MATLAB EXPO

5月28日, 2024 | 北京

# 软件定义汽车: 基于Simulink开发面向服务的应用

*Wei WANG, MathWorks*

MathWorks®

# Agenda

- SW-defined vehicles and new architectures (SOA)
- MathWorks solutions for SOA
- Conclusions and key takeaways

# Agenda

- SW-defined vehicles and new architectures (SOA)
- MathWorks solutions for SOA
- Conclusions and key takeaways

# Motivations for SW defined vehicles

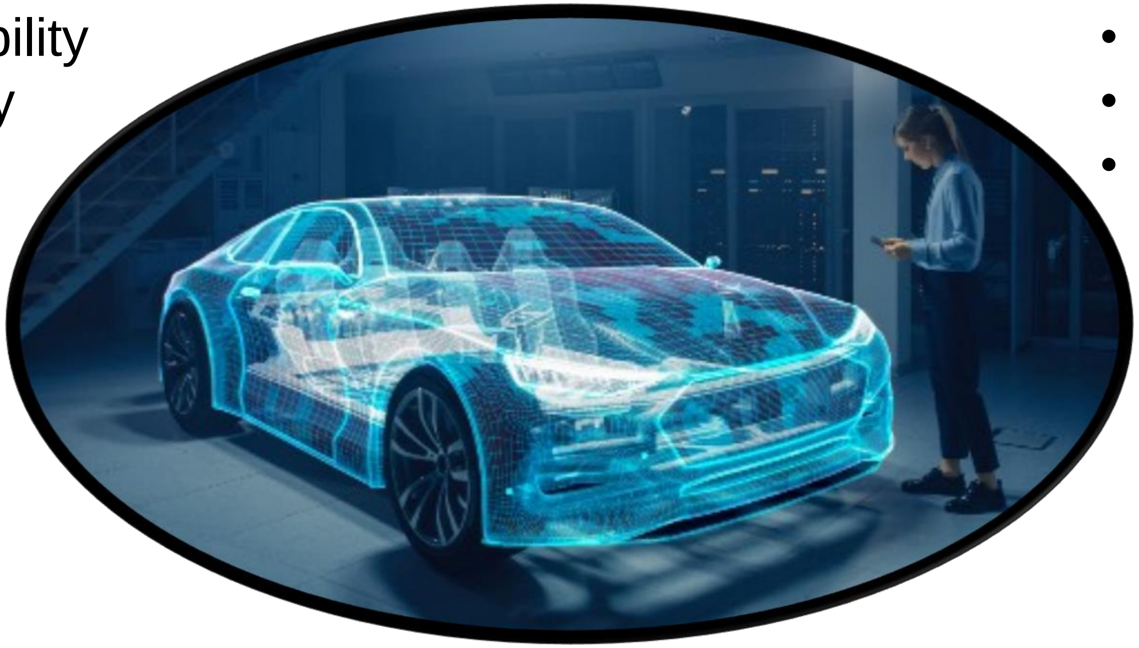
demand

## Customer expectations

- Clean and Safe mobility
- Digital Life continuity

## Technology & Innovation

- Electrification
- Autonomy
- Connectivity



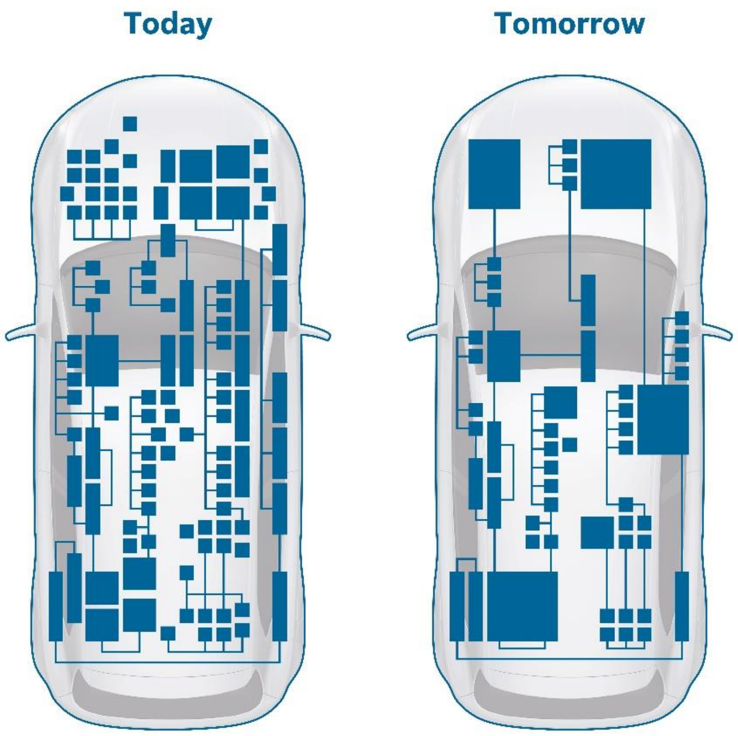
monetize

investment

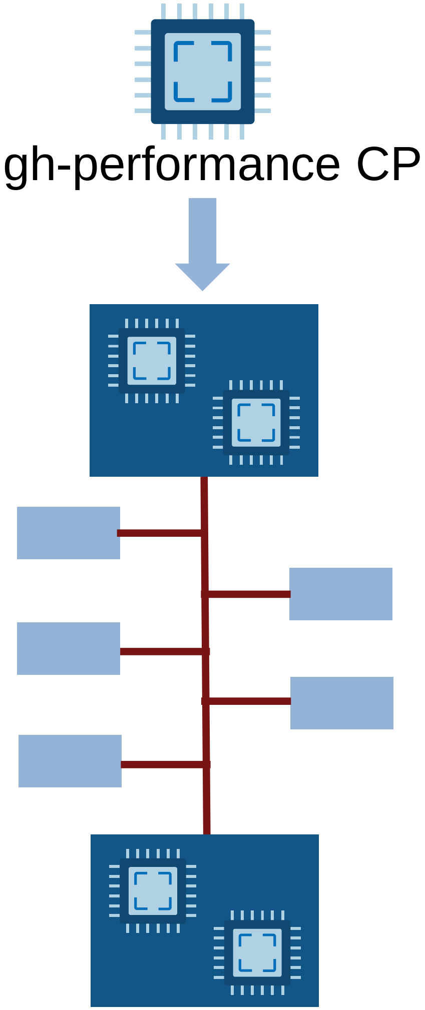
## Business opportunity

- App stores, SW features on demand
- SW services subscription plans

# Centralization of computing and SOA

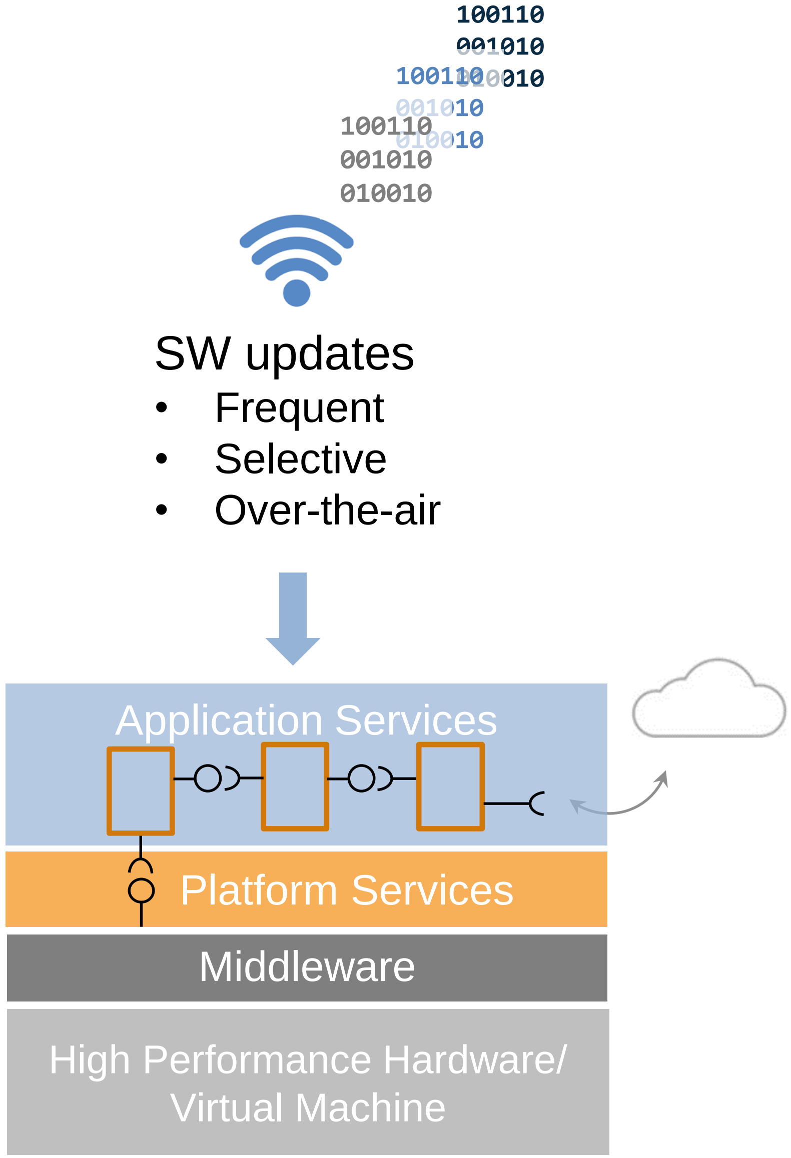


Consolidation and centralization of computing



High-performance CPU/GPU

New E/E zonal architectures

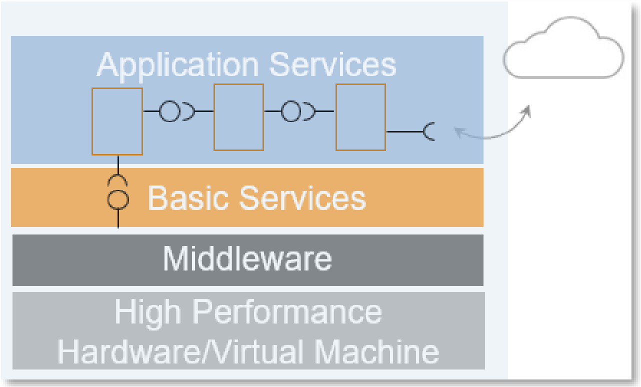


SW updates

- Frequent
- Selective
- Over-the-air

Higher HW abstraction: Service-oriented architectures

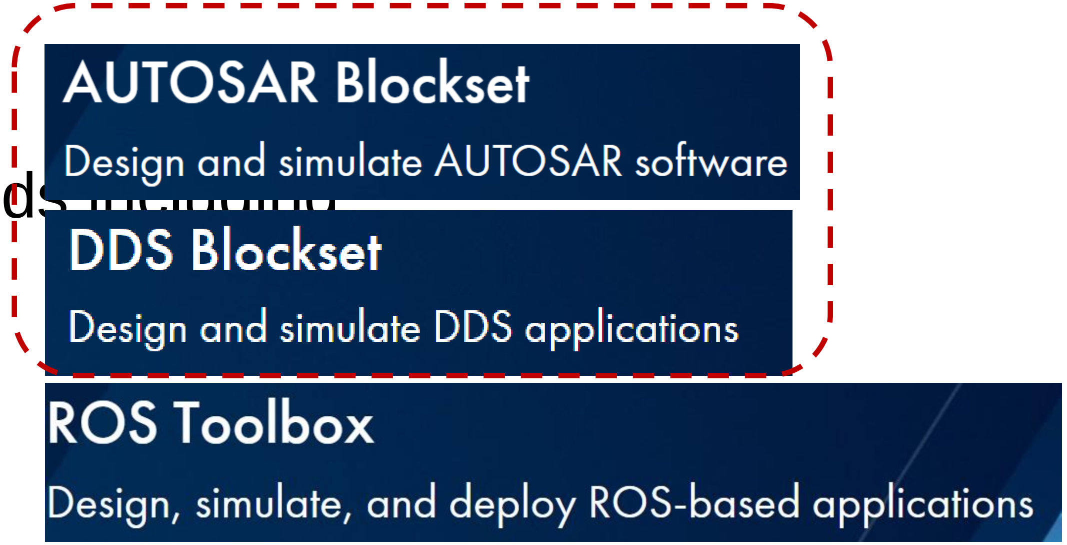
# SOA – What’s it all about?



- With SOA, applications are standalone processes that provide and/or require services distributed across the vehicle computing platform and the cloud

- SOA provides flexibility to add, remove, or update applications without impacting the entire, typically large, software system

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



**AUTOSAR Blockset**

Design and simulate AUTOSAR software

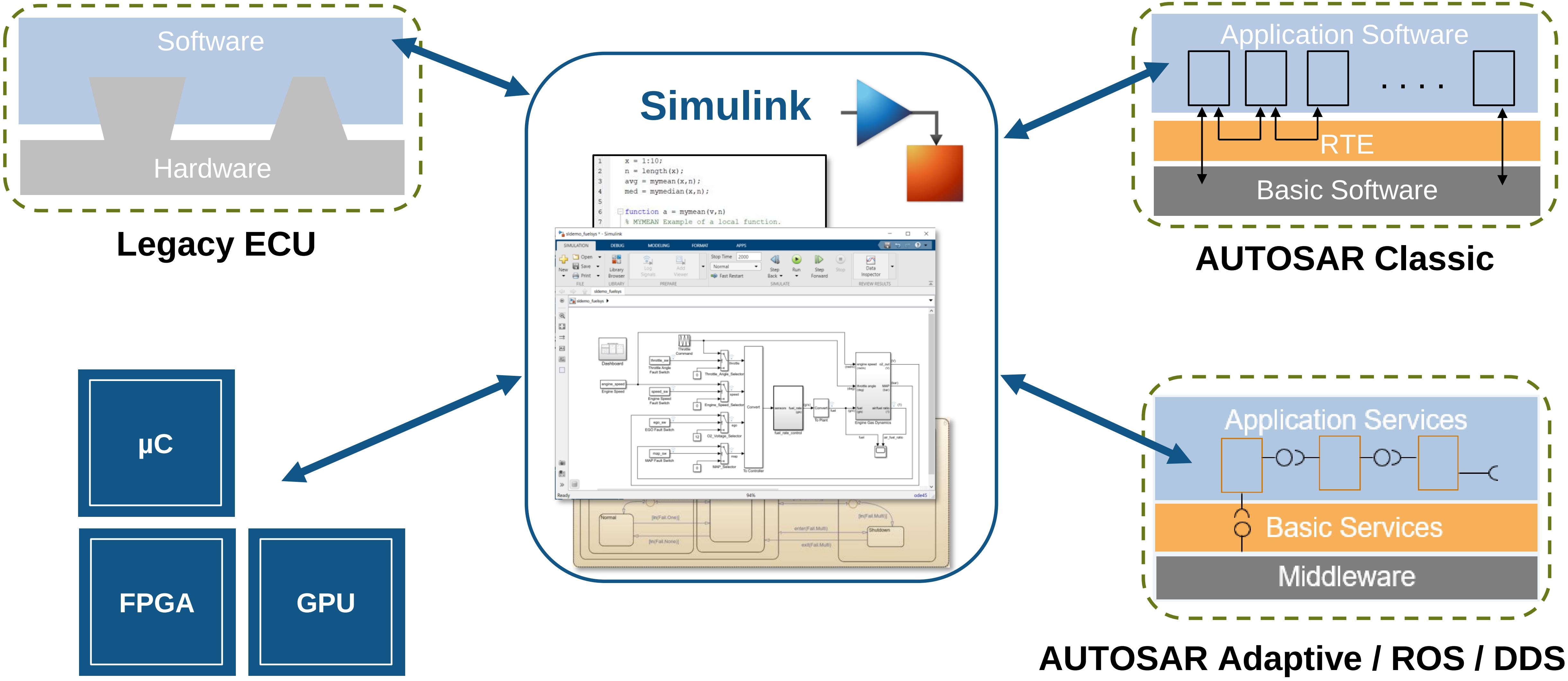
**DDS Blockset**

Design and simulate DDS applications

**ROS Toolbox**

Design, simulate, and deploy ROS-based applications

# Simulink: deploy software to different targets and standards

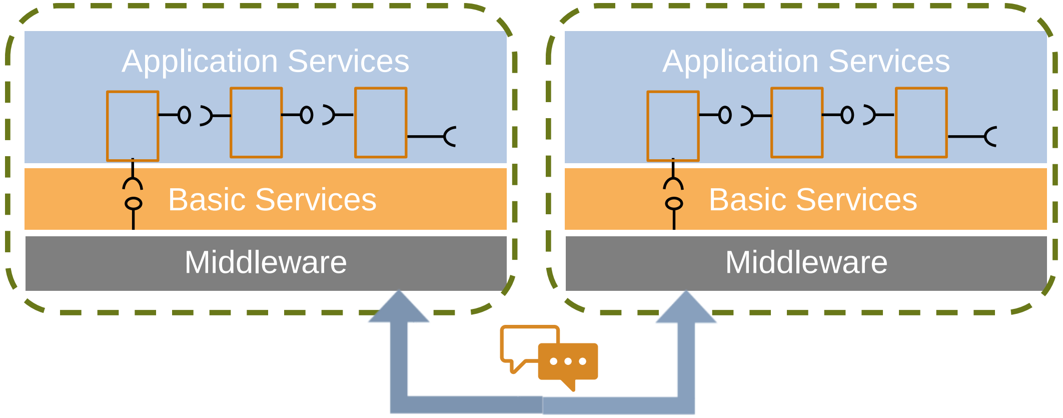


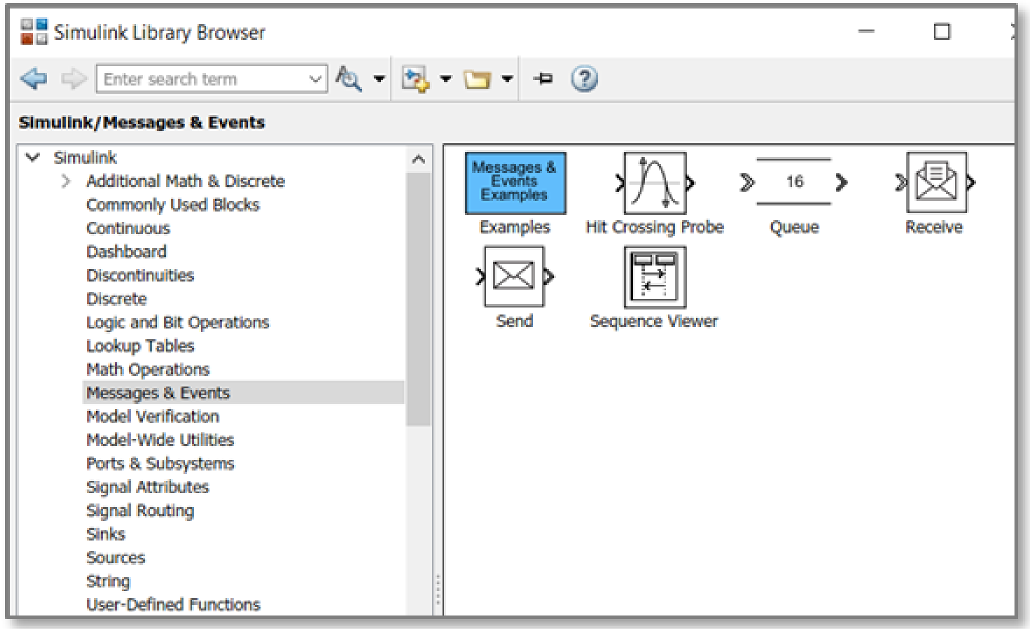
```
x = 1:10;
n = length(x);
avg = mymean(x,n);
med = mymedian(x,n);

function a = mymean(v,n)
% MYMEAN Example of a local function.
```

