Using the Benefits of Model-Based Design to Develop AUTOSAR Basic Software Modules

Mohamed Soliman & Amjad Elshenawy

Mathworks Automotive Conference 2016
Stuttgart, Sep., 21st 2016
Agenda

1. Why use MBD for Developing AUTOSAR BSW Modules?
2. CAN State Manager (CanSM)
3. Challenges Encountered in Developing CanSM using MBD
4. Results of Our Experiment
Agenda

1. Why use MBD for Developing AUTOSAR BSW Modules?
2. CAN State Manager (CanSM)
3. Challenges Encountered in Developing CanSM using MBD
4. Results of Our Experiment
AUTOSAR Embraces Complexity

Number of Basic SW Modules

<table>
<thead>
<tr>
<th>Version</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1.0</td>
<td>2005</td>
</tr>
<tr>
<td>R2.0</td>
<td>2006</td>
</tr>
<tr>
<td>R2.1</td>
<td>2006</td>
</tr>
<tr>
<td>R3.0</td>
<td>2007</td>
</tr>
<tr>
<td>R3.1</td>
<td>2008</td>
</tr>
<tr>
<td>R3.2</td>
<td>2011</td>
</tr>
<tr>
<td>R4.0</td>
<td>2011</td>
</tr>
<tr>
<td>R4.1</td>
<td>2014</td>
</tr>
<tr>
<td>R4.2</td>
<td>2015</td>
</tr>
</tbody>
</table>

- Number of Basic SW Modules:
  - R1.0: 1X
  - R2.0: 33
  - R2.1: 45
  - R3.0: 51
  - R3.1: 53
  - R3.2: 53
  - R4.0: 55
  - R4.1: 80
  - R4.2: 89
  - Total: 98
Characteristics of AUTOSAR Basic Software Modules

- **Pre-compile Configuration**: Enabling/disabling optional functionality
- **Link-time Configuration**: Configuration of modules that are only available as object code (e.g. IP protection)
- **Post-build Configuration**: Change the configuration after building the code (in the run time)

Highly Configurable
Characteristics of AUTOSAR Basic Software Modules

- Standard Interfaces and Standard Types
Motivations for using MBD for Developing AUTOSAR BSW Modules

In our case MBD is selected to provide the following benefits:

- Shorter development time
- Better re-usability and maintainability of design / model.
- Improvement of the product quality
Agenda

1. Why use MBD for Developing AUTOSAR BSW Modules?
2. CAN State Manager (CanSM)
3. Challenges Encountered in Developing CanSM using MBD
4. Results of Our Experiment
One of the basic software communication stack modules.

Responsible for managing the states of the Can networks.
CAN State Manager

Module Complexity

- 280 requirements.
- 26 Configuration parameters.
- 18 Provided Interfaces.
Agenda

1. Why use MBD for Developing AUTOSAR BSW Modules?
2. CAN State Manager (CanSM)
3. Challenges Encountered in Developing CanSM using MBD
4. Results of Our Experiment
SWS_BSW_00029: If the BSW Module contains optional functionality, then this functionality shall be enabled (STD_ON) or disabled (STD_OFF) by a Pre-compile time configuration parameter.

```c
if (CanSM_u8eNetRept[u8NetId] > CanSM_pkstreGlobalConfig->ku8ModeReqRepMax)
{
  #if CanSMDevErrorDetect == STD_ON
  Det_ReportError(CANSM_MODULE_ID, CanSM_u8INSTANCE_ID,
                  CanSM_u8MAIN_FUNCTION_ID, CANSM_E_MODE_REQUEST_TIMEOUT);
  #endif

  CanSM_u8eCurInd[u8NetId] = (uint8_T)CanSM_u8NET_REPT;
  CanSM_u8eNetRept[u8NetId] = 0U;
}
```
Pre-compile Configuration

Using “Variant Subsystem” to generate pre-compile configuration
Pre-compile Configuration

- Generate preprocessor conditional for with variant model blocks.

