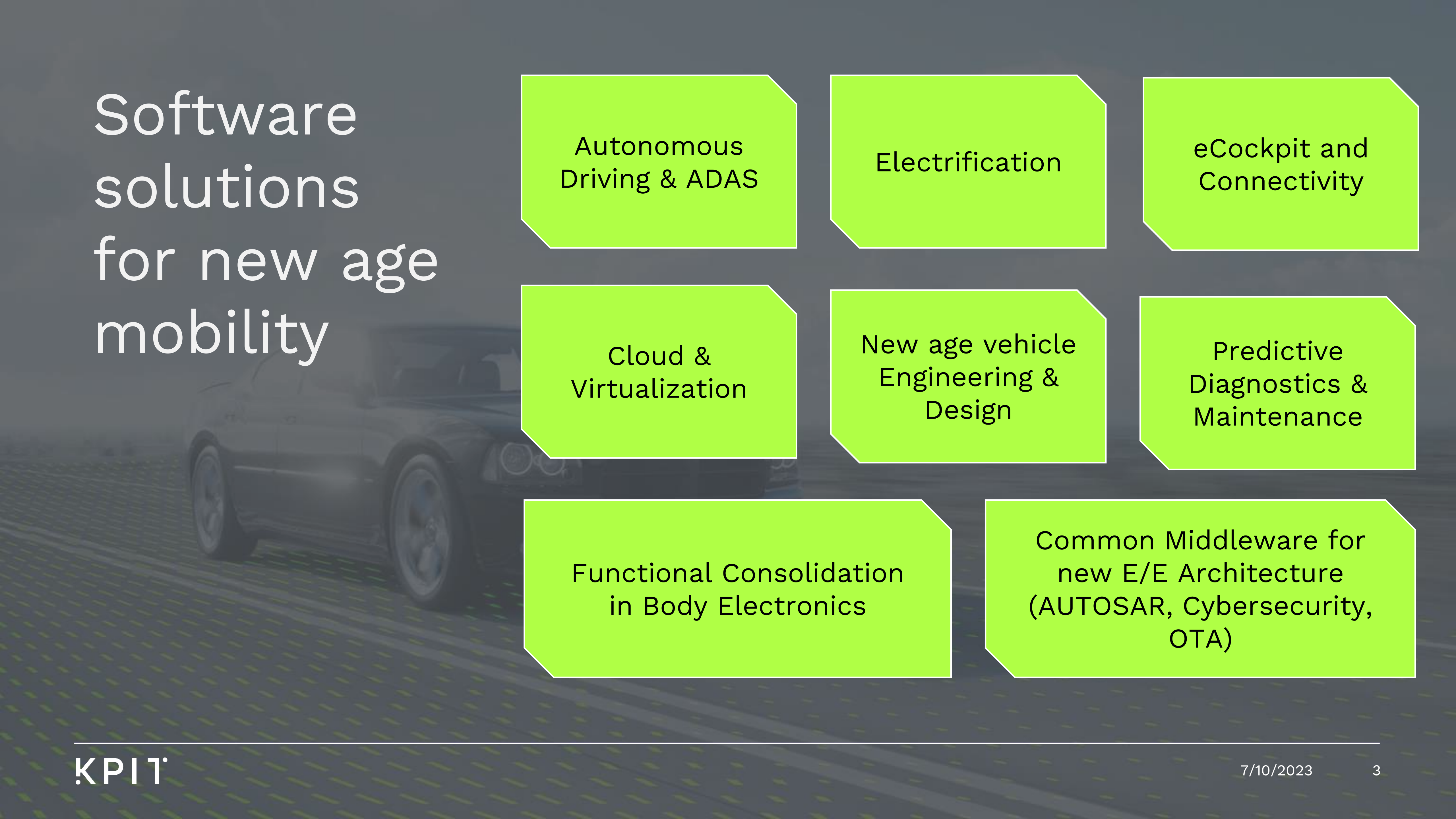
75+

platforms, tools & accelerators

Team of highly talented chief architects, domain experts, designers and engineers