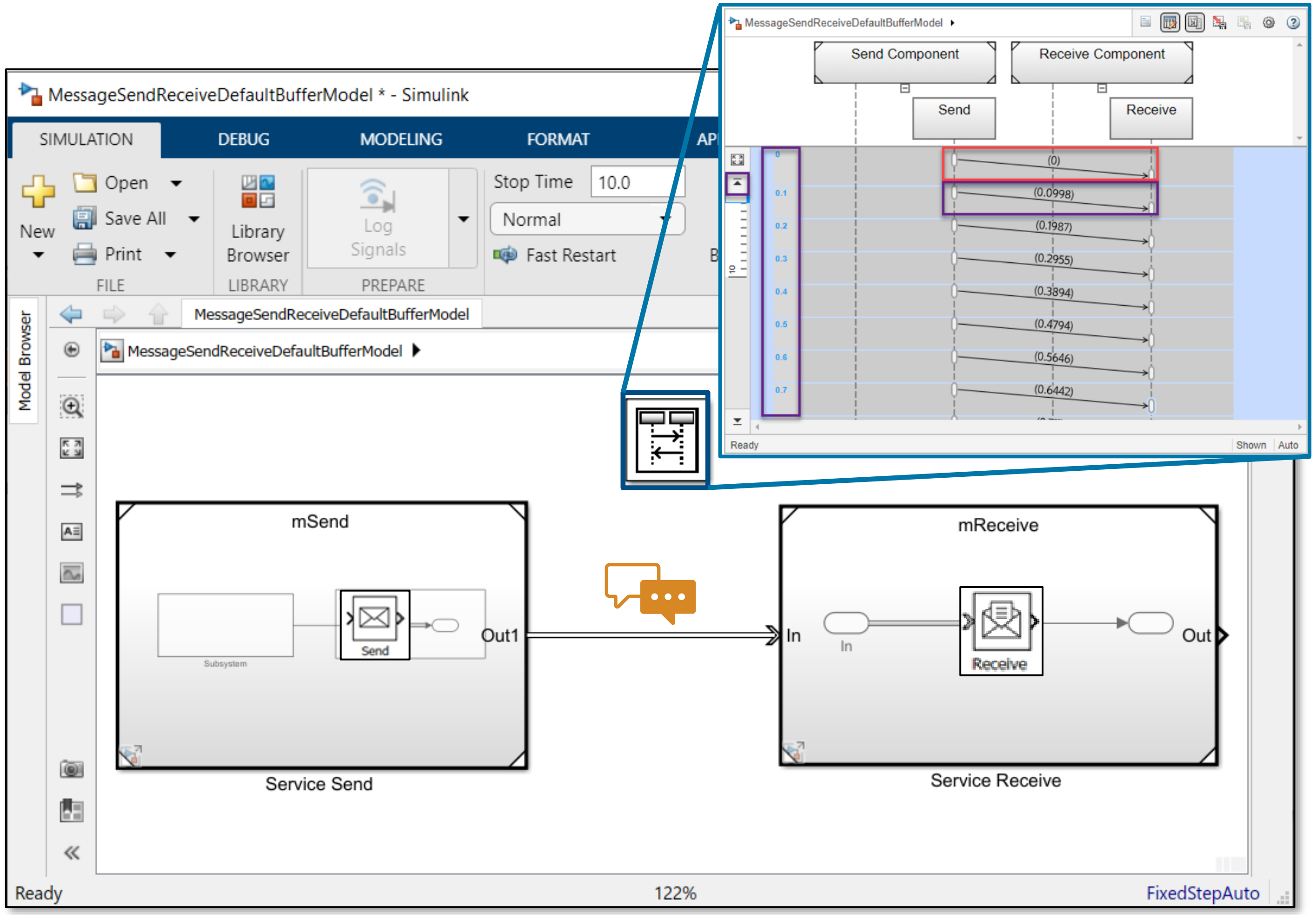
# Agenda

- SW-defined vehicles and new architectures (SOA)
- MathWorks solutions for SOA
- Conclusions and key takeaways

# Simulink Messages for SOA

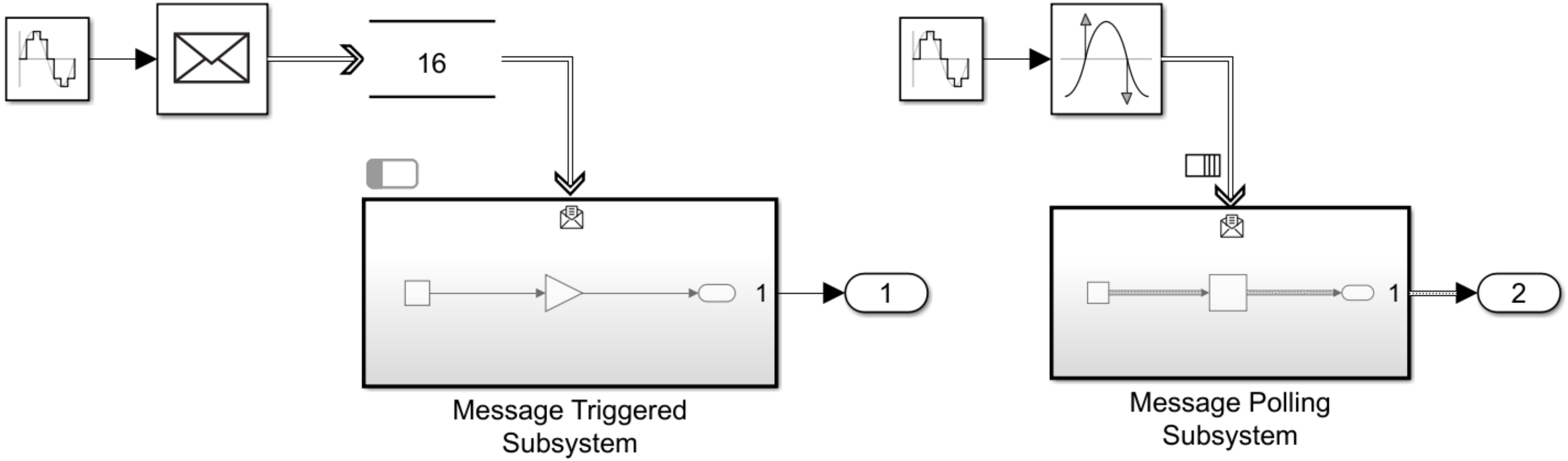






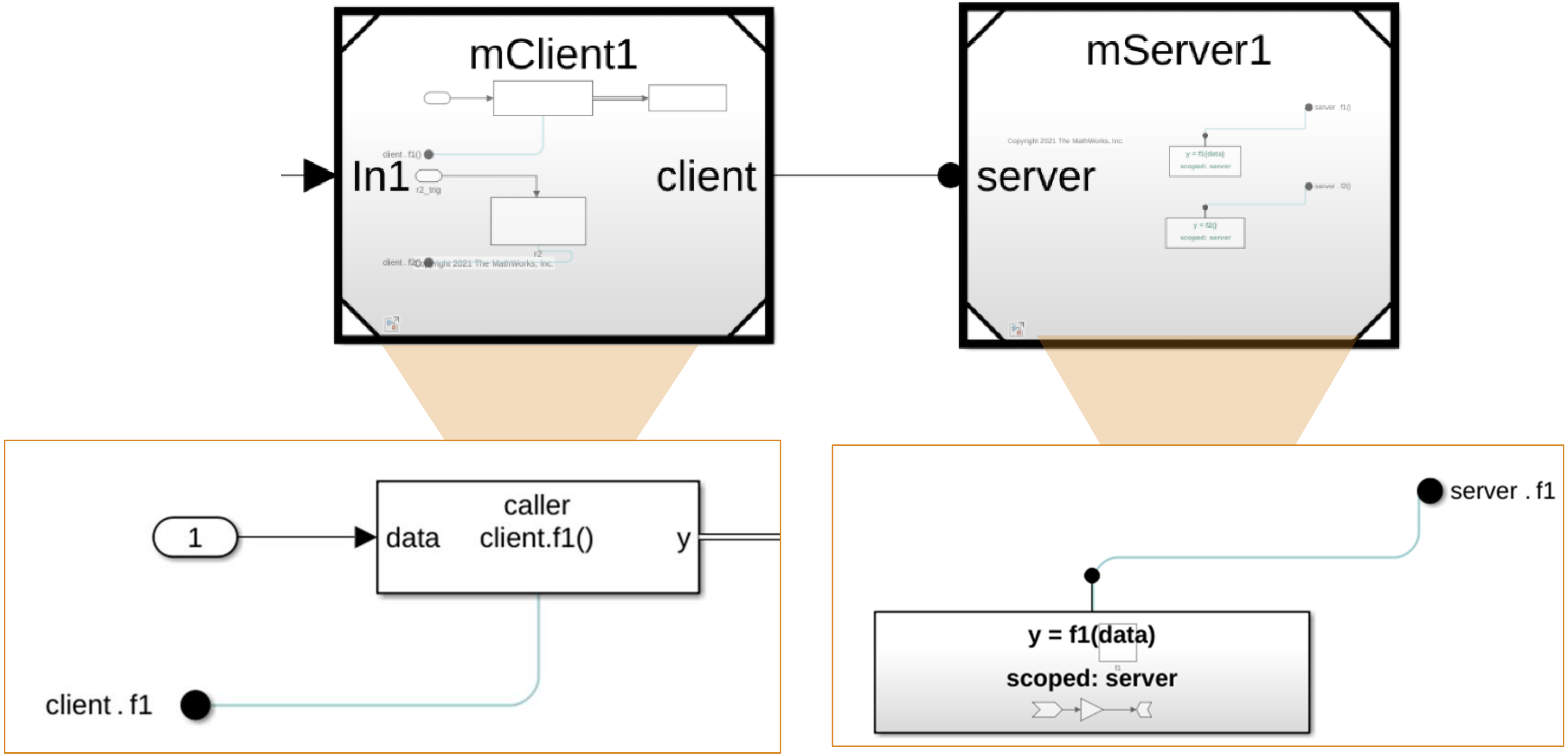
You can model service-oriented communication using messages (Send/Receive)

# Message Triggered/Polling Subsystem for SOA



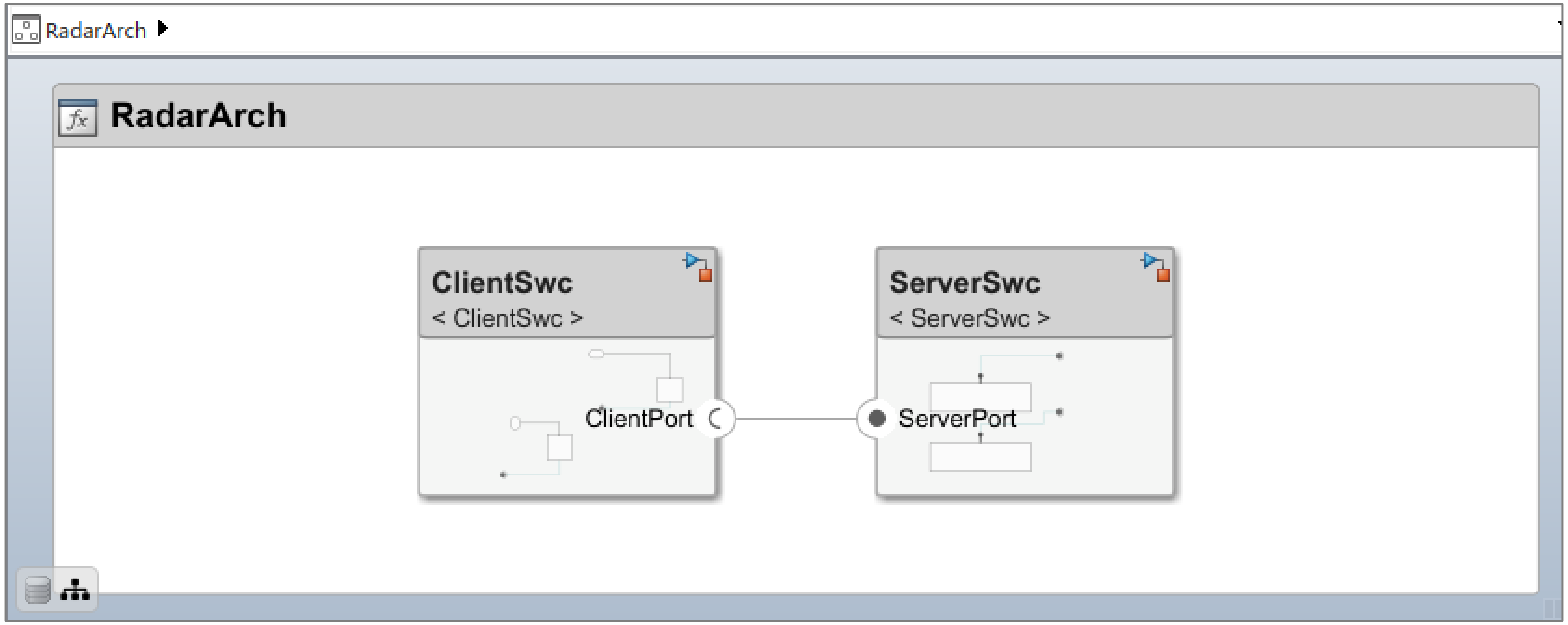
- New blocks to process messages by executing subsystem when message is available
- Model and generate code for components that are executed on message arrival

# Function Ports for SOA



Model client and server components to facilitate data sharing using a functional interface between component models

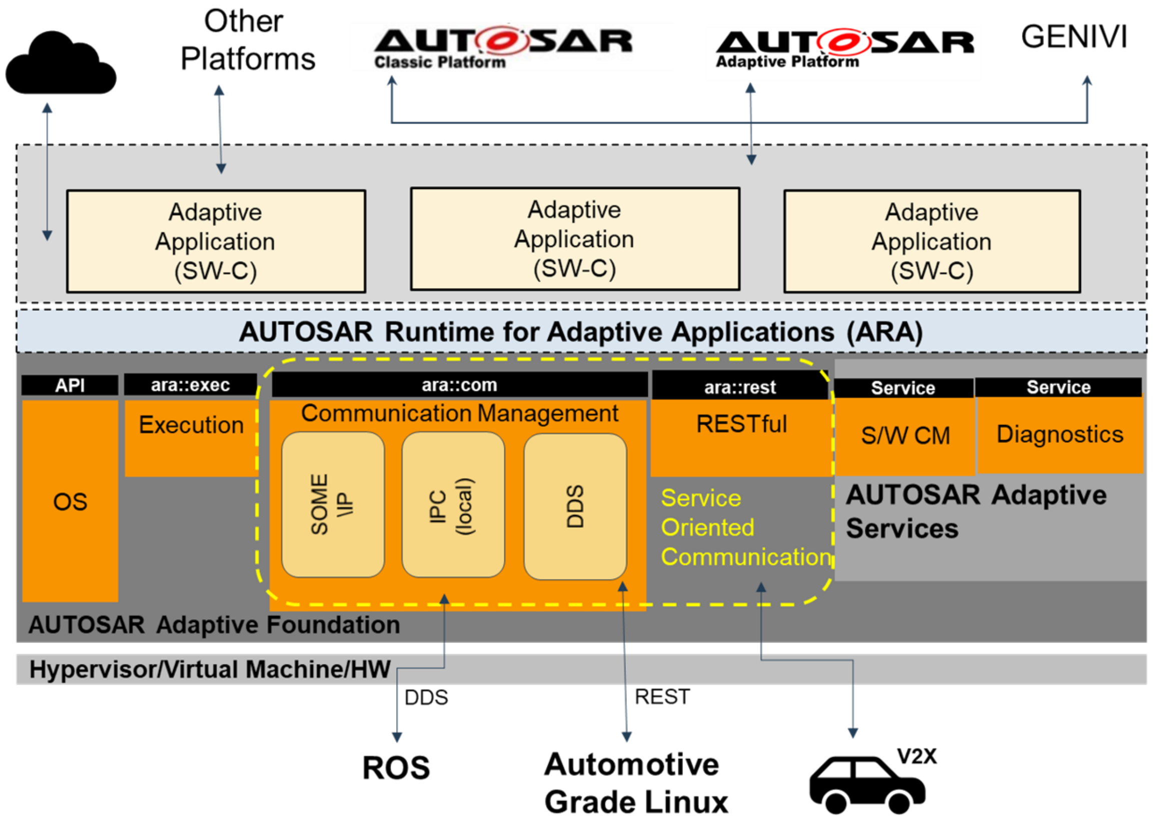
# Author SOA applications in software architecture models



Model client-server connections between software components in software architectures in System Composer

# AUTOSAR Adaptive

**AUTOSAR Adaptive** Platform implements the AUTOSAR Runtime for Adaptive Applications (ARA) for automotive industry.


Other Platforms
AUTOSAR Classic Platform
AUTOSAR Adaptive Platform
GENIVI
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
AUTOSAR Runtime for Adaptive Applications (ARA)
API
ara::exec
ara::com
ara::rest
Service
Service
OS
Execution
Communication Management
SOME \IP
IPC (local)
DDS
RESTful
Service Oriented Communication
S/W CM
Diagnostics
AUTOSAR Adaptive Services
AUTOSAR Adaptive Foundation
Hypervisor/Virtual Machine/HW
DDS
REST
ROS
Automotive Grade Linux
V2X

# AUTOSAR Adaptive Support in Simulink

**AUTOSAR Adaptive** Platform implements the AUTOSAR Runtime for Adaptive Applications (ARA) for automotive industry.

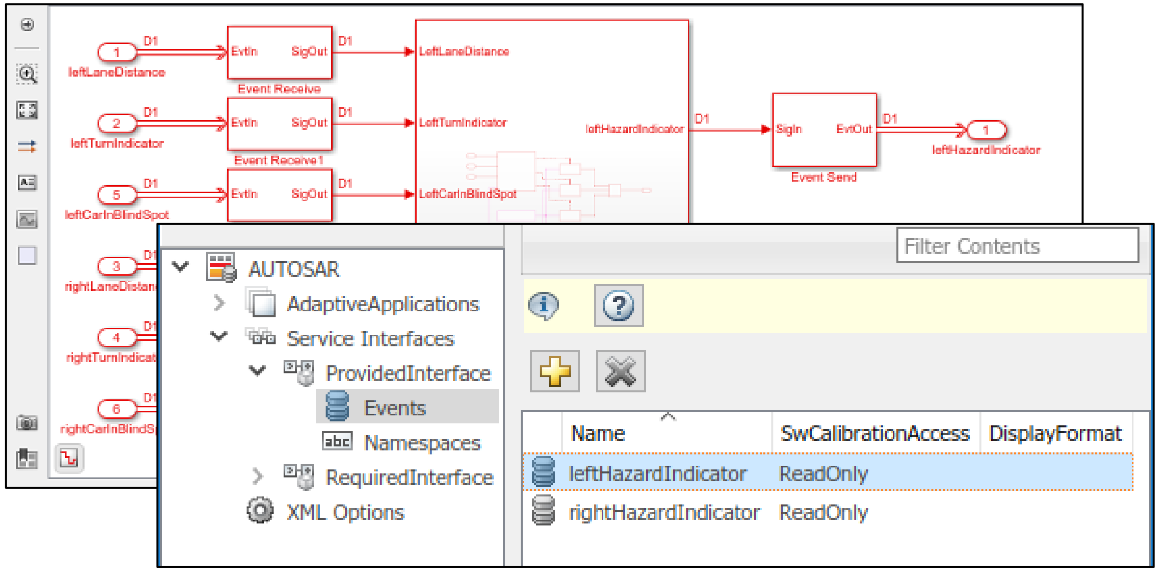
Model, simulate, test and generate C++ code for AUTOSAR Adaptive applications in Simulink.

