



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*

September 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**

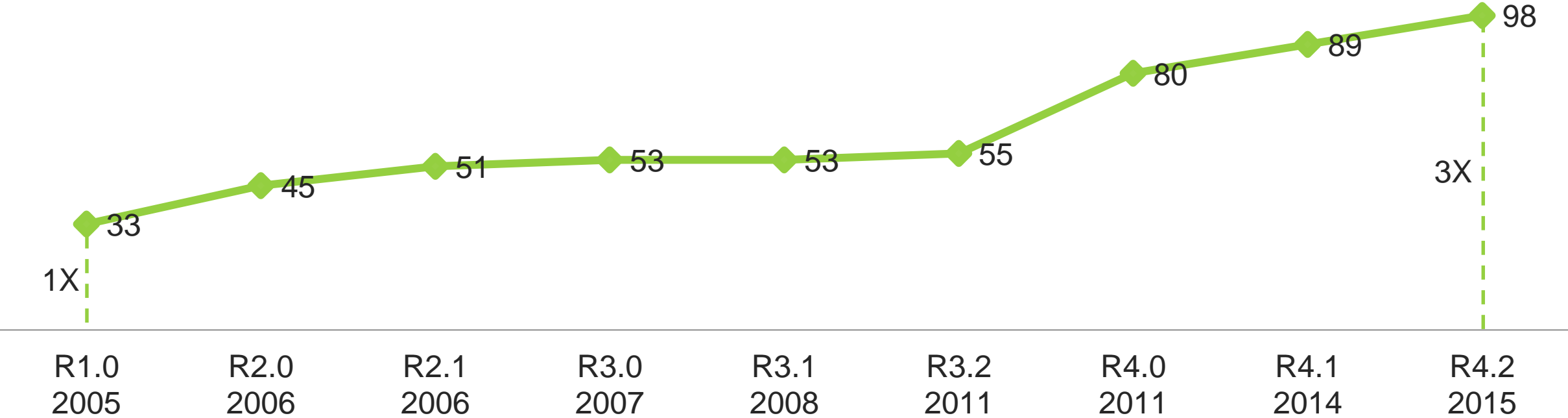
Agenda

- 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

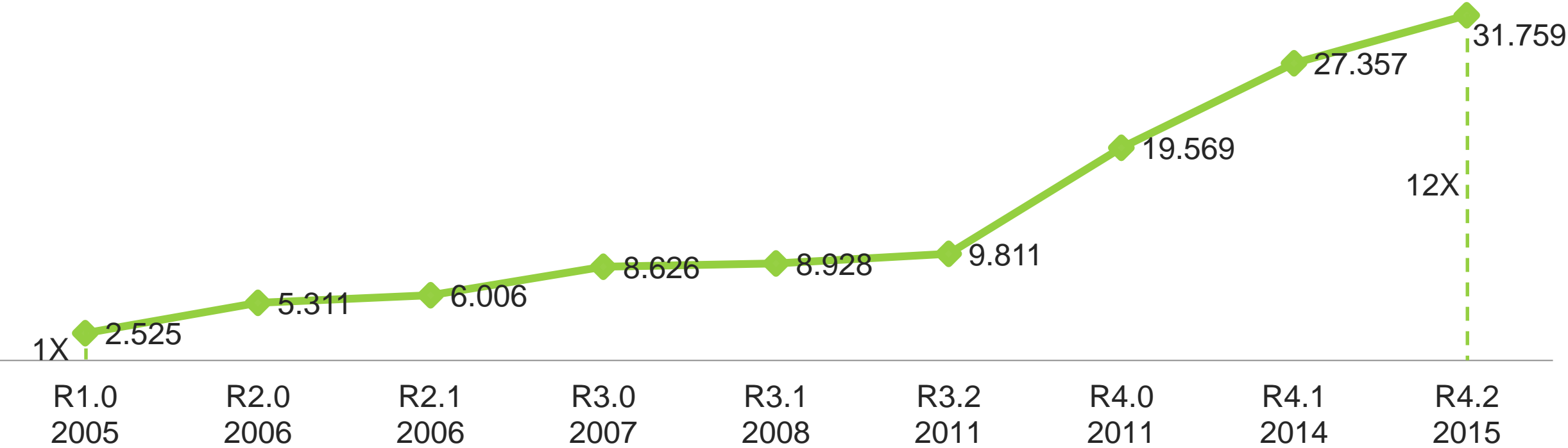
◆ Number of Basic SW Modules



