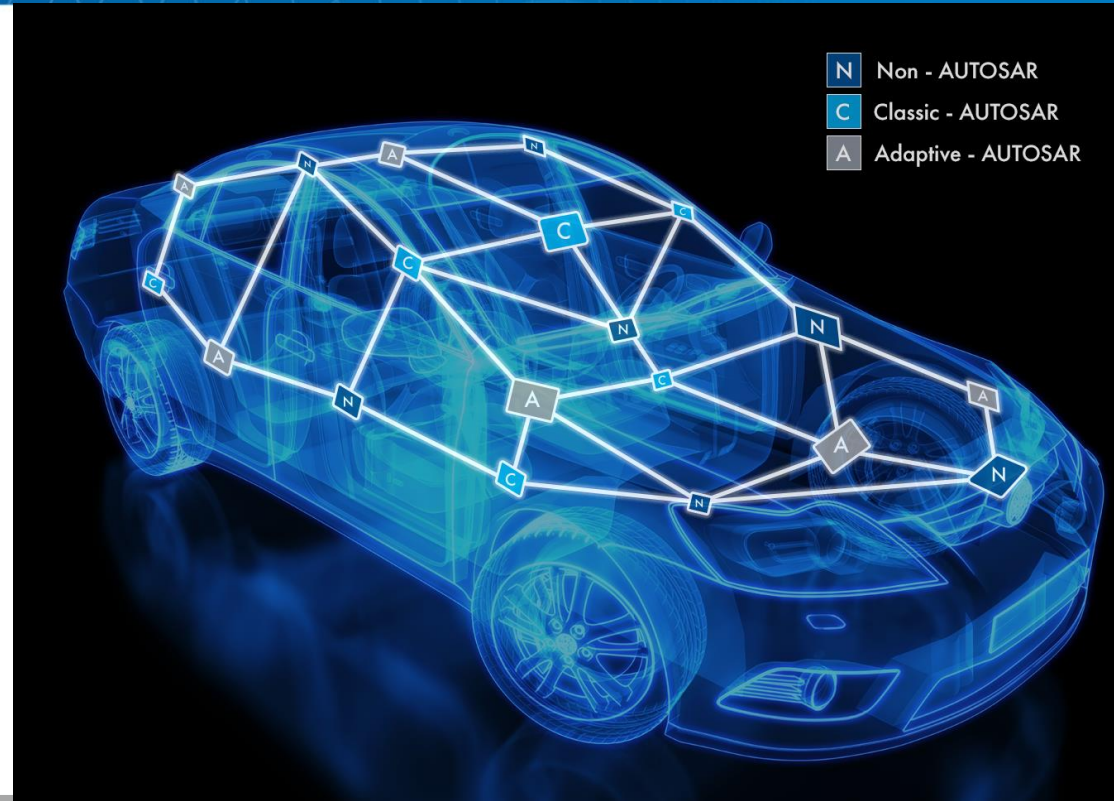


Simulink for AUTOSAR Adaptive

Dr Richard Thompson
Software Engineering Manager



Agenda

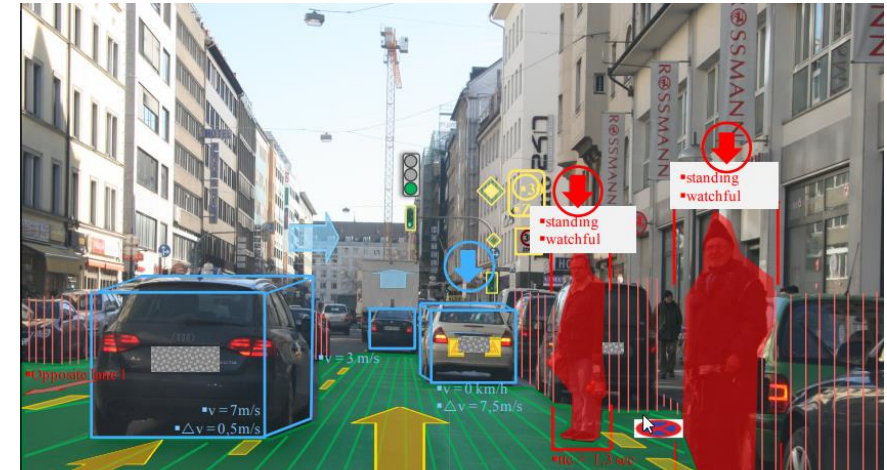
- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
- Additional Resources

Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
- Additional Resources

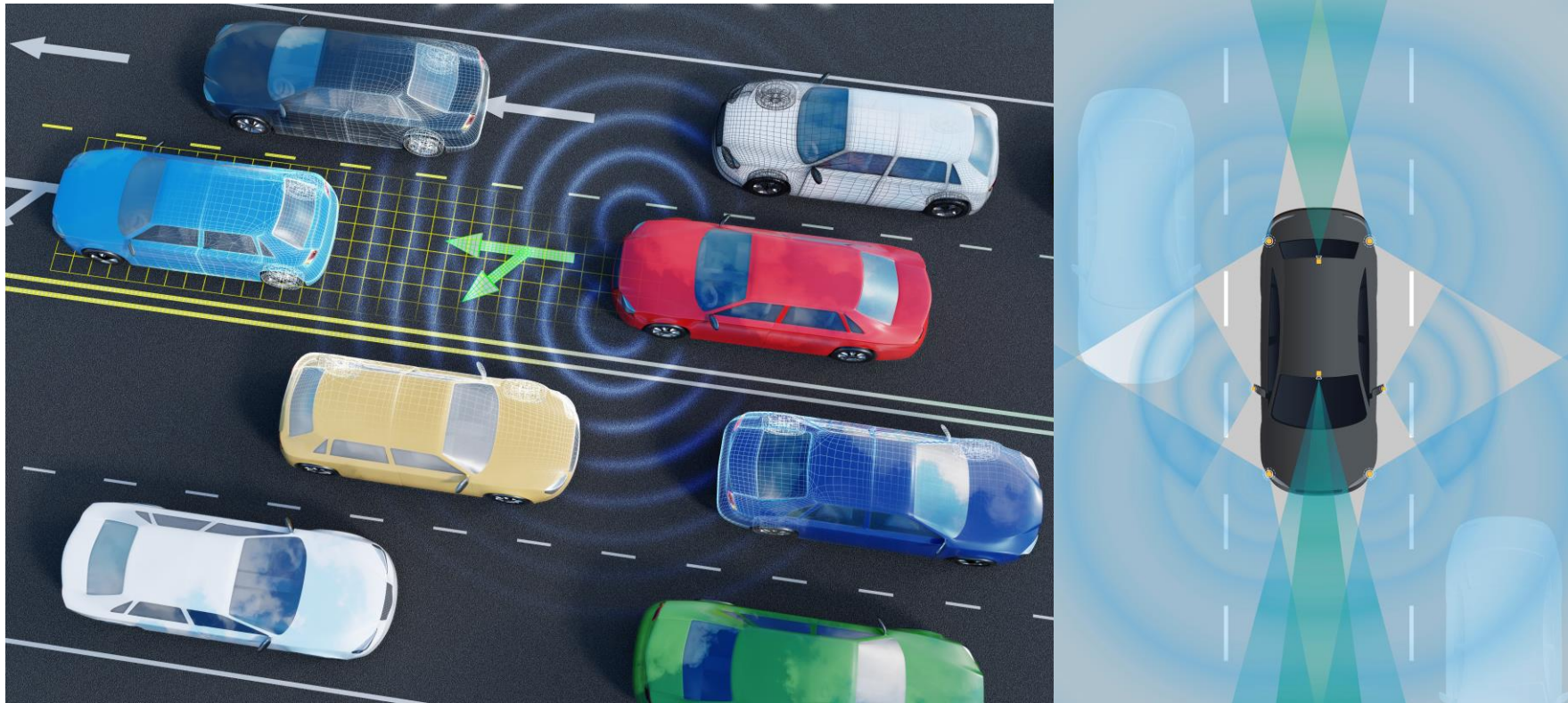
AUTOSAR Classic is already on the road

- [BMW](#) - Model-Based Software Development: And OEM's Perspective
- [FCA Global Powertrain Controls](#) - Leveraging MBD, auto-code generation and AUTOSAR to architect and implement an Engine Control Application for series production
- [LG Chem](#) - Developing AUTOSAR and ISO 26262 Compliant Software for a Hybrid Vehicle Battery Management System with Model-Based Design
- [John Deere](#) - Vertical AUTOSAR System Development at John Deere



Motivation for AUTOSAR Adaptive

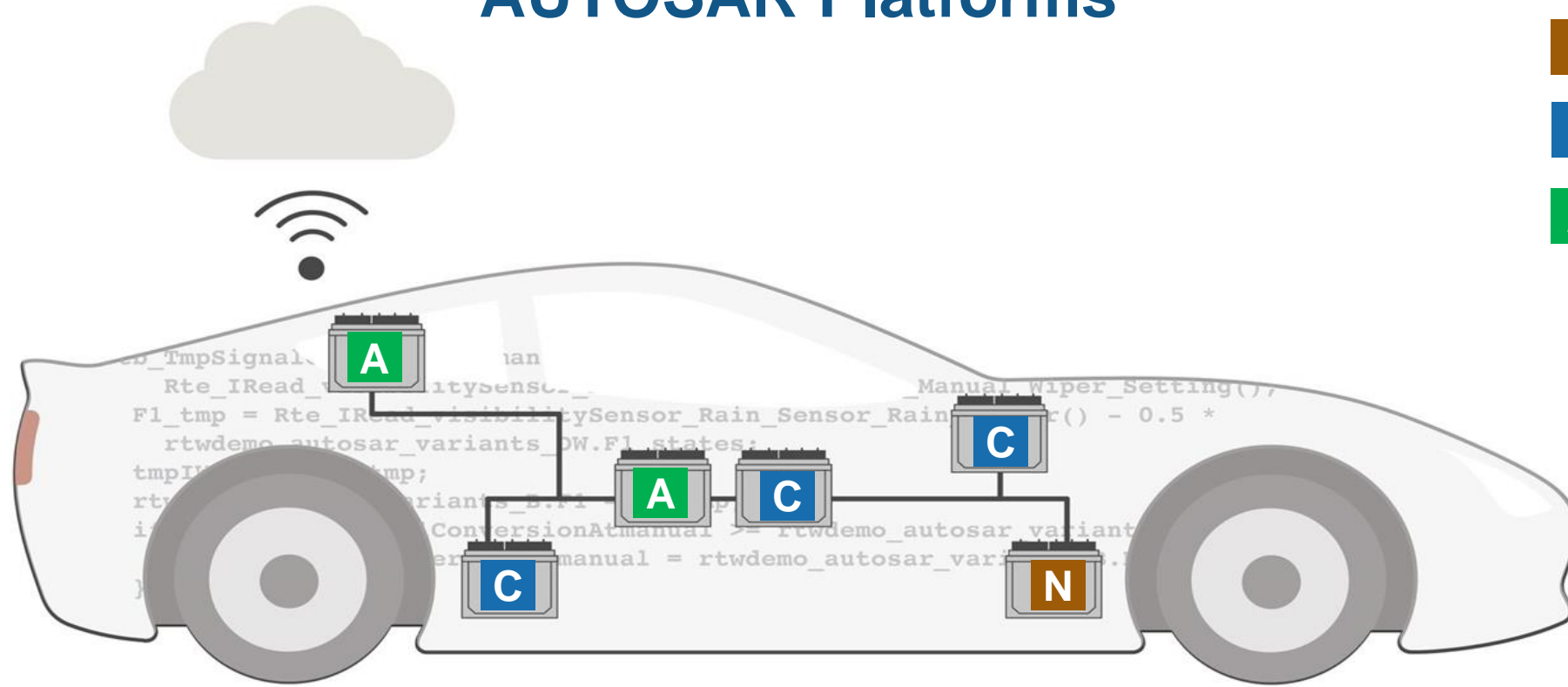
- Main drivers – Automated driving, Car-2-car/infrastructure applications