AUTOSAR Blockset

Design and simulate AUTOSAR software

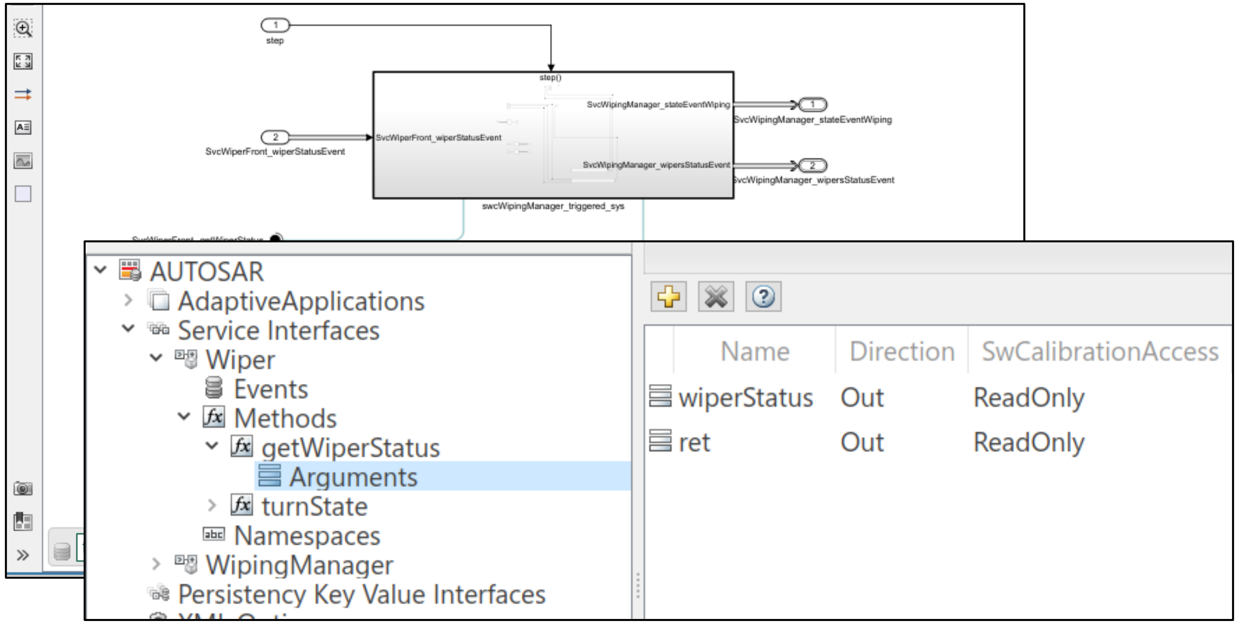
# Simulink support for AUTOSAR Adaptive

- In AUTOSAR Adaptive, services implement **communication** through:
  - **Events**
  - **Methods**
  - **Fields**



| Name | SwCalibrationAccess | DisplayFormat |
|---|---|---|
| leftHazardIndicator | ReadOnly | |
| rightHazardIndicator | ReadOnly | |

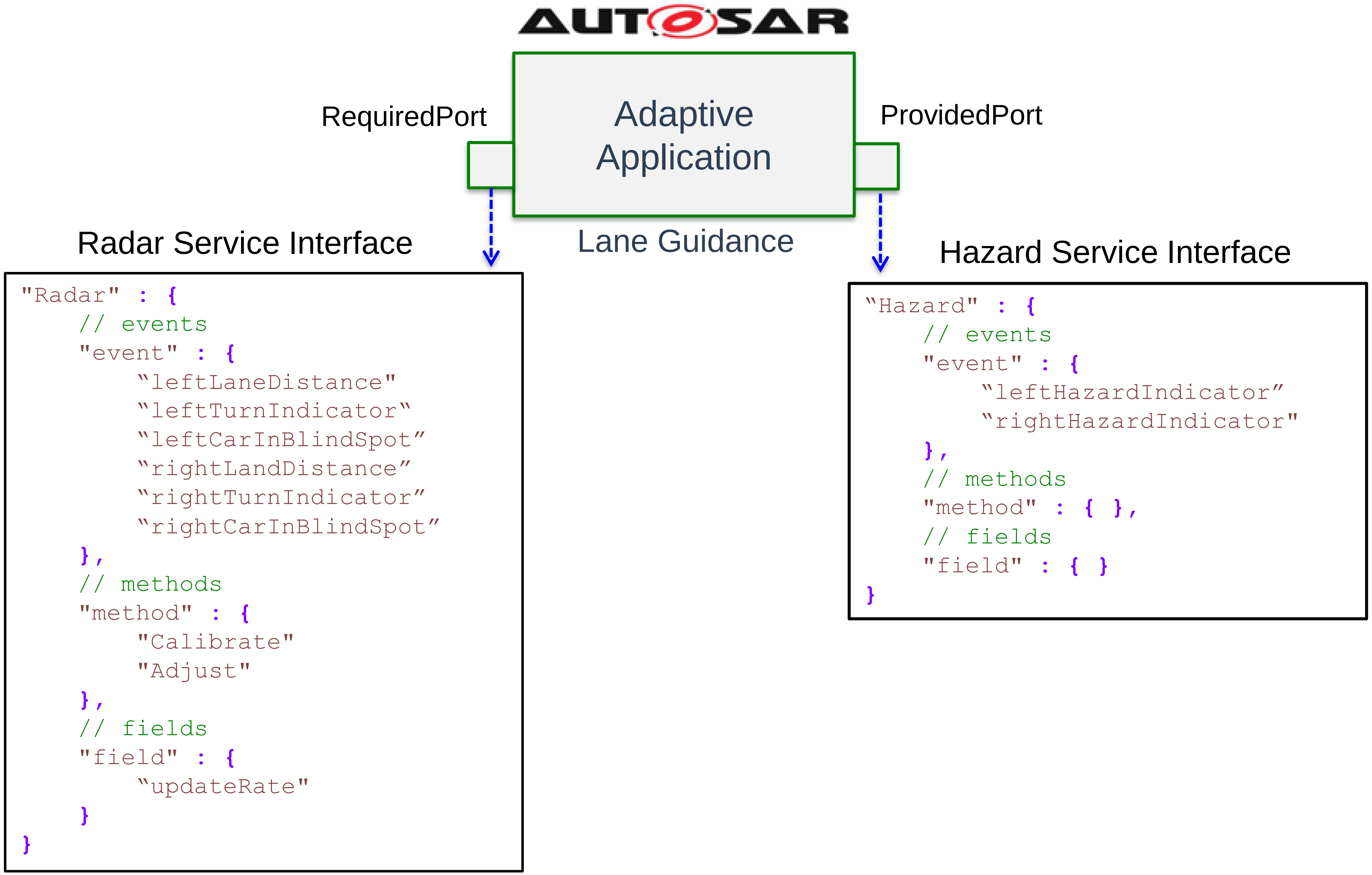
- In Simulink,
  - **ara::com Events** can be modeled as **Messages**
  - **ara::com Methods** (fire-forget and blocking/synchronous Request-Response) can be modeled using **Function Ports**


step
step()
SvcWiperFront_wiperStatusEvent
SvcWipingManager_stateEventWiping
SvcWipingManager_wipersStatusEvent
swcWipingManager_triggered_sys
AUTOSAR
AdaptiveApplications
Service Interfaces
Wiper
Events
Methods
getWiperStatus
Arguments
turnState
Namespaces
WipingManager
Persistency Key Value Interfaces

| Name | Direction | SwCalibrationAccess |
|---|---|---|
| wiperStatus | Out | ReadOnly |
| ret | Out | ReadOnly |

**Generate AUTOSAR Adaptive C++ compliant code Using Embedded Coder.**

# Adaptive SW architecture concepts



AUTOSAR

RequiredPort

Adaptive Application

ProvidedPort

Lane Guidance

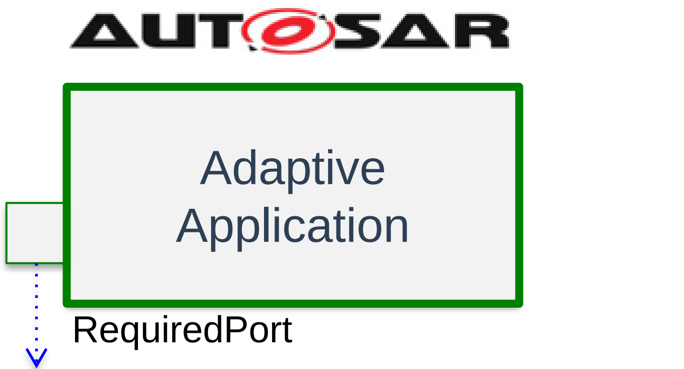
Radar Service Interface

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

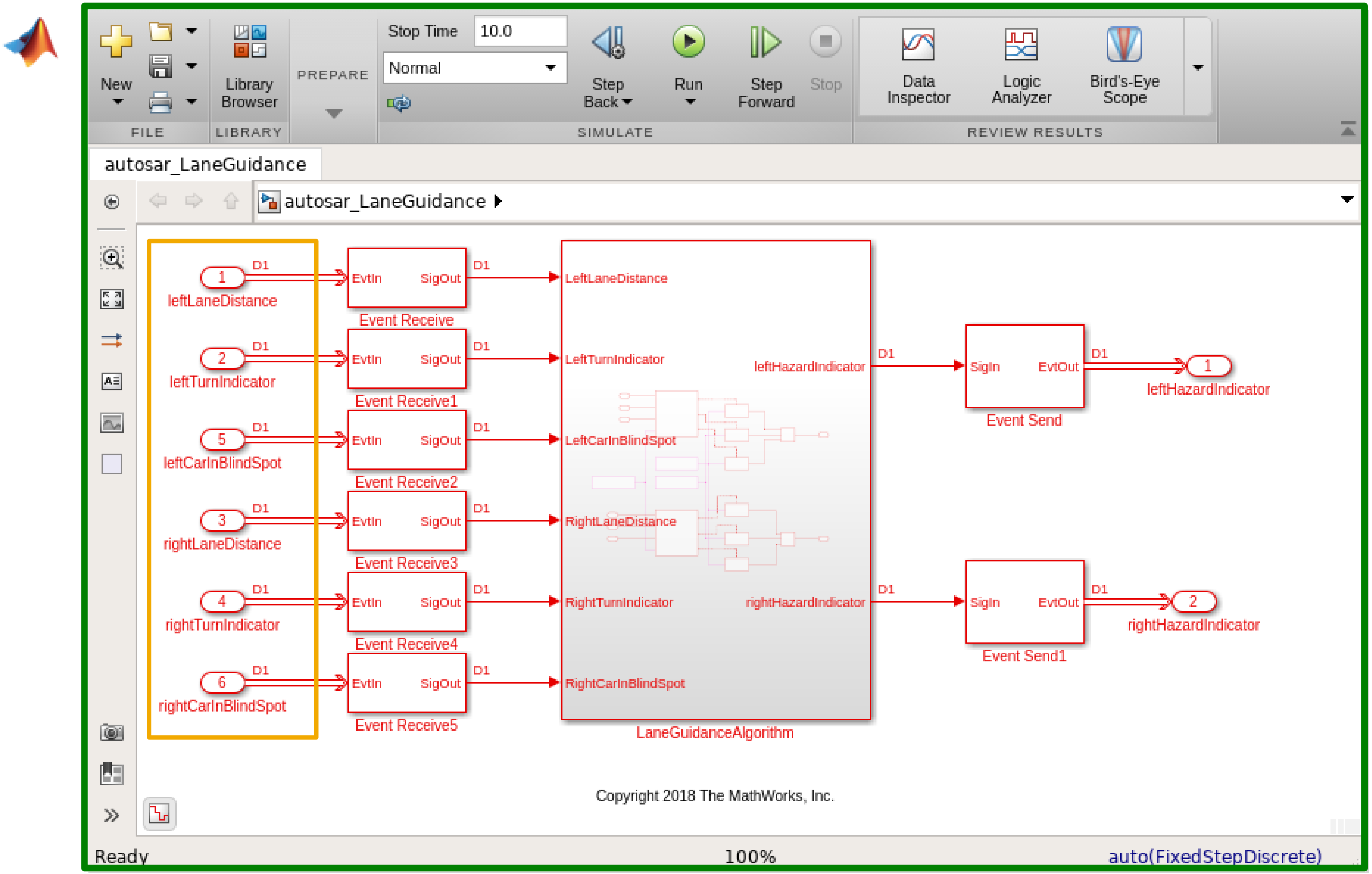
Hazard Service Interface

```
“Hazard" : {
    // events
    "event" : {
        “leftHazardIndicator”
        “rightHazardIndicator"
    },
    // methods
    "method" : { },
    // fields
    "field" : { }
}
```

# Modelling an AUTOSAR Adaptive application in Simulink

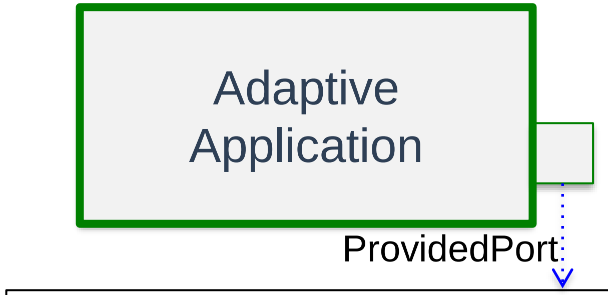


```
"Radar" : {
    // events
    "event" : {
        “leftLaneDistance"
        “leftTurnIndicator“
        “leftCarInBlindSpot”
        “rightLandDistance”
        “rightTurnIndicator”
        “rightCarInBlindSpot”
    },
    // methods
    "method" : {
        "Calibrate"
        "Adjust"
    },
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

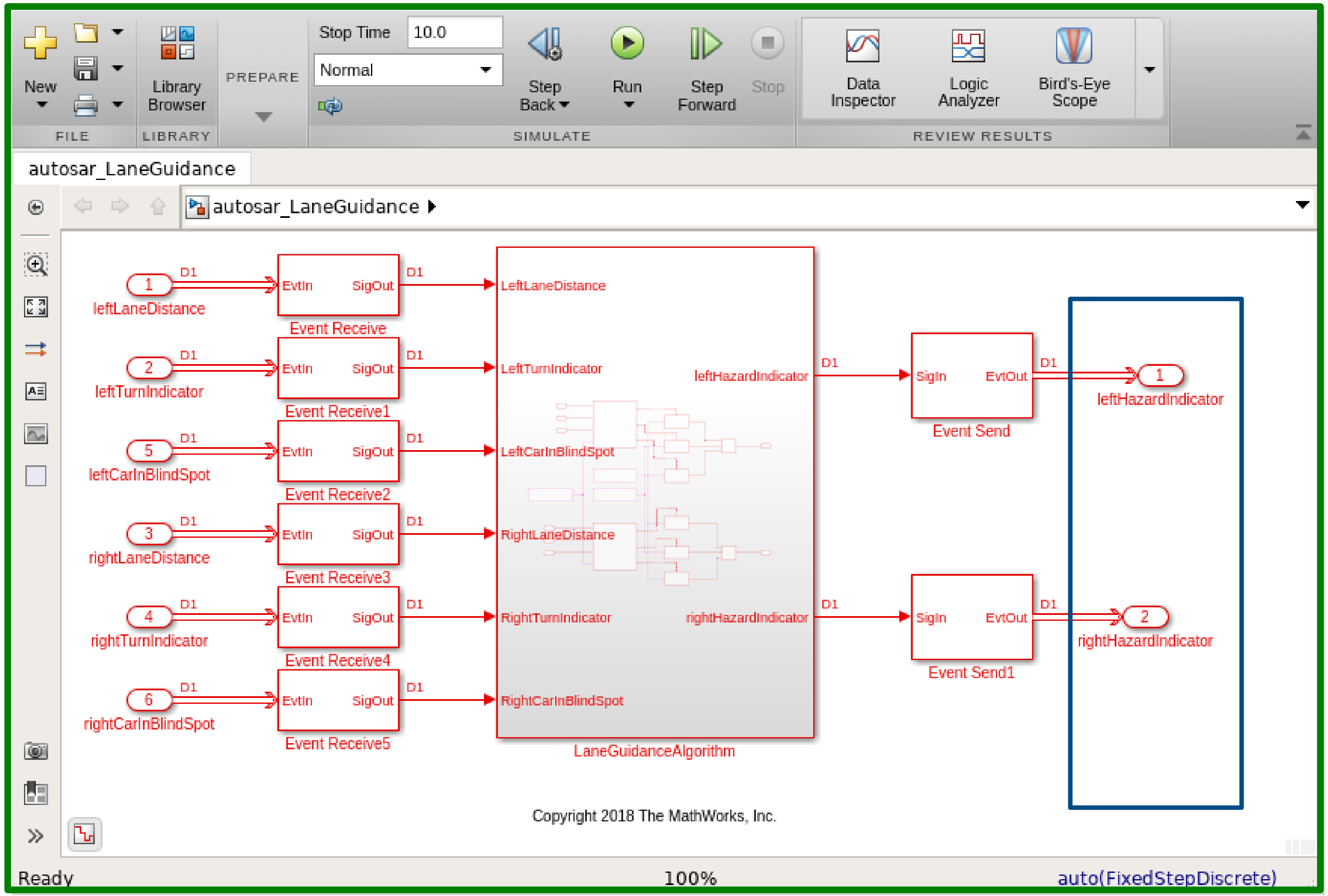


# Modelling an AUTOSAR Adaptive application in Simulink

AUTOSAR



```
“Hazard" : {
    // events
    "event" : {
        “leftHazardIndicator”
        “rightHazardIndicator"
    },
    // methods
    "method" : { },
    // fields
    "field" : { }
}
```



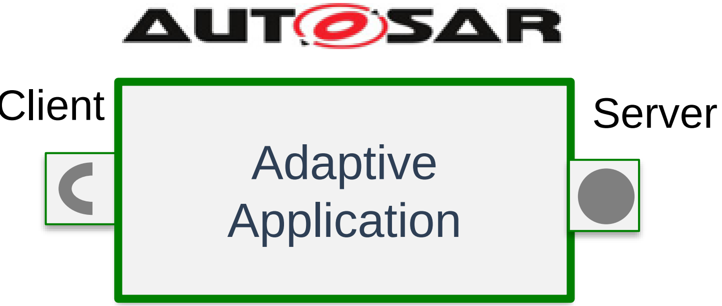
Stop Time 10.0

Normal

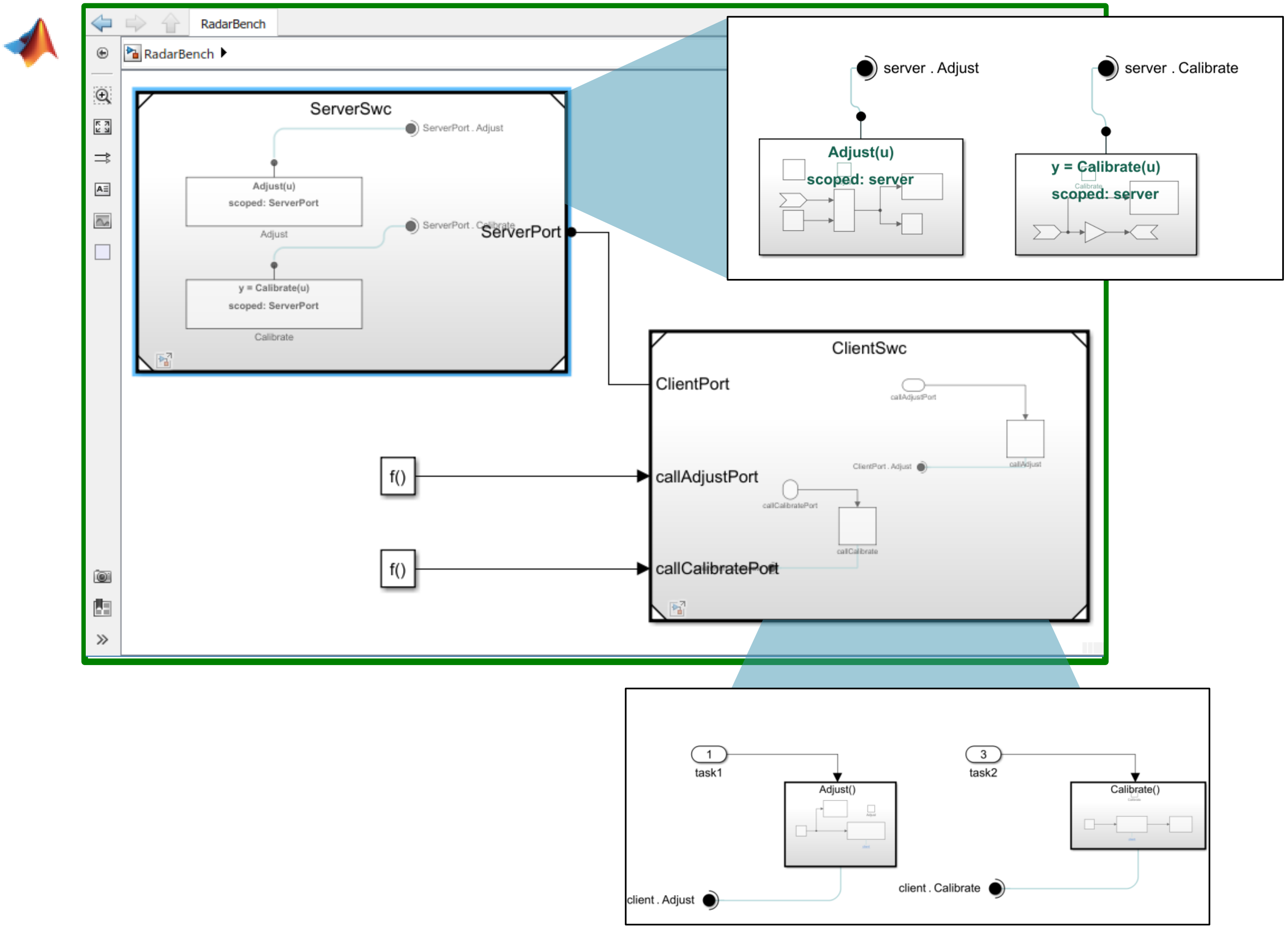
Copyright 2018 The MathWorks, Inc.

Ready 100% auto(FixedStepDiscrete)

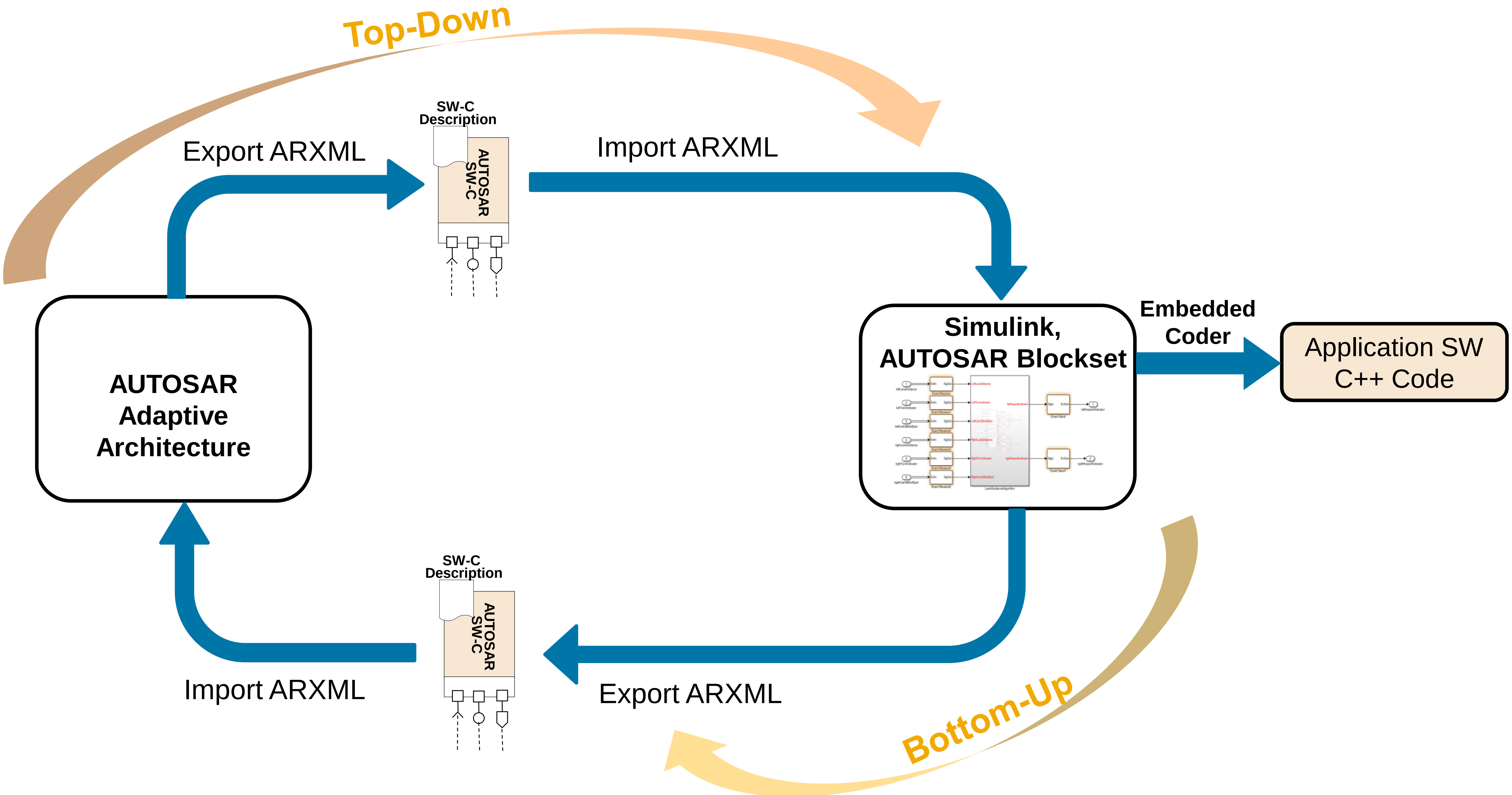
# Modelling an AUTOSAR Adaptive application in Simulink