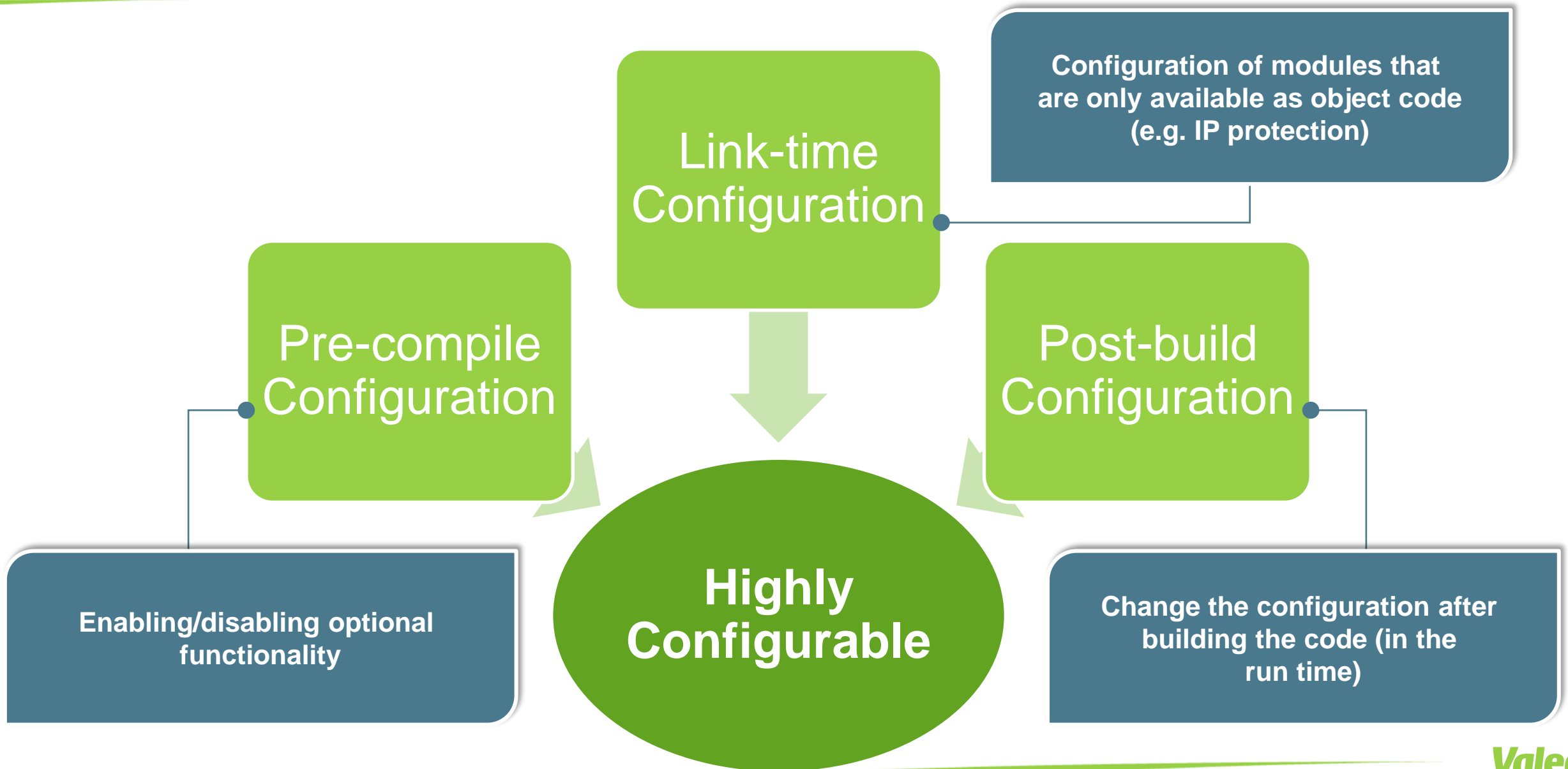
AUTOSAR Embraces Complexity

Number of Requirements

◆ Number of Requirements

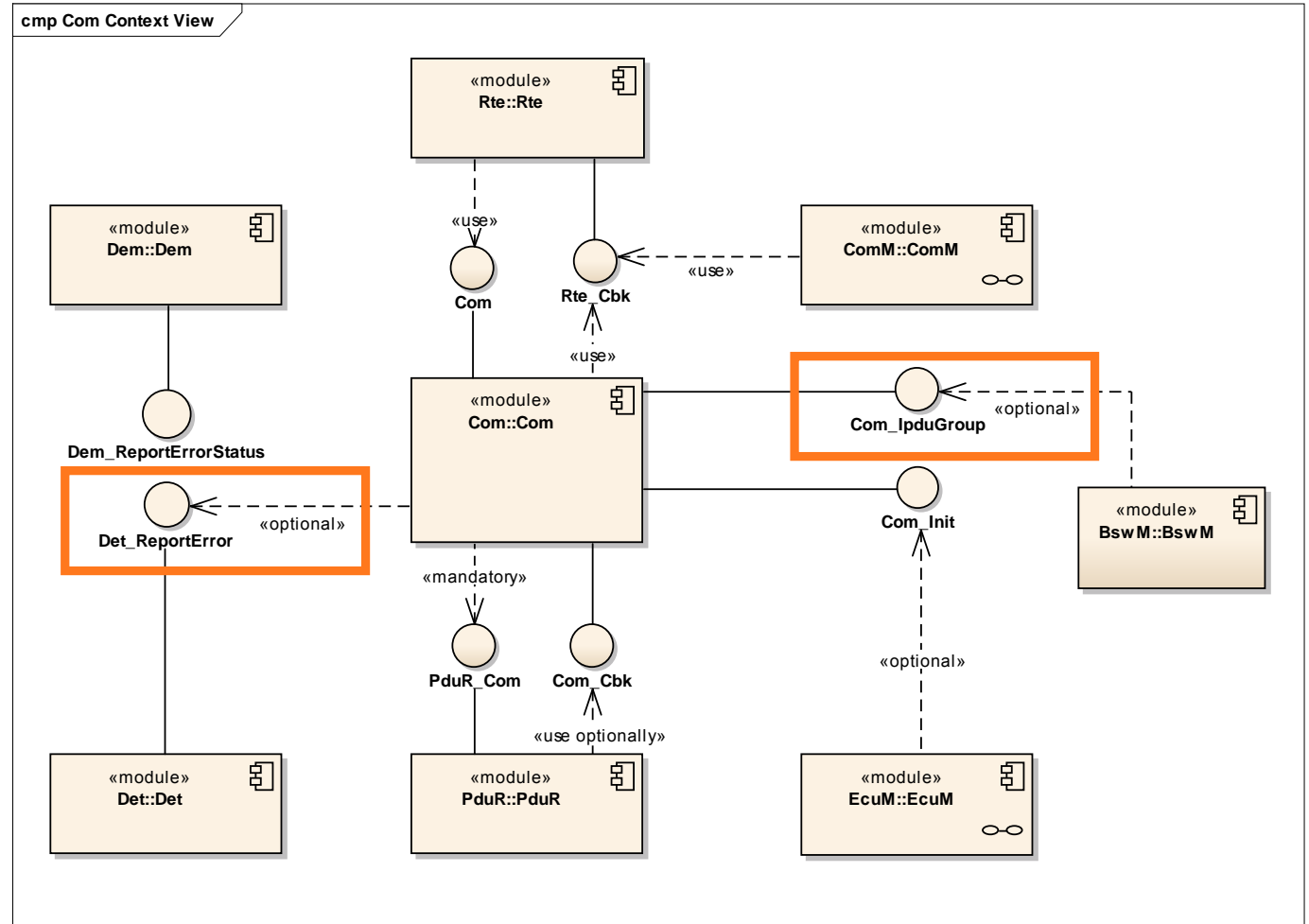


Characteristics of AUTOSAR Basic Software Modules



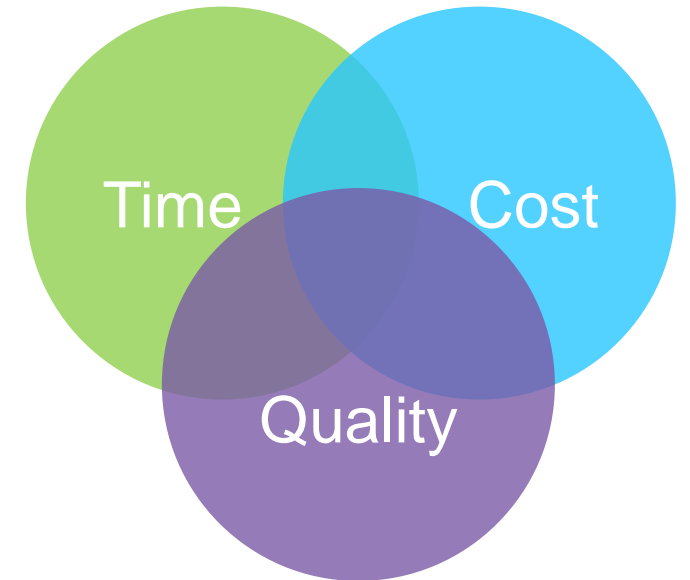
Characteristics of AUTOSAR Basic Software Modules

