

# MATLAB EXPO 2017

## KOREA

4월 27일, 서울

등록 하기 [matlabexpo.co.kr](http://matlabexpo.co.kr)

# From Simulink to AUTOSAR Code

김종헌 부장

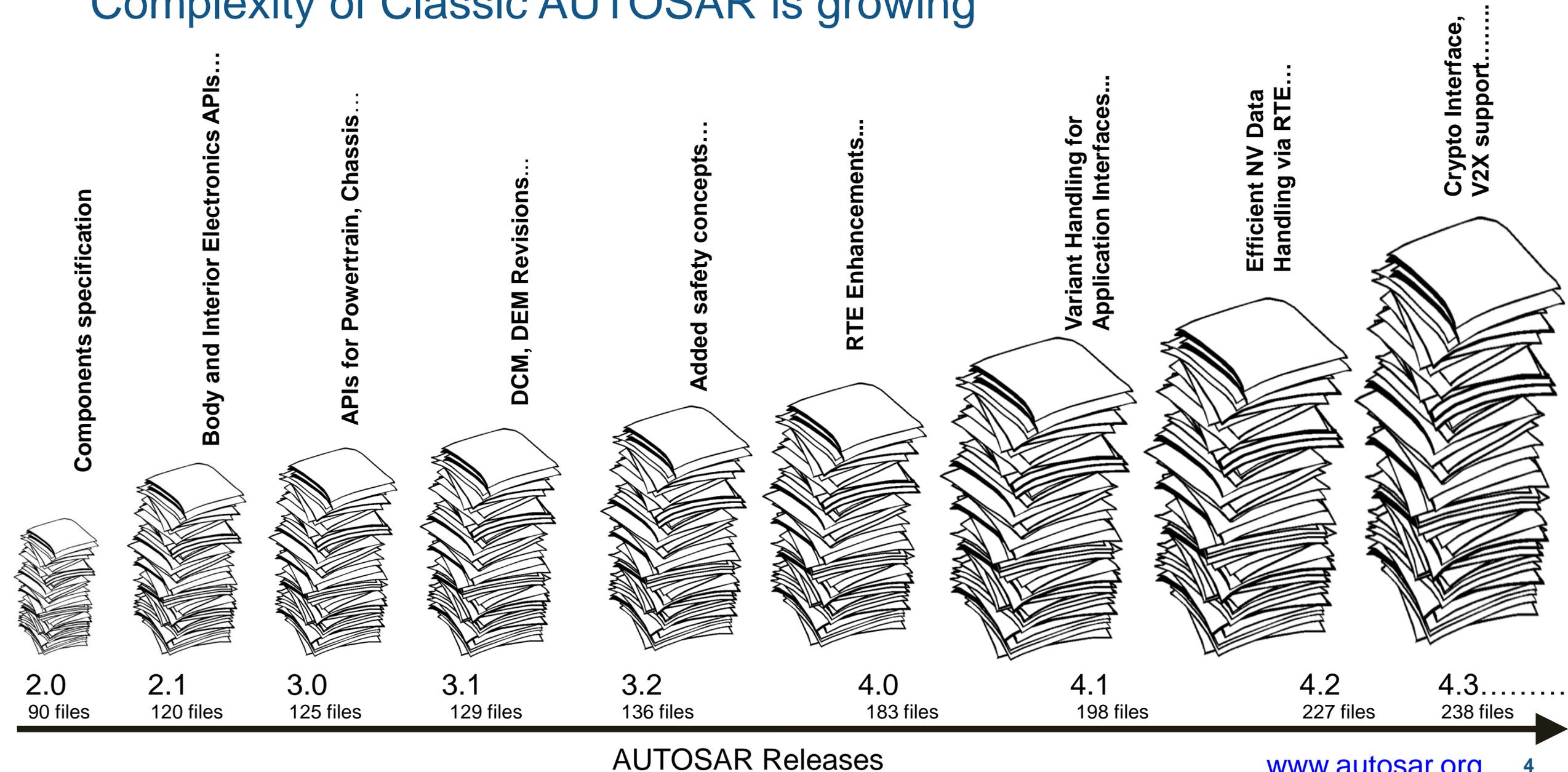
Senior Application Engineer

MathWorks Korea

# Agenda

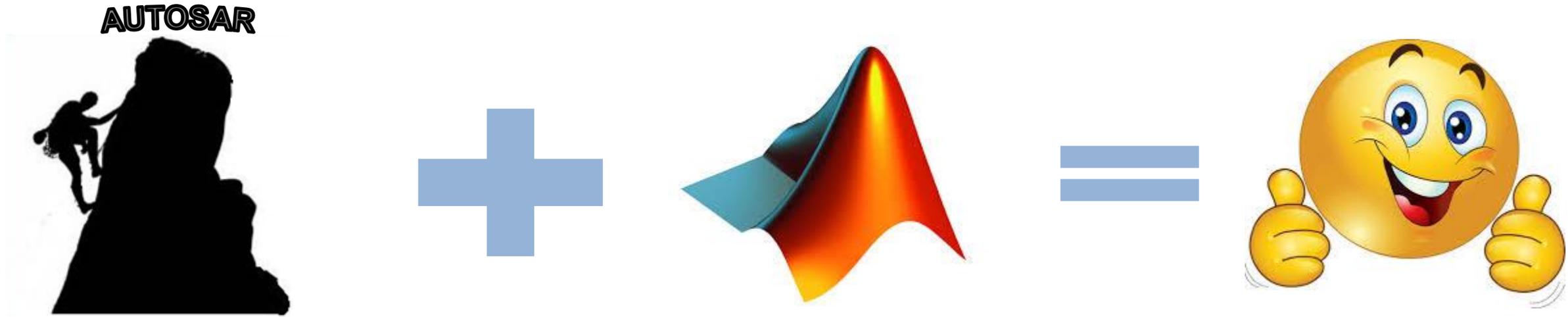
- Introduction to AUTOSAR Standards
  - Simulink approach to AUTOSAR
  
- AUTOSAR design workflows
  - Bottom Up
  - Top Down
  - Round trip
  - Overview:
    - Modeling AUTOSAR Components / Attributes
    - Components, Runnables and Events
    - Modeling styles

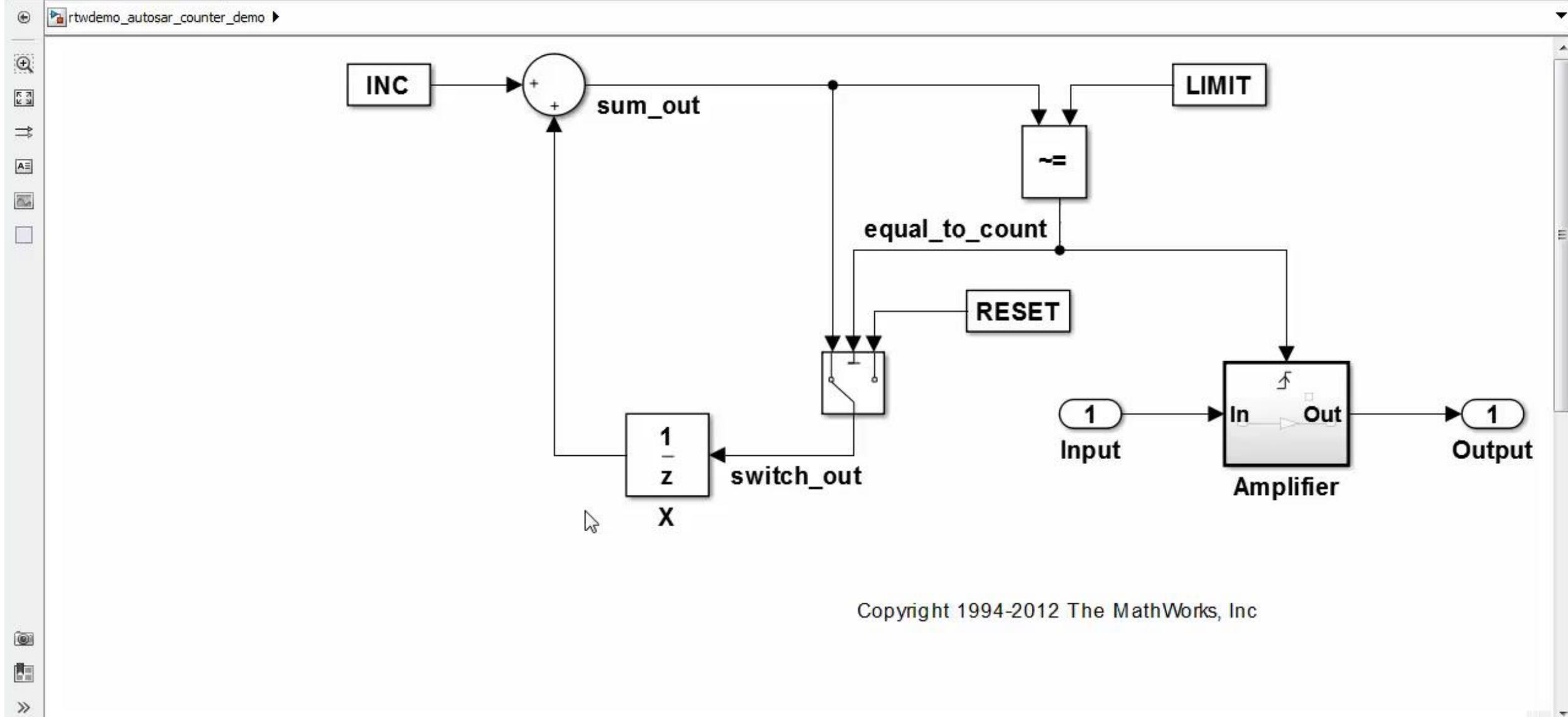
# Complexity of Classic AUTOSAR is growing



AUTOSAR Releases

# AUTOSAR Adoption

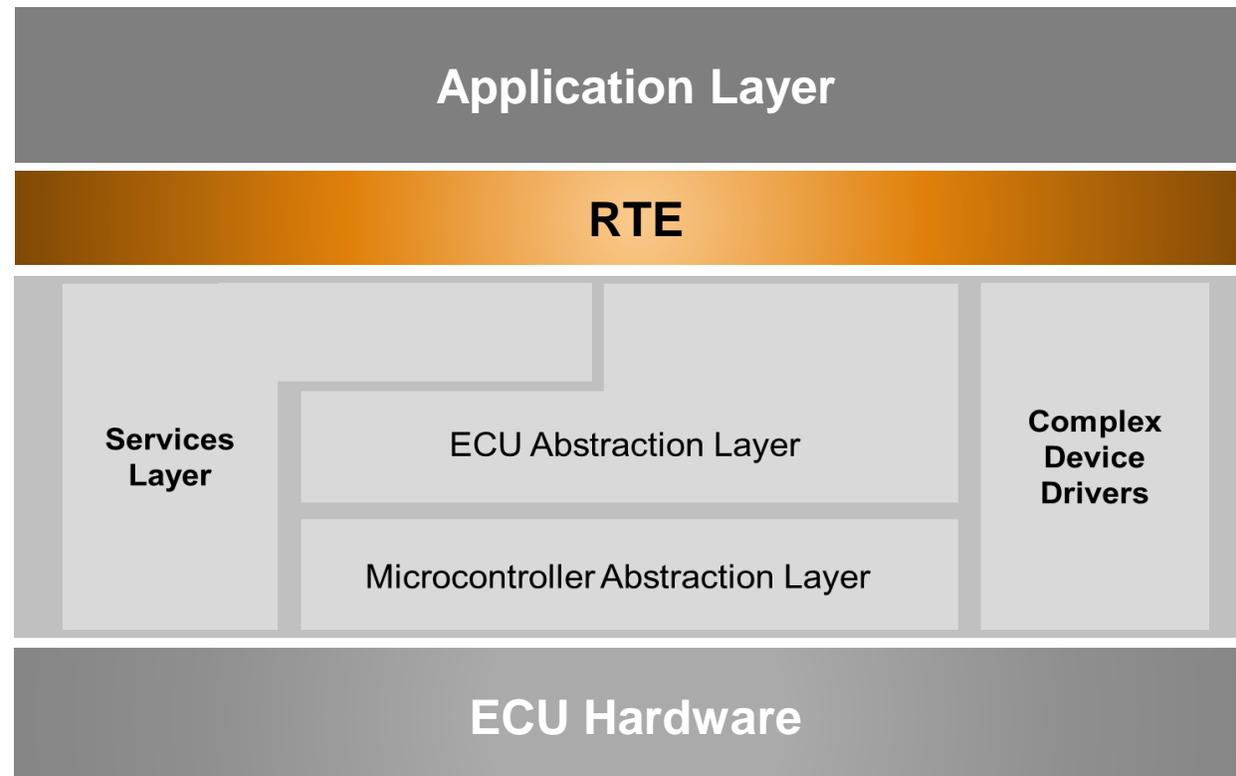




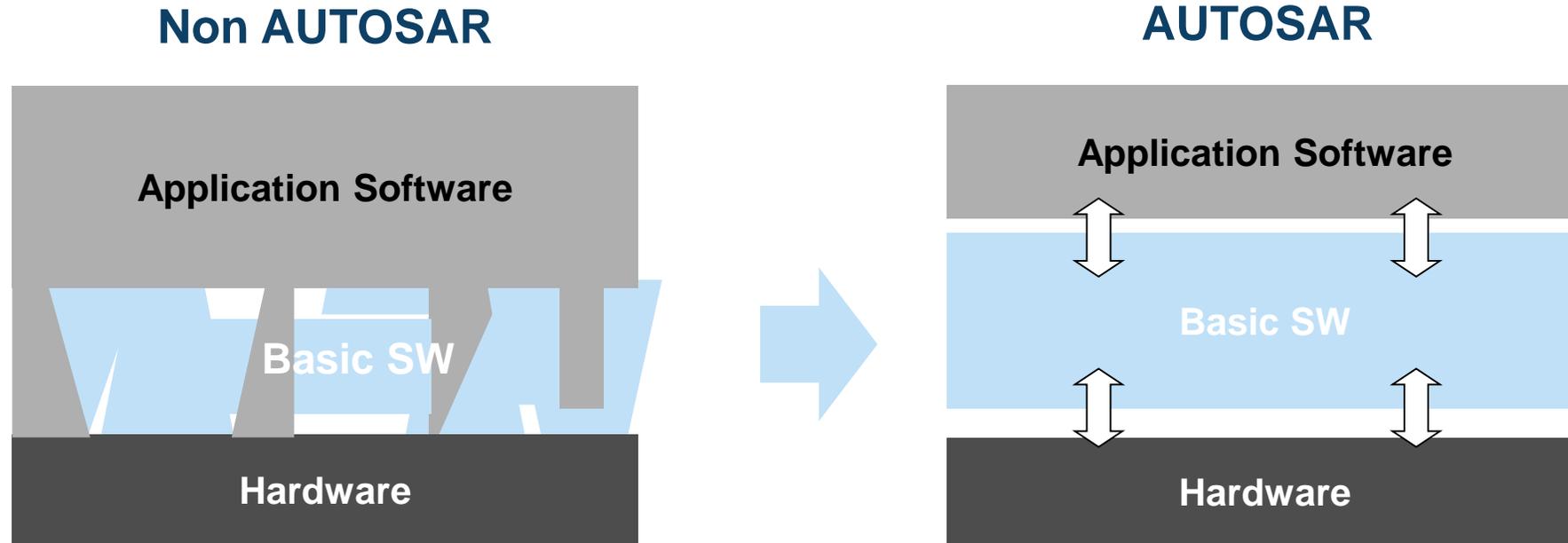
Copyright 1994-2012 The MathWorks, Inc

# What is AUTOSAR?

AUTOSAR® (**AUT**omotive **O**pen **S**ystem **AR**chitecture) is an open and standardized automotive software architecture