```
“Hazard" : {
    // events
    "event" : { },
    // methods
    "method" : {
        “calibrate"
        “adjust"
    },
    // fields
    "field" : { }
}
```



# AUTOSAR Adaptive workflows



Top-Down

SW-C Description

AUTOSAR SW-C

Export ARXML

Import ARXML

AUTOSAR Adaptive Architecture

Simulink, AUTOSAR Blockset

Embedded Coder

Application SW C++ Code

SW-C Description

AUTOSAR SW-C

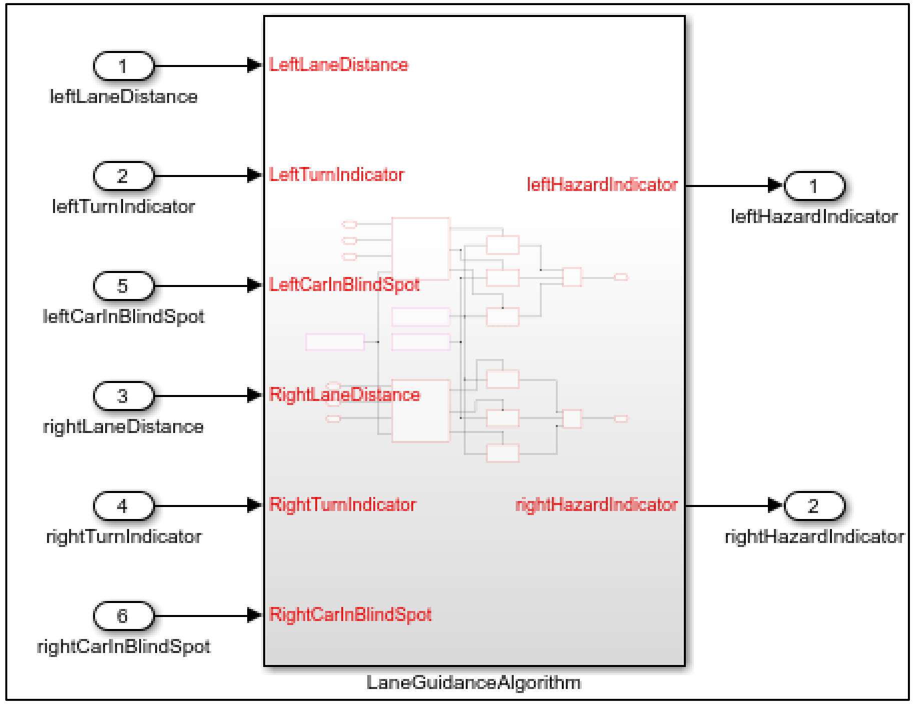
Import ARXML

Export ARXML

Bottom-Up

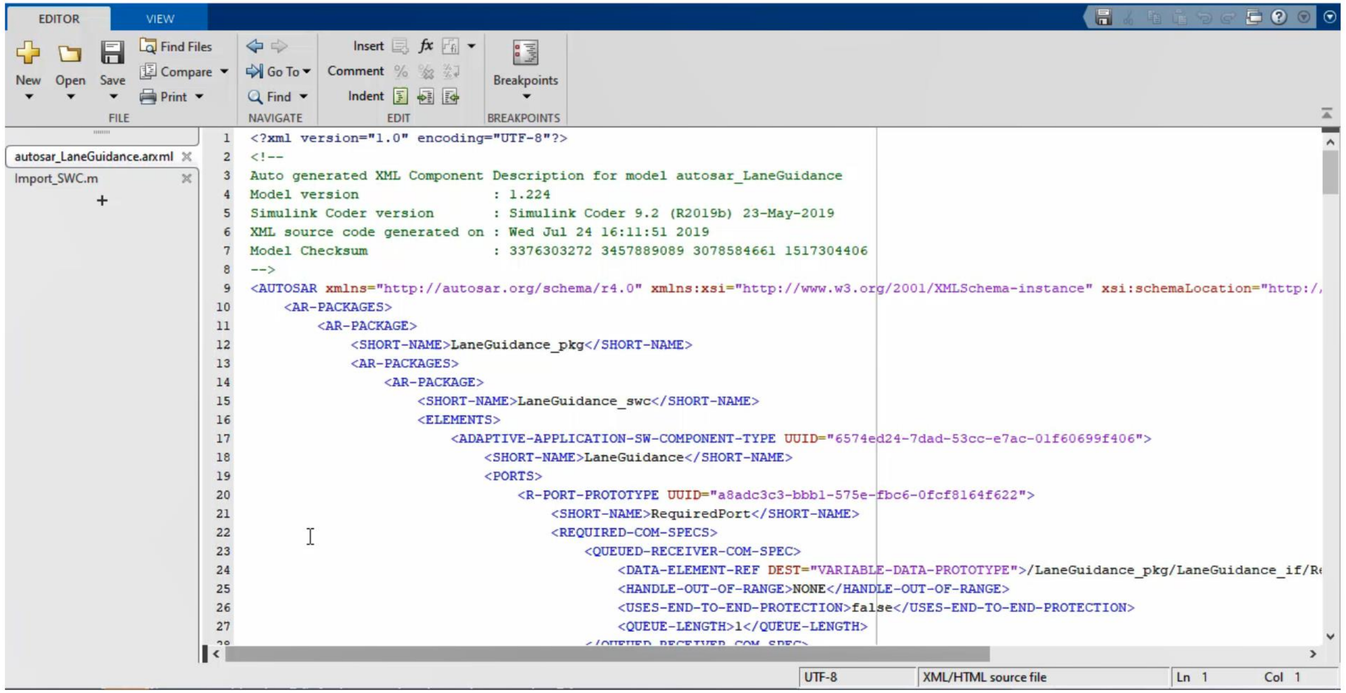
# AUTOSAR Adaptive in action

## Legacy Simulink model



OR

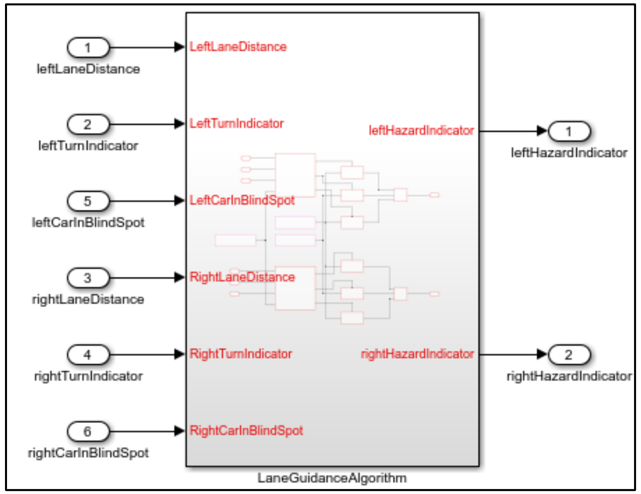
## Start from an AUTOSAR Adaptive ARXML



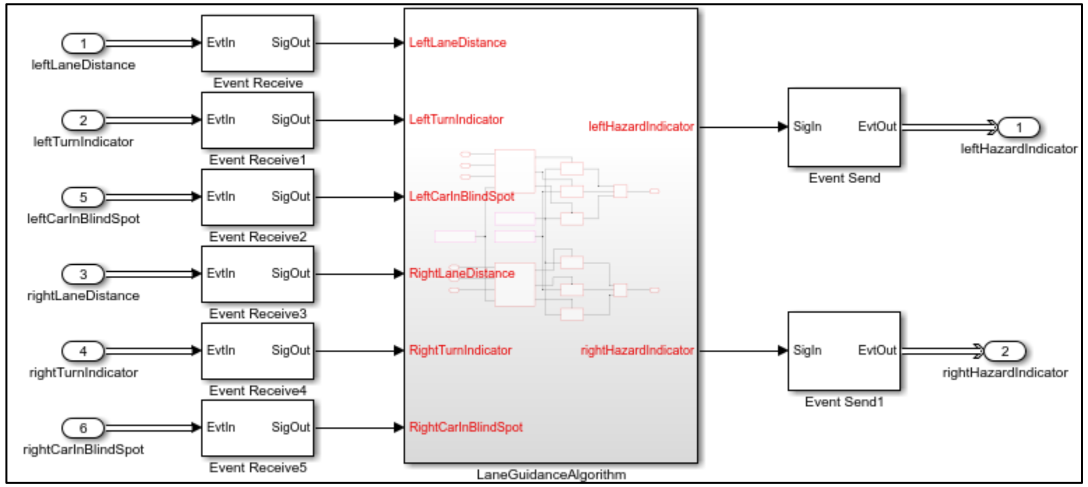
```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Auto generated XML Component Description for model autosar_LaneGuidance
Model version                : 1.224
Simulink Coder version       : Simulink Coder 9.2 (R2019b) 23-May-2019
XML source code generated on : Wed Jul 24 16:11:51 2019
Model Checksum               : 3376303272 3457889089 3078584661 1517304406
-->
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http:/,
    <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/Re
                                            <HANDLE-OUT-OF-RANGE>NONE</HANDLE-OUT-OF-RANGE>
                                            <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                                            <QUEUE-LENGTH>1</QUEUE-LENGTH>
                                        </QUEUED-RECEIVER-COM-SPEC>
```

# AUTOSAR Adaptive in action

Legacy Simulink model

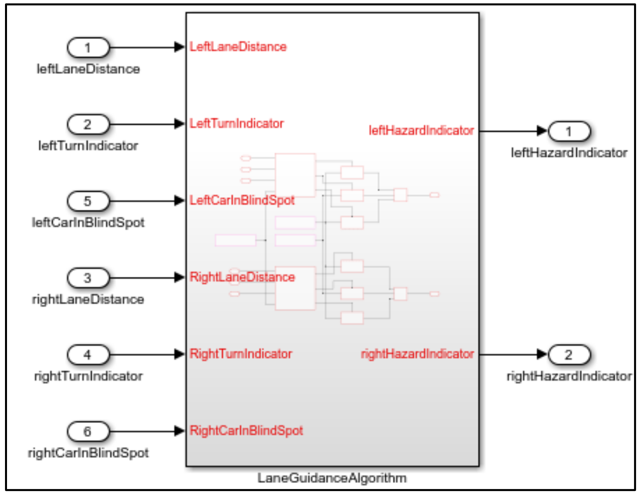


Add blocks to make the necessary event and signal connections

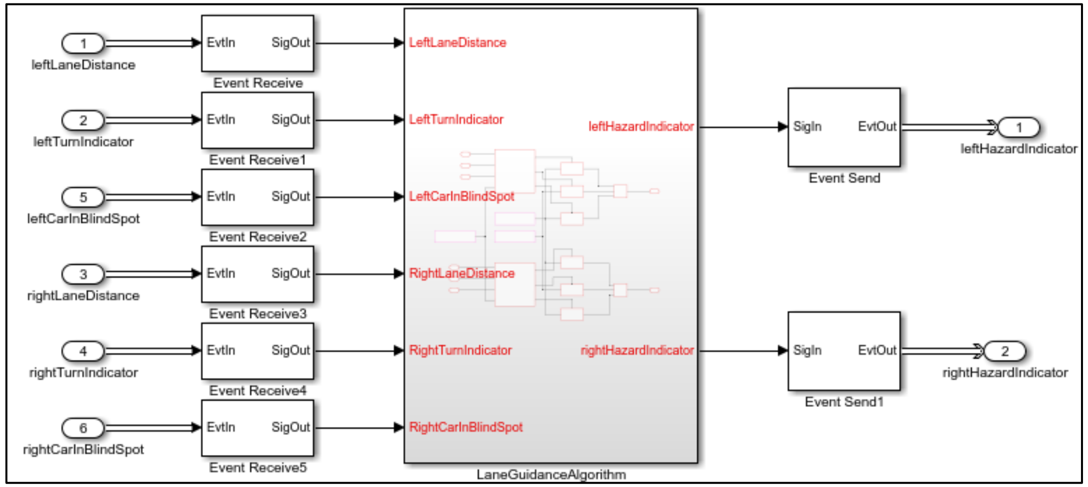


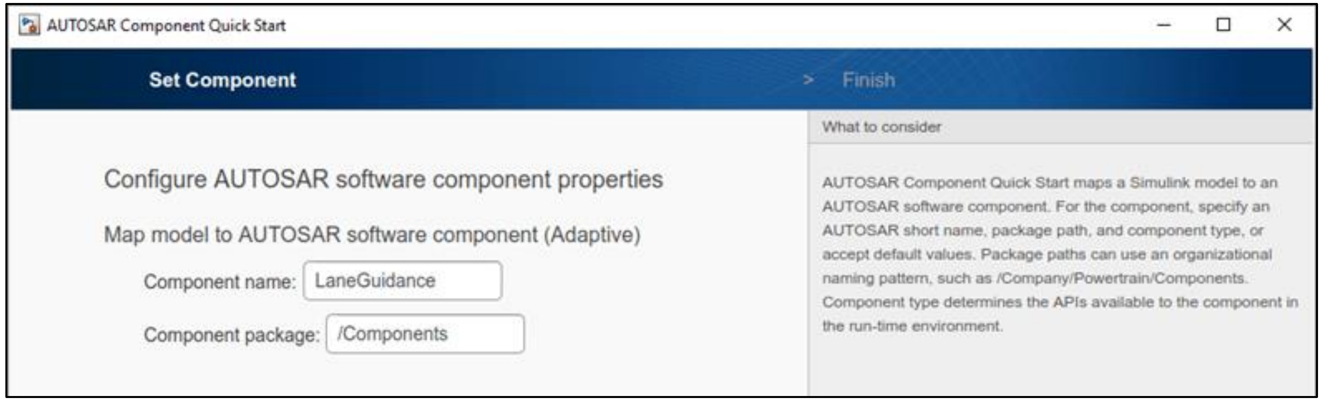
# AUTOSAR Adaptive in action

## Legacy Simulink model



## Add blocks to make the necessary event and signal connections





Configure AUTOSAR software component properties

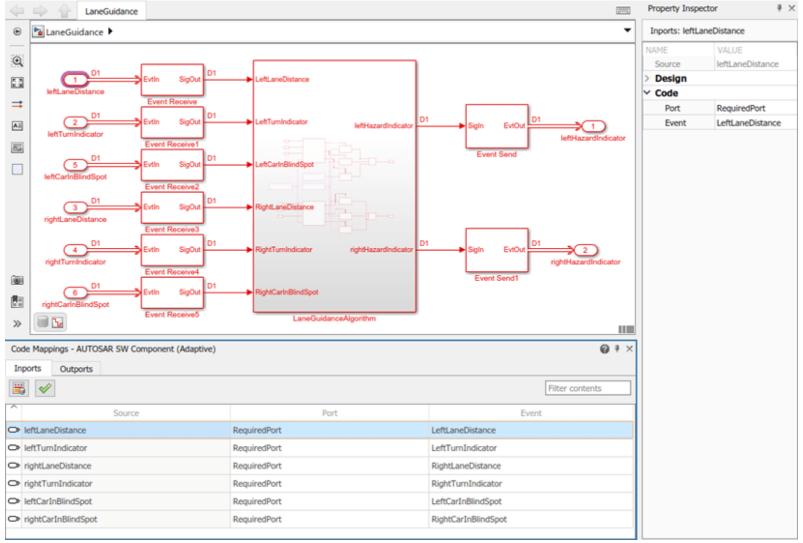
Map model to AUTOSAR software component (Adaptive)

Component name: LaneGuidance

Component package: /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.



Inports: leftLaneDistance

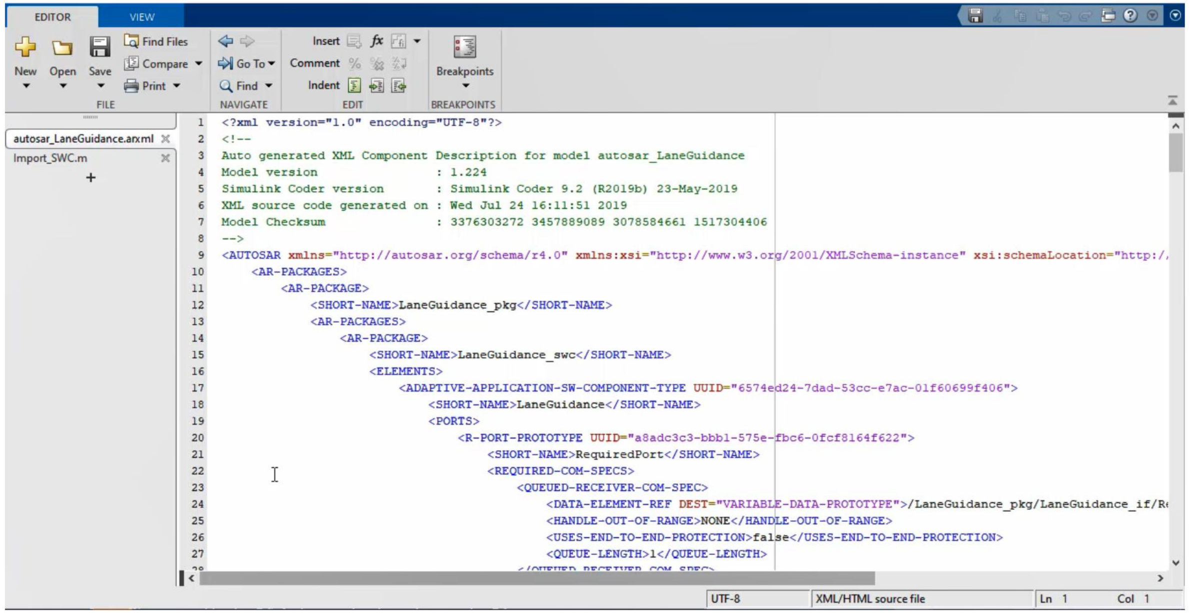
| NAME | VALUE |
|---|---|
| Source | leftLaneDistance |
| Design | |
| Code | |
| Port | RequiredPort |
| Event | LeftLaneDistance |

Code Mappings - AUTOSAR SW Component (Adaptive)

| Source | Port | Event |
|---|---|---|
| leftLaneDistance | RequiredPort | LeftLaneDistance |
| leftTurnIndicator | RequiredPort | LeftTurnIndicator |
| rightLaneDistance | RequiredPort | RightLaneDistance |
| rightTurnIndicator | RequiredPort | RightTurnIndicator |
| leftCarInBlindSpot | RequiredPort | LeftCarInBlindSpot |
| rightCarInBlindSpot | RequiredPort | RightCarInBlindSpot |

# AUTOSAR Adaptive in action

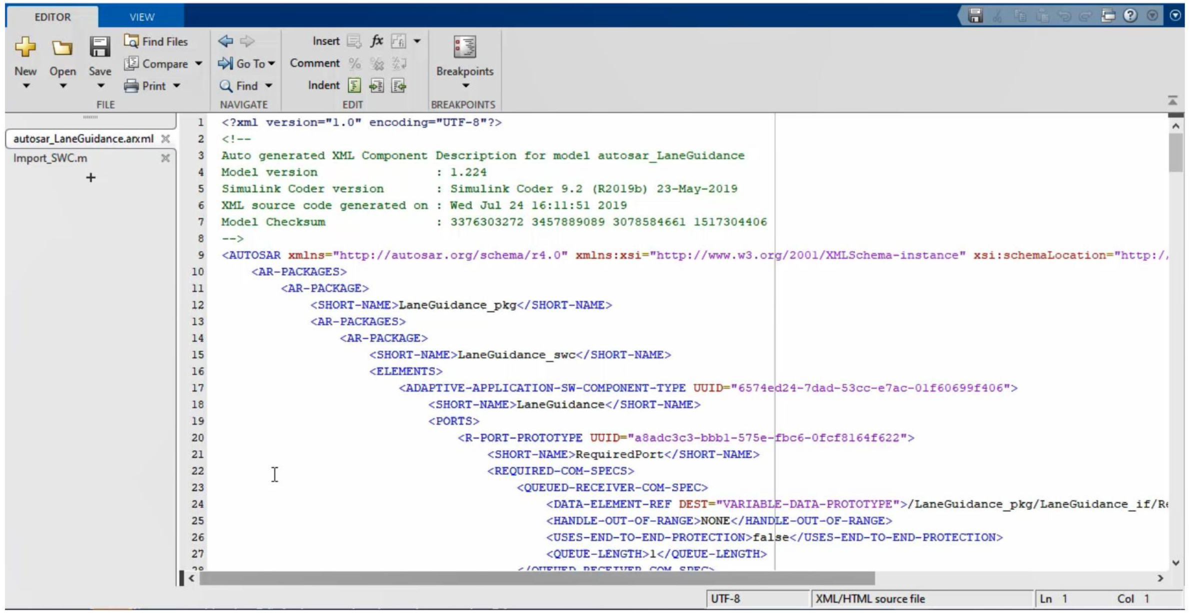
- Create model from ARXML



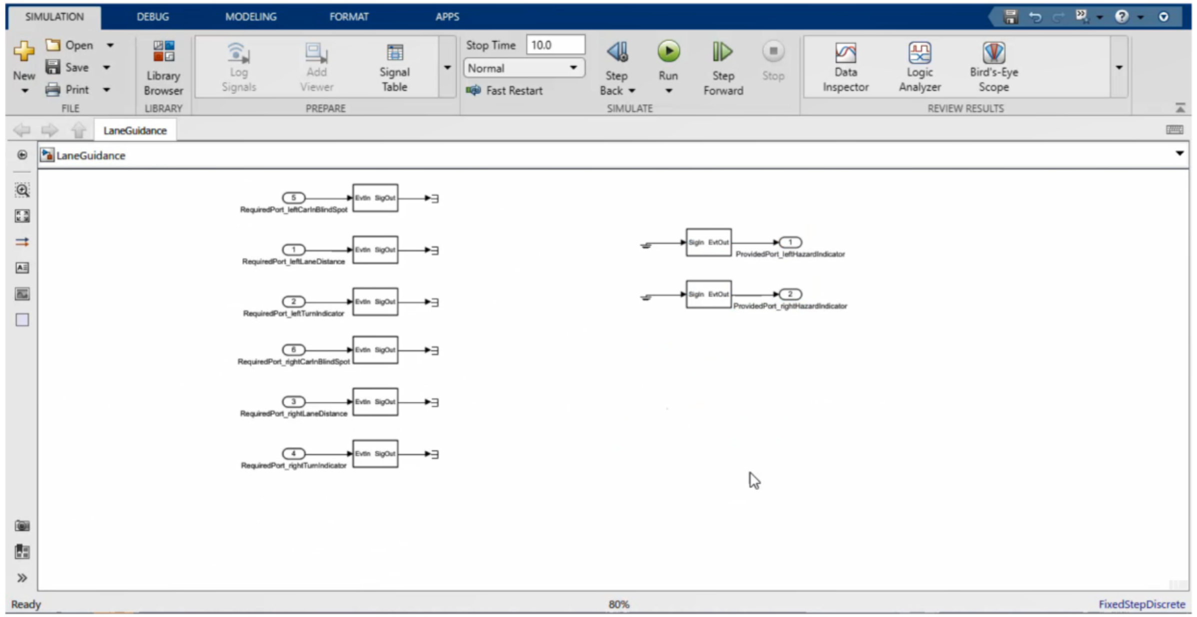
```
<?xml version="1.0" encoding="UTF-8"?>
<!--
Auto generated XML Component Description for model autosar_LaneGuidance
Model version                : 1.224
Simulink Coder version       : Simulink Coder 9.2 (R2019b) 23-May-2019
XML source code generated on : Wed Jul 24 16:11:51 2019
Model Checksum               : 3376303272 3457889089 3078584661 1517304406
-->
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http:/
    <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/R
                                            <HANDLE-OUT-OF-RANGE>NONE</HANDLE-OUT-OF-RANGE>
                                            <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                                            <QUEUE-LENGTH>1</QUEUE-LENGTH>
                                        </QUEUED-RECEIVER-COM-SPEC>
```

# AUTOSAR Adaptive in action

- Create model from ARXML