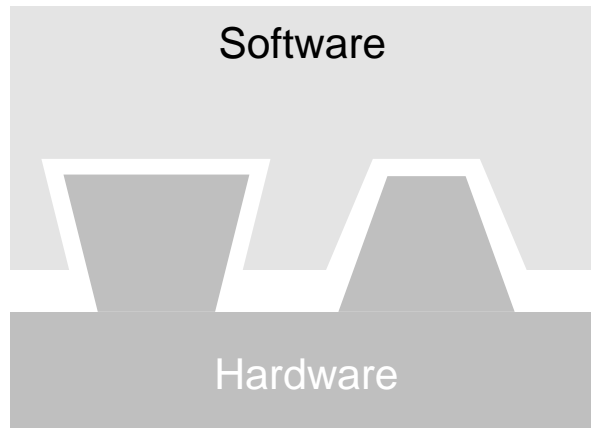
OVER THE AIR UPDATE

AUTOSAR Platforms

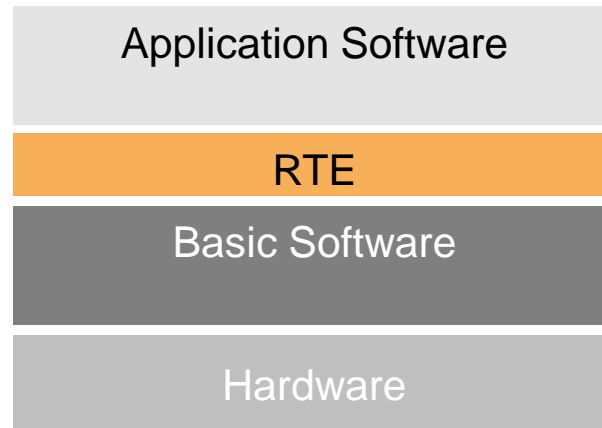
- N** Non - AUTOSAR
- C** Classic - AUTOSAR
- A** Adaptive - AUTOSAR



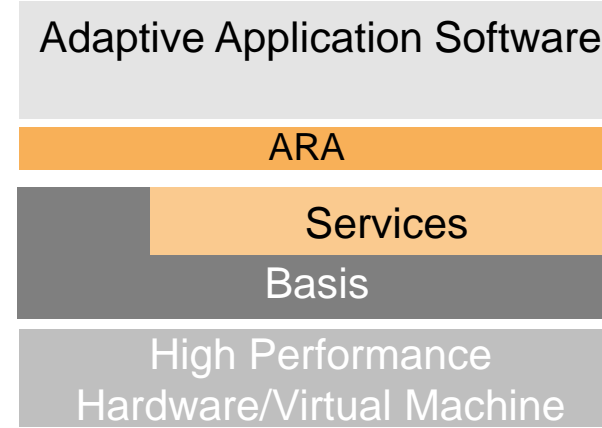
Non- AUTOSAR



Classic AUTOSAR



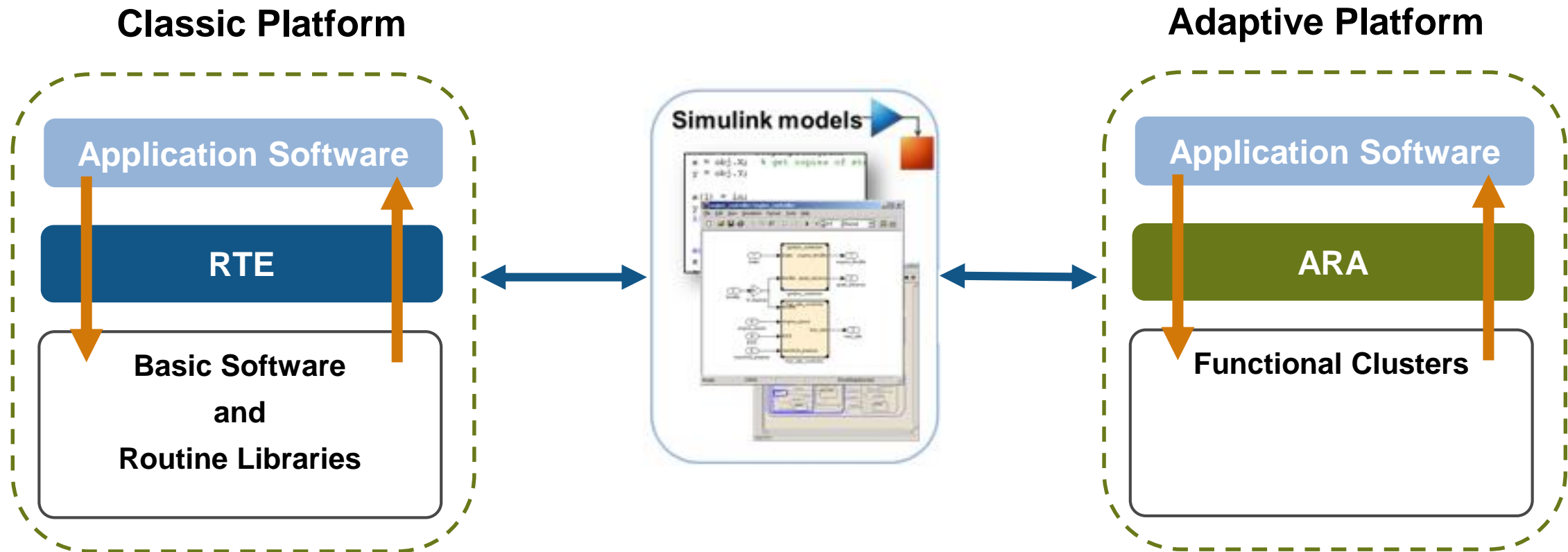
Adaptive AUTOSAR



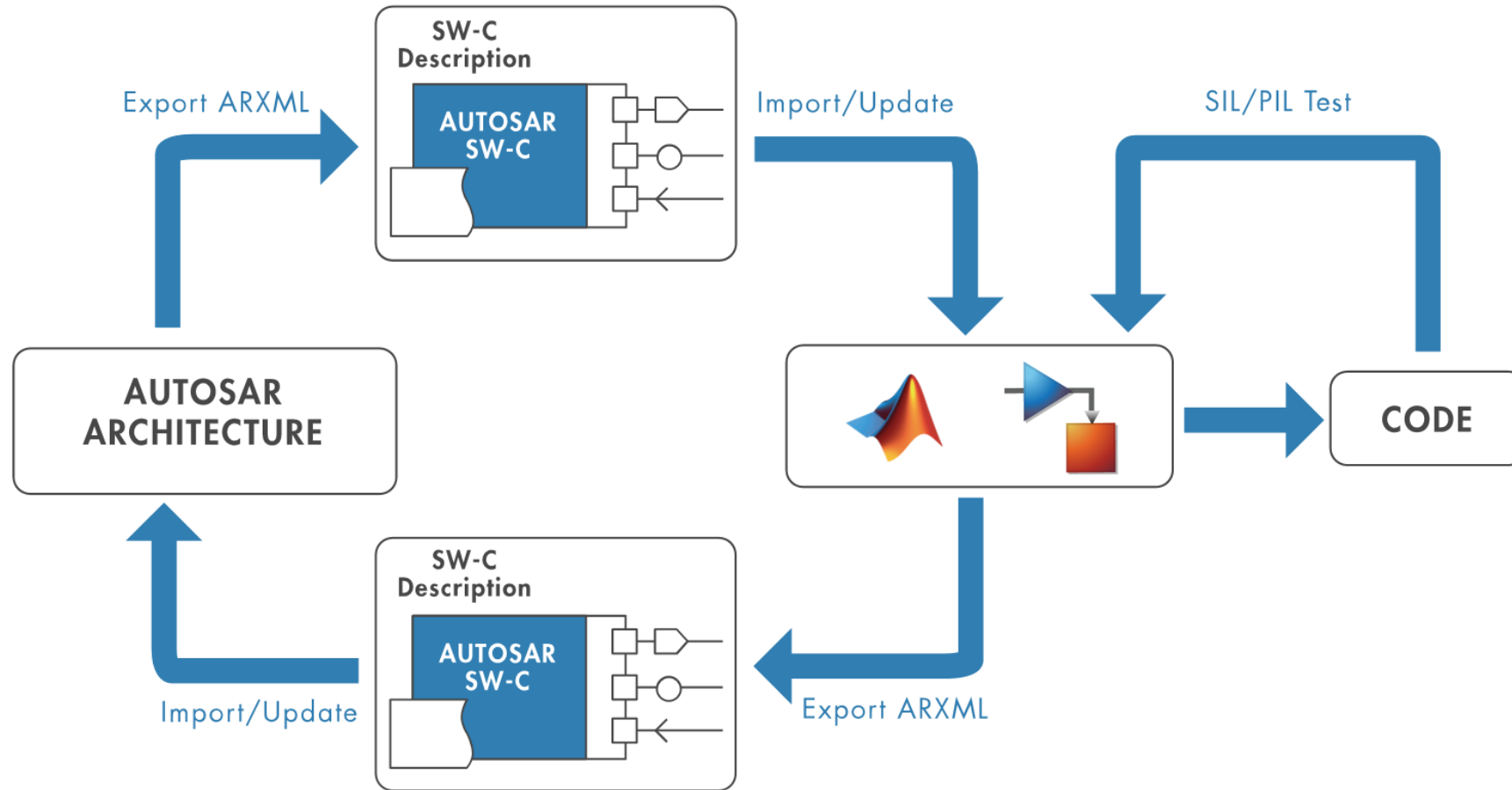
Agenda

- AUTOSAR is already on the road
- **Simulink for AUTOSAR**
 - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
 - Simulation of AUTOSAR ECU software
 - Blocks for AUTOSAR Library routines
 - Scaling from software components to compositions
- Simulink for Adaptive Platform
- Additional Resources

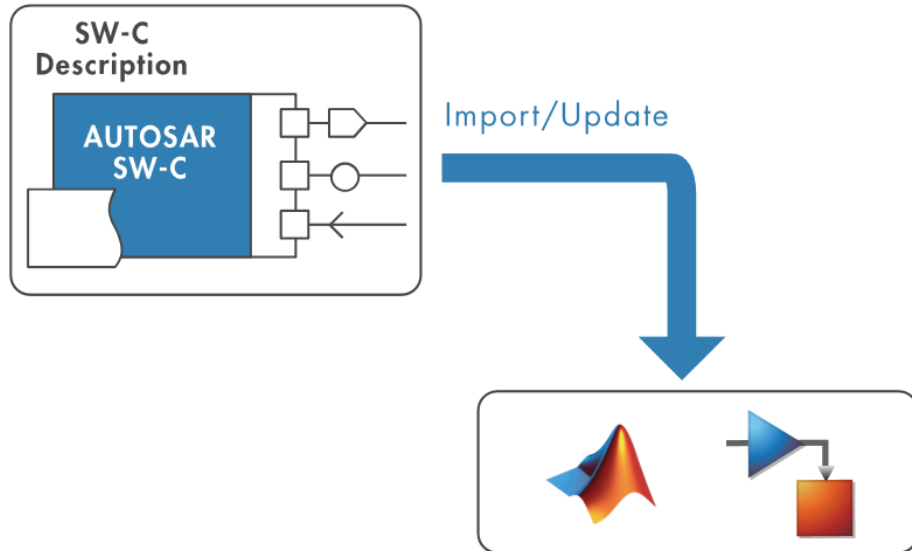
Intuitive and Powerful AUTOSAR Software Design in Simulink



Importing and Exporting AUTOSAR SW-C Descriptions (ARXML files)



It is easy to get started from an AUTOSAR description (Import)