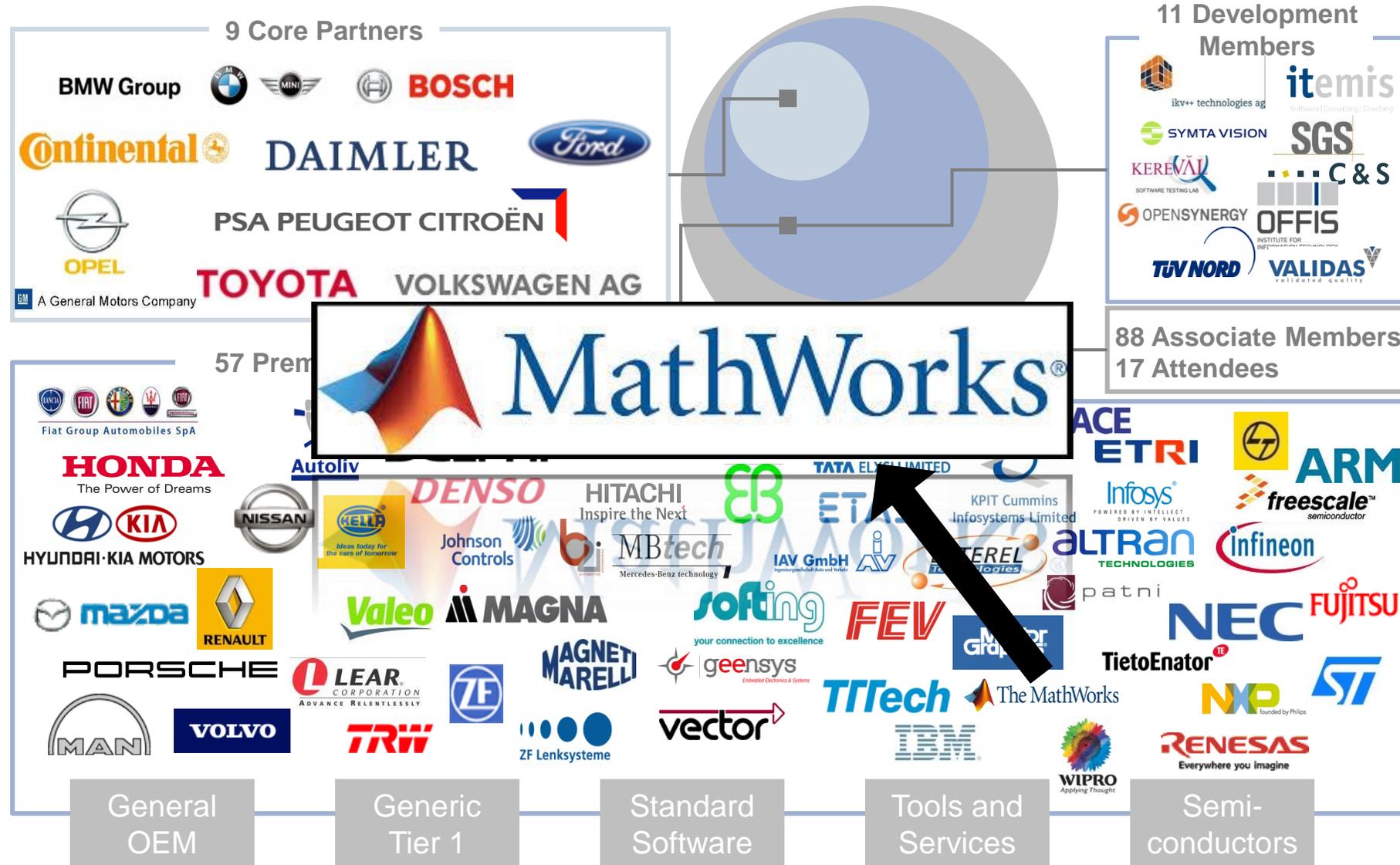
# AUTOSAR Vision



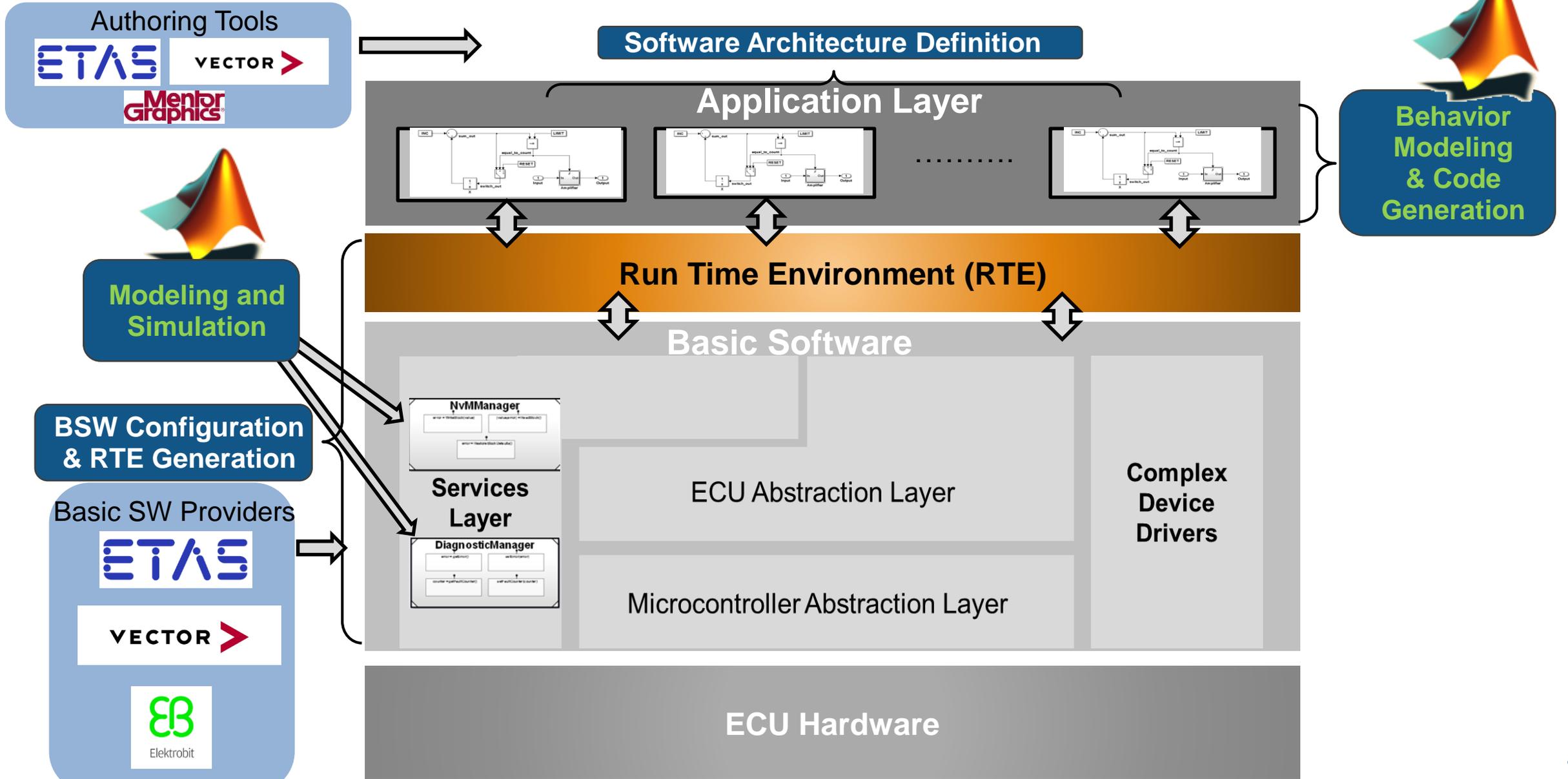
AUTOSAR Slogan-

**“Cooperate on Standards – compete on implementation”**

# AUTOSAR Members



# AUTOSAR Support from Embedded Coder and Simulink

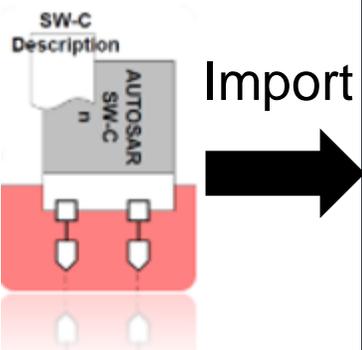


# Simulink Approach to AUTOSAR

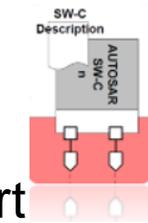
Available via web download

Simulink and Embedded Coder  
+ **AUTOSAR Support package for Embedded Coder**

No separate AUTOSAR Blockset needed



Export



C Code and ARXML

```

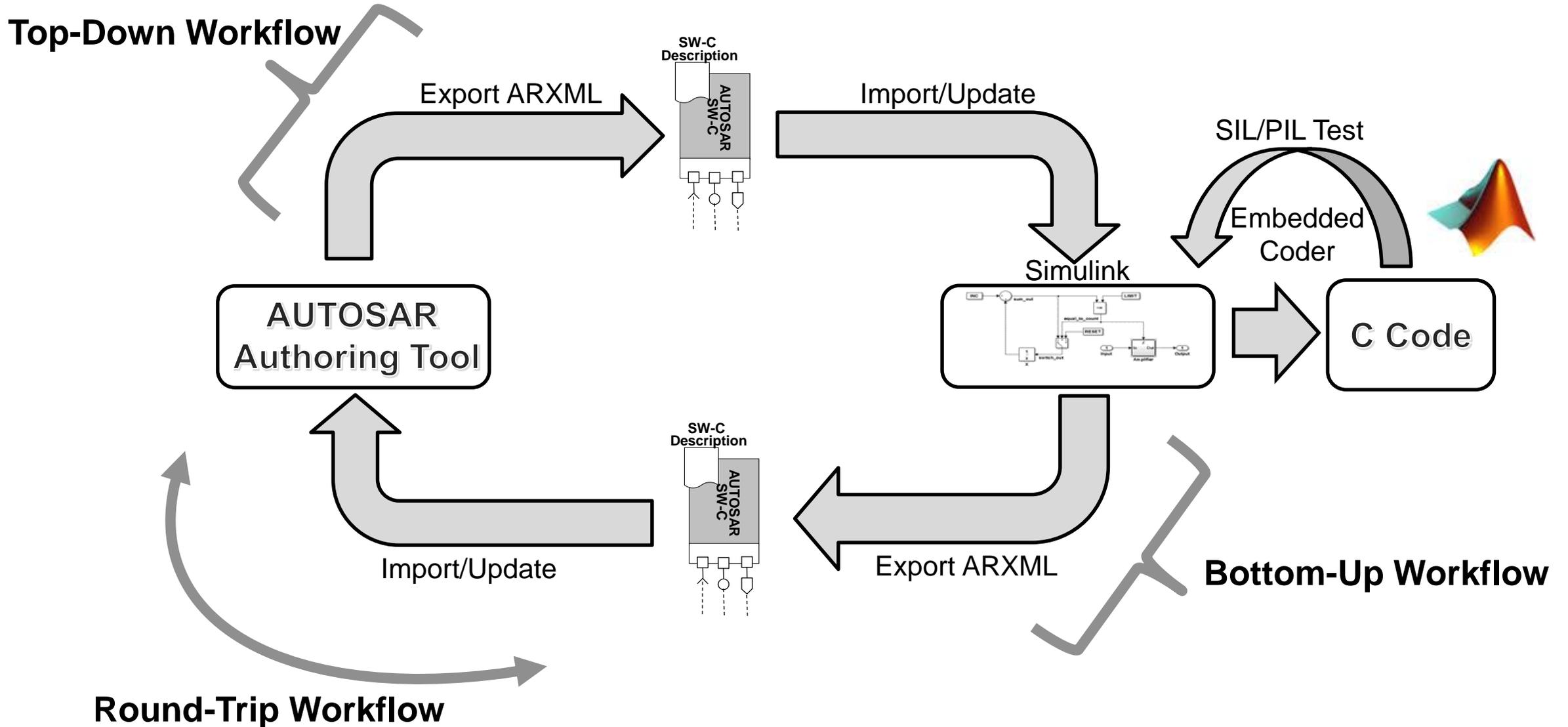
/* Output function for RootInputFunctionCallGenerator: *Root/RootFnCall_InsertedFor_Runnable1_at_output_1
void Runnable1(void)
{
  inst_T Delay_n;

  /* RootInputFunctionCallGenerator: *Root/RootFnCall_InsertedFor_Runnable1_at_output_1 incorporates:
   * Subsystem: *Root/Runnable1_subsystem
   */
  /* UnitDelay: *c1b/Delay */
  Delay_n = z1Dnmxk.Delay_DSTATE_n;

  /* Outputs for Enabled Subsystem: *c1b/Subsystem incorporates:
   * EnablePort: *c1b/Enable */
  /* RelationalOperator: *c1b/Data_Valid incorporates:
   * Input: *SRoot/SRoot_DE1_ErrorStatus */
  /*
  if (Rte_Itatus_Runnable1_SPort_DE1) == 0) {
    ...
  }
  }
  