```xml
<?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 9.2 (R2019b) 23-May-2019
XML source code generated on : Wed Jul 24 16:11:51 2019
Model Checksum                 : 3376303272 3457889089 3078584661 1517304406
-->
<AUTOSAR xmlns="http://autosar.org/schema/r4.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http:/
    <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/R
                                            <HANDLE-OUT-OF-RANGE>NONE</HANDLE-OUT-OF-RANGE>
                                            <USES-END-TO-END-PROTECTION>false</USES-END-TO-END-PROTECTION>
                                            <QUEUE-LENGTH>1</QUEUE-LENGTH>
```



Stop Time 10.0

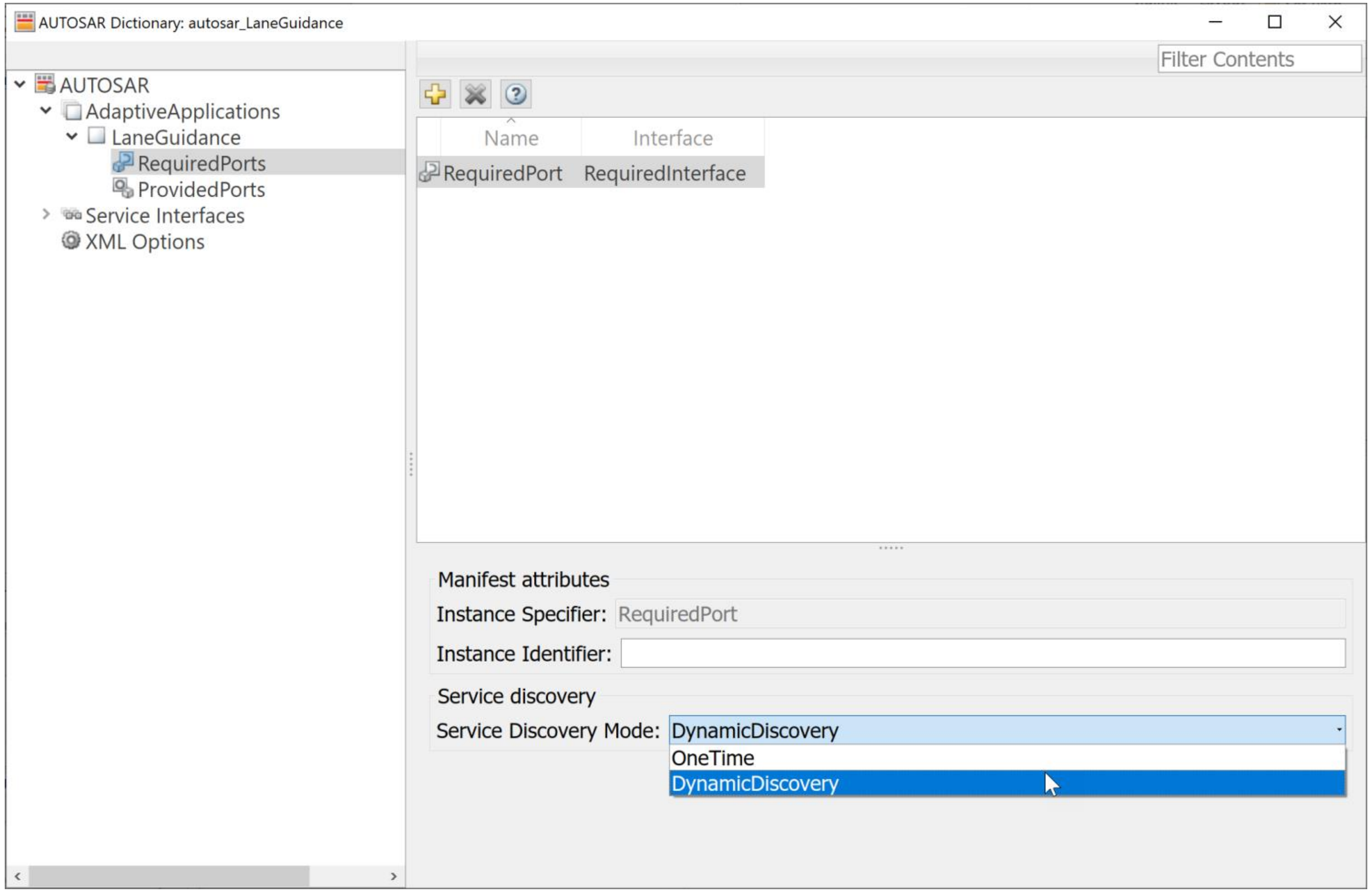
Normal

# AUTOSAR Adaptive in action

- Create model from ARXML

- Configure Service Discovery

  Subscribe to adaptive services
  - Only at startup, or
  - Dynamically, as they become available



| Name | Interface |
|---|---|
| RequiredPort | RequiredInterface |

Manifest attributes

Instance Specifier: RequiredPort

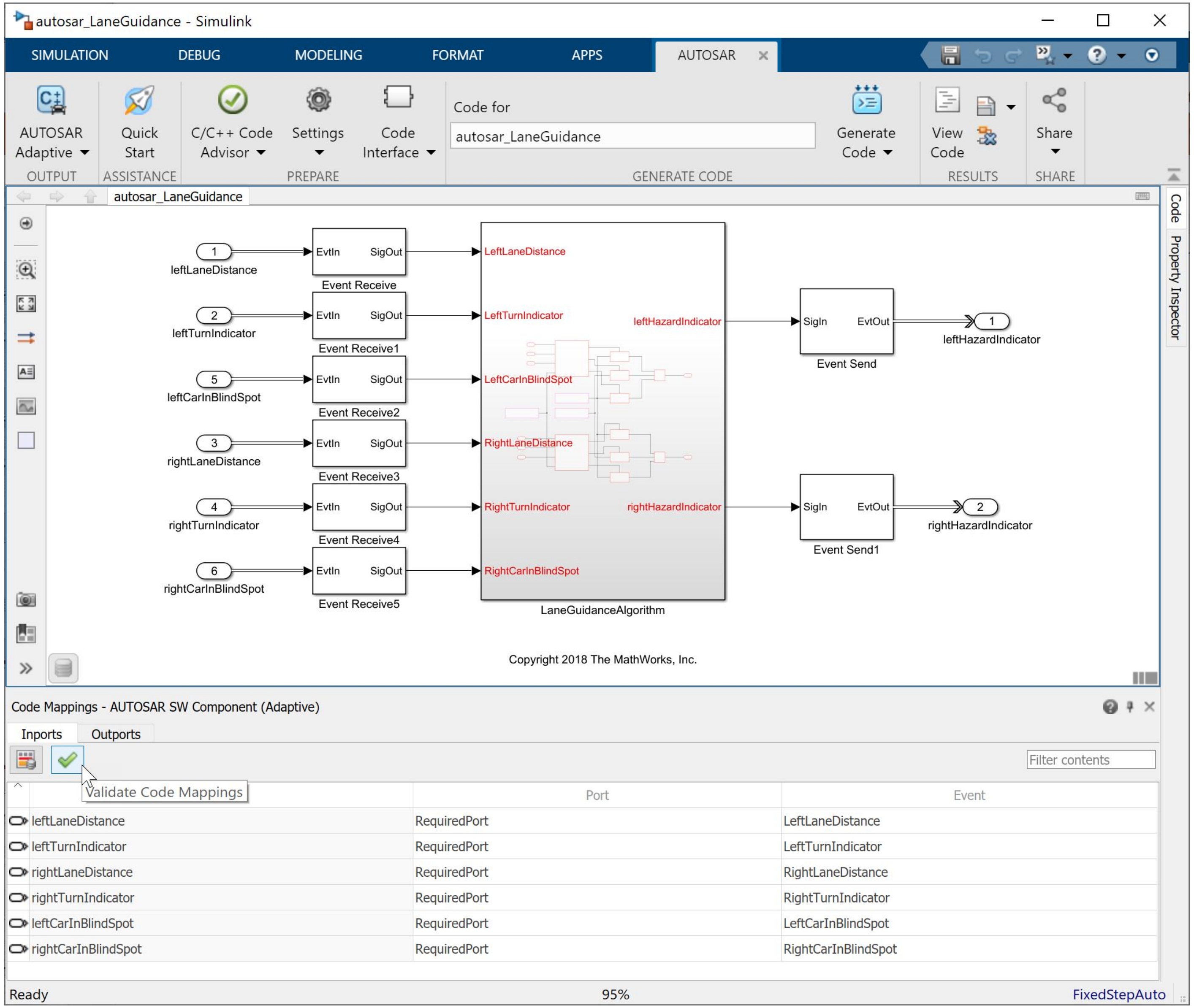
Instance Identifier:

Service discovery

Service Discovery Mode: DynamicDiscovery

# AUTOSAR Adaptive in action

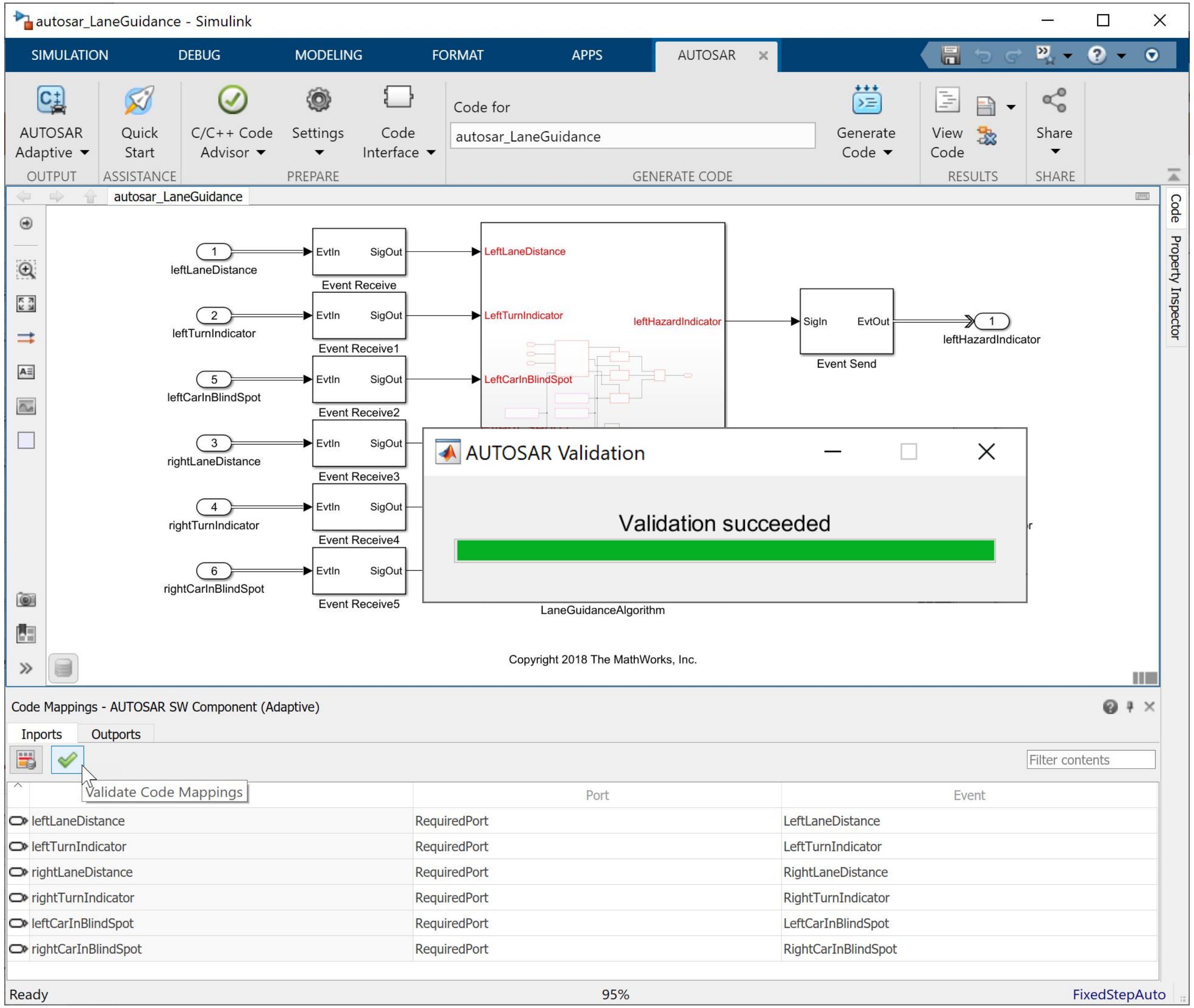
- Create model from ARXML
- Configure Service Discovery
- Verify AUTOSAR properties



| | Port | Event |
|---|---|---|
| leftLaneDistance | RequiredPort | LeftLaneDistance |
| leftTurnIndicator | RequiredPort | LeftTurnIndicator |
| rightLaneDistance | RequiredPort | RightLaneDistance |
| rightTurnIndicator | RequiredPort | RightTurnIndicator |
| leftCarInBlindSpot | RequiredPort | LeftCarInBlindSpot |
| rightCarInBlindSpot | RequiredPort | RightCarInBlindSpot |

# AUTOSAR Adaptive in action

- Create model from ARXML
- Configure Service Discovery
- Verify AUTOSAR properties



AUTOSAR Validation

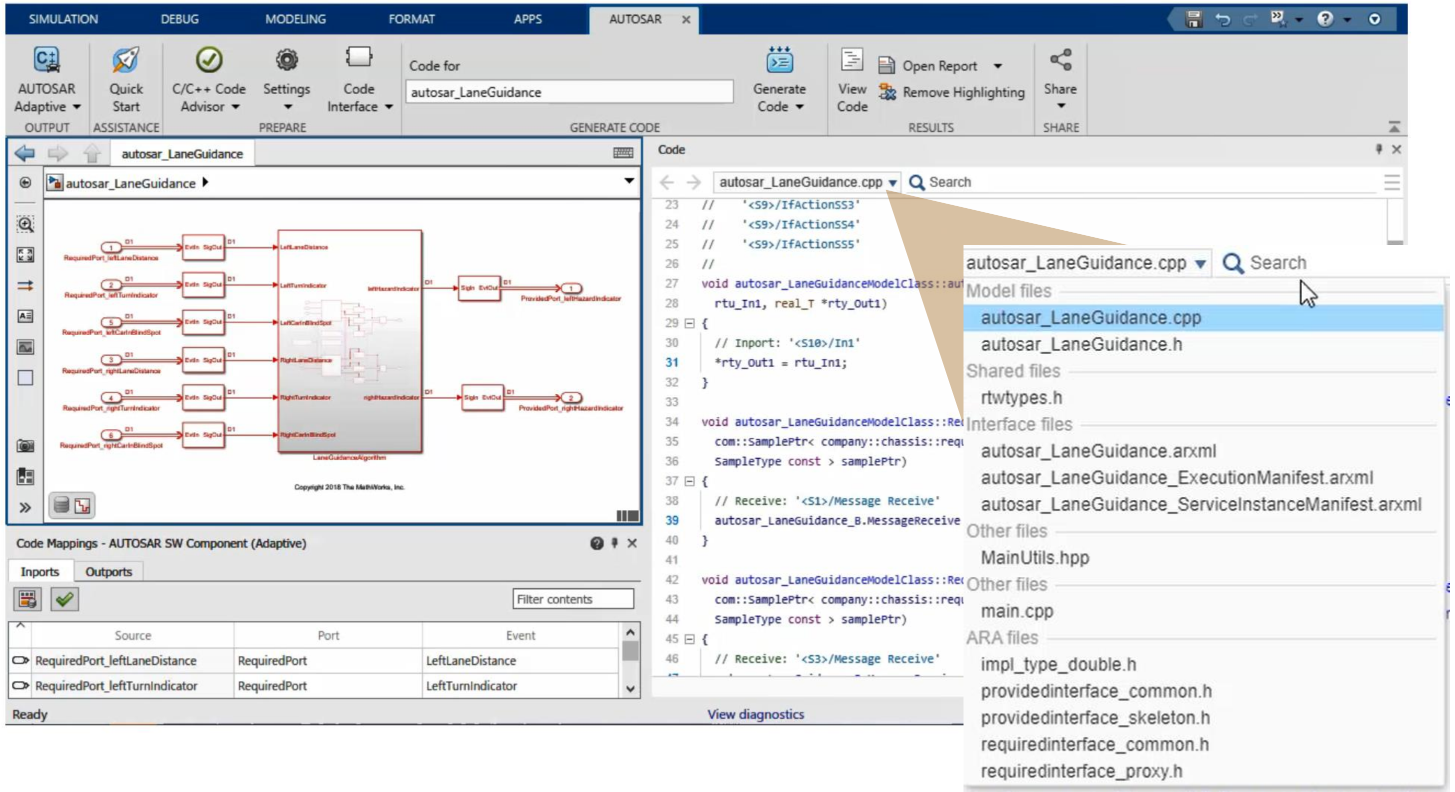
Validation succeeded

Code Mappings - AUTOSAR SW Component (Adaptive)

| | Port | Event |
|---|---|---|
| leftLaneDistance | RequiredPort | LeftLaneDistance |
| leftTurnIndicator | RequiredPort | LeftTurnIndicator |
| rightLaneDistance | RequiredPort | RightLaneDistance |
| rightTurnIndicator | RequiredPort | RightTurnIndicator |
| leftCarInBlindSpot | RequiredPort | LeftCarInBlindSpot |
| rightCarInBlindSpot | RequiredPort | RightCarInBlindSpot |

# AUTOSAR Adaptive in action

- Create model from ARXML
- Configure Service Discovery
- Verify AUTOSAR properties
- Generate code and ARXML



