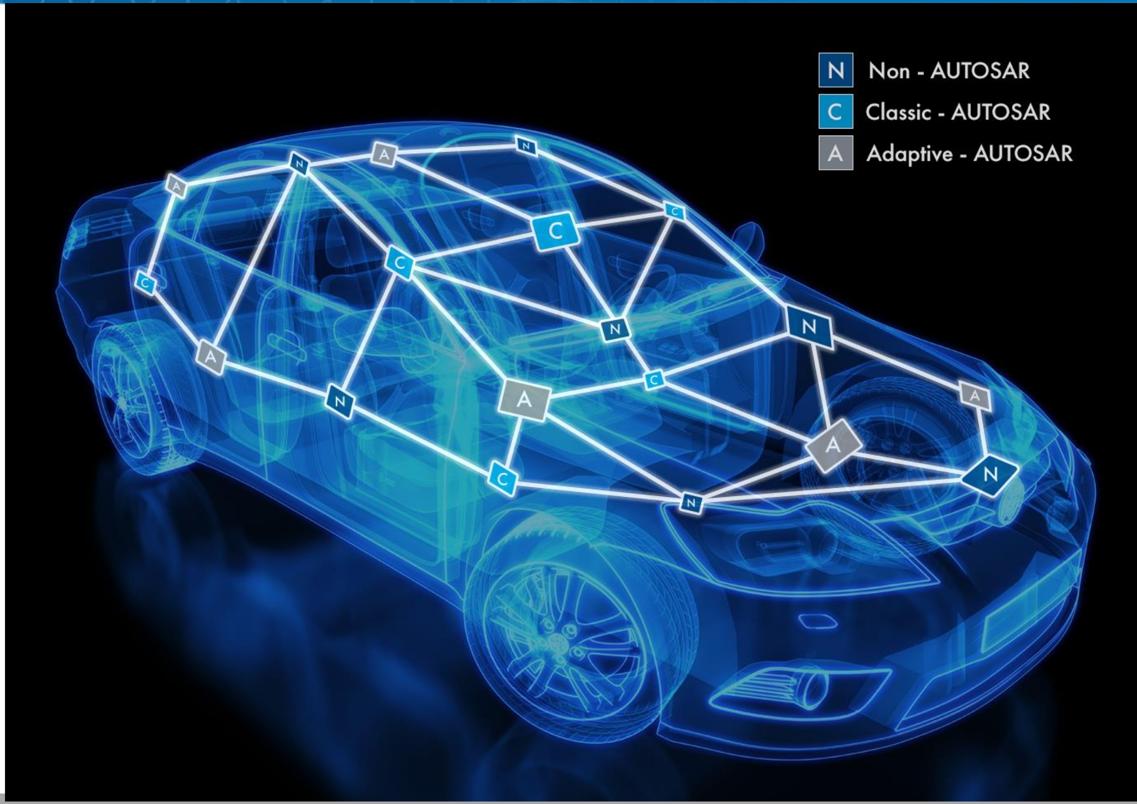
Simulink for AUTOSAR Adaptive

Mark Danielsen

Senior Application Engineer



Agenda

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

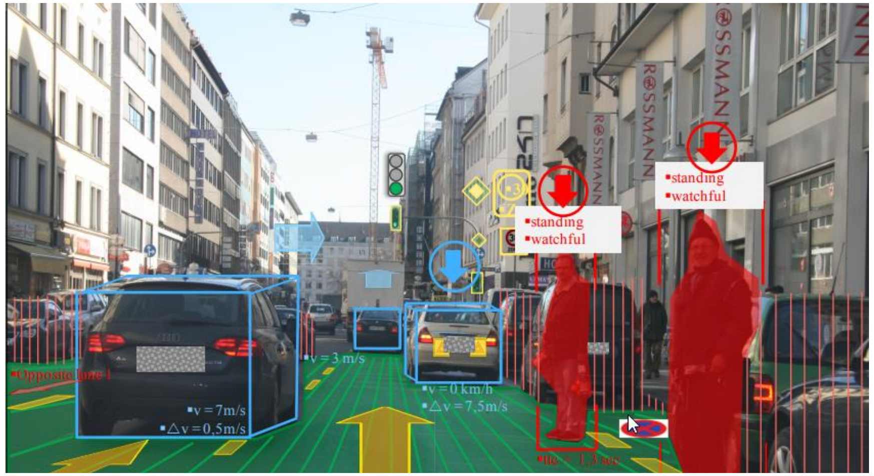
AUTOSAR Classic is already on the road

- BMW - Model-Based Software Development: An 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

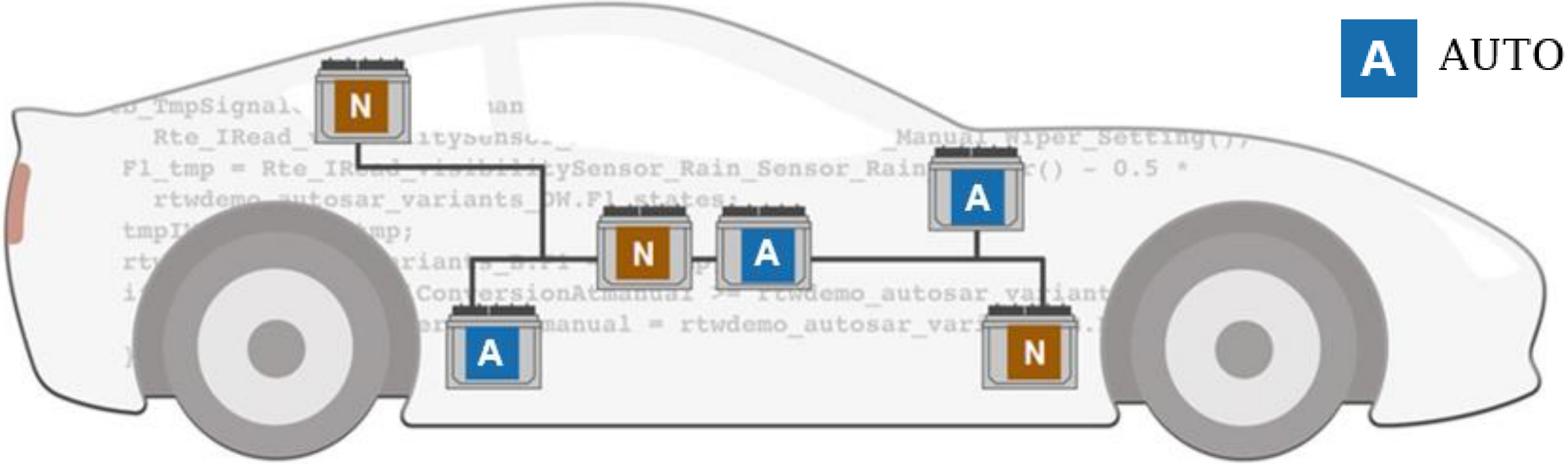
- John Deere - Vertical AUTOSAR System Development at John Deere

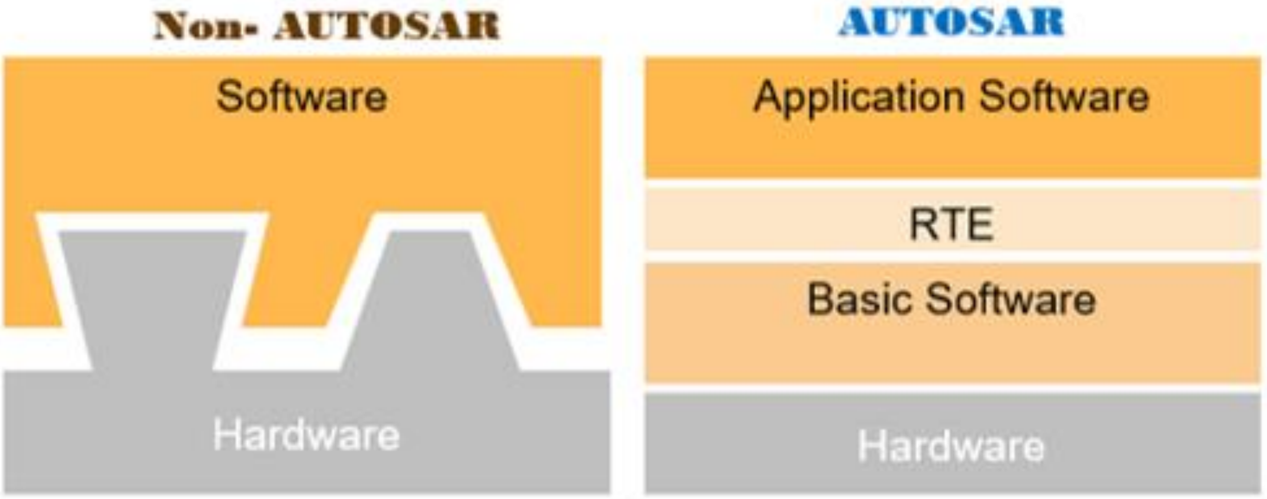


AUTOSAR at a System Level

N Non - AUTOSAR

A AUTOSAR




Non- AUTOSAR

Software

Hardware

AUTOSAR

Application Software

RTE

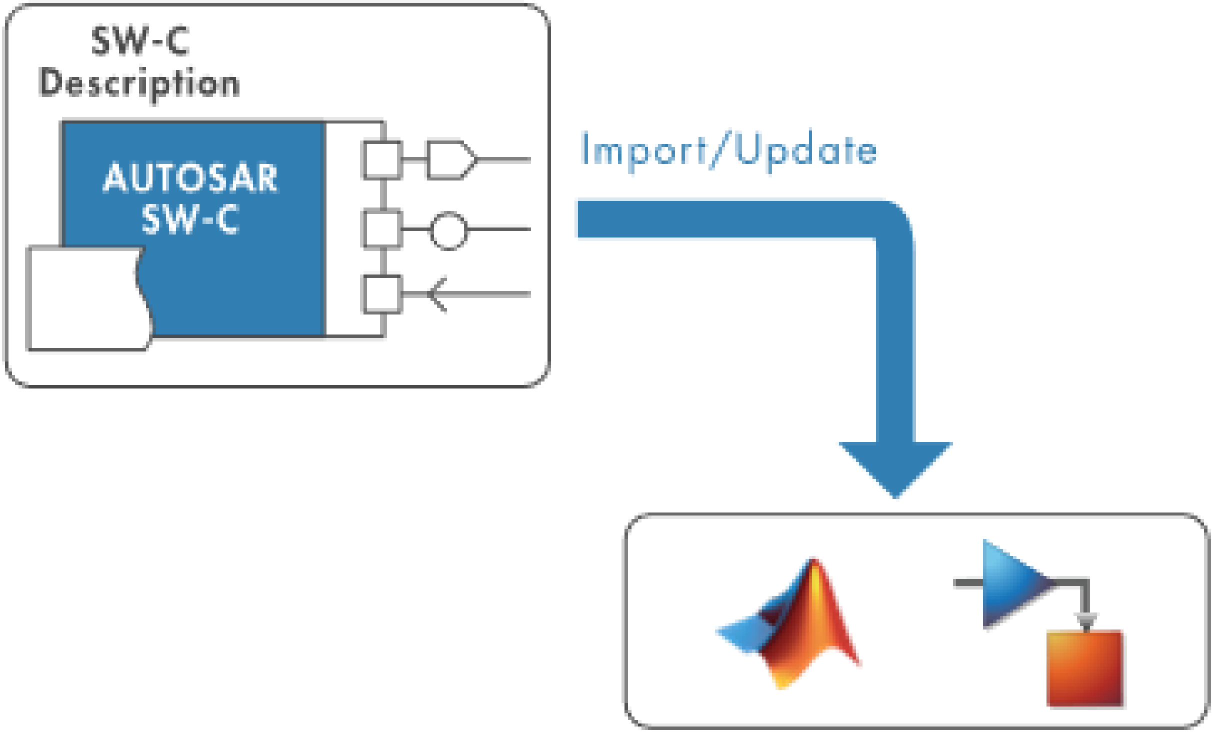
Basic Software

Hardware

Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
  - AUTOSAR Coder Dictionary
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
- Simulink for Adaptive Platform

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

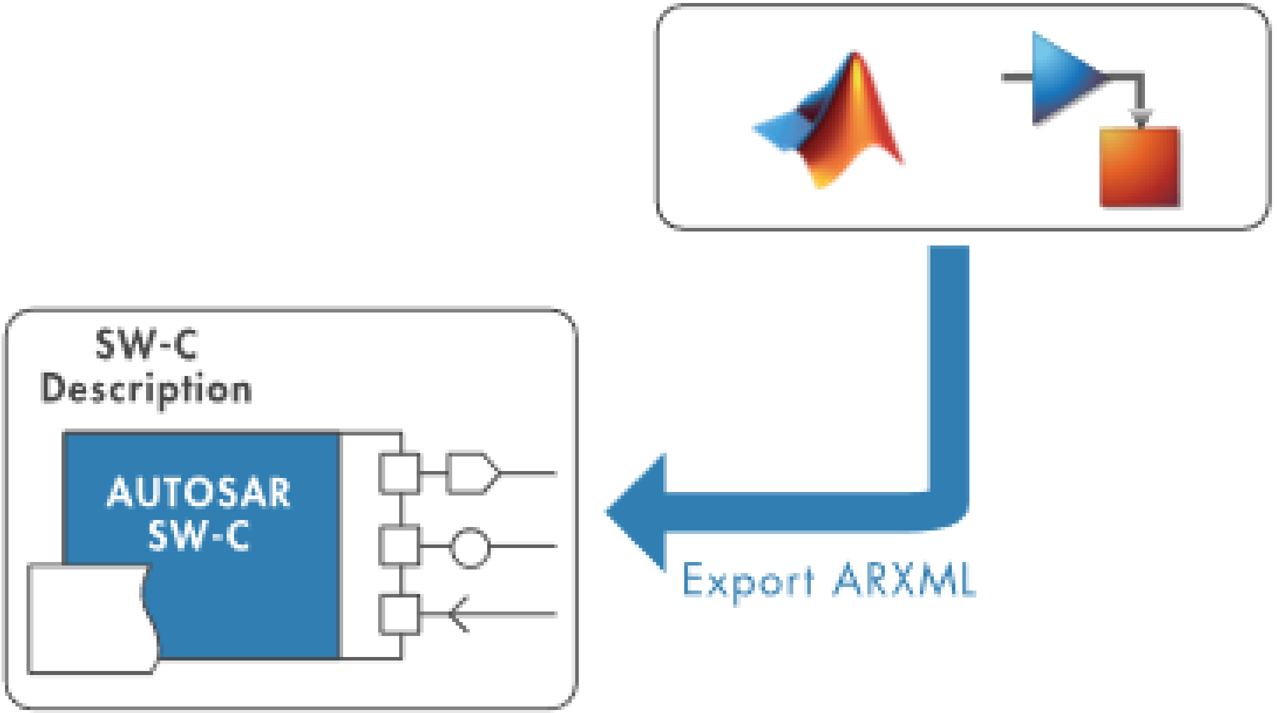


1. Import SW-C description (arxml) & create Simulink model

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

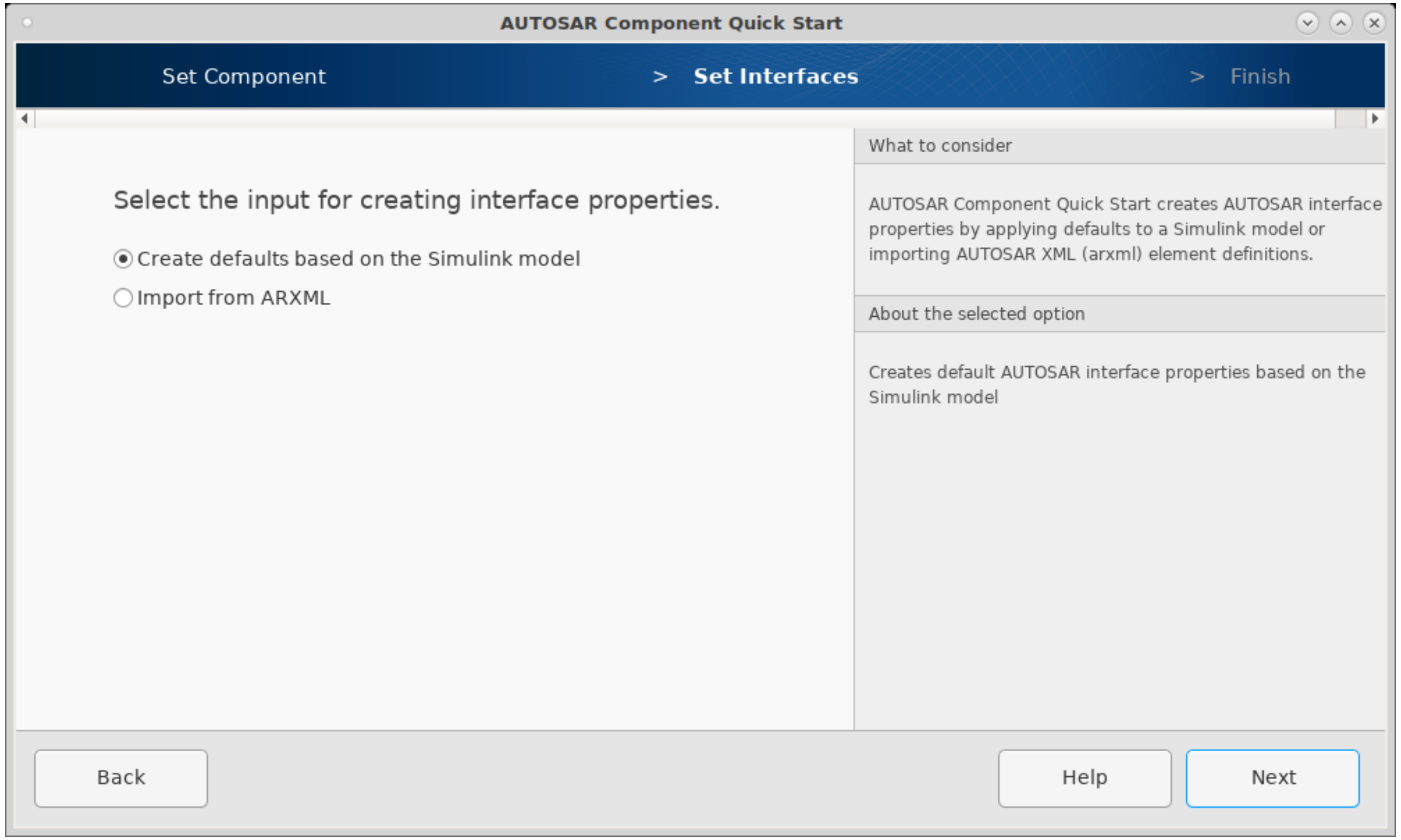
2. Elaborate SW-C Design, implement & generate code from model

It’s quick & easy to configure a Simulink model for AUTOSAR



1. Start with a Simulink model

It’s quick & easy to configure a Simulink model for AUTOSAR



AUTOSAR Component Quick Start
Set Component > Set Interfaces > Finish

Select the input for creating interface properties.

● Create defaults based on the Simulink model

○ Import from ARXML

What to consider

AUTOSAR Component Quick Start creates AUTOSAR interface properties by applying defaults to a Simulink model or importing AUTOSAR XML (arxml) element definitions.

About the selected option

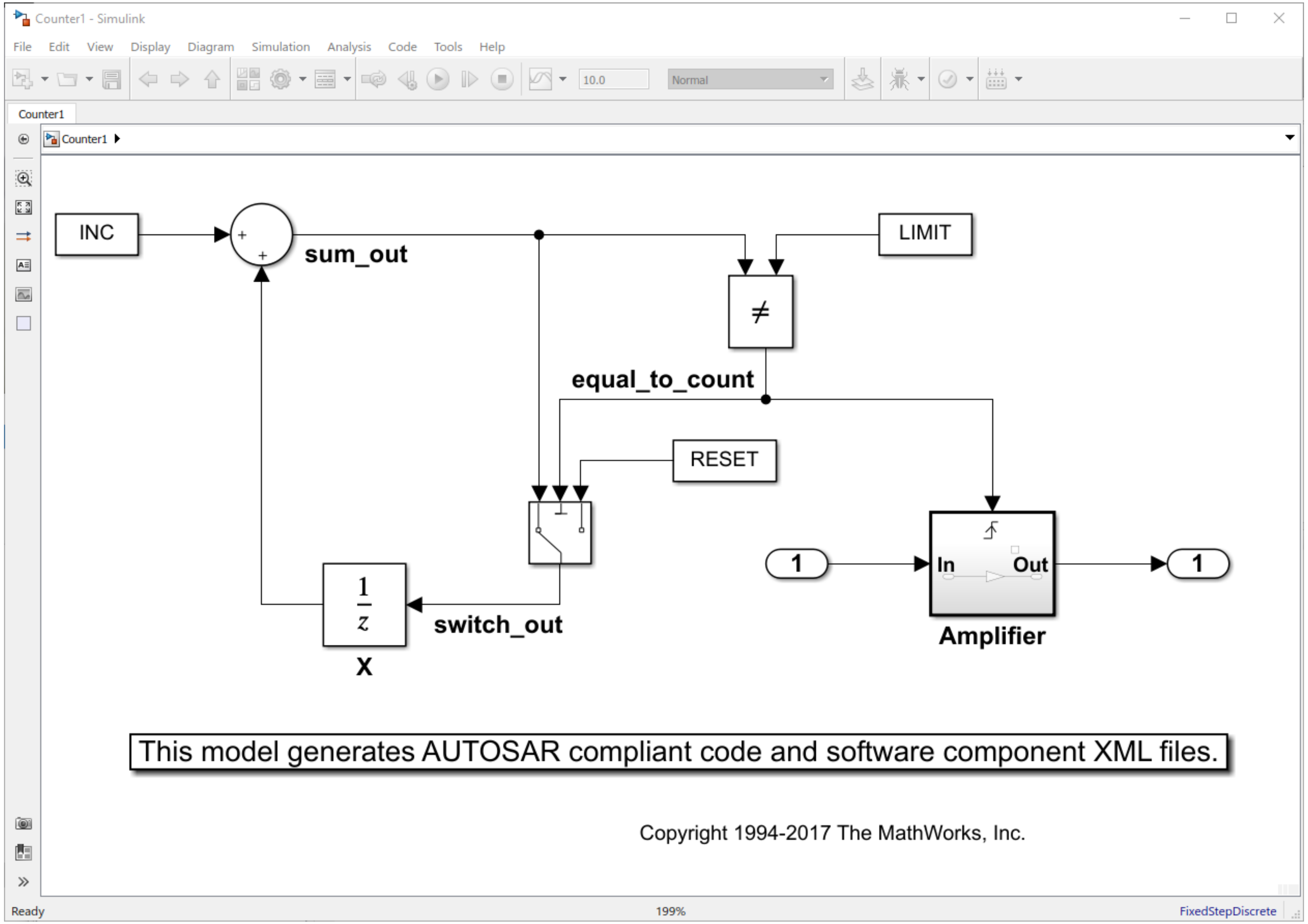
Creates default AUTOSAR interface properties based on the Simulink model

