



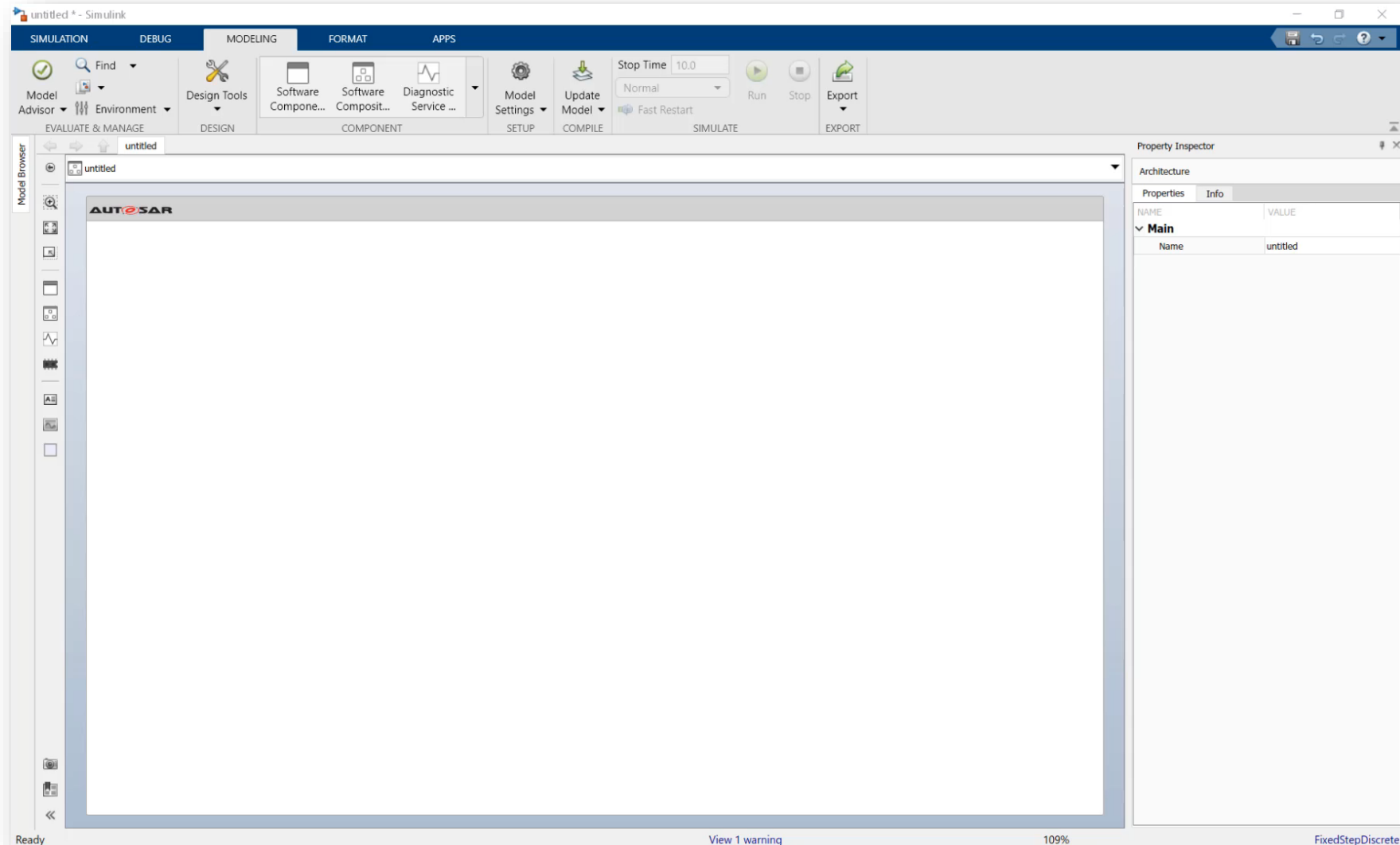
MathWorks  
AUTOMOTIVE  
CONFERENCE 2019

Simulink를 이용한 AUTOSAR SW 개발

From Architecture to Design to Testing

류성연

# Demo: AUTOSAR ASW Architecture Design

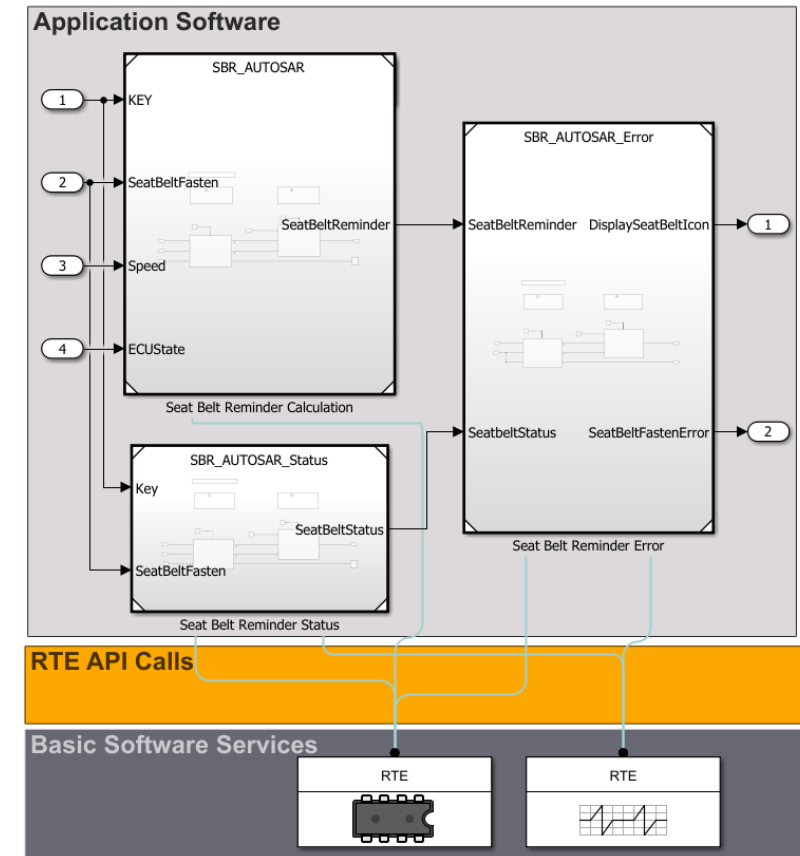


# Agenda

- AUTOSAR Blockset Introduction
  - Adaptive Platform
  - Classic Platform
  
- AUTOSAR ASW Development
  - AUTOSAR ASW architecture design
  - Testing in AUTOSAR Composition Editor
  
- System Composer with AUTOSAR Blockset

# Introduction to AUTOSAR Blockset

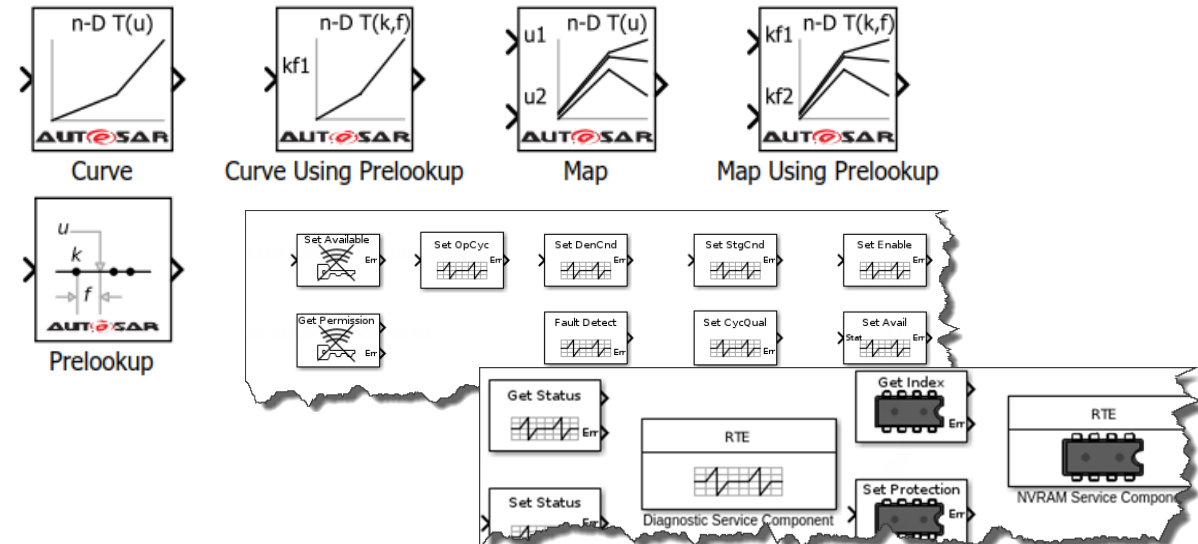
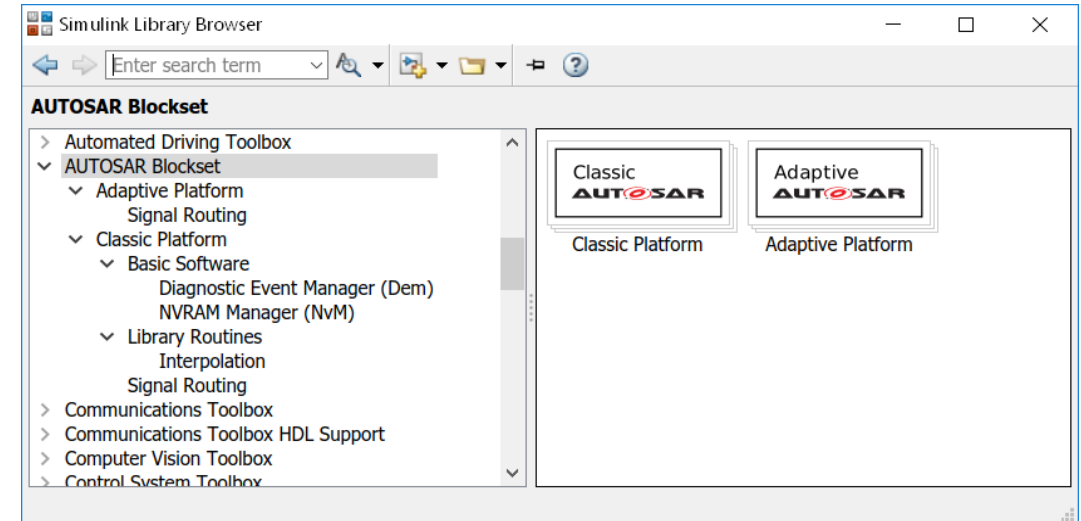
- Model and simulate AUTOSAR software in Simulink
  - Model AUTOSAR Classic and Adaptive software
  - Simulate AUTOSAR compositions and ECUs
  - Import and export AUTOSAR descriptions (ARXML files)
  - Create AUTOSAR software architecture
  
- Supports C/C++ production code generation and AUTOSAR ARXML export (with Embedded Coder)
  - Blocks for AUTOSAR library routines
  - Qualified for ISO 26262 standard (with IEC Cert Kit).



<https://kr.mathworks.com/videos/what-is-autosar-blockset--1550158089115.html>

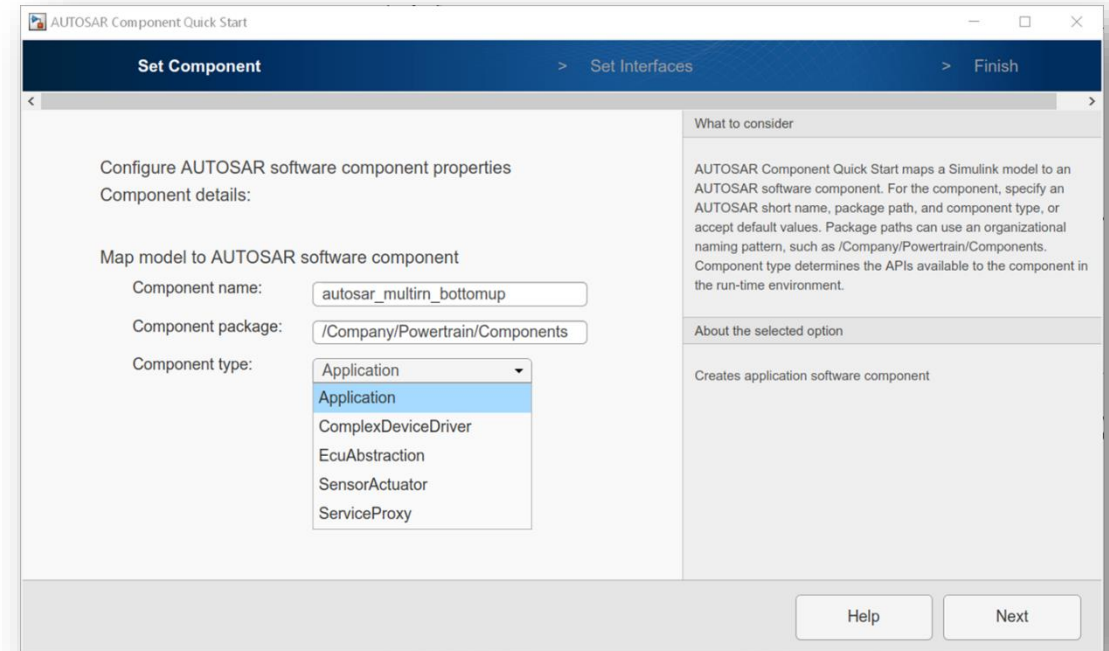
# AUTOSAR Blockset Library

- Blocks to model and simulate Basic Software and AUTOSAR library routines
  - Blocks for both Classic and Adaptive Platforms
  - Out-of-the-box AUTOSAR Basic Software emulation blocks
  - Generate code that calls into Basic Software services using Basic Software caller blocks
  - Reliably generate IFX and IFL AUTOSAR library routines during code generation



# AUTOSAR Component Quick Start

- Motivation
  - Make users more comfortable with the bottom-up workflow
  - AUTOSAR wizard is infrequently used by customers
  - Use same layout as Embedded Coder Quick Start
  
- Design
  - Step-by-step workflow
  - Help panel on the right of the UI
  - Additional configuration options
    - Specify Component type
    - Import properties
  - Supports Adaptive AUTOSAR

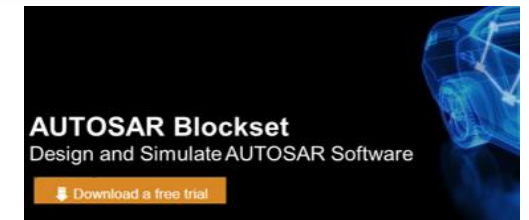
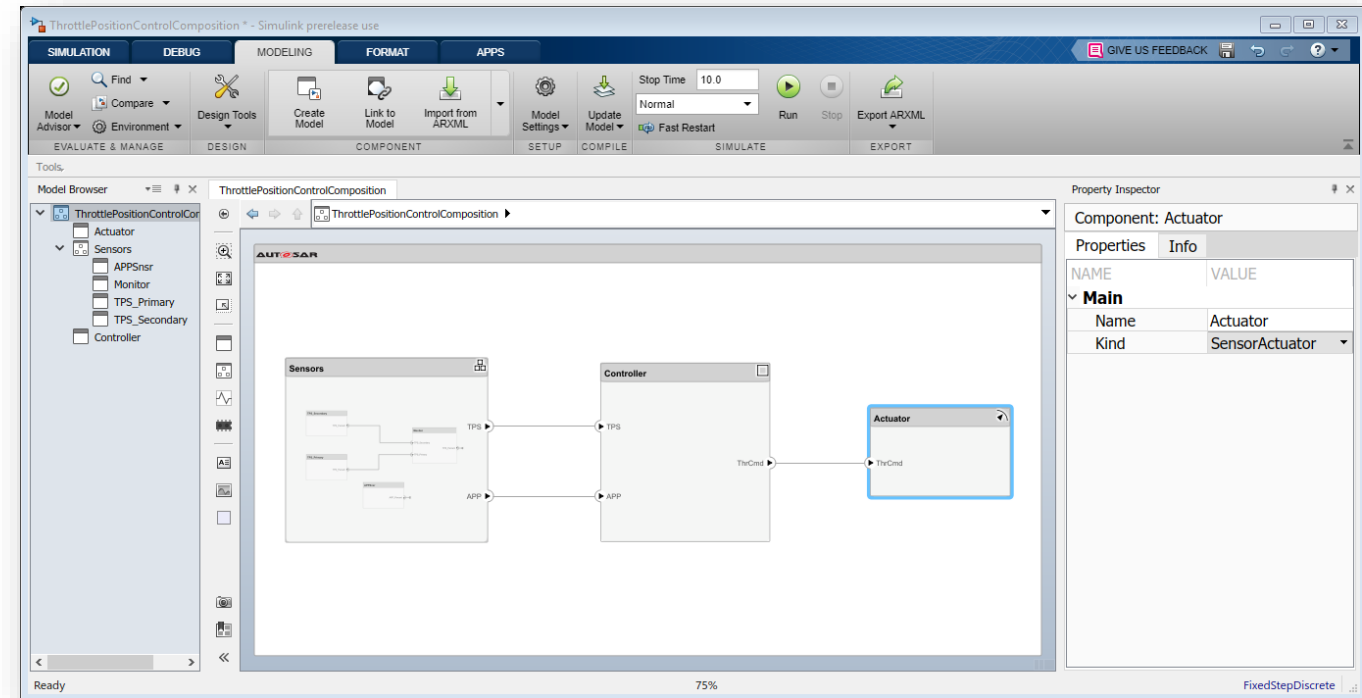


