

Comprehensive workflow for AUTOSAR Classic & Adaptive using Model-based Design

Durvesh Kulkarni Senior Application Engineer MathWorks India



© 2019 The MathWorks, Inc.



#### Agenda

- Introduction to AUTOSAR
- Simulink for Classic Platform
  - Automatic modeling and code generation
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
- Simulink for Adaptive Platform
  - A closer look at the Adaptive layers
  - Motivation for Simulink to support Adaptive
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components
- Polyspace for AUTOSAR
- Additional Resources



#### Agenda

#### Introduction to AUTOSAR

- Simulink for Classic Platform
  - Automatic modeling and code generation
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
- Simulink for Adaptive Platform
  - A closer look at the Adaptive layers
  - Motivation for Simulink to support Adaptive
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components
- Polyspace for AUTOSAR
- Additional Resources



#### **Introduction to AUTOSAR**

- AUTOSAR AUTomotive Open Systems Architecture
  - Middleware standard, jointly developed by automobile manufacturers, electronics and software suppliers and tool vendors.
  - Motto: "cooperate on standards, compete on implementations"







#### A new platform for compute intensive applications

	<b>Classic AUTOSAR</b>	Adaptive AUTOSAR	In	fotainme	ent Platfo	rm	
	Application Software	Adaptive Application Software	GENIVI membe their d	GENIVI members build and integrate compliant products, and their differentiating features, tools, services			
			Audio	Graphics	Multimedia	Speech	
	RTE	ARA	CE- device	External	Connectivity	Positioning	
	Basic Software	Services Basis	Package Management	Access Networking	Security	Personal Information Management	
	Hardware	High Performance Hardware/Virtual Machine	Linux	System operating sys x86 or ARM-	Architecture tem, drivers ar based process	or	
Real time Requirements	High, in the range of micro-sec	Mid, in the range of milli-sec	l	-OW, n the range	e of sec		
Computing power	Low, ~ 1000 MIPs	High, > 20,000 MIPs	ŀ	High, - 10,000 N	llPs		

MIPs: Million Instructions per Second



#### **AUTOSAR Support Transition**

R2018b and earlier



Required

\*Requires MATLAB \*\*Requires MATLAB Coder and Simulink Coder



#### Agenda

- Introduction to AUTOSAR
- Simulink for Classic Platform
  - Automatic modeling and code generation
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
- Simulink for Adaptive Platform
  - A closer look at the Adaptive layers
  - Motivation for Simulink to support Adaptive
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components
- Additional Resources



**ECU Hardware** 



#### Automatic modeling and code generation

 Show quick start demo, edit in code perspective UI and AUTOSAR dict, code gen

	k_start 43
autosar_	.swc_quick_start >
Q	
=	Event Listener
	Runnable_Initialize
	Runnable_1s
	1 Dut1 1
	In1_1s
	► In2 Out2 ► 2
	SS1
	Runnable_2s Integrator
	K Ts

10



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





Basic software functionality is highly dynamic



Simulation of basic software reduces development time and improves software quality



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



**Detailed Specifications** 



#### Simulation of AUTOSAR ECU software

Seat Belt Reminder demo



Block Parameters: Curve

#### **AUTOSAR Library Routines**





#### Importing and exporting AUTOSAR descriptions (ARXML files)





#### **Import AUTOSAR XML to Simulink**

#### Import AUTOSAR Component to Simulink

```
ar = arxml.importer('ThrottlePositionControlComposition.arxml');
createComponentAsModel(ar,'/Company/Components/Controller',...
'ModelPeriodicRunnablesAs','AtomicSubsystem');
```





#### **AUTOSAR Software Components**





#### **AUTOSAR Composition-Software-Component**

- Compositions purely architectural element
  - Do not impact how components interact with RTE, and code



Composition component → Hierarchical aggregation of software components

Live Editor - \\fs-58-ah\vmgr\$\home08\sshwetha\Documents\MATLAB\examples\autosarstandard\ImportAUTOSARCompositionToSimulinkSpkgExample\ImportAUTOSARCompositionToSimulinkSpkgExample....