<table>
<thead>
<tr>
<th>Configuration Parameters: CanSM_Model/Configuration (Active)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Select:</strong></td>
</tr>
<tr>
<td>Solver</td>
</tr>
<tr>
<td>Simulation Target</td>
</tr>
<tr>
<td>Interface</td>
</tr>
<tr>
<td>Software environment</td>
</tr>
<tr>
<td>Standard math library:</td>
</tr>
<tr>
<td>Code replacement library:</td>
</tr>
<tr>
<td>Shared code placement:</td>
</tr>
<tr>
<td>Support:</td>
</tr>
<tr>
<td>☐ floating-point numbers</td>
</tr>
<tr>
<td>☐ absolute time</td>
</tr>
<tr>
<td>☐ variable-size signals</td>
</tr>
<tr>
<td>Multiverse type definitions:</td>
</tr>
<tr>
<td>Code interface</td>
</tr>
<tr>
<td>Code interface packaging:</td>
</tr>
<tr>
<td>☑ Single output/update function</td>
</tr>
<tr>
<td>Generate preprocessor conditionals:</td>
</tr>
<tr>
<td>Suppress error status in real-time model data structure</td>
</tr>
<tr>
<td>Configure Model Functions</td>
</tr>
</tbody>
</table>
SRS_Can_01142: The CAN State Manager shall offer a network abstract API to upper layer.

Example scenario: "Network status change upon Communication Manager module (ComM) request"
Standard Interfaces

CanSM_MainFunction

Block Parameters: CanSM_MainFunction
- Subsystem
- Function packaging: Nonreusable function
- Function name options: User specified
- Function name: CanSM_MainFunction
- File name options: Auto
- Function interface: void
- Memory section for initialize/terminate functions: Inherit from model
- Memory section for execution functions: Inherit from model
SWS_BSW_00127: The BSW Module implementation shall avoid duplication of code.
Code Duplication

- Using library of atomic sub-chart to avoid code duplication.
Compliance with MISRA C Rules

**SWS_BSW_00115:** If the BSW Module implementation is written in C language, then it shall conform to the MISRA C 2004 Standard

- Source complexity (Cyclomatic Complexity): Number of linearly independent paths should not exceed a certain limit.

- Implicit and explicit type conversions (Casting). Example: casting from integer to pointer is prohibited.

- Parentheses “(” and “)“ should be used to emphasis expressions.

- The final clause of a switch statement shall be the default clause.
Compliance with MISRA C Rules

Cyclomatic Complexity control by separating atomic parts in separate functions
Compliance with MISRA C Rules

- Implicit and explicit type conversions (Casting)
- Parentheses level
- The final clause of a switch statement
Non-Functional Requirements

- Maintainability
- Reusability
- Reliability
- Efficiency (Execution Time, Memory consumption, …)
Non-Functional Requirements

- Execution time: Switch Case Vs If Else

Expression

Case #1
- Statement #1

Case #2
- Statement #2

Case #n
- Statement #n

Exp #1
- True
  - Statement #1
- False
  - Exp #2
    - True
      - Statement #2
    - False
      - Exp #n
        - True
          - Statement #n
        - False
          - Default Statement
Non-Functional Requirements

- Execution time optimization: Code generation with Switch Case instead of If Else

```c
/* During 'NoFnPrNoCom': '<S304>:1' */
switch (localDN->u0_is_NoFnPrNoCom)
{
    case CanSM_Model_IN_NFnCtrlSp:
        NFnCtrlSp(u0NetId, localDN);
        break;
    case CanSM_Model_IN_NFnCtrlStp:
        NFnCtrlStp(u0NetId, localDN);
        break;
    case CanSM_Model_IN_NFnTrcNor:
        NFnTrcNor(u0NetId, localDN);
        break;
    case CanSM_Model_IN_NFnTrcStd:
        NFnTrcStd(u0NetId, localDN);
        break;
    default:
        /* During 'a': '<S304>:2' */
        break;
}
```
Smoke testing is non-exhaustive software testing, ascertaining that the most crucial functions of a program work, but not bothering with finer details.

Smoke testing is not a substitute for traditional testing mechanism.
Smoke Testing

- Attaching Microsoft Visual Studio to Matlab process.
Smoke Testing

- Debugging in the Model and the manual code.
Agenda

1. Why use MBD for Developing AUTOSAR BSW Modules?
2. CAN State Manager (CanSM)
3. Challenges Encountered in Developing CanSM using MBD
4. Results of Our Experiment
Results of The Provided Solution

- Development time is about 18% less than the other manually developed modules with similar size.

- Bug fixing is about 34% shorter than the other manually developed modules with similar size.

- Number of issues found during testing phase is about 30% less than the other manually developed modules with similar size.