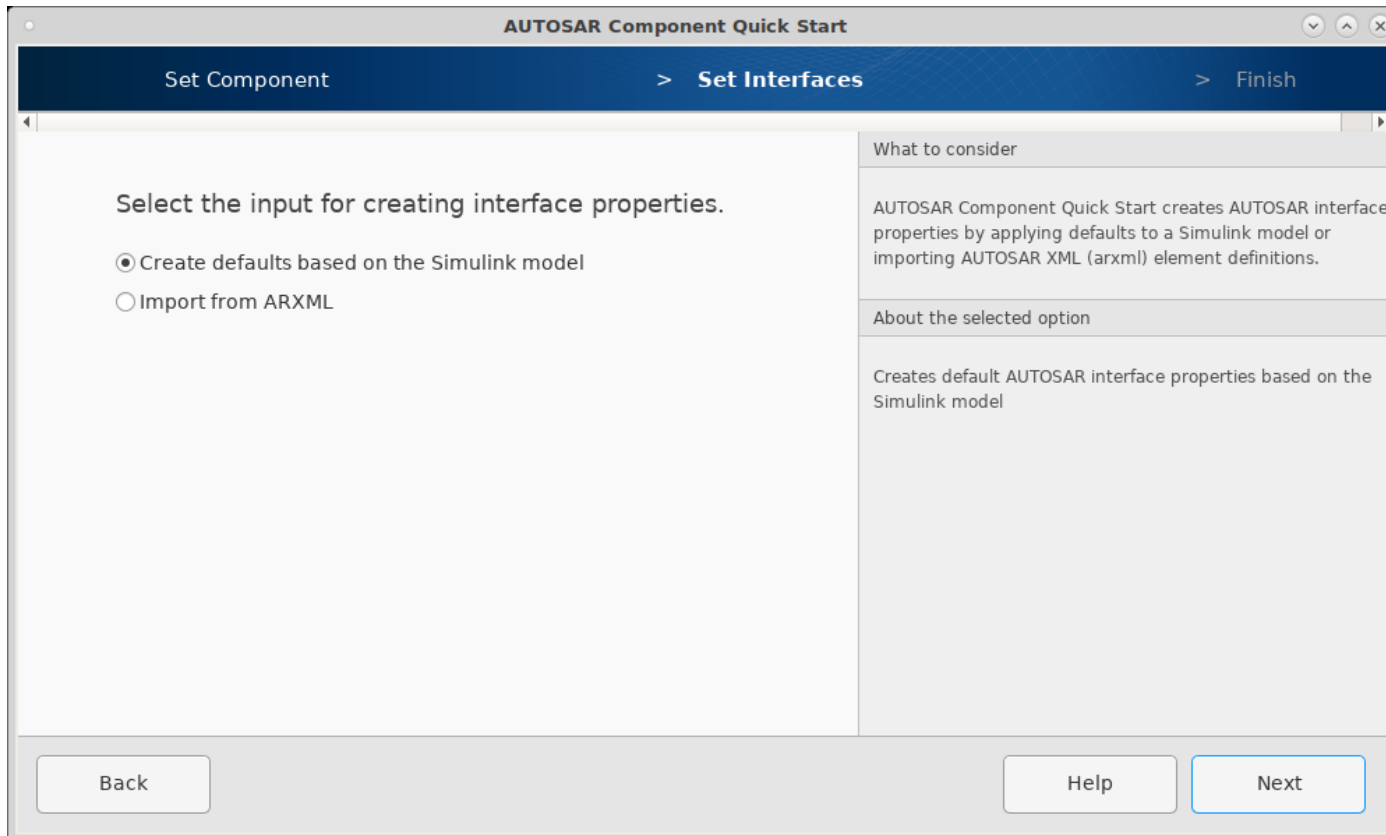



1. Import SW-C description (arxml) & create Simulink model

```
h = arxml.importer('mySWC.arxml')  
h.createComponentAsModel('/path/mySWC')
```

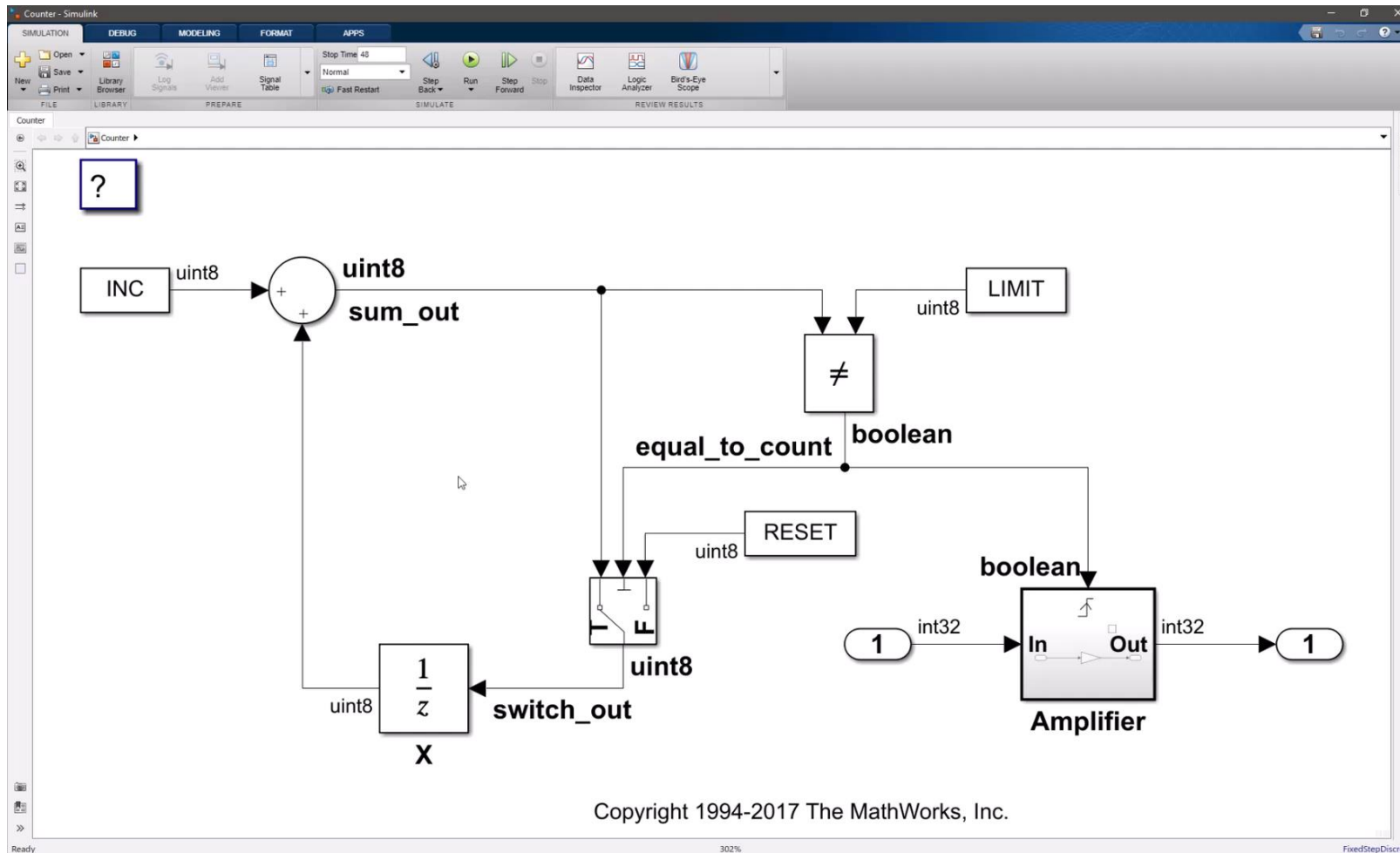
2. Elaborate SW-C Design, implement & generate code from model

It is also easy & quick to configure a Simulink model for AUTOSAR

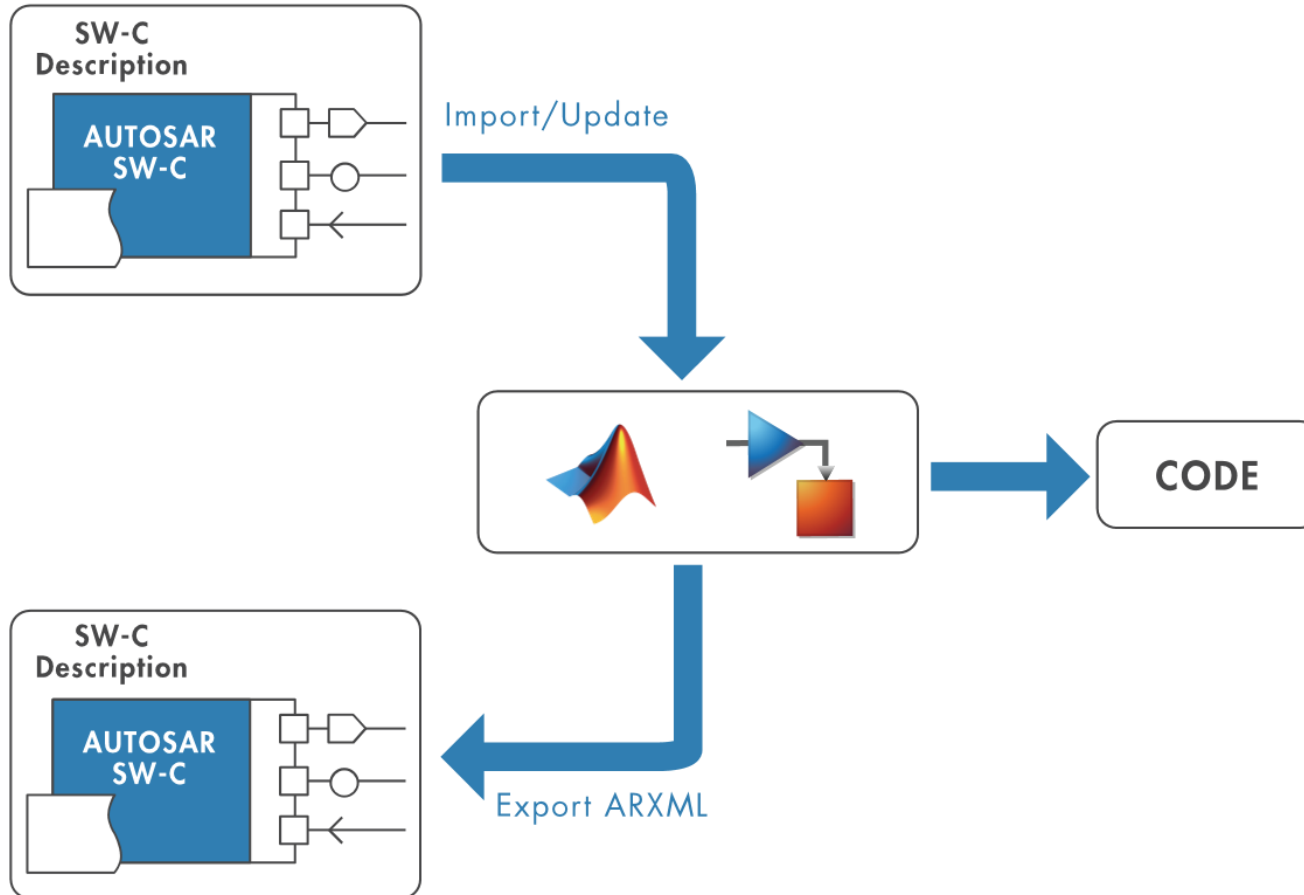


1. Start with a Simulink model
2. Click the AUTOSAR Component Quick Start App
 
3. Elaborate SW-C Design, implement & generate code from model

Example of Configuring a model for AUTOSAR

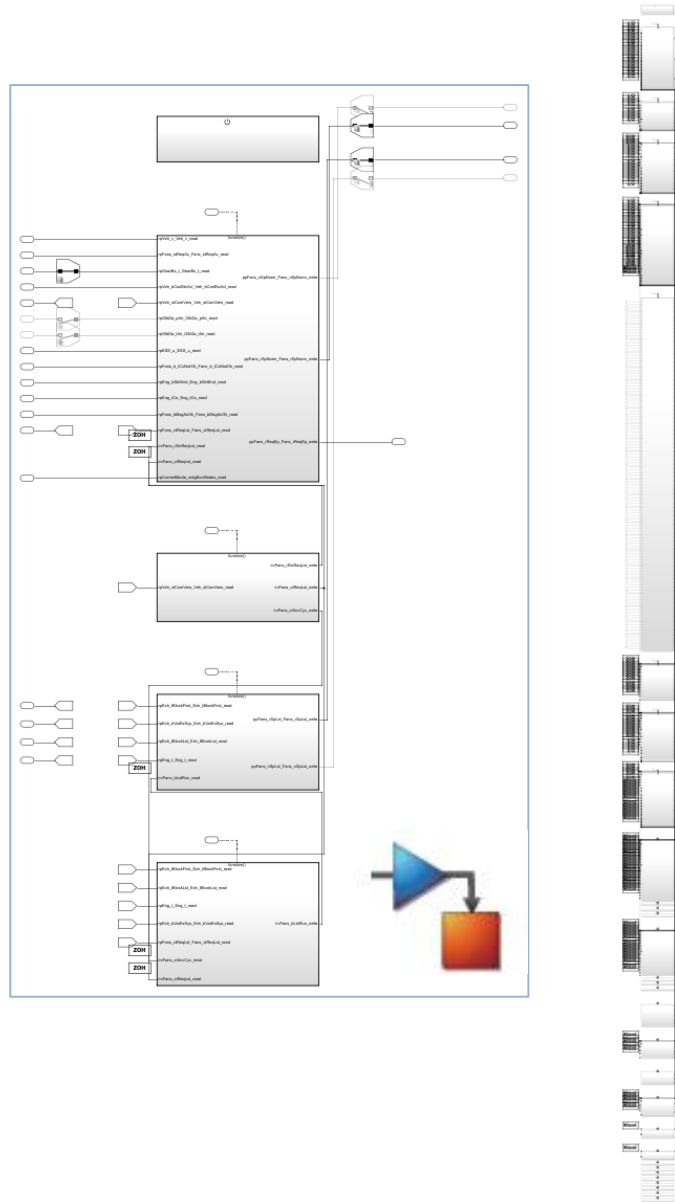


Now we can focus on modeling



1. Start with a Simulink model (or import SW-C description)
2. Elaborate SW-C design, implement & generate code from model

AUTOSAR SW-C design in Simulink



?