```

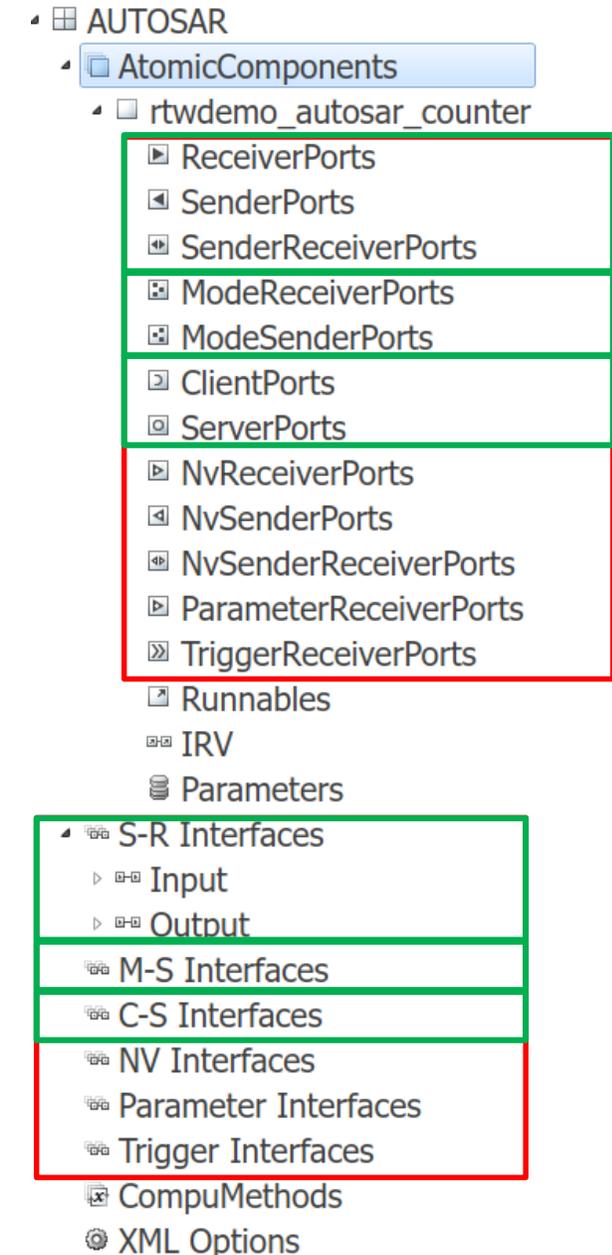
Code-generation through Mapping

# Supported AUTOSAR Design Workflows



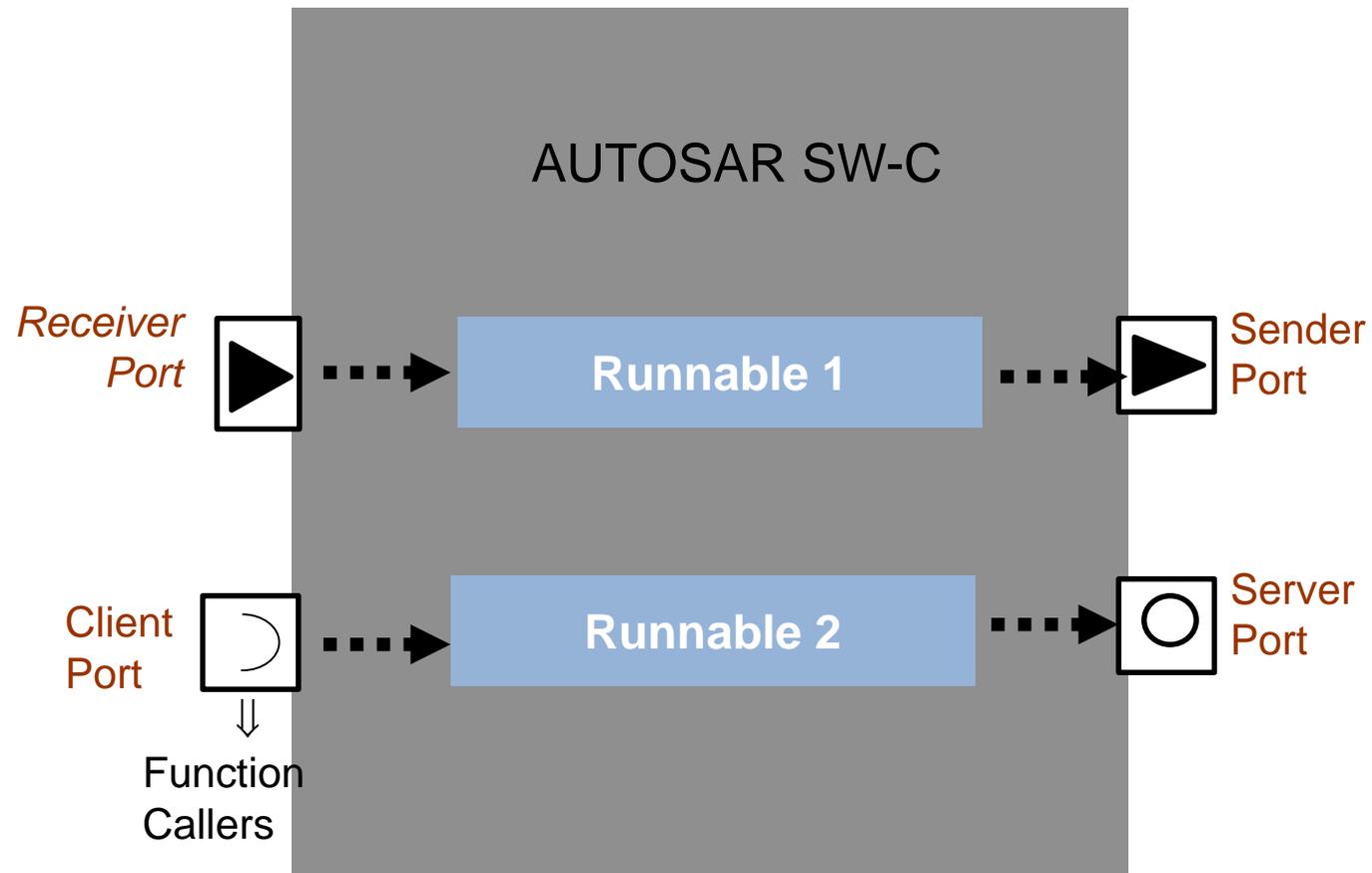
# Modeling AUTOSAR Communication

- Ports in a AUTOSAR software component allow for communication
- Categories of ports based on direction
  - Require port
  - Provide port
- Each port can have either of the following Interfaces



# Runnable Entities

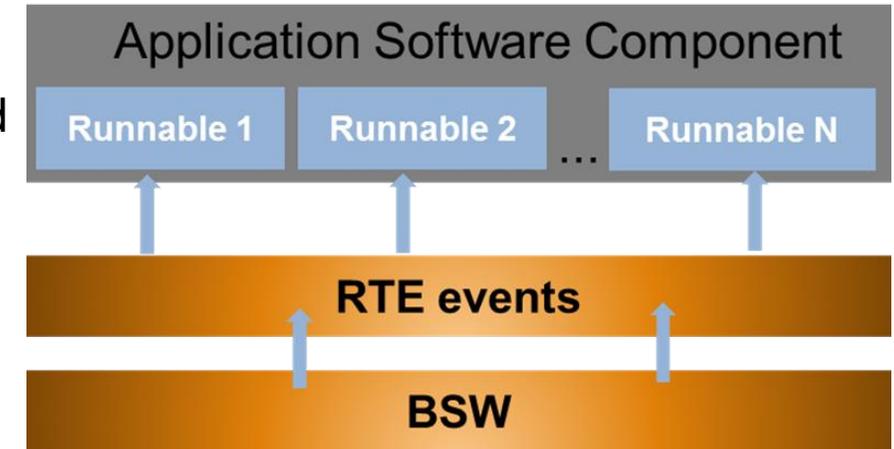
- Each AUTOSAR SW-C is composed by one or more runnables/runnable entities



# Supported Events for a Runnable

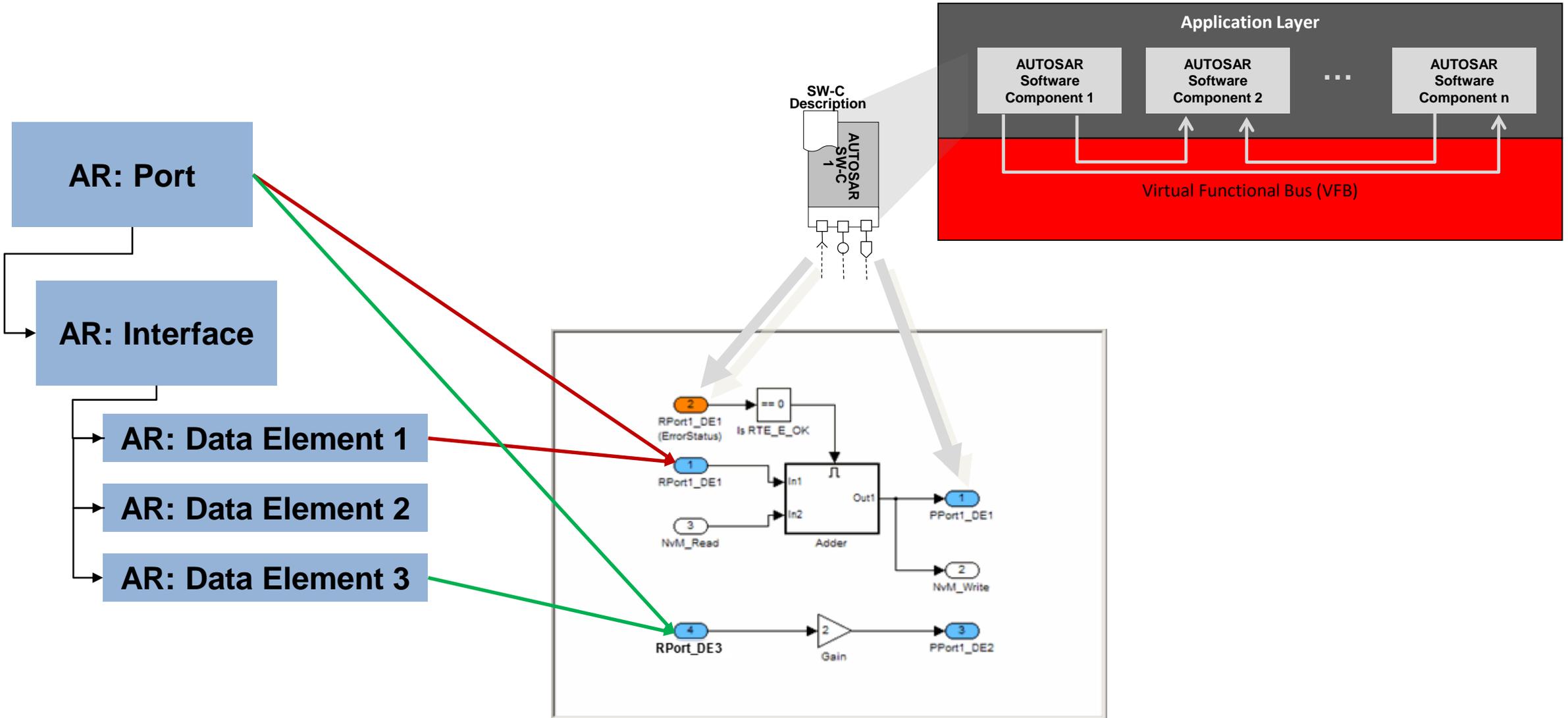
Each Runnable should have at least one event attached

- TimingEvent - Periodically scheduled Runnables
- DataReceivedEvent - Trigger the runnable when data is received
- ModeSwitchEvent - Triggered onEntry, onExit, or onTransition
- OperationInvokedEvent - Client-server type event
- InitEvent - designate an AUTOSAR runnable as an initialization runnable, and then map an initialization function to the runnable.
- DataReceiveErrorEvent - when the communication layer reports an error in data reception by the receiver component
- ExternalTriggerOccuredEvent – used to activate a runnable in an SWC as result of an explicit trigger by a runnable entity of some other SWC



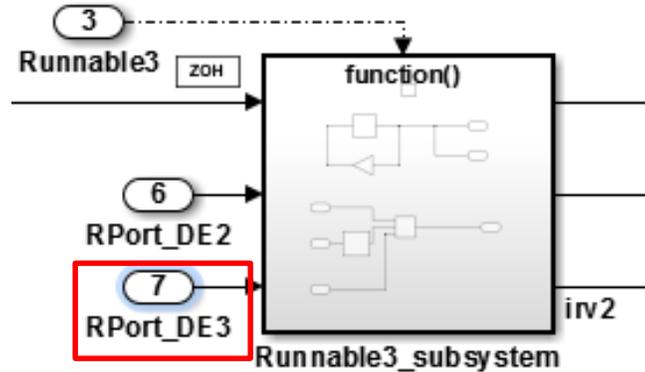
Events	
<input type="button" value="Add Event"/>	<input type="button" value="Delete Event"/>
Event Type	Event Name
TimingEvent	Event_Step
TimingEvent	
DataReceivedEvent	
ModeSwitchEvent	
OperationInvokedEvent	
InitEvent	
DataReceiveErrorEvent	
ExternalTriggerOccuredEvent	

# Mapping Simulink to AUTOSAR



# Example Mapping to a Receiver Port

## Add a New Simulink Port



## Add a New Data Element to the Interface

Name	InvalidationPolicy	SwCalibrationAccess	DisplayFormat
DE1	None	ReadOnly	
DE2	None	ReadOnly	
DE3	None	ReadOnly	

## Validate AUTOSAR to Simulink Mapping

Name	AR:DataAccessMode	AR:Port	AR:Element
MRPort (ECU mode)	ModeReceive	NewModePort	mdgEcuModes
RPort_DE1	ImplicitReceive	RPort	DE1
RPort_DE2	ImplicitReceive	RPort	DE2
RPort_DE3	ImplicitReceive	RPort	DE3

## Map Simulink Port to AUTOSAR Port

Name	AR:DataAccessMode	AR:Port	AR:Element
MRPort (ECU mode)	ModeReceive	NewModePort	mdgEcuModes
RPort_DE1	ImplicitReceive	RPort	DE1
RPort_DE2	ImplicitReceive	RPort	DE2
RPort_DE3	ImplicitReceive	RPort	DE3