KPIT

Software solutions for new age mobility



- Autonomous Driving & ADAS
- Electrification
- eCockpit and Connectivity
- Cloud & Virtualization
- New age vehicle Engineering & Design
- Predictive Diagnostics & Maintenance
- Functional Consolidation in Body Electronics
- Common Middleware for new E/E Architecture (AUTOSAR, Cybersecurity, OTA)

Contents

Arbitration in AD/ADAS vehicles

At all events, arbitration is more rational, just, and humane than the resort to swords.

- Richard Cobden

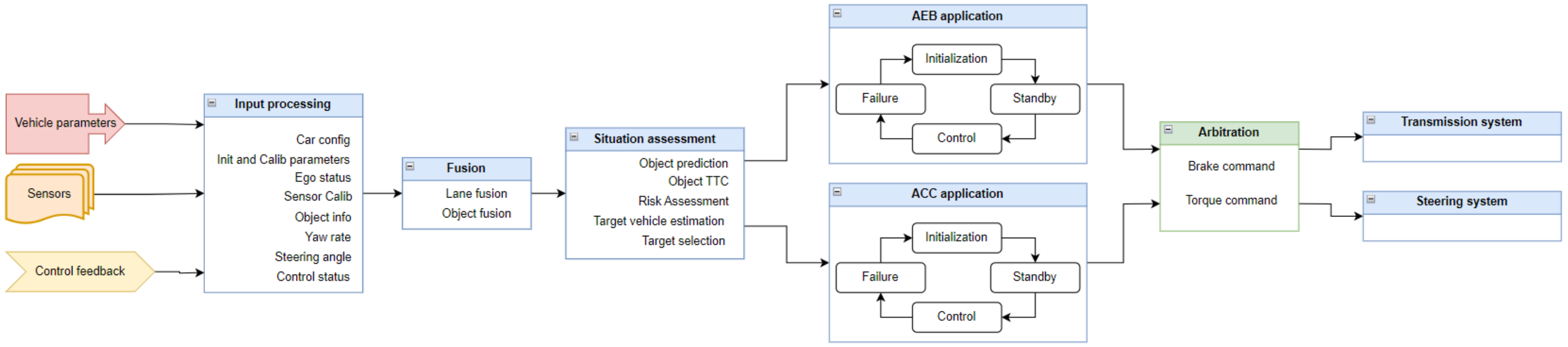
Arbitration in AD/ADAS vehicle is responsible for decision-making between Lateral control, Longitudinal control or hybrid control. The decision made leads to tactical & strategic planning of the vehicle manoeuvre.

At any point of time, Arbitration module shall consider the situation in-hand to make the decisions. Arbitration scheme can be chosen based on various behaviour choices;