```
//     '<S9>/IfActionSS3'
//     '<S9>/IfActionSS4'
//     '<S9>/IfActionSS5'
//
void autosar_LaneGuidanceModelClass::aut
  rtu_In1, real_T *rty_Out1)
{
  // Inport: '<S10>/In1'
  *rty_Out1 = rtu_In1;
}

void autosar_LaneGuidanceModelClass::Re
  com::SamplePtr< company::chassis::requ
  SampleType const > samplePtr)
{
  // Receive: '<S1>/Message Receive'
  autosar_LaneGuidance_B.MessageReceive
}

void autosar_LaneGuidanceModelClass::Re
  com::SamplePtr< company::chassis::requ
  SampleType const > samplePtr)
{
  // Receive: '<S3>/Message Receive'
```

autosar_LaneGuidance.cpp ▾ Search

Model files
- autosar_LaneGuidance.cpp
- autosar_LaneGuidance.h

Shared files
- rtwtypes.h

Interface files
- autosar_LaneGuidance.arxml
- autosar_LaneGuidance_ExecutionManifest.arxml
- autosar_LaneGuidance_ServiceInstanceManifest.arxml

Other files
- MainUtils.hpp

Other files
- main.cpp

ARA files
- impl_type_double.h
- providedinterface_common.h
- providedinterface_skeleton.h
- requiredinterface_common.h
- requiredinterface_proxy.h

Code Mappings - AUTOSAR SW Component (Adaptive)

Inports | Outports

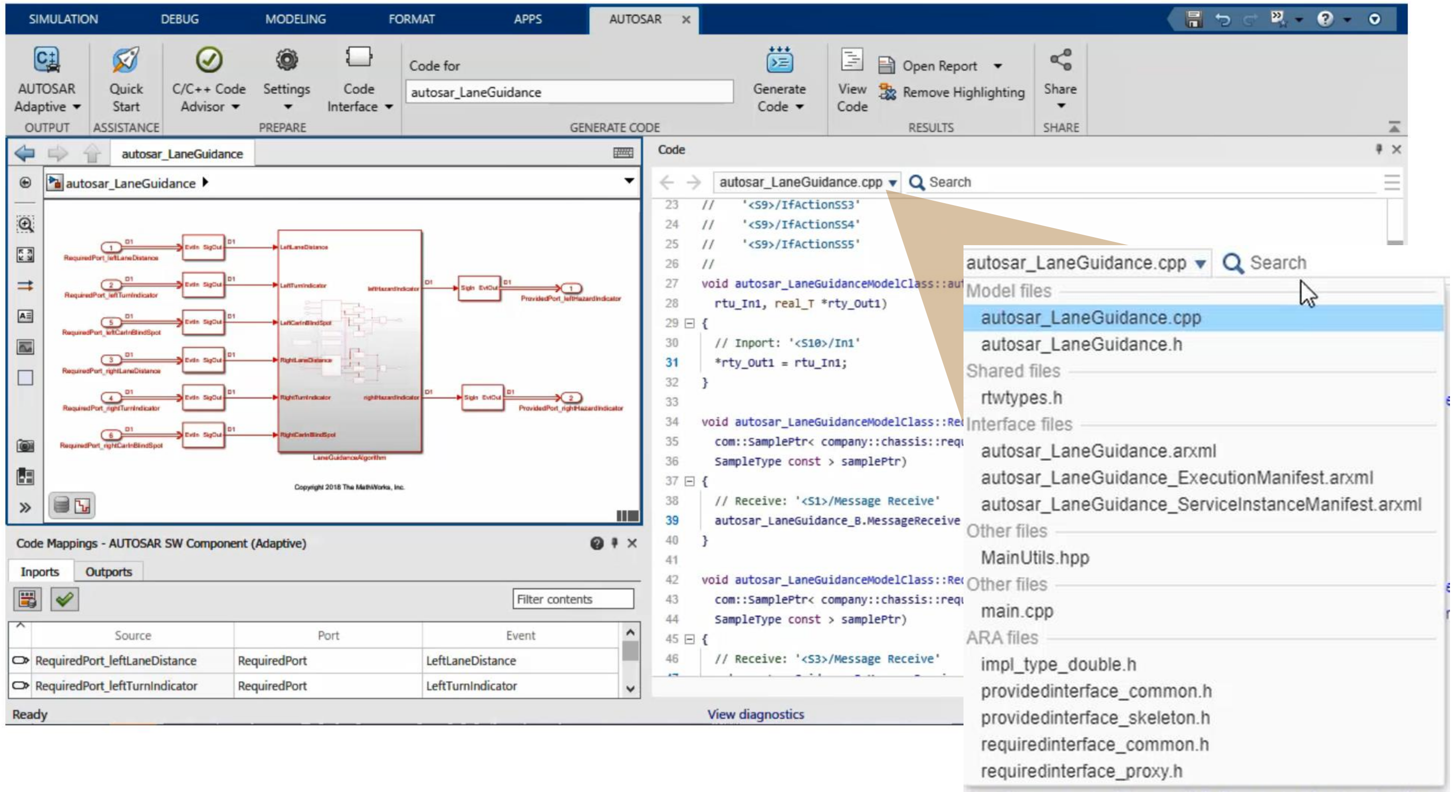
Filter contents

| Source | Port | Event |
|---|---|---|
| RequiredPort_leftLaneDistance | RequiredPort | LeftLaneDistance |
| RequiredPort_leftTurnIndicator | RequiredPort | LeftTurnIndicator |

Ready

# AUTOSAR Adaptive in action

- Create model from ARXML
- Configure Service Discovery
- Verify AUTOSAR properties
- Generate code



autosar_LaneGuidance.cpp

```
//     '<S9>/IfActionSS3'
//     '<S9>/IfActionSS4'
//     '<S9>/IfActionSS5'
//
void autosar_LaneGuidanceModelClass::au
  rtu_In1, real_T *rty_Out1)
{
  // Inport: '<S10>/In1'
  *rty_Out1 = rtu_In1;
}

void autosar_LaneGuidanceModelClass::Re
  com::SamplePtr< company::chassis::req
  SampleType const > samplePtr)
{
  // Receive: '<S1>/Message Receive'
  autosar_LaneGuidance_B.MessageReceive
}

void autosar_LaneGuidanceModelClass::Re
  com::SamplePtr< company::chassis::req
  SampleType const > samplePtr)
{
  // Receive: '<S3>/Message Receive'
```

autosar_LaneGuidance.cpp

Model files
- autosar_LaneGuidance.cpp
- autosar_LaneGuidance.h

Shared files
- rtwtypes.h

Interface files
- autosar_LaneGuidance.arxml
- autosar_LaneGuidance_ExecutionManifest.arxml
- autosar_LaneGuidance_ServiceInstanceManifest.arxml

Other files
- MainUtils.hpp

Other files
- main.cpp

ARA files
- impl_type_double.h
- providedinterface_common.h
- providedinterface_skeleton.h
- requiredinterface_common.h
- requiredinterface_proxy.h

Code Mappings - AUTOSAR SW Component (Adaptive)

Inports | Outports

| Source | Port | Event |
|---|---|---|
| RequiredPort_leftLaneDistance | RequiredPort | LeftLaneDistance |
| RequiredPort_leftTurnIndicator | RequiredPort | LeftTurnIndicator |

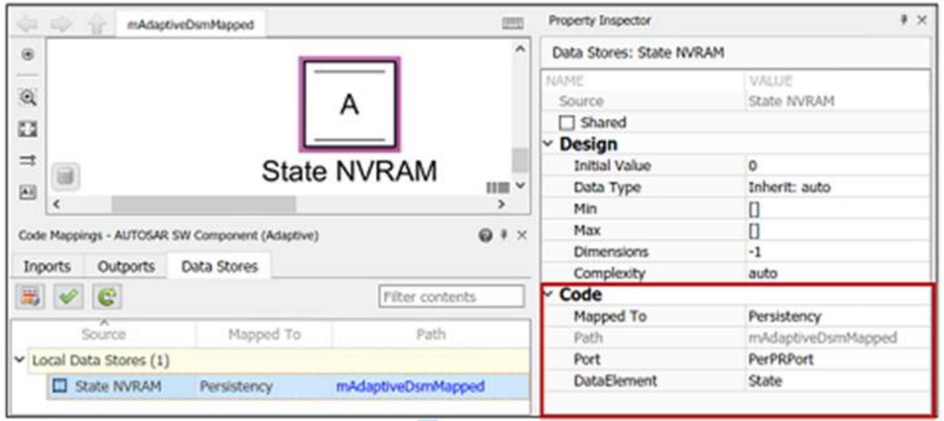
- Integrate Applications with third-party Adaptive stack
- Build Out of the Box Linux Executable from AUTOSAR Adaptive Model

# AUTOSAR Adaptive in Action

## Persistent Memory (ara::per)

**Model Persistent Memory (ara::per) for AUTOSAR adaptive applications**

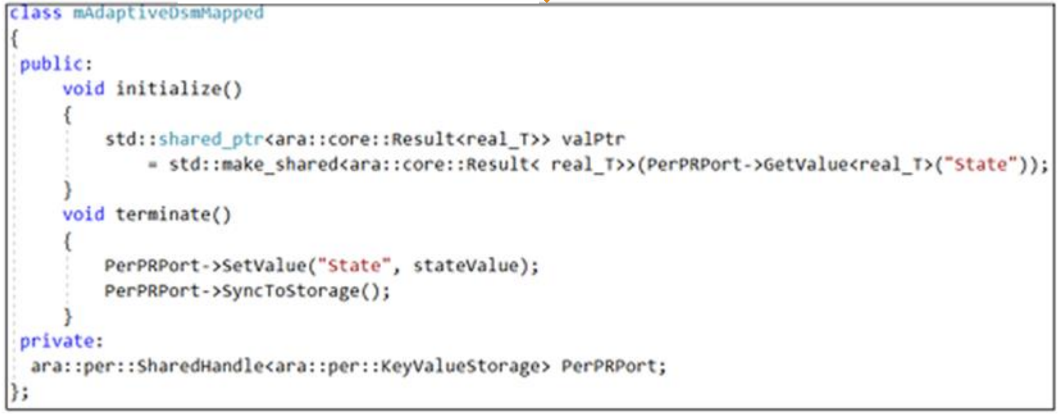
- Configure and Persistency Ports and Interfaces in AUTOSAR dictionary
- Generate C++ code for accessing persistency (via ara::per APIs)
- Import/Export arxml describing persistency ports and Interfaces.



| Source | Mapped To | Path |
|---|---|---|
| Local Data Stores (1) | | |
| State NVRAM | Persistency | mAdaptiveDsmMapped |

Data Stores: State NVRAM

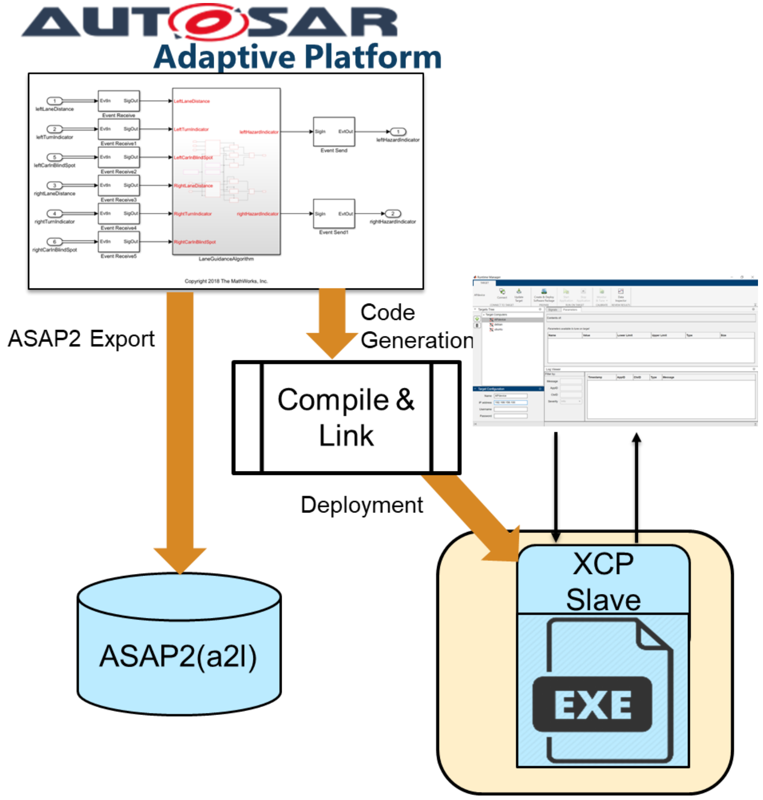
| NAME | VALUE |
|---|---|
| Source | State NVRAM |
| □ Shared | |
| **Design** | |
| Initial Value | 0 |
| Data Type | Inherit: auto |
| Min | [] |
| Max | [] |
| Dimensions | -1 |
| Complexity | auto |
| **Code** | |
| Mapped To | Persistency |
| Path | mAdaptiveDsmMapped |
| Port | PerPRPort |
| DataElement | State |

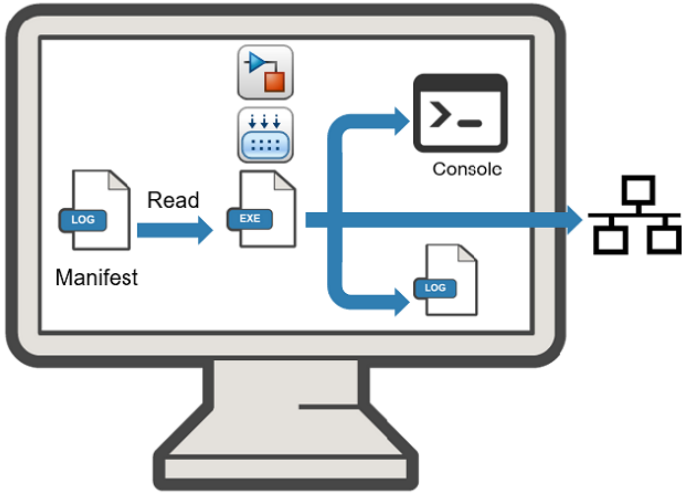


```
class mAdaptiveDsmMapped
{
 public:
     void initialize()
     {
         std::shared_ptr<ara::core::Result<real_T>> valPtr
             = std::make_shared<ara::core::Result< real_T>>(PerPRPort->GetValue<real_T>("State"));
     }
     void terminate()
     {
         PerPRPort->SetValue("State", stateValue);
         PerPRPort->SyncToStorage();
     }
 private:
  ara::per::SharedHandle<ara::per::KeyValueStorage> PerPRPort;
};
```

# AUTOSAR Adaptive Deployment

- Create Linux executables for Run-Time Calibration and Measurement
- Run-time logging (ara::log) for adaptive executables
  - Forward event logging information to a console, file, or network, as defined in the AUTOSAR Diagnostic Log and Trace specification
  - Collate and analyze log data from multiple applications



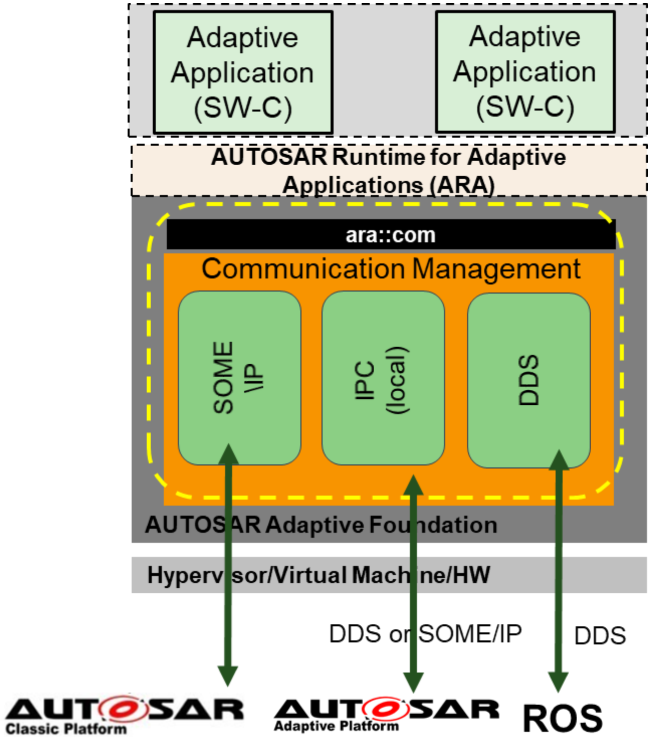


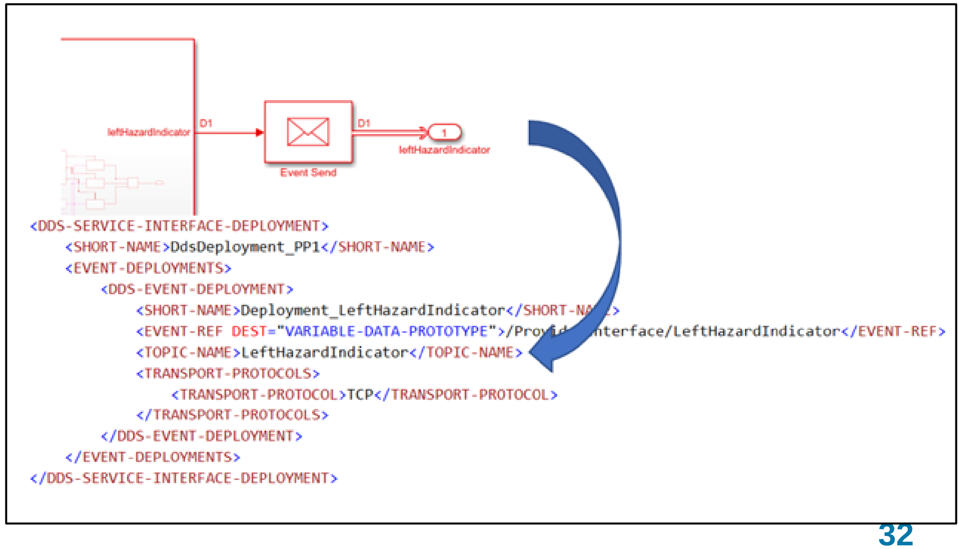
# AUTOSAR Adaptive Deployment

## Supports both DDS or SOME/IP

- Supports DDS binding for ara::com enabling communication between adaptive AUTOSAR applications
  - Generated `ServiceInstanceManifest.arxml` contains DDS deployment artifacts
- Supports SOME/IP Communication between AUTOSAR Classic and Adaptive applications

Learn more about Designing and deploying interoperable AUTOSAR and non-AUTOSAR applications for heterogeneous automated driving  platforms


Adaptive Application (SW-C)
Adaptive Application (SW-C)
AUTOSAR Runtime for Adaptive Applications (ARA)
ara::com
Communication Management
SOME /IP
IPC (local)
DDS
AUTOSAR Adaptive Foundation
Hypervisor/Virtual Machine/HW
DDS or SOME/IP
DDS
AUTOSAR Classic Platform
AUTOSAR Adaptive Platform
ROS



