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

Durvesh Kulkarni
Senior Application Engineer
MathWorks India
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”

![Diagram showing the layers of partners: Attendees, Associate Partners, Development Partners, Premium Partners, Core Partners.](image-url)
OVER THE AIR UPDATE

AUTOSAR Platforms

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

Non- AUTOSAR

Software

Hardware

Classic AUTOSAR

Application Software

RTE

Basic Software

Hardware

Adaptive AUTOSAR

Adaptive Application Software

ARA

Services

Basis

High Performance Hardware/Virtual Machine
A new platform for compute intensive applications

**New Platform Requirements**

<table>
<thead>
<tr>
<th>Requirements</th>
<th>Classic AUTOSAR</th>
<th>Adaptive AUTOSAR</th>
<th>Infotainment Platform</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computing power</td>
<td>High, in the range of micro-sec</td>
<td>Mid, in the range of milli-sec</td>
<td>Low, in the range of sec</td>
</tr>
<tr>
<td>Low, ~ 1000 MIPs</td>
<td>High, ~ 20,000 MIPs</td>
<td>High, ~ 10,000 MIPs</td>
<td></td>
</tr>
</tbody>
</table>

**Real time Requirements**

- High, in the range of micro-sec
- Mid, in the range of milli-sec
- Low, in the range of sec

**MIPs**: Million Instructions per Second

Source: AUTOSAR Adaptive Platform Joint Meeting May 2017
AUTOSAR Support Transition

- R2018b and earlier

*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
AUTOSAR Support from Embedded Coder and Simulink

Software Architecture Definition

Application Layer

Run Time Environment (RTE)

Basic Software

ECU Abstraction Layer

Microcontroller Abstraction Layer

Complex Device Drivers

ECU Hardware

Services Layer

Behavior Modeling & Code Generation

BSW Configuration & RTE Generation

Modeling and Simulation

Authoring Tools

Basic SW Providers

Autosar

Embedded Coder

Simulink
Automatic modeling and code generation

- Show quick start demo, edit in code perspective UI and AUTOSAR dict, code gen
Functional simulation of AUTOSAR basic software is critical for AUTOSAR ECU development

Many calls between application software and basic software

Basic software functionality is highly dynamic

Simulation of basic software reduces development time and improves software quality
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
AUTOSAR Library Routines

Rte_IWriteRunnable_Step_Out1_Out1(I1Fp_1ntIpoCur_f32_f32
(Rte_IReadRunnable_Step_In1_In1(), _L4_single.Nx, _L4_single.Bp1,
_L4_single.Table));
Importing and exporting AUTOSAR descriptions (ARXML files)
Import AUTOSAR XML to Simulink

Import AUTOSAR Component to Simulink

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

ThrottlePositionControlComposition.arxml
AUTOSAR Software Components

CompositionSW Component Type

ApplicationSW Component Type

SensorActuatorSW Component Type

ParameterSW Component Type

ServiceProxySW Component Type

NvBlockSW Component Type

RTE

ServicesSW Component Type

ECUAbstractionSW Component Type

ComplexDeviceDrive rSW Component Type

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

Composition component ➔ Hierarchical aggregation of software components
Import AUTOSAR Composition to Simulink

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

Import AUTOSAR Composition from arxml 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

Adaptive AUTOSAR Foundation

High Performance Hardware/Virtual Machine

Adaptive AUTOSAR Run-time for Adaptive (ARA)

Components
Run-time
Basic Services
Hardware

API
OS

Execution
Communication

Service
S/W CM
Diagnostics

Adaptive AUTOSAR Services

Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Adaptive Application (SW-C)
Key Concept #1
Everything is a process .. as in “OS process”

OS Process #1
Adaptive Application (SW-C)

OS Process #2
Adaptive Application (SW-C)

OS Process #3
Adaptive Application (SW-C)

OS Process #4
Adaptive Application (SW-C)

AUTOSAR Run-time for Adaptive (ARA)

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”