# AUTOSAR Software Architecture Modeling

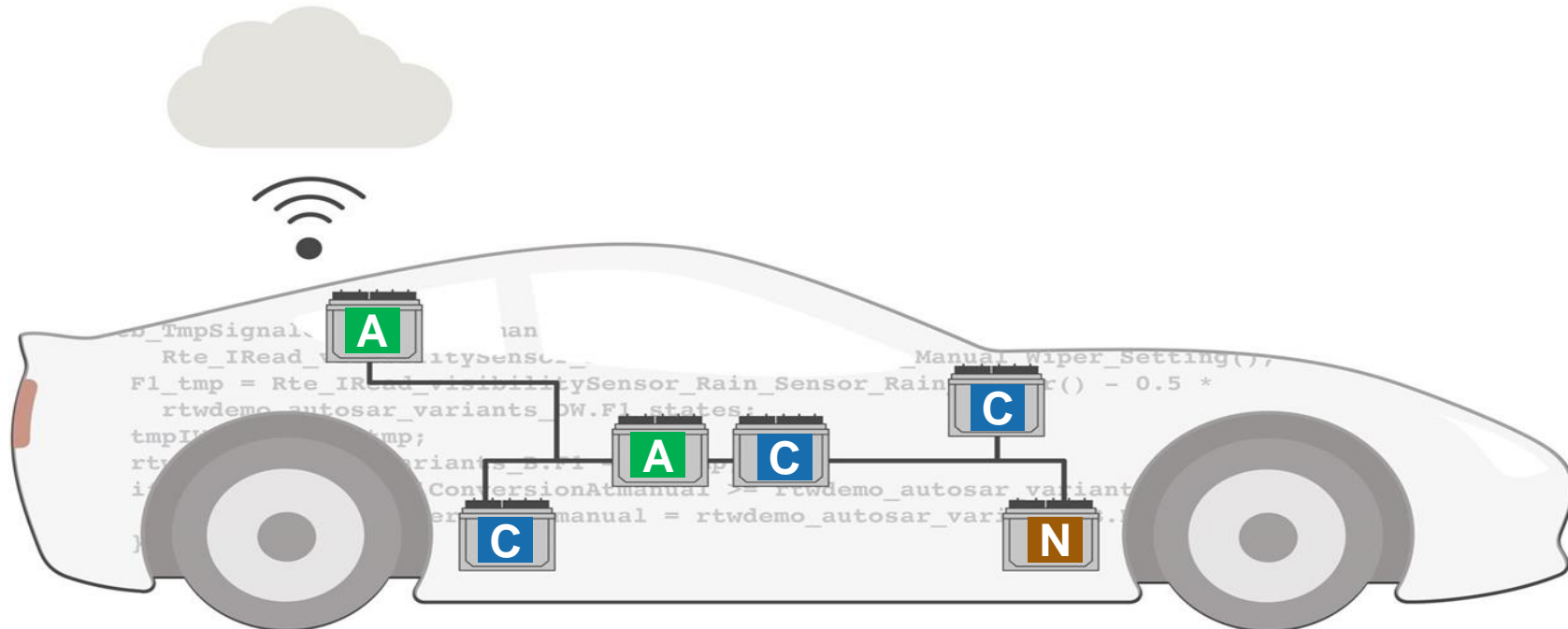
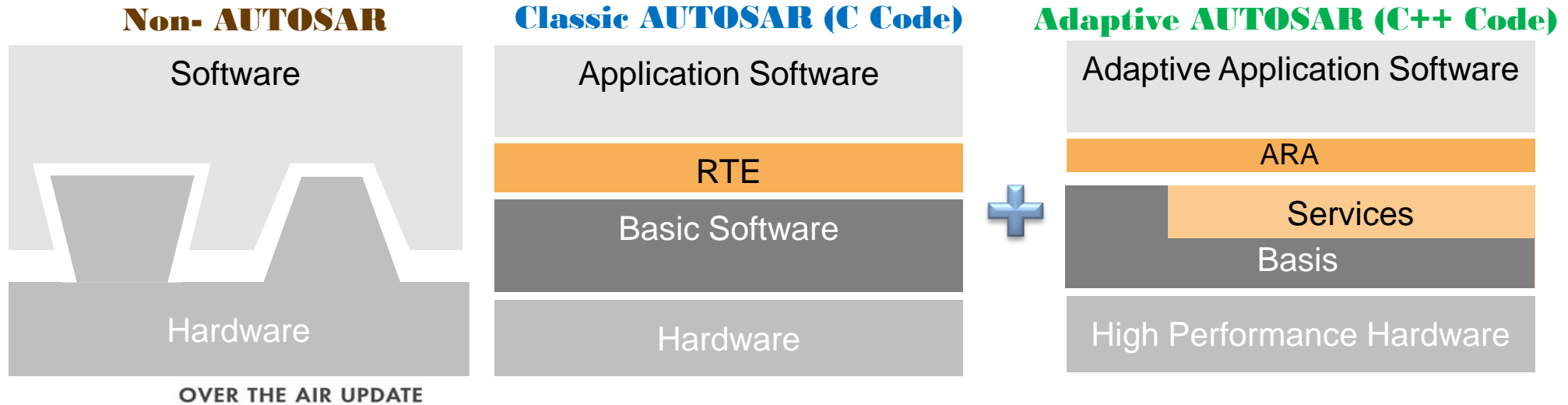
## AUTOSAR Composition Editor

→ Create architecture models

- Create an AUTOSAR architecture model in a canvas for developing AUTOSAR composition and component models for the Classic Platform.
- In the architecture model:
  - Add and connect AUTOSAR compositions and components.
  - Link components to requirements (requires Simulink Requirements™).
  - Define component behavior by creating or linking Simulink models.



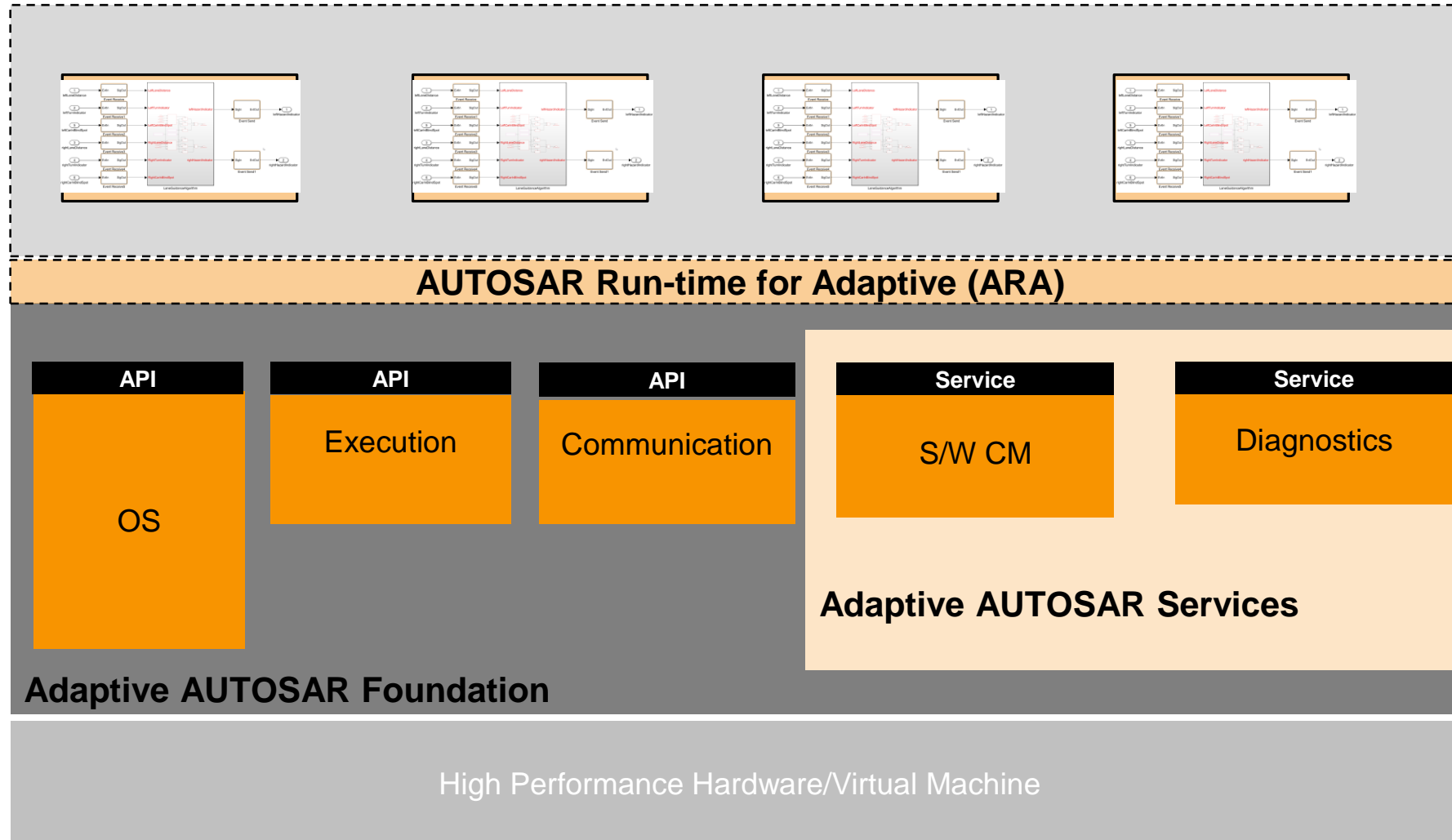
# AUTOSAR Today



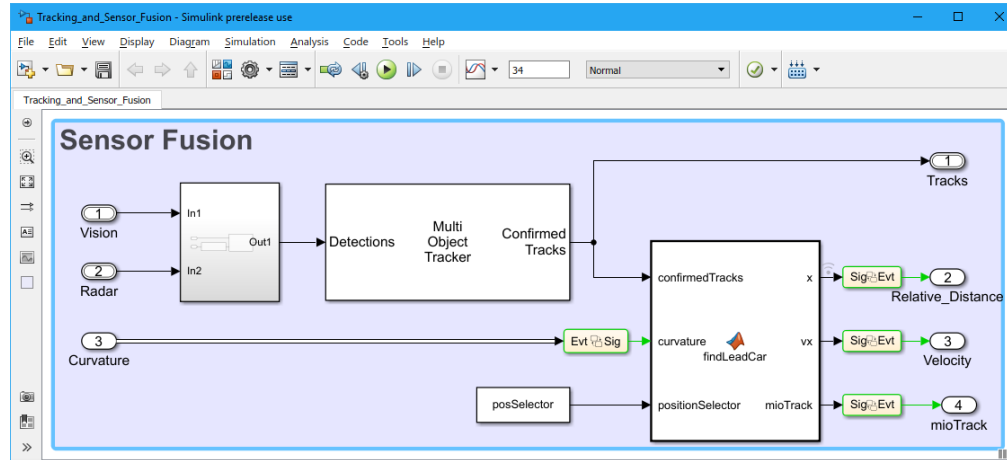
- N Non - AUTOSAR
- C Classic - AUTOSAR
- A Adaptive - AUTOSAR



# Support for AUTOSAR Adaptive Platform



# Generate Production AUTOSAR Adaptive C++ Code



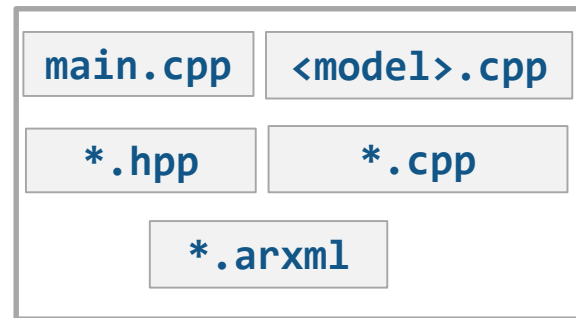
**Configuration Parameters:**

Target selection

System target file:

Language:

Description:



## Out-of-box AUTOSAR support

1. Configure Model
  - ✓ Target: autosar\_adaptive.tlc
  - ✓ AUTOSAR Dictionary
2. Generate C++ code

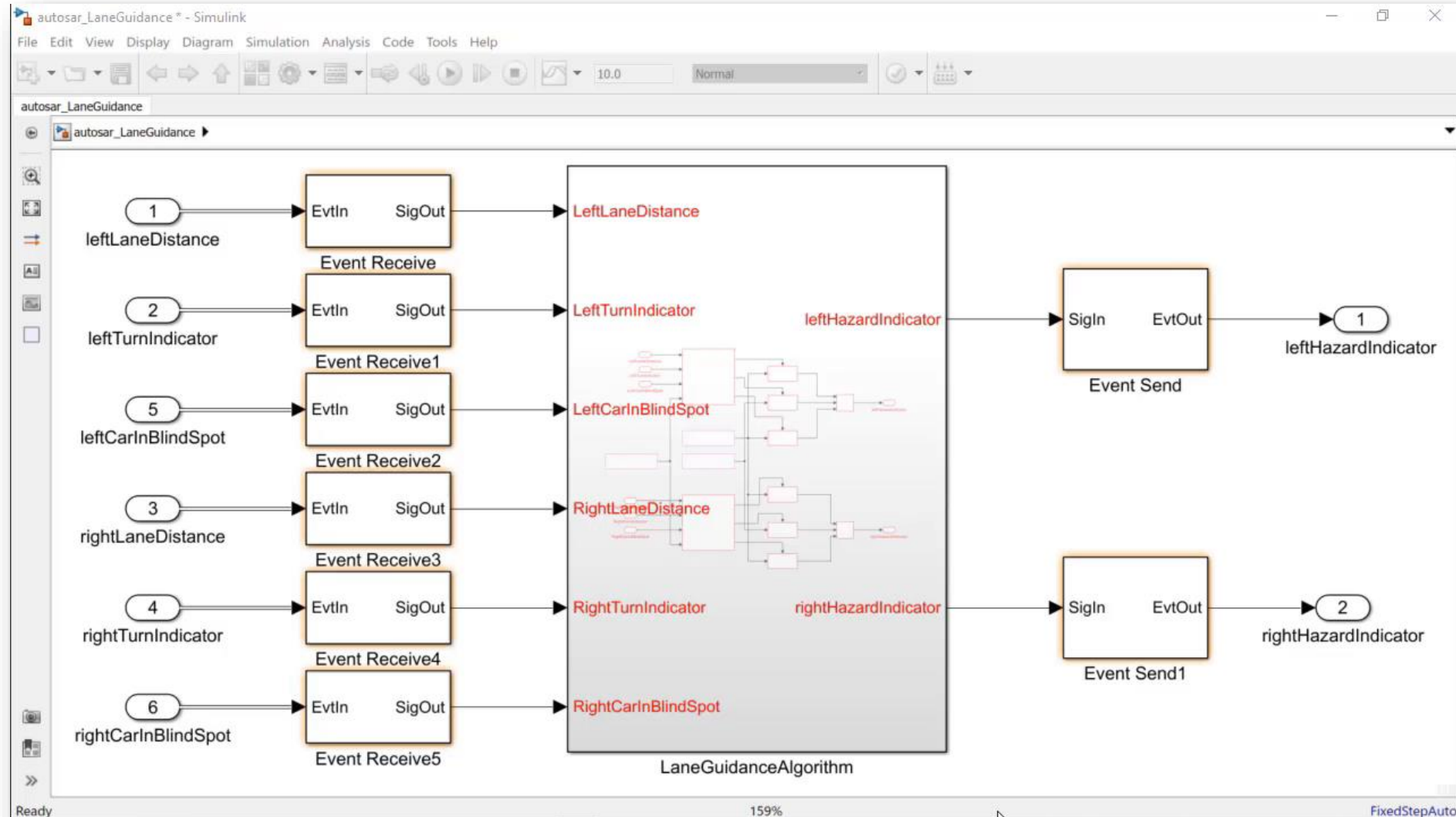
**AUTOSAR Dictionary**

Filter Contents

- ▼ AUTOSAR
  - ▼ AdaptiveApplications
    - ▼ Tracking\_and\_Sensor\_Fusion
      - RequiredPorts
      - ProvidedPorts
    - Service Interfaces
      - ▼ RadarInterface
        - Methods
        - Events
        - Fields
        - Namespaces
      - ServiceInterface2
    - XML Options

Name	SwCalibrationAccess
Curvature	ReadOnly
Prediction_Time	ReadOnly
Radar	ReadOnly
Vision	ReadOnly