- 1) What blocks in this model need to be configured for AUTOSAR?
- 2) How do I change my AUTOSAR properties in the model?
- 3) Where do I get more information/help?

Introducing AUTOSAR “perspective” in a Simulink model

Quick Help

Help on configuring model for AUTOSAR

The screenshot shows the Embedded Coder interface for a Simulink model named 'rtwdemo_autosar_swc_slfcns'. The main workspace contains a Simulink diagram with several blocks: 'Initialize_Function' (containing an 'initialize' block), 'curValIRV' (a 'Mixed' block), and 'SS1' (a 'Runnable_1s' block). The 'SS1' block has three inputs: 'Trigger_1s', 'SubVal', and 'Override', and one output: 'DataOut'. A 'Code Mappings - AUTOSAR' spreadsheet is open at the bottom, showing the following data:

Source	DataAccessMode	Port	Element
SubVal	ImplicitReceive	RPort	SubVal
Override	ImplicitReceive	RPort	Override

The 'Property Inspector' window on the right shows the properties for the 'SubVal' element:

NAME	VALUE
Source	SubVal
Code	
DataAccessMode	ImplicitReceive
Port	RPort
Element	SubVal
Communication attributes	
AliveTimeout	60
HandleNeverRec...	false
InitValue	0

Property Inspector

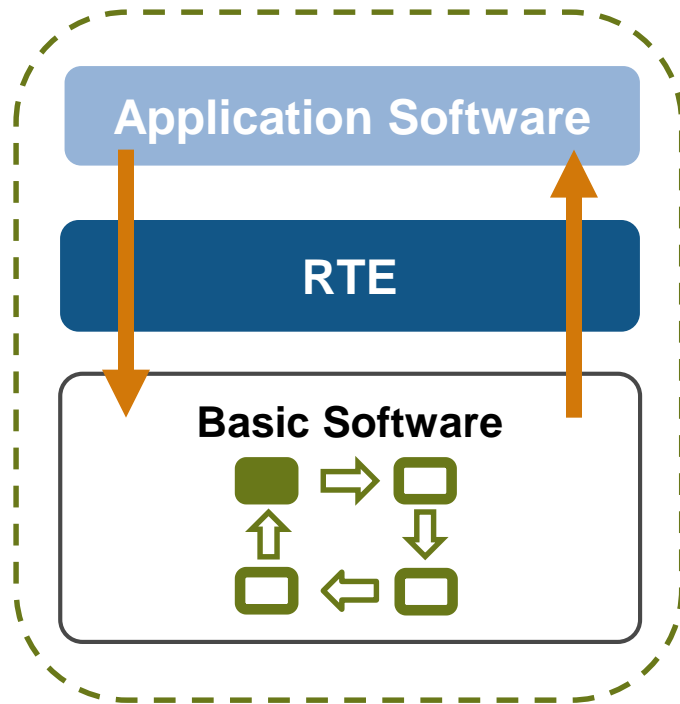
View/Edit AUTOSAR SW-C Properties

Code Mappings Spreadsheet

View/Edit all blocks and elements configured for AUTOSAR

Functional simulation of AUTOSAR basic software is critical for AUTOSAR ECU development

AUTOSAR ECU layered architecture



Many calls between application software and basic software



Basic software functionality is highly dynamic



Simulation of basic software reduces development time and improves software quality

Basic software library makes functional simulation of AUTOSAR basic software as easy as pressing the play button



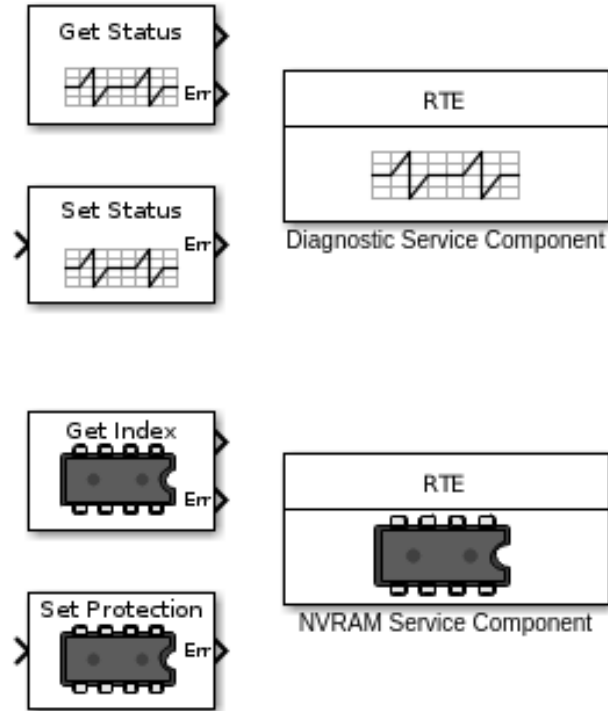
Specification of Diagnostic Event Manager
AUTOSAR Release 4.2.2

Document Title	Specification of Diagnostic Event Manager
Document Owner	AUTOSAR
Document Responsibility	AUTOSAR
Document Identification No	019
Document Classification	Standard

Document Status	Final
Part of AUTOSAR Release	4.2.2

Document Change History		
Release	Changed by	Description
4.2.2	AUTOSAR Release Management	<ul style="list-style-type: none"> New APIs <code>Dem_GetEventFreezeFrameDataEx</code> and <code>Dem_GetEventExtendedDataRecordEx</code> with buffersize as parameter and corrected return value definitions. Providing OBD FreezeFrame for UDS service 0x19 0x05 ISO 14229-1:2013[1] NRC handling for service 0x14 Refined service interfaces for DataElements minor corrections / clarifications / editorial changes; For details please refer to the ChangeDocumentation
4.2.1	AUTOSAR Release Management	<ul style="list-style-type: none"> Support of ISO 27145 (WWH-OBD / Euro VI)[2] Update to support ISO 14229-1:2013[3] Introduction of event dependencies Refined DTC/Event suppression
4.1.3	AUTOSAR Release Management	<ul style="list-style-type: none"> Further clarification of event combination Clarification of DTC groups Editorial changes
4.1.2	AUTOSAR Release Management	<ul style="list-style-type: none"> Added API table for service interfaces Clarification of event combination Editorial changes Removed chapter(s) on change documentation

Encapsulated in



1 of 475

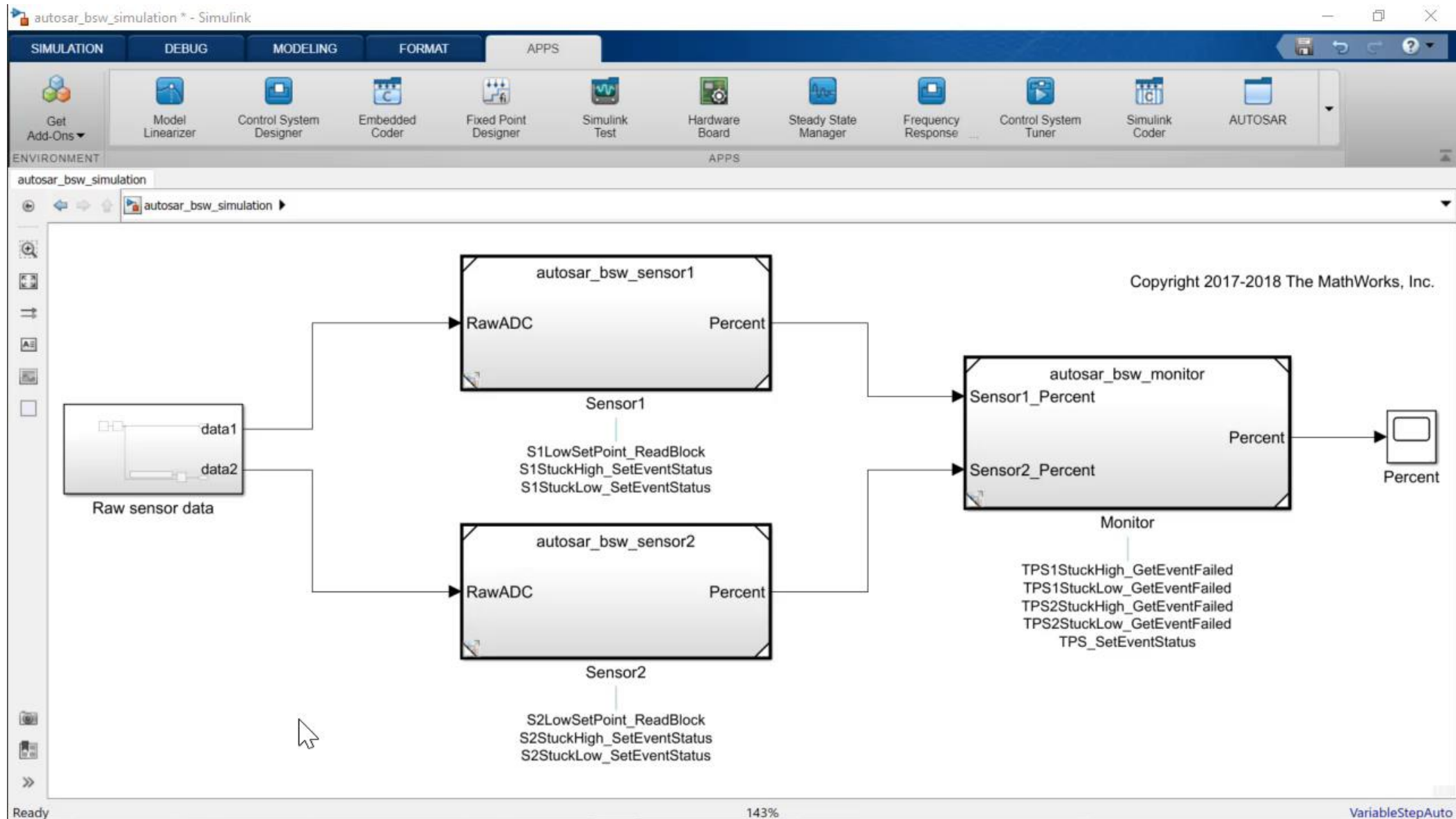
Document ID 019: AUTOSAR_SWS_DiagnosticEventManager
— AUTOSAR CONFIDENTIAL —

