Simulink for AUTOSAR Adaptive

Mark Danielsen
Senior Application Engineer
Agenda

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

- **BMW** - Model-Based Software Development: An 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
AUTOSAR at a System Level

Non - AUTOSAR

AUTOSAR
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
  - Importing and exporting AUTOSAR descriptions artifacts (ARXML files)
  - AUTOSAR Coder Dictionary
  - Simulation of AUTOSAR ECU software
  - Blocks for AUTOSAR Library routines
- Simulink for Adaptive Platform
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’s quick & easy to configure a Simulink model for AUTOSAR

1. Start with a Simulink model

![Diagram showing the process of configuring a Simulink model for AUTOSAR](image)
It’s quick & easy 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

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

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

AUTOSAR Quick Start - Set Component Type

Configure AUTOSAR software component properties
Component details:

Map model to AUTOSAR software component
Component name: Counter1
Component package: /Company/Powetrain/Components
Component type:
- Application
- ComplexDeviceDriver
- EcuAbstraction
- SensorActuator
- ServiceProxy

What to consider
AUTOSAR Component Quick Start maps a Simulink model to an AUTOSAR software component. For the component, specify an AUTOSAR short name, package path, and component type, or accept default values. Package paths can use an organizational naming pattern, such as /Company/Powetrain/Components. Component type determines the APIs available to the component in the runtime environment.

About the selected option
Creates application software component
Quick Start – Set Interfaces

Select the input for creating interface properties.

- Create defaults based on the Simulink model
- Import from ARXML

What to consider

AUTOSAR Component Quick Start creates AUTOSAR interface properties by applying defaults to a Simulink model or importing AUTOSAR XML (.arxml) element definitions.

About the selected option

Creates default AUTOSAR interface properties based on the Simulink model.
Once Quick Start is finished, you can view the configuration.
AUTOSAR Code Mappings

Sync Model and Code Mappings

Validate Mappings & AUTOSAR Attributes

Launch AUTOSAR Dictionary
Launch the AUTOSAR Dictionary
Once Configured, the user can generate AUTOSAR complaint code
Importing and Exporting AUTOSAR SW-C Descriptions (ARXML files)
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
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
BSW library Blocks allows user to Simulate Client / Server Calls

Detailed Specifications of Diagnostic Event Manager

BSW AUTOSAR Specs Encapsulated in

Client Block Resides in SWC Application

Server Block Resides in Simulation Test Harness
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);
Agenda

- AUTOSAR is already on the road
- Simulink for AUTOSAR
- Simulink for Adaptive Platform
  - Motivation for New AUTOSAR Platforms
  - A closer look at the Adaptive layers
  - Mapping Adaptive platform to Simulink
  - Code Generation for Adaptive components
Motivation for new AUTOSAR Platforms

- Main drivers – Automated driving, Car-2-car/infrastructure applications
Expansion of AUTOSAR based on Autonomous Applications

- In 2016 work started on creating these additional AUTOSAR Platforms
- March of 2017 is the first published release of AUTOSAR Adaptive Platform

The platforms are organized by 5 AUTOSAR standards:

- Acceptance Test
- Application Interfaces
- Classic Platform
- Adaptive Platform
- Foundation

From AUTOSAR.org – AUTOSAR Introduction
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

Basic Services

High Performance Hardware/Virtual Machine
Either AUTOSAR Platform benefits from Design in Simulink

Power of Simulation in the Application Layer aligns well with Algorithm Development
AUTOSAR Layered Software Architecture

Adaptive AUTOSAR Foundation

Components

Run-time

Basic Services

Hardware

Adaptive AUTOSAR Run-time for Adaptive (ARA)

API

Execution

Communication

Service

S/W CM

Diagnostics

Adaptive AUTOSAR Services

High Performance Hardware/Virtual Machine
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)

Provides multi-process capability

Process scheduling

Inter-Process Communication

OS (POSIX Compliant)

Execution

Communication

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++

AUTOSAR Run-time for Adaptive (ARA)

- Adaptive Application
- Adaptive Application
- Adaptive Application
- Adaptive Application
- ASW::XYZ Non-PF Service
- ASW::ABC Non-PF Service