- 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

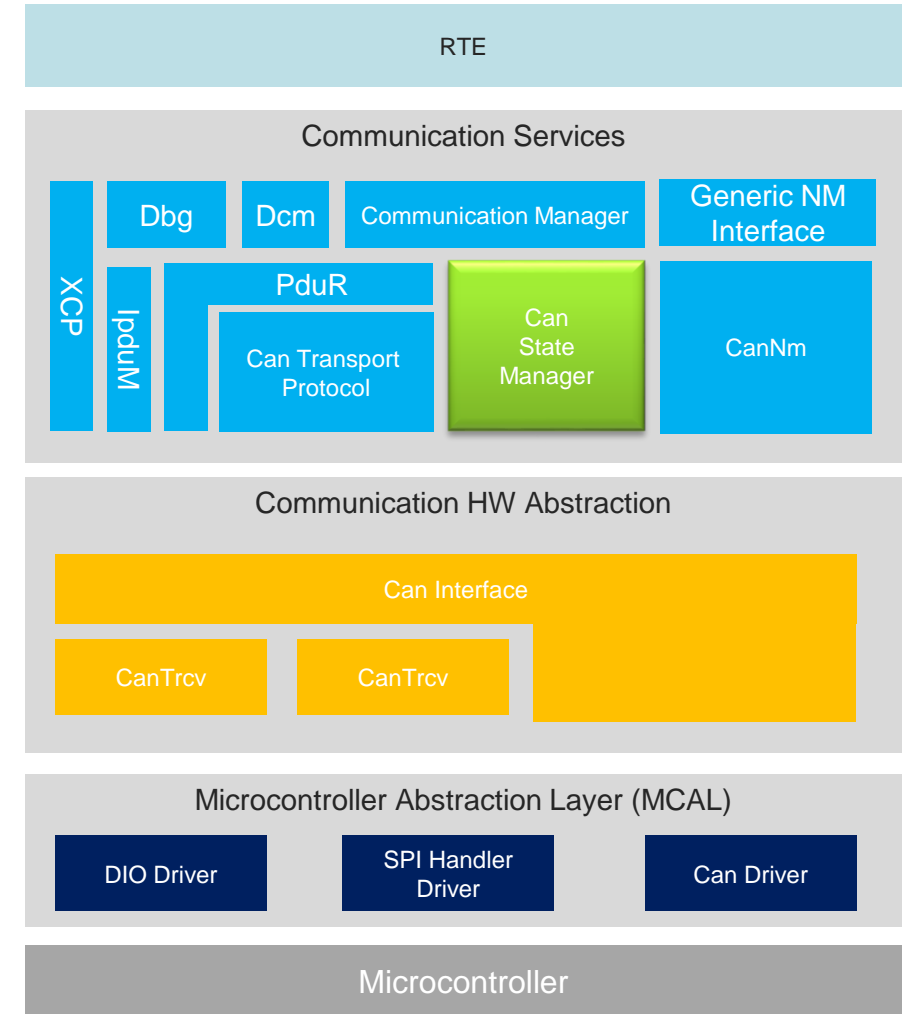
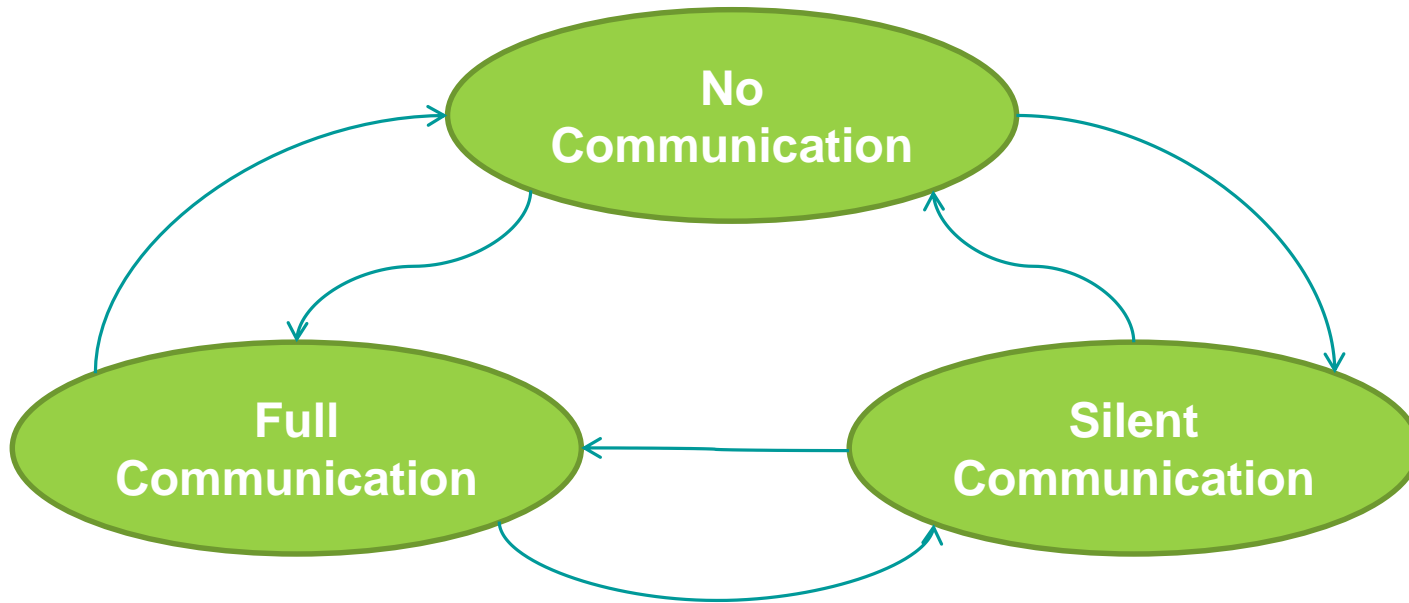