Detailed Specifications

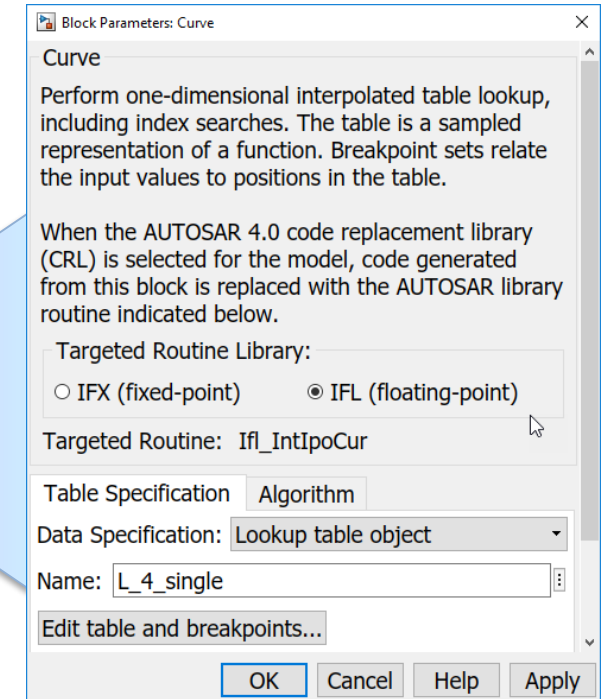
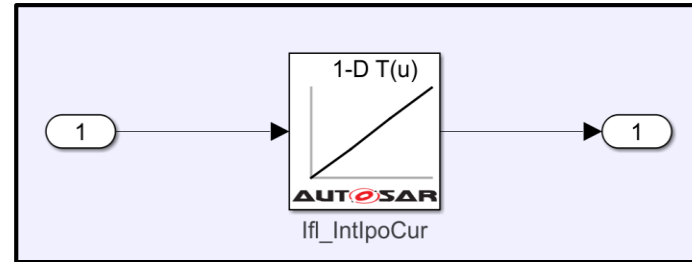
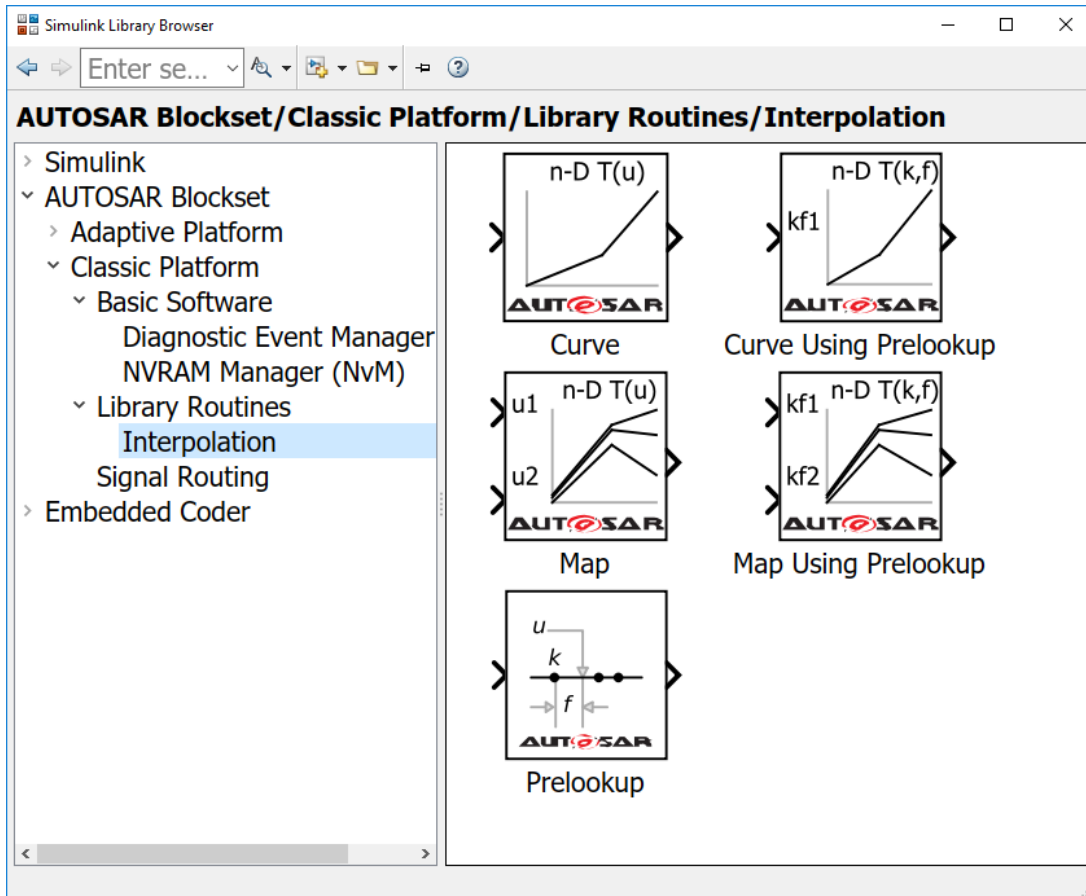
R2017b

Basic Software Library

It's easy to configure and play!



AUTOSAR Library Routines



```

Rte_IWrite_Runnable_Step_Out1_Out1 (Ifl_IntIpoCur_f32_f32
(Rte_IRead_Runnable_Step_In1_In1(), Rte_CData_L_4_single()->Nx,
Rte_CData_L_4_single()->Bp1, Rte_CData_L_4_single()->Table));
  
```

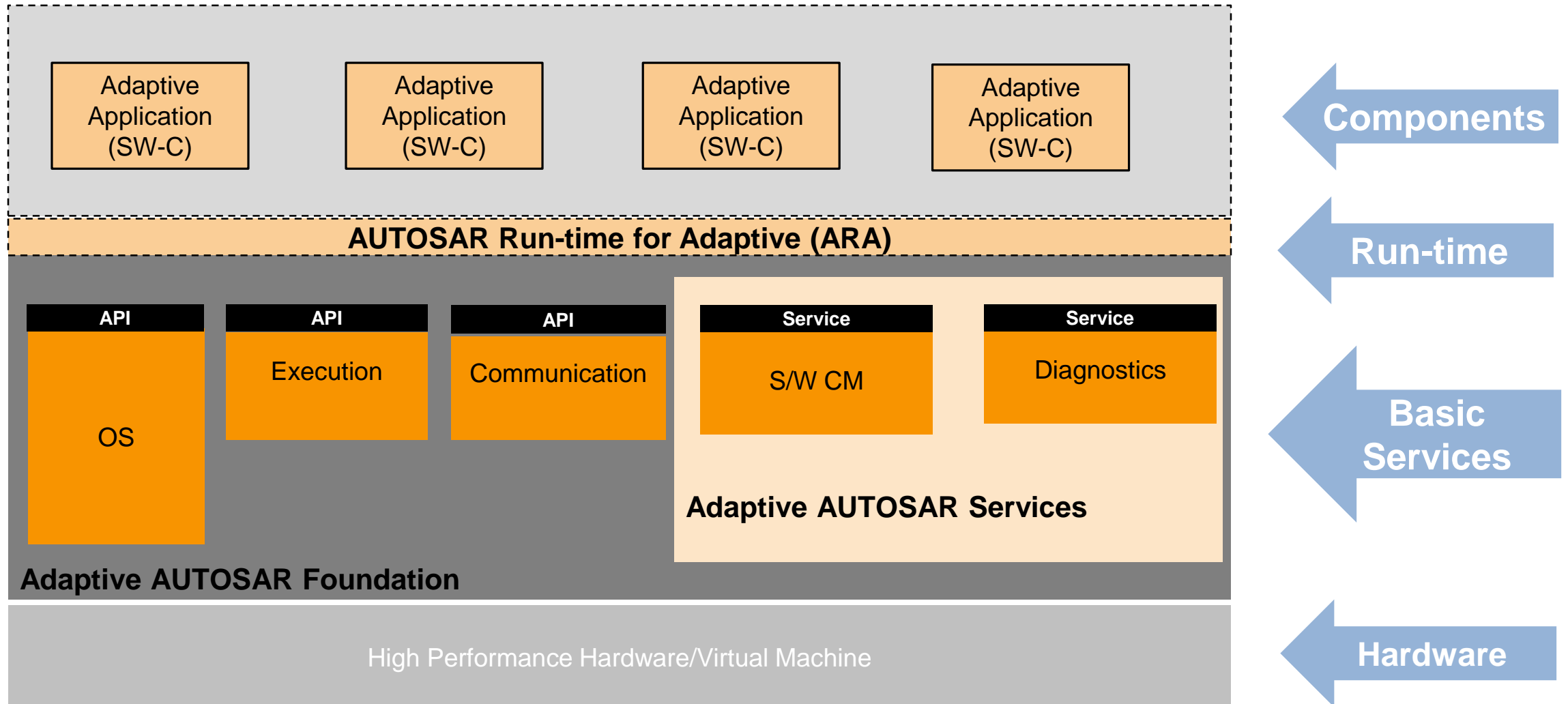
Scaling from software components to compositions



Agenda

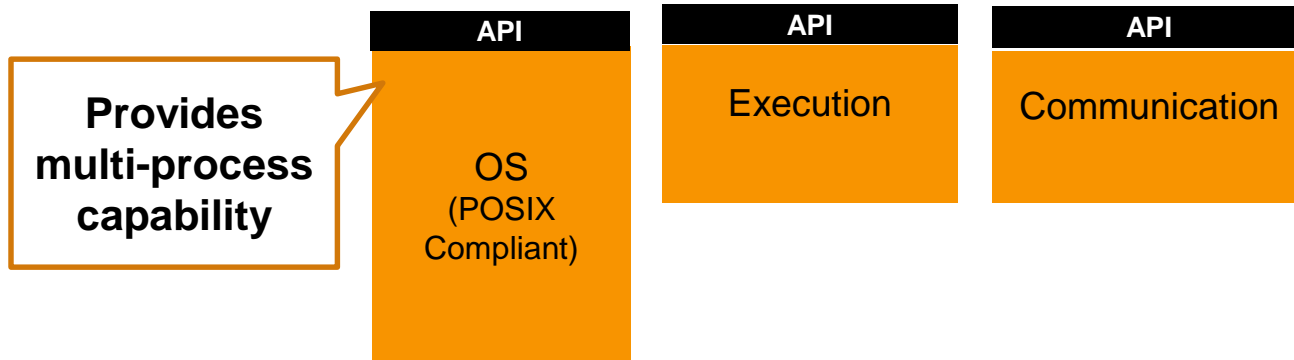
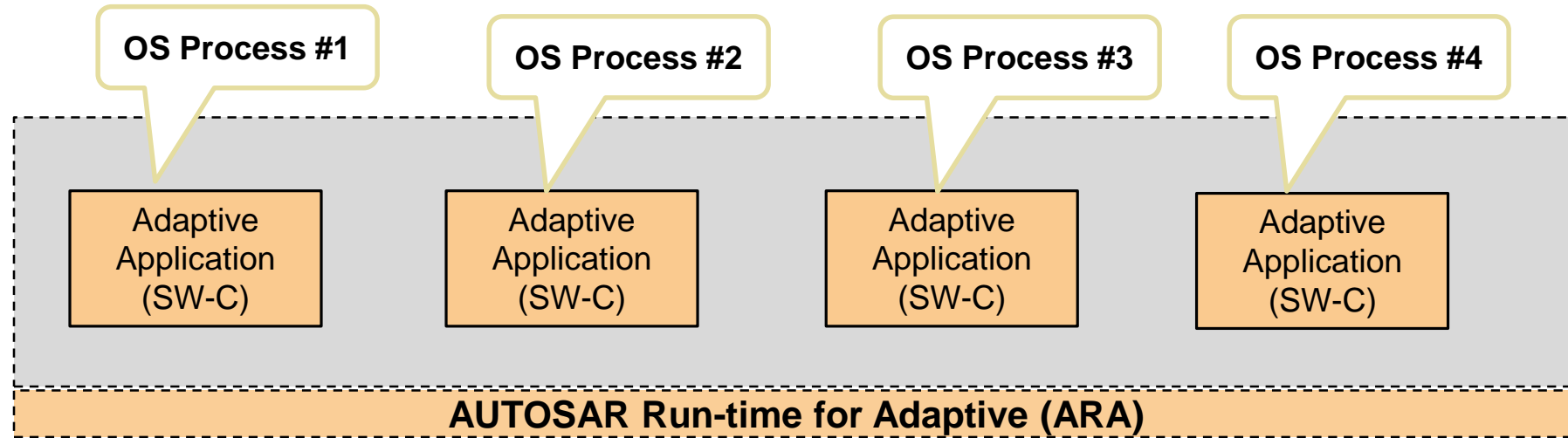
- AUTOSAR is already on the road
- Simulink for AUTOSAR
- **Simulink for Adaptive Platform**
 - A closer look at the Adaptive layers
 - Motivation for Simulink to support Adaptive
 - Mapping Adaptive platform to Simulink
 - Code Generation for Adaptive components
- Additional Resources

AUTOSAR Layered Software Architecture



Key Concept #1

Everything is a process .. as in “OS process”