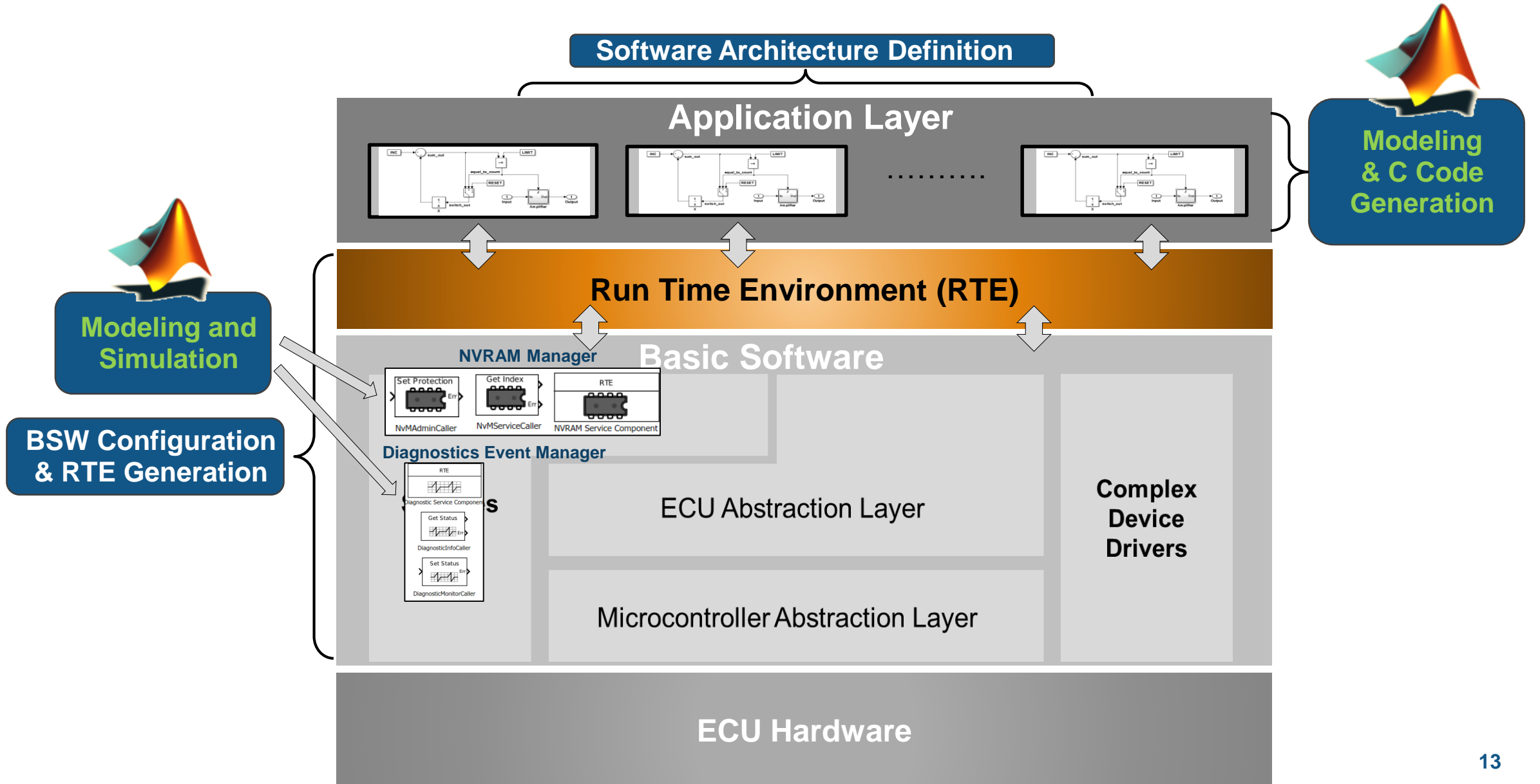
# Demo: Model Adaptive Software Components



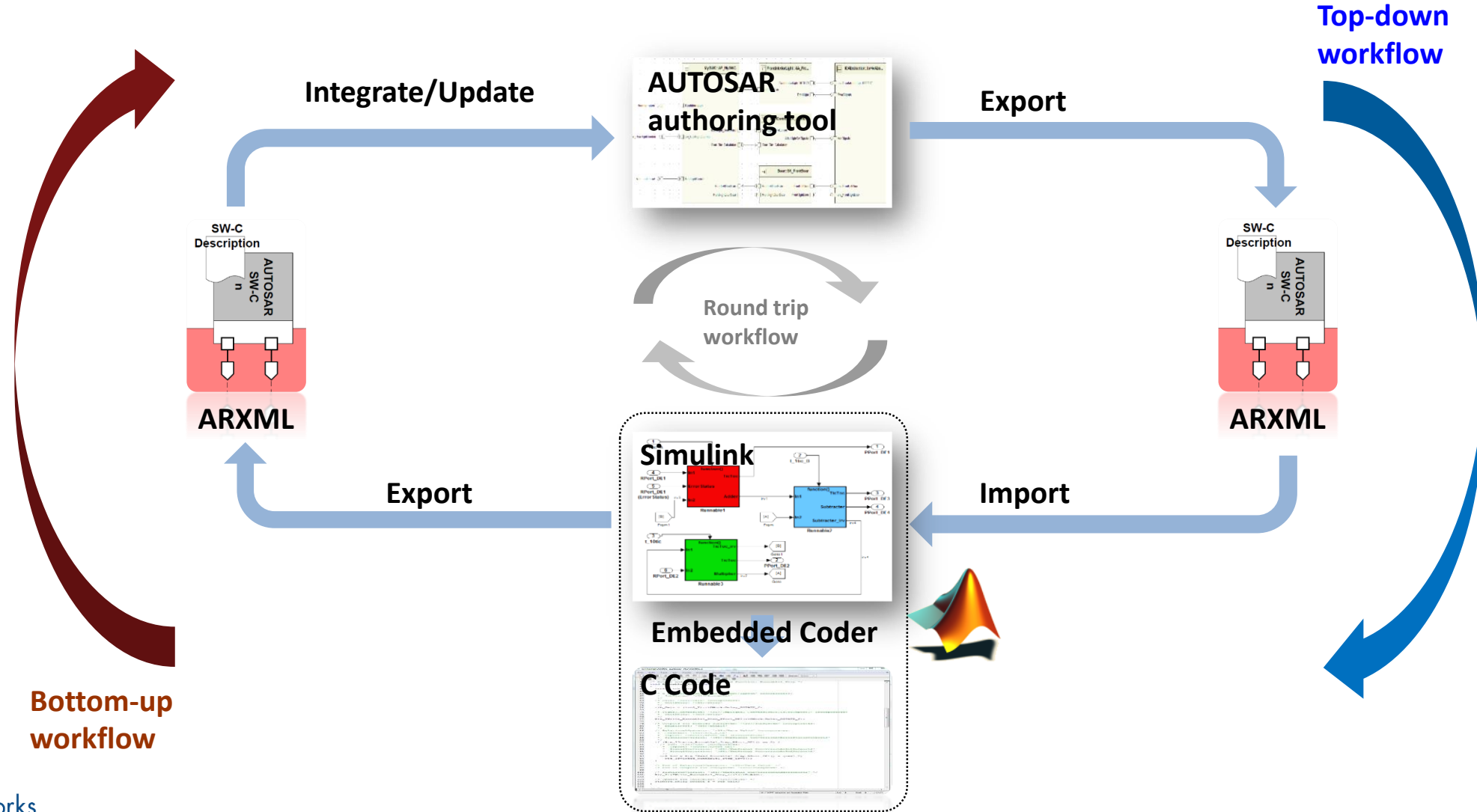
# Agenda

- AUTOSAR Blockset Introduction
  - Adaptive Platform
  - Classic Platform
  
- **AUTOSAR ASW Development**
  - **AUTOSAR ASW architecture design**
  - Testing in AUTOSAR Composition Editor
  
- System Composer with AUTOSAR Blockset

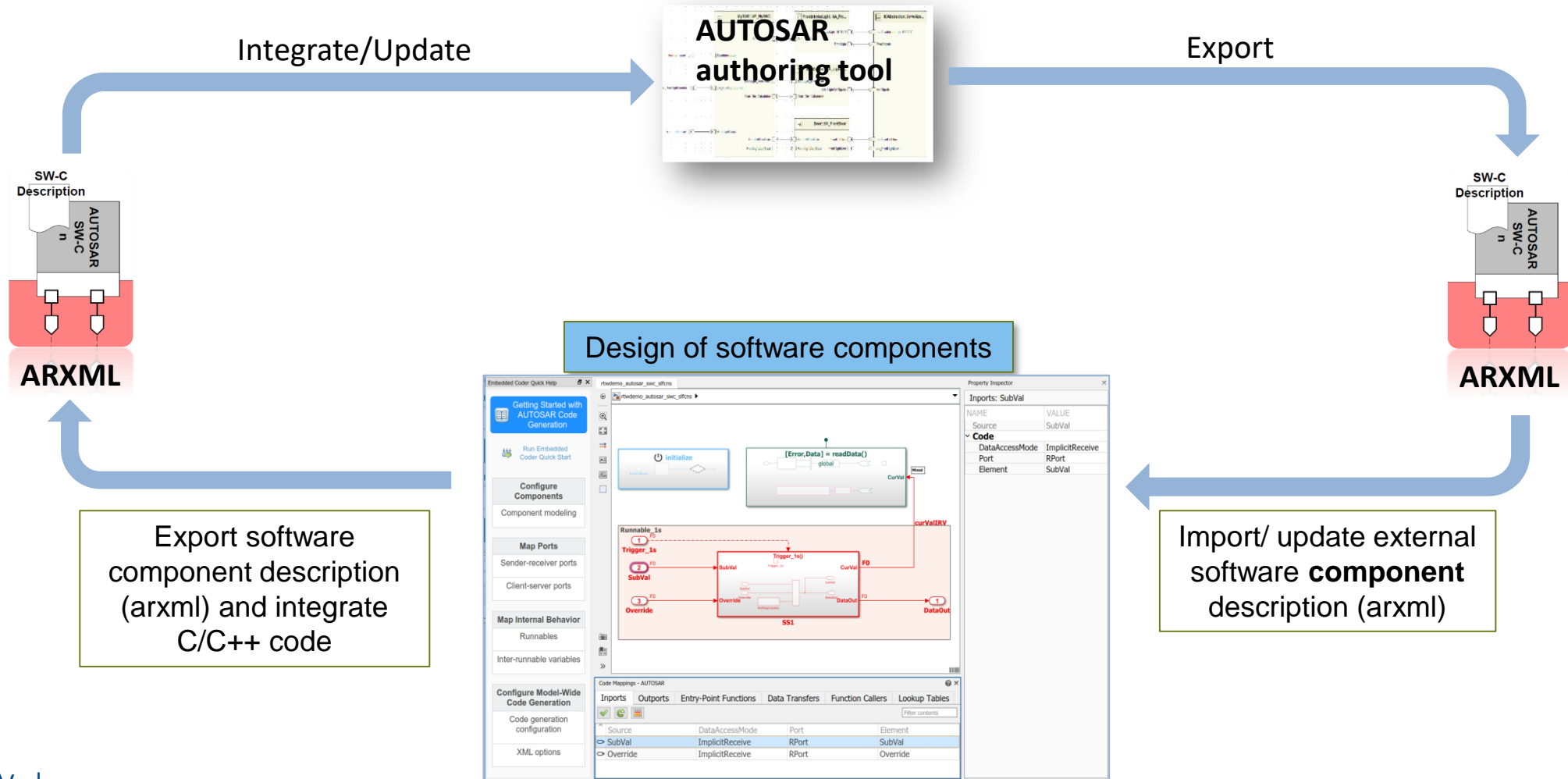
# AUTOSAR Blockset and Embedded Coder for Classic Platform