Agenda

- 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

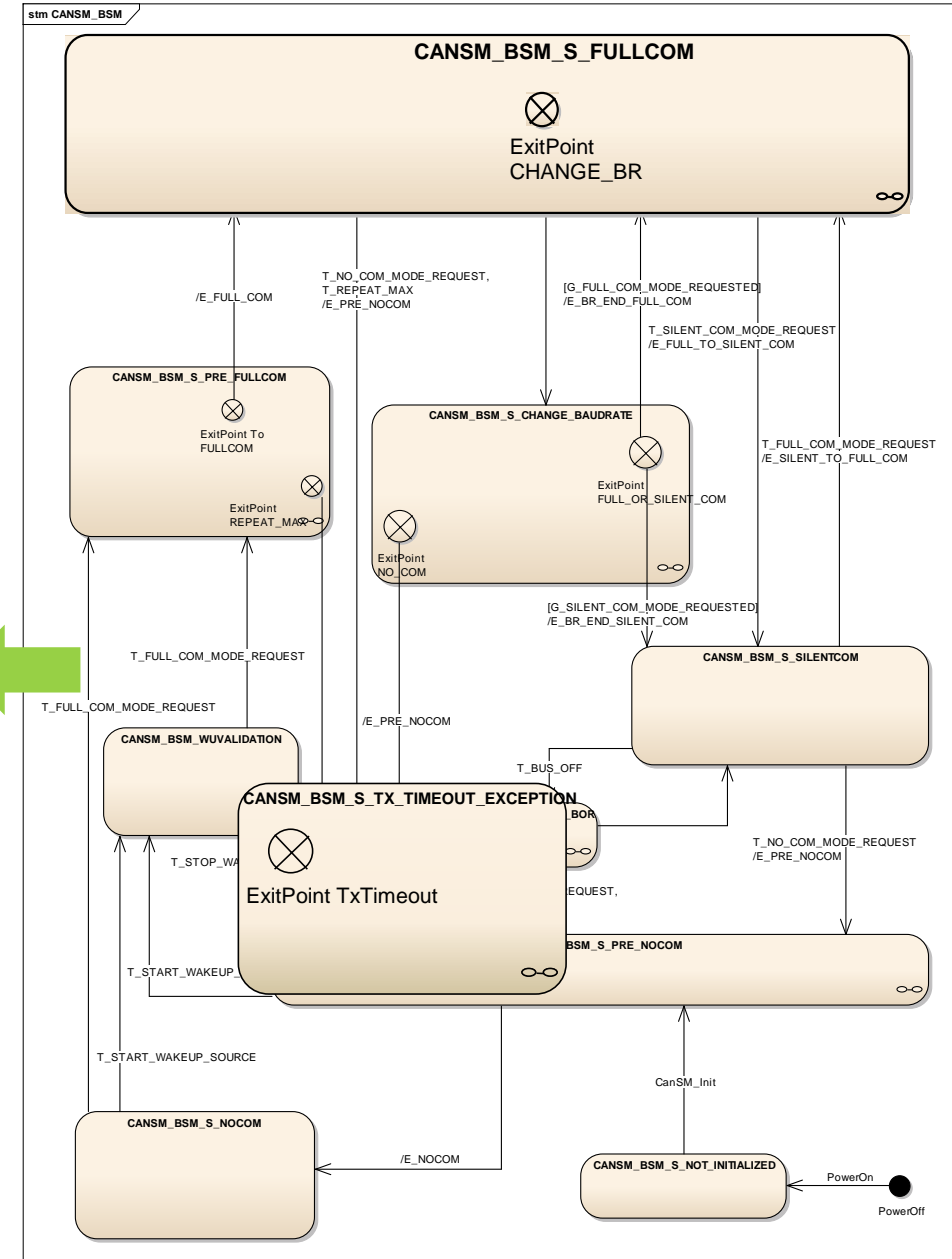
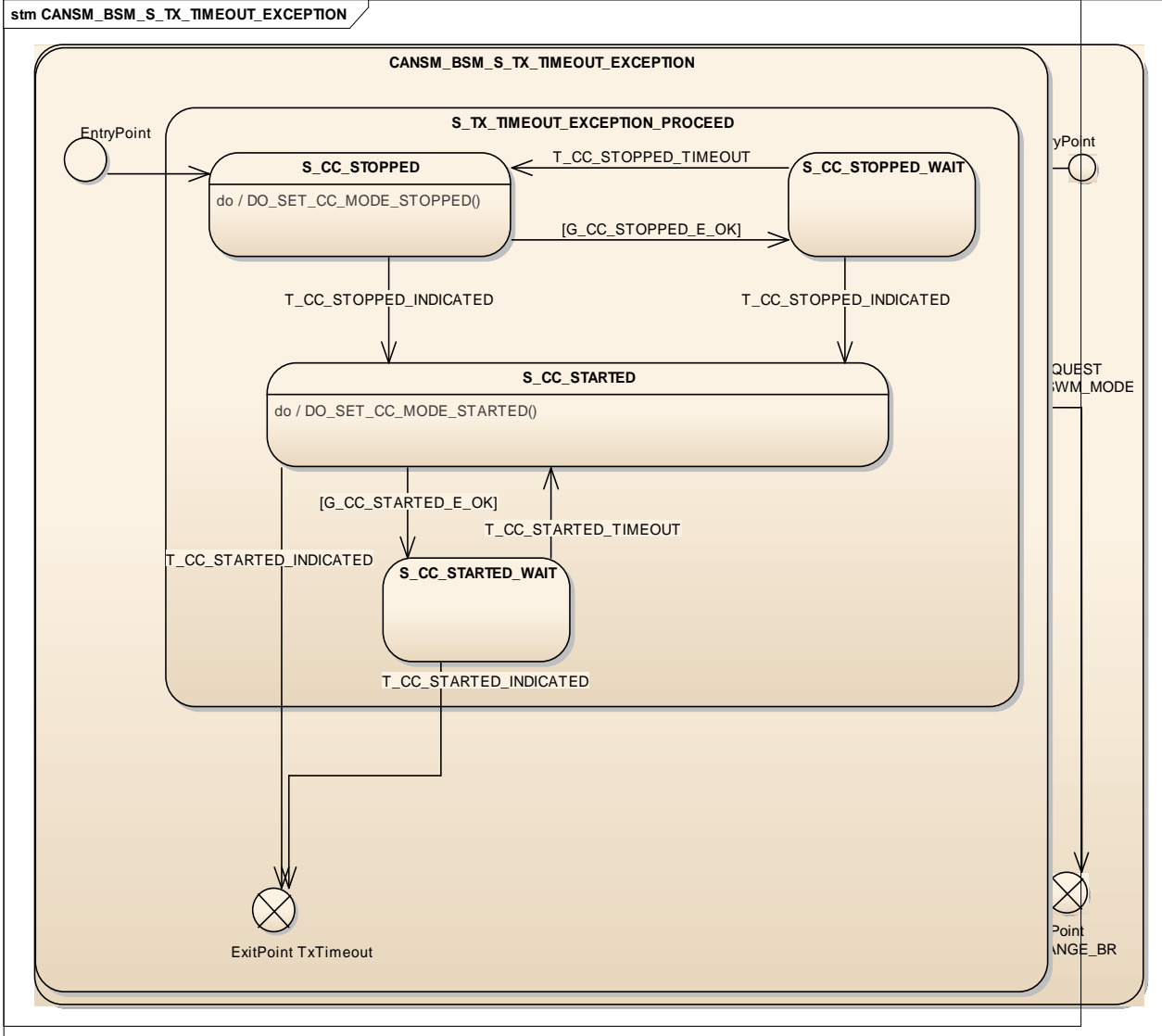
CAN State Manager

- One of the basic software communication stack modules.
- Responsible for managing the states of the Can networks.



CAN State Manager

State Machine Complexity



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

Pre-compile Configuration

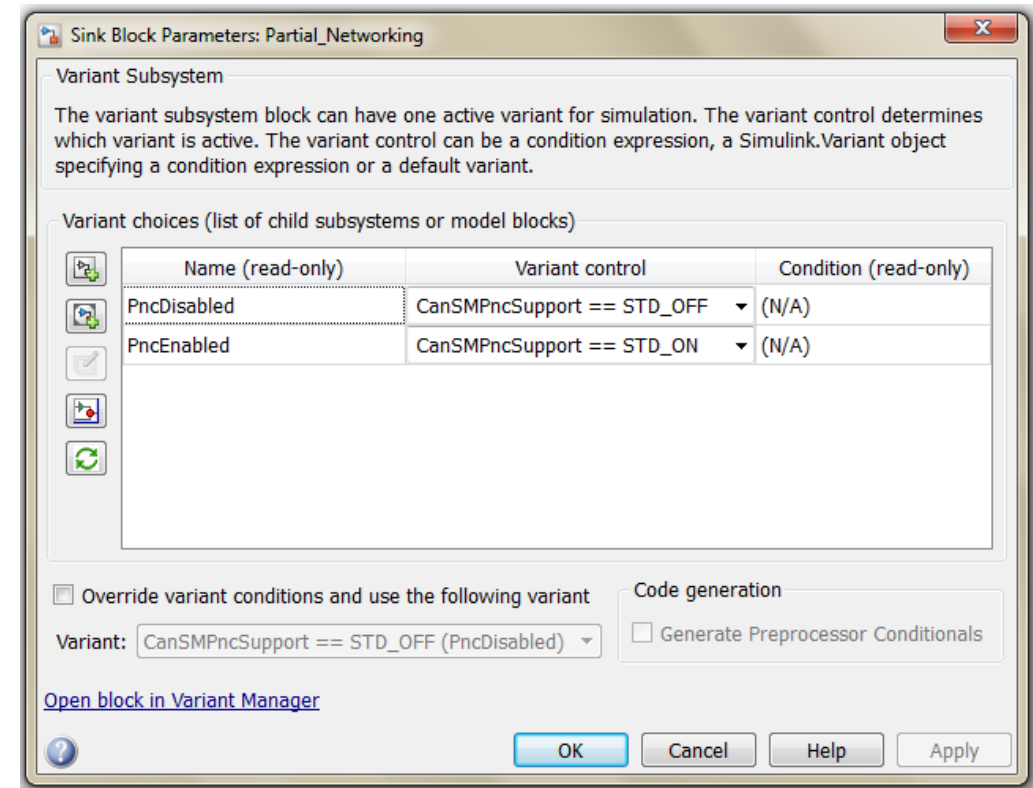
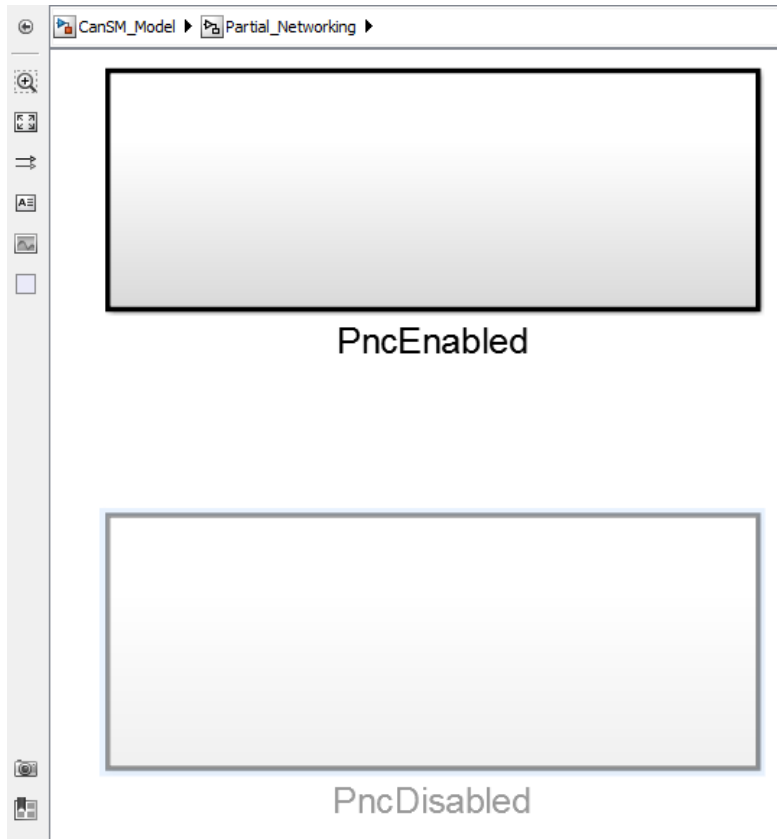
SWS_BSW_00029: If the BSW Module contains optional functionality, then this functionality shall be enabled (STD_ON) or disabled (STD_OFF) by a Pre-compile time configuration parameter.



```
652     if (CanSM_au8eNetRept[u8NetId] >
653         CanSM_pkstreGlobalConfig->ku8ModeReqRepMax)
654     {
655
656 #if CanSMDevErrorDetect == STD_ON
657
658     Det_ReportError(CANSM_MODULE_ID, CanSM_u8INSTANCE_ID,
659                   CanSM_u8MAIN_FUNCTION_ID, CANSM_E_MODE_REQUEST_TIMEOUT)
660
661 #endif
662
663     CanSM_au8eCurInd[u8NetId] = (uint8_T)CanSM_u8NET_REPT;
664     CanSM_au8eNetRept[u8NetId] = 0U;
665 }
666
```