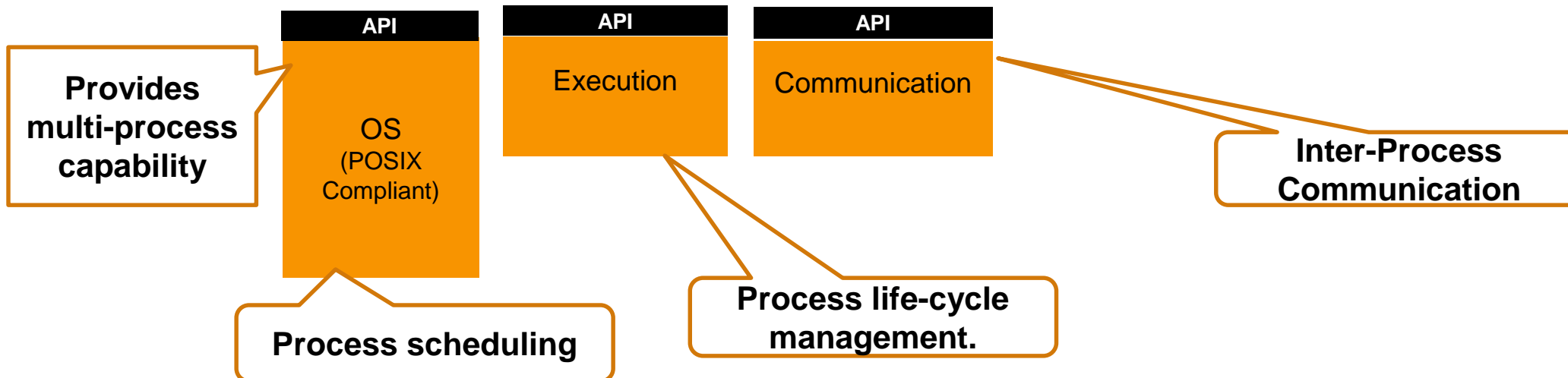
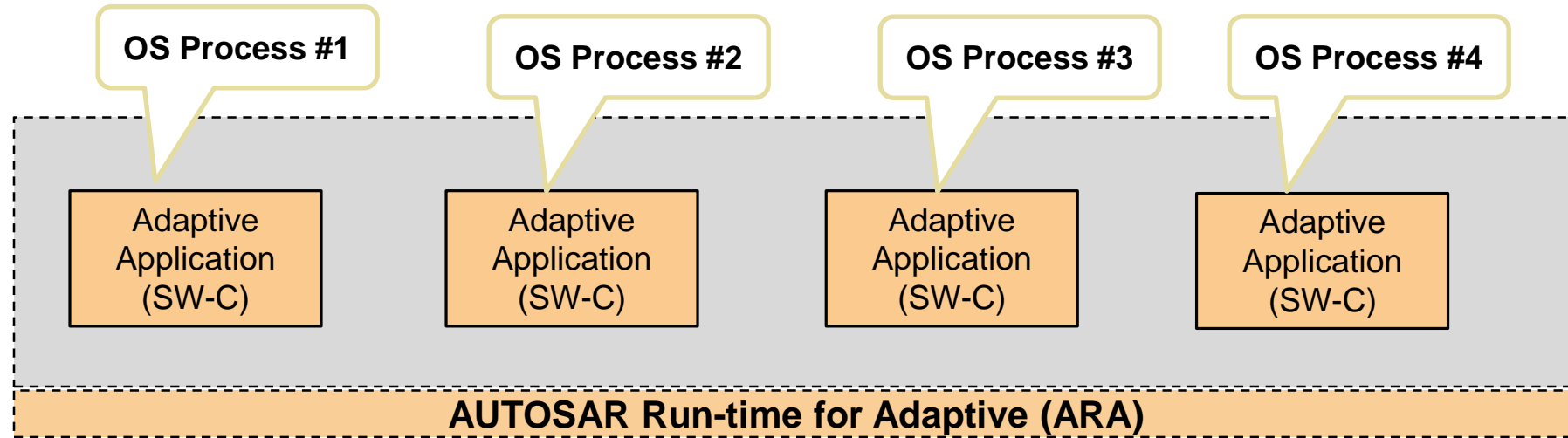


Notes: Each OS Process

- Corresponds to main() in C/C++ code
- Has own memory space & namespace
- Can be single or multi-threaded

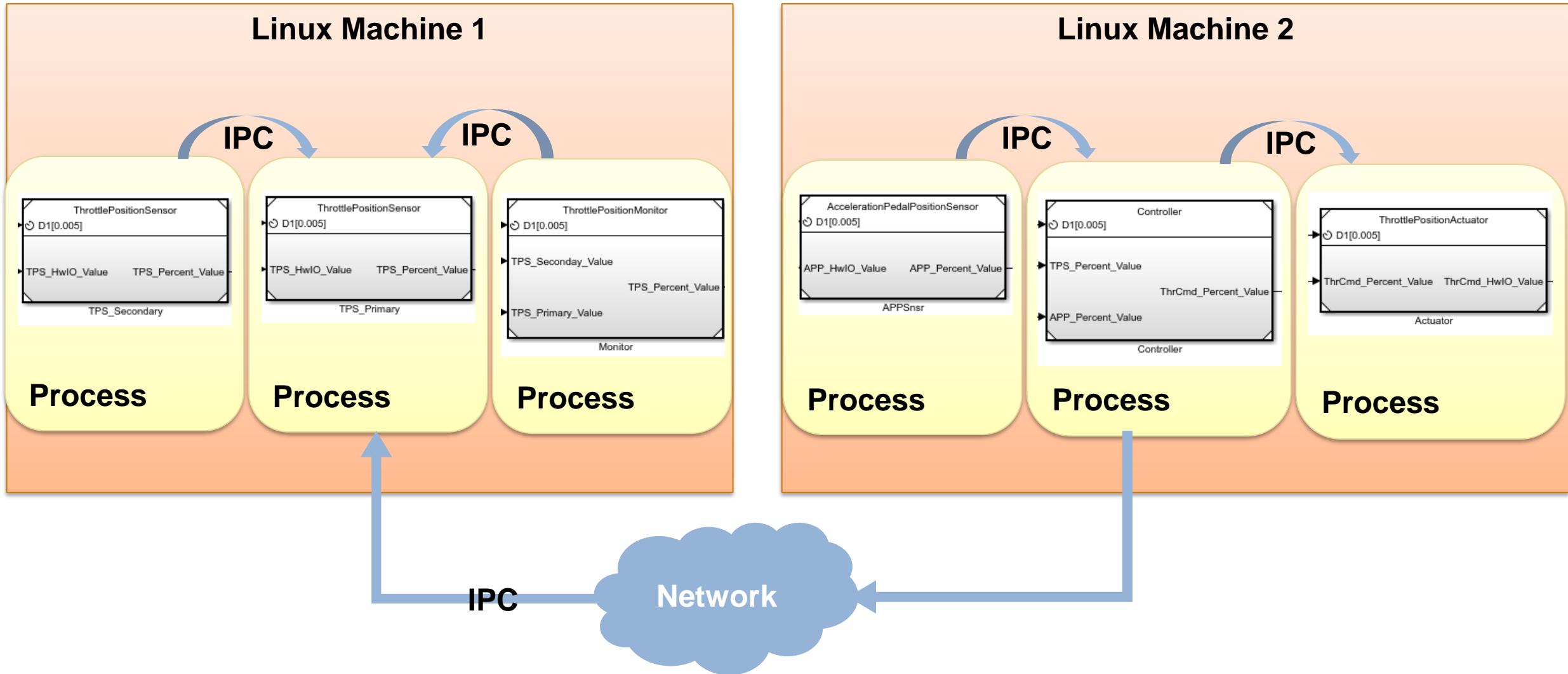
Key Concept #1

Everything is a process .. as in “OS process”



Key Concept #2

Service-oriented inter-process communication



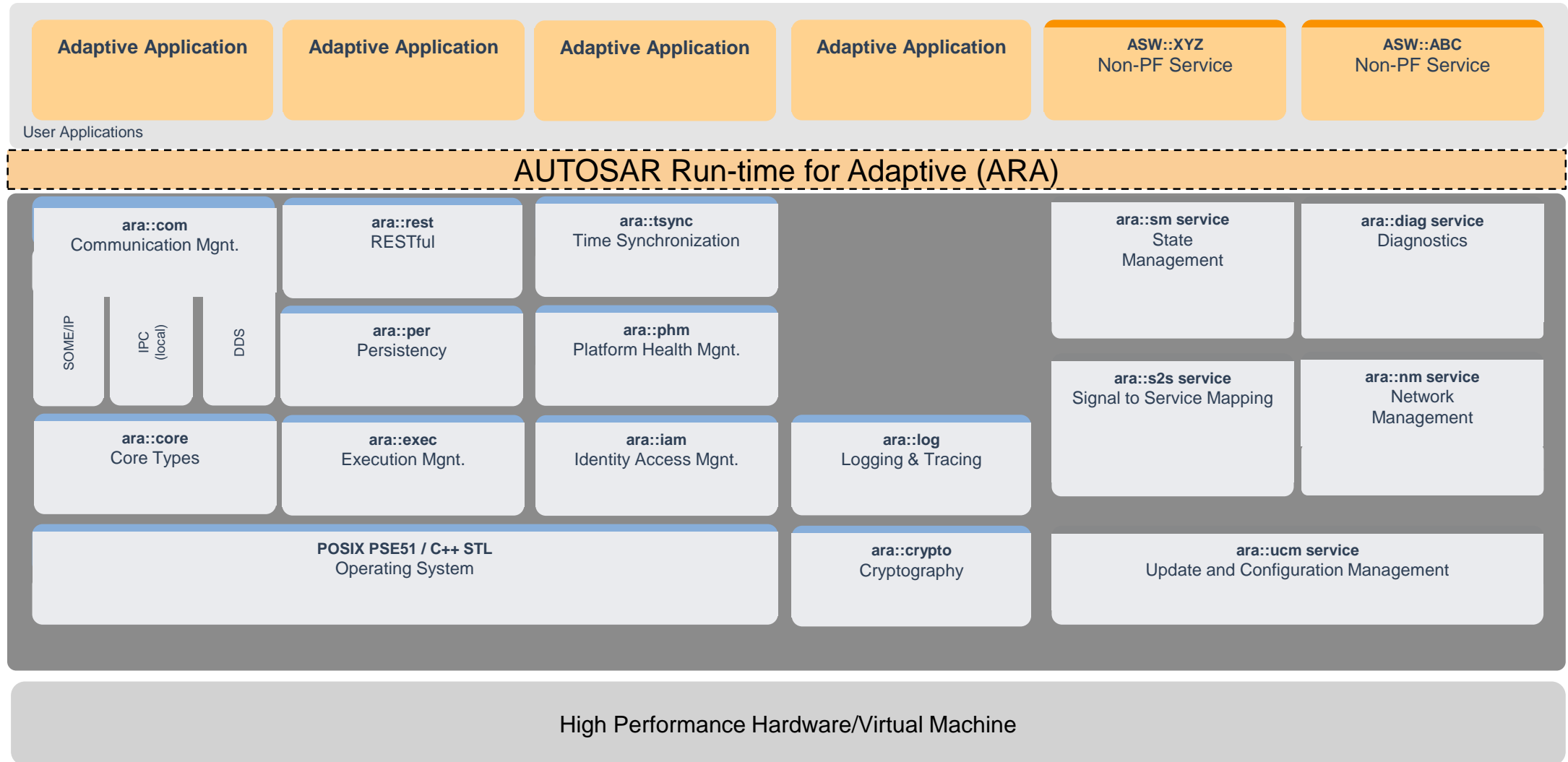
Key Concept #2

Service-oriented communication

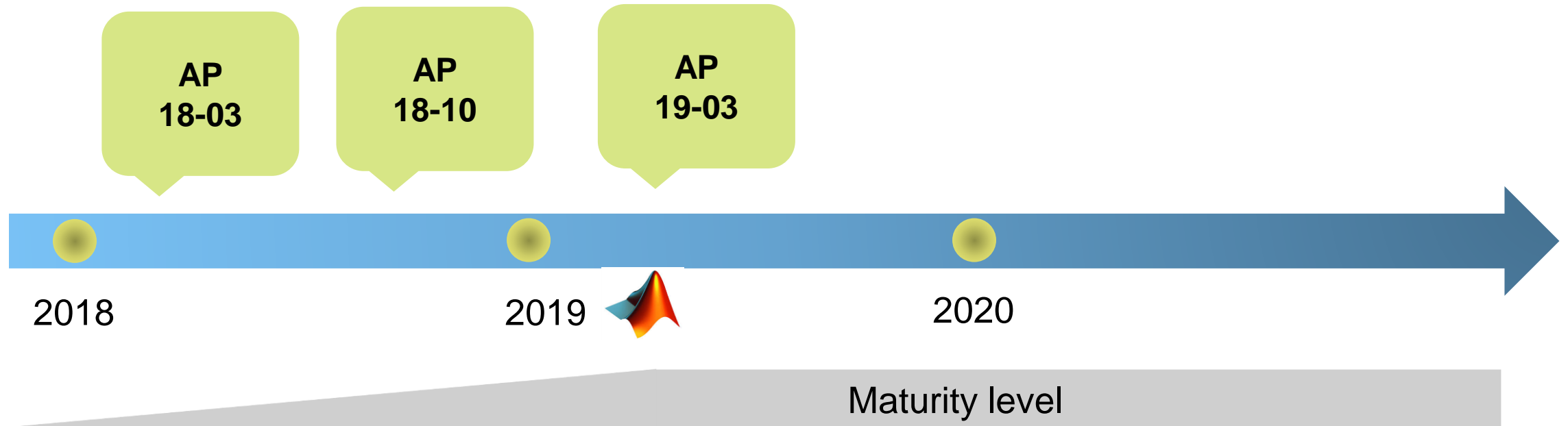
- Service Interface can contain
 - Methods (Functions)
 - Events (Messages)
 - Fields (Data)

```
    <<interface example>>  
    RadarService  
  
    • result = Calibrate(config)  
    • [success, out_pos] = Adjust(in_pos)  
  
    • BrakeEvent  
  
    • UpdateRate
```