# Using MATLAB for automating common tasks

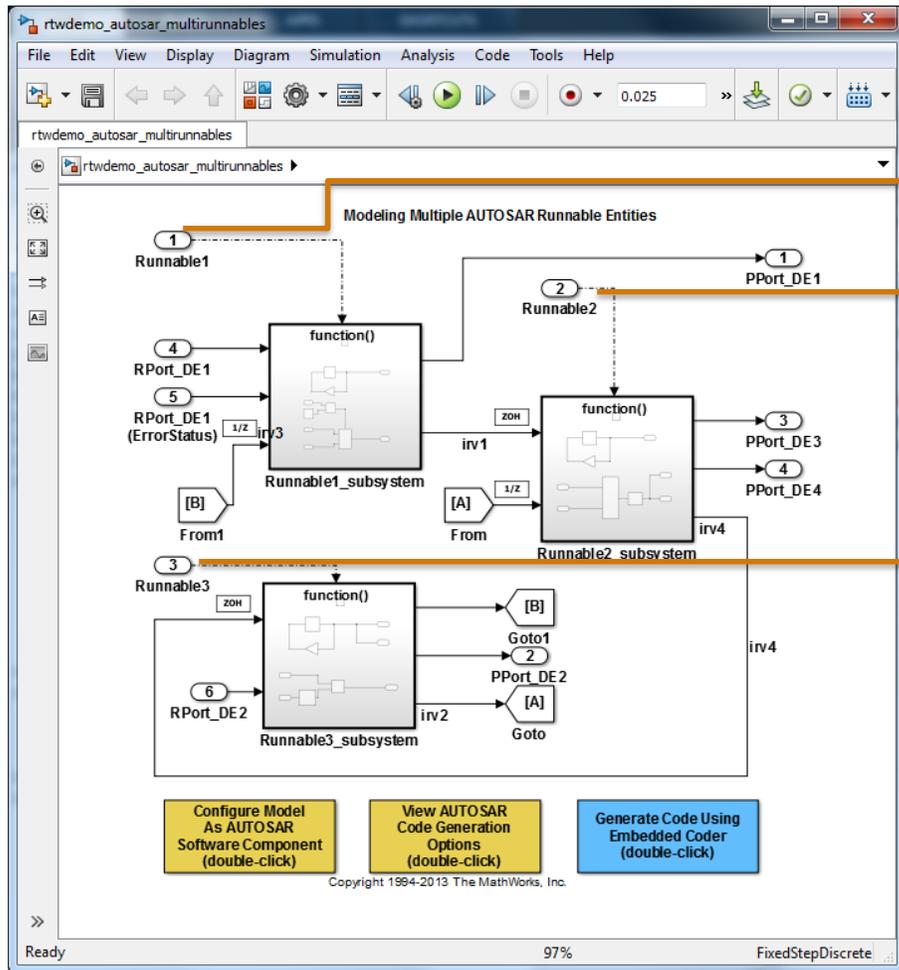
```
%% Setup AUTOSAR Configuration programmatically

model = 'Average_VehicleSpeed_Calculation';

% Modify AUTOSAR Properties
autosarProps = autosar.api.getAUTOSARProperties(model);
set(autosarProps, 'Input', 'IsService', true);
set(autosarProps, 'XmlOptions', 'ArxmlFilePackaging', 'SingleFile');

% Modify Simulink Mapping to AUTOSAR
slMap = autosar.api.getSimulinkMapping(model);
mapInport(slMap, 'Input' , 'Input', 'Input' , 'ExplicitReceive');
mapOutport(slMap, 'Output' , 'Output', 'Output' , 'ExplicitSend');
```

# Modeling Multiple Entry-Point Functions



Source Block Parameters: Runnable1

**Inport**

Provide an input port for a subsystem or model. For Triggered Subsystems, 'Latch input by delaying outside signal' produces the value of the subsystem input at the previous time step. For Function-Call Subsystems, turning 'On' the 'Latch input for feedback signals of function-call subsystem outputs' prevents the input value to this subsystem from changing during its execution. The other parameters can be used to explicitly specify the input signal attributes.

Main | Signal Attributes

Output function call

Minimum: [ ] Maximum: [ ]

Data type:  >>

Lock output data type setting against changes by the fixed-point tools

Port dimensions (-1 for inherited):

Variable-size signal:

Sample time (-1 for inherited):

Signal type:

Sampling mode:

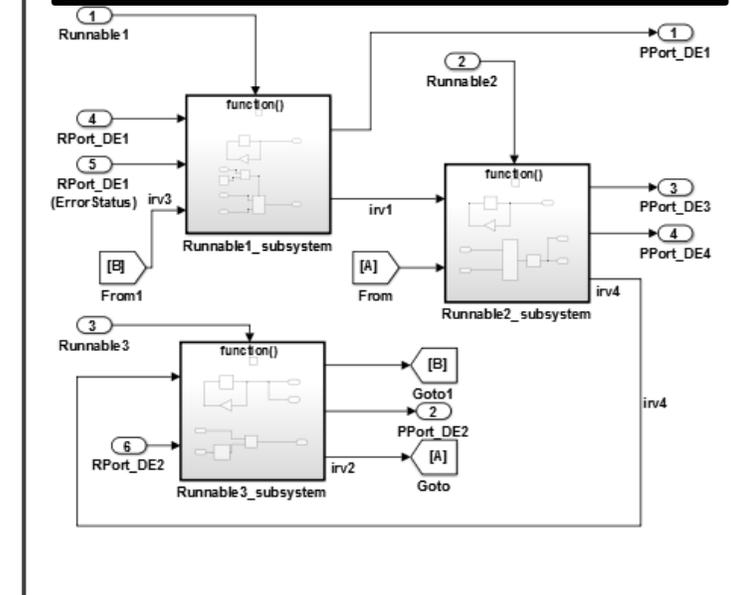
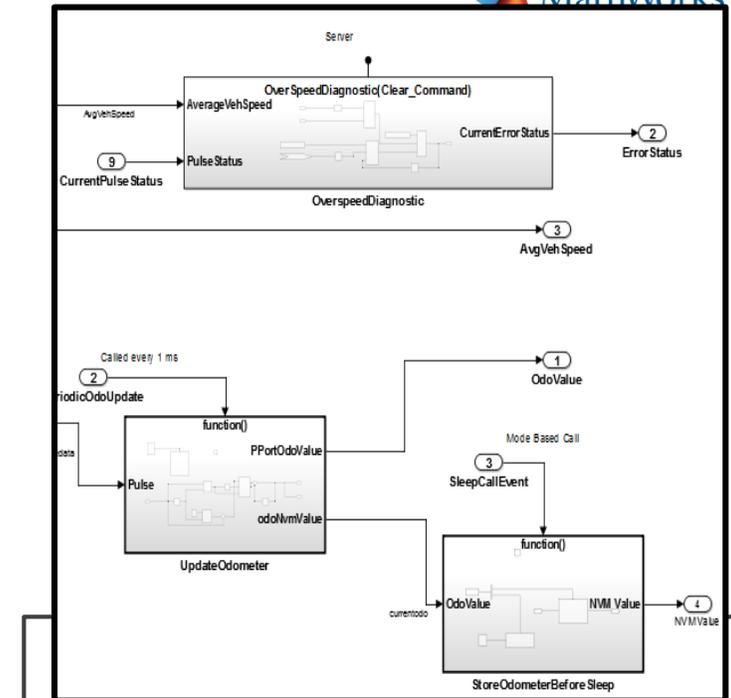
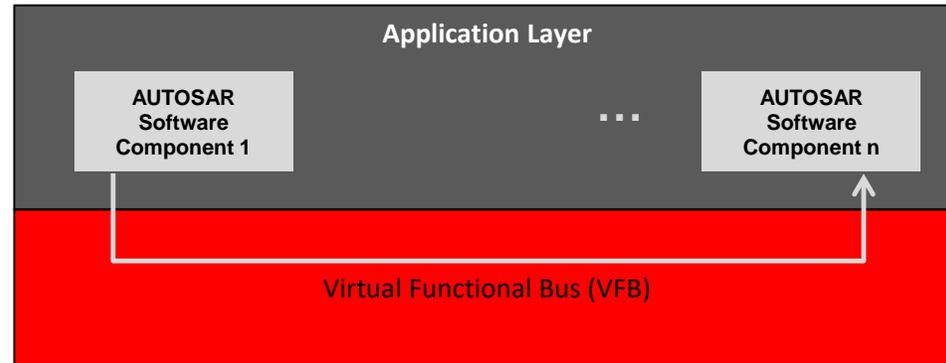
OK Cancel Help Apply

For the multi-runnable modeling pattern, each input port that represents a runnable trigger event will need to have the check box for Output function call checked as part of the inport properties.

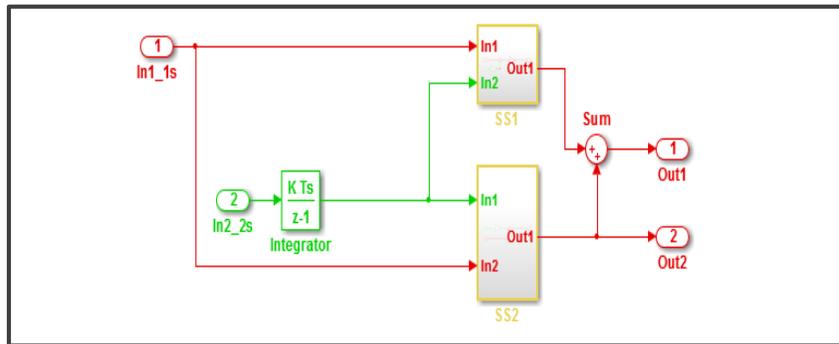
This allows for function triggers from a test harness model to be passed across a model reference boundary.

>> rtwdemo\_autosar\_multirunnables

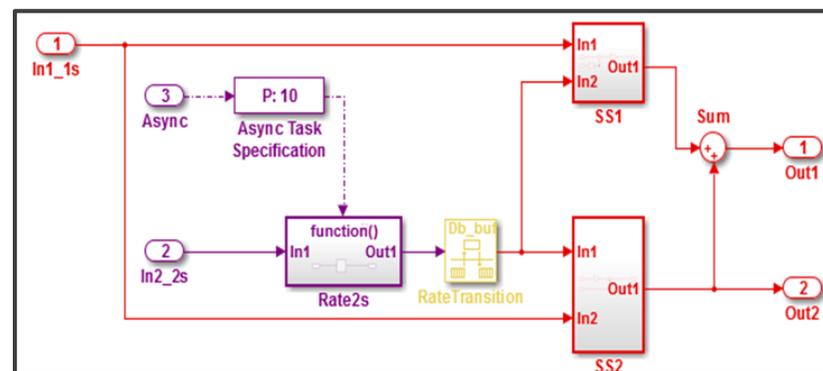
# Model AUTOSAR Components



**Multi-rate and Asynchronous**



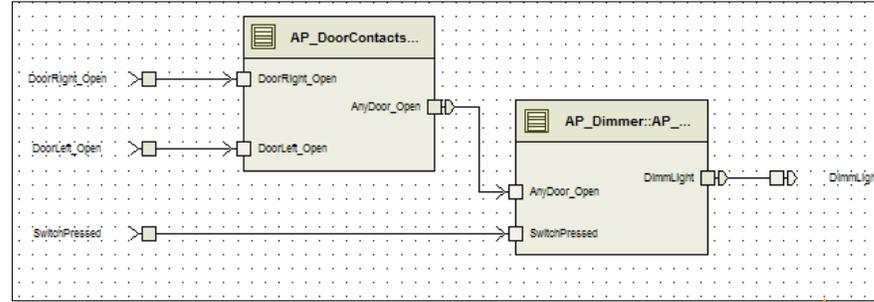
**Periodic rate-based**



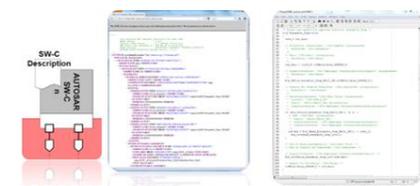
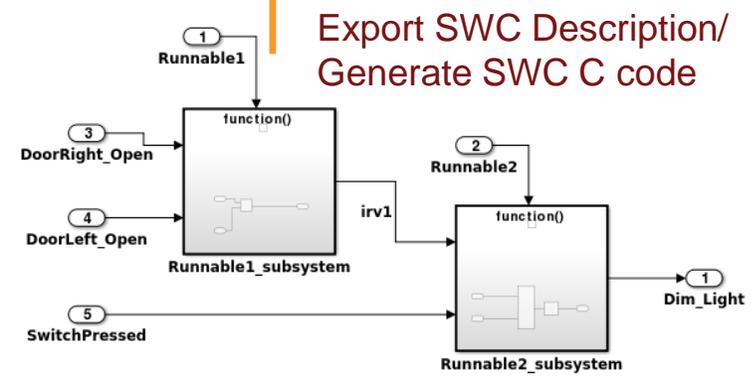
**Periodic and Asynchronous**