L	IVE EDITO	OR	INSERT	VIEV					
New	Open	Save	Find Files	Go To	Aa Title ▼ B I U M := 1= □ = =	Code	E Run Section Section Break Run and Advance	Run Step Stop	
		FILE		NAVIGATE	TEXT	CODE	SECTION	RUN	Ā

 $\times$ 

5.0

ImportAUTOSARCompositionToSimulinkSpkgExample.mlx 💥 🕂

#### Import AUTOSAR Composition to Simulink

Create Simulink® representation of AUTOSAR composition imported from AUTOSAR authoring tool anxml file

#### Import AUTOSAR Composition from arxm1 File to Simulink

Here is an AUTOSAR software composition that implements a throttle position control system. The composition contains six interconnected AUTOSAR software component prototypes -- four sensor/actuator components and two application components.

The composition was created in an AUTOSAR authoring tool and exported to the file ThrottlePositionControlComposition.arxml. (You can access the arxml files used in this example in the installed AUTOSAR support package tree, at autosarroot/autosar\_examples/ThrottlePositionControlSystem/arxml.)





#### Agenda

- Introduction to AUTOSAR
- Simulink for Classic Platform
  - Automatic modeling and code generation
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
- Simulink for Adaptive Platform
  - A closer look at the Adaptive layers
  - Motivation for Simulink to support Adaptive
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components
- Polyspace for AUTOSAR
- Additional Resources



#### **AUTOSAR Layered software architecture**



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





Notes: Each OS Process

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



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







#### Key Concept #2 Service-oriented inter-process communication





## Key Concept #2 Service-oriented communication

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





#### Key Concept #3: Everything is C++





#### **Motivation for Simulink to support Adaptive**

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



#### **Adaptive SW Architecture Concepts**





# Mapping AUTOSAR AP Concepts to Simulink









#### Mapping AUTOSAR AP Concepts to Simulink





#### **Generate Production AUTOSAR Adaptive C++ Code**





#### **Develop Adaptive AUTOSAR Components**

autosar\_LaneGuidance shipping demo



#### Agenda

- Introduction to AUTOSAR
- Simulink for Classic Platform
  - Automatic modeling and code generation
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
- Simulink for Adaptive Platform
  - A closer look at the Adaptive layers
  - Motivation for Simulink to support Adaptive
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components
- Polyspace for AUTOSAR
- Additional Resources



## **Polyspace for AUTOSAR**





#### What if...?

- The communication between the software architect and developer is made easy
- An easy setup process for verification
- Setup only needs the ARXML and code implementations
- And also check for the Run-Time errors in the code

## Check if...

- Implementation of Software Components Follow Specifications
- Edits to Specifications Impact
- Implementation for Run-time Errors and Mismatch with Specifications
- Implementation Against Specification Update



📣 MathWorks

## **Polyspace for AUTOSAR(R2018a)**



New view to detail the

AUTOSAR specification

Use Polyspace to perform a sound unit static analysis of the components of an AUTOSAR software