Pre-compile Configuration

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



Pre-compile Configuration

- Generate preprocessor conditional for with variant model blocks.

Configuration Parameters: CanSM_Model/Configuration (Active)

Select:

- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
 - Symbols
 - Custom Code
- Code Generation
 - Report
 - Comments
 - Symbols
 - Custom Code
 - Debug
 - Interface**
 - Verification
 - Code Style
 - Templates
 - Code Placement
 - Data Type Replacement
 - Memory Sections

Software environment

Standard math library: C89/C90 (ANSI)

Code replacement library: None

Shared code placement: Shared location

Support: floating-point numbers non-finite numbers
 absolute time continuous time
 variable-size signals

Multiword type definitions: User defined

Code interface

Code interface packaging: Nonreusable function

Classic call interface
 Single output/update function Terminate function required

Generate preprocessor conditionals: Enable all

Suppress error status in real-time model data structure Combine signal/state structures

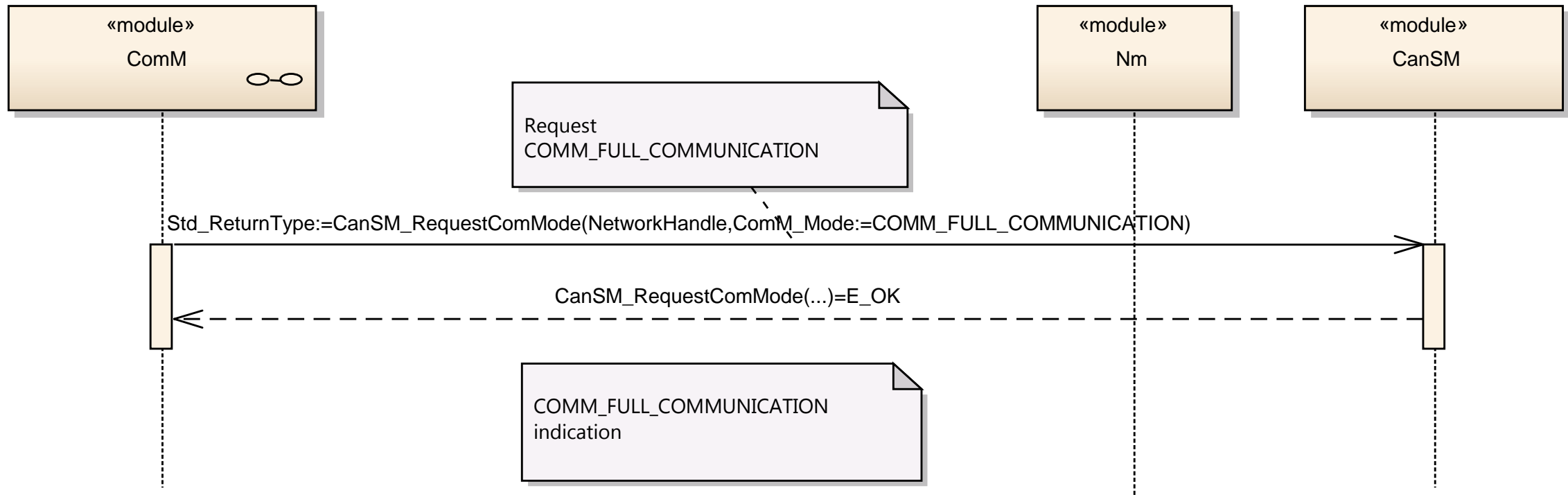
Configure Model Functions

Standard Interfaces

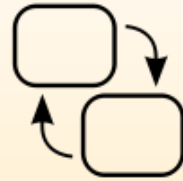


SRS_Can_01142 : The CAN State Manager shall offer a network abstract API to upper layer