```
<DDS-SERVICE-INTERFACE-DEPLOYMENT>
    <SHORT-NAME>DdsDeployment_PP1</SHORT-NAME>
    <EVENT-DEPLOYMENTS>
        <DDS-EVENT-DEPLOYMENT>
            <SHORT-NAME>Deployment_LeftHazardIndicator</SHORT-NAME>
            <EVENT-REF DEST="VARIABLE-DATA-PROTOTYPE">/ProvidedInterface/LeftHazardIndicator</EVENT-REF>
            <TOPIC-NAME>LeftHazardIndicator</TOPIC-NAME>
            <TRANSPORT-PROTOCOLS>
                <TRANSPORT-PROTOCOL>TCP</TRANSPORT-PROTOCOL>
            </TRANSPORT-PROTOCOLS>
        </DDS-EVENT-DEPLOYMENT>
    </EVENT-DEPLOYMENTS>
</DDS-SERVICE-INTERFACE-DEPLOYMENT>
```

# Data Distribution Services (DDS)

Data Distribution Services (DDS) uses SOA methodology, and directly addresses publish and subscribe communications for real-time and embedded systems.

DDS addresses the needs of applications that require real-time data exchange in industries like aerospace and defense, automotive, and robotics.

# Data Distribution Services (DDS)

Data Distribution Services (DDS) uses SOA methodology, and directly addresses publish and subscribe communications for real-time and embedded systems.

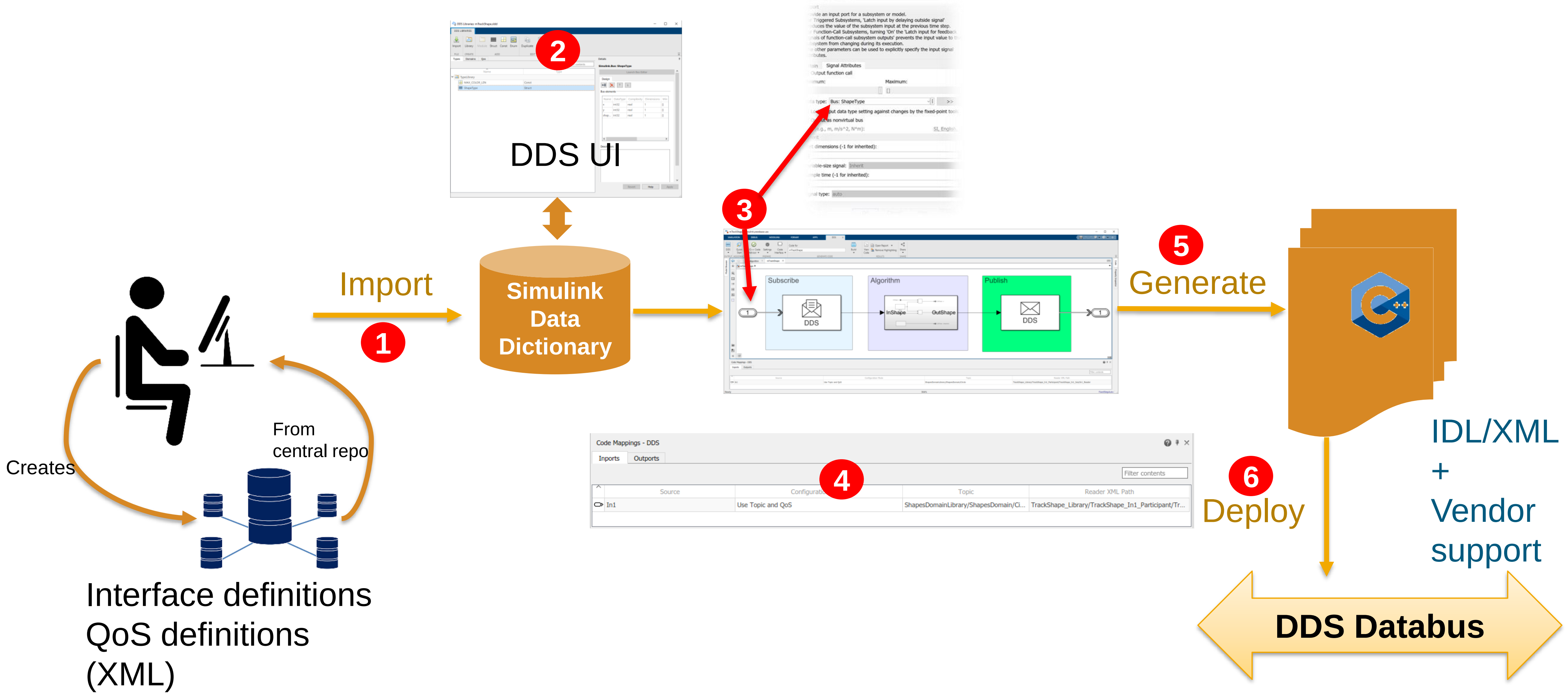
DDS addresses the needs of applications that require real-time data exchange in industries like aerospace and defense, automotive, and robotics.

R2021a

DDS Blockset

Design and simulate DDS applications

# User Workflow with UI Steps



ide an input port for a subsystem or model.
Triggered Subsystems, 'Latch input by delaying outside signal'
duces the value of the subsystem input at the previous time step.
Function-Call Subsystems, turning 'On' the 'Latch input for feedback
nals of function-call subsystem outputs' prevents the input value to th
system from changing during its execution.
other parameters can be used to explicitly specify the input signal
ibutes.

Signal Attributes

Output function call

imum: Maximum: []

a type: Bus: ShapeType

put data type setting against changes by the fixed-point too

as nonvirtual bus

(e.g., m, m/s^2, N*m): SI, English

dimensions (-1 for inherited):

able-size signal: Inherit

ple time (-1 for inherited):

nal type: auto

Code Mappings - DDS

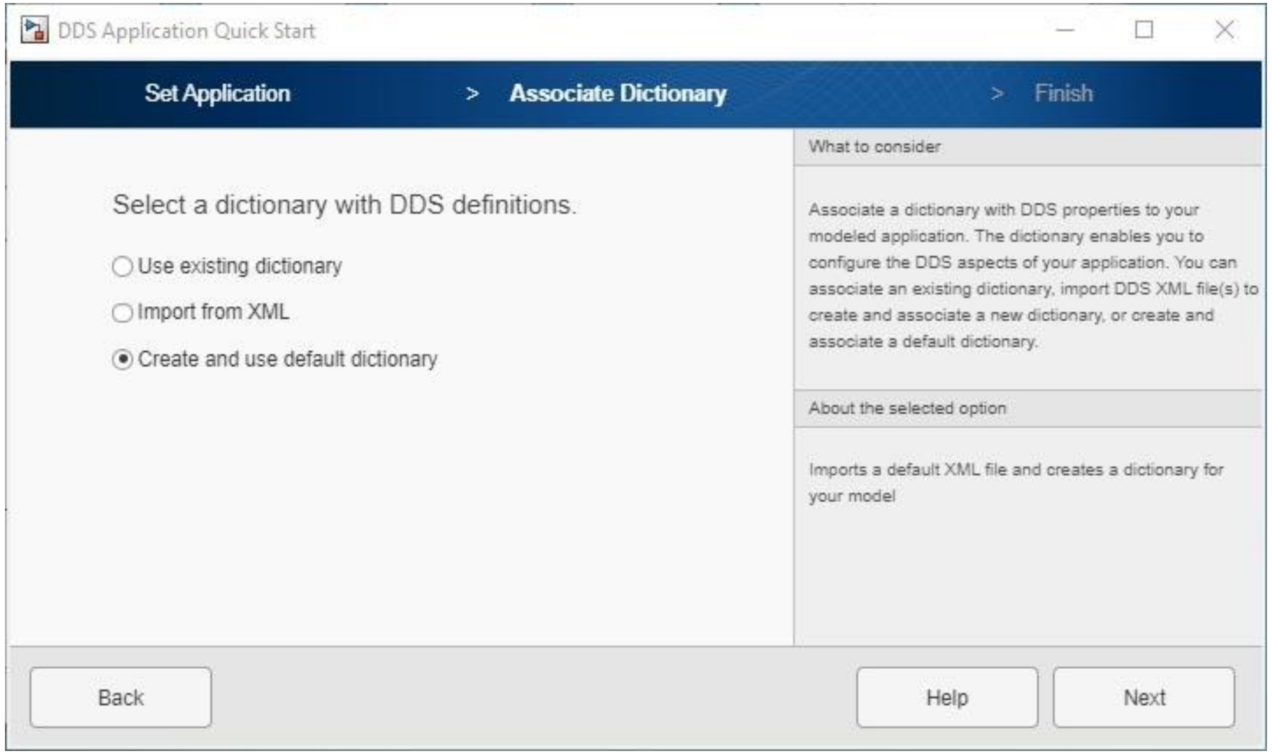
Inports | Outports

Filter contents

| Source | Configuration | Topic | Reader XML Path |
|---|---|---|---|
| In1 | Use Topic and QoS | ShapesDomainLibrary/ShapesDomain/Ci... | TrackShape_Library/TrackShape_In1_Participant/Tr... |

# DDS Blockset in action

- Import DDS definitions from XML or create new Definitions



Select a dictionary with DDS definitions.

○ Use existing dictionary

○ Import from XML

● Create and use default dictionary

What to consider

Associate a dictionary with DDS properties to your modeled application. The dictionary enables you to configure the DDS aspects of your application. You can associate an existing dictionary, import DDS XML file(s) to create and associate a new dictionary, or create and associate a default dictionary.

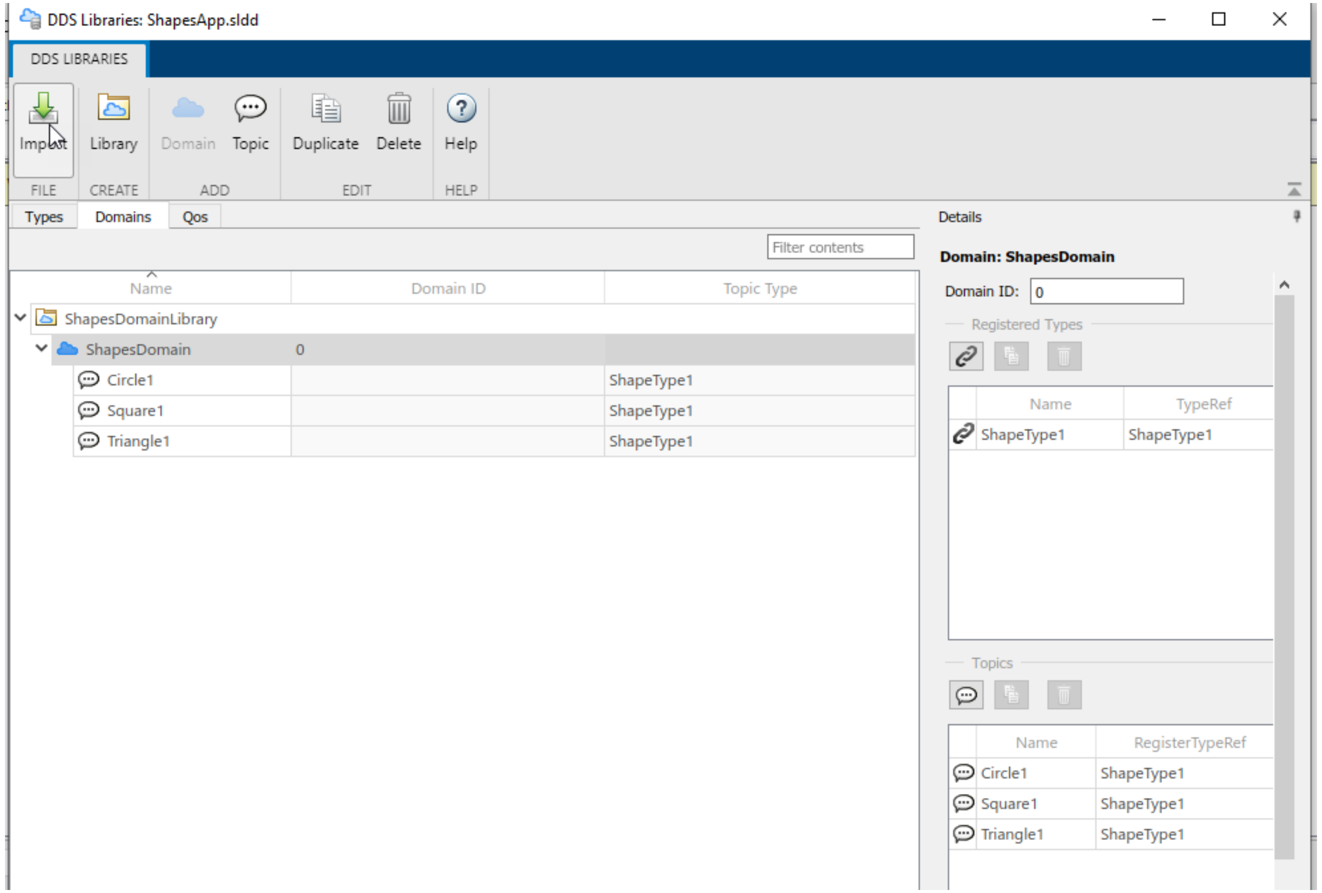
About the selected option

Imports a default XML file and creates a dictionary for your model

# DDS Blockset in action

- Import DDS definitions from XML or create new Definitions

- Define/Modify DDS definitions in DDS Dictionary
  - Topic Types
  - Domains
  - QoS



| Name | Domain ID | Topic Type |
|---|---|---|
| ShapesDomainLibrary | | |
| ShapesDomain | 0 | |
| Circle1 | | ShapeType1 |
| Square1 | | ShapeType1 |
| Triangle1 | | ShapeType1 |

**Domain: ShapesDomain**

Domain ID: 0

Registered Types

| Name | TypeRef |
|---|---|
| ShapeType1 | ShapeType1 |

Topics

| Name | RegisterTypeRef |
|---|---|
| Circle1 | ShapeType1 |
| Square1 | ShapeType1 |
| Triangle1 | ShapeType1 |

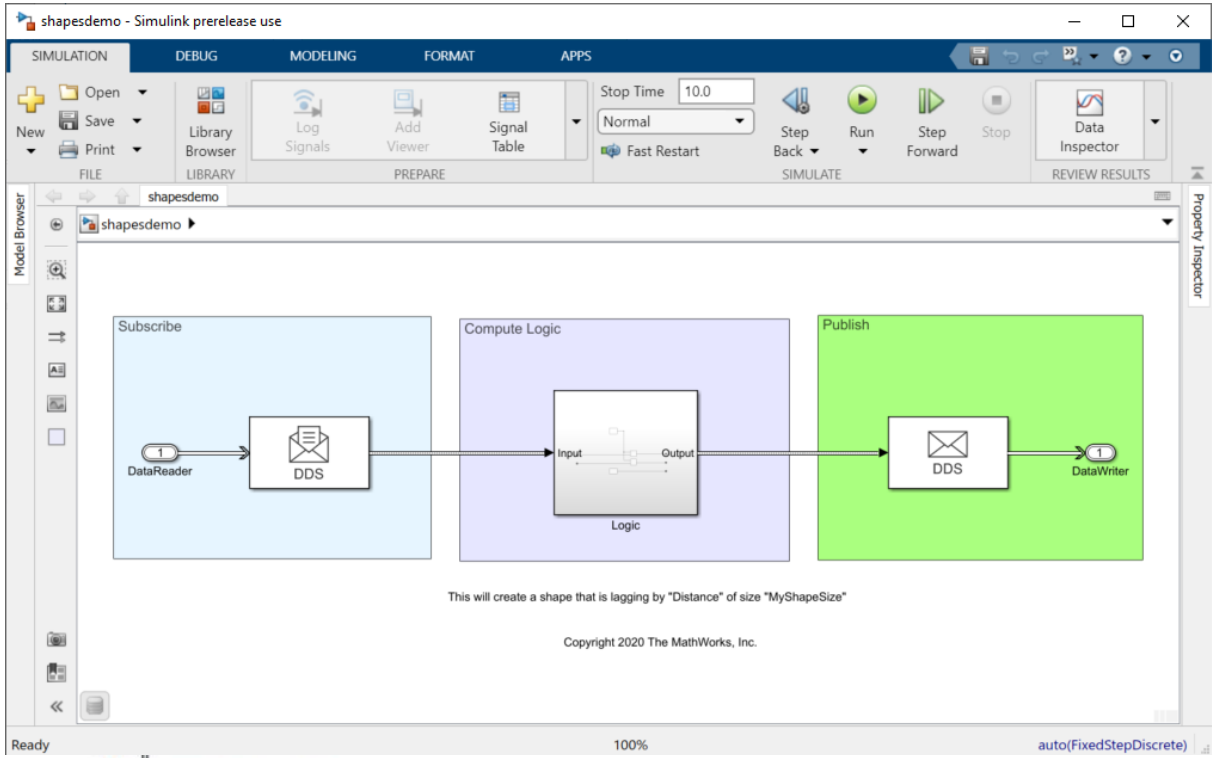
# DDS Blockset in action

- Import DDS definitions from XML or create new Definitions

- Define/Modify DDS definitions in DDS Dictionary

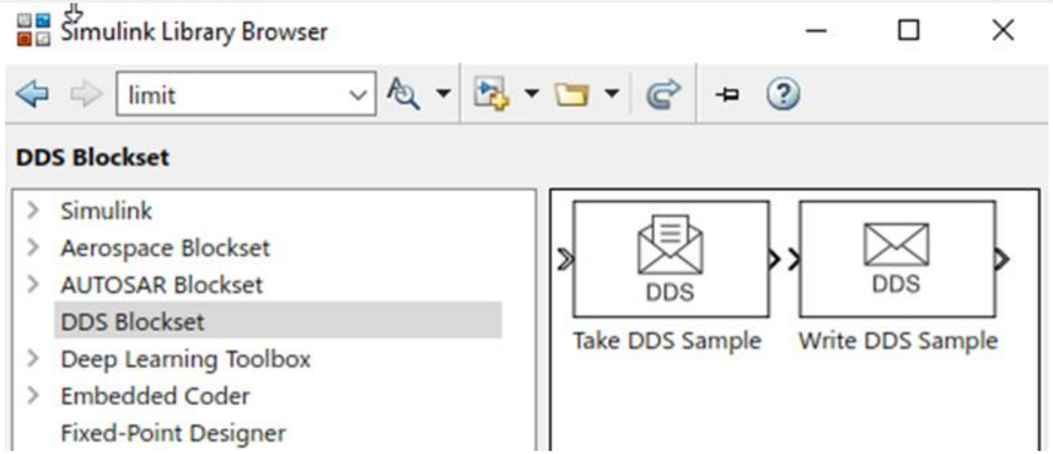
- Model applications

Use DDS Blocks to model a Publisher or Subscriber



This will create a shape that is lagging by "Distance" of size "MyShapeSize"

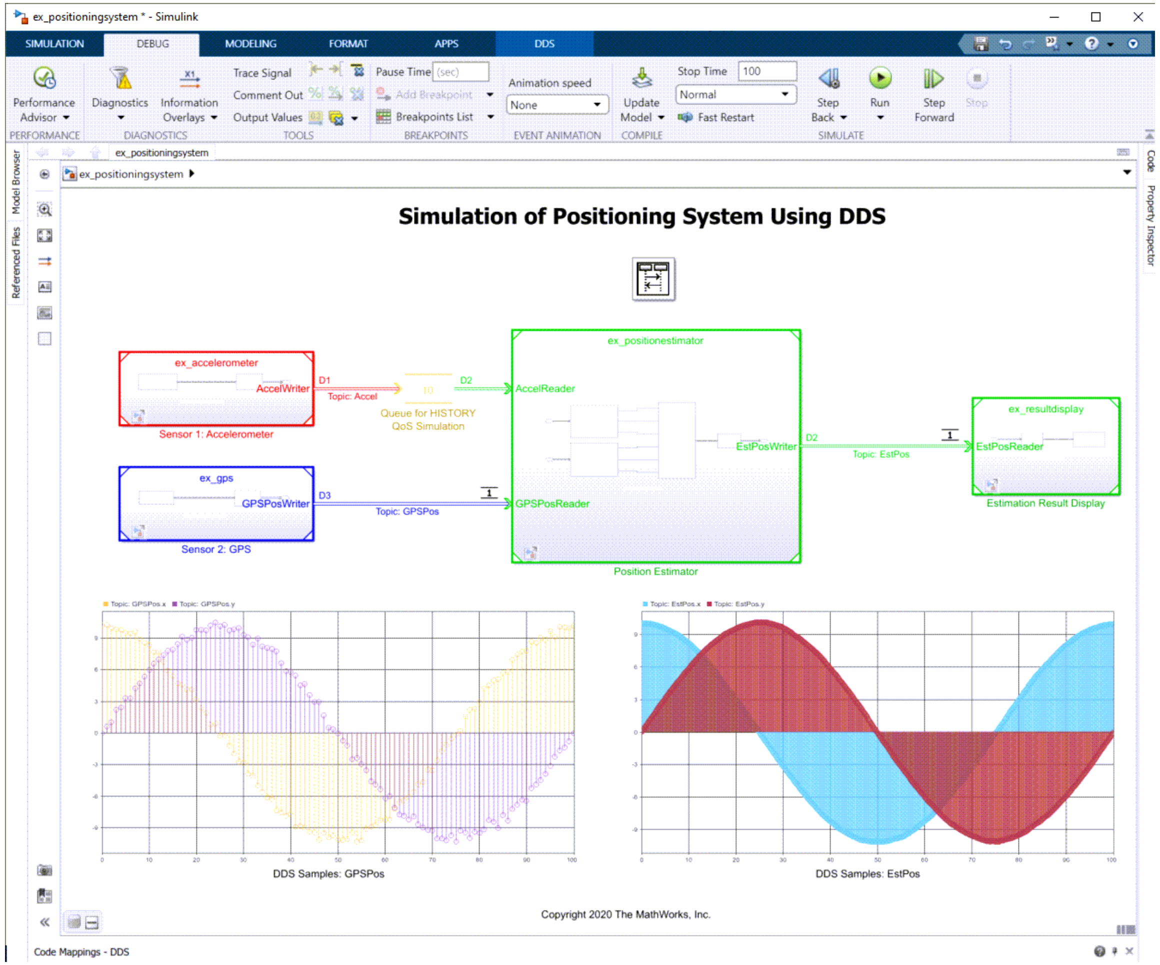
Copyright 2020 The MathWorks, Inc.



# DDS Blockset in action

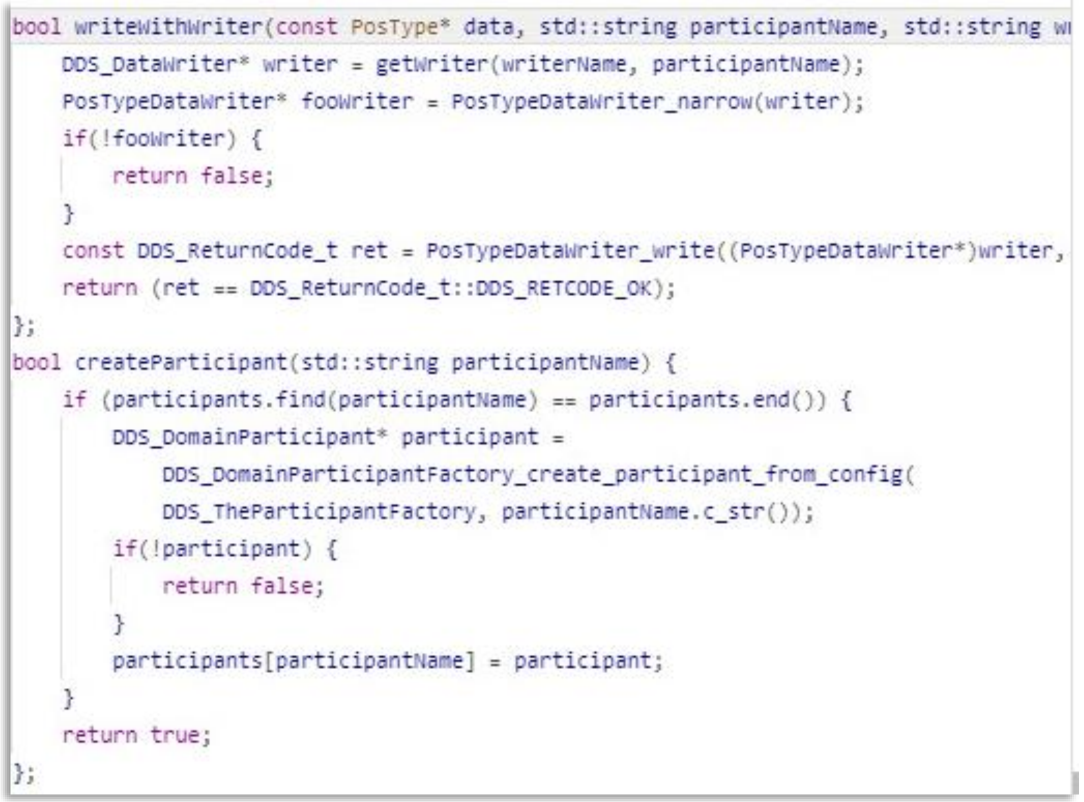
- Import DDS definitions from XML or create new Definitions
- Define/Modify DDS definitions in DDS Dictionary
- Model applications
- Simulate DDS models including QoS

Use Simulink to model and simulation Quality of Services (QoS) policies including **history** to verify the runtime behavior.



# DDS Blockset in action

- Import DDS definitions from XML or create new Definitions
- Define/Modify DDS definitions in DDS Dictionary
- Model applications
- Simulate DDS models including QoS
- Generate DDS executables and deploy on a DDS network



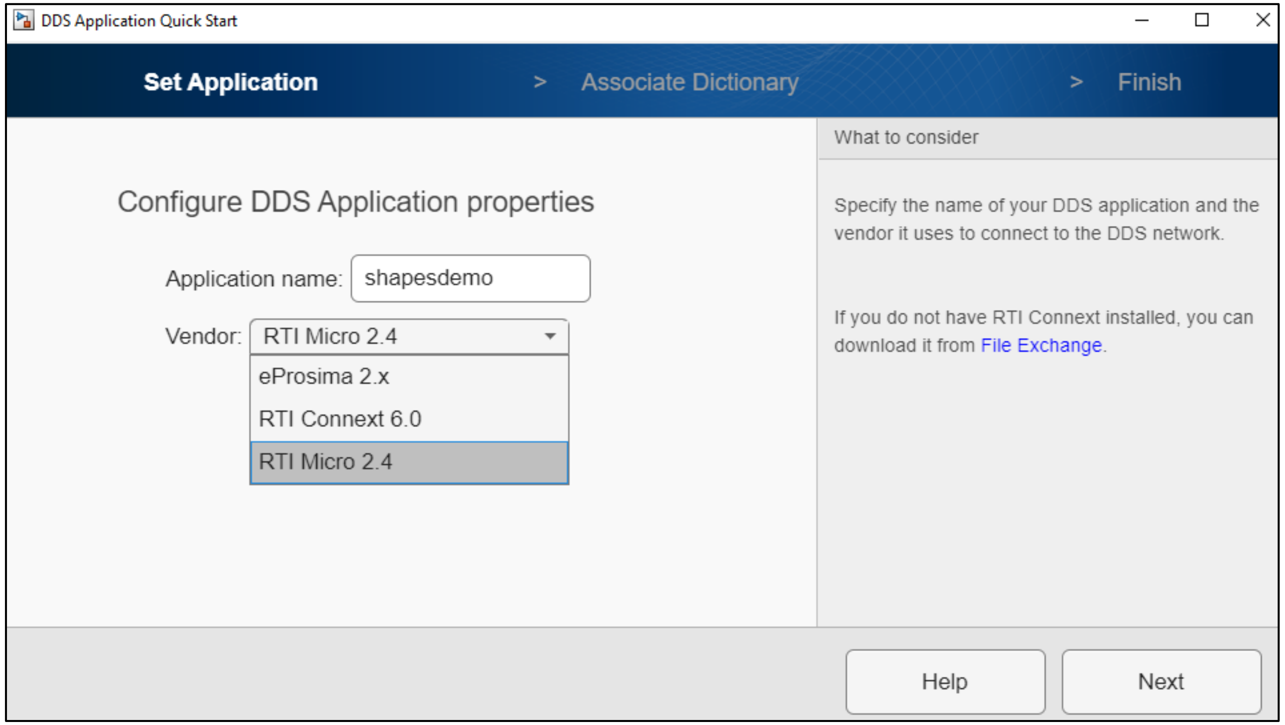
```
bool writeWithWriter(const PosType* data, std::string participantName, std::string wi
    DDS_DataWriter* writer = getWriter(writerName, participantName);