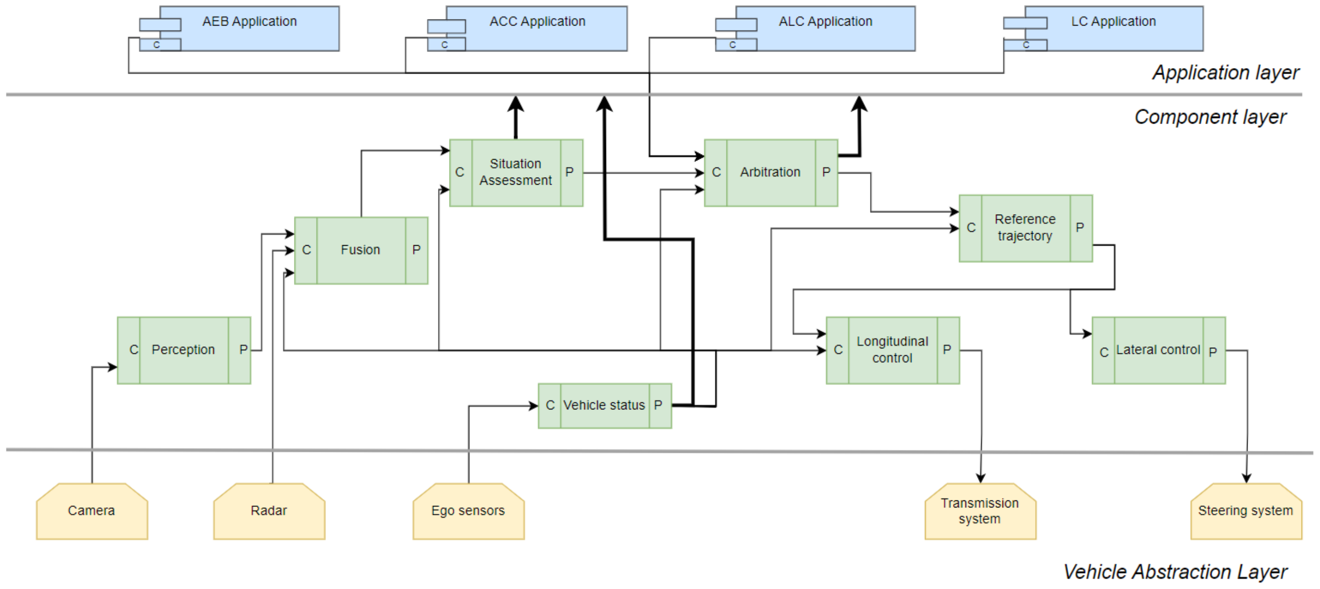
- Priority
- Pre-defined order
- Cost-based rules