	V AUTOSAR Behavior Specification
📽 😼 🕅 🏂 marray.c / Runnable_Step()	🔶 🔿 🔁 🗟 🛤 😅
<ul> <li>Invalid use of AUTOSAR runtime environment function 3</li> <li>Function 'Rte_Write_Output_Output' is called with valid argument(s)</li> <li>Conditions on first argument 'data' (see <u>spec</u>):         <ul> <li>✓ data meets its specification.</li> <li>Specification: non-NULL</li> <li>✓ data meets its specification.</li> </ul> </li> </ul>	<ul> <li>▼ provided runnable-functions</li> <li>▼ used RTE-functions</li> <li>▼ all functions</li> <li>▼ terminology</li> </ul> Function Parameter
Specification: allocated (*data)[] meets its specification. Specification: [-2147483648 2147483647] Attraction: [-2147483648 2147483647]	Rte_Write_Output_Output Function required by Autosar Software- Component
Configuration Result Details Specified Constraints	<b>IN</b> parameter data is a rt_Array_SInt32_10ConstRef
Source  Source Source Source  Source Source Source Source Source Source Source Source Source Source Source Source Source Source Source Source Source S	constant pointer to a constant <b>Matrix type</b> rt_Array_SInt32_10 1D-Matrix[10] of <b>Integer</b> implementation type
<pre>100 /* Outport: '<root>/Output' */ 101 (void) <u>Rte_Write_Output_Output</u>(marray_B.ZeroOrderHold3); 102</root></pre>	values must be in full - range [-2147483648 2147483647] ▼base software-type
<pre>103 /* Outport: '<root>/Outputl' */ 104 <u>Rte_IWrite_Runnable_Step_Outputl_Output1(marray_B.ZeroOrderHold4);</u> 105 107 108 109 109 109 109 109 109 109 109 109 109</root></pre>	▼ physical-range project-checksum=(11336998240055185542)
Dashboard Source Output Summary Graph Run Log	4

New checks to prove

the specification

that the code matches

Polyspace code prover



AUT@SAR

#### **Polyspace for AUTOSAR: How do I launch from UI?**

V Busiest	Description	
Define pro	ject	<ul> <li>Create project using AUTOSAR sp</li> <li>Create project using AUTOSAR</li> <li>Create project using AUTOSAR sp</li> </ul>
Project defini	tion and location	
Project name	psar_demo	Create project using AUTOSAR
Version	1.0	Project has been successfully generat
Author	cbard	AUTOSAR
		Specify AUTOSAR ARXML folder
Use defau	It location	F:\Customers\AUTOSAR_Demo\AUTOSA
location E:V	ustomers ALTOSAR Demo ALTOSAR	Specify AUTOSAR source folder
Locoson 1 . le		F:\Customers\AUTOSAR_Demo\AUTOSA
Project config	puration	
Use temp	ate om build command	
Create fro	om AUTOSAR specification	Command output
	Back Next	info: End update ComponentProver_cont info: End Configure CodeProver Projects warning: Event 'execute_prove' triggers info: End Project Execution with 0 errors info: Start Save project to file 'F:\Custom

ecification specification ecification specification ted, please click on finish for opening AR\_DEMO\sw\arxml AR\_DEMO\sw\code Stop Run evictoonic\_benevictoonic\_bene figuration (42ms,0 errors,0 warnings) (182ms,0 errors,0 warnings) transition from 'prover\_configured' to and 0 warnings. The state of the pro ners\AUTOSAR\_Demo\AUTOSAR\_DE ers\AUTOSAR Demo\AUTOSAR DEM info: Browse project-file at: file://localhost/F:/Customers/AUTOSAR\_Demo/AU warning: End Polyspace-AUTOSAR Project (3388ms,0 errors, 1 warnings)

Back

Next

Finish

Cancel

▼ Polyspace F:\Customers\AUTOSAR\_Demo\AUTOSAR\_DEMO\psar\_demo\psar\_project.psprj

File Reporting Metrics Tools	Window Help	
🏝 🍓 🔚 ╞ Run Code Prover 🔻	Stop 🔍	
🐴 Project Browser	Bug Finder	Configuration
🕂 🗑 🚳   💐 🖳 🛨 于 🔽	Code Prover	tions ×
psar_demo project Source Files	Create new Bug Finder result folder	Target & Com Macros
Project Include Folders	Create new Code Prover result folder	Environme
jyb.tst002.swc001bh	Run All Modules	Inputs & Stubi
Module Source Files	Run analysis on all	modules in priAn
		Code Prover V
🔀 options		Verification
e Result		Drecision
prover		Scaling
Image: Byb.tsto02.swc002.bhv		Reporting
		Run Settings
		Advanced Sett
1		
1		
v		
>		



