12 Years of AUTOSAR
Enabling Innovation with Model-Based Design

Michael Fröstl – Pilot Engineering @ MathWorks Germany

AUTOMOTIVE CONFERENCE 2015
switch(braindump)
{
}
case ‘AUTOSAR Acronym’:
AUTOSAR means

1. AUTOmobile Search And Rescue
2. AUTHentic Sportscar Aspect Ratio
3. AUTomotive Open System Architecture
4. AUTocar Occupant Specific Aversion Rate
5. AUTOrecovery Software Abstraction Reloaded
default:
  printf("Wrong session?");
}
AUTOSAR_Overview();
AUTOSAR – 3-layered Architecture

- Application Layer
  - Application Software Component
  - Actuator Software Component
  - Sensor Software Component
  - ... Application Software Component

- RTE (Real-Time Environment)

- Services Layer
  - ECU Abstraction Layer
  - Microcontroller Abstraction Layer

- ECU Hardware

- Complex Device Drivers
# MathWorks AUTOSAR Approach

<table>
<thead>
<tr>
<th>No separate AUTOSAR Blockset needed</th>
<th>• Code-generation through Mapping</th>
</tr>
</thead>
</table>
| **AUTOSAR Software Component Approach with Simulink** | • Simulink for developing behavior  
• Import and Export of SW Component Description Files (ARXML) |
| **Simultaneous generation of C-code and ARXML-Files** | • Consistency between C code and ARXML SW-C description files |
| **AUTOSAR Support Package for Embedded Coder** | • Available via web download  
• Allows more frequent updates and fixes |

* [http://www.mathworks.com/hardware-support/autosar.html](http://www.mathworks.com/hardware-support/autosar.html)
Support for AUTOSAR Workflows
Capabilities

Simulation

Code Generation

Configuration
Getting Started

- **Bottom-Up Approach**
  Start with an existing Simulink model

- **Top-Down Approach**
  Start with ARXML files containing AUTOSAR Component descriptions
switch(topics)
{

case ‘AUTOSAR – Top 5’:
Embedded Coder® Support Package for AUTOSAR Standard
Embedded Coder® Support Package for AUTOSAR Standard

Embedded Coder® add-on support for the AUTOSAR standard

- Flexible infrastructure to introduce important new capabilities, also in-between half-yearly MathWorks Release cycle

- Perform a wide range of AUTOSAR-related workflows in Simulink®, including:
  - Create and modify an AUTOSAR configuration for a model
  - Model AUTOSAR elements
  - Generate ARXML and AUTOSAR-compatible C code from a model

http://de.mathworks.com/hardware-support/autosar.html
AUTOSAR 4.1.3 / 4.2.1
AUTOSAR 4.1.3 / 4.2.1

Seamless support for AUTOSAR Release 4.2.1 and 4.1.3 schema

- Import detects AUTOSAR 4.2.1 release from ARXML file
- User selects AUTOSAR release from configuration set options for code generation and ARXML export
- AUTOSAR 4.1+ features
  - PRPortPrototype
  - InitEvent
  - …
AUTOSAR Variant Handling
AUTOSAR Variant Handling

Model AUTOSAR variants in Simulink

- VariationPointProxy with
  - Condition Access – Simulink Variant Subsystem
  - Value Access – AUTOSAR.Parameter with CSC System Constant
    - VariationPointProxy objects automatically generated
    - System constant definitions generated in separate ARXML file

```c
/* Gain: '<Root>/Gain' */
if (Rte_SysCon_vpp_liters > 7) {
    tmp = MAX_uint8_T;
} else {
    tmp = (uint8_T)(Rte_SysCon_vpp_liters << 5);
}
```
AUTOSAR Client-Server Semantics
AUTOSAR Client-Server Semantics

Leverage Simulink Functions for AUTOSAR Client/Server

- ARXML import and update support for AUTOSAR Client/Server
  - AUTOSAR client port and operation represented as Function Caller
  - AUTOSAR server runnable represented as Simulink Function
- Use AUTOSAR APPLICATION-ERROR status for C/S communication
  - Helps to Communicate with AUTOSAR basic software that use error status, e.g. Diagnostic Event Manager (DEM)

```c
Std_ReturnType fcnWErr(int8 x1, int8 x2)
{
    if (uh_oh)
    {
        return RTE_E_NOT_OK;
    }
    ..
    return RTE_E_OK;
}

<POSSIBLE-ERRORS>
<APPLICATION-ERROR>
    <SHORT-NAME>E_OK</SHORT-NAME>
    <ERROR-CODE>0</ERROR-CODE>
</APPLICATION-ERROR>
<APPLICATION-ERROR>
    <SHORT-NAME>E_NOT_OK</SHORT-NAME>
    <ERROR-CODE>1</ERROR-CODE>
</APPLICATION-ERROR>
</POSSIBLE-ERRORS>
```
Advanced AUTOSAR APIs
Advanced AUTOSAR APIs

Simulink modeling and ARXML roundtrip support for

- RTE APIs
  - Conditional Rte_Read / Rte_Write
  - Rte_IsUpdated
  - Rte_Invalidate
- Asynchronous NvM Service calls
  - NvM_WriteBlock, NvM_ReadBlock, …
- E2E wrapper
- MFX / MFL / IFX / IFL Library Routines
- ReferenceBase Support
- …

```c
if (Rte_IsUpdated_Input_Input()) {
    Rte_Read_Input_Input(&tmp);
    ...
}
```

```xml
<NONQUEUED-RECEIVER-COM-SPEC>
  ...
  <ENABLE-UPDATE>true</ENABLE-UPDATE>
</NONQUEUED-RECEIVER-COM-SPEC>
```
case ‘AUTOSAR – Best Practices’:
Use one AUTOSAR workflow
#1 Use one AUTOSAR workflow
- Select top-down or bottom-up approach
- Round-trip works best with one clear owner of data