Arbitration comparison

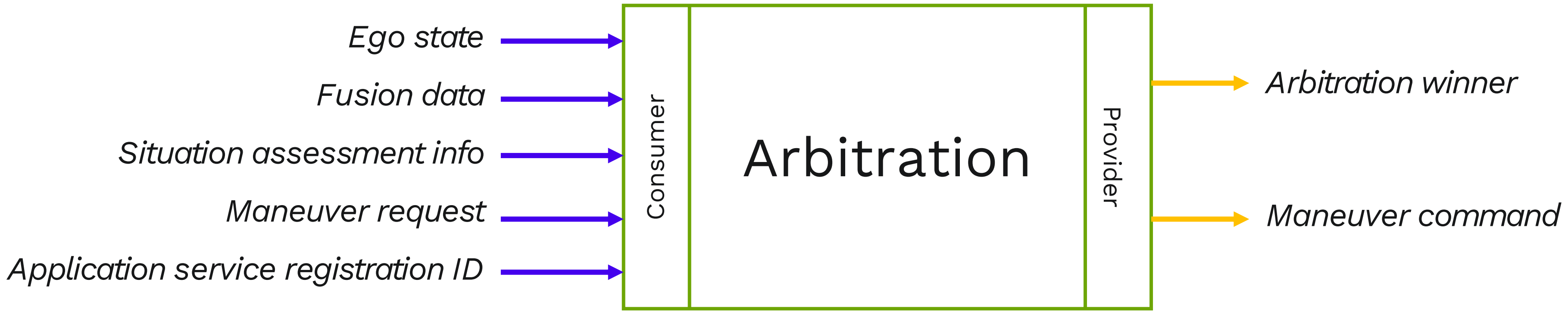
Legacy architecture



Service Oriented Architecture



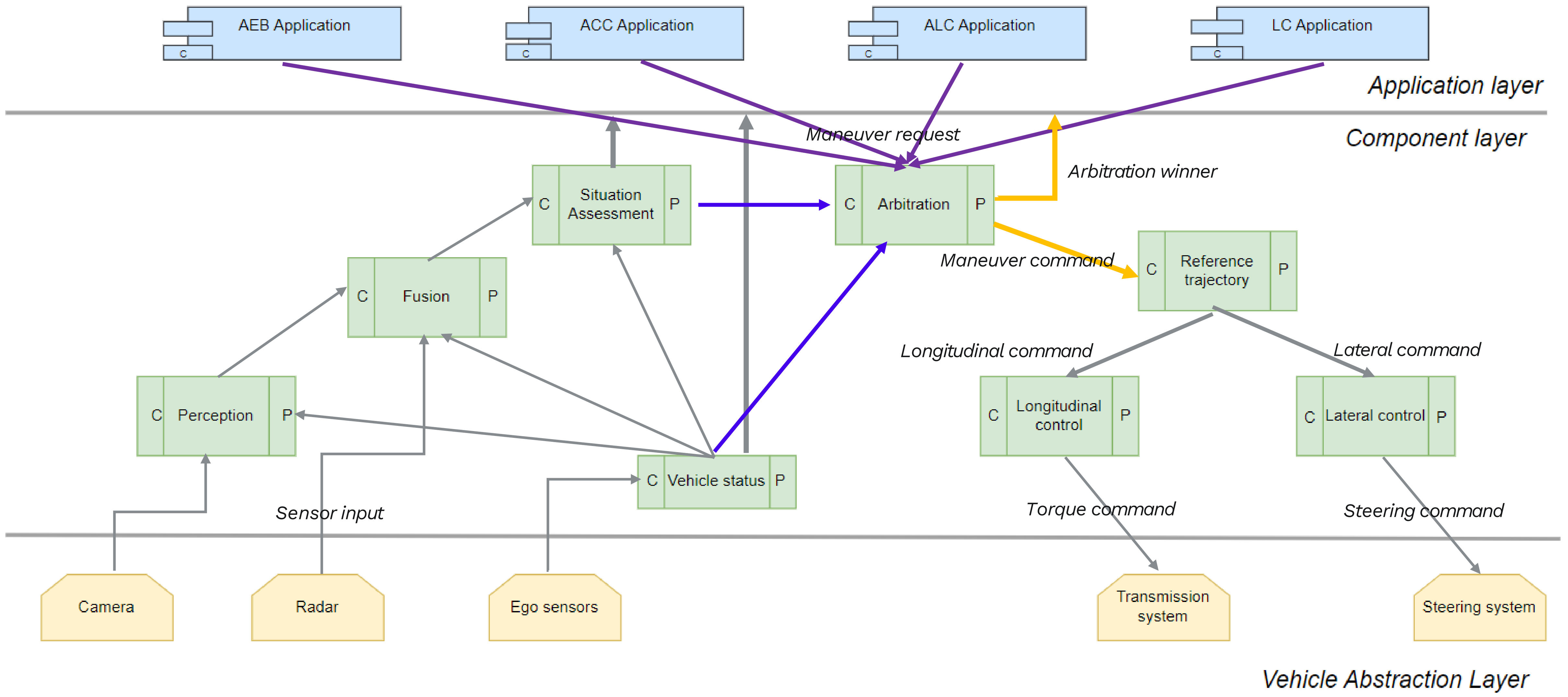
Interface strategy for Arbitration application