1. Start with a Simulink model

2. Click the AUTOSAR Component Quick Start App

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

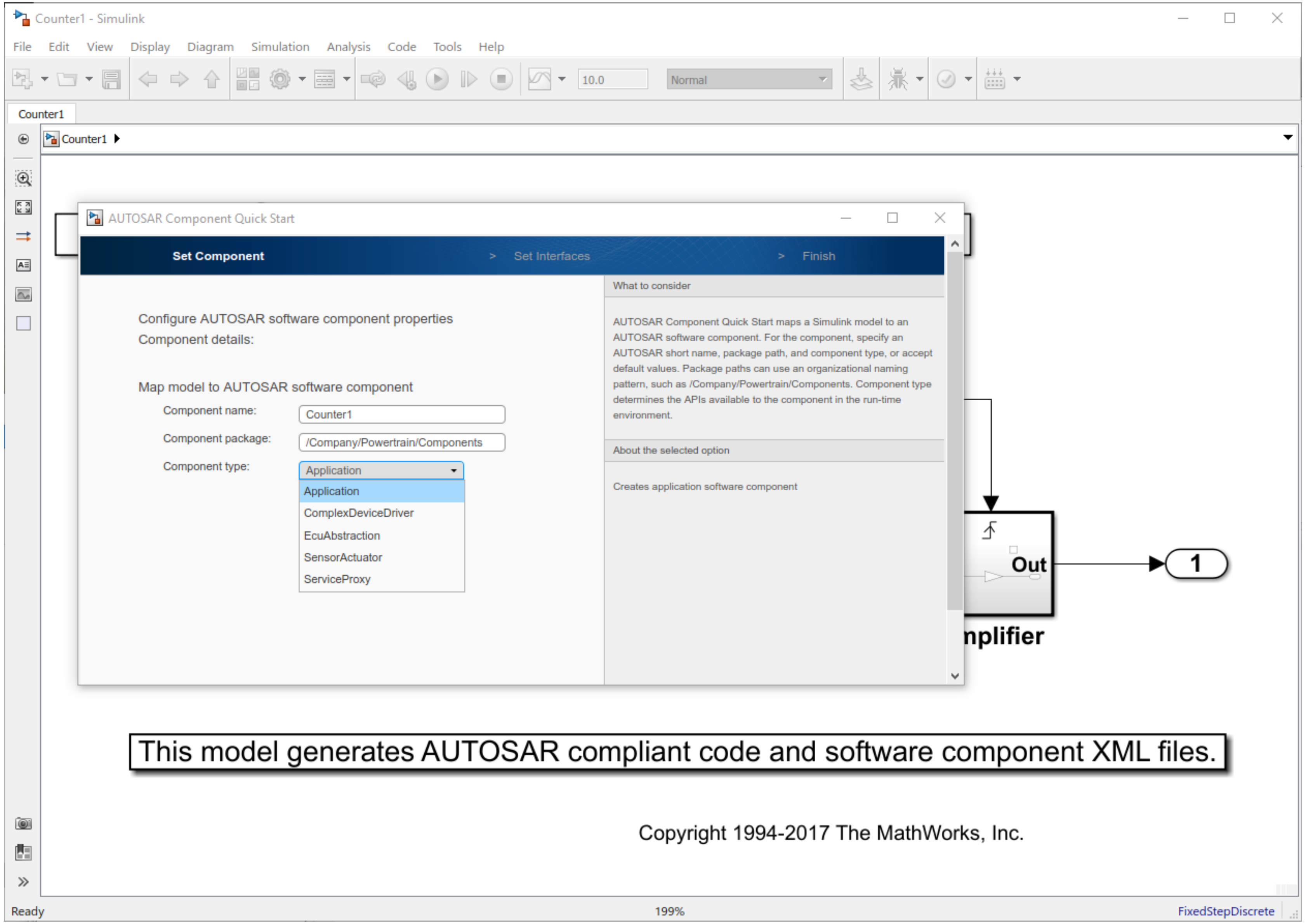
Example of Configuring a model for AUTOSAR



This model generates AUTOSAR compliant code and software component XML files.

Copyright 1994-2017 The MathWorks, Inc.

Launch Quick Start


Counter1 - Simulink
File Edit View Display Diagram Simulation Analysis Code Tools Help
10.0
Normal
Counter1
Counter1
AUTOSAR Component Quick Start
Set Component > Set Interfaces > Finish
Out
1
nplifier
Ready
199%
FixedStepDiscrete

Configure AUTOSAR software component properties

Component details:

Map model to AUTOSAR software component

Component name: Counter1

Component package: /Company/Powertrain/Components

Component type: Application

- Application
- ComplexDeviceDriver
- EcuAbstraction
- SensorActuator
- ServiceProxy

What to consider

AUTOSAR Component Quick Start maps a Simulink model to an AUTOSAR software component. For the component, specify an AUTOSAR short name, package path, and component type, or accept default values. Package paths can use an organizational naming pattern, such as /Company/Powertrain/Components. Component type determines the APIs available to the component in the run-time environment.

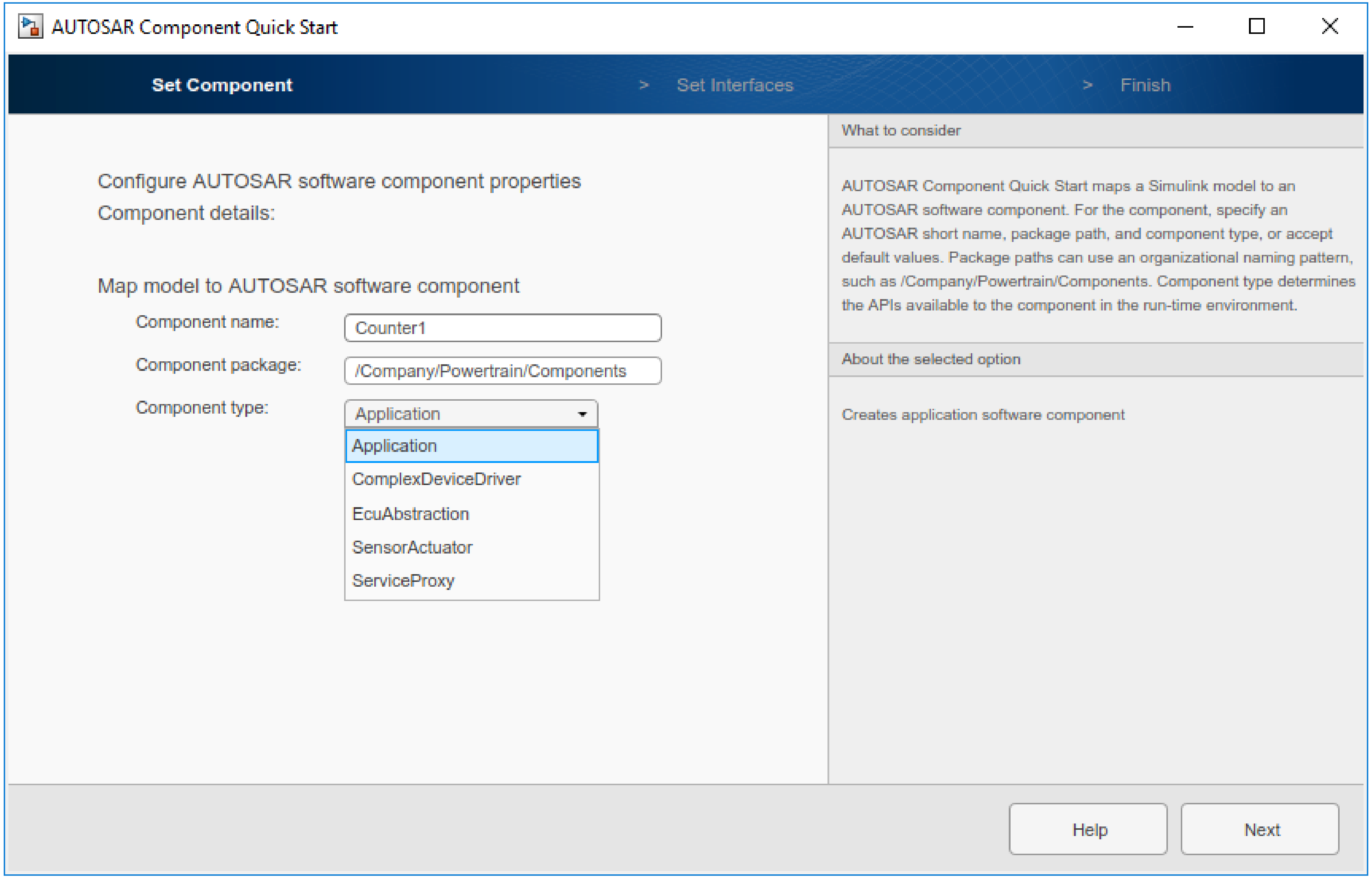
About the selected option

Creates application software component

This model generates AUTOSAR compliant code and software component XML files.

Copyright 1994-2017 The MathWorks, Inc.

AUTOSAR Quick Start - Set Component Type



AUTOSAR Component Quick Start

Configure AUTOSAR software component properties

Component details:

Map model to AUTOSAR software component

Component name: Counter1

Component package: /Company/Powertrain/Components

Component type: Application

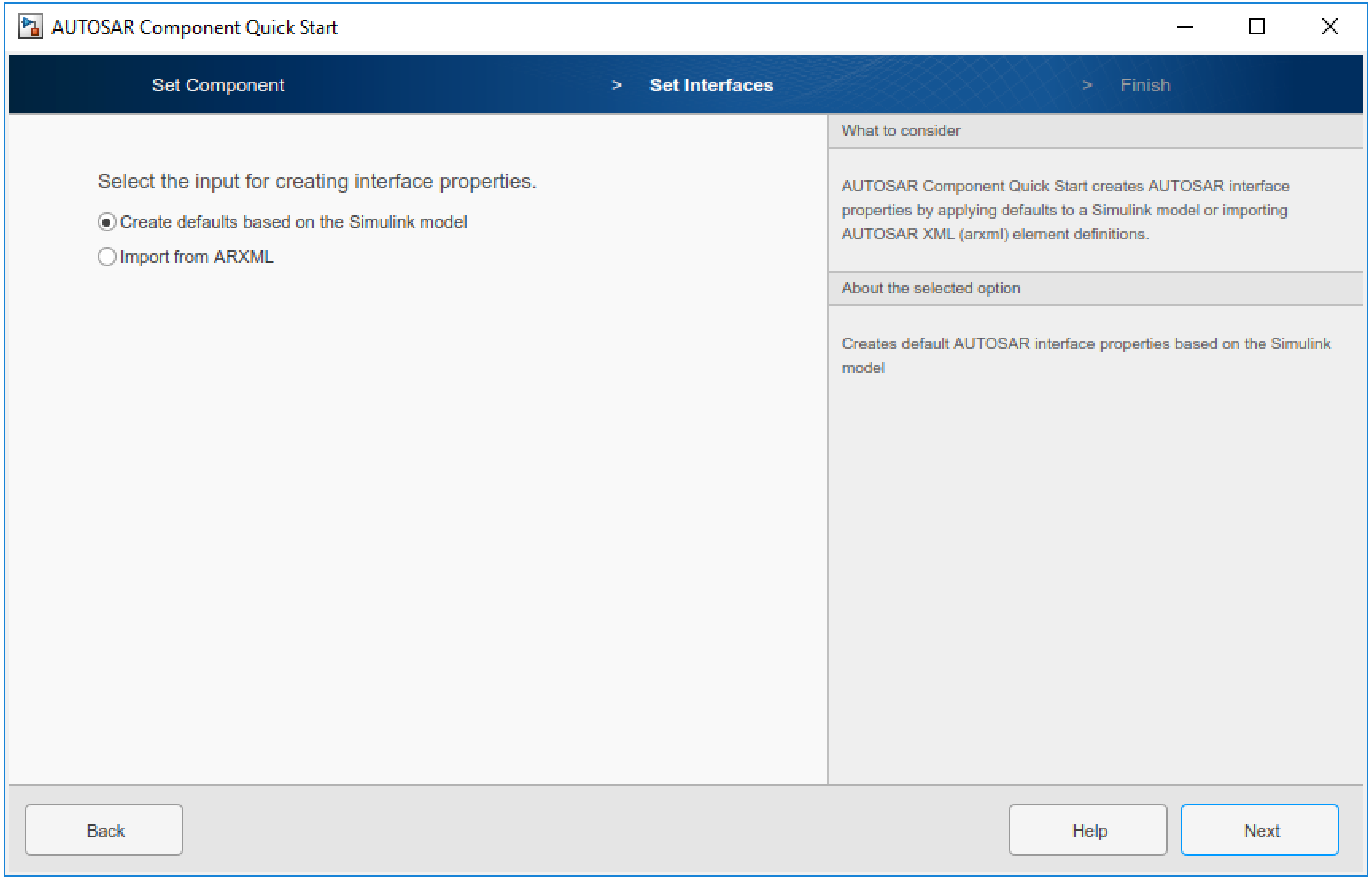
What to consider

AUTOSAR Component Quick Start maps a Simulink model to an AUTOSAR software component. For the component, specify an AUTOSAR short name, package path, and component type, or accept default values. Package paths can use an organizational naming pattern, such as /Company/Powertrain/Components. Component type determines the APIs available to the component in the run-time environment.

About the selected option

Creates application software component

Quick Start – Set Interfaces



Select the input for creating interface properties.

● Create defaults based on the Simulink model

○ Import from ARXML

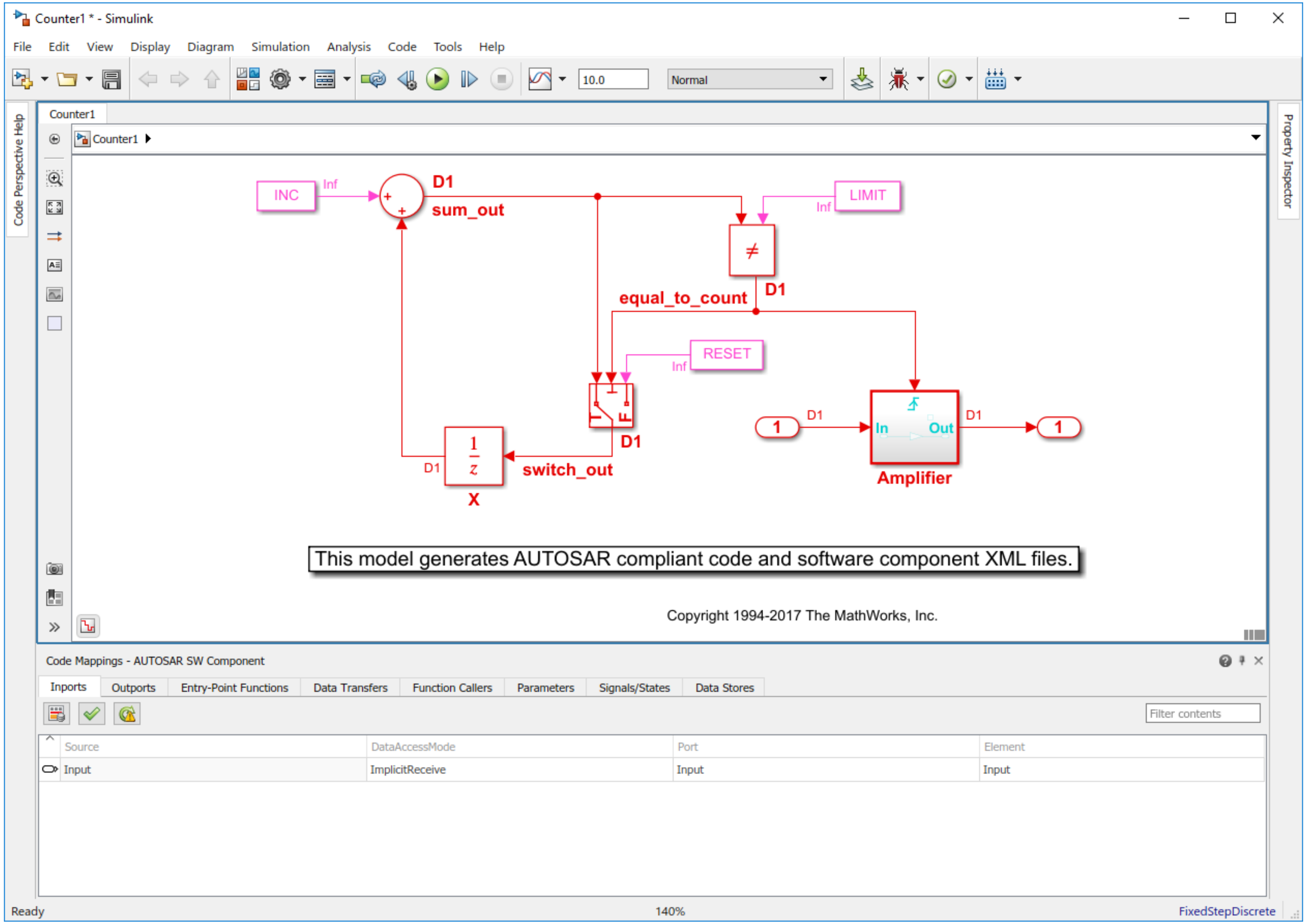
What to consider

AUTOSAR Component Quick Start creates AUTOSAR interface properties by applying defaults to a Simulink model or importing AUTOSAR XML (arxml) element definitions.

About the selected option

Creates default AUTOSAR interface properties based on the Simulink model

Once Quick Start is finished, You can view the configuration



This model generates AUTOSAR compliant code and software component XML files.

Copyright 1994-2017 The MathWorks, Inc.

Code Mappings - AUTOSAR SW Component

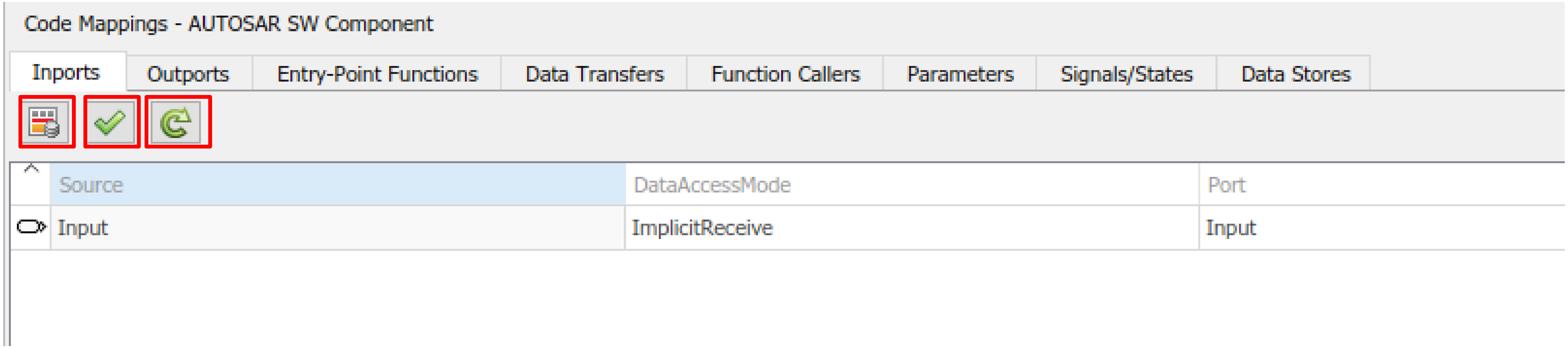
Inports | Outports | Entry-Point Functions | Data Transfers | Function Callers | Parameters | Signals/States | Data Stores