User Applications

- ara::com Communication Mgmt.
- ara::rest RESTful
- ara::sync Time Synchronization
- ara::per Persistency
- ara::phm Platform Health Mgmt.
- ara::s2s service Signal to Service Mapping
- ara::diag service Diagnostics
- ara::nm service Network Management
- ara::ucm service Update and Configuration Management

- ara::core Core Types
- ara::exec Execution Mgmt.
- ara::iam Identity Access Mgmt.
- ara::log Logging & Tracing
- ara::crypto Cryptography

- POSIX PSE51 / C++ STL Operating System
- High Performance Hardware/Virtual Machine
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
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" : {
    // events
    "event" : {
        "leftHazardIndicator",
        "rightHazardIndicator"
    },
    // methods
    "method" : {
        "Calibrate",
        "Adjust"
    },
    // fields
    "field" : {
        "updateRate"
    }
}
Example of Configuring a model for Adaptive Platform
Change Target to AUTOSAR Adaptive
Enter Code Perspective to start the Configuration process
AUTOSAR Quick Start – Set Component

Configure AUTOSAR software component properties
Component details:

Map model to AUTOSAR software component (Adaptive)

Component name: autosar_LANE_Guidance
Component package: /Company/Powertrain/Components

AUTOSAR Component Quick Start maps a Simulink model to an AUTOSAR software component. For the component, specify an AUTOSAR short name, package path, and component type, or accept default values. Package paths can use an organizational naming pattern, such as /Company/Powertrain/Components. Component type determines the APIs available to the component in the run-time environment.
Quick Start Complete – Code Mappings setup for AS Port Events
Adaptive AUTOSAR Dictionary – Notice the Service Interfaces
Generate Code for the Adaptive AUTOSAR Model

Code Generation Report

Model Information

<table>
<thead>
<tr>
<th>Author</th>
<th>The MathWorks, Inc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last Modified By</td>
<td>The MathWorks, Inc.</td>
</tr>
<tr>
<td>Model Version</td>
<td>1.2.24</td>
</tr>
<tr>
<td>Tasking Mode</td>
<td>SingleTasking</td>
</tr>
</tbody>
</table>

Configuration settings at time of code generation

<table>
<thead>
<tr>
<th>Code Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Target File</td>
</tr>
<tr>
<td>Hardware Device Type</td>
</tr>
<tr>
<td>Simulink Coder Version</td>
</tr>
<tr>
<td>Timestamp of Generated Source Code</td>
</tr>
<tr>
<td>Location of Generated Source Code</td>
</tr>
<tr>
<td>Type of Build</td>
</tr>
<tr>
<td>Objectives Specified</td>
</tr>
</tbody>
</table>

Additional Information

| Code Generation Advisor | Not run |

OK  Help
C++ Adaptive AS Code ara Functional Cluster API

```cpp
// Function for Chart: 'CS1/Event Receive'
bool autosar_LaneGuidanceModelClass::autosar_LaneGuidance_sf_msg_pop_evtIn
(void)
{
    bool_t_isPresent;
    const auto::SampleContainer auto::SamplePtr auto::SampleContainer;
    if (autosar_LaneGuidance_DwEvtIn_isValid_1) {
        isPresent = true;
    } else {
        // Fetch data for event "leftLaneDistance" from ARA middleware
        if (RequiredPort->leftLaneDistance.Update()) {
            // Access event data
            sampleContainer = &RequiredPort->leftLaneDistance.GetCachedSamples();
            // Copy event data to application
            autosar_LaneGuidance_DwEvtIn_msgData_pa = **sampleContainer->begin();
            autosar_LaneGuidance_DwEvtIn_msgDataPtr_pa =
            &autosar_LaneGuidance_DwEvtIn_msgData_pa;
            // Received new event data
            isPresent = true;
            // Explicitly clean the event data cache
            if (RequiredPort->leftLaneDistance.cleanup()) {
                // Event data not received
            } else {
            }
        }
    }
}
```
Software Component Description Files Generated
Adaptive Standalone Application Code needs a main.cpp
Generate Production AUTOSAR Adaptive C++ Code

1. Configure Model
   - System Target File
   - AUTOSAR Dictionary

2. Generate C++ code
To learn more, please visit AUTOSAR webpage

Come see us at the demo booth