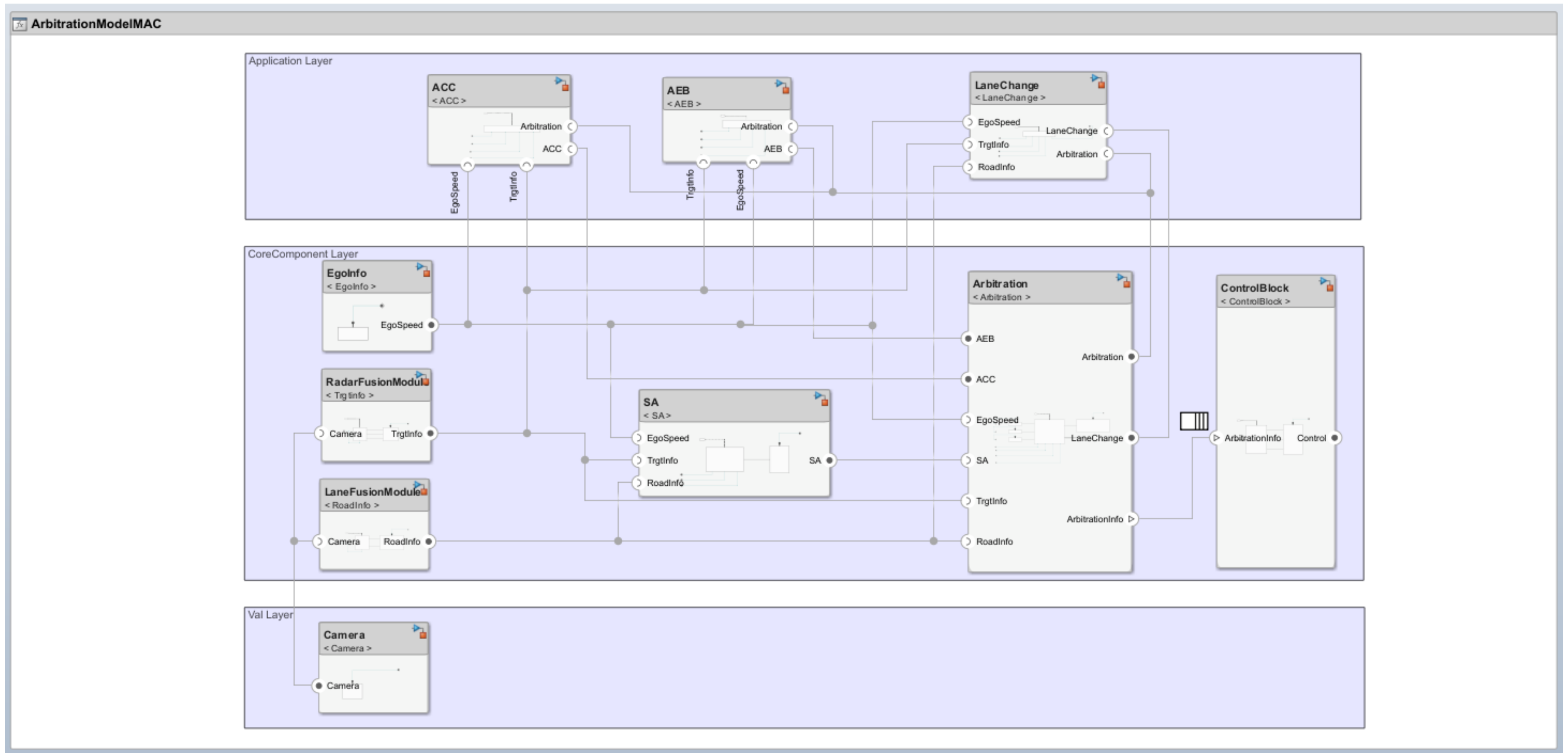
In addition to **synchronizing of all applications** communicating to the arbitration module, the following strategy was considered;

- The applications available is made aware only through **application registry**. This shall ensure cyber security and the safety level of the application.
- The application ID is indicated by a **unique identifier for each application** (AAACCSSTT) which comprises of;
  1. Application itself (AAA – as assigned by the service registry)
  2. Relevant control access (CC – Lat/Long/Hybrid)
  3. ASIL safety level (SS) of the application
  4. Cycle time (TT) in ms of the application
- The **maneuver requested** message needs to be **standardized** across all applications.
- The **arbitration winner** provides the unique identifier of the application which won as **acknowledgement**.