Source	DataAccessMode	Port	Element
Input	ImplicitReceive	Input	Input

AUTOSAR Code Mappings

Sync Model and Code Mappings

Validate Mappings & AUTOSAR Attributes

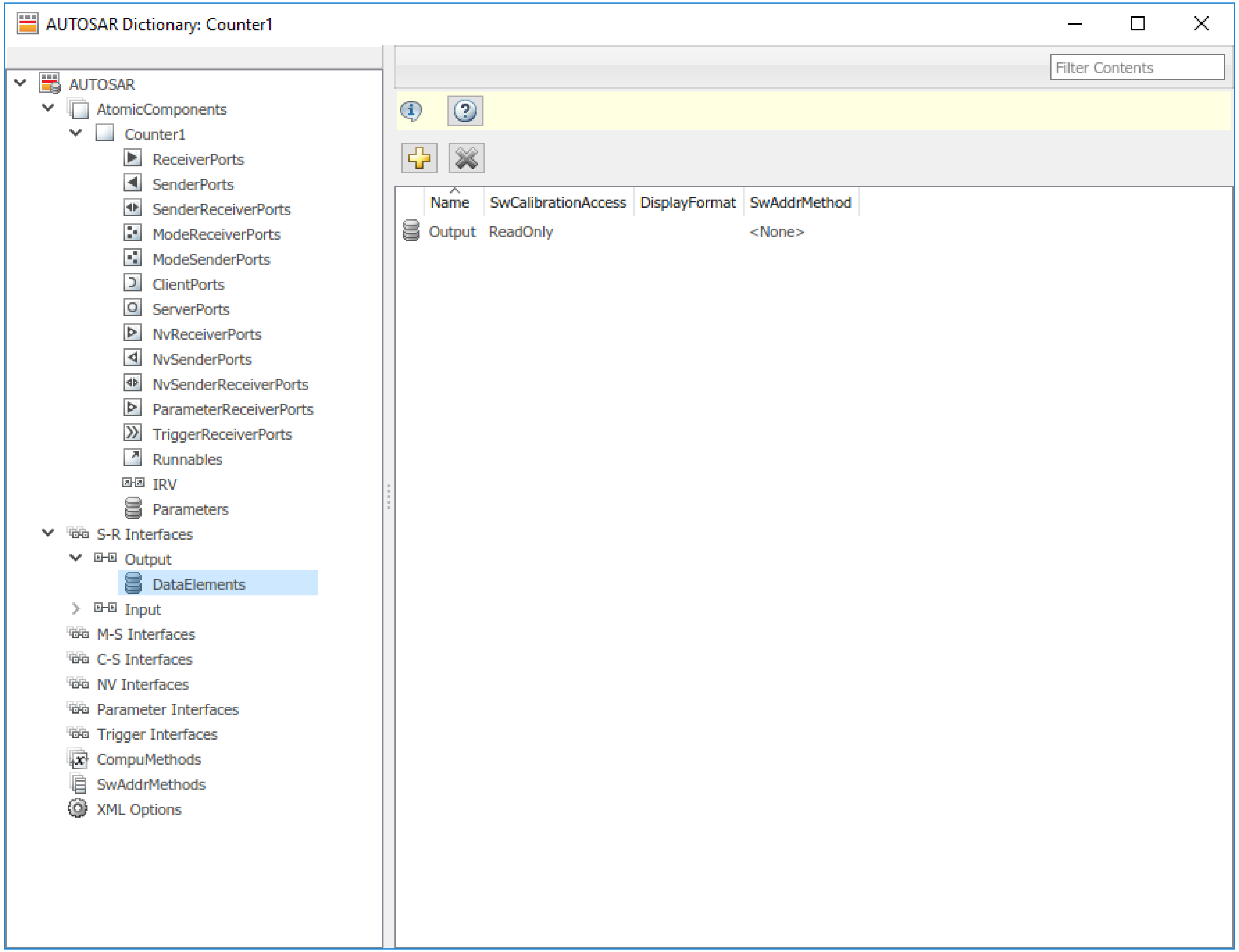


Code Mappings - AUTOSAR SW Component

Source	DataAccessMode	Port
Input	ImplicitReceive	Input

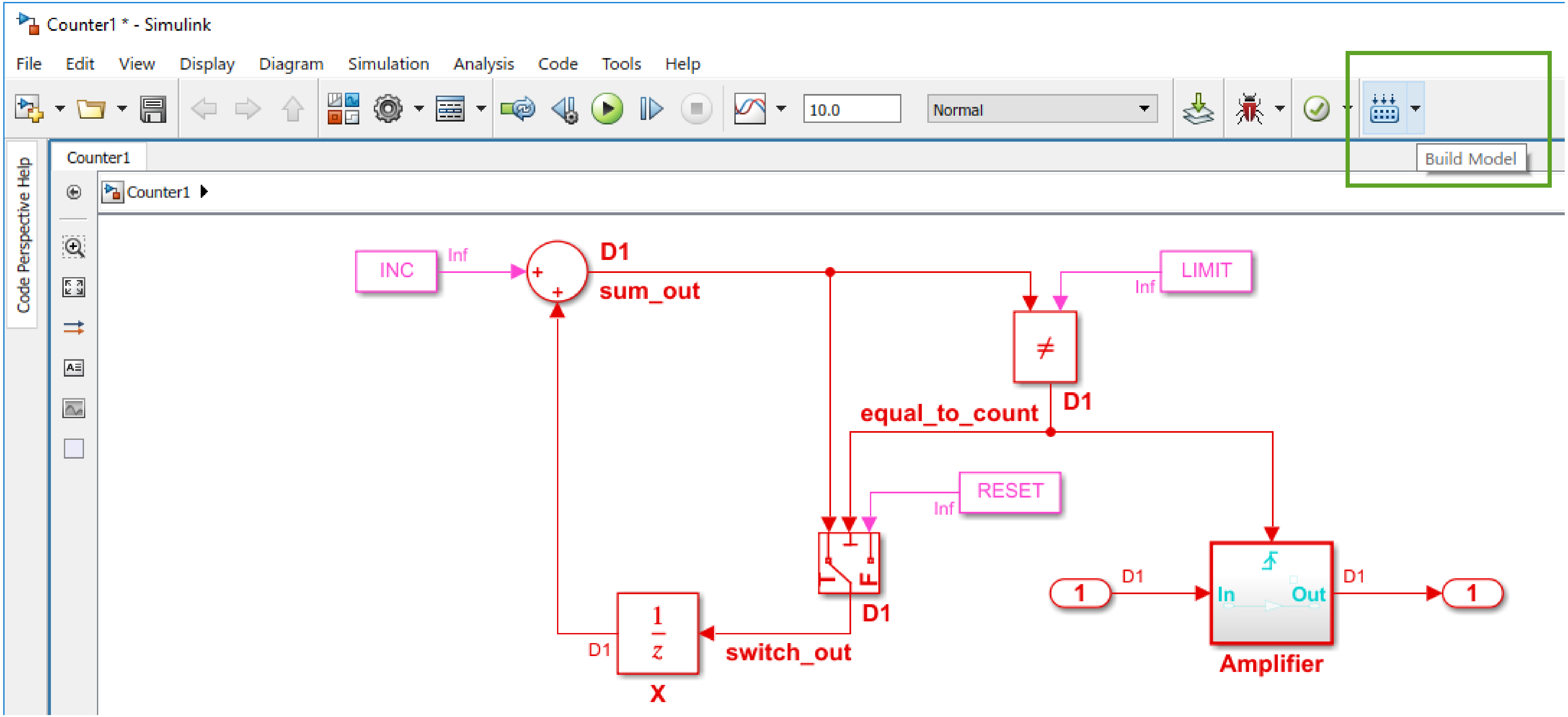
Launch AUTOSAR Dictionary

Launch the AUTOSAR Dictionary

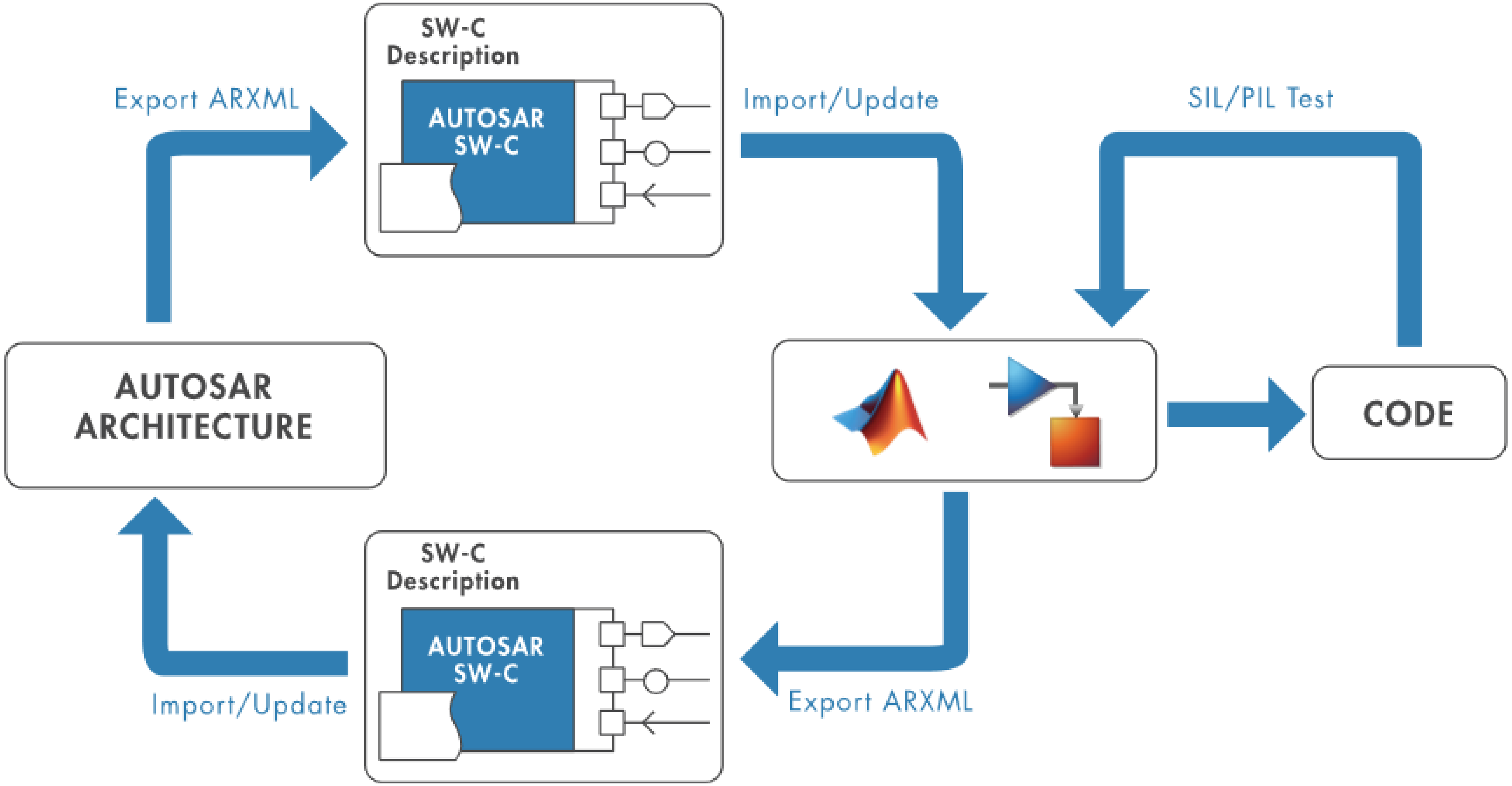


Name	SwCalibrationAccess	DisplayFormat	SwAddrMethod
Output	ReadOnly		<None>

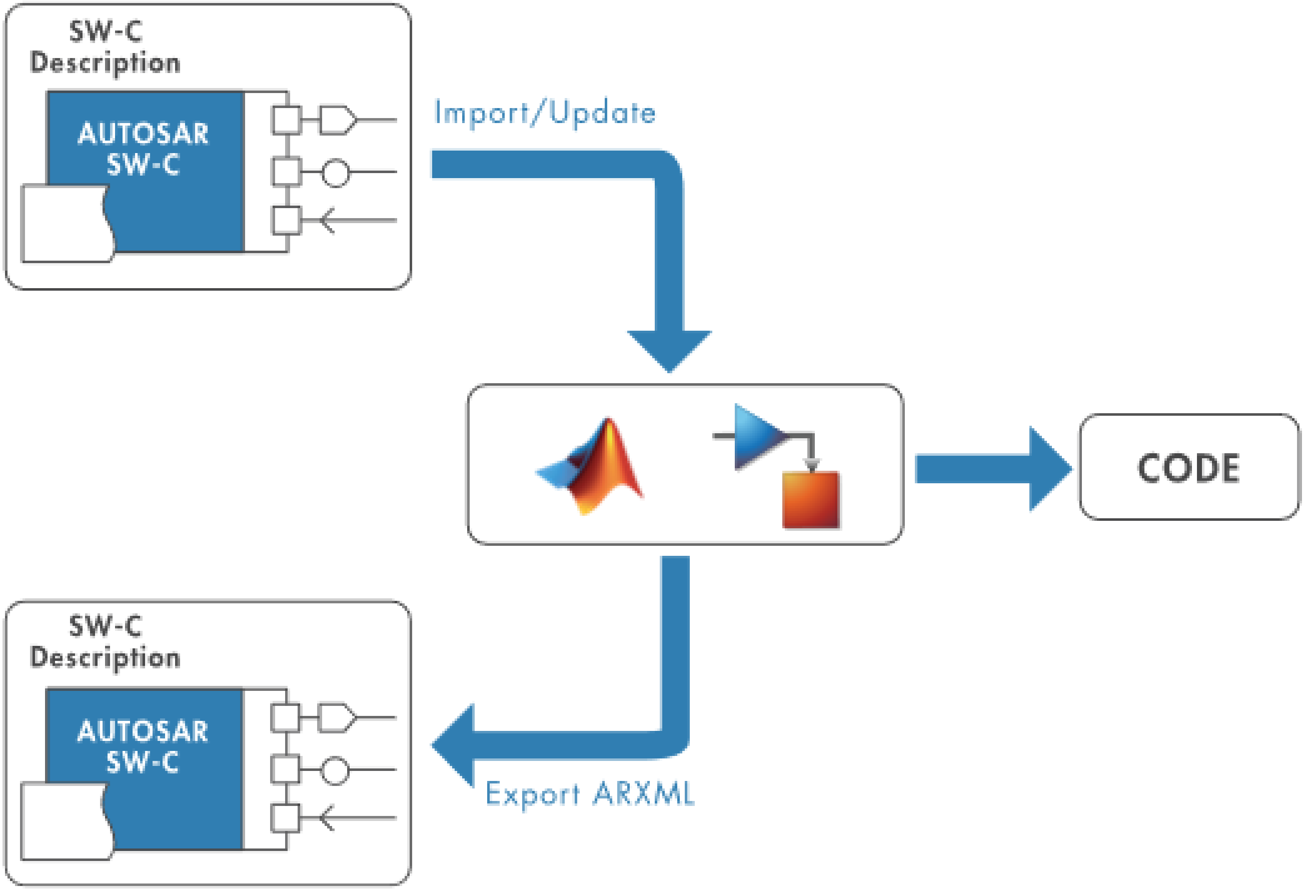
Once Configured, the user can generate AUTOSAR complaint code



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



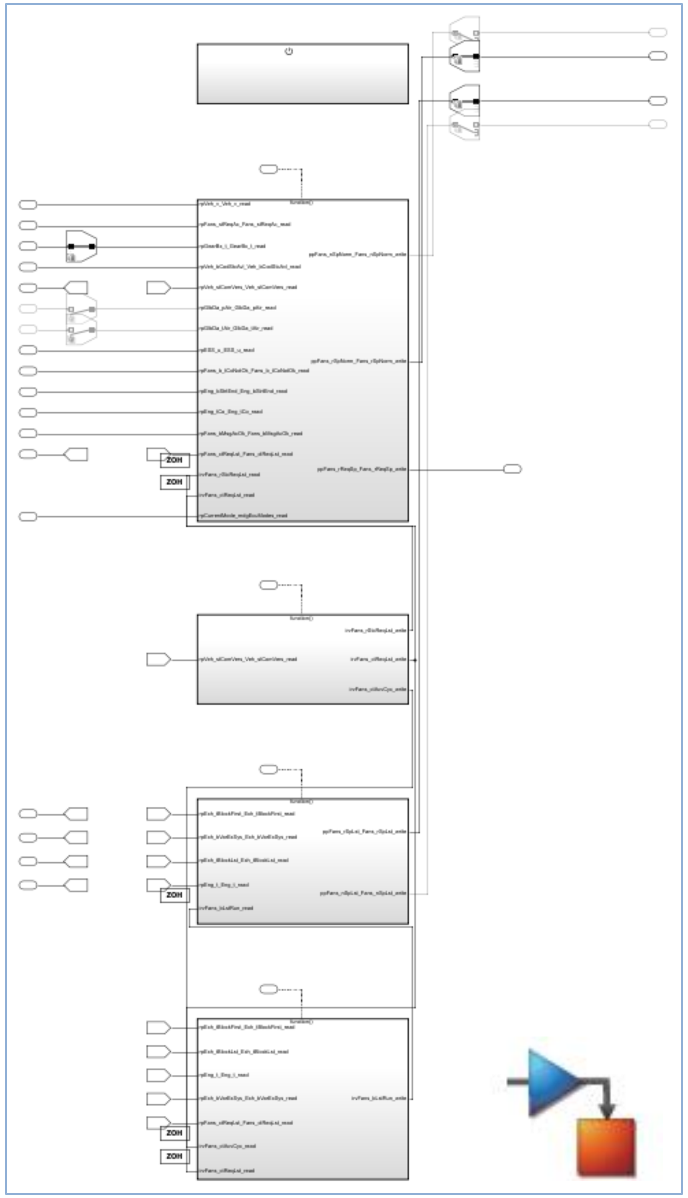
Now we can focus on modeling



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?

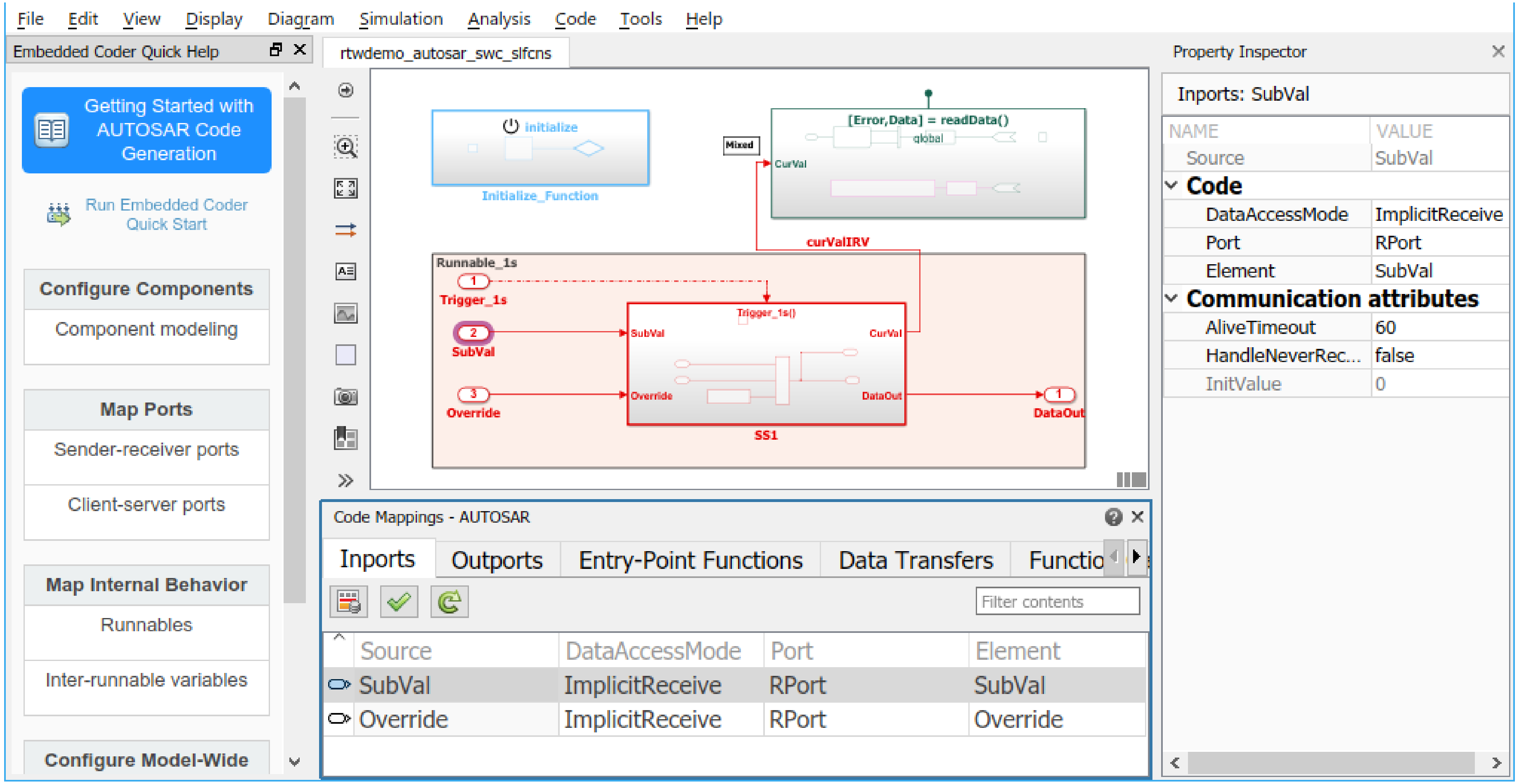
2) How do I change my AUTOSAR properties in the model?

3) Where do I get more information/help?

Introducing AUTOSAR “perspective” in a Simulink model

Quick Help

Help on configuring model for AUTOSAR



Getting Started with AUTOSAR Code Generation

Run Embedded Coder Quick Start

Configure Components

Component modeling

Map Ports

Sender-receiver ports

Client-server ports

Map Internal Behavior

Runnables

Inter-runnable variables

Configure Model-Wide

Inports: SubVal

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

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

Property Inspector

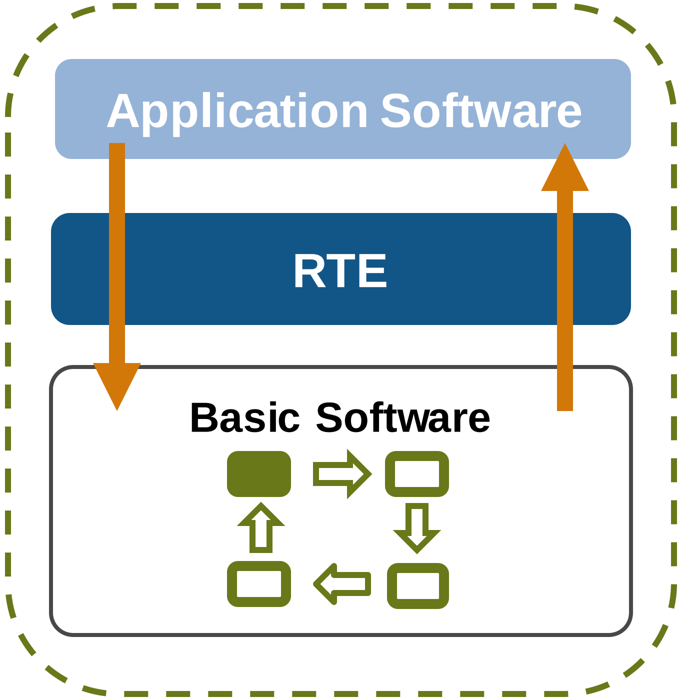
View/Edit AUTOSAR SW-C Properties

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

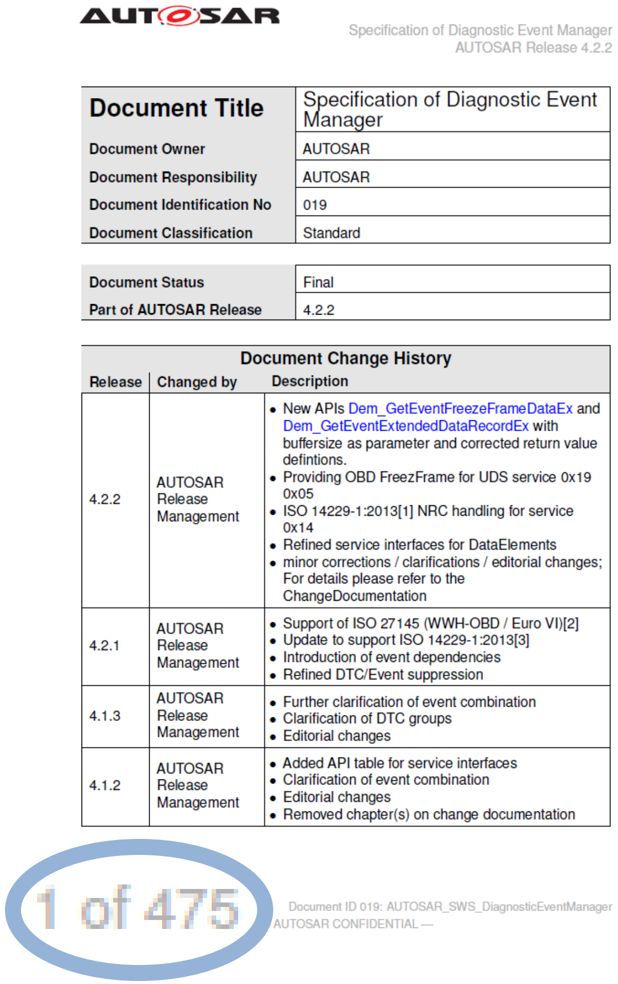


Many calls between application software and basic software

Basic software functionality is highly dynamic

Simulation of basic software reduces development time and improves software quality

