Simulink for AUTOSAR Adaptive

Dr Richard Thompson
Software Engineering Manager
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
- Additional Resources
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
- Additional Resources
AUTOSAR Classic is already on the road

- **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
Motivation for AUTOSAR Adaptive

- Main drivers – Automated driving, Car-2-car/infrastructure applications
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
  - Scaling from software components to compositions
- Simulink for Adaptive Platform
- Additional Resources
Intuitive and Powerful AUTOSAR Software Design in Simulink

Classic Platform

- Application Software
- RTE
- Basic Software and Routine Libraries

Adaptive Platform

- Application Software
- ARA
- Functional Clusters
Importing and Exporting AUTOSAR SW-C Descriptions (ARXML files)
It is easy to get started from an AUTOSAR description (Import)

1. Import SW-C description (arxml) & create Simulink model
   
   ```
   h = arxml.importer('mySWC.arxml')
   h.createComponentAsModel('/path/mySWC')
   ```

2. Elaborate SW-C Design, implement & generate code from model
It is also easy & quick to configure a Simulink model for AUTOSAR

1. Start with a Simulink model

2. Click the AUTOSAR Component Quick Start App

3. Elaborate SW-C Design, implement & generate code from model
Example of Configuring a model for AUTOSAR
Now we can focus on modeling

1. Start with a Simulink model (or import SW-C description)

2. Elaborate SW-C design, implement & generate code from model
AUTOSAR SW-C design in Simulink

1) What blocks in this model need to be configured for AUTOSAR?

2) How do I change my AUTOSAR properties in the model?

3) Where do I get more information/help?
Introducing AUTOSAR “perspective” in a Simulink model

Quick Help
Help on configuring model for AUTOSAR

Property Inspector
View/Edit AUTOSAR SW-C Properties

Code Mappings Spreadsheet
View/Edit all blocks and elements configured for AUTOSAR
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
It’s easy to configure and play!
Rte_IWrite_Runnable_Step_Out1_Out1(Ifl_IntIpoCur_f32_f32(Rte_IRead_Runnable_Step_In1_In1()), Rte_CData_L_4_single()->Nx, Rte_CData_L_4_single()->Bp1, Rte_CData_L_4_single()->Table);
Scaling from software components to compositions
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- 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 Layered Software Architecture

Adaptive AUTOSAR Foundation

High Performance Hardware/Virtual Machine

Components

Run-time

Basic Services

Hardware
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)

**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)

**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)
Execution
Communication

Provides multi-process capability
Process scheduling
Process life-cycle management.
Inter-Process Communication
Key Concept #2
Service-oriented inter-process communication

Linux Machine 1

- Process
- IPC

Linux Machine 2

- Process
- IPC
- Network
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++
Adaptive Platform Roadmap

**Source** - 11th AUTOSAR Open Conference (www.autosar.org)

**Maturity level**

- **Early adopters** – Volkswagen, BMW, Bosch, LG Electronics…

**AP 18-03**

- 2018

**AP 18-10**

- 2019

**AP 19-03**

- 2020
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

Service Interface

"Radar" : {
  // events
  "event" : {
    "brakeEvent"
    "parkingBrakeEvent"
  },
  // methods
  "method" : {
    "Calibrate"
    "Adjust"
  },
  // fields
  "field" : {
    "updateRate"
  }
}
MAPPING AUTOSAR AP CONCEPTS TO SIMULINK

"Radar": {
  // events
  "event": {
    "leftLaneDistance",
    "leftTurnIndicator",
    "leftCarInBlindSpot",
    "rightLaneDistance",
    "rightTurnIndicator",
    "rightCarInBlindSpot"
  },
  // methods
  "method": {
    "Calibrate",
    "Adjust"
  },
  // fields
  "field": {
    "updateRate"
  }
}
Mapping AUTOSAR AP Concepts to Simulink

"Radar": {
    "event": {
        "leftHazardIndicator",
        "rightHazardIndicator"
    },
    "method": {
        "Calibrate",
        "Adjust"
    },
    "field": {
        "updateRate"
    }
}
AUTOSAR Adaptive in Action
Generate Production AUTOSAR Adaptive C++ Code

1. Configure Model
   ✓ Target
   ✓ AUTOSAR Dictionary
2. Generate C++ code
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
- Additional Resources
To learn more, please visit AUTOSAR Blockset page

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 data types, 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 back software (BSW) services, including WWAN and Diagnostics. By simulating the I2T services together with your application software models, 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 EC Certification Kit).

Come see us at the demo booth