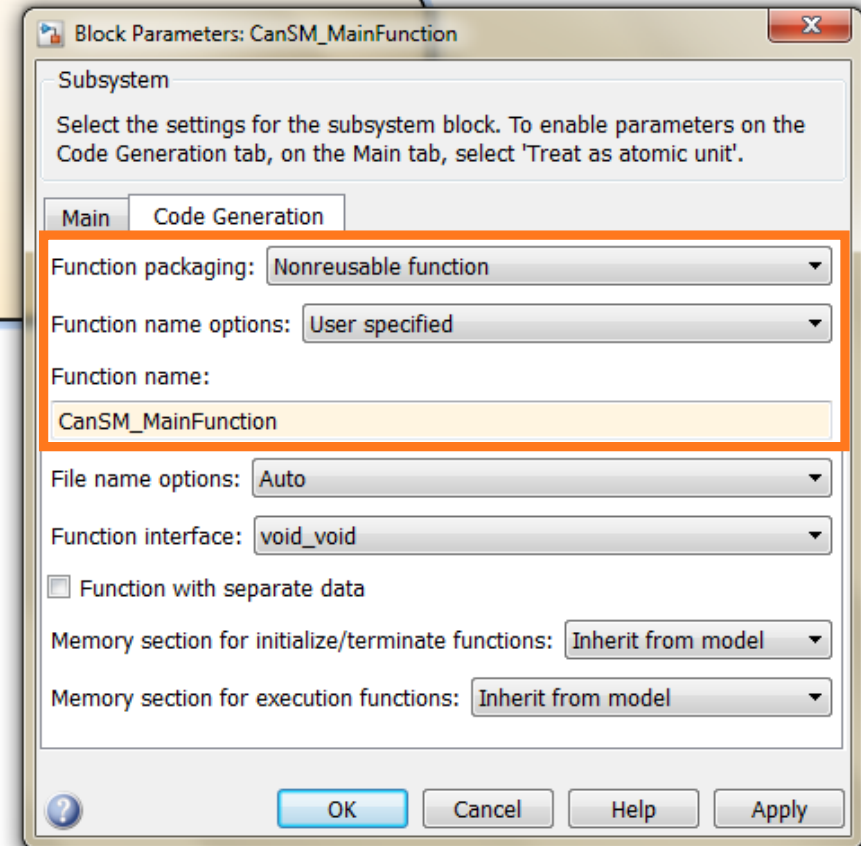
Example scenario: "Network status change upon Communication Manager module (ComM) request"



Standard Interfaces

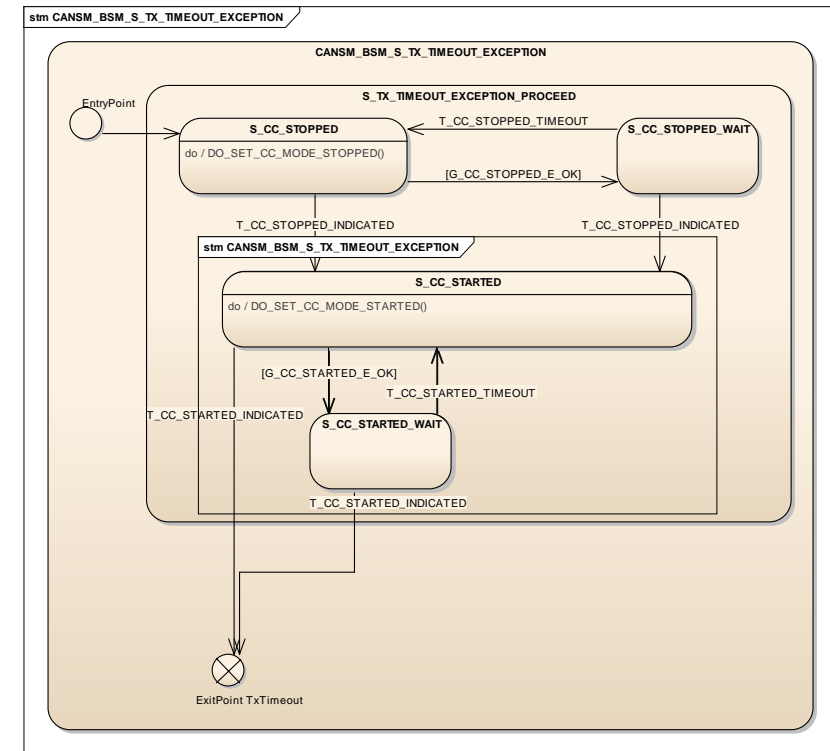
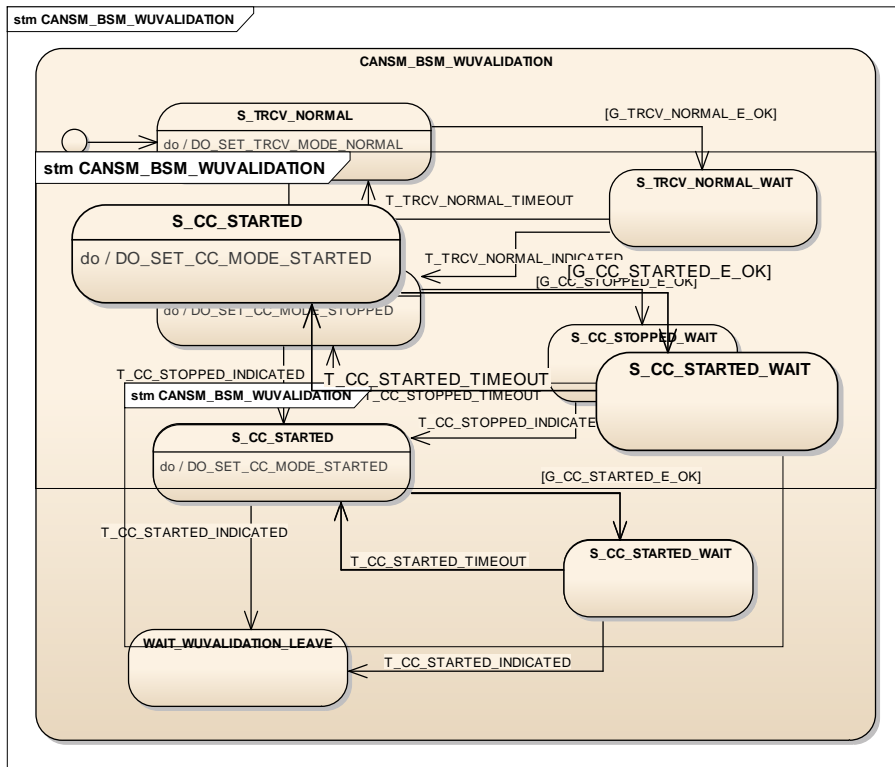