BSW library Blocks allows user to Simulate Client / Server Calls



AUTOSAR

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	• New APIs Dem_GetEventFreezeFrameDataEx and Dem_GetEventExtendedDataRecordEx with buffersize as parameter and corrected return value defintions. • Providing OBD FreezFrame 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	• 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	• Further clarification of event combination • Clarification of DTC groups • Editorial changes
4.1.2	AUTOSAR Release Management	• Added API table for service interfaces • Clarification of event combination • Editorial changes • Removed chapter(s) on change documentation

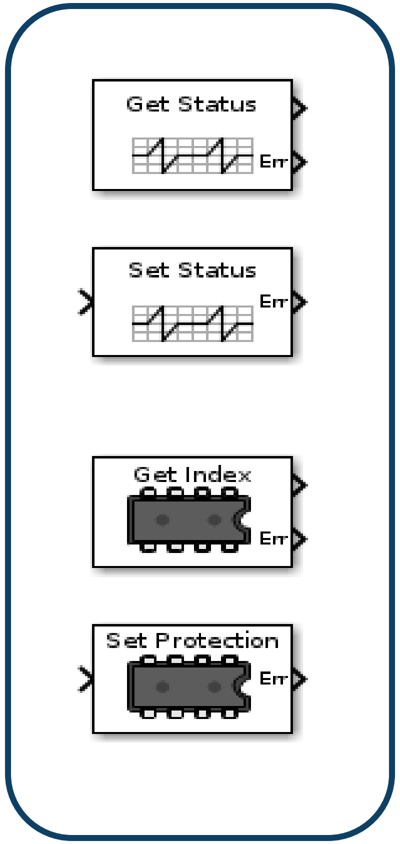
1 of 475

Document ID 019: AUTOSAR_SWS_DiagnosticEventManager
AUTOSAR CONFIDENTIAL —

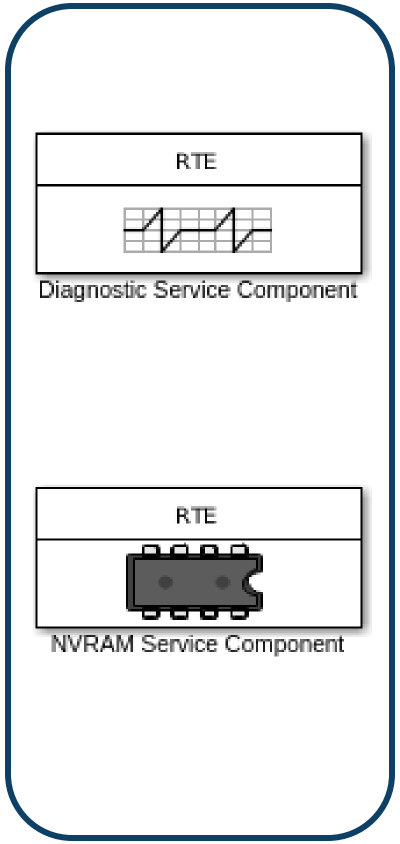
Detailed Specifications of Diagnostic Event Manager

BSW AUTOSAR Specs Encapsulated in

Basic Software Library

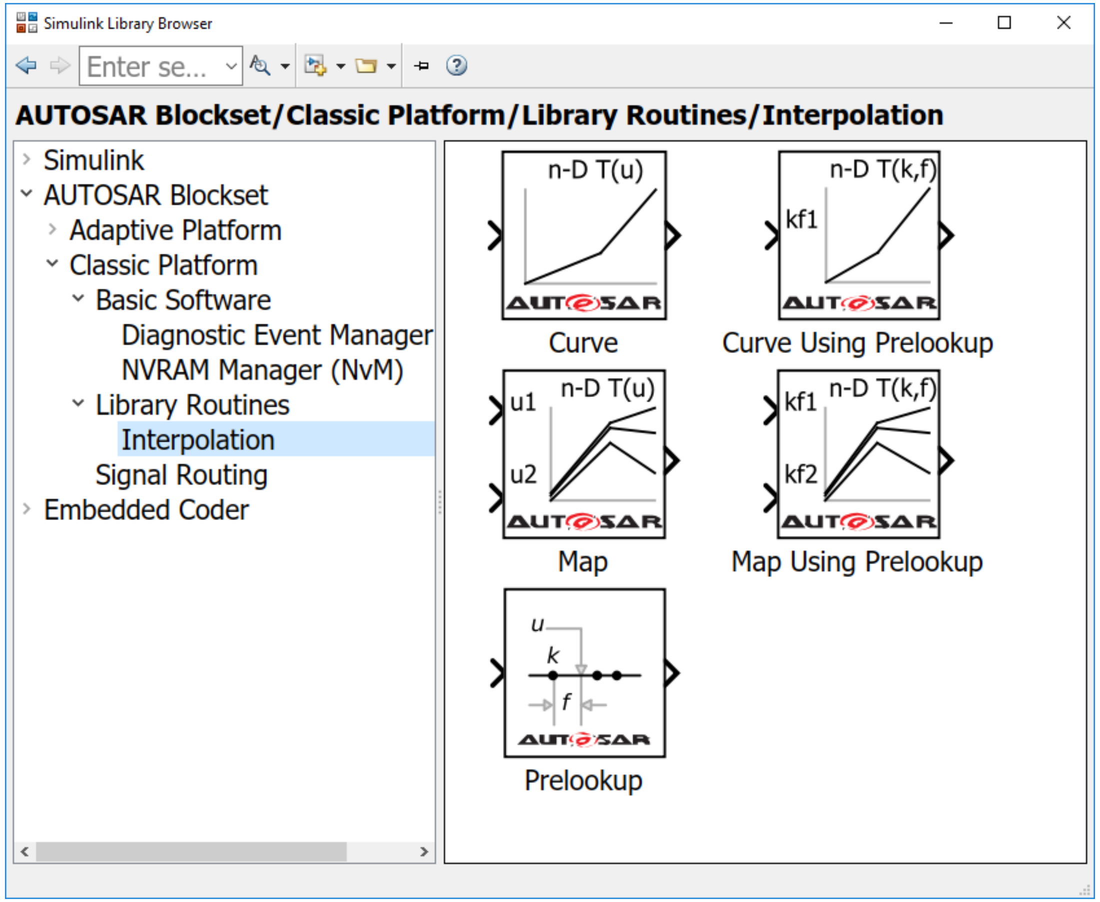


Client Block Resides in SWC Application



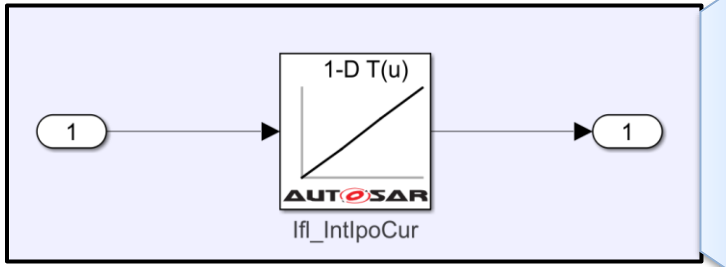
Server Block Resides in Simulation Test Harness

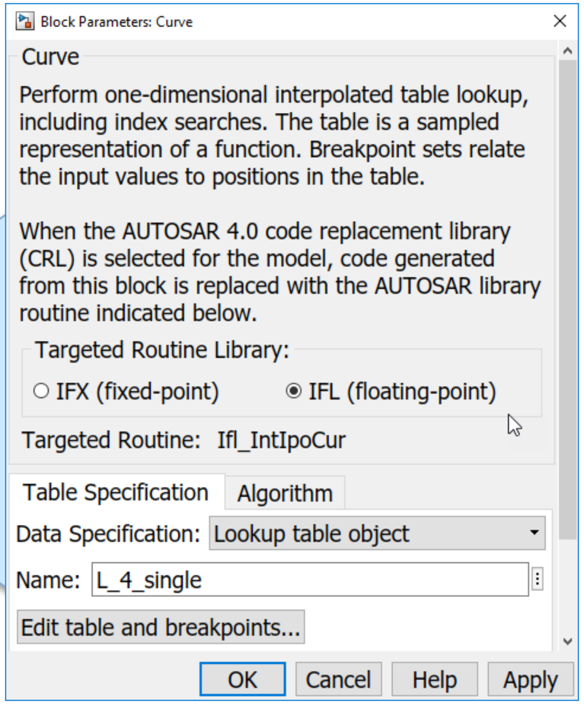
AUTOSAR Library Routines



Simulink Library Browser

AUTOSAR Blockset/Classic Platform/Library Routines/Interpolation





Block Parameters: Curve

Curve

Perform one-dimensional interpolated table lookup, including index searches. The table is a sampled representation of a function. Breakpoint sets relate the input values to positions in the table.

When the AUTOSAR 4.0 code replacement library (CRL) is selected for the model, code generated from this block is replaced with the AUTOSAR library routine indicated below.

Targeted Routine Library:

○ IFX (fixed-point) ● IFL (floating-point)

Targeted Routine: Ifl_IntIpoCur

Table Specification | Algorithm

Data Specification: Lookup table object

Name: L_4_single

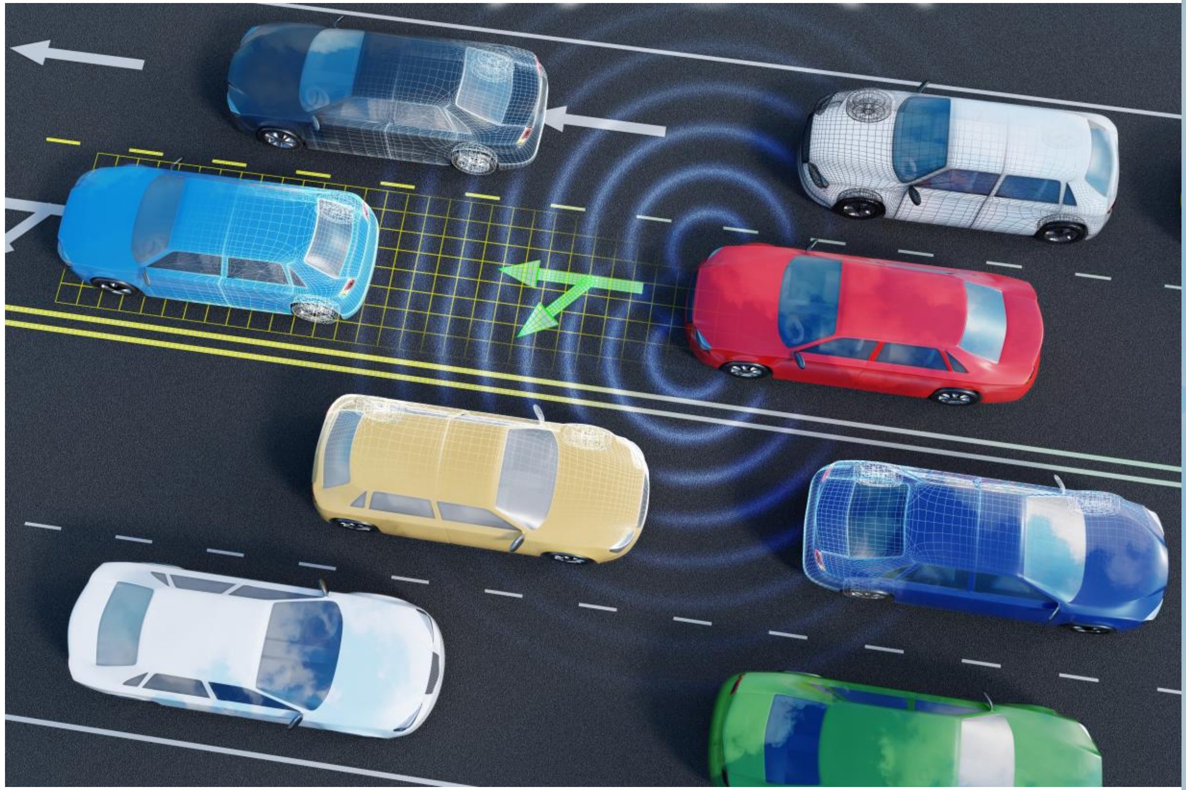
```
Rte_IWrite_Runnable_Step_Out1_Out1(Ifl_IntIpoCur_f32_f32
(Rte_IRead_Runnable_Step_In1_In1(), Rte_CData_L_4_single()->Nx,
Rte_CData_L_4_single()->Bp1, Rte_CData_L_4_single()->Table));
```

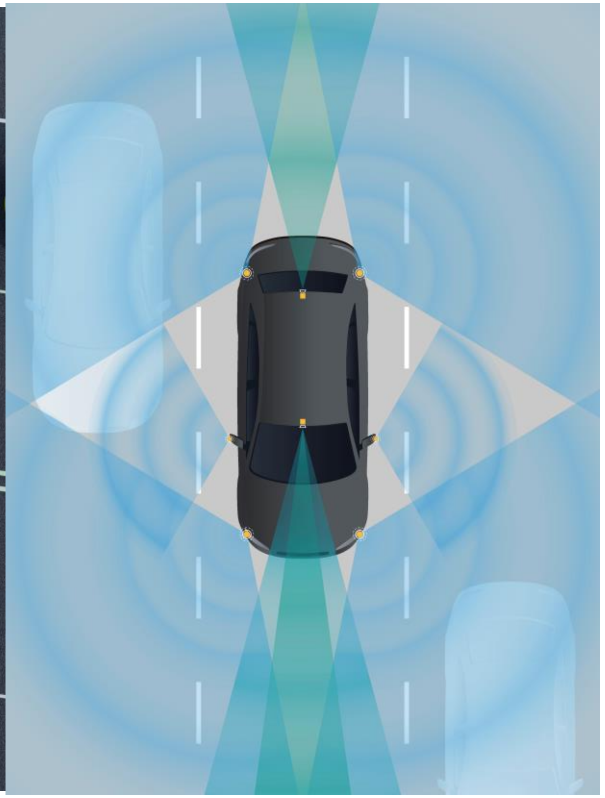
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
  - Motivation for New AUTOSAR Platforms
  - A closer look at the Adaptive layers
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components

Motivation for new AUTOSAR Platforms

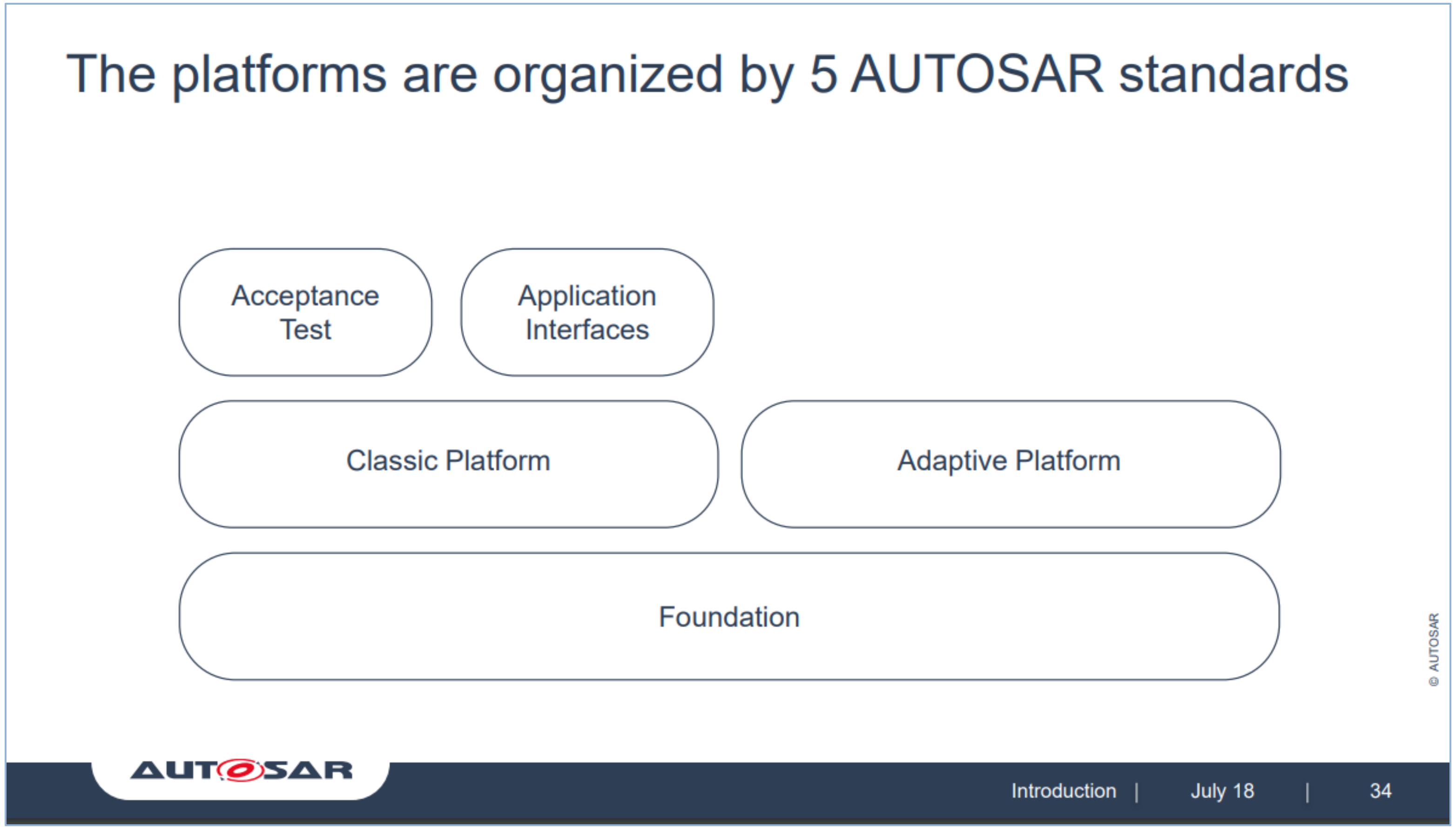
- Main drivers – Automated driving, Car-2-car/infrastructure applications





Expansion of AUTOSAR based on Autonomous Applications

- In 2016 work started on creating these additional AUTOSAR Platforms
- March of 2017 is the first published release of AUTOSAR Adaptive Platform