- Select tools that best support your workflow and AUTOSAR concepts
- Select simplest approach for applying AUTOSAR configuration to your Simulink model
Decide data management
#2 Decide data management

- Will Simulink or AUTOSAR tools manage data?
- Will projects or teams define and manage data?
- How will change management be handled?
Establish modeling standards
#3 Establish modeling standards
- For Simulink and AUTOSAR

- Base it on your workflow and data management
- Use Simulink Model Advisor to enforce modeling style early in model development
Simulate before you generate code
#4 Simulate before you generate code
- Take advantage of early verification through simulation

- Make sure SWC implementation is correct early
- Simulate multiple SWC’s together in Simulink before code integration
- Use SIL and PIL to verify generated code at the unit level before RTE generation
Plan ahead for ISO 26262
#5 Plan ahead for ISO 26262
- Determine how your AUTOSAR process will address safety-standards

- **Products supported for ISO 26262 tool qualification include:**
  - Embedded Coder
  - Simulink V&V
  - Simulink Design Verifier
  - PolySpace Code Verifiers

- **Artifacts certified by TÜV SÜD**
  - Requires use of V&V workflow

- **ISO 26262 Advisory Service available**
Use Simulink to help migrate your legacy code to AUTOSAR
#6 Use Simulink to help migrate your legacy code to AUTOSAR

- Reuse of Legacy Code
  - Integration for simulation, production code generation
  - Can generate AUTOSAR RTE API access points

```c
void Runnable_Runnable1(void)
{
    real32_T rtb_TmpSignalConversionAtIn1Out;
    real32_T rtb_UnitDelay;
    real32_T rtb_sldemo_sfun_filterV1;
    rtb_TmpSignalConversionAtIn1Out = Rte_IRead_Runnable_Runnable1_Fast_in_Fast_in();
    rtb_UnitDelay = Component_DWork.UnitDelay_DSTATE;
    rtb_sldemo_sfun_filterV1 = filterV1((real32_T)rtb_TmpSignalConversionAtIn1Out,
                                        (real32_T)rtb_UnitDelay,
                                        (real32_T)Component_P.sldemo_sfun_filterV1_p1);
    Rte_IrvIWrite_Runnable_Runnable1_a(rtb_sldemo_sfun_filterV1);
    Component_DWork.UnitDelay_DSTATE = rtb_sldemo_sfun_filterV1;
}
```
Automate, automate, automate
#7 Automate, automate, automate
- Use API’s for workflow automation!

- **Manual process difficult due to:**
  - The complexity of the standard, naming conventions
  - Iterative work cycles with AUTOSAR
  - Complex code APIs and XML file definitions

- Use documented MATLAB APIs to configure SWCs in Simulink

```matlab
%% Setup AUTOSAR Configuration programmatically

model = 'rtwdemo_autosar_counter';

% Modify AUTOSAR Properties
autosarProps = autosar.api.getAUTOSARProperties(model);
set(autosarProps, 'Input', 'IsService', true);
set(autosarProps, 'XmlOptions',
    'ArxmlFilePackaging','SingleFile');
```
Use production code generation
#8 Use production code generation
- Hand coding AUTOSAR is painful

```c
void Runnable_simple_alg_Step(void)
{
    real_T rtb_Gain;
    real_T rtb_Delay;
    real_T rtb_Delay1;
    real_T rtb_TmpSignalConversionAtFast_i;
    if ((simple_alg_M->Timing.TaskCounters.TID[1] == 0) {  
        Rte_Receive_Fast_in_Fast_in(&rtb_TmpSignalConversionAtFast_i);
        rtb_Delay = simple_alg_DWork.Delay_DSTATE;
        rtb_Delay1 = simple_alg_DWork.Delay1_DSTATE;
        rtb_Gain = simple_alg_DWork.Delay2_DSTATE;
        if ((simple_alg_M->Timing.TaskCounters.TID[2] == 0) {  
            simple_alg_B.RateTransition = rtb_Gain;
        }
        simple_alg_DWork.Delay_DSTATE = rtb_TmpSignalConversionAtFast_i;
        simple_alg_DWork.Delay1_DSTATE = rtb_Delay;
        simple_alg_DWork.Delay2_DSTATE = rtb_Delay1;
    }  
    if ((simple_alg_M->Timing.TaskCounters.TID[2] == 0) {  
        Rte_IWriteRunnable_simple_alg_Step_Out1_Out1(simple_alg_B.RateTransition + Rte_IReadRunnable_simple_alg_Step_Slow_in_Slow_in());
    }
}
...
```
Active plan for migration
#9 Actively plan for migration
- Tools and standards are changing rapidly

- Account for:
  - New versions of AUTOSAR
  - New versions of Simulink

- Consider:
  - How often to upgrade
  - What will drive upgrade

Source: 7th AUTOSAR Open Conference, 22.10.2014
case ‘Assistance’:
Training Services

Developing Embedded Targets
Advisory Service
case ‘AUTOSAR Demo Pod’:
default :
    printf("Brain up-to-date!");
}
And one last thing …
AUTOSAR – Antagonizing the „German Coast Guard“ Effect

Source: https://youtu.be/zkalf0odHs8 German Coast Guard Commercial ‘We are Sinking’ [HD]