Key Concept #3: Everything is C++



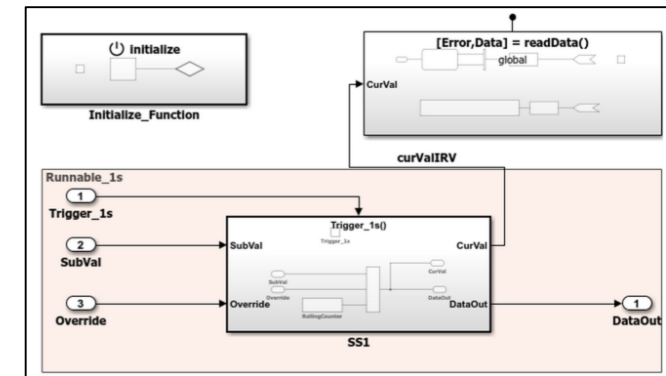
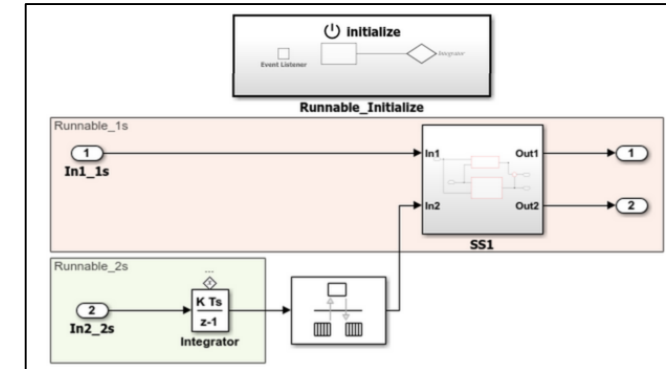
Adaptive Platform Roadmap



Early adopters – Volkswagen, BMW, Bosch, LG Electronics...

Motivation for Simulink to support Adaptive

- Simulink is heavily used for AUTOSAR Classic
- Customers have requested Simulink support for Adaptive platform
- Simulink supports service oriented modelling
- Embedded Coder generates C and C++ code
- MathWorks participates in the AUTOSAR standard development, including both Classic and Adaptive platforms



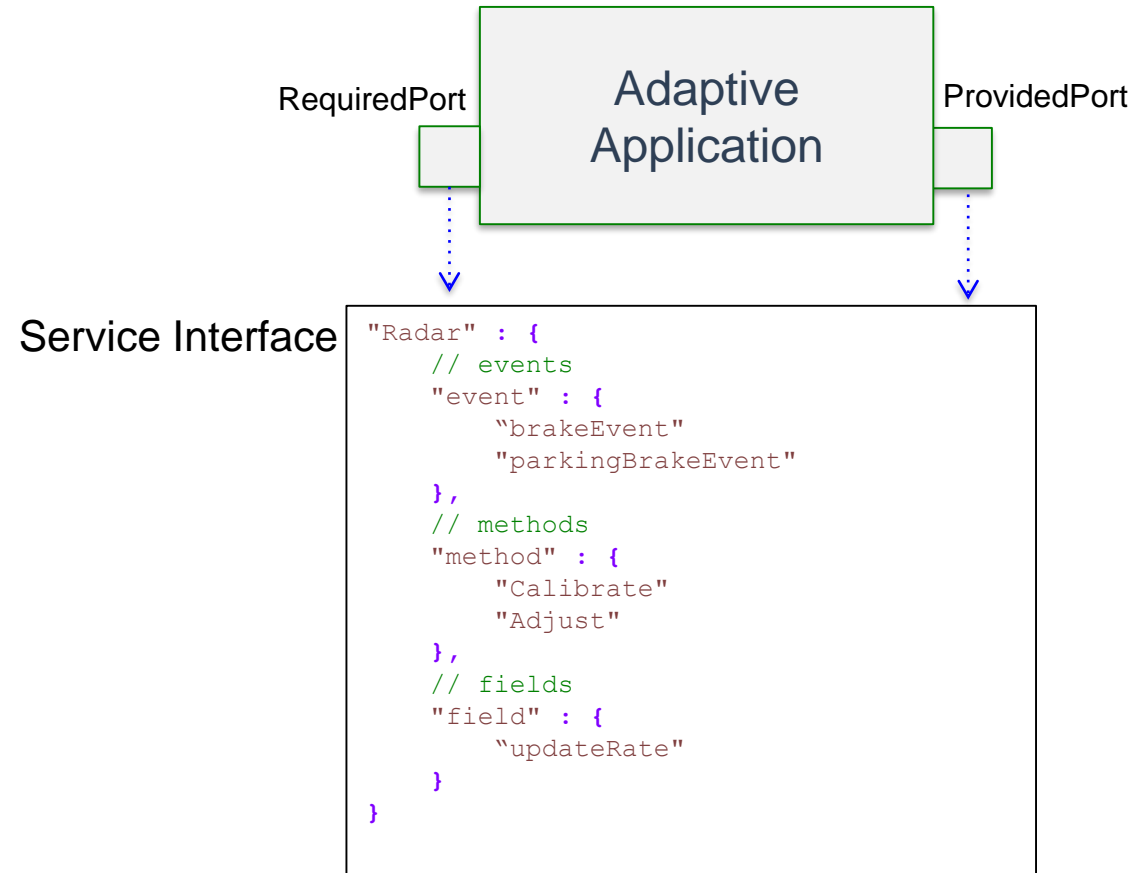
```

void autosar_Lane_Guidance>IfActionSS(real_T rtu_In1, real_T *rty_Out1)
{
    // Inport: '<S18>/In1'
    *rty_Out1 = rtu_In1;
}

// Function for Chart: '<S1>/Event_Receive'
boolean_T autosar_Lane_GuidanceModelClass::
autosar_Lane_Guidance_sf_msg_pop_EvtIn(void)
{
    boolean_T isPresent;
    const ara::com::SampleContainer< ara::com::SamplePtr< const real_T > >
    *sampleContainer;
    ara::com::SamplePtr< const real_T > samples;
    if (autosar_Lane_Guidance_DW.EvtIn_isValid_i) {
        isPresent = true;
    }
}

```

Adaptive SW Architecture Concepts



Mapping AUTOSAR AP Concepts to Simulink

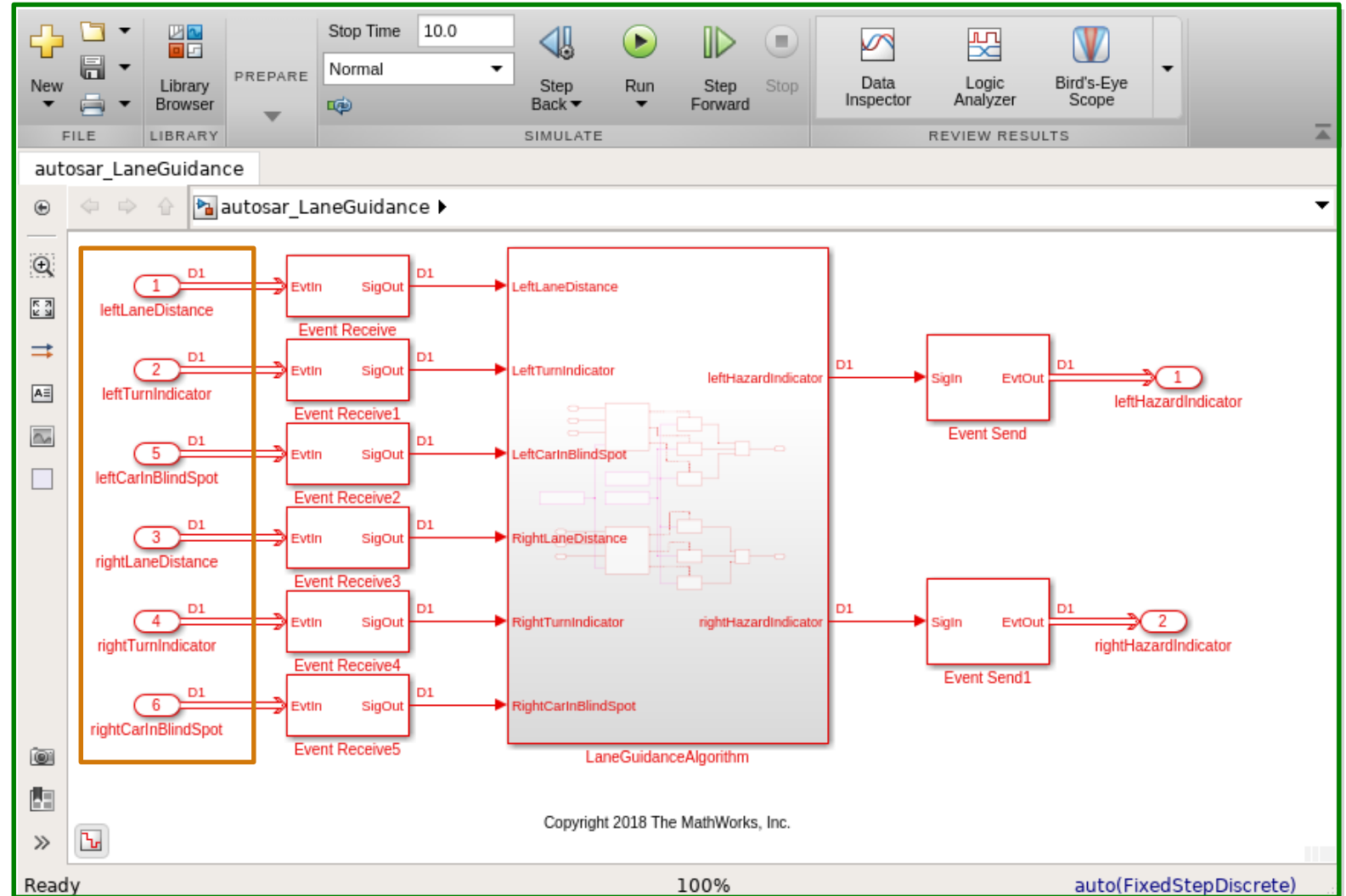


Adaptive Application

RequiredPort

```

"Radars" : {
  // events
  "event" : {
    "leftLaneDistance"
    "leftTurnIndicator"
    "leftCarInBlindSpot"
    "rightLandDistance"
    "rightTurnIndicator"
    "rightCarInBlindSpot"
  },
  // methods
  "method" : {
    "Calibrate"
    "Adjust"
  },
  // fields
  "field" : {
    "updateRate"
  }
}
  