The platforms are organized by 5 AUTOSAR standards

Acceptance Test

Application Interfaces

Classic Platform

Adaptive Platform

Foundation

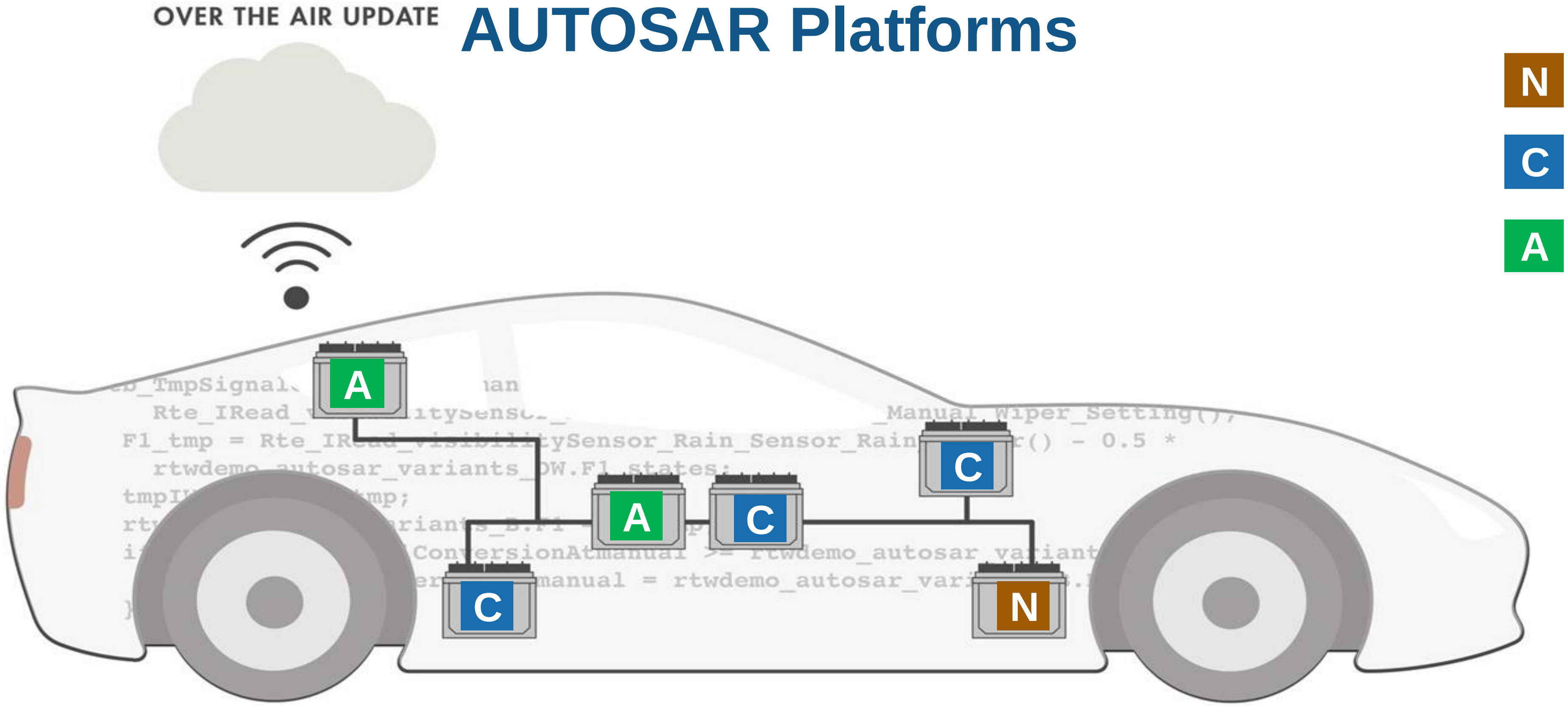
© AUTOSAR

AUTOSAR

Introduction | July 18 | 34

From AUTOSAR.org – AUTOSAR Introduction

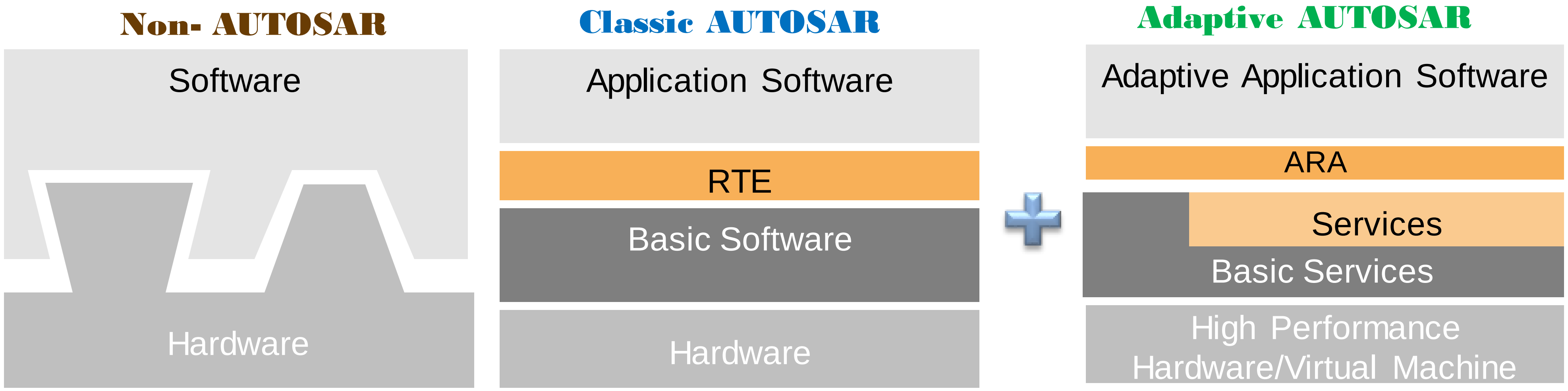
AUTOSAR Platforms



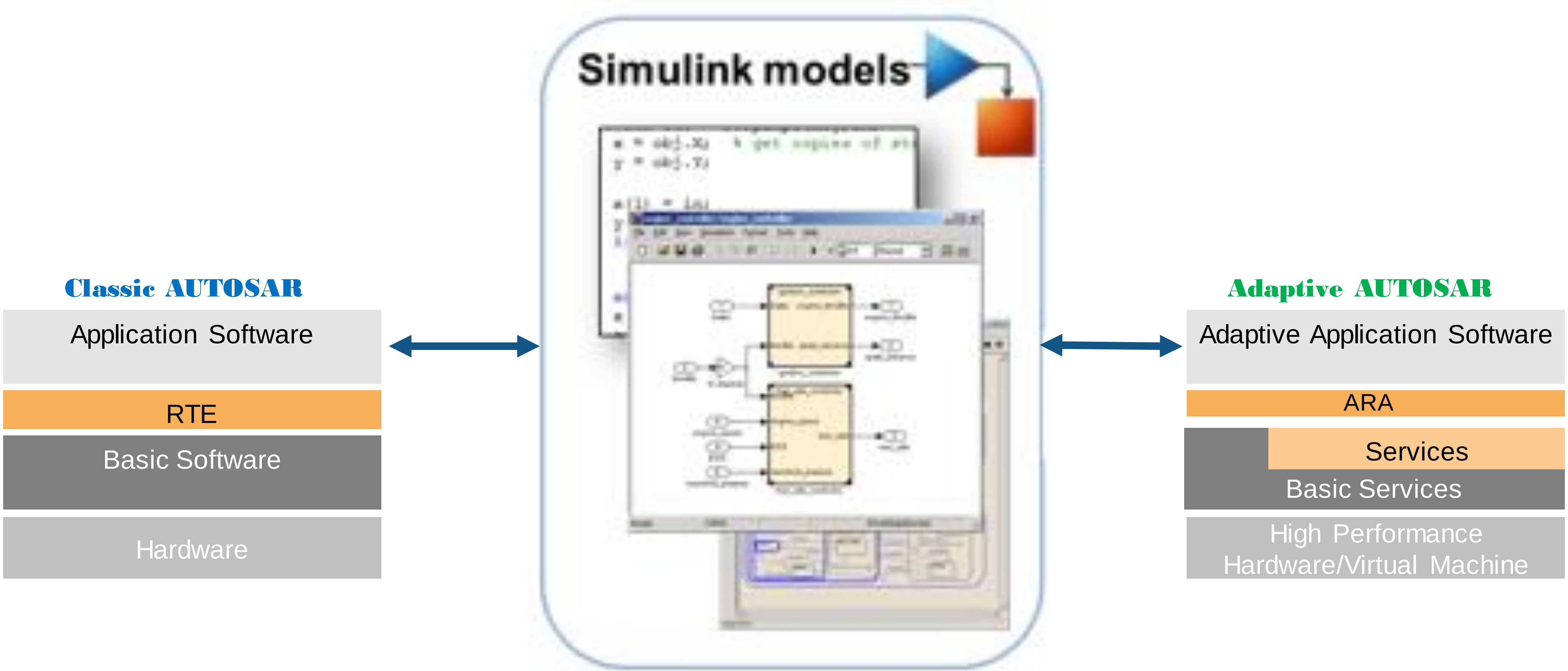
N Non - AUTOSAR

C Classic - AUTOSAR

A Adaptive - AUTOSAR



Either AUTOSAR Platform benefits from Design in Simulink



Classic AUTOSAR

- Application Software
- RTE
- Basic Software
- Hardware

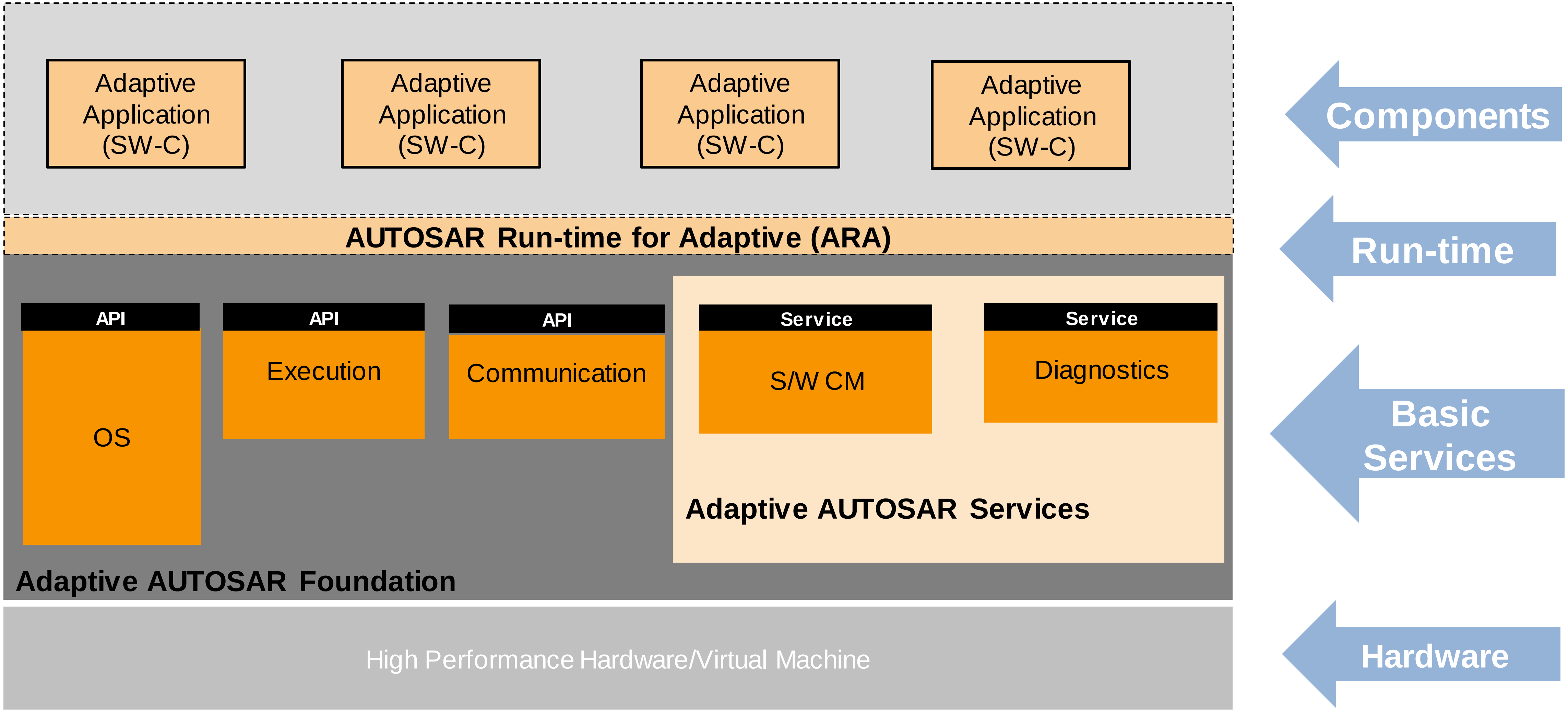
Simulink models

Adaptive AUTOSAR

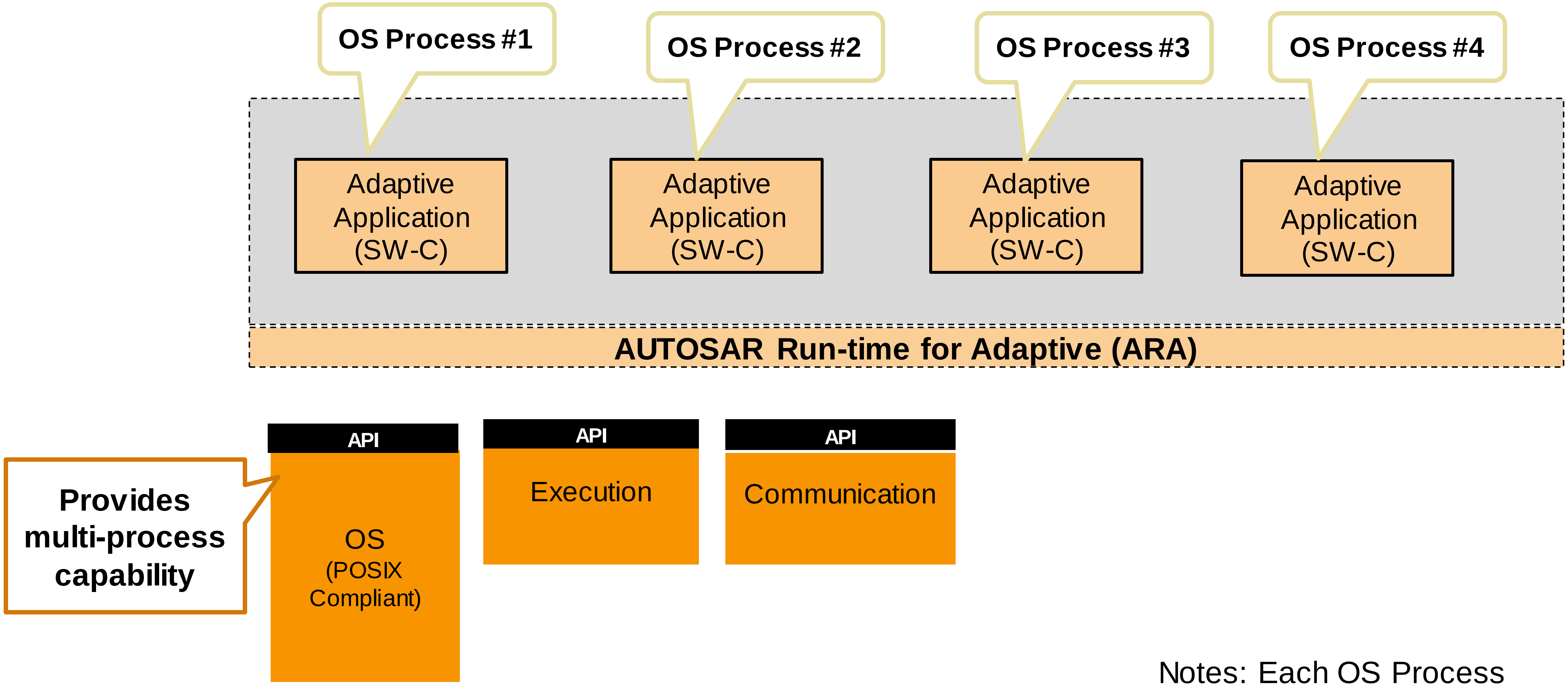
- Adaptive Application Software
- ARA
- Services
- Basic Services
- High Performance Hardware/Virtual Machine

Power of Simulation in the Application Layer aligns well with Algorithm Development

AUTOSAR Layered Software Architecture


Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
AUTOSAR Run-time for Adaptive (ARA)
API
OS
API
Execution
API
Communication
Service
S/W CM
Service
Diagnostics
Adaptive AUTOSAR Services
Adaptive AUTOSAR Foundation
High Performance Hardware/Virtual Machine
Components
Run-time
Basic Services
Hardware

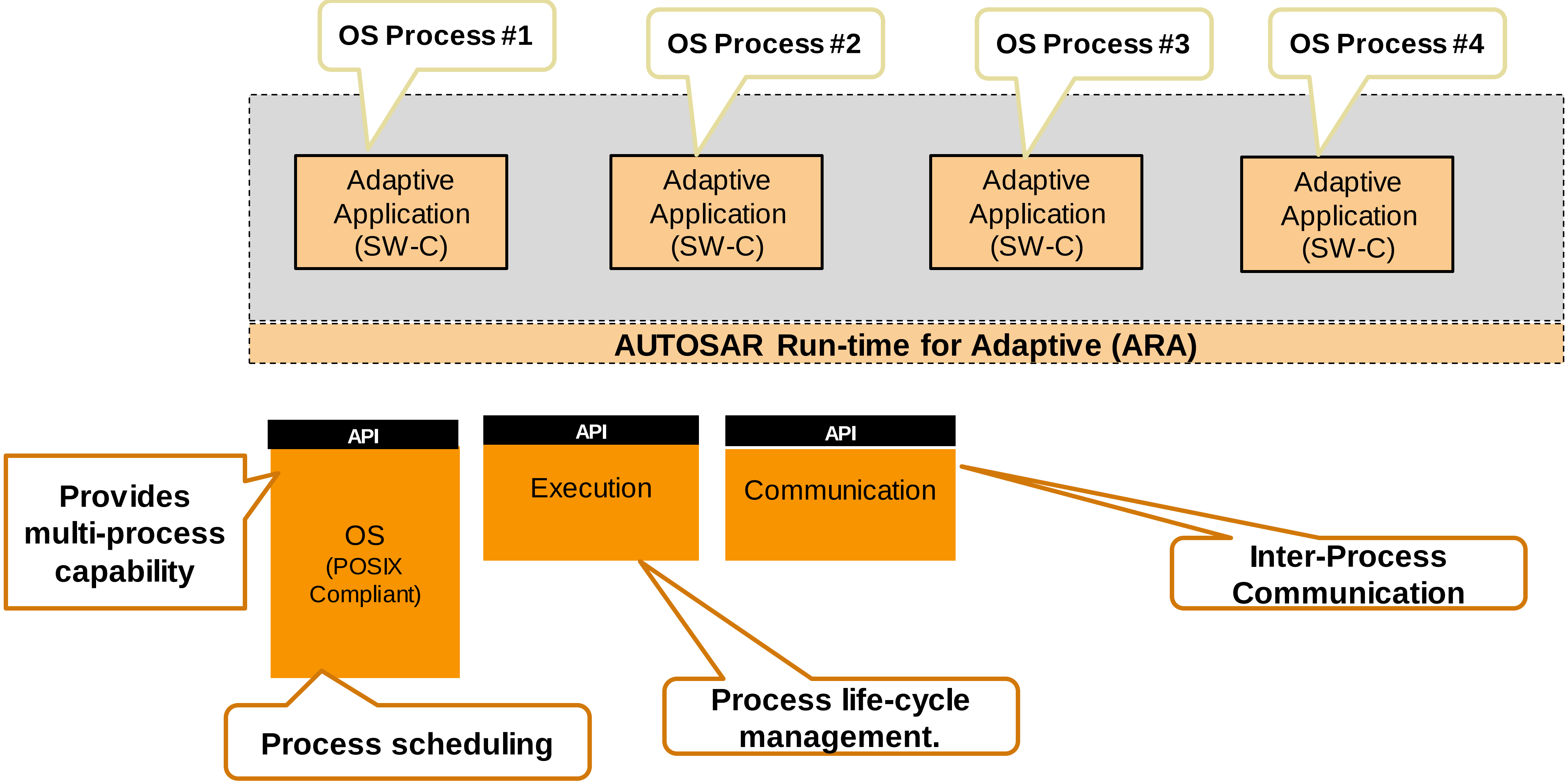
Key Concept #1
Everything is a process .. as in “OS process”


OS Process #1
OS Process #2
OS Process #3
OS Process #4
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
AUTOSAR Run-time for Adaptive (ARA)
API
OS (POSIX Compliant)
API
Execution
API
Communication
Provides multi-process capability

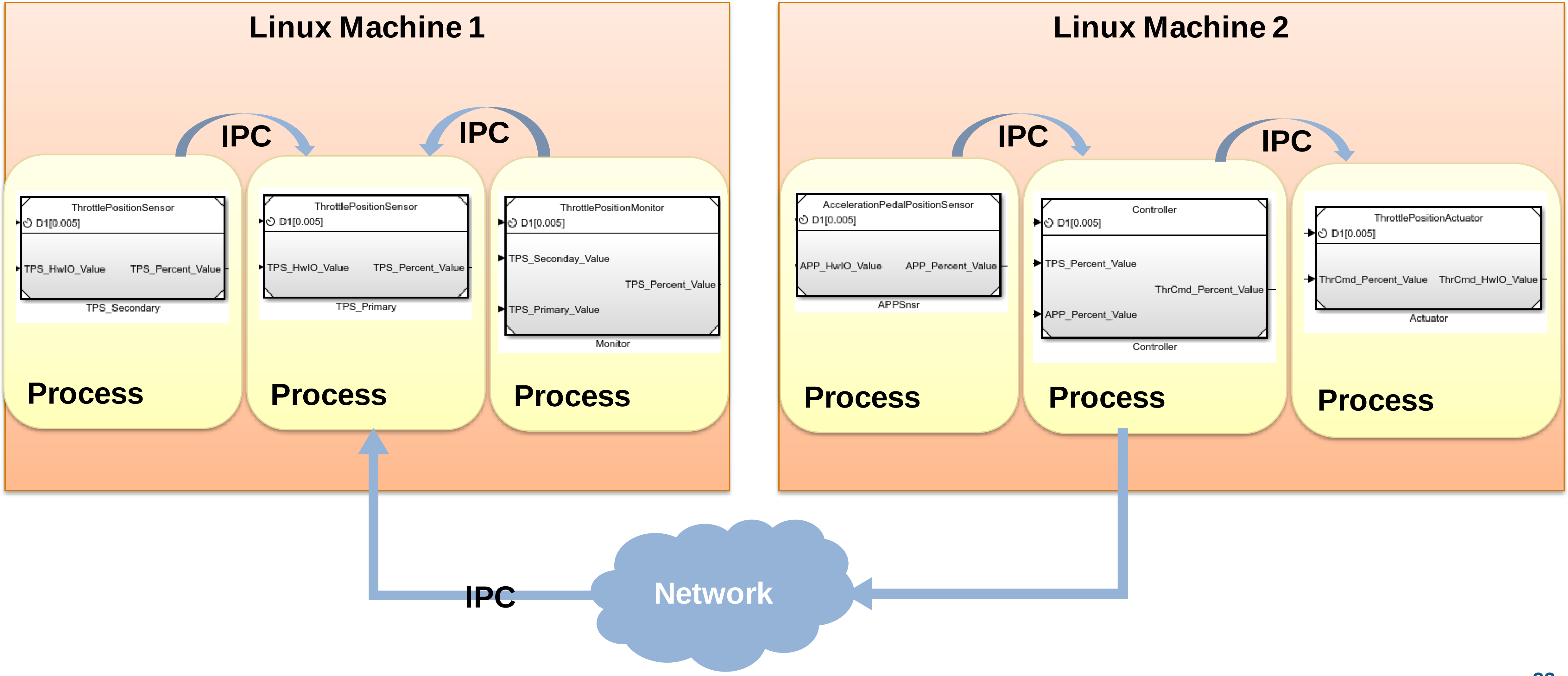
Notes: Each OS Process

- 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”


OS Process #1
OS Process #2
OS Process #3
OS Process #4
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
AUTOSAR Run-time for Adaptive (ARA)
API
API
API
OS (POSIX Compliant)
Execution
Communication
Provides multi-process capability
Process scheduling
Process life-cycle management.
Inter-Process Communication

Key Concept #2
Service-oriented inter-process communication



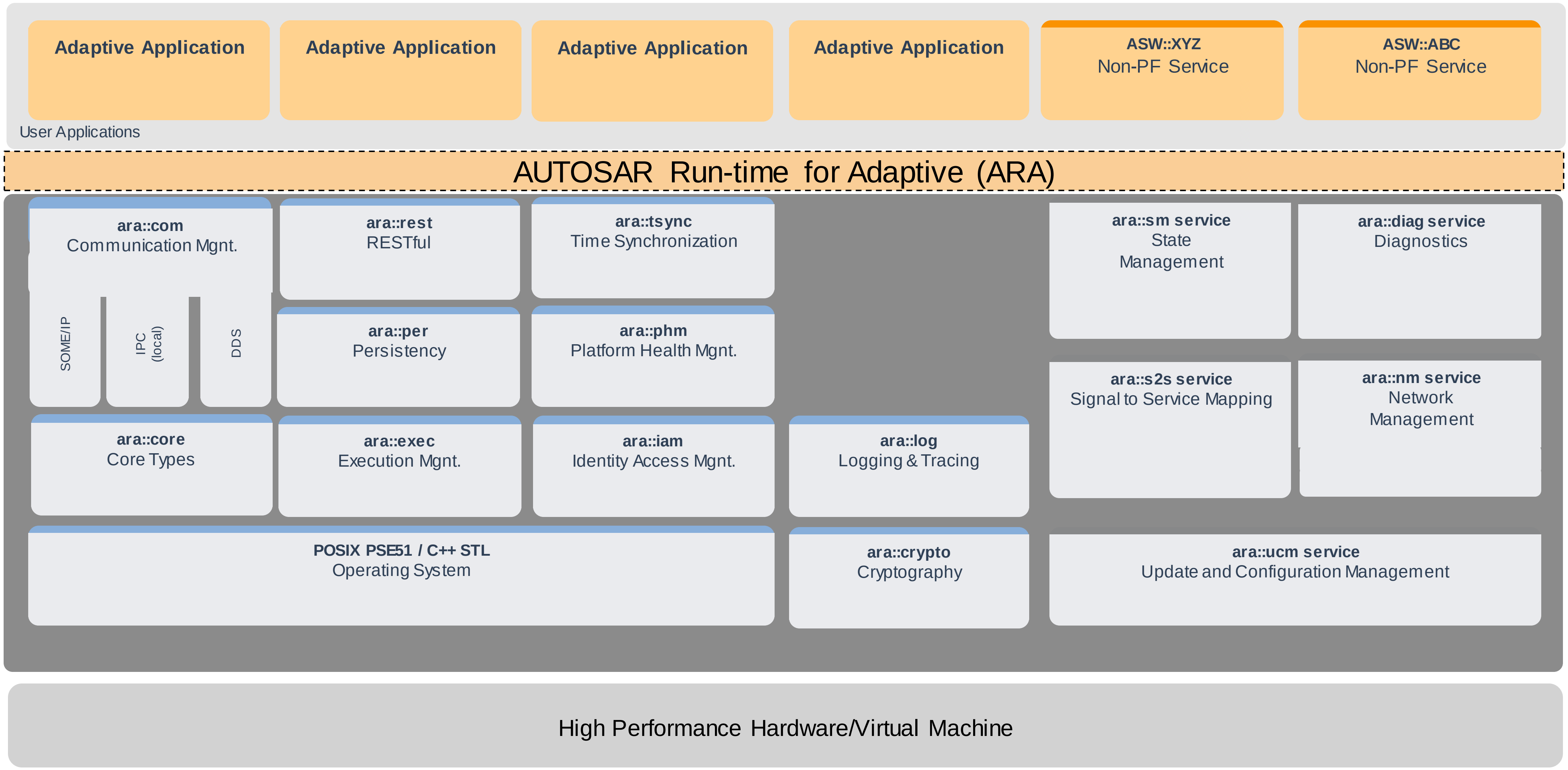
Key Concept #2
Service-oriented communication

- 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 Application

Adaptive Application

Adaptive Application

Adaptive Application

ASW::XYZ
Non-PF Service

ASW::ABC
Non-PF Service

User Applications

AUTOSAR Run-time for Adaptive (ARA)

ara::com
Communication Mgnt.