Design flow of SOA Arbitration



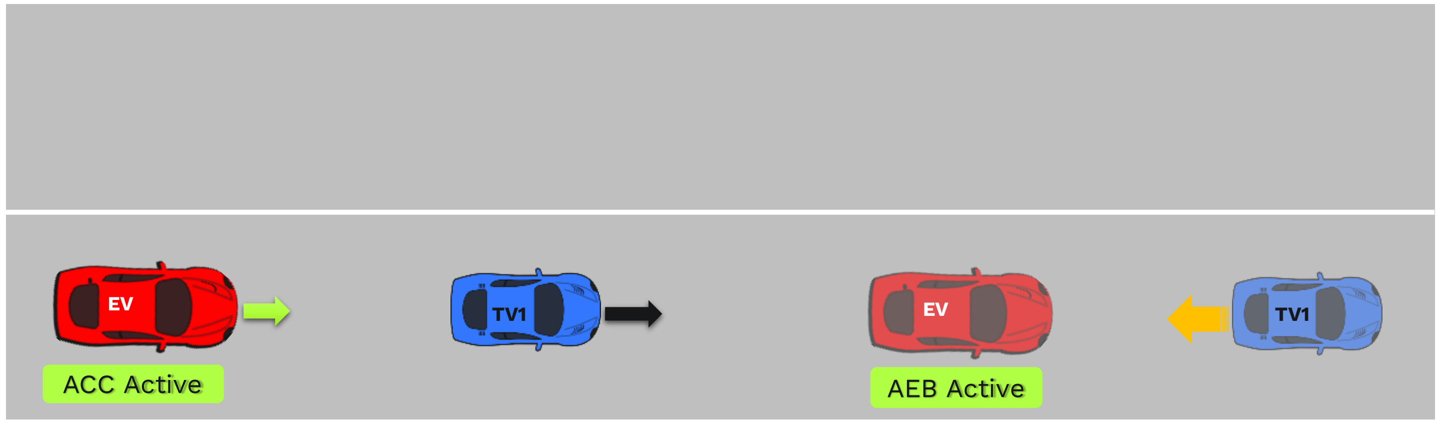
SOA Architecture Matlab framework



- Egoinfo, RadarFusionModule and LaneFusion use Events to send data across to other Software Components
- Application to Arbitration is a Fire and Forget
- Arbitration to ControlBlock has a request response relation.
- Arbitration to Applications is an On-Change based trigger Event