# Supported AUTOSAR Design Workflows

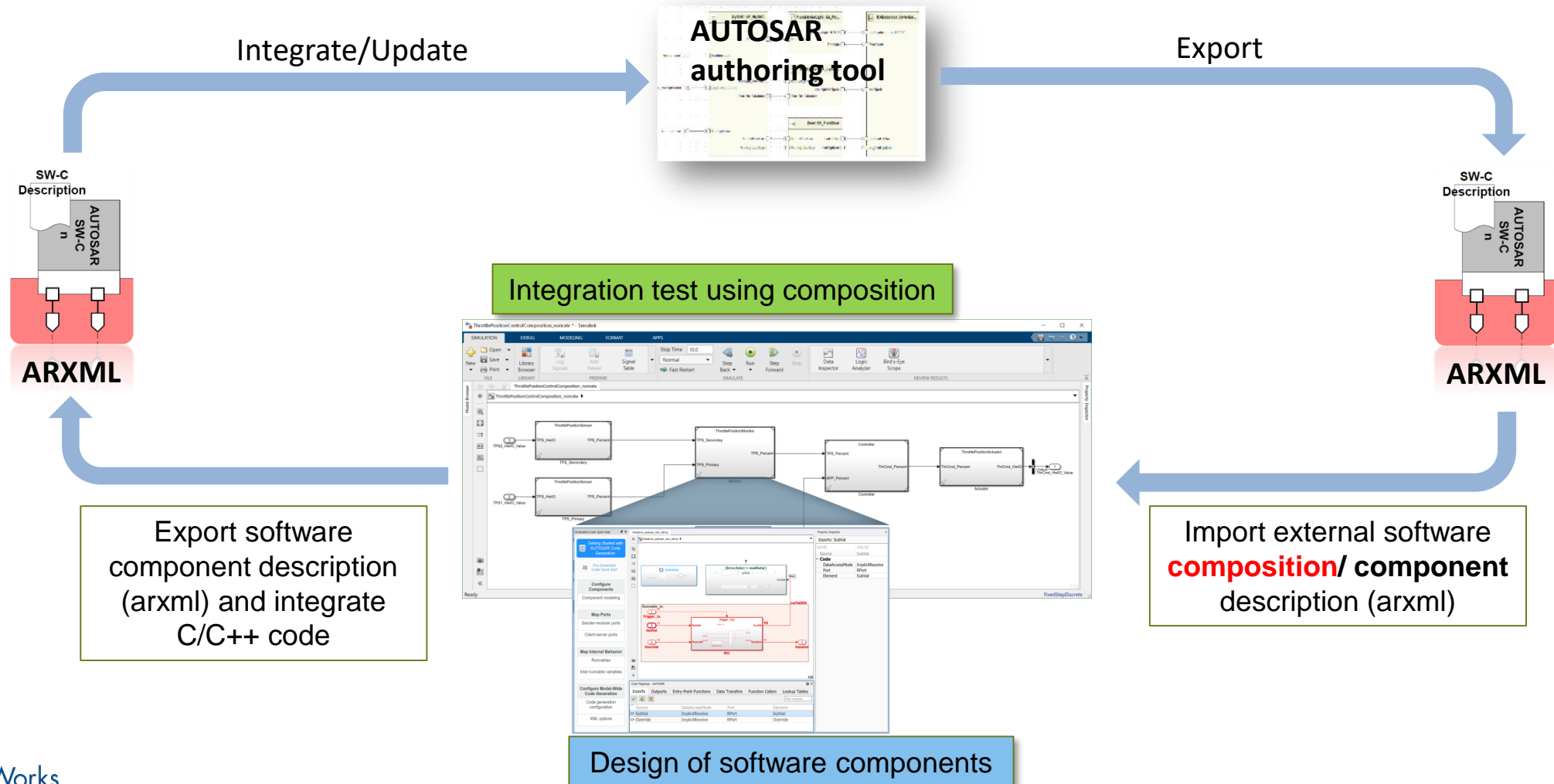


# Remind S/W Component Workflows



# AUTOSAR Software Architecture

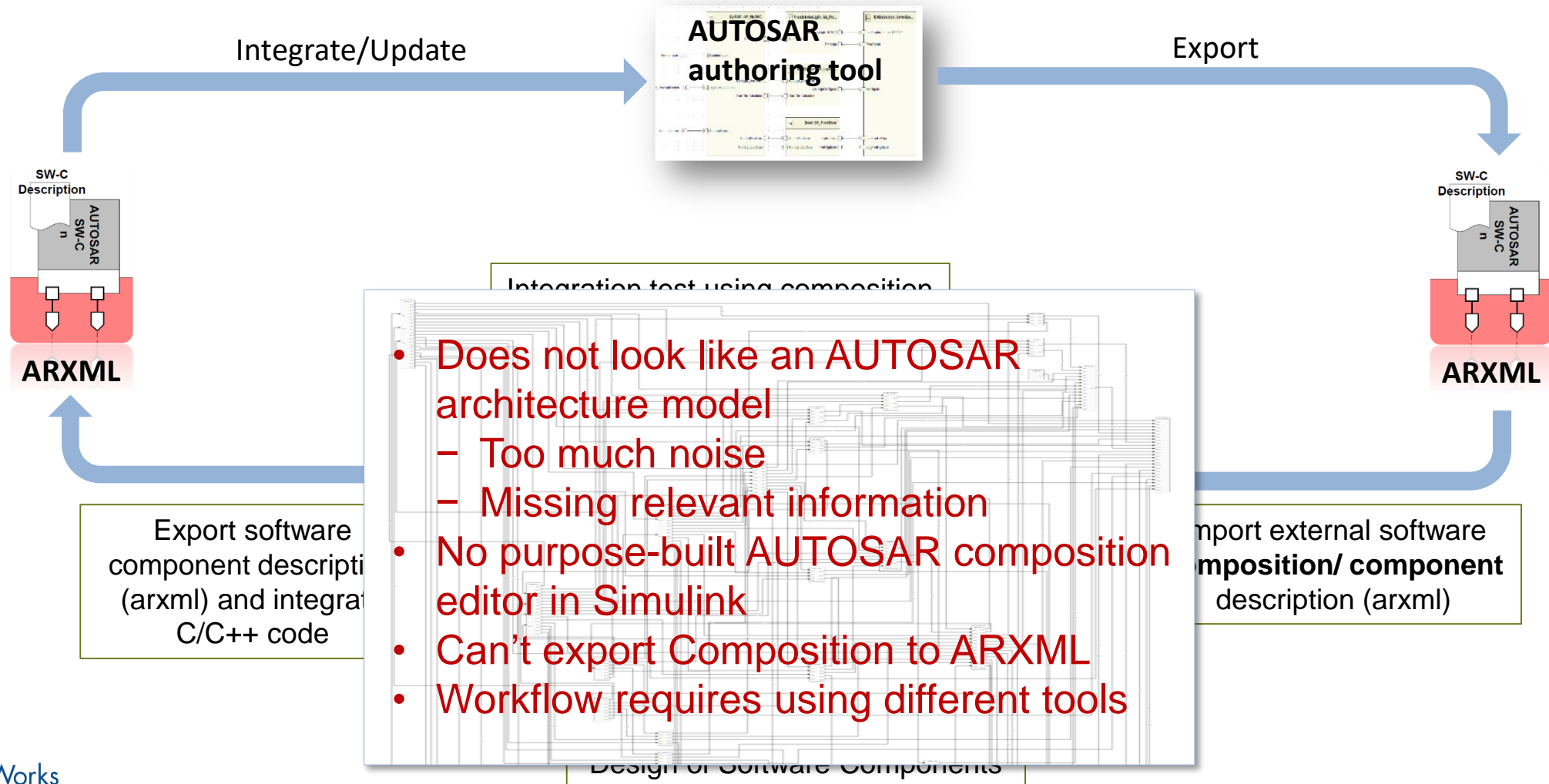
## Success so far



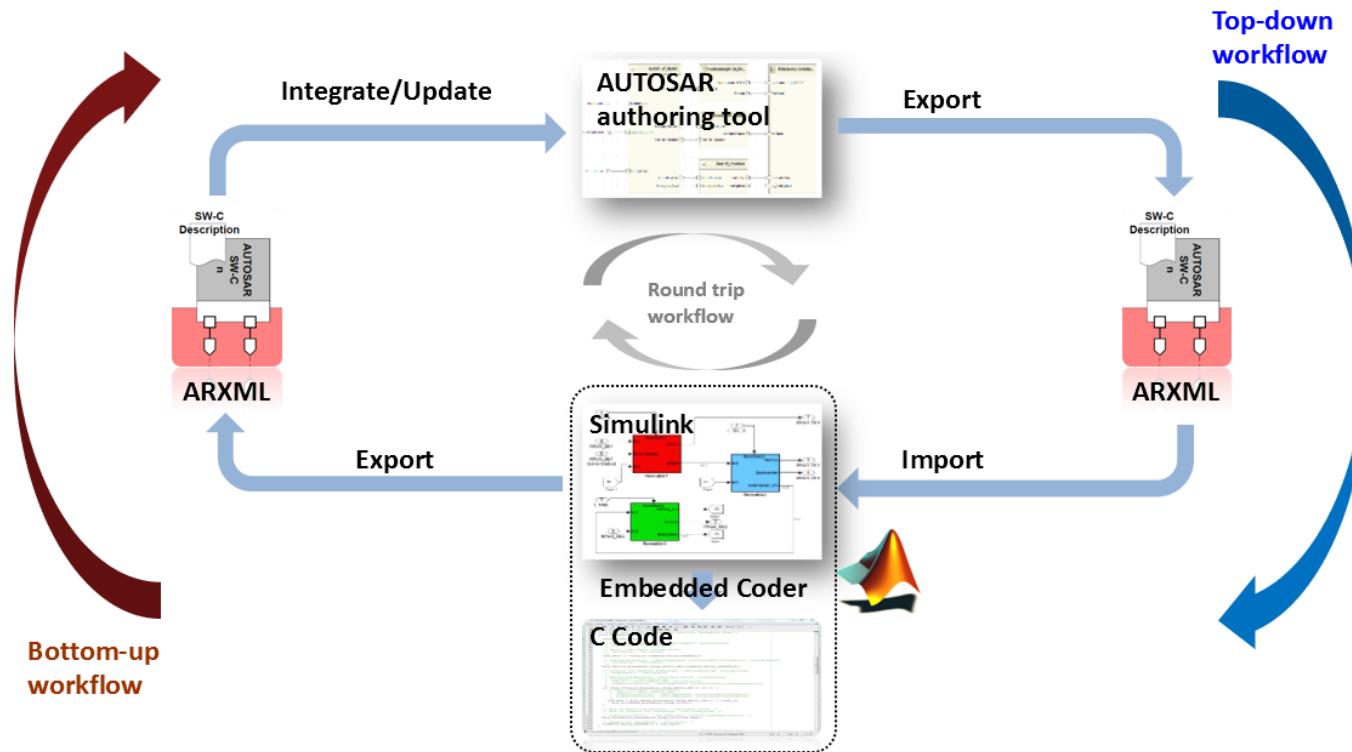


# AUTOSAR Software Architecture

## Success so far?

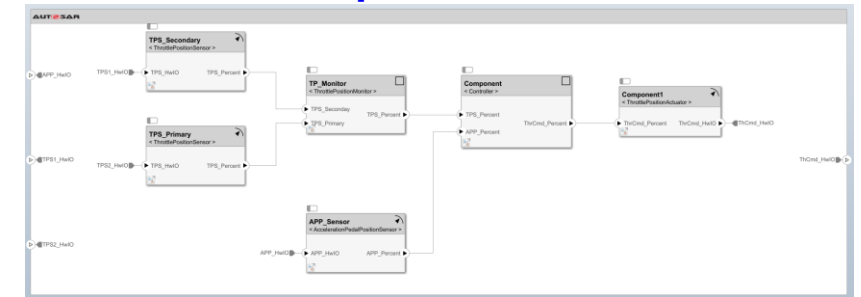


# Now, You can start from AUTOSAR architecture design!



+  $\alpha$

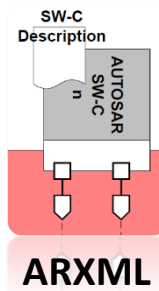
## AUTOSAR Composition Editor



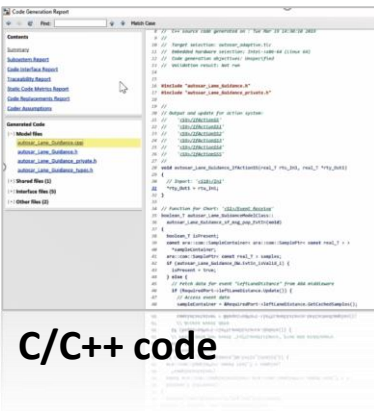
# AUTOSAR Software Architecture

## Today's Focus - Make it easier

INTEGRATE INTO PRODUCTION

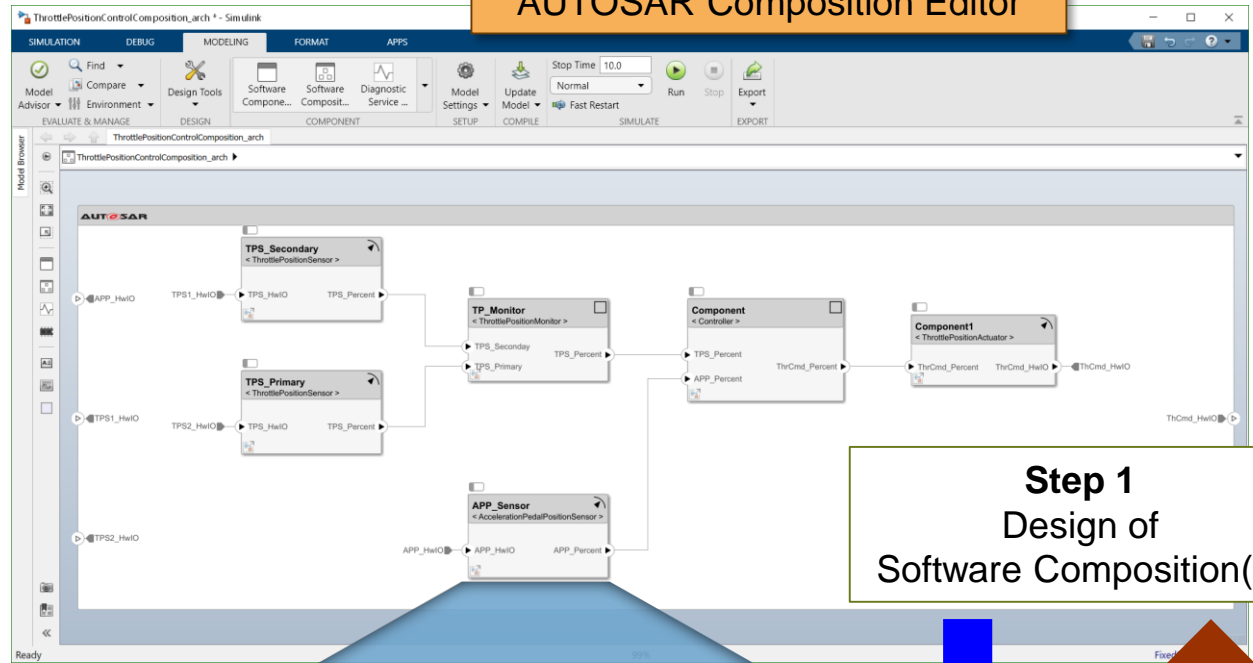


**Step 3**  
Integrate software description (arxml) and C/C++ code

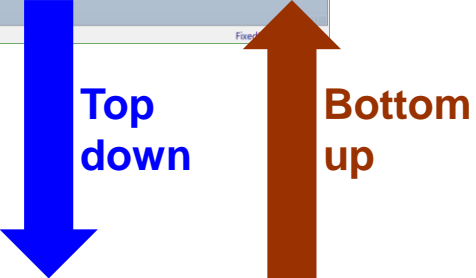


C/C++ code

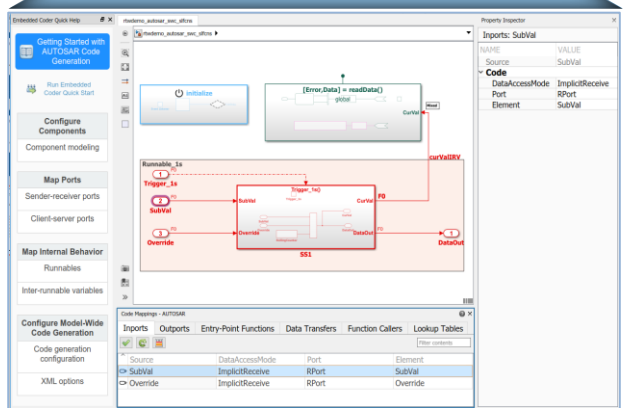
AUTOSAR Composition Editor



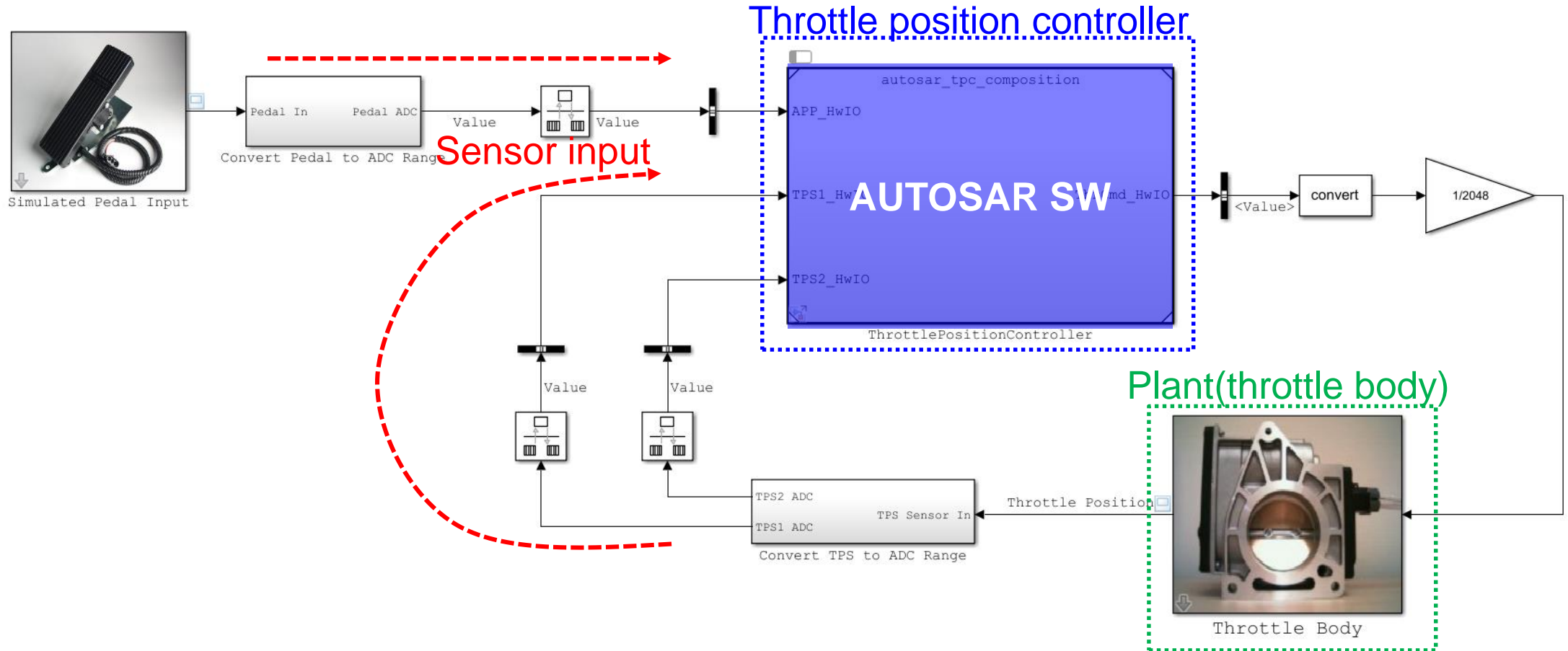
**Step 1**  
Design of Software Composition(s)



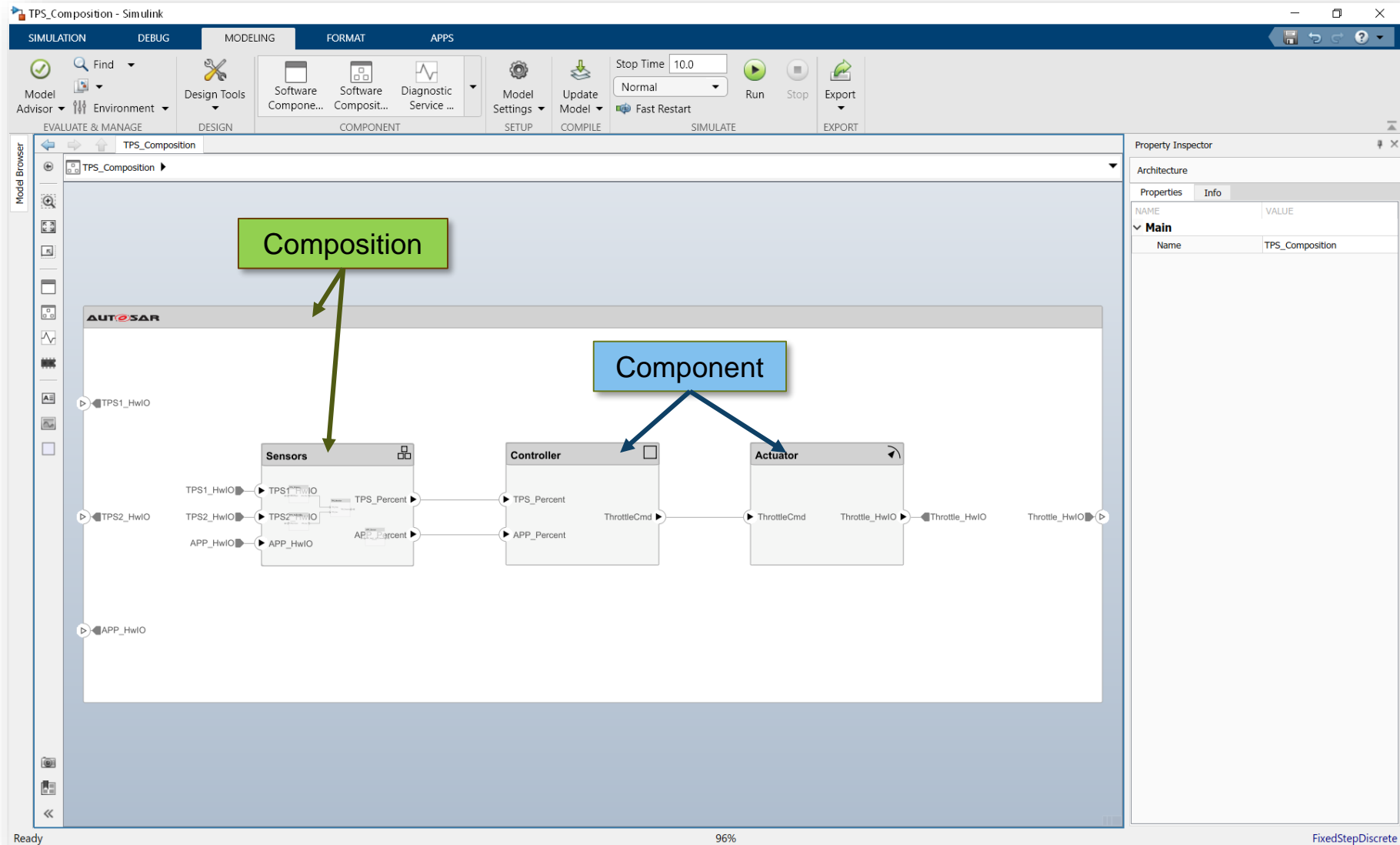
**Step 2**  
Design of Software Components & AUTOSAR configuration