SOME/IP

IPC
(local)

DDS

ara::rest
RESTful

ara::tsync
Time Synchronization

ara::per
Persistency

ara::phm
Platform Health Mgnt.

ara::sm service
State
Management

ara::diag service
Diagnostics

ara::s2s service
Signal to Service Mapping

ara::nm service
Network
Management

ara::core
Core Types

ara::exec
Execution Mgnt.

ara::iam
Identity Access Mgnt.

ara::log
Logging & Tracing

POSIX PSE51 / C++ STL
Operating System

ara::crypto
Cryptography

ara::ucm service
Update and Configuration Management

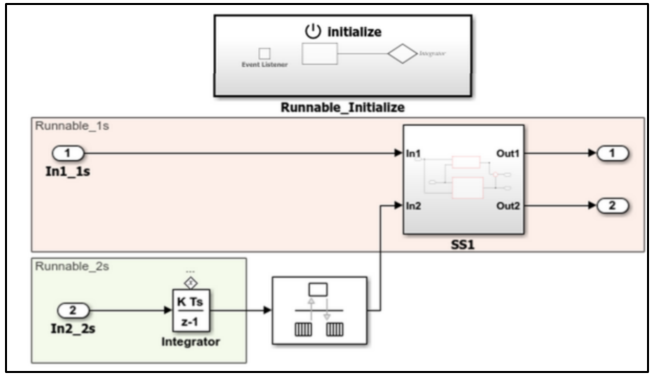
High Performance Hardware/Virtual Machine

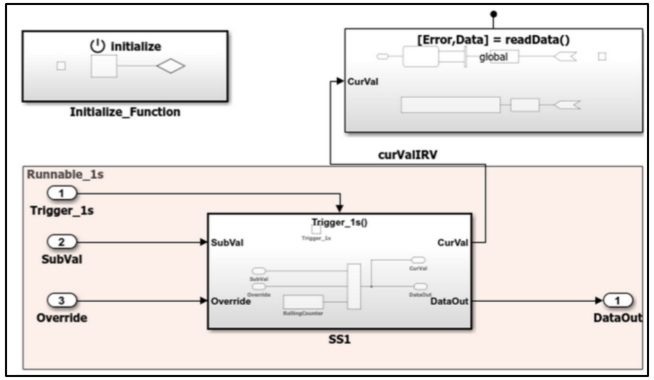
Motivation for Simulink to support Adaptive

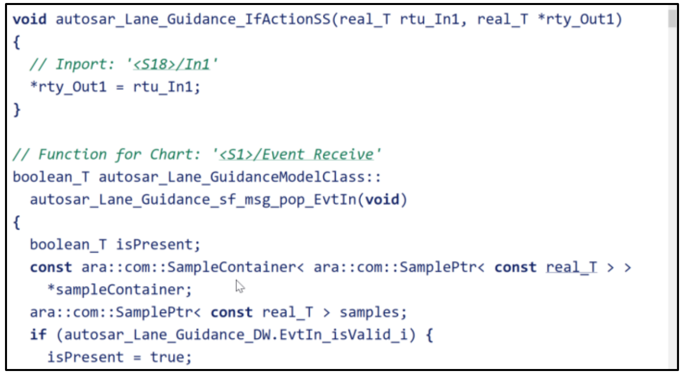
- Simulink is heavily used for AUTOSAR Classic
- Customers have requested Simulink support for Adaptive platform

- 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





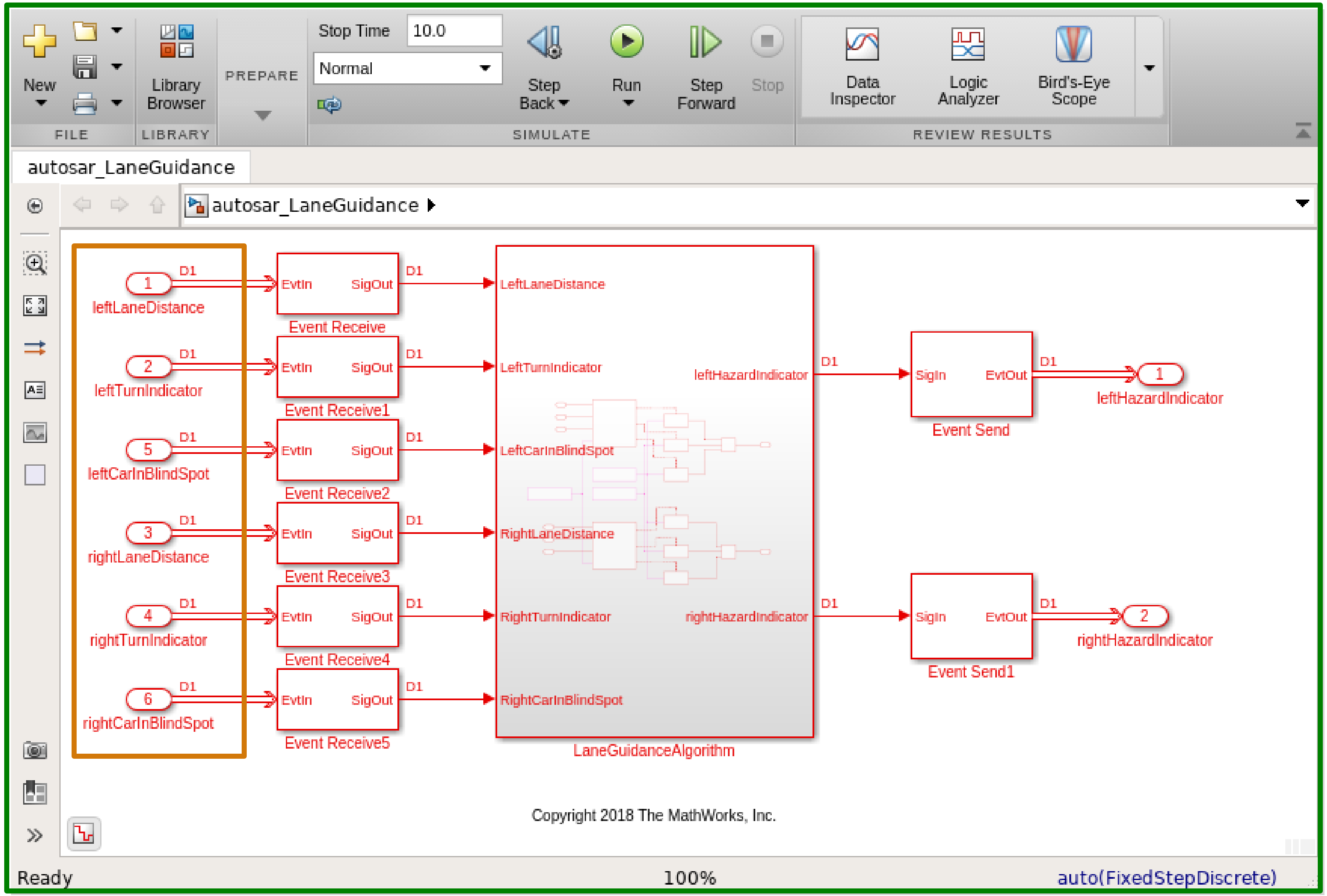


```
void autosar_Lane_Guidance_IfActionSS(real_T rtu_In1, real_T *rty_Out1)
{
  // Inport: '<S18>/In1'
  *rty_Out1 = rtu_In1;
}

// 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;
```

Mapping AUTOSAR AP Concepts to Simulink

Adaptive Application
RequiredPort

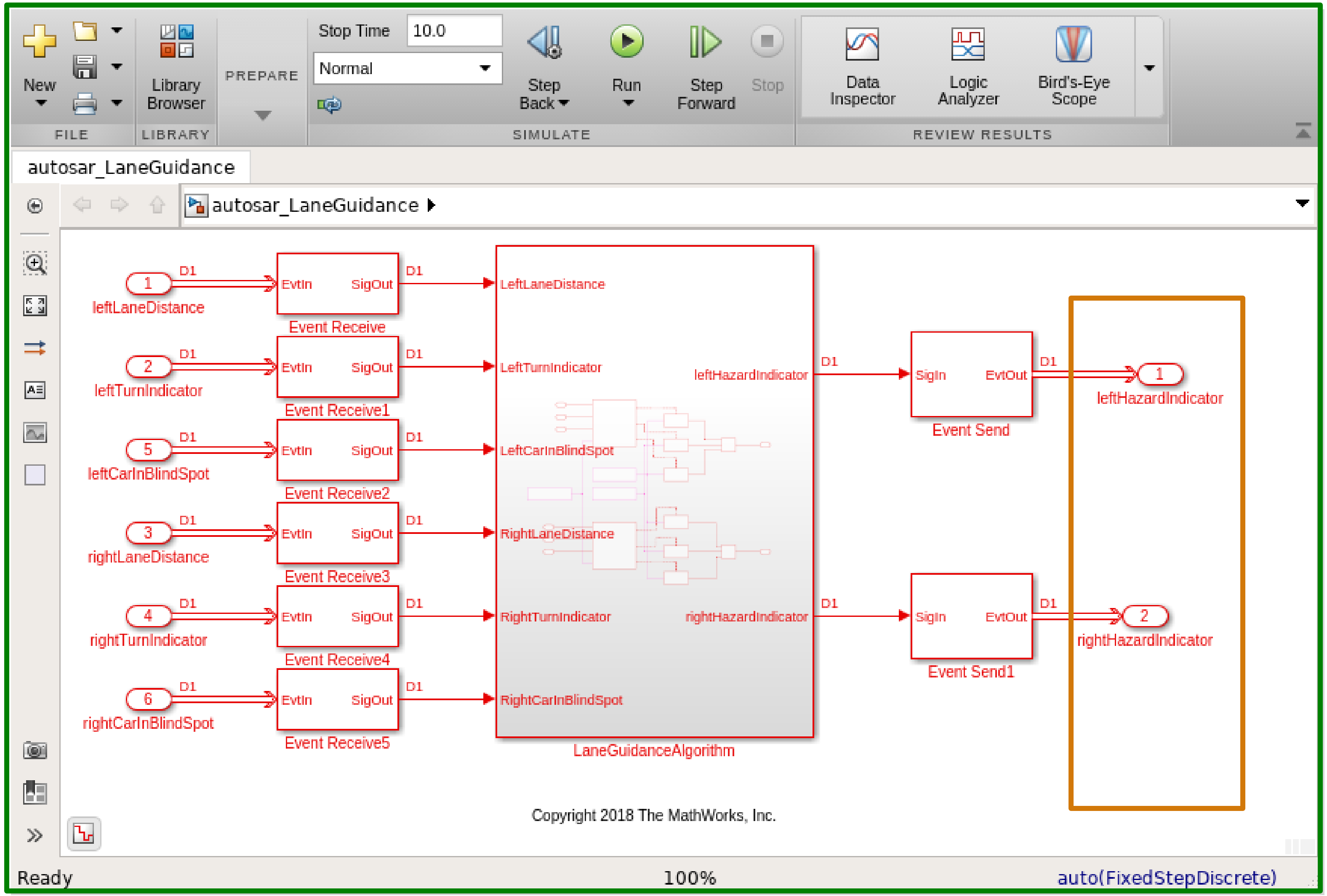
```
"Radar" : {
    // events
    "event" : {
        “leftLaneDistance”
        “leftTurnIndicator”
        “leftCarInBlindSpot”
        “rightLandDistance”
        “rightTurnIndicator”
        “rightCarInBlindSpot”
    },
    // methods
    "method" : {
        "Calibrate"
        "Adjust"
    },
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



Mapping AUTOSAR AP Concepts to Simulink

Adaptive Application
ProvidedPort

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



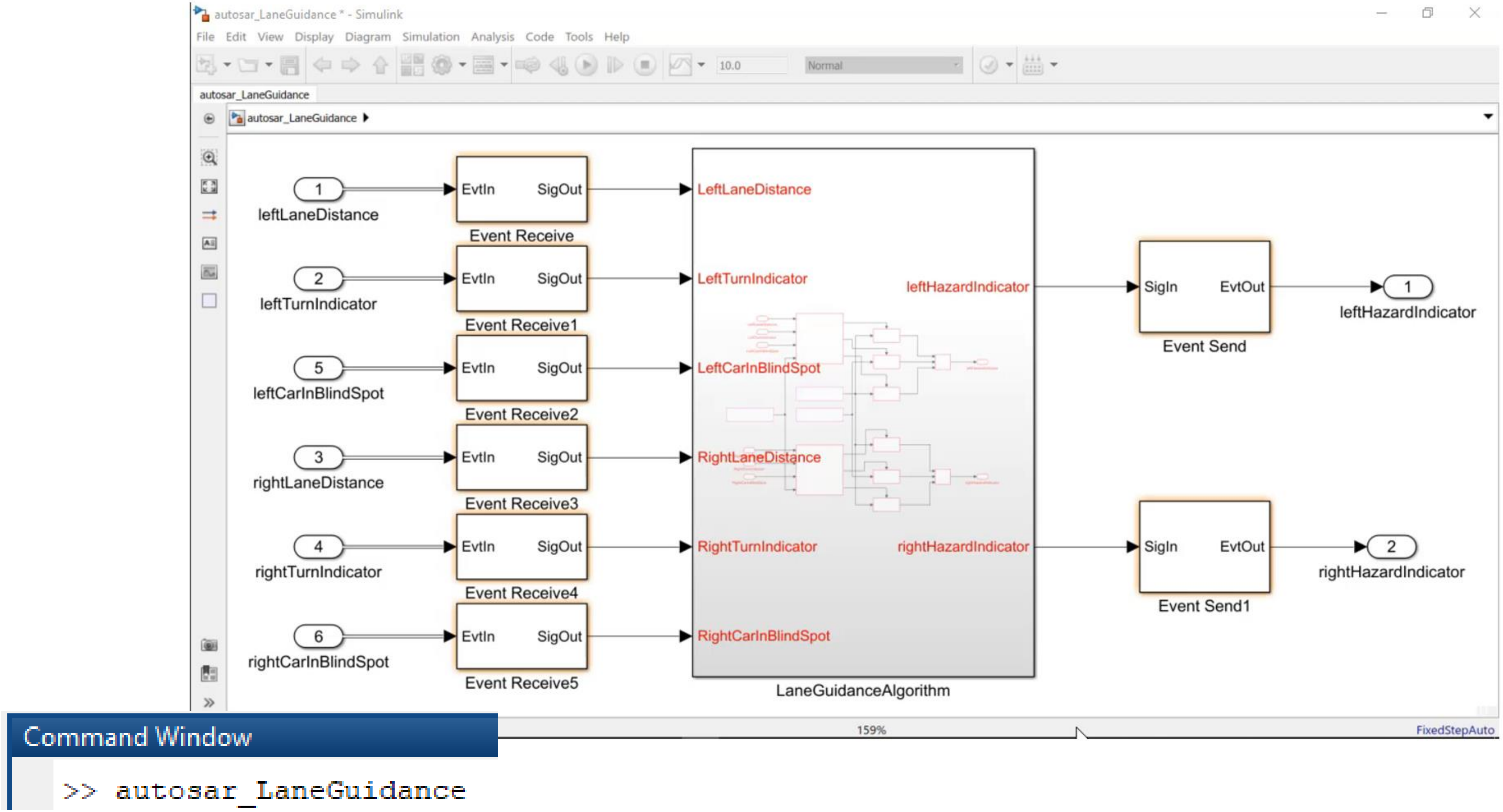
Stop Time 10.0

Normal

Copyright 2018 The MathWorks, Inc.

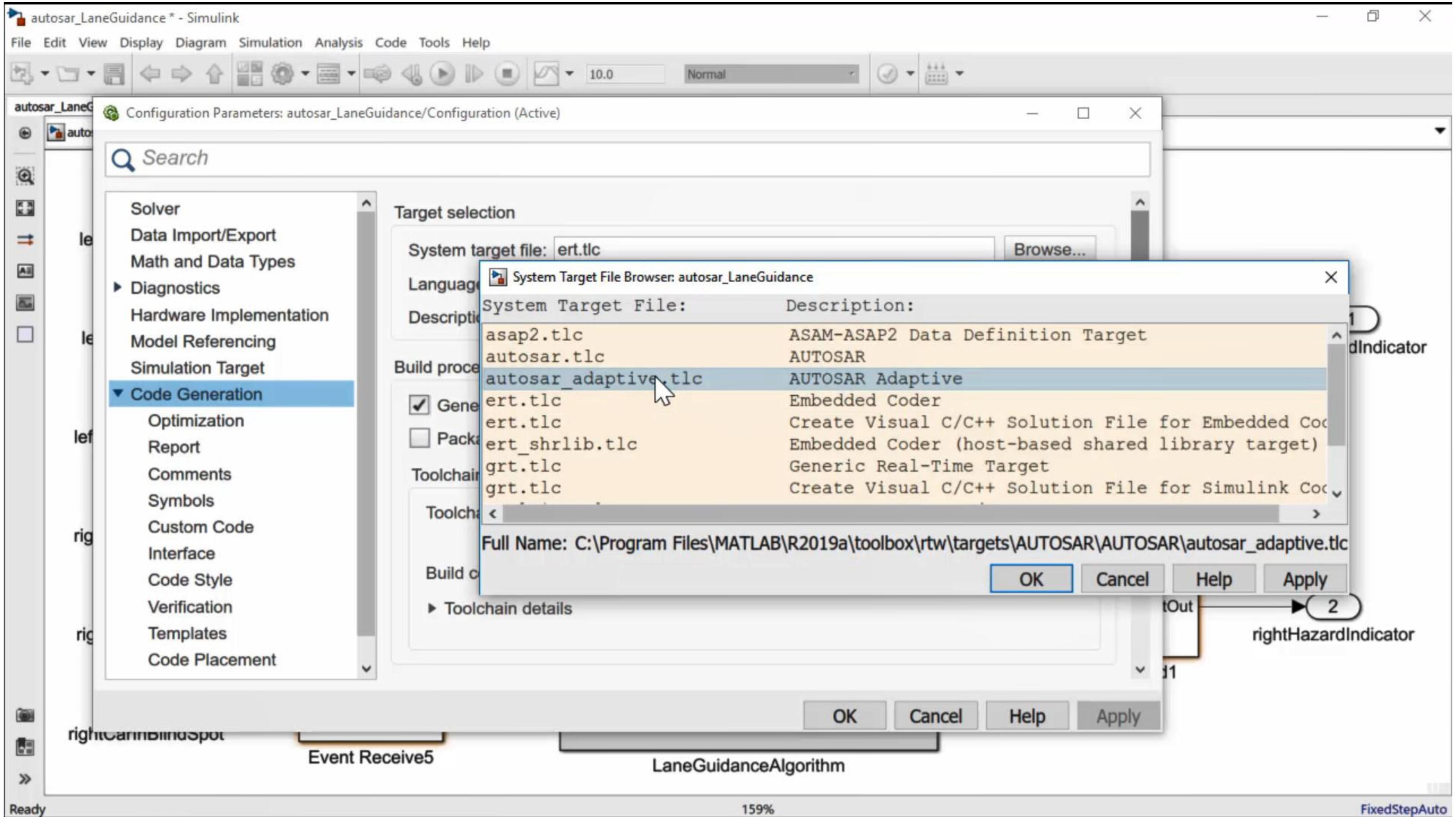
Ready 100% auto(FixedStepDiscrete)

Example of Configuring a model for Adaptive Platform