Sample scenario explanation

Arbitration between ACC and AEB

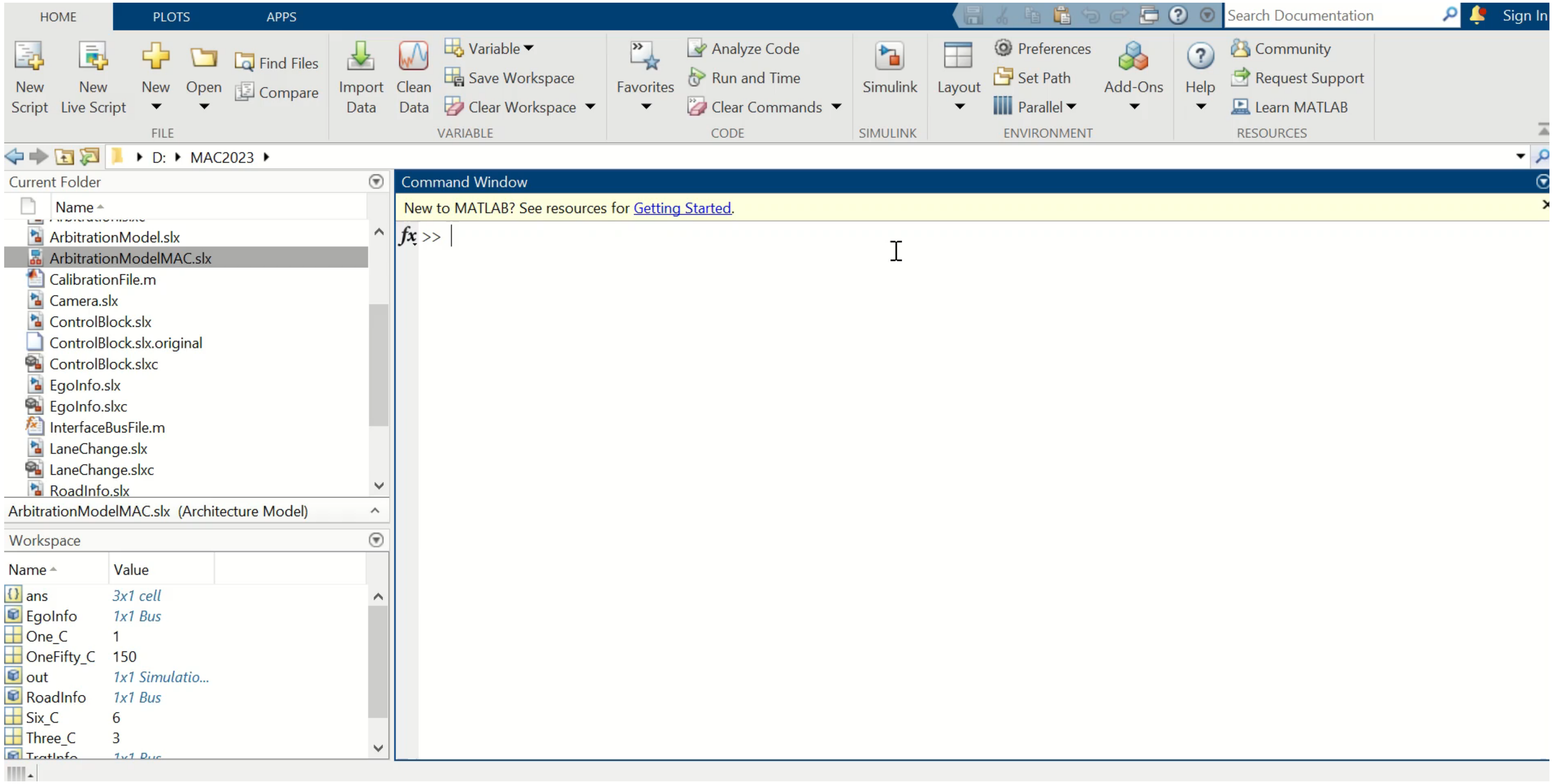


Description: -

- 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.

Demo

SOA simulation in MATLAB using System Composer



New to MATLAB? See resources for Getting Started.

```
fx >>
```

ArbitrationModel.slx
ArbitrationModelMAC.slx
CalibrationFile.m
Camera.slx
ControlBlock.slx
ControlBlock.slx.original
ControlBlock.slxc
EgoInfo.slx
EgoInfo.slxc
InterfaceBusFile.m
LaneChange.slx
LaneChange.slxc
RoadInfo.slx