CanSM_MainFunction



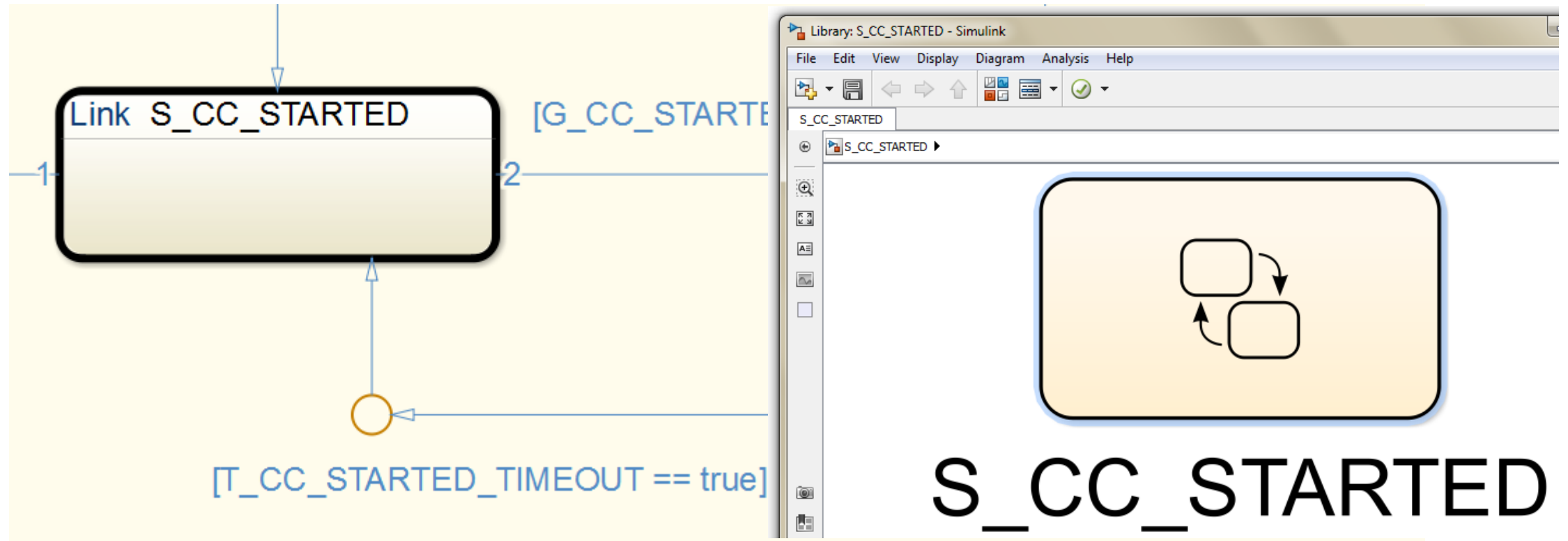
Code Duplication

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

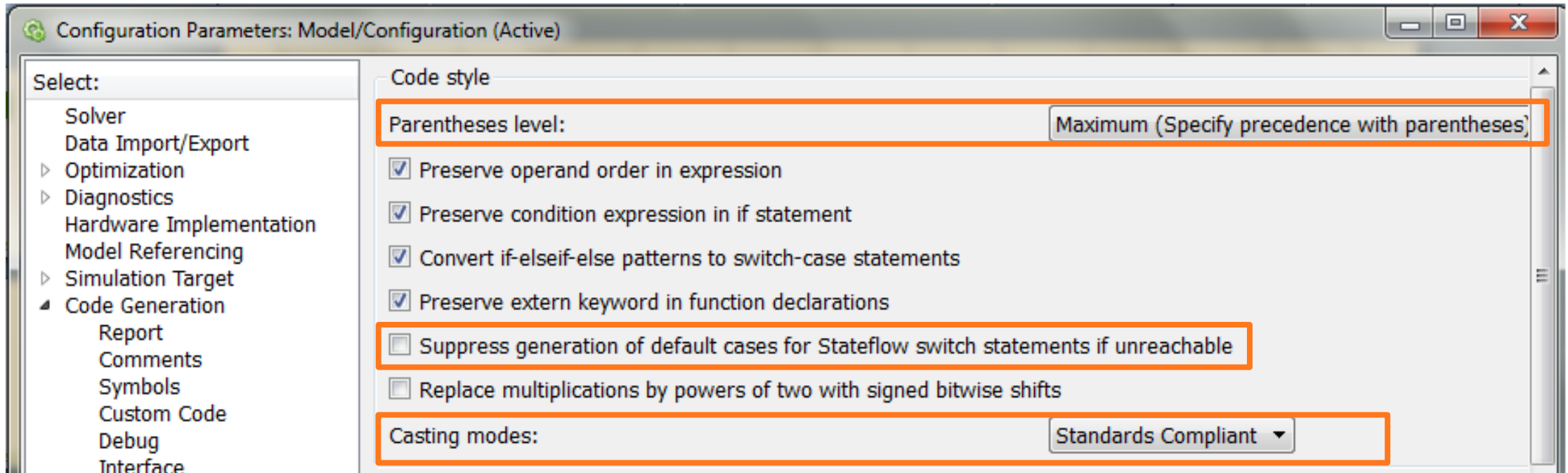
The image illustrates a software development practice for controlling cyclomatic complexity. On the left, a state machine diagram shows a sub-state named `CANSM_BSM_DeinitPnSupported`. A transition arrow points to this sub-state with the label `/* Enter sub state CANSM_BSM_DeinitPnSupported */`. On the right, a configuration dialog box titled `State CANSM_BSM_DeinitPnSupported` is shown. The dialog has three tabs: `General`, `Logging`, and `Documentation`. The `General` tab is active, showing the following settings:

- Name: `CANSM_BSM_DeinitPnSupported`
- State Output: Create output port for monitoring: `Child activity`
- Function Inline Option: `Function` (highlighted with an orange box)
- Label: `CANSM_BSM_DeinitPnSupported`

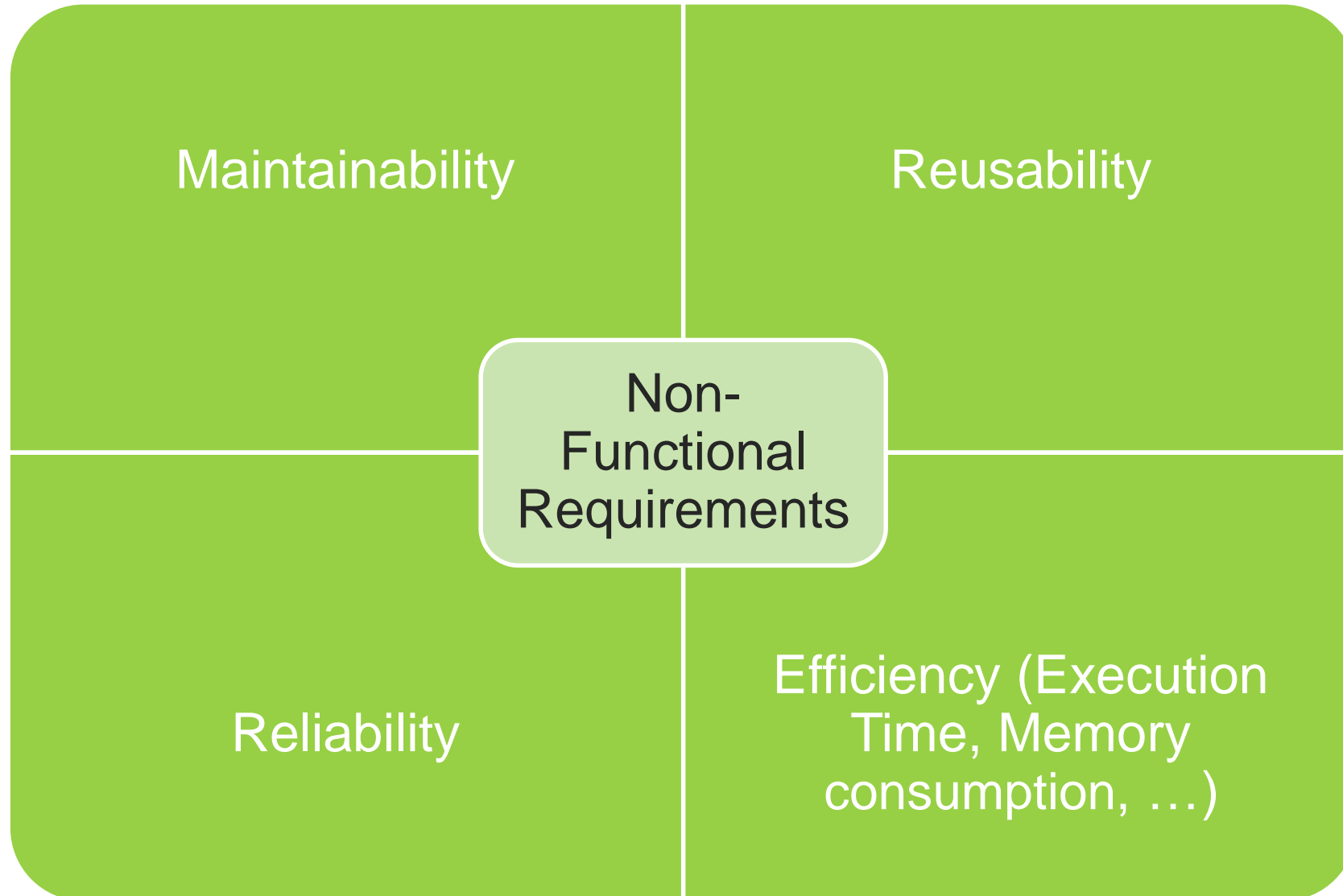
Buttons at the bottom of the dialog include `OK`, `Cancel`, `Help`, and `Apply`.