# Example: Automotive Throttle Body



# Create AUTOSAR ASW Composition



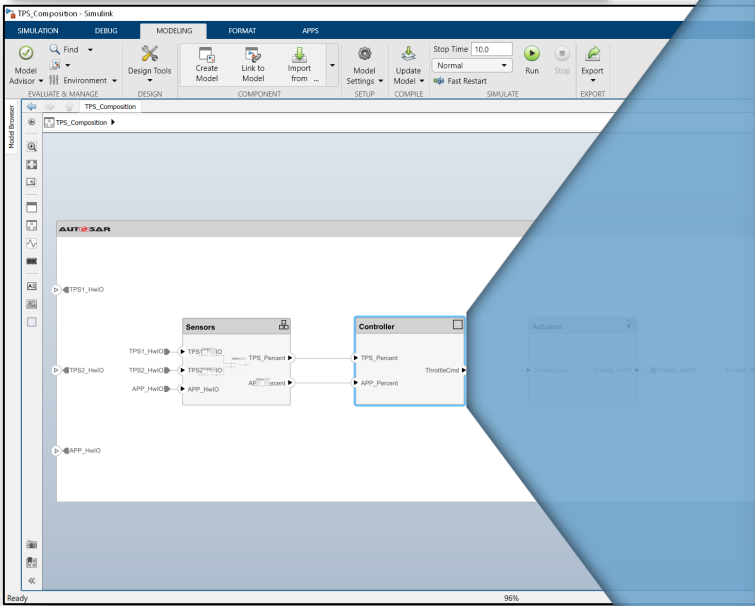
# Create Models from AUTOSAR ASW Component

The screenshot shows the Simulink environment with the following elements:

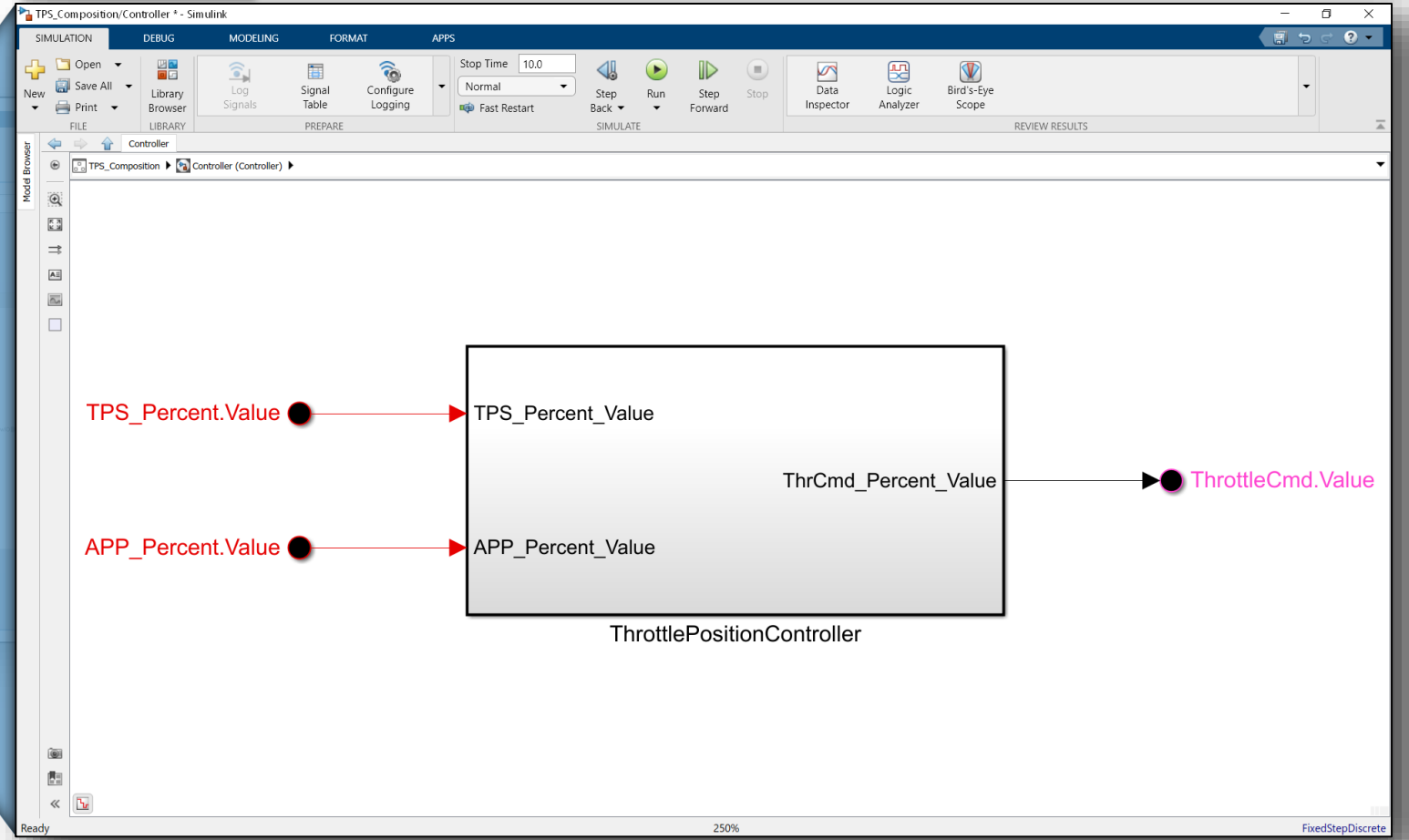
- Toolbar:** Includes buttons for Model Advisor, Design Tools, Create Model, Link to Model, Import from..., Model Settings, Update Model, Stop Time (10.0), Normal, Run, Stop, Export, and Fast Restart.
- Model Browser:** Shows the current model structure with 'TPS\_Composition' selected.
- Diagram:** An AUTOSAR diagram with a 'Sensors' block and a 'Controller' block. The 'Controller' block is selected, and a red arrow points to it. A context menu is open over the 'Controller' block, listing options: Cut (Ctrl+X), Copy (Ctrl+C), Paste (Ctrl+V), Delete (Del), **Create Model...** (highlighted), Link to Model..., Create Component Model from ARXML..., Arrange, Signals & Ports, Requirements, and Help.
- Dialog Box:** A 'Create Simulink Model' dialog box is open, containing the text: 'Create a Simulink model for this software component, and link the component to the created model.' Below this is a text field labeled 'Specify model name:' with the value 'Controller' entered. At the bottom are 'OK' and 'Cancel' buttons.

# Detailed Design using Simulink

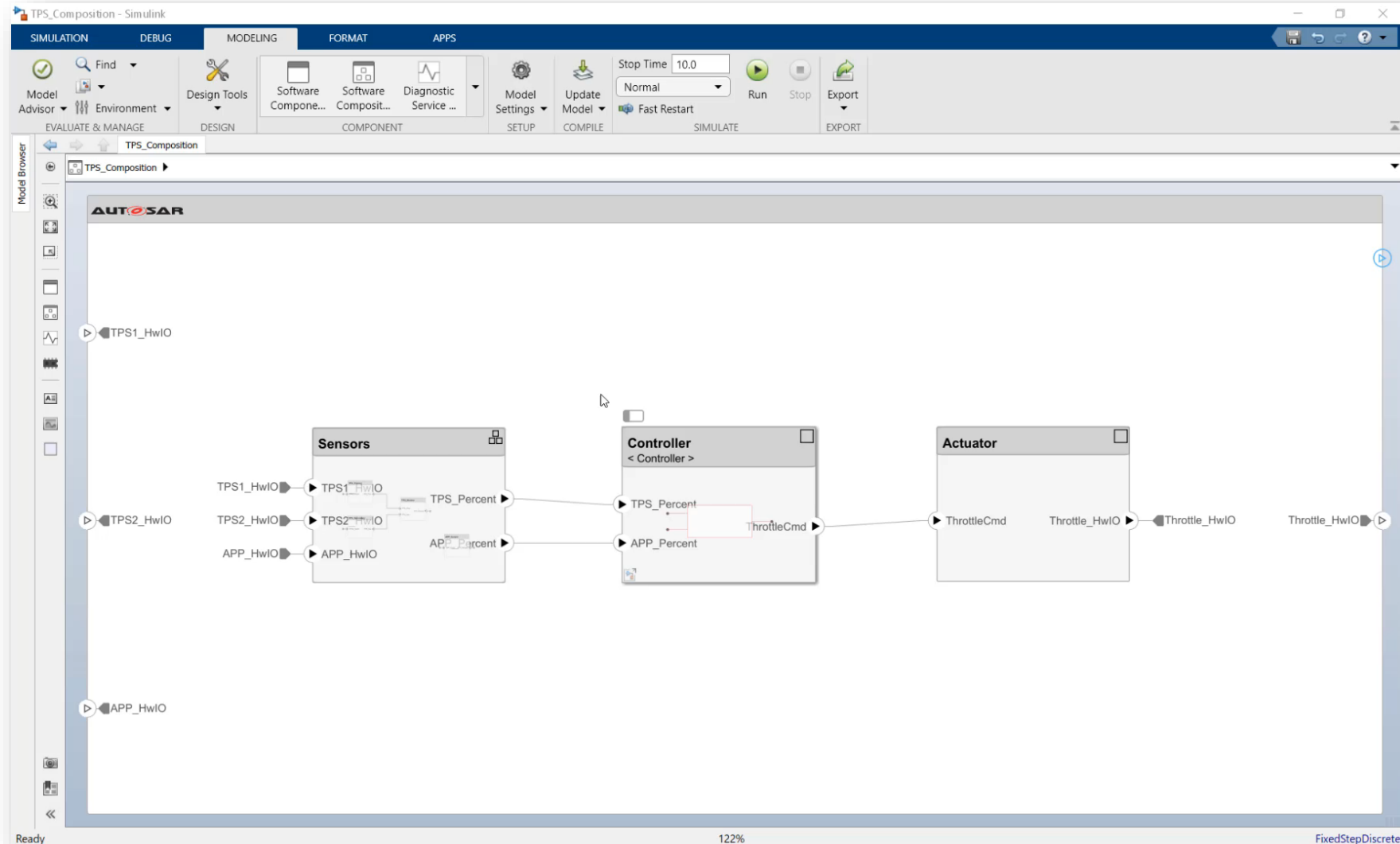
**AUTOSAR Composition Editor**



**Simulink**



# Design and Configure for AUTOSAR SW

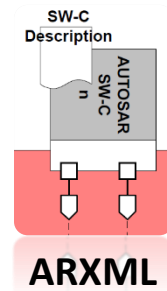




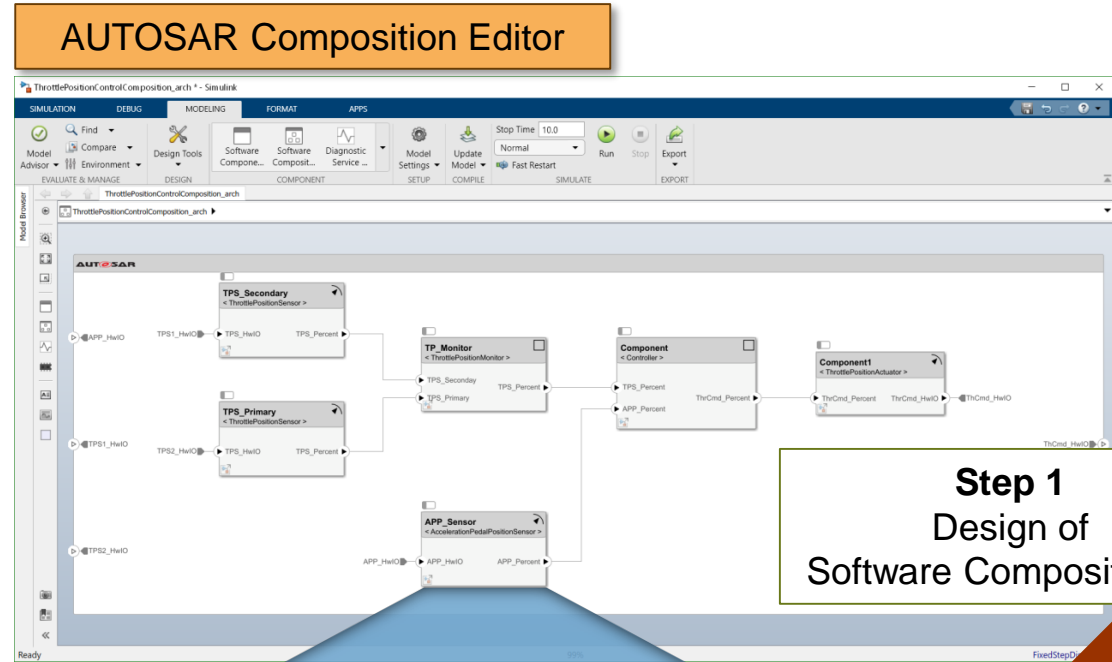
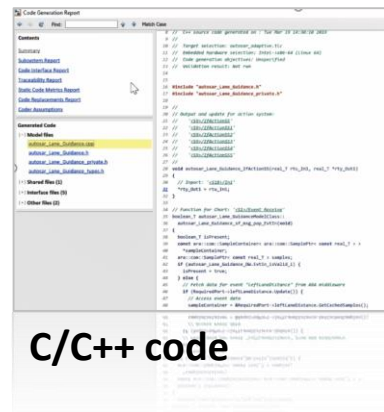
# AUTOSAR Software Architecture

## Bottom-up Workflow

INTEGRATE INTO PRODUCTION



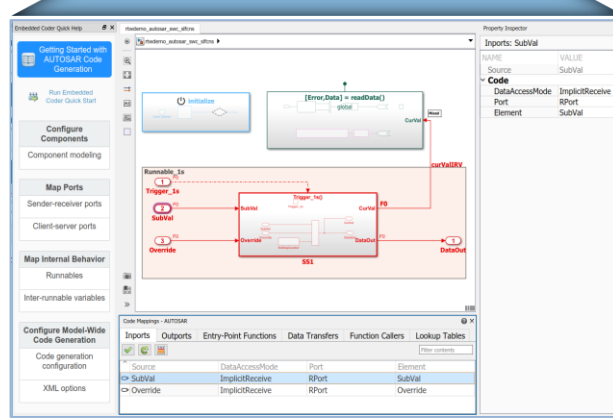
**Step 3**  
Integrate software description (arxml) and C/C++ code



**Step 1**  
Design of Software Composition(s)

**Bottom up**

**Step 2**  
Design of Software Components & AUTOSAR configuration



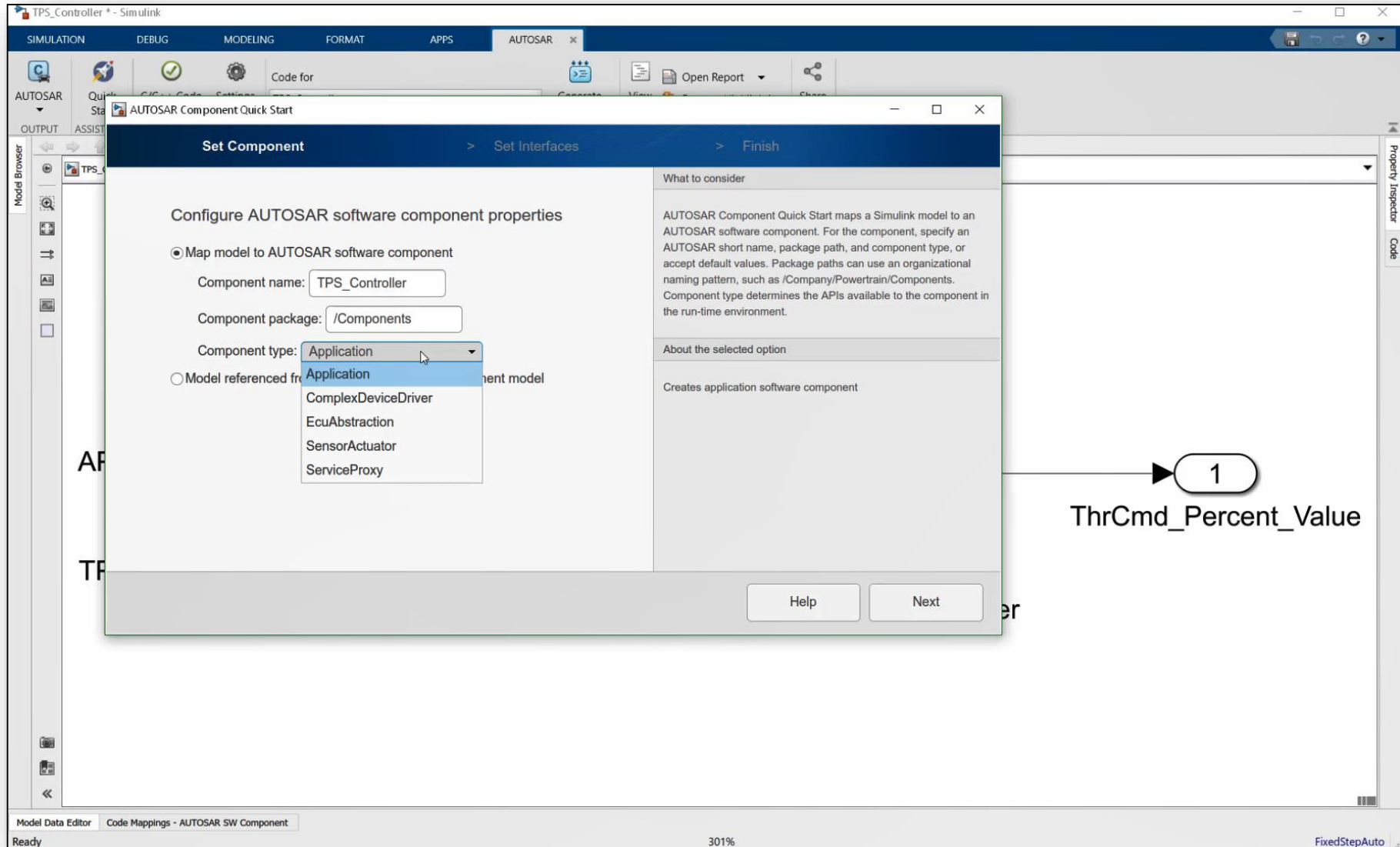
# Launch Code Perspective for AUTOSAR Configuration

The screenshot displays the Simulink environment for a project named 'TPS\_Controller'. The 'MODELING' tab is active, and the 'Model Settings' icon in the top toolbar is highlighted with a red box. A dialog box titled 'Configuration Parameters: TPS\_Controller/Configuration (Active)' is open, showing the 'Code Generation' section. Within this dialog, the 'Target selection' section is highlighted with a red box, containing the following fields:

- System target file: `autosar.tlc` (with a 'Browse...' button)
- Language: `C` (dropdown menu)
- Description: `AUTOSAR`

Below the dialog, a Simulink block diagram is visible, featuring a 'Discrete PID Controller' block. Two input signals, 'APP\_Percent\_Value' (labeled '2') and 'TPS\_Percent\_Value' (labeled '1'), are connected to the controller's inputs. The controller's output is labeled 'error'. At the bottom of the interface, a perspective selection bar shows three options: 'Code' (highlighted with a red box), 'Requirements', and 'Interface'.