```
>> autosar_LaneGuidance
```

Change Target to AUTOSAR Adaptive

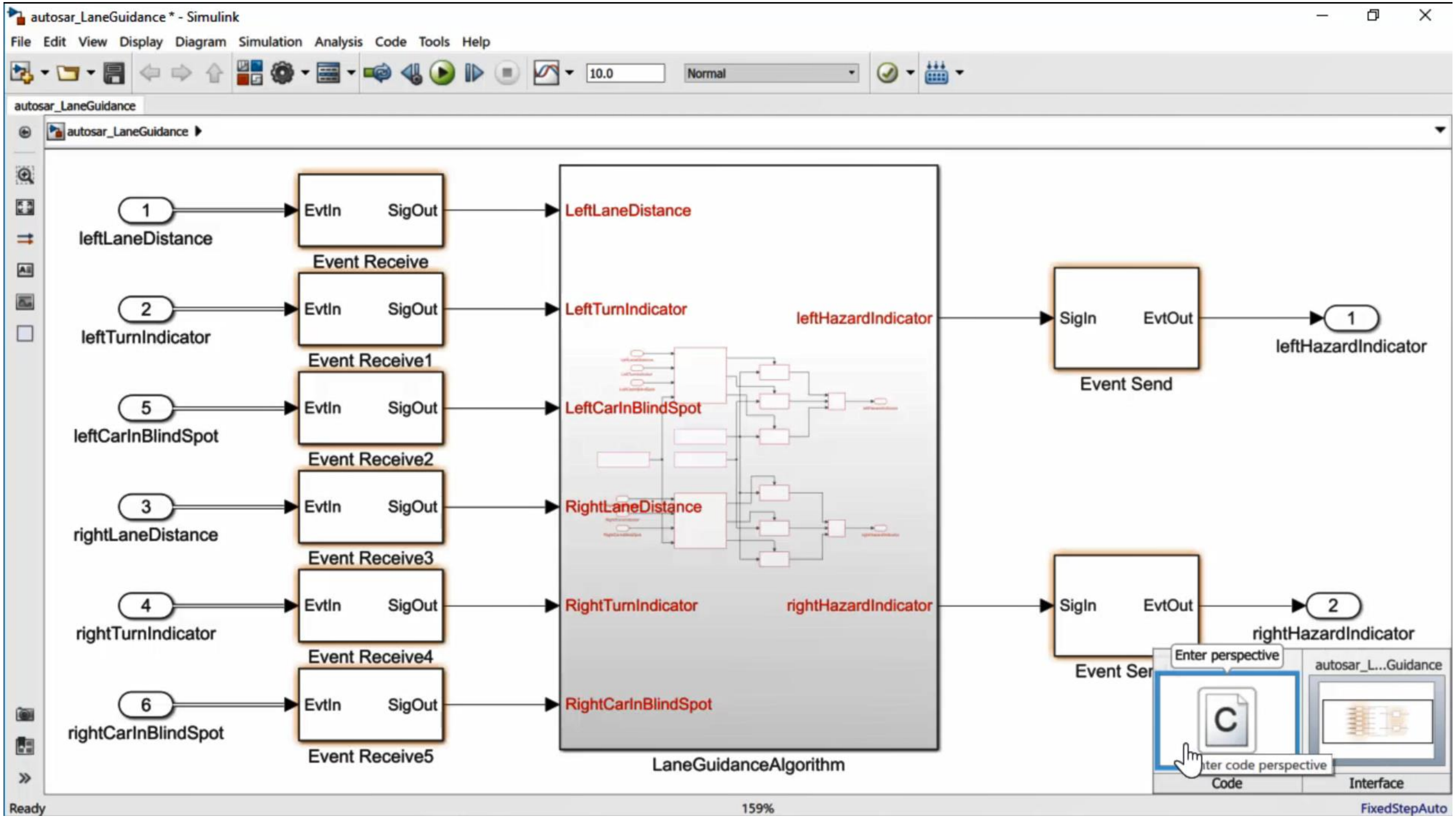


System target file: ert.tlc

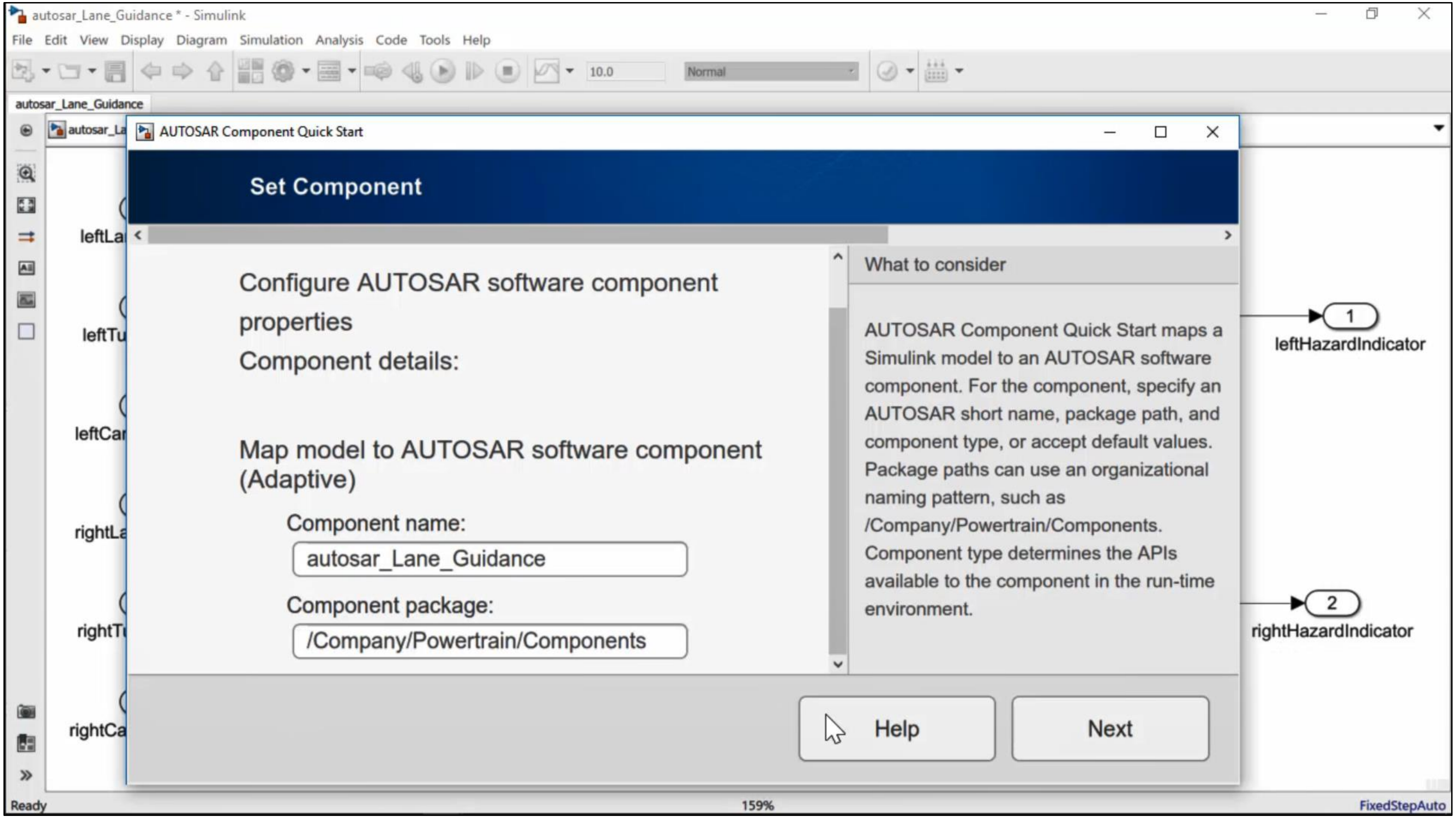
System Target File:	Description:
asap2.tlc	ASAM-ASAP2 Data Definition Target
autosar.tlc	AUTOSAR
autosar_adaptive.tlc	AUTOSAR Adaptive
ert.tlc	Embedded Coder
ert.tlc	Create Visual C/C++ Solution File for Embedded Coc
ert_shrlib.tlc	Embedded Coder (host-based shared library target)
grt.tlc	Generic Real-Time Target
grt.tlc	Create Visual C/C++ Solution File for Simulink Coc

Full Name: C:\Program Files\MATLAB\R2019a\toolbox\rtw\targets\AUTOSAR\AUTOSAR\autosar_adaptive.tlc

Enter Code Perspective to start the Configuration process



AUTOSAR Quick Start – Set Component



Set Component

Configure AUTOSAR software component properties

Component details:

Map model to AUTOSAR software component (Adaptive)

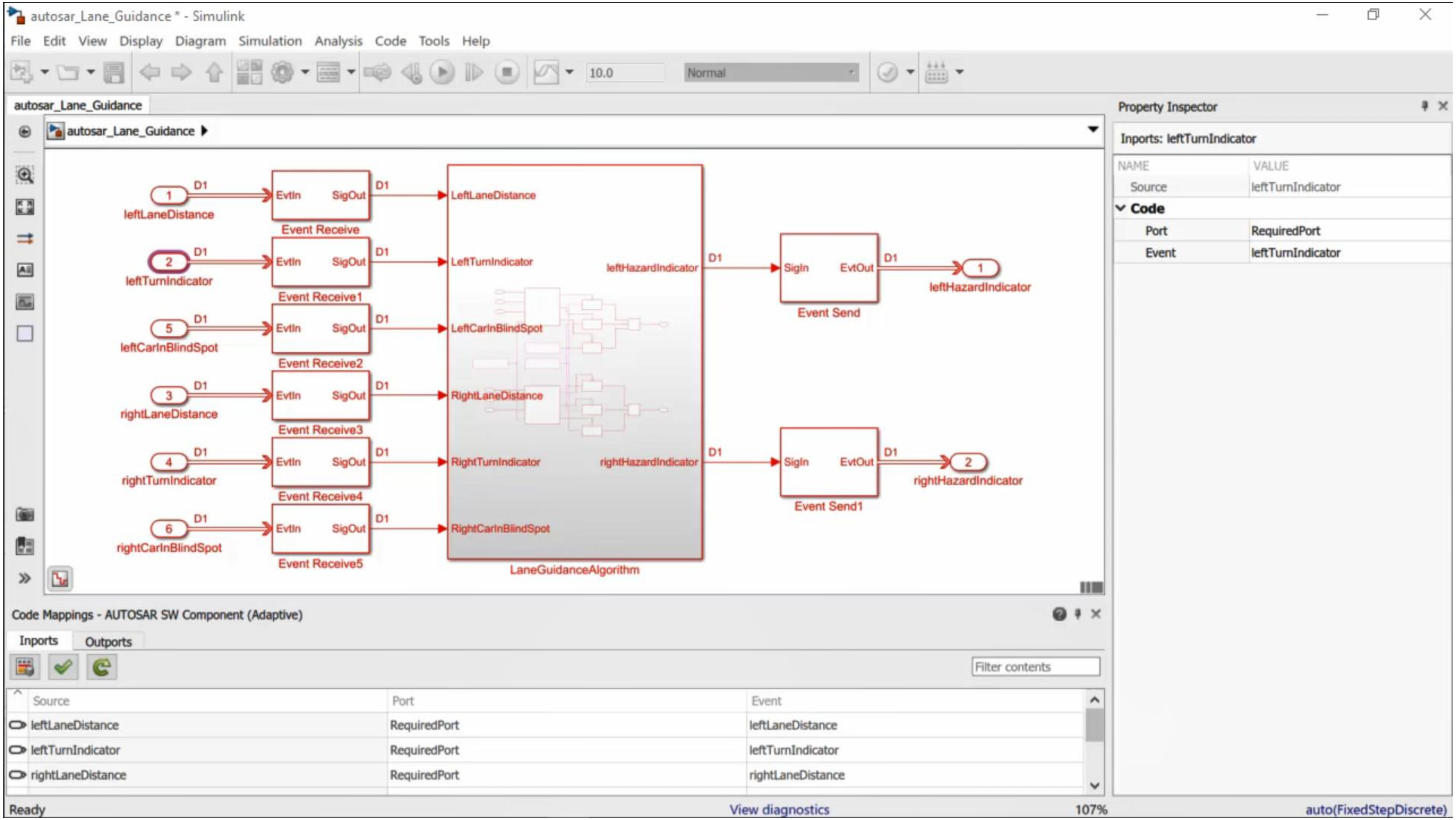
Component name: autosar_Lane_Guidance

Component package: /Company/Powertrain/Components

What to consider

AUTOSAR Component Quick Start maps a Simulink model to an AUTOSAR software component. For the component, specify an AUTOSAR short name, package path, and component type, or accept default values. Package paths can use an organizational naming pattern, such as /Company/Powertrain/Components. Component type determines the APIs available to the component in the run-time environment.

Quick Start Complete – Code Mappings setup for AS Port Events



Property Inspector

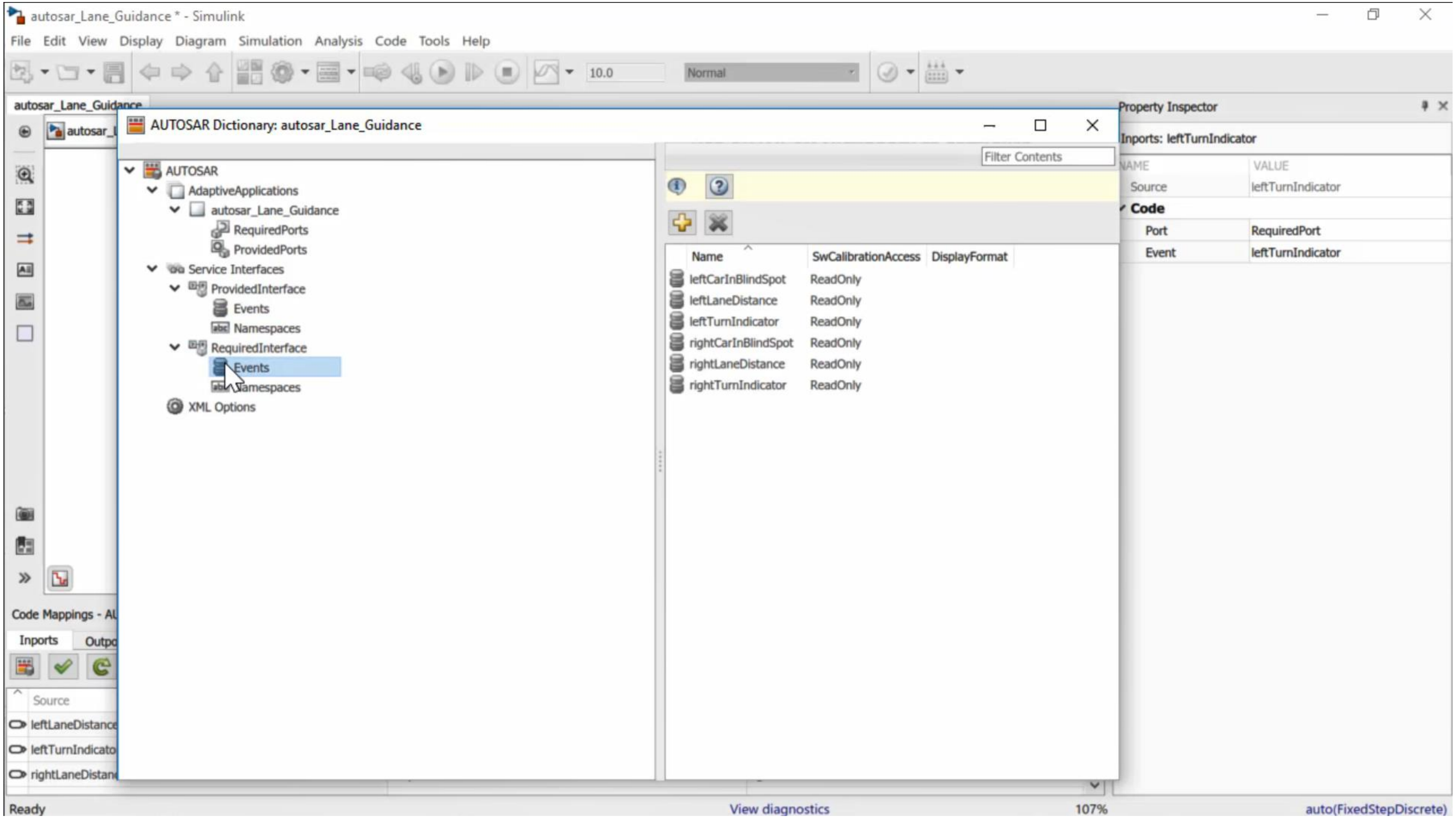
Inports: leftTurnIndicator

NAME	VALUE
Source	leftTurnIndicator
Code	
Port	RequiredPort
Event	leftTurnIndicator

Code Mappings - AUTOSAR SW Component (Adaptive)

Source	Port	Event
leftLaneDistance	RequiredPort	leftLaneDistance
leftTurnIndicator	RequiredPort	leftTurnIndicator
rightLaneDistance	RequiredPort	rightLaneDistance

Adaptive AUTOSAR Dictionary – Notice the Service Interfaces



AUTOSAR Dictionary: autosar_Lane_Guidance

- AUTOSAR
  - AdaptiveApplications
    - autosar_Lane_Guidance
      - RequiredPorts
      - ProvidedPorts
  - Service Interfaces
    - ProvidedInterface
      - Events
      - Namespaces
    - RequiredInterface
      - Events
      - Namespaces
  - XML Options

Filter Contents

Name	SwCalibrationAccess	DisplayFormat
leftCarInBlindSpot	ReadOnly	
leftLaneDistance	ReadOnly	
leftTurnIndicator	ReadOnly	
rightCarInBlindSpot	ReadOnly	
rightLaneDistance	ReadOnly	
rightTurnIndicator	ReadOnly	

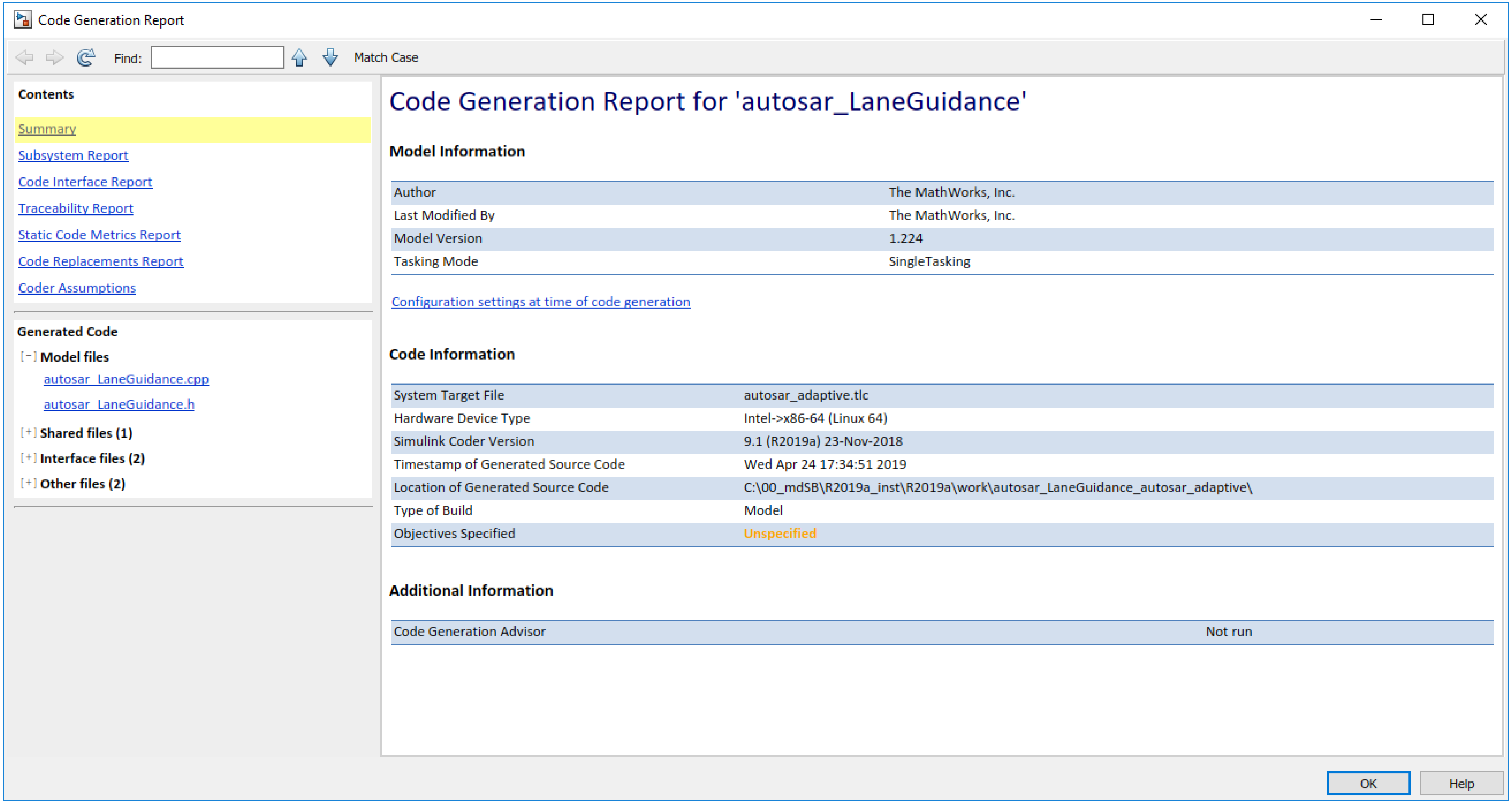
Property Inspector

Inports: leftTurnIndicator

NAME	VALUE
Source	leftTurnIndicator
Code	
Port	RequiredPort
Event	leftTurnIndicator

Source
leftLaneDistance
leftTurnIndicato
rightLaneDistan

Generate Code for the Adaptive AUTOSAR Model



Contents

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report
- Coder Assumptions

Generated Code

[-] **Model files**

- autosar_LaneGuidance.cpp
- autosar_LaneGuidance.h

[+] **Shared files (1)**

[+] **Interface files (2)**

[+] **Other files (2)**

Code Generation Report for 'autosar_LaneGuidance'

Model Information

Author	The MathWorks, Inc.
Last Modified By	The MathWorks, Inc.
Model Version	1.224
Tasking Mode	SingleTasking

Configuration settings at time of code generation

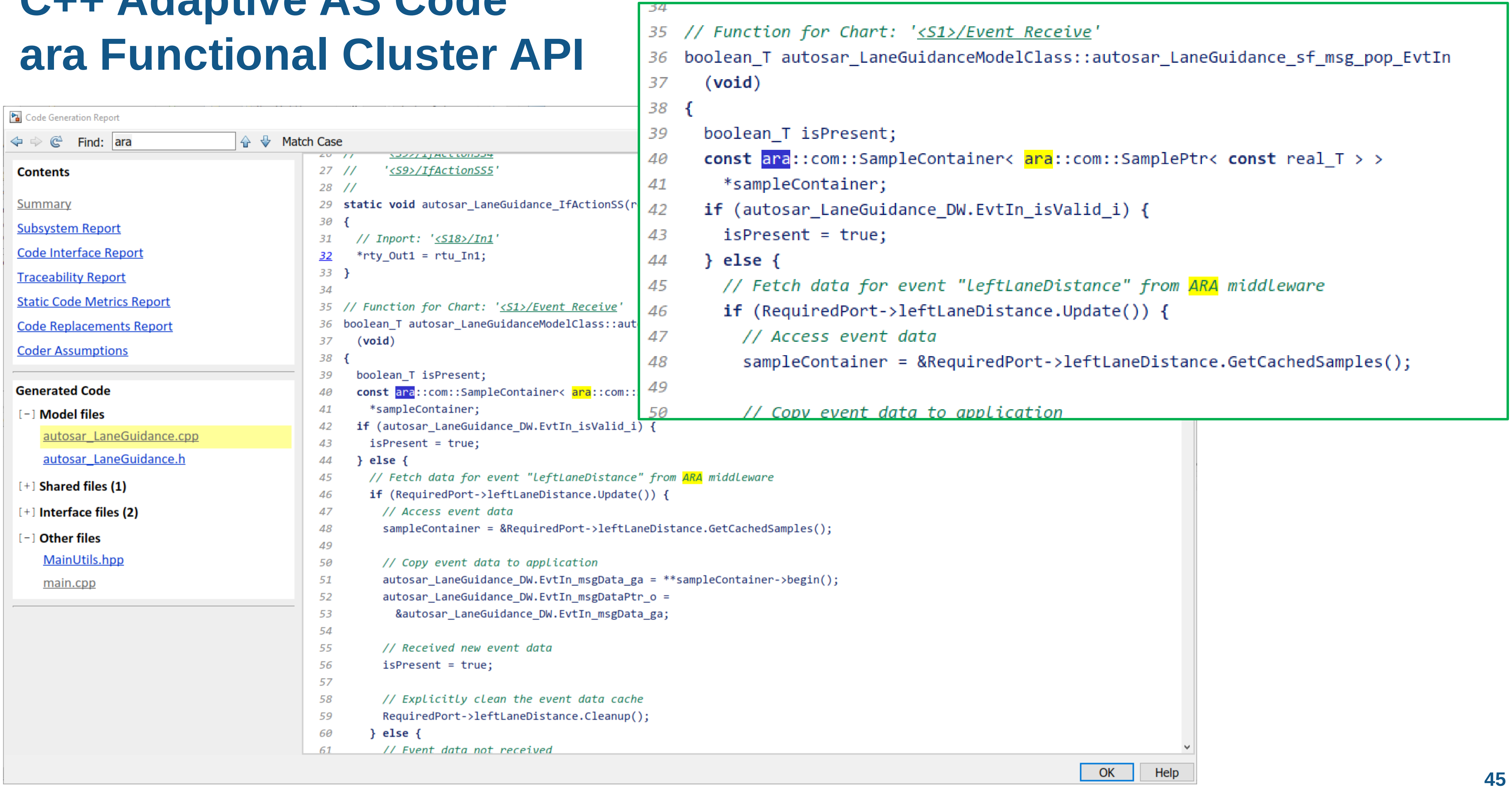
Code Information

System Target File	autosar_adaptive.tlc
Hardware Device Type	Intel->x86-64 (Linux 64)
Simulink Coder Version	9.1 (R2019a) 23-Nov-2018
Timestamp of Generated Source Code	Wed Apr 24 17:34:51 2019
Location of Generated Source Code	C:\00_mdSB\R2019a_inst\R2019a\work\autosar_LaneGuidance_autosar_adaptive\
Type of Build	Model
Objectives Specified	Unspecified

Additional Information

Code Generation Advisor	Not run

C++ Adaptive AS Code
ara Functional Cluster API



Contents

Summary
Subsystem Report
Code Interface Report
Traceability Report
Static Code Metrics Report
Code Replacements Report
Coder Assumptions

Generated Code

[-] **Model files**
- autosar_LaneGuidance.cpp
- autosar_LaneGuidance.h

[+] **Shared files (1)**

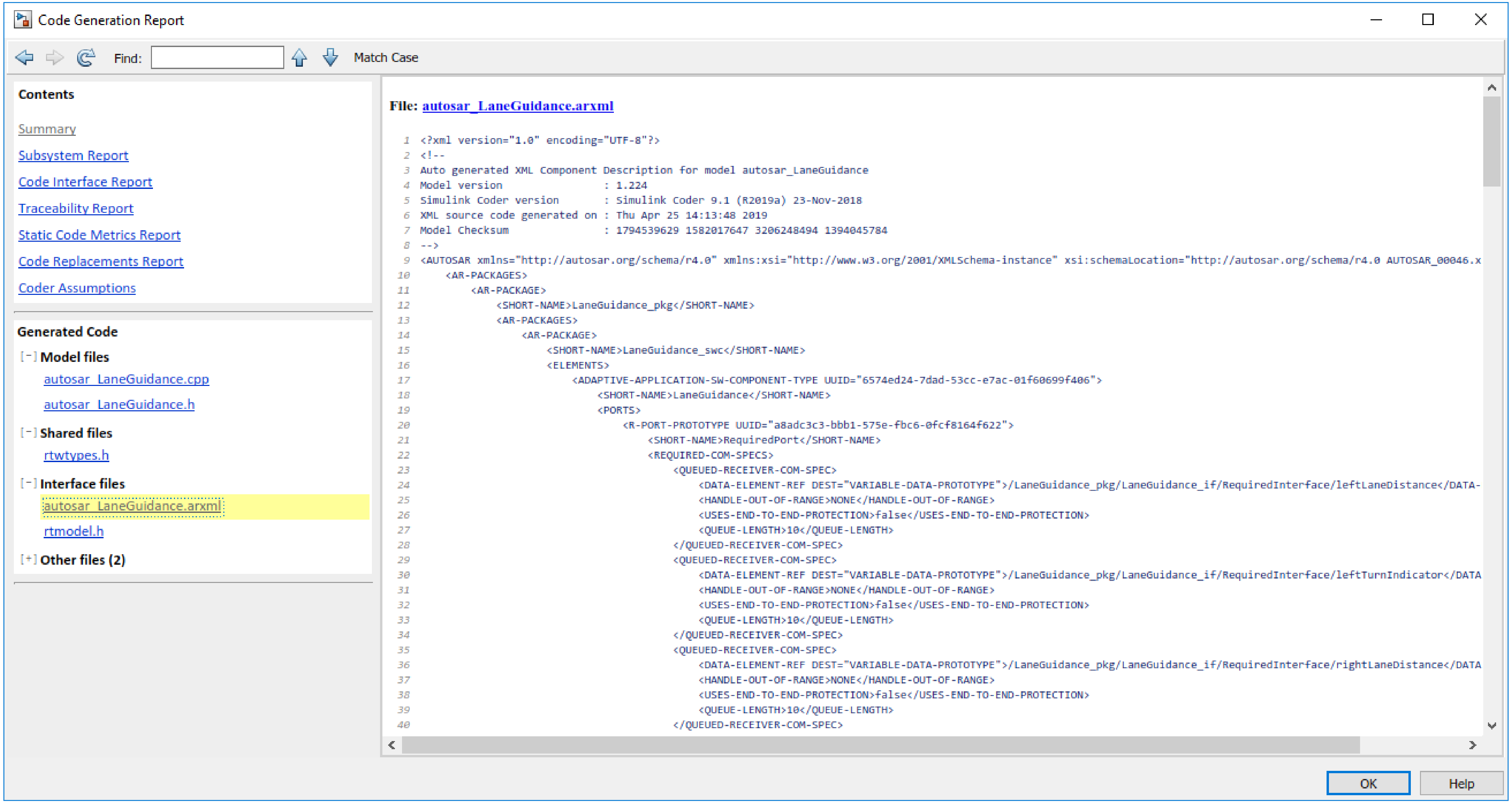
[+] **Interface files (2)**

[-] **Other files**
- MainUtils.hpp
- main.cpp