#### **Polyspace for AUTOSAR**

Verify ARXML against Code Demo



#### **Polyspace checks AUTOSAR C++14 Rules**

Guidelines for	the use of the C-	+14 langu	age in critical and safet AUTOSAR A	y-related sy	/stems	UTOSA	Select file ■ MISRA C:2004 (131/131) R C++14	Browse DNew to Save Chan Select rules in category: ⊘All ⊘required ⊘advisory (105/105)
Docume	nt Title	Guidel C++14 safety-	ines for the use language in crit related systems	of the tical and	L. I		SELCERT C++ (121/121)     ISO/IEC TS 17961 (46/46)     AUTOSAR C++14 (195/195)     O Language independent issues     1 General     2 Lexical conventions     -3 Basic concepts	3 Basic concepts     4 Standard conversions     5 Expressions     6 Statements     7 Declaration     9 Ocasses     9 Ocasses
Jocument Owr	ier	AUTOSA	.R			<ul> <li>4 Standard conversions</li> <li>5 Expressions</li> </ul>		
Document Responsibility AUTOSA		SAR				-6 Statements -7 Declaration		
Ocument Iden	Coding Standards & Co	de Metrics		(1) (1) (1) (1) (1) (1) (1) (1) (1) (1)	9 Classes     10 Derived Classes     11 Member access control     12 Special member functions     13 Overloading     14 Templates	15 Exception handling     16 Preprocessing directives     17 Library introduction     18 Language support library     19 Diagnostics library     20 General utilities library		
Part of AU	Environment Settings outs & Stubbing					~	-15 Exception handling	Z1 Strings library
Part of Sta	Sta     Multitasking     Coding Standards & Code Metrics     Bug Finder Analysis     Code Prover Verification     Verification Assumptions     Check Behavior		Check JSF AV C++ Check SEI CERT-C Check SEI CERT-C++ Check ISO/IEC TS 17961	shali-rules all all all	View View View View		16 Preprocessing directives     17 Library introduction     18 Language support library     19 Diagnostics library     20 General utilities library     21 Strings library	
Date			Check AUTOSAR C++14	ali	View	-	-23 Containers library	
017-03-31 Re	Precision Scaling porting		Code Metrics	Edit			23 Augonomis library 27 Input/output library @Custom (43/43)	
								Finish Ca

#### https://www.autosar.org/fileadmin/user\_upload/standards/adaptive/17-03/AUTOSAR\_RS\_CPP14Guidelines.pdf



#### **Benefits of using Polyspace for AUTOSAR**

- Polyspace automatically modularizes analysis based on AUTOSAR components
- Polyspace detects mismatch between code and AUTOSAR XML spec
  - AUTOSAR runnable not implemented
  - Invalid result of AUTOSAR runnable implementation
  - Invalid use of AUTOSAR runtime environment function
- Prove absence of certain types of run-time errors in runnables (e.g. OverFlow, divByZero)



## Summary

#### **AUTOSAR Blockset :**

- Model AUTOSAR Classic and Adaptive software components
- Simulate AUTOSAR compositions and ECUs
- Generate optimized AUTOSAR C/C++ code, roundtrip ARXML, and perform SIL and PIL verification (requires Embedded Coder<sup>®</sup>)
- Is well-suited for applications involving embedded production deployment
- Is key part of Model-Based Design by providing detailed specification of embedded software



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).



#### **AUTOSAR Support Transition**

R2018b and earlier

#### **AUTOSAR Blockset**

R2019a and later



**Required for sim Required for code** 

\*Requires MATLAB

Required

\*\*Requires MATLAB Coder and Simulink Coder



#### **User Articles/Presentations**

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







#### To learn more, please visit AUTOSAR Blockset page



#### Come see us at our demo booth

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).

*****		
122		-t-
Party of Lot.	Sale Sale	
	100.000	
-	Concernant of the second	
7.5	71524	the second s
	son Cra-1	