# AUTOSAR Component Quick Start



# Simulink to AUTOSAR Configuration

The screenshot displays the Simulink AUTOSAR configuration environment. The main workspace shows a Simulink model with two input blocks, 'APP\_Percent\_Value' (labeled '2') and 'TPS\_Percent\_Value' (labeled '1'), feeding into a summing junction. The output of the summing junction is labeled 'error' and is connected to a 'Discrete PID Cont' block.

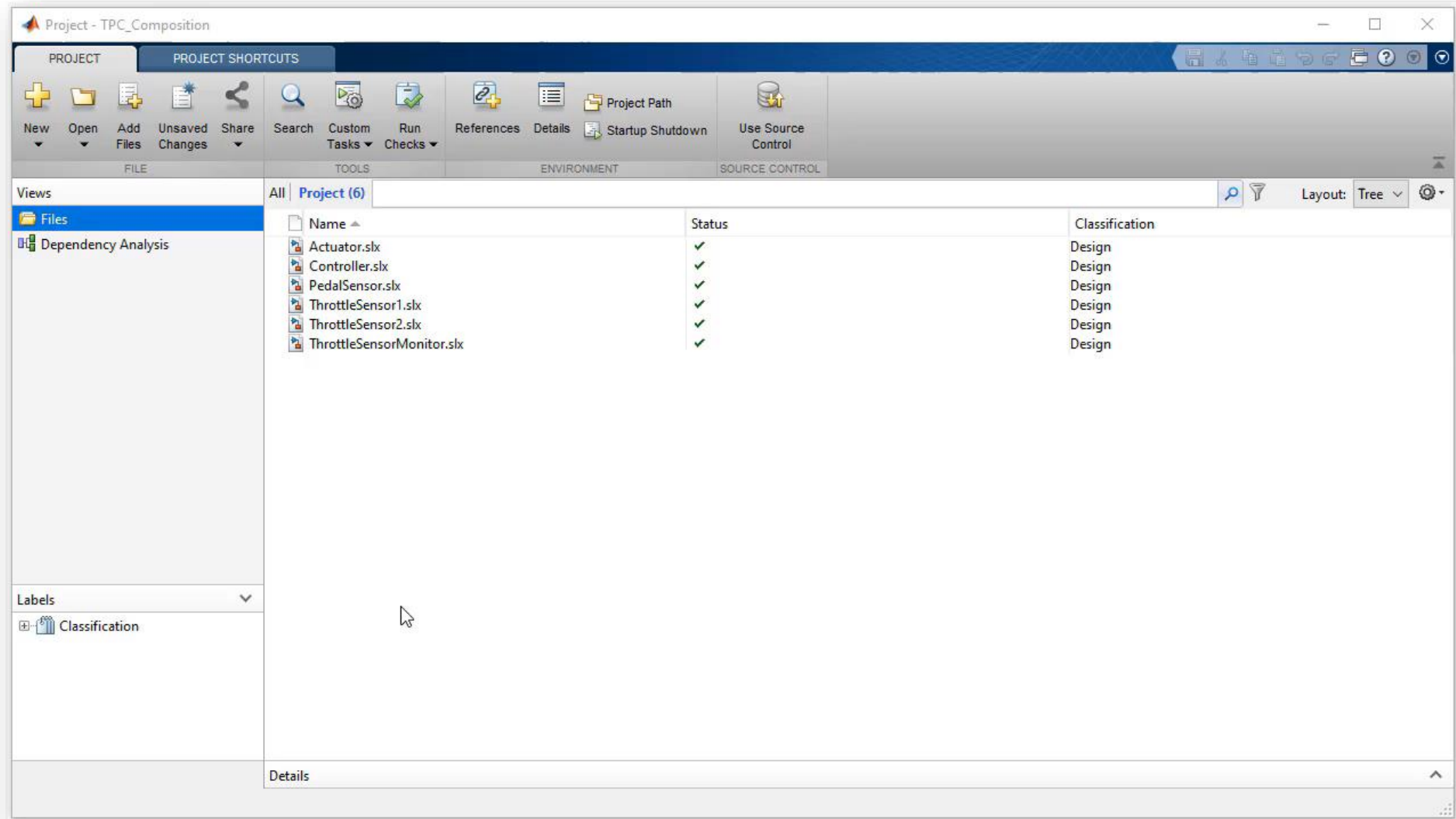
On the right side, the 'AUTOSAR Dictionary: EtcComposition' window is open, showing a tree view of AUTOSAR components. A table titled 'AUTOSAR Dictionary' lists the following interfaces:

Name	Interface
APP_HwIO_Value	APP_HwIO_Value
TPS1_HwIO_Value	TPS1_HwIO_Value
TPS2_HwIO_Value	TPS2_HwIO_Value

At the bottom, the 'Code Mappings - AUTOSAR SW Component' window is open, showing the 'Simulink-AUTOSAR Mapping Editor'. It lists the following mappings:

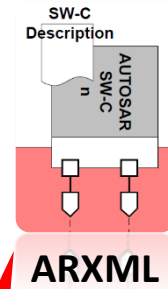
Functions	Inports	Outports	Parameters	Data Stores	Signals/States	Data Transfers	Function Callers
Initialize Function							TPS_Controller_Init
Step Function [Sample Time:0.2s]							TPS_Controller_Step

# Assemble AUTOSAR SW-C Models into a Composition

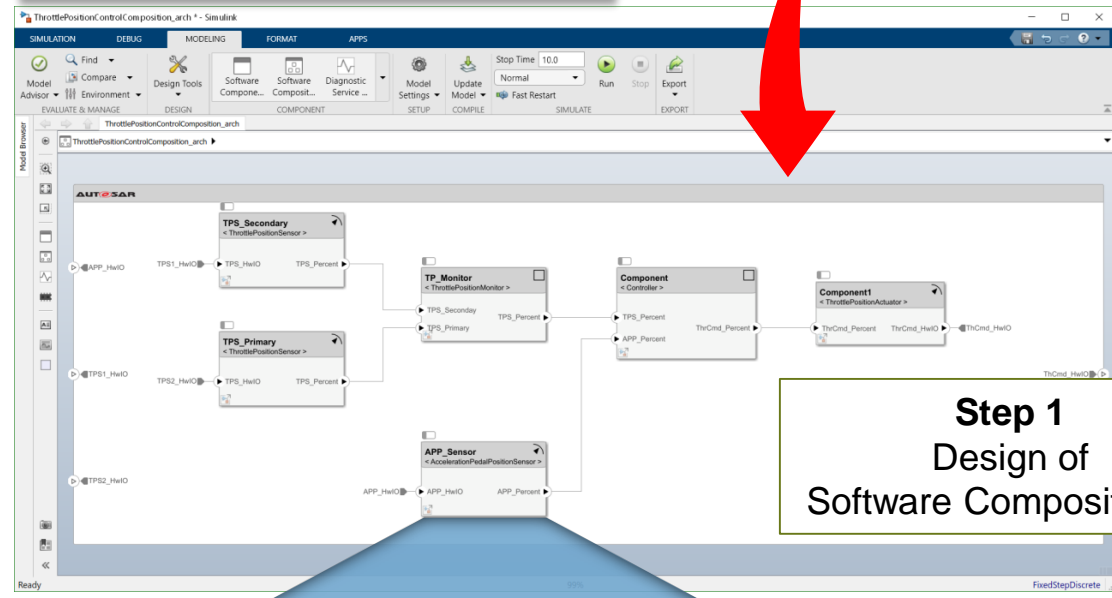


# AUTOSAR Software Architecture

## Import ARXML

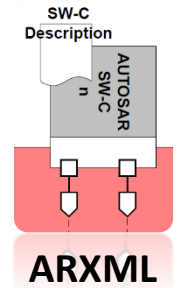


### AUTOSAR Composition Editor

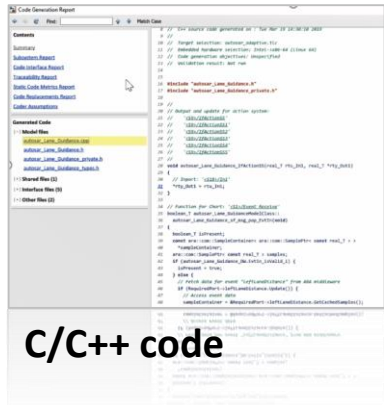


**Step 1**  
Design of Software Composition(s)

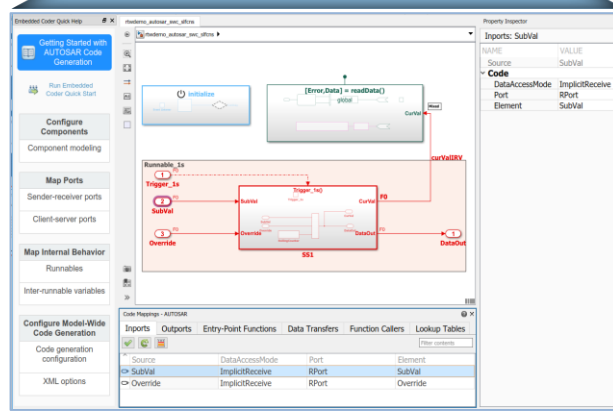
**Step 3**  
Integrate software description (arxml) and C/C++ code



INTEGRATE INTO PRODUCTION

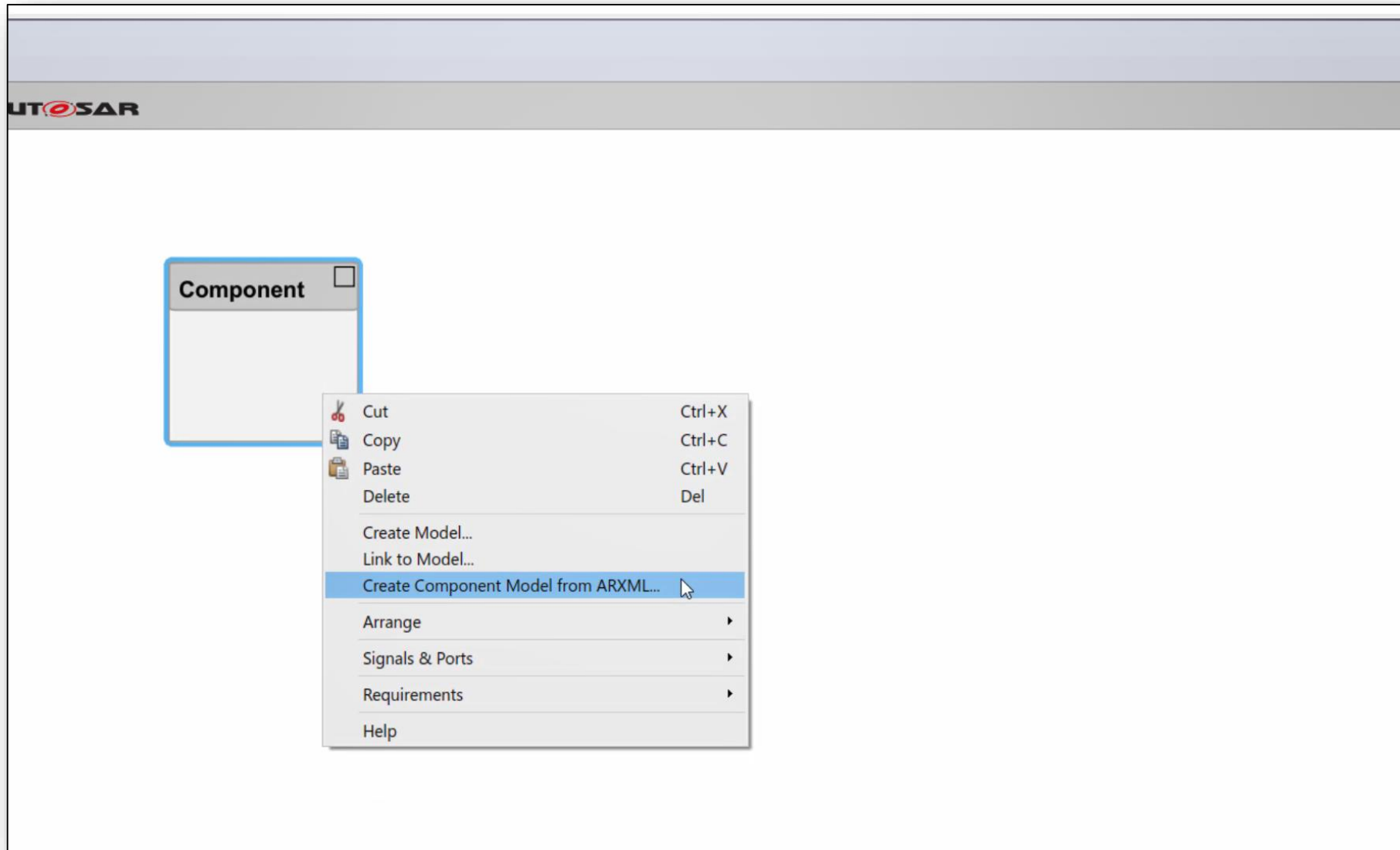


C/C++ code



**Step 2**  
Design of Software Components & AUTOSAR configuration

# AUTOSAR Architecture from ARXML



# Agenda

- AUTOSAR Blockset Introduction
  - Adaptive Platform
  - Classic Platform
- AUTOSAR ASW Development
  - AUTOSAR ASW architecture design
  - **Testing in AUTOSAR Composition Editor**
- System Composer with AUTOSAR Blockset



# Simulate with BSW Service Blocks and Schedule Editor

**EXECUTION ORDER**

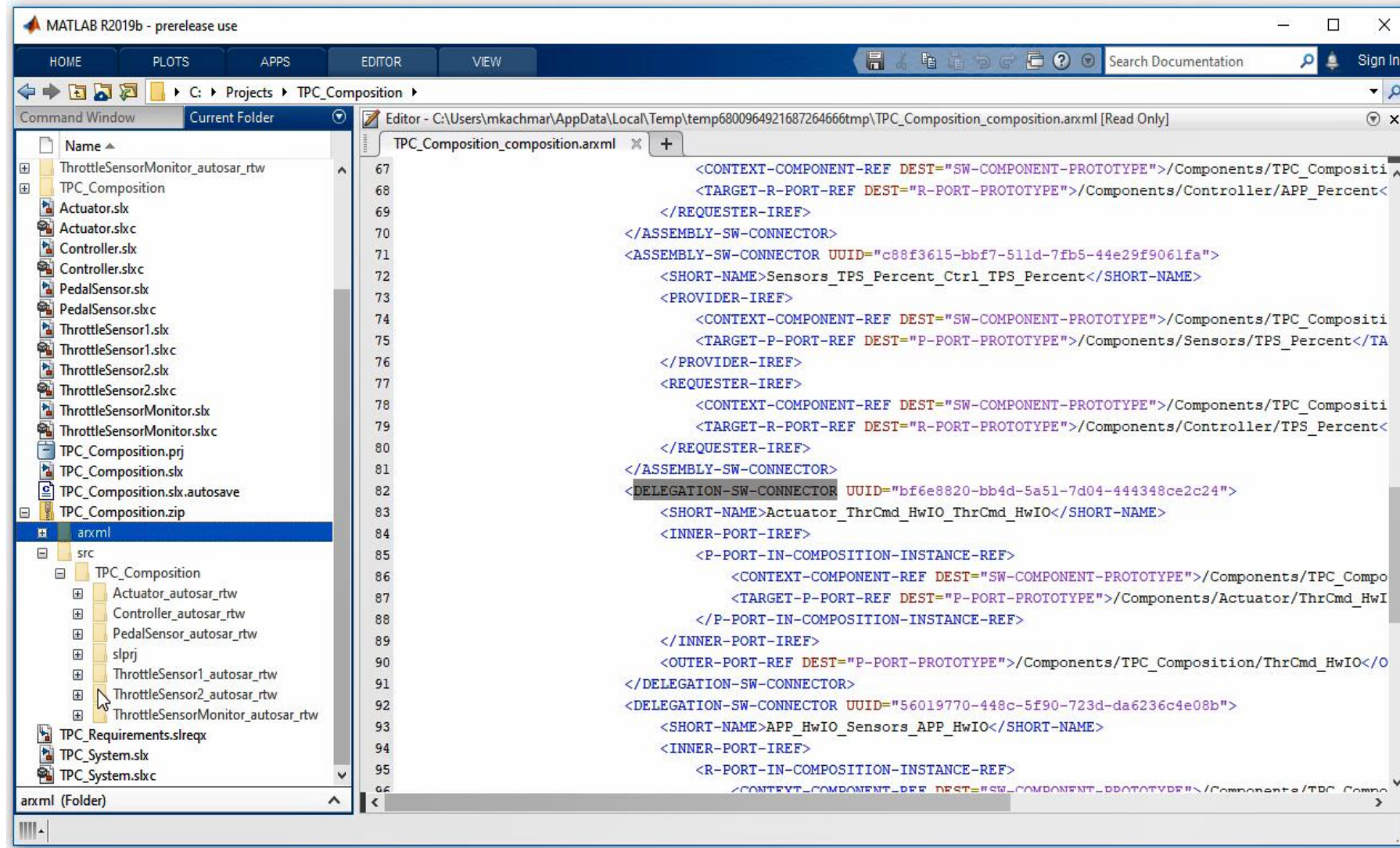
Order	Name	Rate
1	Cont implicit	0
2	D1 implicit	0.005
3	Model.D1	0.005
4	lSensor.D1	0.005
5	_Primary.D1	0.005
6	secondary.D1	0.005
7	l.Monitor.D1	0.005
8	Model.Ctrl.D1	0.005
	Actuator.D1	0.005