# Bottom-Up Workflow (Starting from Simulink)

## AUTOSAR Authoring Tool



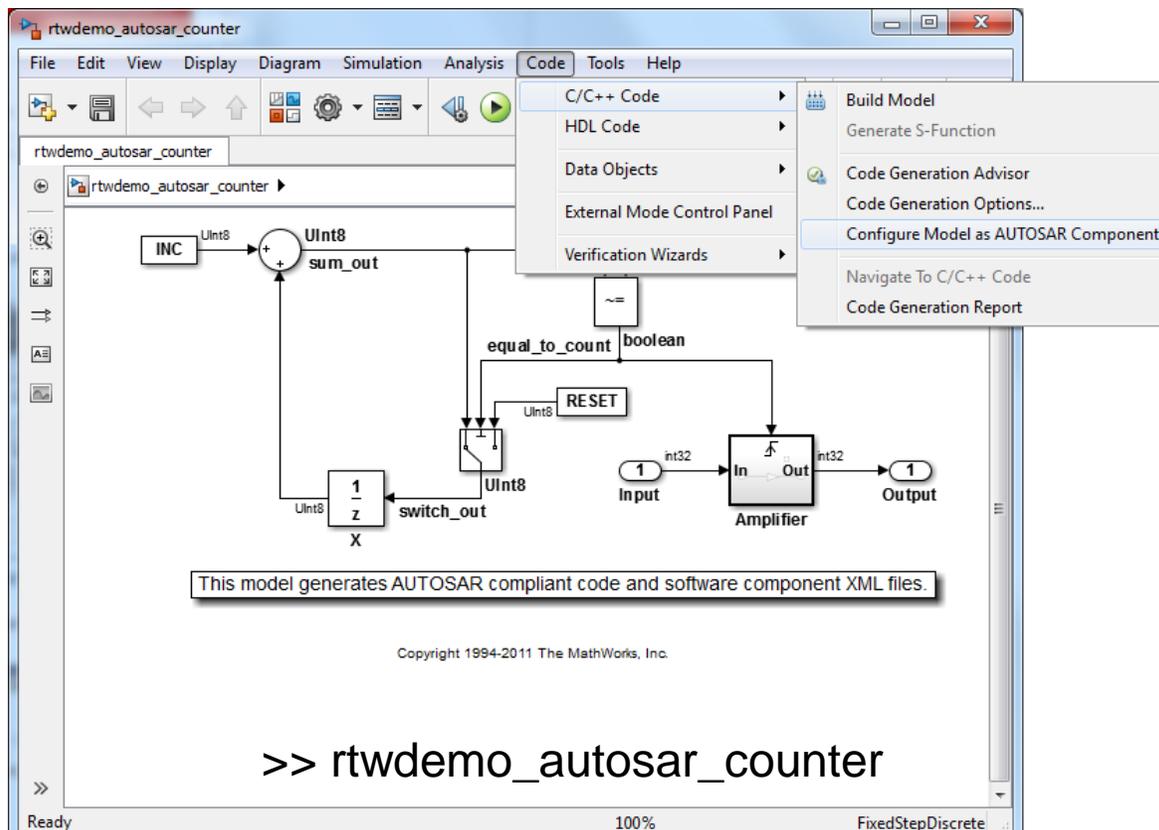
Import SWC Description

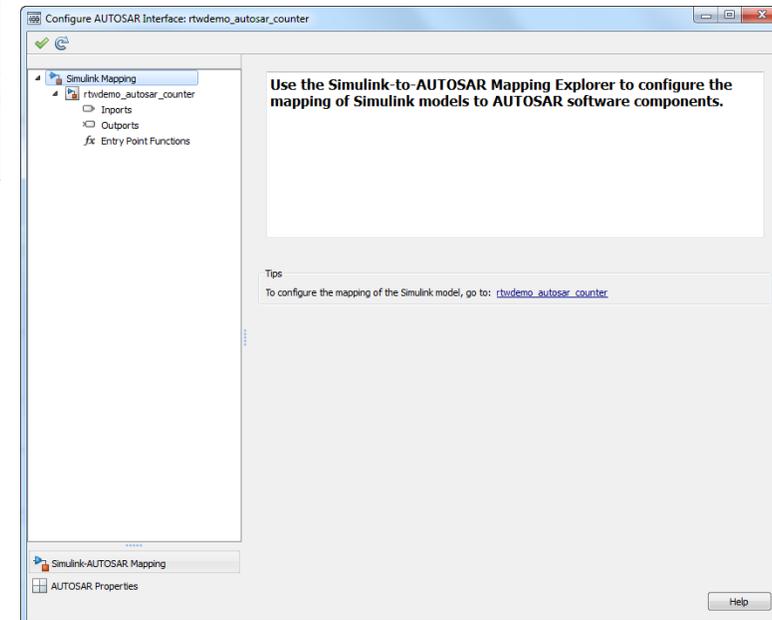
Export SWC Description/  
Generate SWC C code

# Launch AUTOSAR Configuration

- The method for Configuring AUTOSAR Properties is
  - selecting the Code Menu and then selecting C/C++ Code
  - Configure Model as an AUTOSAR Component. (This will bring up a dialog screen as shown.)



Configure AUTOSAR Interface (User Dialog)



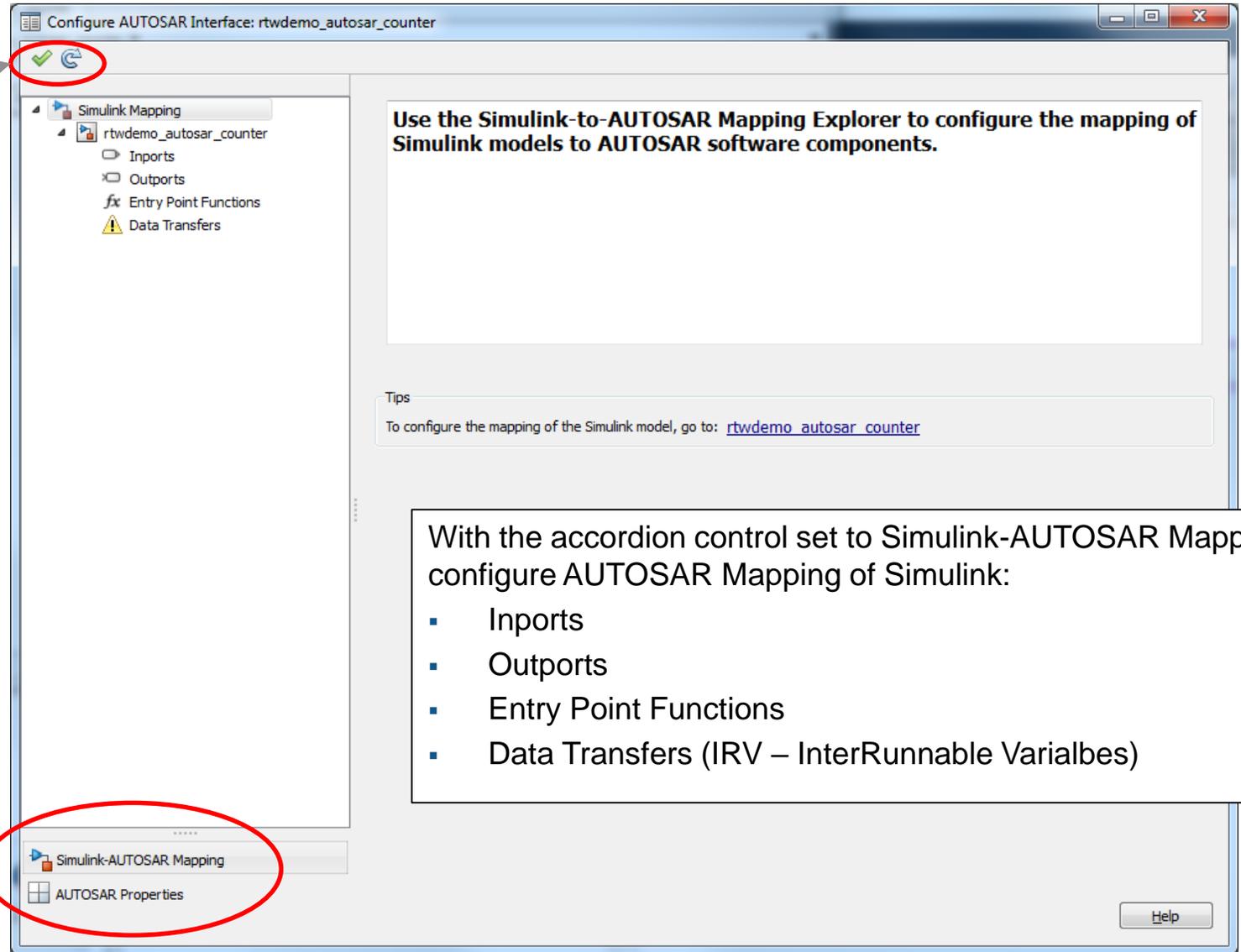
# Simulink-AUTOSAR Mapping Editor

View / Edit AUTOSAR Properties and Simulink Mappings

The Green Check mark is to Validate the AUTOSAR Configuration.

The blue Circle arrow is to refresh the Simulink elements.

The AUTOSAR Interface Configuration is split between two areas: Simulink-AUTOSAR Mapping and AUTOSAR Properties. See this accordion based UI control at the bottom left corner of the dialog screen



With the accordion control set to Simulink-AUTOSAR Mapping, configure AUTOSAR Mapping of Simulink:

- Inports
- Outports
- Entry Point Functions
- Data Transfers (IRV – InterRunnable Variables)

# Editing AUTOSAR Properties

View / Edit AUTOSAR Properties and Simulink Mappings

Configure AUTOSAR Interface: rtwdemo\_autosar\_counter

AUTOSAR

- AtomicComponents
  - rtwdemo\_autosar\_counter
    - ReceiverPorts
    - SenderPorts
    - SenderReceiverPorts
    - ModeReceiverPorts
    - ClientPorts
    - ServerPorts
    - NvReceiverPorts
    - NvSenderPorts
    - NvSenderReceiverPorts
    - Runnables
    - IRV
- S-R Interfaces
  - Input
  - Output
- M-S Interfaces
- C-S Interfaces
- NV Interfaces
- CompuMethods
- XML Options

Simulink-AUTOSAR Mapping

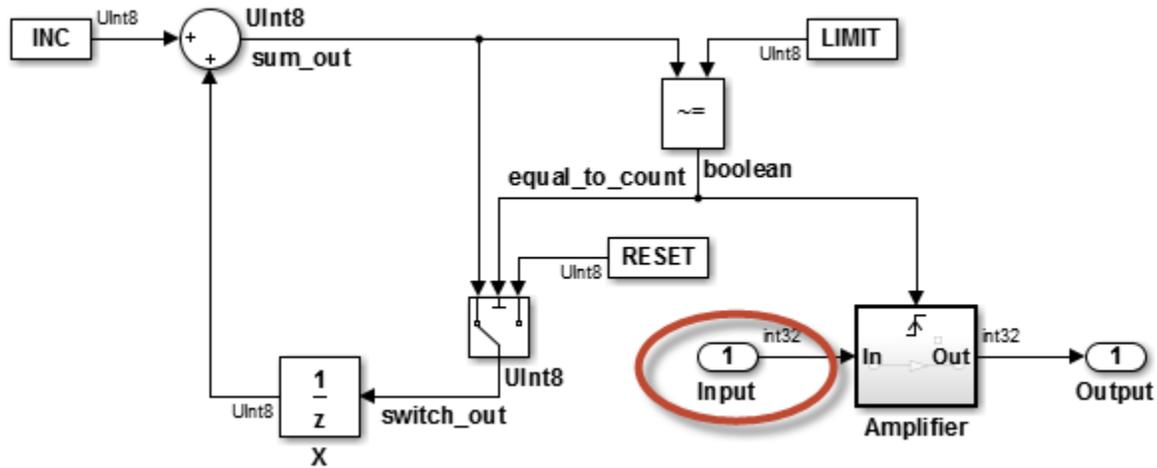
**AUTOSAR Properties**

Name	Kind
rtwdemo_autosar_counter	Application

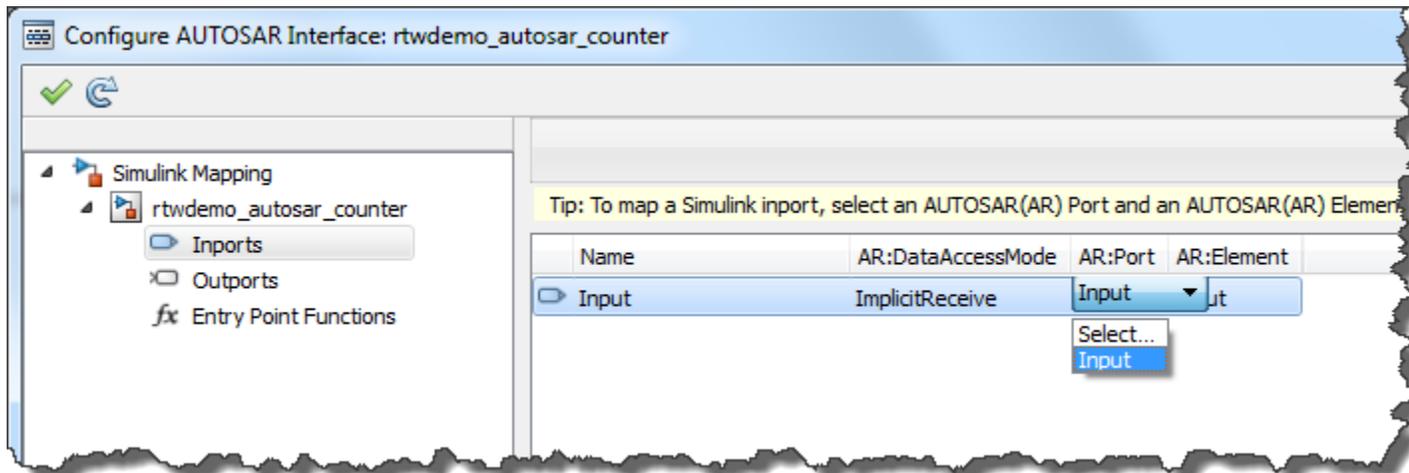
With the accordion control set to AUTOSAR Properties, the user can configure AUTOSAR elements / attributes such as:

- Add / Remove / Edit AUTOSAR Entities such as Components / Ports and Interfaces
- Configure ARXML options such as modular or single file generated on export or build; package paths; allow or not allow implementation types.