    PosTypeDataWriter* fooWriter = PosTypeDataWriter_narrow(writer);
    if(!fooWriter) {
        return false;
    }
    const DDS_ReturnCode_t ret = PosTypeDataWriter_write((PosTypeDataWriter*)writer,
    return (ret == DDS_ReturnCode_t::DDS_RETCODE_OK);
};
bool createParticipant(std::string participantName) {
    if (participants.find(participantName) == participants.end()) {
        DDS_DomainParticipant* participant =
            DDS_DomainParticipantFactory_create_participant_from_config(
            DDS_TheParticipantFactory, participantName.c_str());
        if(!participant) {
            return false;
        }
        participants[participantName] = participant;
    }
    return true;
};
```

With Embedded coder, generate

- C++ production code with DDS APIs
- XML or IDL files from Simulink models to deploy

# DDS Blockset in action

- Import DDS definitions from XML or create new Definitions
- Define/Modify DDS definitions in DDS Dictionary
- Model applications
- Simulate DDS models including QoS
- Generate DDS executables and deploy on a DDS network



Configure DDS Application properties

Application name: shapesdemo

Vendor: RTI Micro 2.4

eProsima 2.x

RTI Connext 6.0

RTI Micro 2.4

What to consider

Specify the name of your DDS application and the vendor it uses to connect to the DDS network.

If you do not have RTI Connext installed, you can download it from File Exchange.

Full integration with third-party DDS stacks including RTI Connext, RTI Micro and eProsima Fast DDS

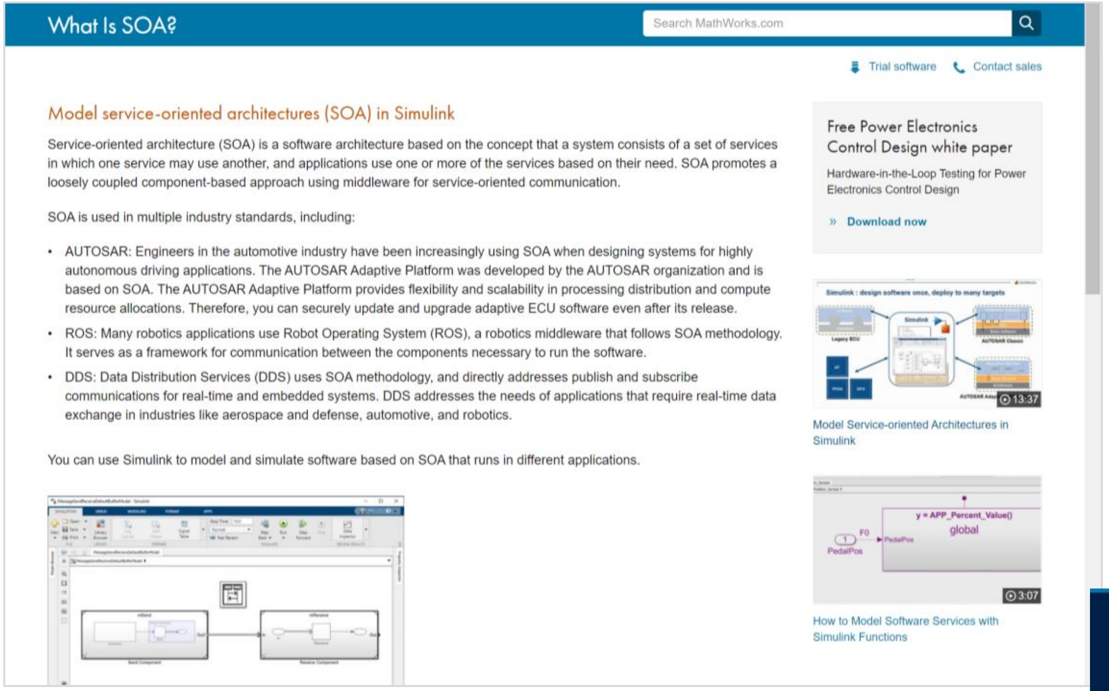
# Agenda

- SW-defined vehicles and new architectures (SOA)
- MathWorks solutions for SOA
- Conclusions and key takeaways

# Conclusions and Key takeaways

- **Automotive E/E and SW architecture are evolving**, pushed by need for advanced, complex functions
- New, **service-oriented architectures** are required to **master complexity** and enable **frequent updates**
- You can **design, simulate and generate** code to deploy service-oriented applications in **Simulink**
- You can **reuse your existing expertise and models** to mitigate the risk of migration to SOA applications

# To learn more, visit the SOA, AUTOSAR & DDS Blockset pages



## Model service-oriented architectures (SOA) in Simulink

Service-oriented architecture (SOA) is a software architecture based on the concept that a system consists of a set of services in which one service may use another, and applications use one or more of the services based on their need. SOA promotes a loosely coupled component-based approach using middleware for service-oriented communication.

SOA is used in multiple industry standards, including:

- AUTOSAR: Engineers in the automotive industry have been increasingly using SOA when designing systems for highly autonomous driving applications. The AUTOSAR Adaptive Platform was developed by the AUTOSAR organization and is based on SOA. The AUTOSAR Adaptive Platform provides flexibility and scalability in processing distribution and compute resource allocations. Therefore, you can securely update and upgrade adaptive ECU software even after its release.
- ROS: Many robotics applications use Robot Operating System (ROS), a robotics middleware that follows SOA methodology. It serves as a framework for communication between the components necessary to run the software.
- DDS: Data Distribution Services (DDS) uses SOA methodology, and directly addresses publish and subscribe communications for real-time and embedded systems. DDS addresses the needs of applications that require real-time data exchange in industries like aerospace and defense, automotive, and robotics.

You can use Simulink to model and simulate software based on SOA that runs in different applications.

Free Power Electronics Control Design white paper

Hardware-in-the-Loop Testing for Power Electronics Control Design

» Download now

Model Service-oriented Architectures in Simulink

How to Model Software Services with Simulink Functions

*www.mathworks.com/discovery/soa.html*

**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).

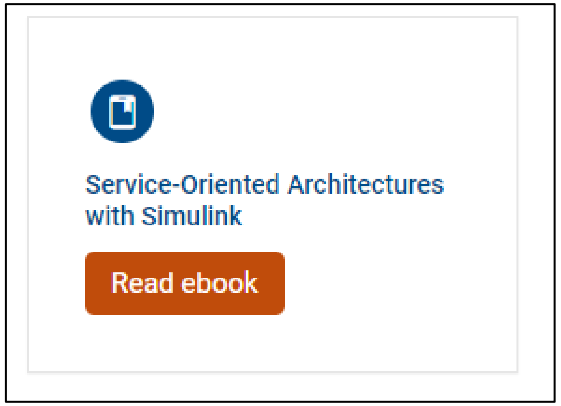
*www.mathworks.com/products/autosar.html*

**DDS Blockset**

Design and simulate DDS applications

DDS Blockset provides apps and blocks for modeling and simulating software applications that publish or subscribe to Data Distribution Service (DDS) middleware. The blockset includes a DDS dictionary that lets you manage, create, and edit your DDS definitions in Simulink®. You can import DDS specifications as XML files to create a skeleton Simulink model as a starting point for developing algorithms for DDS applications.

DDS Blockset provides blocks for publishing and subscribing samples to DDS, including their corresponding Quality of Service (QoS.). It fully



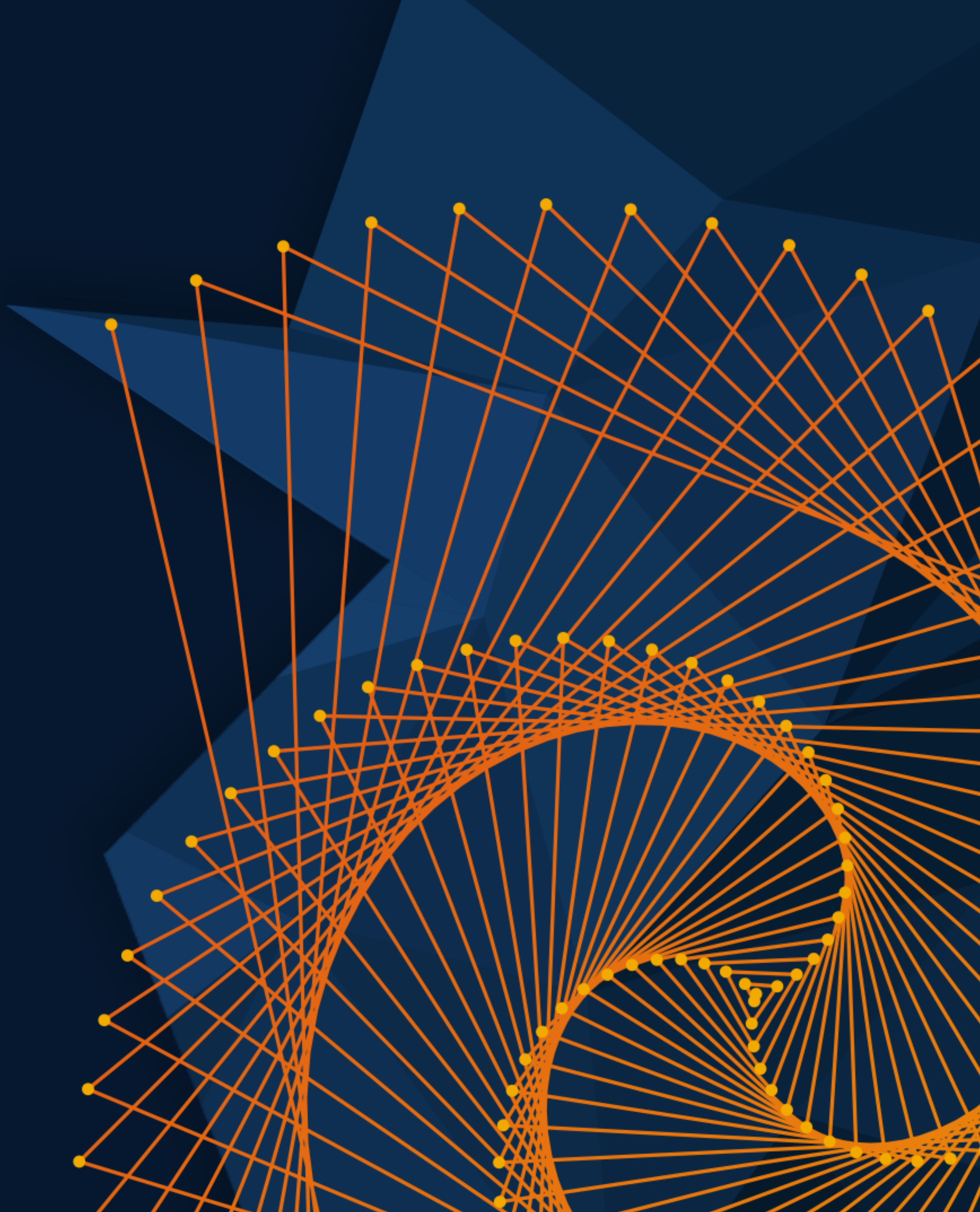
Service-Oriented Architectures with Simulink

Read ebook

*www.mathworks.com/products/dds.html*

MATLAB EXPO

# Thank you



MathWorks