**PROPERTY INSPECTOR**

Partition

Name	Model.TPS_Primary.D1
Rate	0.005
Type	Explicit periodic partition

# Generate AUTOSAR Code and Export ARXML



MATLAB R2019b - prerelease use

HOME PLOTS APPS EDITOR VIEW

Search Documentation Sign In

C:\Projects\TPC\_Composition

Command Window Current Folder

Editor - C:\Users\rkachmar\AppData\Local\Temp\temp6800964921687264666tmp\TPC\_Composition\_composition.arxml [Read Only]

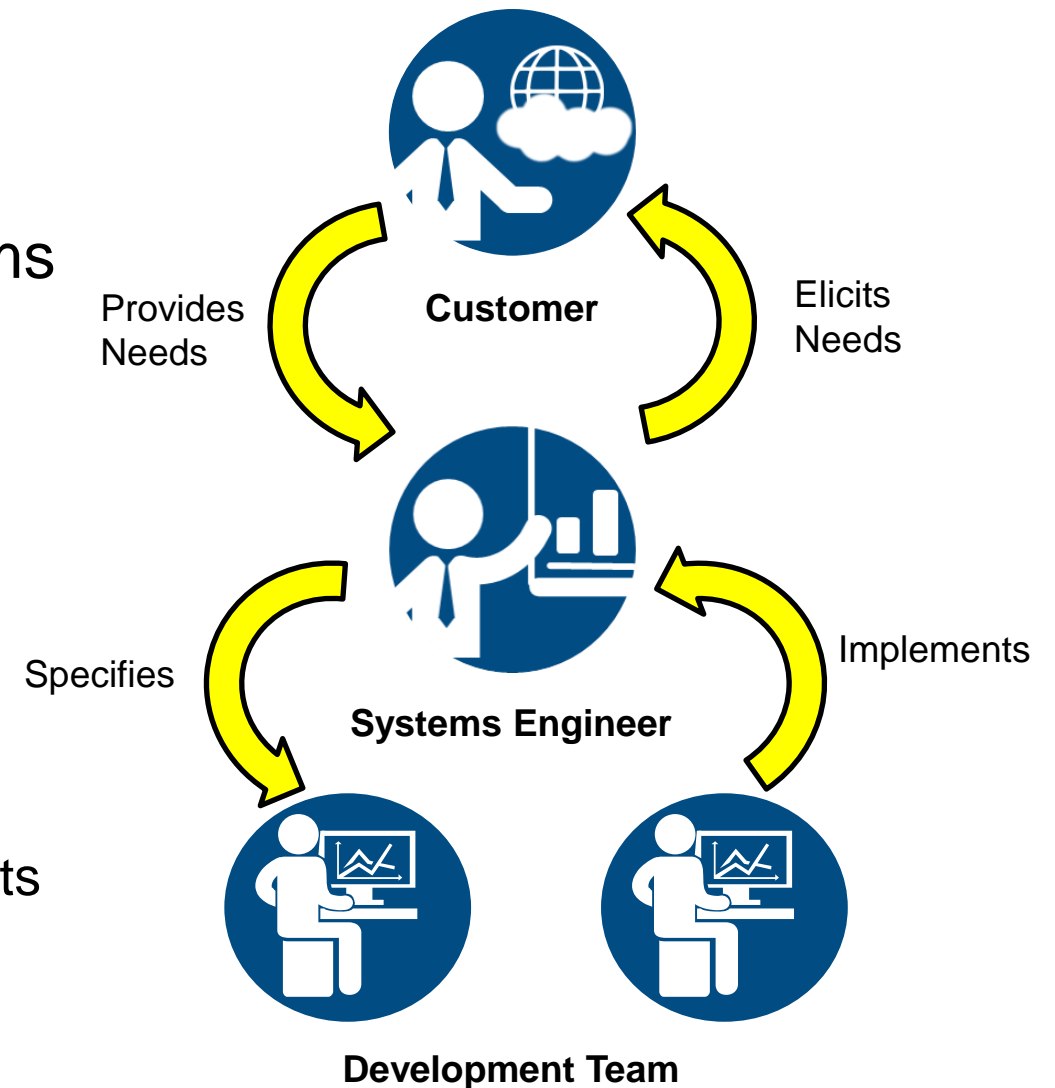
```
67 <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>/Components/TPC_Compositi
68 <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE"/>/Components/Controller/APP_Percent<
69 </REQUESTER-IREF>
70 </ASSEMBLY-SW-CONNECTOR>
71 <ASSEMBLY-SW-CONNECTOR UUID="c88f3615-bbf7-511d-7fb5-44e29f9061fa">
72 <SHORT-NAME>Sensors_TPS_Percent_Ctrl_TPS_Percent</SHORT-NAME>
73 <PROVIDER-IREF>
74 <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>/Components/TPC_Compositi
75 <TARGET-P-PORT-REF DEST="P-PORT-PROTOTYPE"/>/Components/Sensors/TPS_Percent</TA
76 </PROVIDER-IREF>
77 <REQUESTER-IREF>
78 <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>/Components/TPC_Compositi
79 <TARGET-R-PORT-REF DEST="R-PORT-PROTOTYPE"/>/Components/Controller/TPS_Percent<
80 </REQUESTER-IREF>
81 </ASSEMBLY-SW-CONNECTOR>
82 <DELEGATION-SW-CONNECTOR UUID="bf6e8820-bb4d-5a51-7d04-444348ce2c24">
83 <SHORT-NAME>Actuator_ThrCmd_HwIO_ThrCmd_HwIO</SHORT-NAME>
84 <INNER-PORT-IREF>
85 <P-PORT-IN-COMPOSITION-INSTANCE-REF>
86 <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>/Components/TPC_Comp
87 <TARGET-P-PORT-REF DEST="P-PORT-PROTOTYPE"/>/Components/Actuator/ThrCmd_HwI
88 </P-PORT-IN-COMPOSITION-INSTANCE-REF>
89 </INNER-PORT-IREF>
90 <OUTER-PORT-REF DEST="P-PORT-PROTOTYPE"/>/Components/TPC_Composition/ThrCmd_HwIO</O
91 </DELEGATION-SW-CONNECTOR>
92 <DELEGATION-SW-CONNECTOR UUID="56019770-448c-5f90-723d-da6236c4e08b">
93 <SHORT-NAME>APP_HwIO_Sensors_APP_HwIO</SHORT-NAME>
94 <INNER-PORT-IREF>
95 <R-PORT-IN-COMPOSITION-INSTANCE-REF>
96 <CONTEXT-COMPONENT-REF DEST="SW-COMPONENT-PROTOTYPE"/>/Components/TPC_Comp
```

# Agenda

- AUTOSAR Blockset Introduction
  - Adaptive Platform
  - Classic Platform
  
- AUTOSAR ASW Development
  - AUTOSAR ASW architecture design
  - Testing in AUTOSAR Composition Editor
  
- **System Composer with AUTOSAR Blockset**

# What is Systems Engineering?

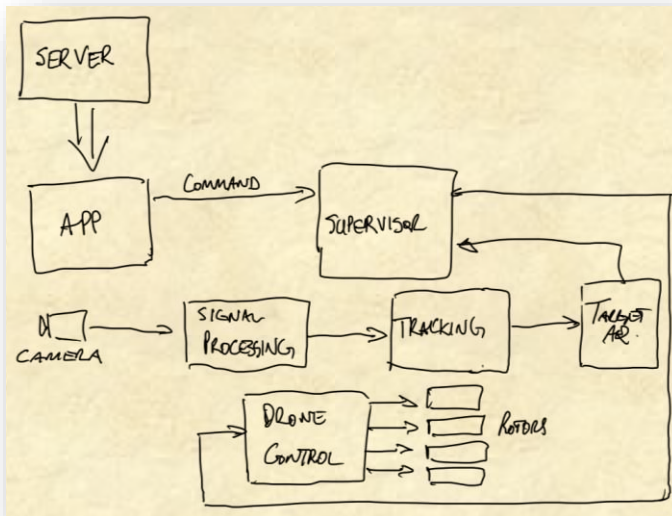
- Interdisciplinary approach and means to enable the realization of successful systems
- Systems engineers:
  - Ensure requirements of customers, users and other stakeholders are met
  - Design optimized system architectures
  - Validate system architecture meets requirements
  - Evaluate system level behaviors



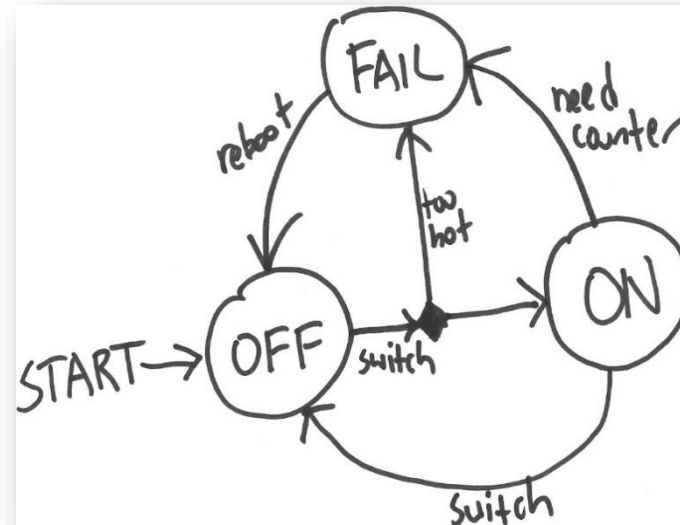
# What is System Architecture?

- A conceptual model that defines the structure, behavior, and other views of a system, organized in a way that supports reasoning about the system