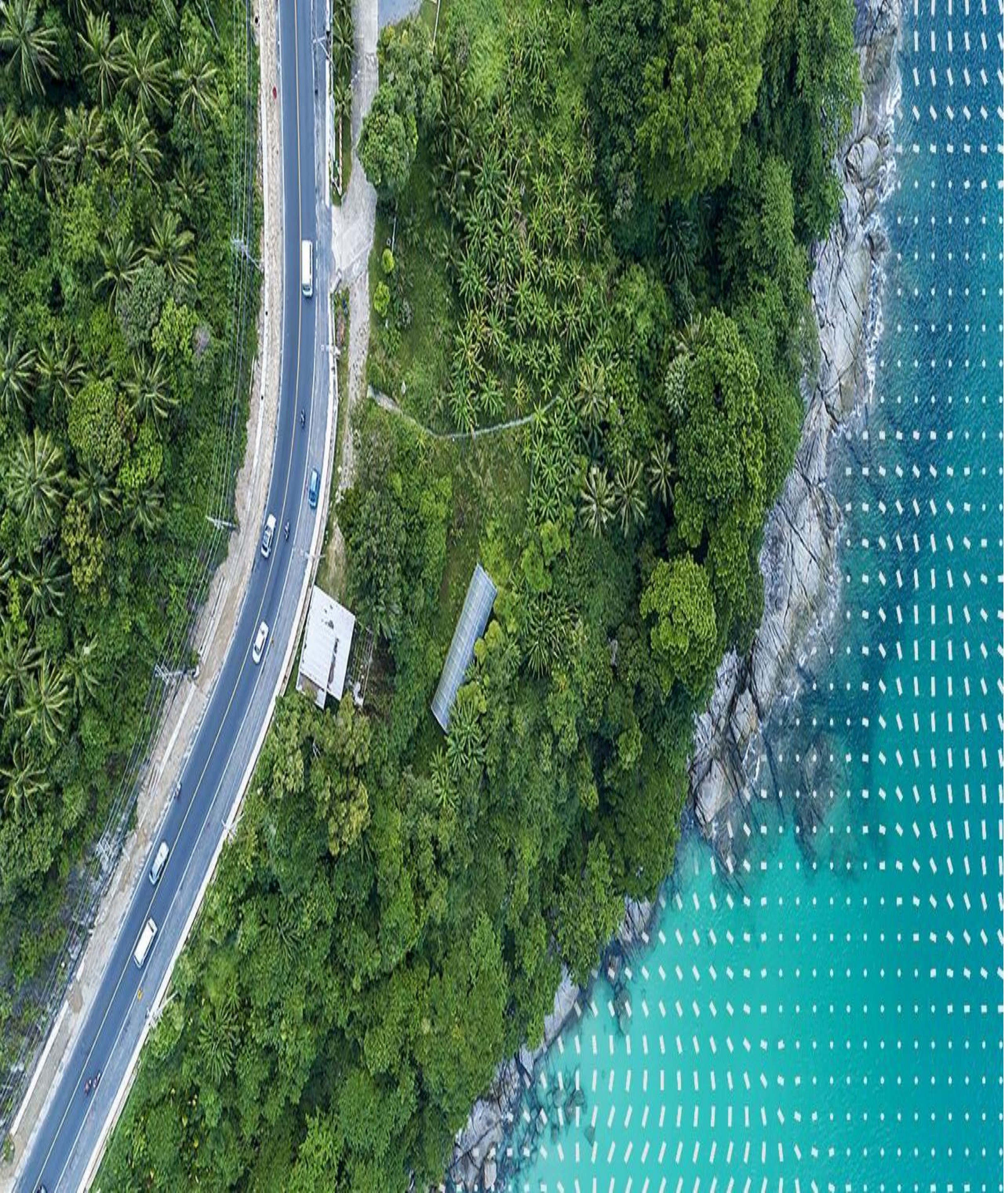
ArbitrationModelMAC.slx (Architecture Model)

Name	Value
ans	3x1 cell
EgoInfo	1x1 Bus
One_C	1
OneFifty_C	150
out	1x1 Simulatio...
RoadInfo	1x1 Bus
Six_C	6
Three_C	3

Advantages over conventional architecture

- **Scalability** : All components are designed to communicate using services resulting in ease for future enhancement. New software components can be designed and incorporated without affecting existing components
- **Re-usability** : Services can be easily discovered and used when a new feature is deployed. A newly developed feature can depend on services provided by existing software components without updating or redeploying the entire software.
- **Bandwidth and memory** requirement for OTA is less as only specific software components need to update.
- **Optimization of redundant softwar**e components between cross-domain. Services could be discovered and used across different automotive domains.
- Running components in **Shadow Mode** in order to test newly deployed version of a software component without affecting the original behaviour or a feature.

Queries ?



THANK YOU