```



Mapping AUTOSAR AP Concepts to Simulink

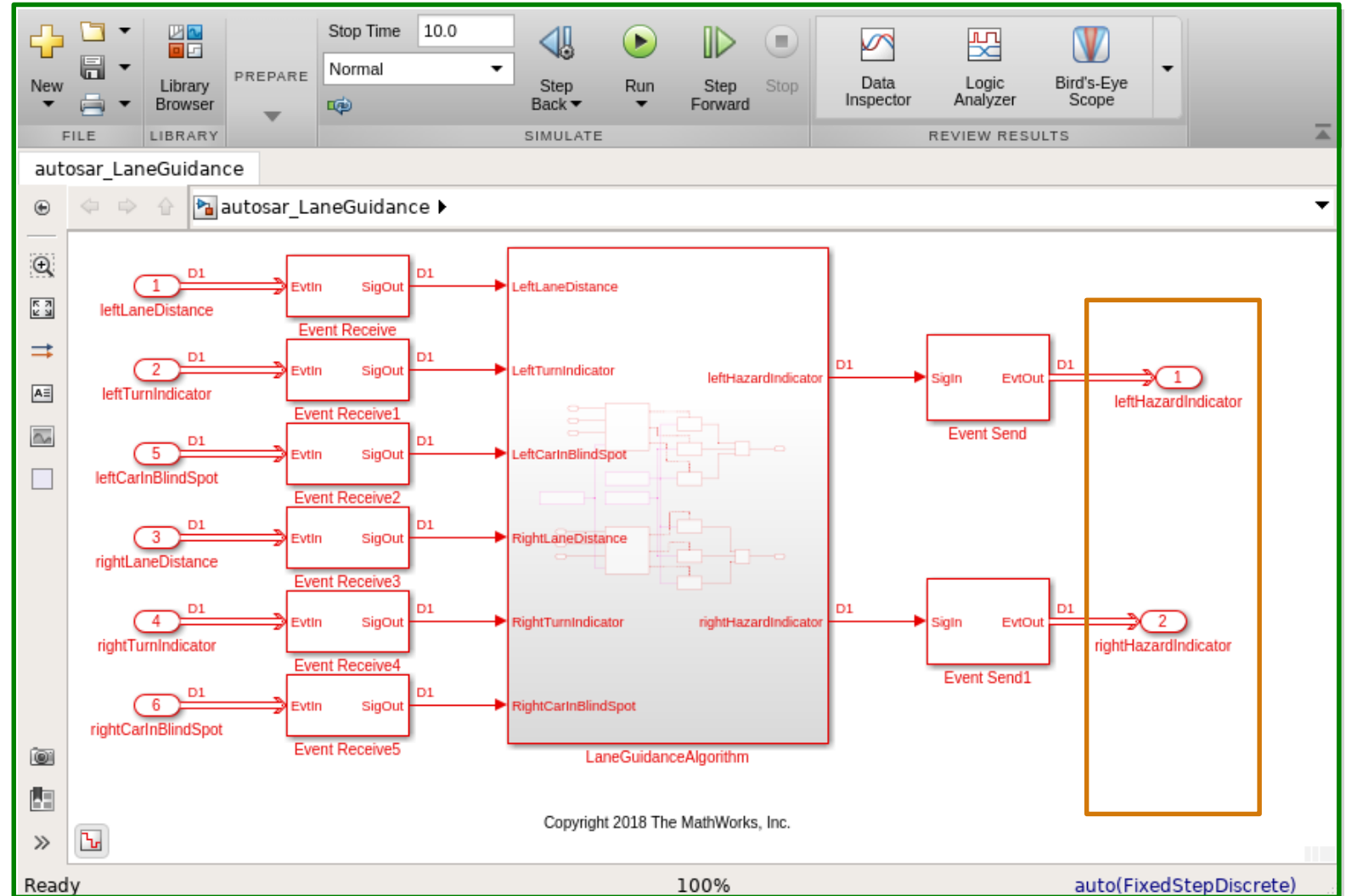


Adaptive Application

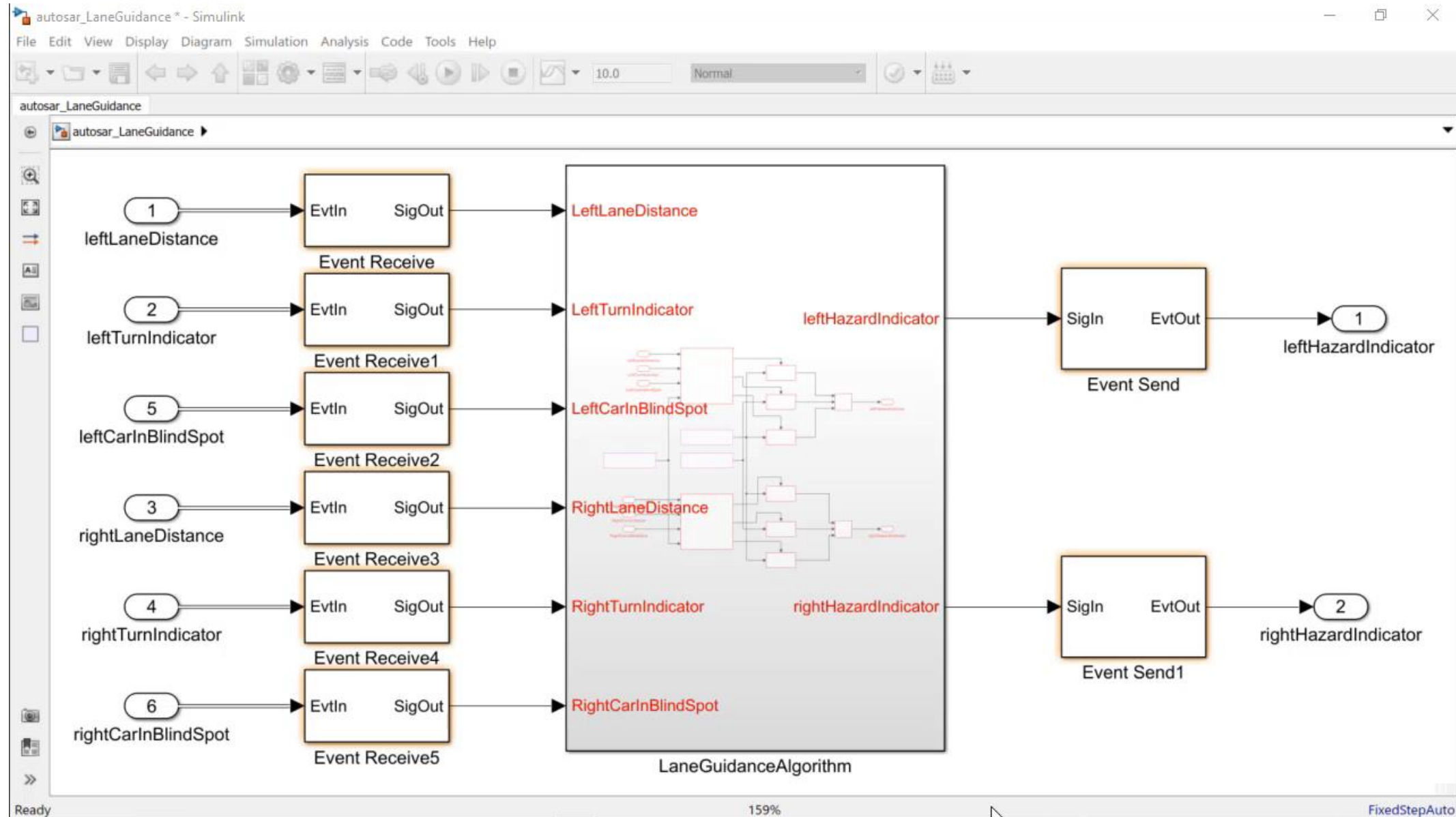
ProvidedPort

```

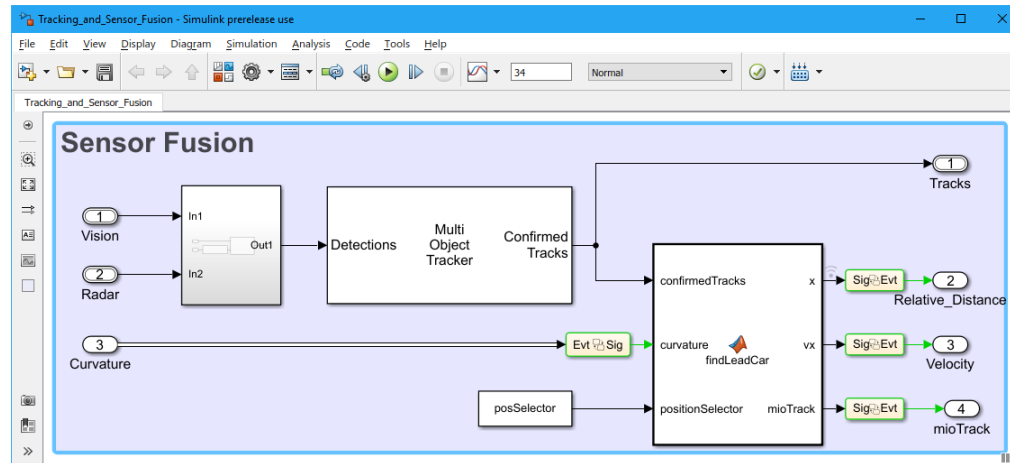
"Radar" : {
  // events
  "event" : {
    "leftHazardIndicator"
    "rightHazardIndicator"
  },
  // methods
  "method" : {
    "Calibrate"
    "Adjust"
  },
  // fields
  "field" : {
    "updateRate"
  }
}
    
```



AUTOSAR Adaptive in Action



Generate Production AUTOSAR Adaptive C++ Code



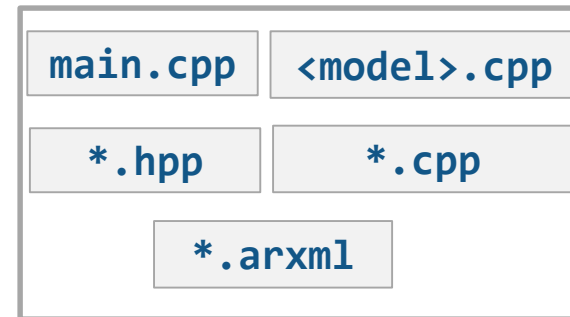
Configuration Parameters:

Target selection

System target file:

Language:

Description:



Out-of-box AUTOSAR support

1. Configure Model
 - ✓ Target
 - ✓ AUTOSAR Dictionary
2. Generate C++ code

AUTOSAR Dictionary

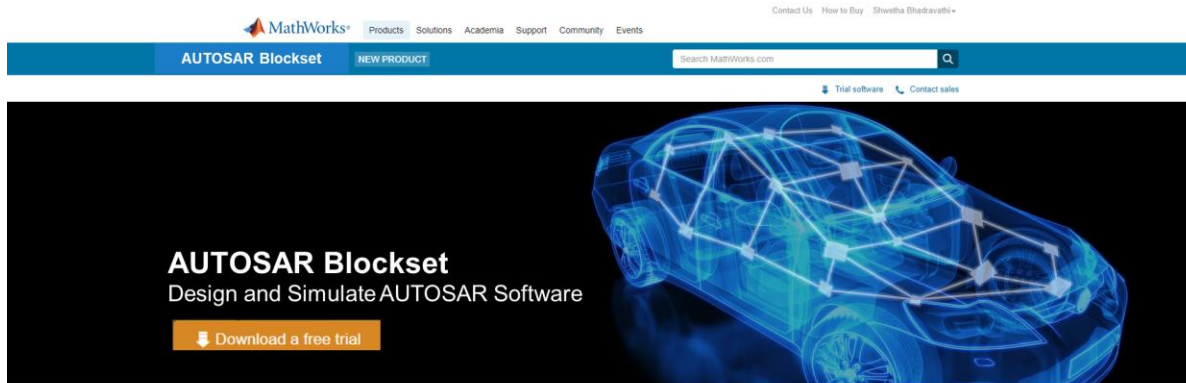
- Events

Name	SwCalibrationAccess
Curvature	ReadOnly
Prediction_Time	ReadOnly
Radar	ReadOnly
Vision	ReadOnly

Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
- **Additional Resources**

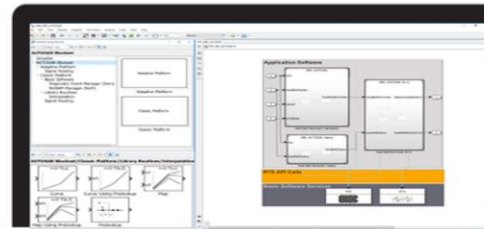
To learn more, please visit AUTOSAR Blockset page



AUTOSAR Blockset provides an AUTOSAR dictionary and blocks for developing Classic and Adaptive AUTOSAR software using Simulink® models. You can define AUTOSAR software component properties, interfaces, and datatypes, and map them to existing Simulink models using the AUTOSAR editor. Alternatively, the blockset provides an application interface that lets you automatically generate new Simulink models for AUTOSAR by importing software component and composition descriptions from AUTOSAR XML files.

AUTOSAR Blockset provides blocks and constructs for AUTOSAR library routines and Basic Software (BSW) services, including NVRAM and Diagnostics. By simulating the BSW services together with your application software model, you can verify your AUTOSAR ECU software without leaving Simulink.

AUTOSAR Blockset supports C and C++ production code generation and AUTOSAR XML file export (with Embedded Coder®). It is qualified for use with the ISO 26262 standard (with IEC Certification Kit).



Come see us at the demo booth