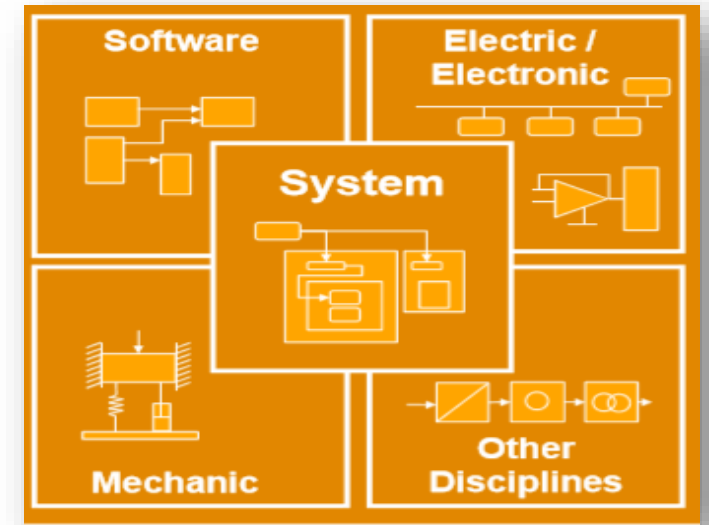
Structure



Behavior

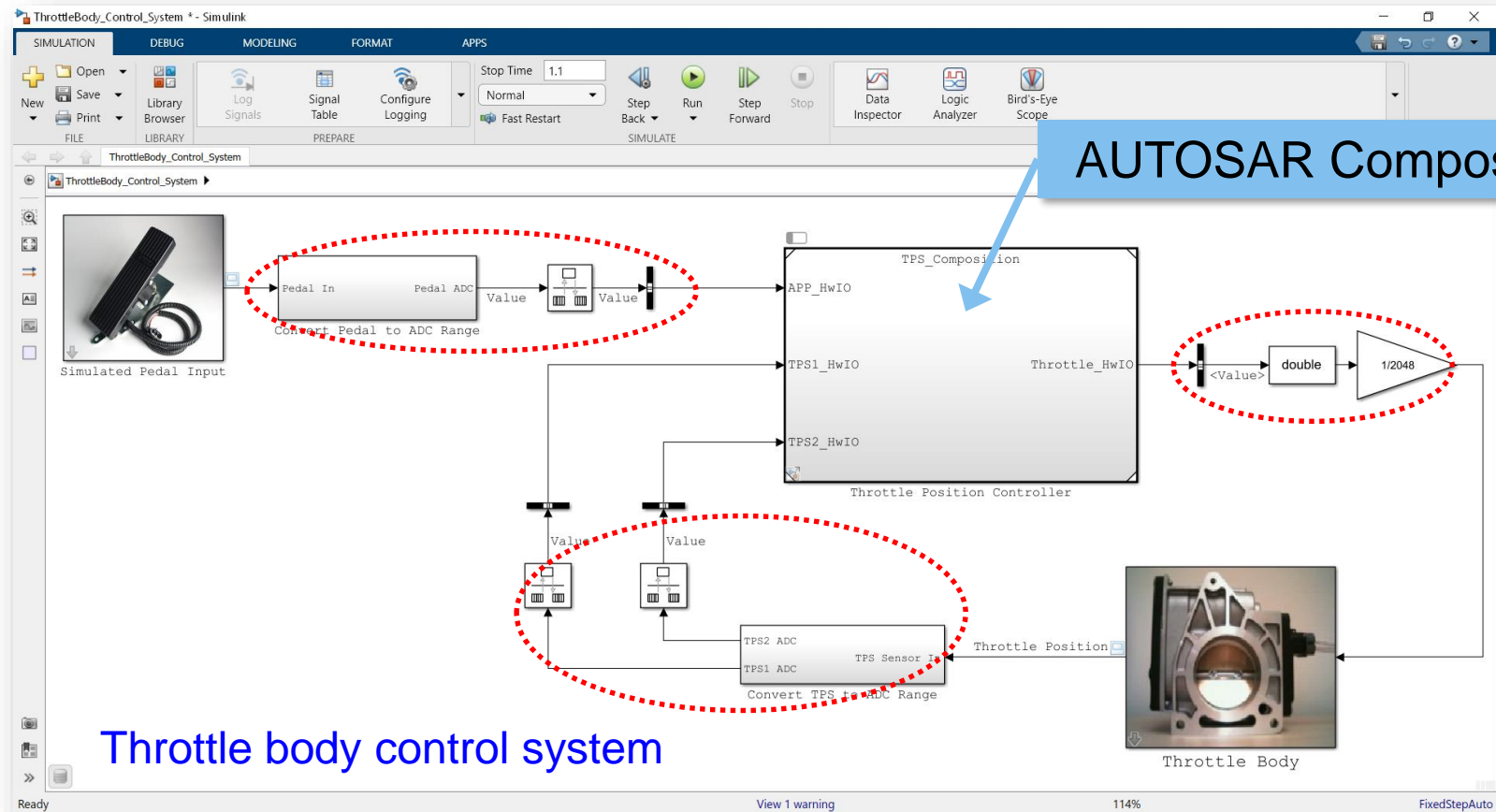


Other Views



# Consideration in System Architecture

- Simulation in a system level using Simulink

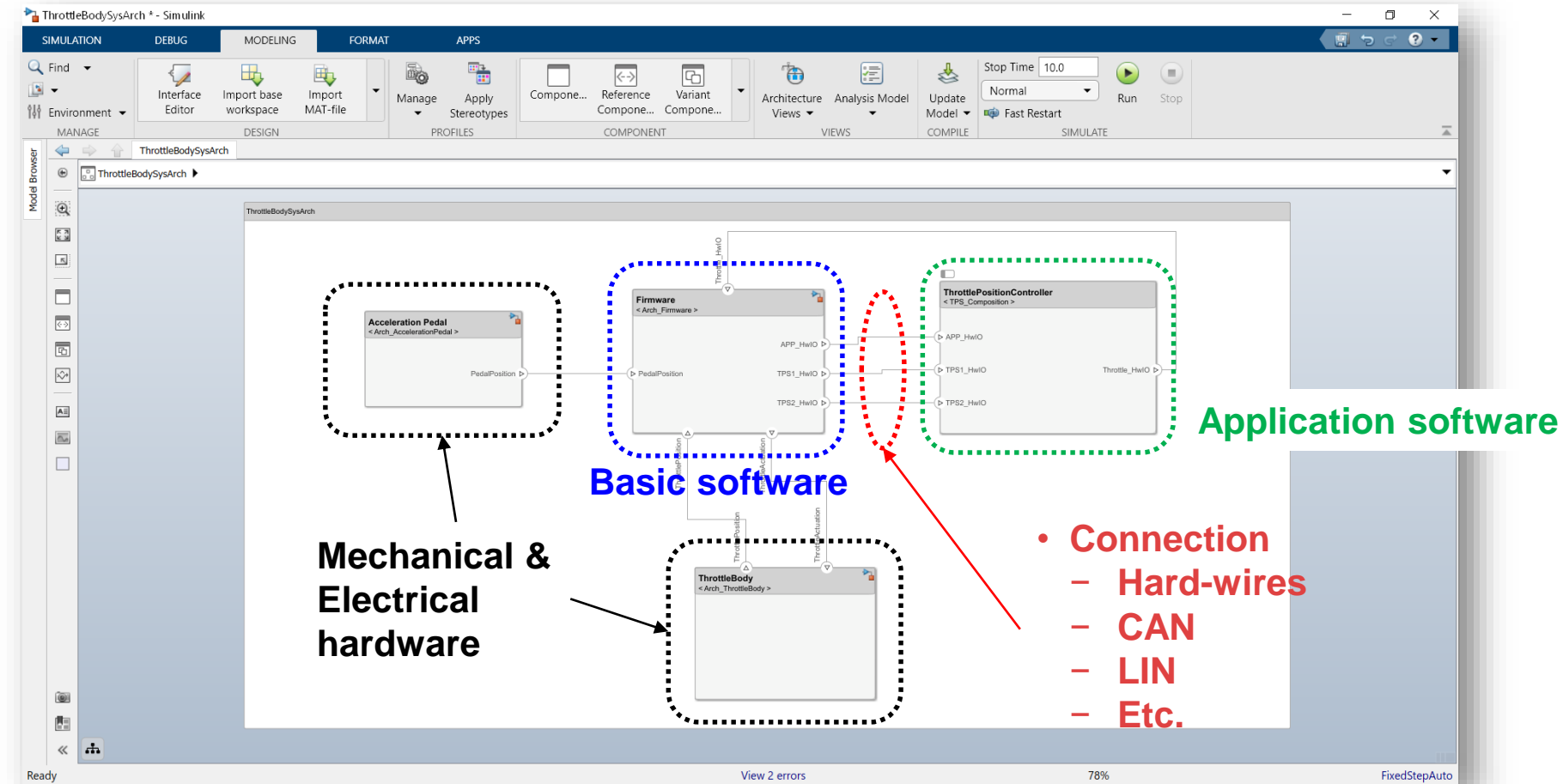


AUTOSAR Composition

Throttle body control system

# System Architecture with System Composer

- System architect considers physical and logical architecture



# Demo: Simulation in System Composer

The screenshot shows the MathWorks System Composer interface. The main workspace displays a block diagram of a ThrottleBodySystem. The diagram includes the following components and connections:

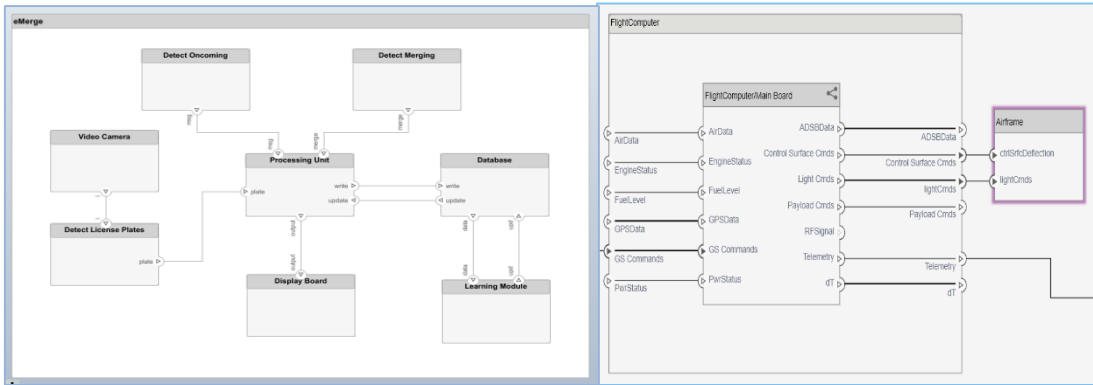
- AcceleratorPedal** (Arch\_AccelerationPedal) is connected to the **Firmware** block.
- Firmware** (Arch\_Firmware) contains sub-blocks **APP\_HwIo**, **TPS1\_HwIo**, and **TPS2\_HwIo**.
- ThrottlePositionController** (tps\_composition\_ref) contains sub-blocks **APP\_HwIo**, **TPS1\_HwIo**, and **TPS2\_HwIo**.
- ThrottleBody** (Arch\_ThrottleBody) contains a sub-block **Throttle\_HwIo**.
- Connections:
  - Firmware** is connected to **ThrottlePositionController** via **APP\_HwIo**, **TPS1\_HwIo**, and **TPS2\_HwIo**.
  - ThrottlePositionController** is connected to **ThrottleBody** via **Throttle\_HwIo**.
  - ThrottleBody** is connected to **Firmware** via **ThrottlePosition**, **ThrottleActuation**, and **ThrottleActualIn**.

The interface also shows a toolbar with simulation controls (Run, Step Forward, Step Back, Stop), a Property Inspector on the right, and a Model Browser on the left. The status bar at the bottom indicates 'Ready', '102%', and 'auto(FixedStepDiscrete)'.

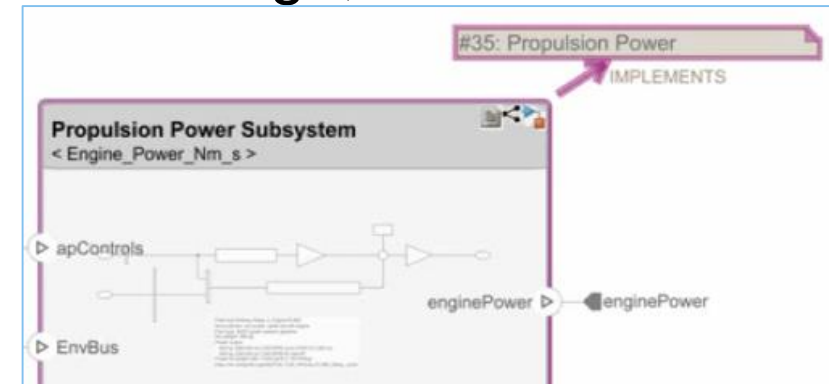


# System Engineering with System Composer

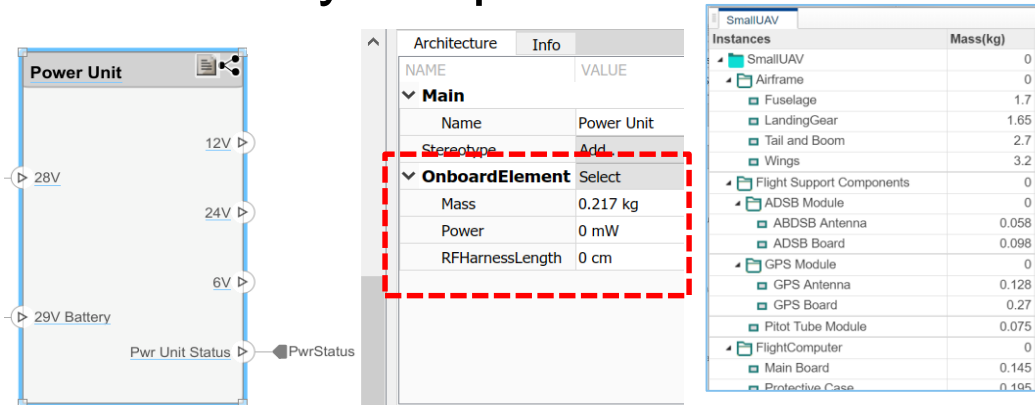
Intuitively design system and software architectures



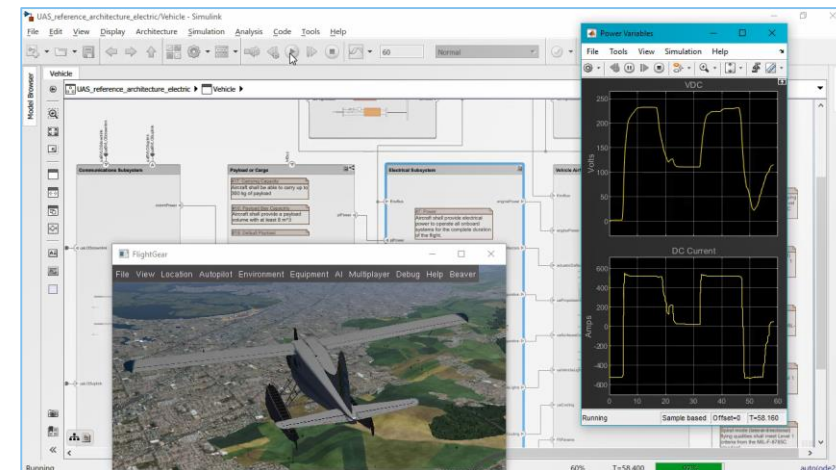
Link requirements, architectures, design, code and test



Add stereotypes and trade study to optimize architecture



Simulation with multi-domain environment



# 비디오 및 웨비나

비디오 검색

비디오

비디오 홈 | 검색

영업 상담 소프트웨어 평가판 신청

**Digital Thread from Requirements to Architecture and Design**  
Simulink Requirements

Author requirements or view from external source

Link requirements, architectures, design, code and test

Identify gaps in architecture or design

Identify impact of requirement changes

Implemented: 16, Justified: 0, None: 2, Total: 18

MATLAB EXPO 2019

피드백

## 요구사항부터 아키텍처 설계와 시뮬레이션까지 시스템 엔지니어링을 위한 방안

류성연 차장, MathWorks Korea

시스템 엔지니어링과 모델 기반 시스템 엔지니어링은 서로 다른 그룹에 대해 서로 다른 의미를 가질 수 있지만, 대부분의 정의는 시스템 세분화 및 요구 사항 할당 프로세스를 수행하는 데 사용되는 일련의 시스템 레벨의 요구 사항에서 시작하는 것을 포함한 공통적인 개념을 가집니다. 그런 다음 시스템 아키텍처 대안에 대해 트레이드 오프 분석을 통하여 설계가 진행되고, 요구 사항이 충족되는지 확인하기 위해 시뮬레이션 가능한 후보 아키텍처를

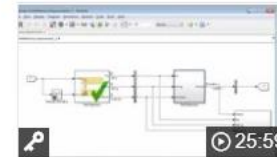
[Link](#)

평가판 신청

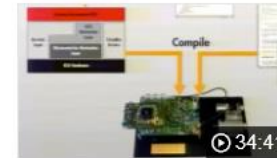
관련한 비디오 및 웨비나 보기



플랫폼 모델링 및 시스템 레벨 시뮬레이션



Simulation Testing in Model-Based Design



System Engineering of Automotive Electronic Control Systems...



Teaching Engineering Through Theory, Simulation, and...



BAE Systems Surface





# System Composer

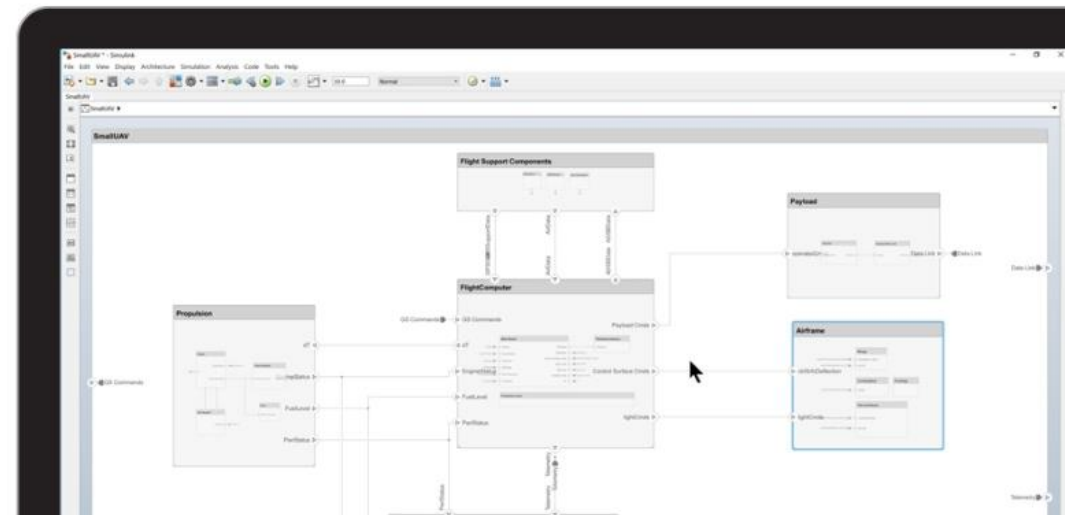
## Design and analyze system and software architectures

[Download a free trial](#)

<https://kr.mathworks.com/products/system-composer.html>

System Composer™ enables the definition, analysis, and specification of architectures and compositions for model-based systems engineering and software design. With System Composer, you allocate requirements while refining an architecture model that can then be designed and simulated in Simulink®.

System Composer lets you create or import architecture models that describe a system in terms of components and interfaces. You can also populate an architecture model from the architectural elements of Simulink designs or C/C++ code. You can create custom live views of the model to study specific design or analysis concerns. With these architecture models you can analyze requirements, capture properties via stereotyping, perform trade studies, and produce specifications and ICDS.



# AUTOSAR 블록셋

## AUTOSAR 소프트웨어 설계 및 시뮬레이션

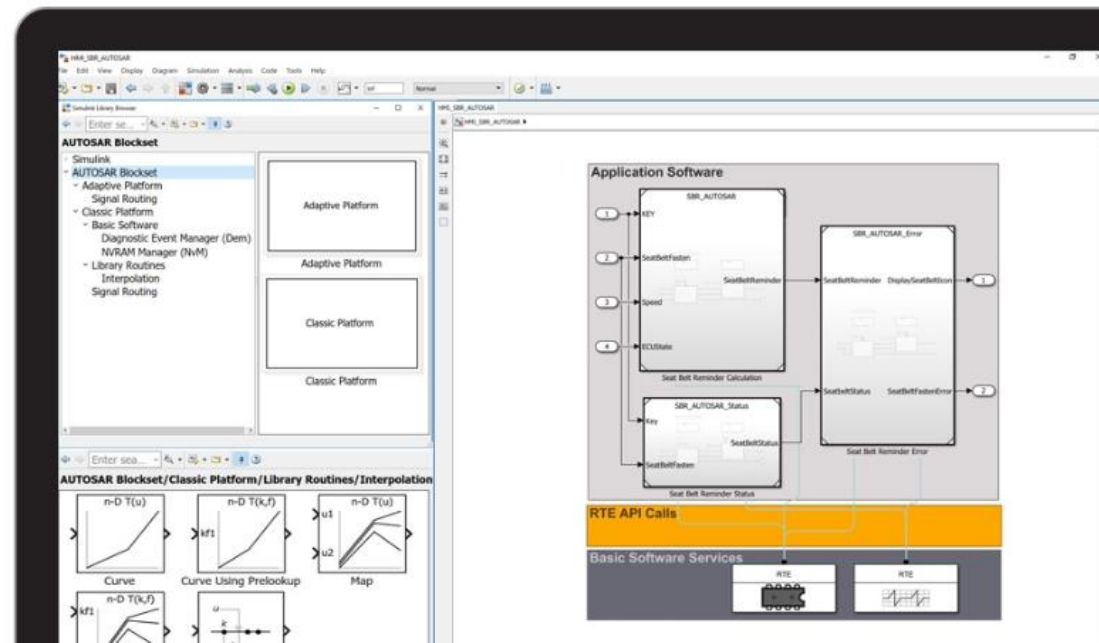
▶ 비디오 보기

↓ 소프트웨어 평가판 신청

AUTOSAR 블록셋은 Simulink® 모델을 사용하여 AUTOSAR Dictionary 과 클래식 및 어댑티브 AUTOSAR 소프트웨어 개발을 위한 블록을 제공합니다. AUTOSAR 소프트웨어 컴포넌트 속성, 인터페이스 및 데이터형을 정의한 다음, 이를 AUTOSAR 편집기를 사용하여 기존 Simulink 모델에 매핑할 수 있습니다. 또는, 블록셋은 AUTOSAR XML 파일에서 소프트웨어 컴포넌트와 컴포지션 설명을 가져와서 AUTOSAR의 새로운 Simulink 모델을 자동으로 생성할 수 있는 응용 프로그램 인터페이스를 제공합니다.

AUTOSAR 블록셋은 NVRAM 및 Diagnostics를 포함한 AUTOSAR 라이브러리 루틴과 BSW(기본 소프트웨어) 서비스용 블록과 구문을 제공합니다. 응용 프로그램 소프트웨어 모델과 함께 BSW 서비스를 시뮬레이션함으로써 Simulink를 떠나지 않고도 AUTOSAR ECU 소프트웨어를 검증할 수 있습니다.

AUTOSAR 블록셋은 C 및 C++ 제품 코드 생성 및 AUTOSAR XML 파일 내보내기를 지원합니다(Embedded Coder®와 함께) ISO 26262 표준(IEC 인증 키트 포함)과 함께 사용할 수 있습니다.



# Thank You!