```
//     '<S9>/IfActionSS4'
//     '<S9>/IfActionSS5'
//
static void autosar_LaneGuidance_IfActionSS(r
{
  // Inport: '<S18>/In1'
  *rty_Out1 = rtu_In1;
}

// Function for Chart: '<S1>/Event Receive'
boolean_T autosar_LaneGuidanceModelClass::aut
  (void)
{
  boolean_T isPresent;
  const ara::com::SampleContainer< ara::com::
    *sampleContainer;
  if (autosar_LaneGuidance_DW.EvtIn_isValid_i) {
    isPresent = true;
  } else {
    // Fetch data for event "leftLaneDistance" from ARA middleware
    if (RequiredPort->leftLaneDistance.Update()) {
      // Access event data
      sampleContainer = &RequiredPort->leftLaneDistance.GetCachedSamples();

      // Copy event data to application
      autosar_LaneGuidance_DW.EvtIn_msgData_ga = **sampleContainer->begin();
      autosar_LaneGuidance_DW.EvtIn_msgDataPtr_o =
        &autosar_LaneGuidance_DW.EvtIn_msgData_ga;

      // Received new event data
      isPresent = true;

      // Explicitly clean the event data cache
      RequiredPort->leftLaneDistance.Cleanup();
    } else {
      // Event data not received
```

```
// Function for Chart: '<S1>/Event Receive'
boolean_T autosar_LaneGuidanceModelClass::autosar_LaneGuidance_sf_msg_pop_EvtIn
  (void)
{
  boolean_T isPresent;
  const ara::com::SampleContainer< ara::com::SamplePtr< const real_T > >
    *sampleContainer;
  if (autosar_LaneGuidance_DW.EvtIn_isValid_i) {
    isPresent = true;
  } else {
    // Fetch data for event "leftLaneDistance" from ARA middleware
    if (RequiredPort->leftLaneDistance.Update()) {
      // Access event data
      sampleContainer = &RequiredPort->leftLaneDistance.GetCachedSamples();

      // Copy event data to application
```

Software Component Description Files Generated



Contents

- Summary
- Subsystem Report
- Code Interface Report
- Traceability Report
- Static Code Metrics Report
- Code Replacements Report
- Coder Assumptions

Generated Code

[-] **Model files**

- autosar_LaneGuidance.cpp
- autosar_LaneGuidance.h

[-] **Shared files**

- rtwtypes.h

[-] **Interface files**

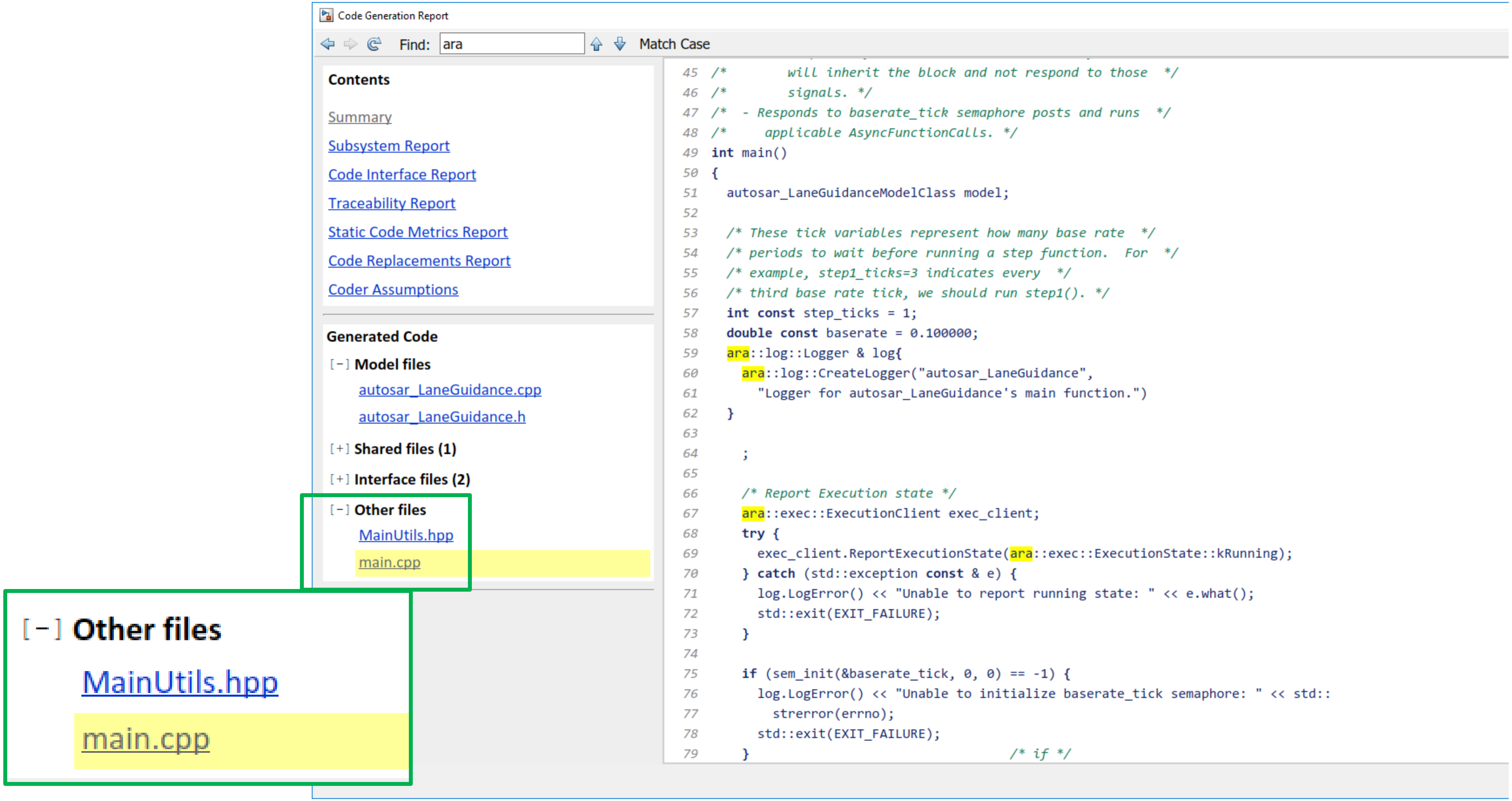
- autosar_LaneGuidance.arxml
- rtmodel.h

[+] **Other files (2)**

File: autosar_LaneGuidance.arxml

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Auto generated XML Component Description for model autosar_LaneGuidance
Model version                : 1.224
Simulink Coder version       : Simulink Coder 9.1 (R2019a) 23-Nov-2018
XML source code generated on : Thu Apr 25 14:13:48 2019
Model Checksum               : 1794539629 1582017647 3206248494 1394045784
-->
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://autosar.org/schema/r4.0 AUTOSAR_00046.x
    <AR-PACKAGES>
        <AR-PACKAGE>
            <SHORT-NAME>LaneGuidance_pkg</SHORT-NAME>
            <AR-PACKAGES>
                <AR-PACKAGE>
                    <SHORT-NAME>LaneGuidance_swc</SHORT-NAME>
                    <ELEMENTS>
                        <ADAPTIVE-APPLICATION-SW-COMPONENT-TYPE UUID="6574ed24-7dad-53cc-e7ac-01f60699f406">
                            <SHORT-NAME>LaneGuidance</SHORT-NAME>
                            <PORTS>
                                <R-PORT-PROTOTYPE UUID="a8adc3c3-bbb1-575e-fbc6-0fcf8164f622">
                                    <SHORT-NAME>RequiredPort</SHORT-NAME>
                                    <REQUIRED-COM-SPECS>
                                        <QUEUED-RECEIVER-COM-SPEC>
                                            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/LaneGuidance_pkg/LaneGuidance_if/RequiredInterface/leftLaneDistance</DATA-
                                            <HANDLE-OUT-OF-RANGE>NONE</HANDLE-OUT-OF-RANGE>
                                            <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                                            <QUEUE-LENGTH>10</QUEUE-LENGTH>
                                        </QUEUED-RECEIVER-COM-SPEC>
                                        <QUEUED-RECEIVER-COM-SPEC>
                                            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/LaneGuidance_pkg/LaneGuidance_if/RequiredInterface/leftTurnIndicator</DATA
                                            <HANDLE-OUT-OF-RANGE>NONE</HANDLE-OUT-OF-RANGE>
                                            <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                                            <QUEUE-LENGTH>10</QUEUE-LENGTH>
                                        </QUEUED-RECEIVER-COM-SPEC>
                                        <QUEUED-RECEIVER-COM-SPEC>
                                            <DATA-ELEMENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/LaneGuidance_pkg/LaneGuidance_if/RequiredInterface/rightLaneDistance</DATA
                                            <HANDLE-OUT-OF-RANGE>NONE</HANDLE-OUT-OF-RANGE>
                                            <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                                            <QUEUE-LENGTH>10</QUEUE-LENGTH>
                                        </QUEUED-RECEIVER-COM-SPEC>
```

Adaptive Standalone Application Code needs a main.cpp



Contents

Summary

Subsystem Report

Code Interface Report

Traceability Report

Static Code Metrics Report

Code Replacements Report

Coder Assumptions

Generated Code

[-] **Model files**

autosar_LaneGuidance.cpp

autosar_LaneGuidance.h

[+] **Shared files (1)**

[+] **Interface files (2)**

[-] **Other files**

MainUtils.hpp

main.cpp

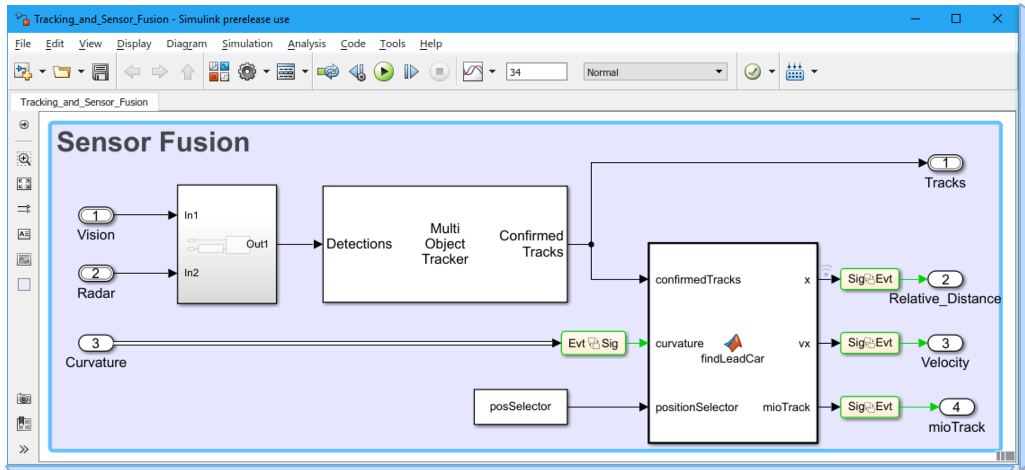
```
/*        will inherit the block and not respond to those  */
/*        signals. */
/*  - Responds to baserate_tick semaphore posts and runs  */
/*     applicable AsyncFunctionCalls. */
int main()
{
  autosar_LaneGuidanceModelClass model;

  /* These tick variables represent how many base rate  */
  /* periods to wait before running a step function.  For  */
  /* example, step1_ticks=3 indicates every  */
  /* third base rate tick, we should run step1(). */
  int const step_ticks = 1;
  double const baserate = 0.100000;
  ara::log::Logger & log{
    ara::log::CreateLogger("autosar_LaneGuidance",
      "Logger for autosar_LaneGuidance's main function.")
  }

    ;

    /* Report Execution state */
    ara::exec::ExecutionClient exec_client;
    try {
      exec_client.ReportExecutionState(ara::exec::ExecutionState::kRunning);
    } catch (std::exception const & e) {
      log.LogError() << "Unable to report running state: " << e.what();
      std::exit(EXIT_FAILURE);
    }

    if (sem_init(&baserate_tick, 0, 0) == -1) {
      log.LogError() << "Unable to initialize baserate_tick semaphore: " << std::
        strerror(errno);
      std::exit(EXIT_FAILURE);
    }                                  /* if */
```

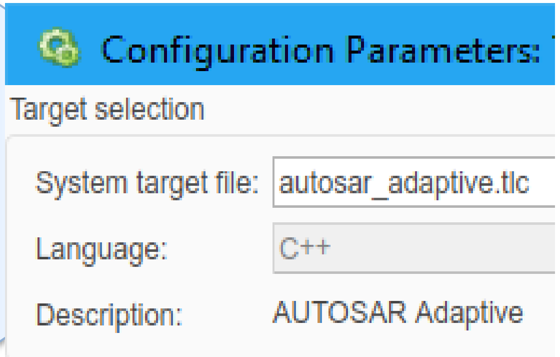
[-] **Other files**

MainUtils.hpp

main.cpp

Generate Production AUTOSAR Adaptive C++ Code



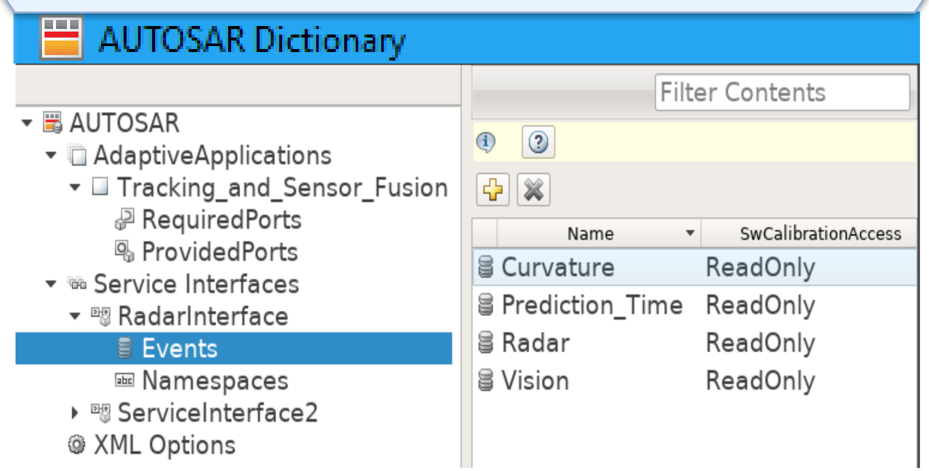

Configuration Parameters:

Target selection

System target file: autosar_adaptive.tlc

Language: C++

Description: AUTOSAR Adaptive


AUTOSAR Dictionary

- AUTOSAR
  - AdaptiveApplications
    - Tracking_and_Sensor_Fusion
      - RequiredPorts
      - ProvidedPorts
  - Service Interfaces
    - RadarInterface
      - Events
      - Namespaces
    - ServiceInterface2
  - XML Options

Filter Contents

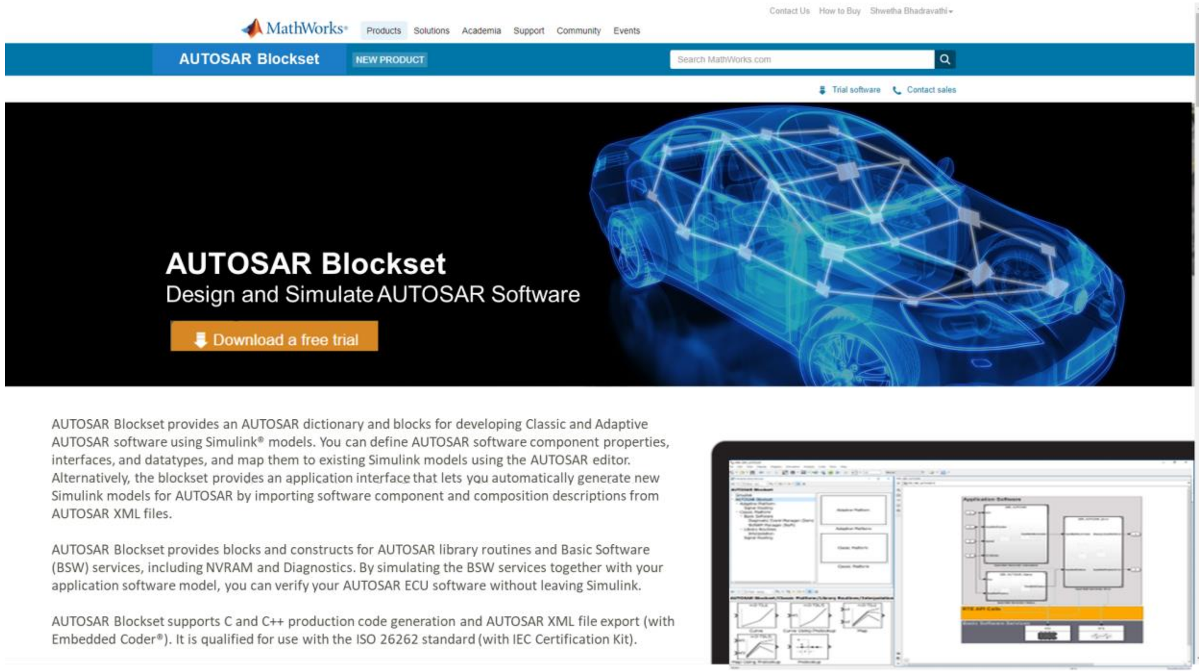
Name	SwCalibrationAccess
Curvature	ReadOnly
Prediction_Time	ReadOnly
Radar	ReadOnly
Vision	ReadOnly

```
main.cpp    <model>.cpp
*.hpp       *.cpp
*.arxml
```

AUTOSAR support

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

To learn more, please visit AUTOSAR webpage



AUTOSAR Blockset

Design and Simulate AUTOSAR Software

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