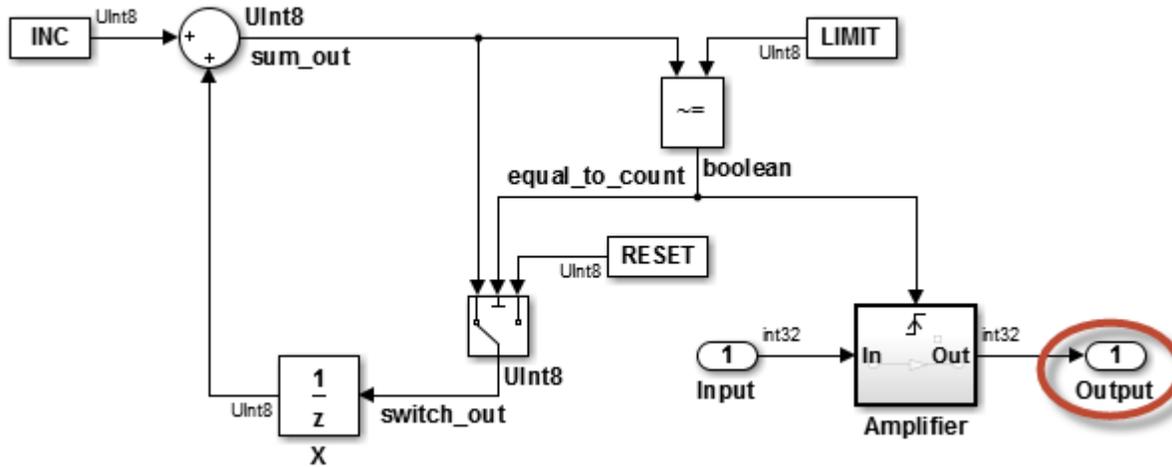
# Mapping Inports to AUTOSAR Receiver Ports



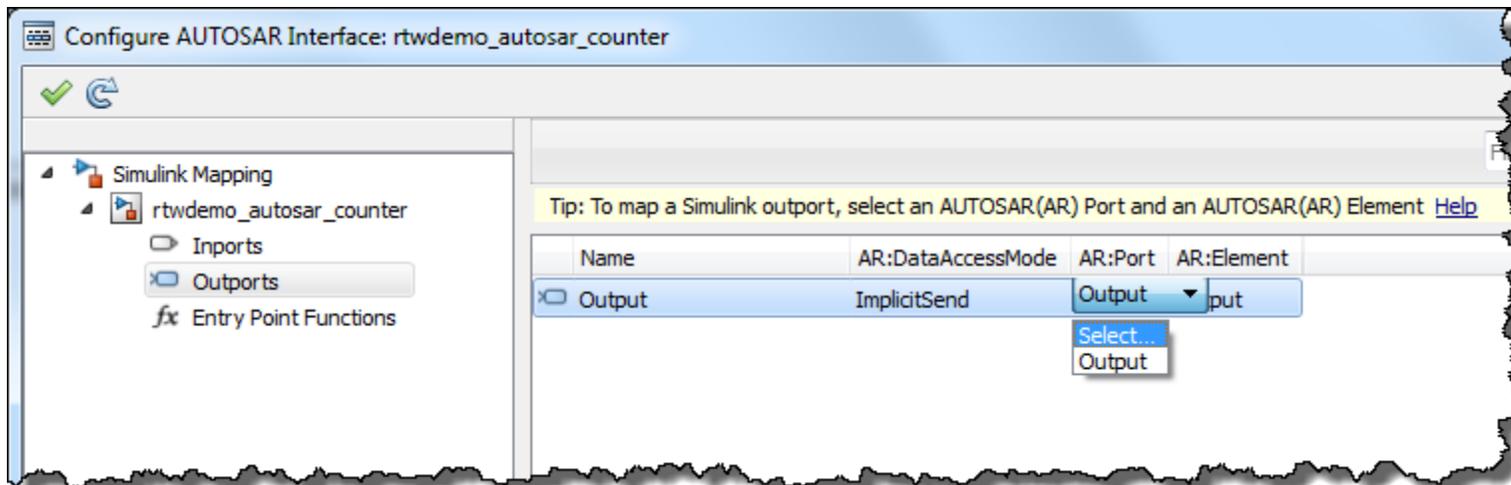
Here is an example of mapping a Simulink Inport to a AUTOSAR Port Data Element. In this case, you can see the Simulink Inport called Input is mapped to the AR:Port of Input. The actual selection of the AR:Element is a bit hidden in this view. Notice that the AR:DataAccessMode is also available for the user to select from this dialog.



# Map Outputs to AUTOSAR Sender Ports



Here is an example of mapping a Simulink Inport to a AUTOSAR Port Data Element. In this case, you can see the Simulink Inport called Input is mapped to the AR:Port of Input. The actual selection of the AR:Element is a bit hidden in this view. Notice that the AR:DataAccessMode is also available for the user to select from this dialog.



# Map Entry Point Functions for a Model

Configure AUTOSAR Interface: rtwdemo\_autosar\_counter

Tip: To map a Simulink entry point function, select an AUTOSAR(AR) Runnable [Help](#)

Name	AR:Runnable
fx Step Function	Runnable_Step
fx Initialize Function	Select ... Runnable_Init Runnable_Step

Here is an example of mapping a Simulink Function Entry Point to an AUTOSAR Runnable. Note: The Initialize Function most likely will get mapped to the Runnable\_Step (In this example). This Initialize Function is for blocks within the Software Component that need initialization such as 1/z blocks. This is not the same as user initialization of signals or variables.

# Generate Code

rtwdemo\_autosar\_counter

File Edit View Display Diagram Simulation Analysis Code Tools Help

rtwdemo\_autosar\_counter

rtwdemo\_autosar\_counter

INC UInt8

sum\_out UInt8

LIMIT UInt8

equal\_to\_count boolean

RESET UInt8

switch\_out UInt8

1/z X

Input int32

Amplifier

Output int32

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

Copyright 1994-2011 The MathWorks, Inc.

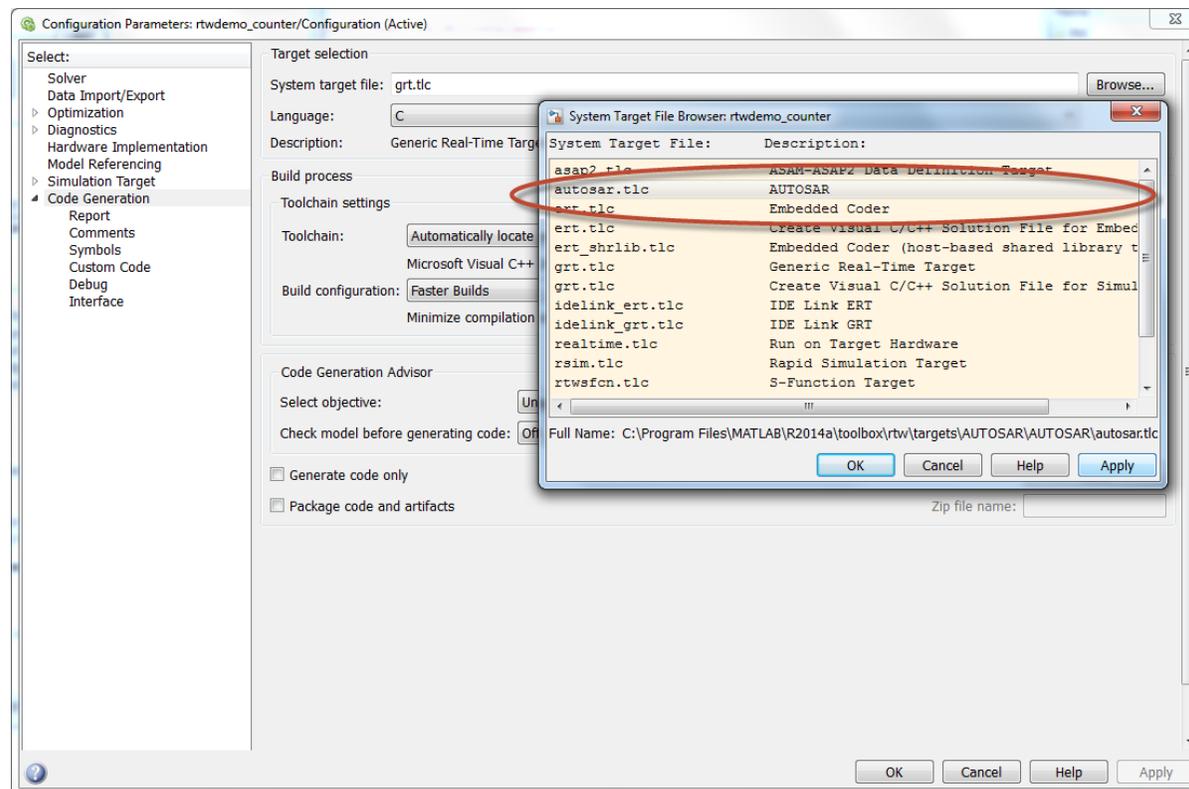
Ready 108% FixedStepDiscrete

Build model into AUTOSAR compliant Code (Control B or build Icon show to the left).

- Generates both C Code & AUTOSAR Software Component description files (ARXML files).
- Code uses RTE APIs to access AUTOSAR Ports such as Sender Receiver ports or Client / Server Ports as needed.

# Select AUTOSAR Target

- The mechanism for selecting an AUTOSAR target is similar to selecting an ERT target
  - via the Simulink Configuration Parameters menu, Code Generation Tab , System target file.

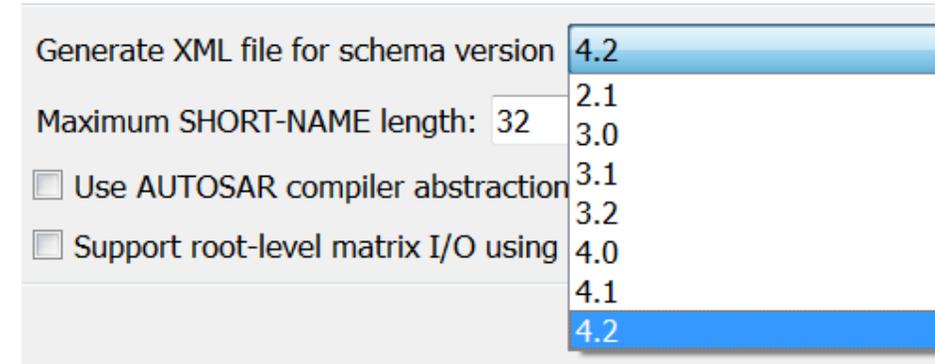


# Overview of Generated ARXML

- Generated ARXML contains
  - Component and Internal Behavior
  - Datatypes
  - Implementation information
    - Lists all generated source code and ARXML files
  - Interfaces
  - Other entities

# AUTOSAR Schema Versions

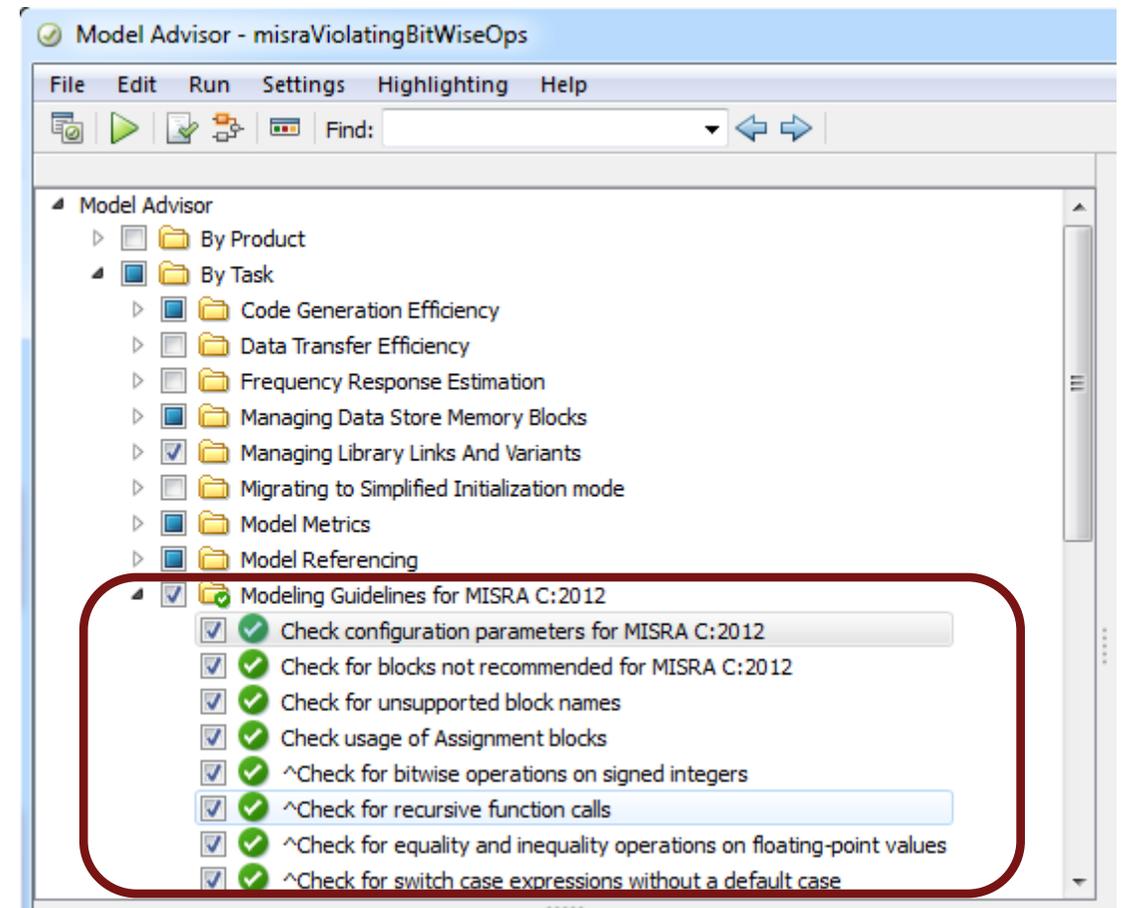
- Seamless support for AUTOSAR Releases
  - Import detects AUTOSAR 2.x – 4.x release from arxml file
  - User selects AUTOSAR release from configuration set options for code generation and arxml export



MATLAB Release	AUTOSAR Release
R2015b, R2016a/b, R2017a	2.1, 3.0, 3.1, <b>3.2</b> (Rev 3.2.2), 4.0, 4.1, <b>4.2</b> (Rev 4.2.1, 4.2.2)
R2014b, R2015a	2.1, 3.0, 3.1, 3.2, 4.0, <b>4.1</b> (Rev 4.1.1)
R2012a/b, R2013a/b, R2014a	2.1, 3.0, 3.1, 3.2, <b>4.0</b> (Rev 4.0.2)
R2011b	2.0, 2.1, 3.0, 3.1, 3.2
R2010a/b, R2011a	2.0, 2.1, 3.0, 3.1
R2009a/b	2.0, 2.1, 3.0
R2008a/b	2.0, 2.1

# MISRA C:2012 for AUTOSAR target

- 100% Compliance with MISRA C:2012 Mandatory and Required rules



HOME PLOTS APPS

Search Documentation

New Script New Open Compare Import Data Save Workspace New Variable Open Variable Clear Workspace Analyze Code Run and Time Clear Commands Simulink Library Layout Preferences Set Path Parallel Add-Ons Help Community Request Support