AUTOSAR Run-time for Adaptive (ARA)

OS Process #1
Adaptive Application (SW-C)

OS Process #2
Adaptive Application (SW-C)

OS Process #3
Adaptive Application (SW-C)

OS Process #4
Adaptive Application (SW-C)

API
OS (POSIX Compliant)

API
Execution

API
Communication

Inter-Process Communication

Provides multi-process capability

Process scheduling

Process life-cycle management.
Key Concept #2
Service-oriented inter-process communication
Key Concept #2
Service-oriented communication

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

<<interface example>>

```
RadarService

• result = Calibrate(config)
• [success, out_pos] = Adjust(in_pos)

• BrakeEvent

• UpdateRate
```
Key Concept #3: Everything is C++
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

Adaptive Application

Service Interface

"Radar" : {
   // events
   "event" : {
      "brakeEvent"
      "parkingBrakeEvent"
   },
   // methods
   "method" : {
      "Calibrate"
      "Adjust"
   },
   // fields
   "field" : {
      "updateRate"
   }
}
Mapping AUTOSAR AP Concepts to Simulink

```
"Radar" : {
    "event" : {
        "brakeEvent",
        "parkingBrakeEvent"
    },
    "method" : {
        "Calibrate",
        "Adjust"
    },
    "field" : {
        "updateRate"
    }
}
```
Mapping AUTOSAR AP Concepts to Simulink

Adaptive Application

ProvidedPort

"Radar" : {
  // events
  "event" : {
    "brakeEvent"
    "parkingBrakeEvent"
  },
  // methods
  "method" : {
    "Calibrate"
    "Adjust"
  },
  // fields
  "field" : {
    "updateRate"
  }
}
Generate Production AUTOSAR Adaptive C++ Code

Out-of-box AUTOSAR support
1. Configure Model
   ✓ Target
   ✓ AUTOSAR Dictionary
2. Generate 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
Polyspace and AUTOSAR

AUTOSAR architecture

Application Layer

RTE

Services Layer
- ECU Abstraction Layer
- Complex Device Drivers
- Microcontroller Abstraction Layer

ECU Hardware

ARXML

Polyspace verifies the match between code and ARXML

Polyspace verifies the Application Layer

Polyspace stubs the RTE Layer
RTE Layer not verified by “Polyspace for AUTOSAR”

Not verified by “Polyspace for AUTOSAR”

ARXML provides specification of Application Layer and link with RTE
Polyspace for AUTOSAR(R2018a)

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

New checks to prove that the code matches the specification

New view to detail the AUTOSAR specification

Polyspace code prover
Polyspace for AUTOSAR: How do I launch from UI?
Polyspace for AUTOSAR

- Verify ARXML against Code Demo
Polyspace checks AUTOSAR C++14 Rules

Guidelines for the use of the C++14 language in critical and safety-related systems

<table>
<thead>
<tr>
<th>Document Title</th>
<th>Guidelines for the use of the C++14 language in critical and safety-related systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>Document Owner</td>
<td>AUTOSAR</td>
</tr>
<tr>
<td>Document Responsibility</td>
<td>AUTOSAR</td>
</tr>
<tr>
<td>Document Identification No</td>
<td>839</td>
</tr>
</tbody>
</table>

AUTOSAR C++14 (195/195)

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®)

- Is well-suited for applications involving embedded production deployment
- Is key part of Model-Based Design by providing detailed specification of embedded software
AUTOSAR Support Transition

- R2018b and earlier

AUTOSAR Blockset

- R2019a and later

*Requires MATLAB
**Requires MATLAB Coder and Simulink Coder
User Articles/Presentations

- **BMW** - 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

- **LG Chem** - Developing AUTOSAR and ISO 26262 Compliant Software for a Hybrid Vehicle Battery Management System with Model-Based Design

- **John Deere** - Vertical AUTOSAR System Development at John Deere
To learn more, please visit AUTOSAR Blockset page

Come see us at our demo booth