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

C: > Work > AUTOSAR > Demo

Current Folder

Name ^
Average_VehicleSpeed_Calculation.slx

Details

Select a file to view details

Command Window

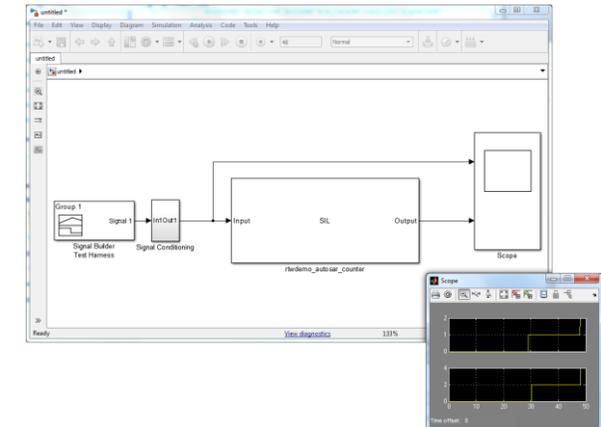
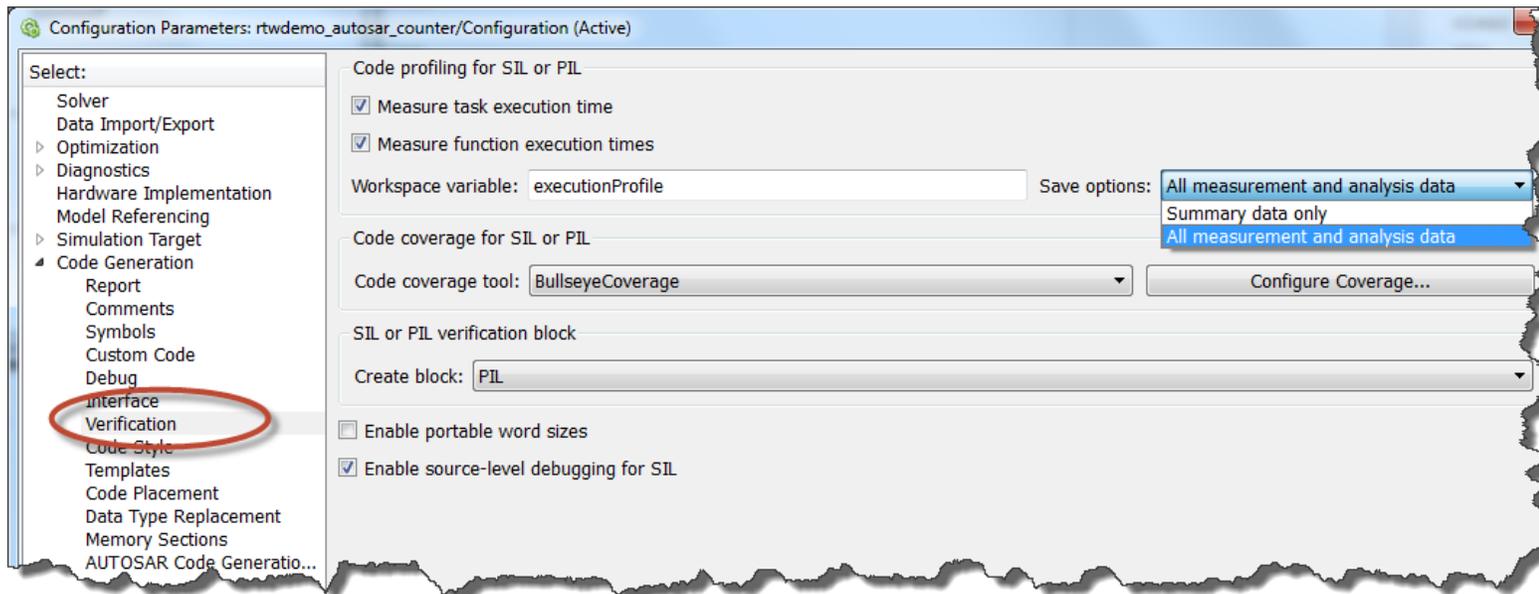
```
fx >>
```

Workspace

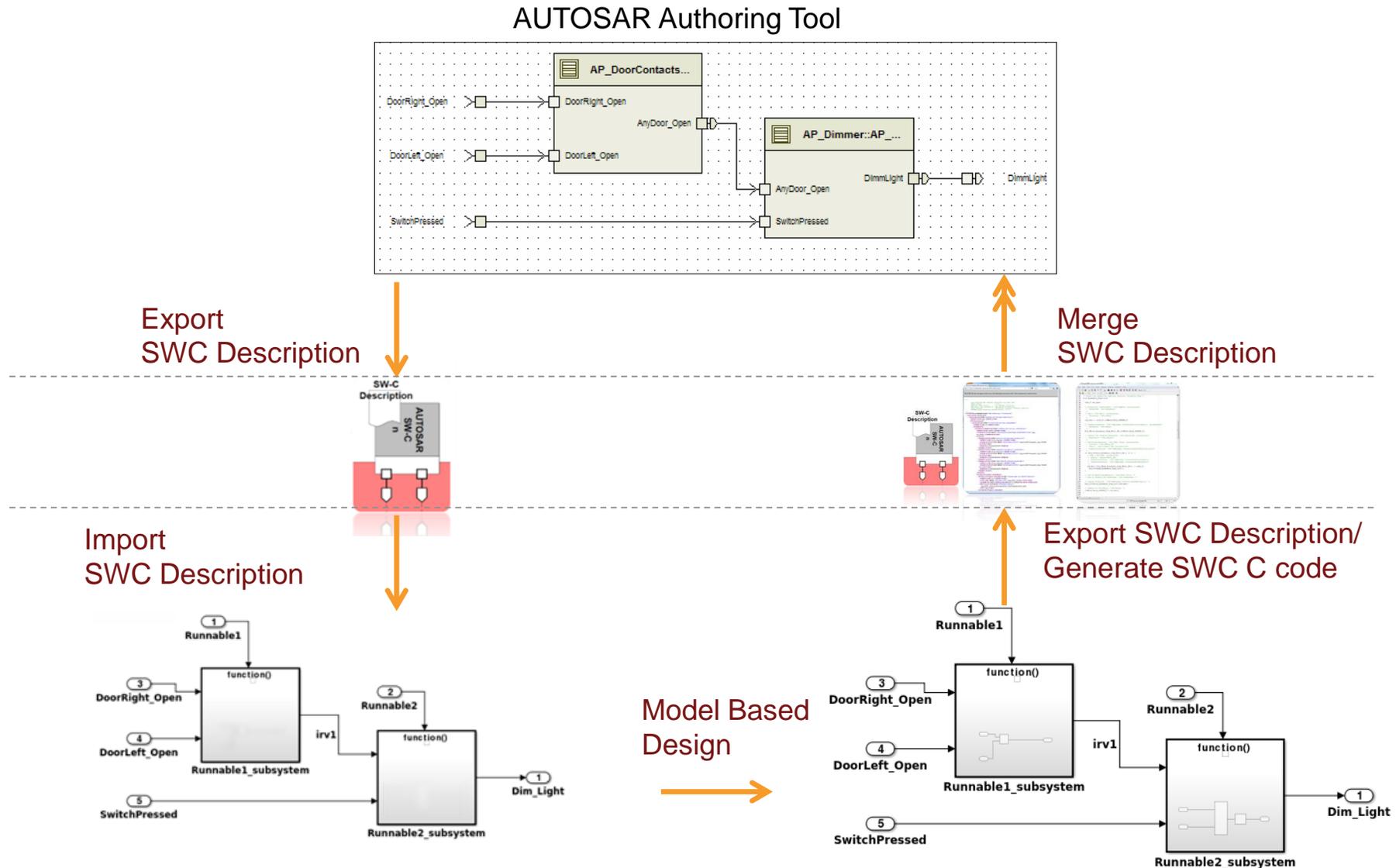
Name ^	Value
--------	-------

# Verification with Software- and Processor-In-The-Loop (PIL)

- Support for SIL/PIL with AUTOSAR target
- Profile code and measure execution time on target
- Develop a custom PIL target for AUTOSAR using the toolchain build approach

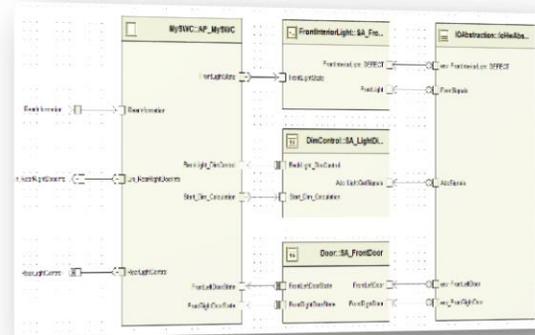


# Top-Down Workflow (Starting from SWC Description)



# Top Down Workflow

## AUTOSAR Authoring Tool

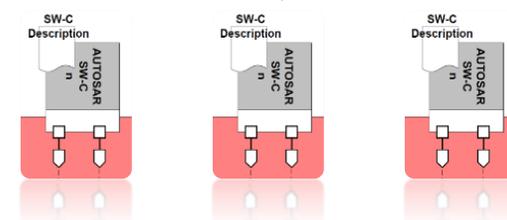


## Top Down Workflow

Starts with Authoring Tool, then user exports ARXML files from Authoring tool.

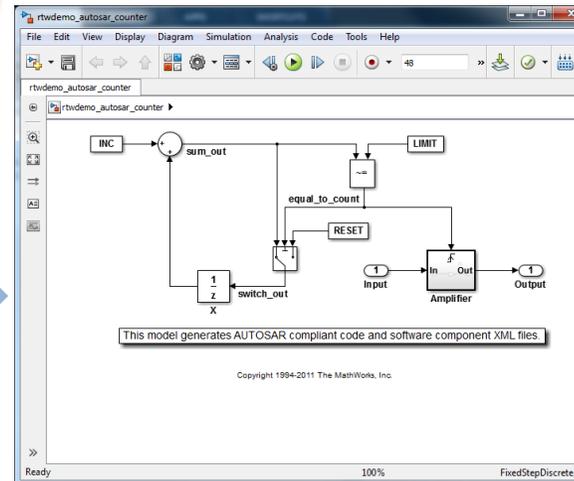
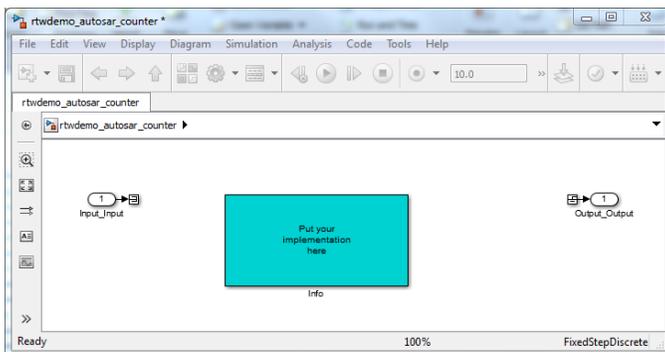
User can then either import the ARXML files into a new Simulink Skeleton model or Update an existing Simulink Model.

## ARXML Files



## Update existing Simulink model

## Import as new Simulink model



OR

# Importing ARXML Files

```
%Import ARXML Files
importerObj = arxml.importer('rtwdemo_autosar_multirunnables.arxml')

%Create new model with interfaces
model = importerObj.createComponentAsModel('/pkg/swc/ASWC');
```

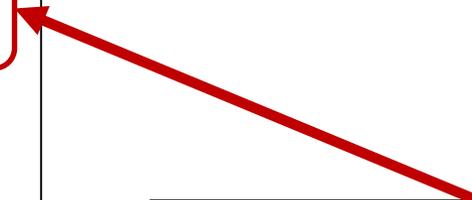
## Top Down Workflow

Commands to create a skeleton model with no internal behavior setup in the skeleton model (no runnables)

```
importerObj =
```

The file "C:\Backup\General\_Work\_PSPs\Documents\MAC US 2014\rtwdemo\_autosar\_multirunnables.arxml" contains:

- 1 Application-Software-Component-Type: '/pkg/swc/ASWC'
- 0 Sensor-Actuator-Software-Component-Type.
- 0 CalPrm-Component-Type.
- 0 Client-Server-Interface.



Note: After running the arxml.importer command, here is the package that is needed for the second command of create Component as Model

# Import with Internal Behavior

```

1 %cleanup
2 - bdclose('all');
3 - clear;
4
5 %Import ARXML Files
6 % After Creating the Importer Object, you will observe
7 % what components are available.
8 - importerObj = arxml.importer('rtwdemo_autosar_multirunnables.arxml') %#ok<NOPTS>
9
10 %Create new model with interfaces
11 % model = importerObj.createComponentAsModel('/pkg/swc/ASWC');|
12
13 %Create new model with interfaces and internal behavior
14 - model = importerObj.createComponentAsModel('/pkg/swc/ASWC', ...
15 'CreateInternalBehavior',true);

```

```
>> importExample
```

```
importerObj =
```

The file "C:\00\_mdInfo\AUTOSAR\AUTOSAR\_Seminar\Demo2\_TopDown\rtwdemo\_autosar\_multirunnables.arxml" contains:

```
1 Application-Software-Component-Type:
  '/pkg/swc/ASWC'
```

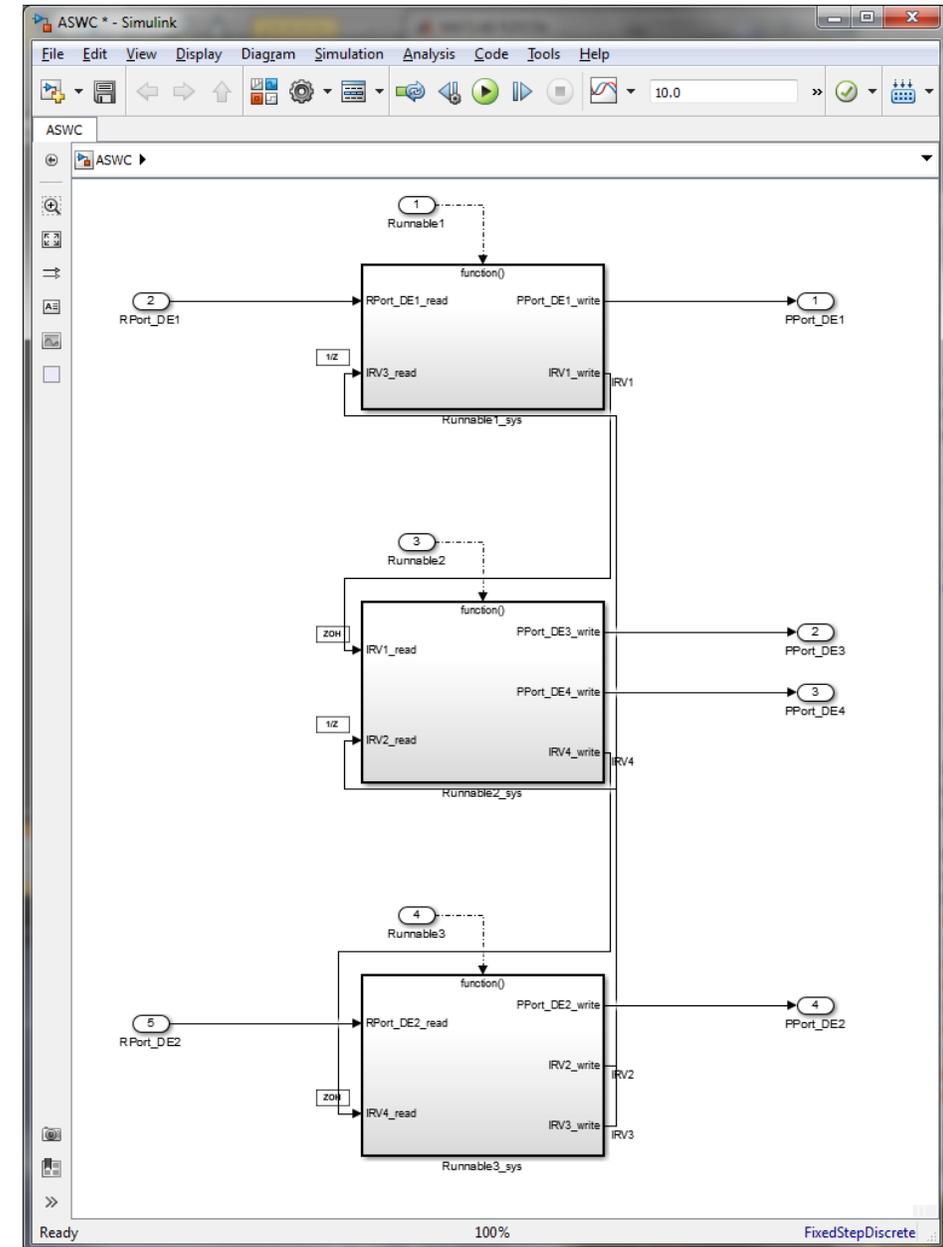
```
0 Sensor-Actuator-Software-Component-Type.
```

```
0 Parameter-Software-Component-Type.
```

```
0 Client-Server-Interface.
```

## Top Down Workflow

Commands to create a skeleton model with internal behavior setup in the skeleton model (create with runnables)



# Updating Existing Models from ARXML

V1.arxml

Updated to V2.arxml

The screenshot shows a text comparison window for two ARXML files. The left pane shows V1.arxml and the right pane shows V2.arxml. The differences are highlighted in red in the V2 pane:

```

<TIMING-EVENT UUID="c89mmmjk9083-1aa1-5915-9e56-b61a2122761e">
  <SHORT-NAME>Event_NewRunnable</SHORT-NAME>
  <START-ON-EVENT-REF DEST="RUNNABLE-ENTITY"/>/pkg/swc/ASWC/IB/R
  <PERIOD>1</PERIOD>
</TIMING-EVENT>

<RUNNABLE-ENTITY UUID="c78xurkx-d65d-514b-2b2e-asdfsdfw3341">
  <SHORT-NAME>NewRunnable</SHORT-NAME>
  <MINIMUM-START-INTERVAL>0</MINIMUM-START-INTERVAL>
  <CAN-BE-INVOKED-CONCURRENTLY>false</CAN-BE-INVOKED-CONCURRENTLY>
  <SYMBOL>NewRunnable</SYMBOL>
</RUNNABLE-ENTITY>
  
```

The status bar at the bottom indicates "2 difference section(s)", "Same", "Insert", and "Load time: 0.09 seconds".

# Update Existing Models from ARXML

```

1 %cleanup
2 - bdclose('all');
3 - clear;
4
5 - open_system('ASWC'); % Model needs to be open in order to perform update Model Command
6 |
7 %Import ARXML Files
8 - importerObj = arxml.importer('rtwdemo_autosar_multirunnables_v2.arxml')
9
10 %Update existing model
11 - importerObj.updateModel('ASWC')

```

## Top Down Workflow

Commands to update an existing Simulink model that already has an AUTOSAR configuration.

Notice that a report was created such that the user can understand what has changed in the model.

## AUTOSAR Update Report for ASWC

Software component `/pkg/swc/ASWC`  
Original model saved as: `ASWC_backup`

This report details the updates applied to Simulink model `ASWC` based on differences between the imported arxml and the existing AUTOSAR configuration contained in the model. A backup of the original model has been saved to `ASWC_backup` ([compare models](#)). The report also recommends manual model changes.

### Simulink

#### Automatic Model Changes

#### Automatic Workspace Changes

#### Required Manual Model Changes

#### Optional Manual Workspace Changes

### AUTOSAR

#### Automatic AUTOSAR Element Changes

**Added** ConstantSpecification /pkg/dt/Ground/DefaultInitValue\_Single  
**Added** FloatingPoint /pkg/dt/Single  
**Updated** Type reference of InData /pkg/swc/ASWC/IB/IRV4 from /pkg/dt/Double to /pkg/dt/Single

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Print Go To Find Comment Indent Breakpoints Run Run and Advance Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Search Documentation

C:\Work\AUTOSAR\AAM\ARXML

Current Folder

- slprj
- ImportARXML.m
- SBR\_Logic.slx
- SBR\_Logic\_backup.slx
- SBR\_Logic\_swc.arxml
- SBR\_Logic\_swc\_modified.arxml
- SBR\_UpdateModel.m

Editor - C:\Work\AUTOSAR\AAM\ARXML\ImportARXML.m

```
1 %cleanup
2 bdclose('all');
3 clear;
4
5 %Import ARXML Files
6 Obj = arxml.importer('SBR_Logic_swc.arxml');
7
8 %Create new model with interfaces and internal behavior
9 model = Obj.createComponentAsModel('/SBR_Logic_pkg/SBR_Logic_swc
```

Workspace

Name	Value
------	-------

Command Window

```
fx >>
```

Select a file to view details

제품 솔루션 대학 커리큘럼 지원 커뮤니티 이벤트 안내

[문의하기](#) [구입 방법](#) [Allen님](#)

Hardware Support

Search Hardware Support

Hardware Support ▼ 🔍

[Overview](#) | [Search Hardware Support](#) | [Request Hardware Support](#)

[Trial software](#) [Contact sales](#)

## AUTOSAR Support from Embedded Coder

Author and develop AUTOSAR software components for automotive systems

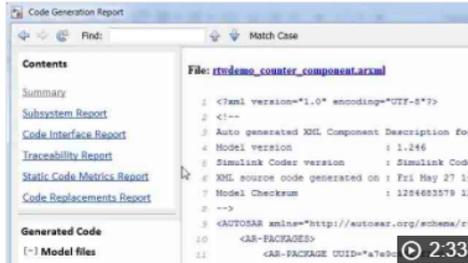
AUTOSAR (AUTomotive Open System ARchitecture) is an open and standardized automotive software architecture jointly developed by automobile manufacturers, suppliers, and tool developers.

Embedded Coder® Support Package for AUTOSAR Standard lets engineers model and simulate AUTOSAR software components, generate AUTOSAR production code, and verify AUTOSAR generated code using software- and processor-in-the-loop simulations. The support package also enables import and export of AUTOSAR Software Component descriptions that support top-down, bottom-up, and round-trip workflows involving third-party AUTOSAR authoring tools such as [DaVinci Developer](#).

### Platform and Release Support

See the [hardware support package system requirements table](#) for current and prior version, release, and platform availability.

View new features in the [release notes](#).



AUTOSAR Support from Simulink and Embedded Coder

#### Ready to install?

Before installing the support package, confirm you have the correct setup. [View system requirements and installation options](#).

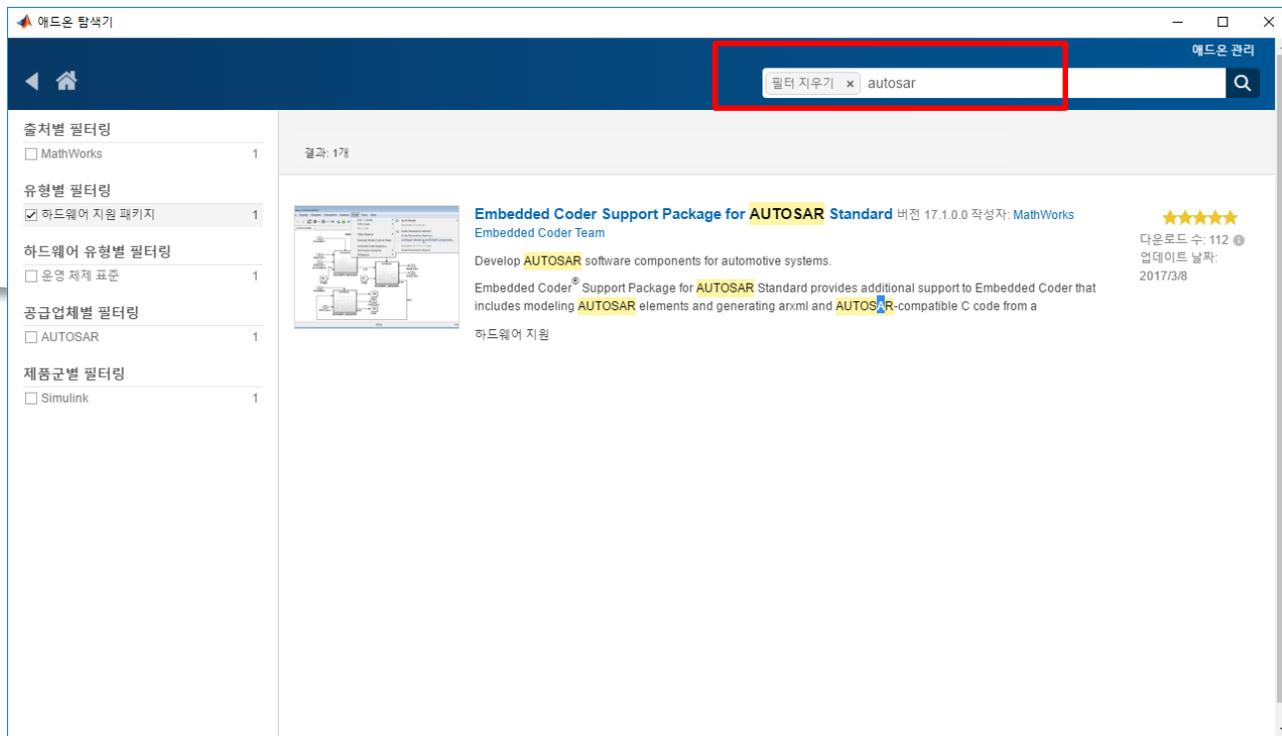
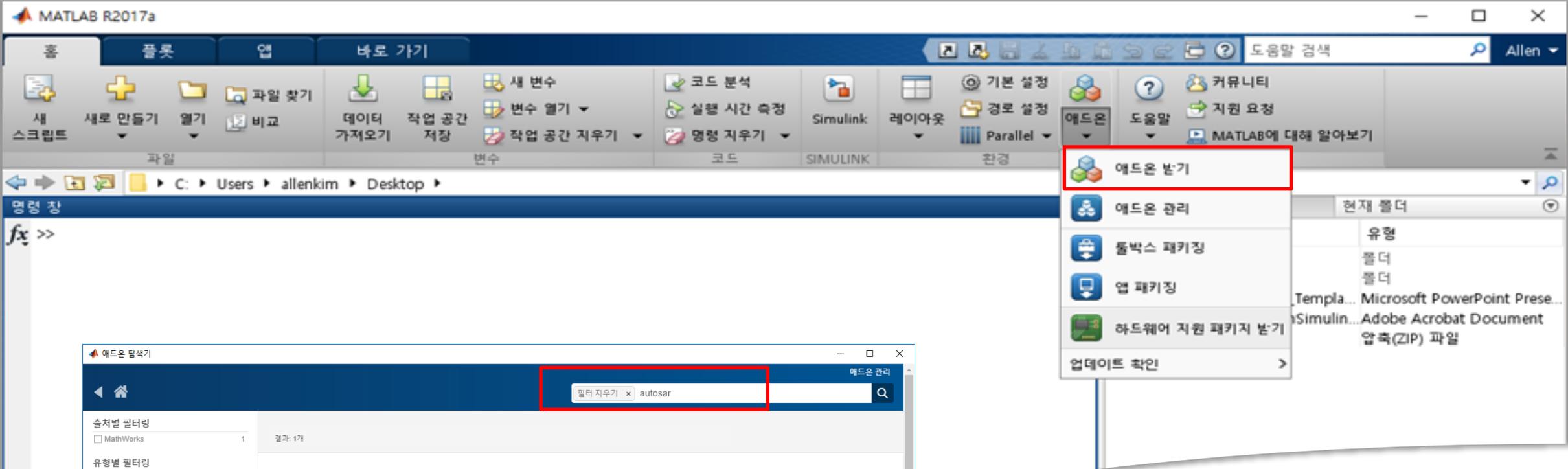
Get Support Package

#### Getting Started Resources

Expand all

▼ Videos

- [Model-Based Software Development: An OEM's Perspective \(24:55\)](#)
- [AUTOSAR Code Generation for Multiple Runnable Entities \(2:46\)](#)
- [Automatic Code Generation of AUTOSAR Software Components for Mass Production Application of Engine Management Systems: Process and Benefits \(26:03\)](#)
- [From Simulink to AUTOSAR Production Code \(5:16\)](#)
- [AUTOSAR Client-Server SIL Simulation \(5:38\)](#)
- [Simulation of AUTOSAR Software Components \(6:01\)](#)
- [Model AUTOSAR Variants in Simulink \(4:53\)](#)
- [Model AUTOSAR ECU Power-Up and Power-Down Behavior in Simulink \(7:20\)](#)



# Thank You!