Compliance with MISRA C Rules

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

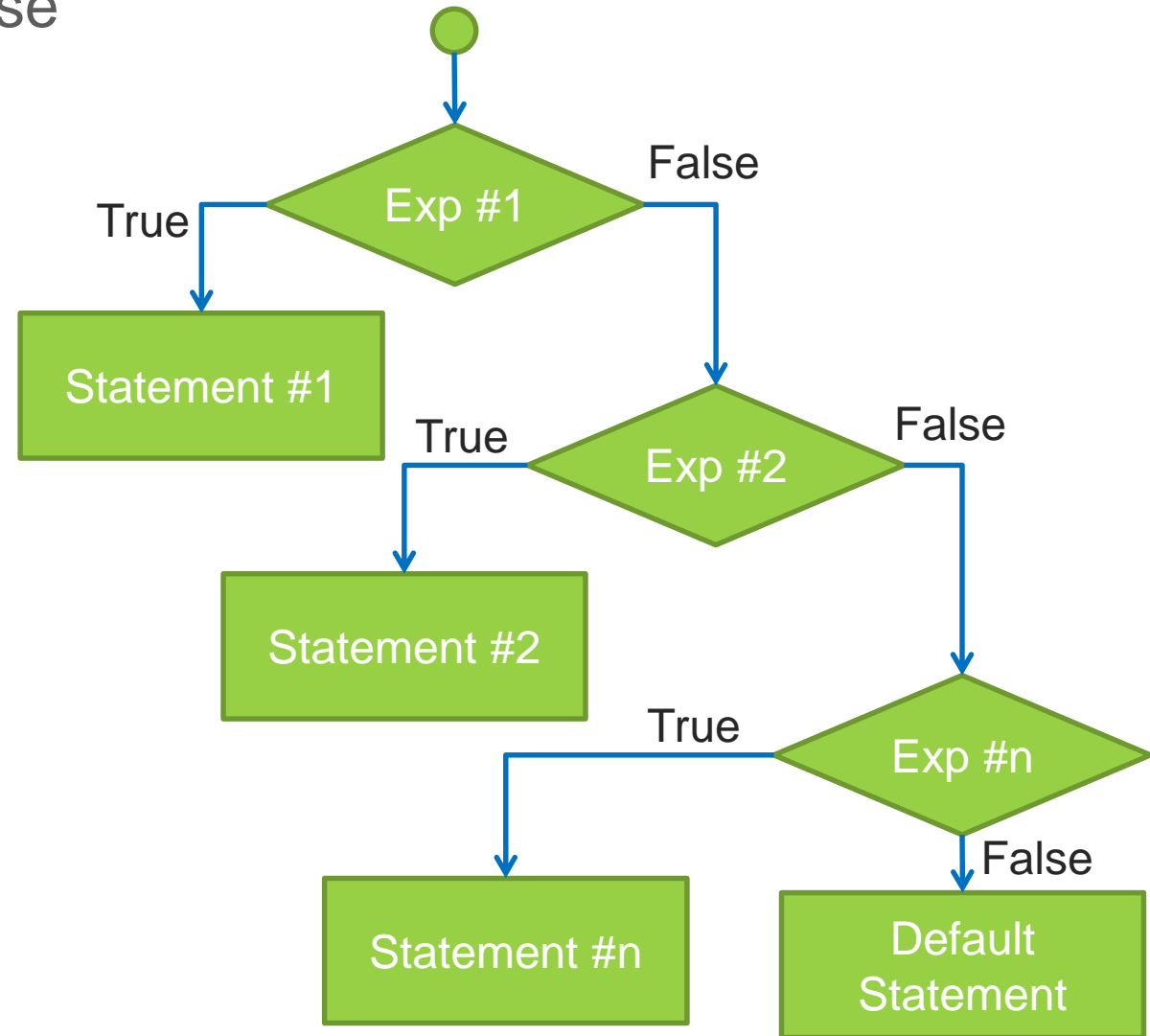
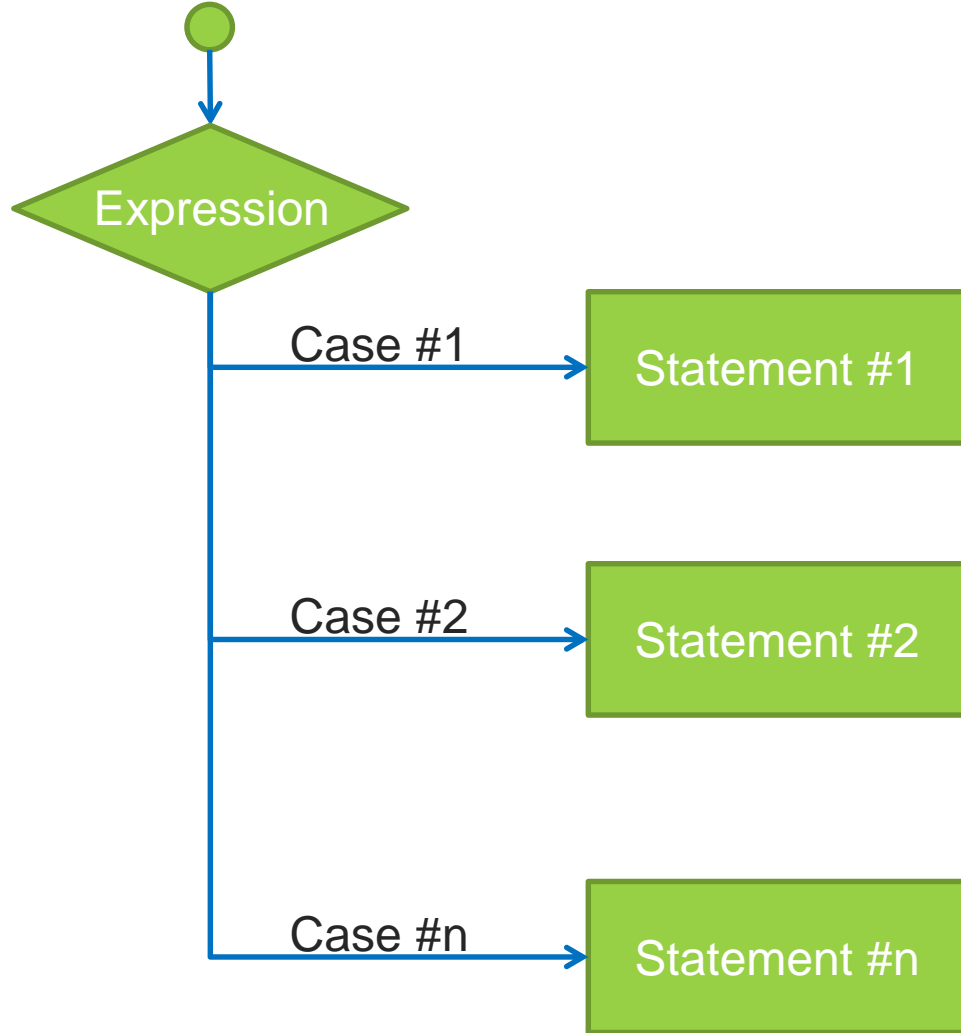


Non-Functional Requirements



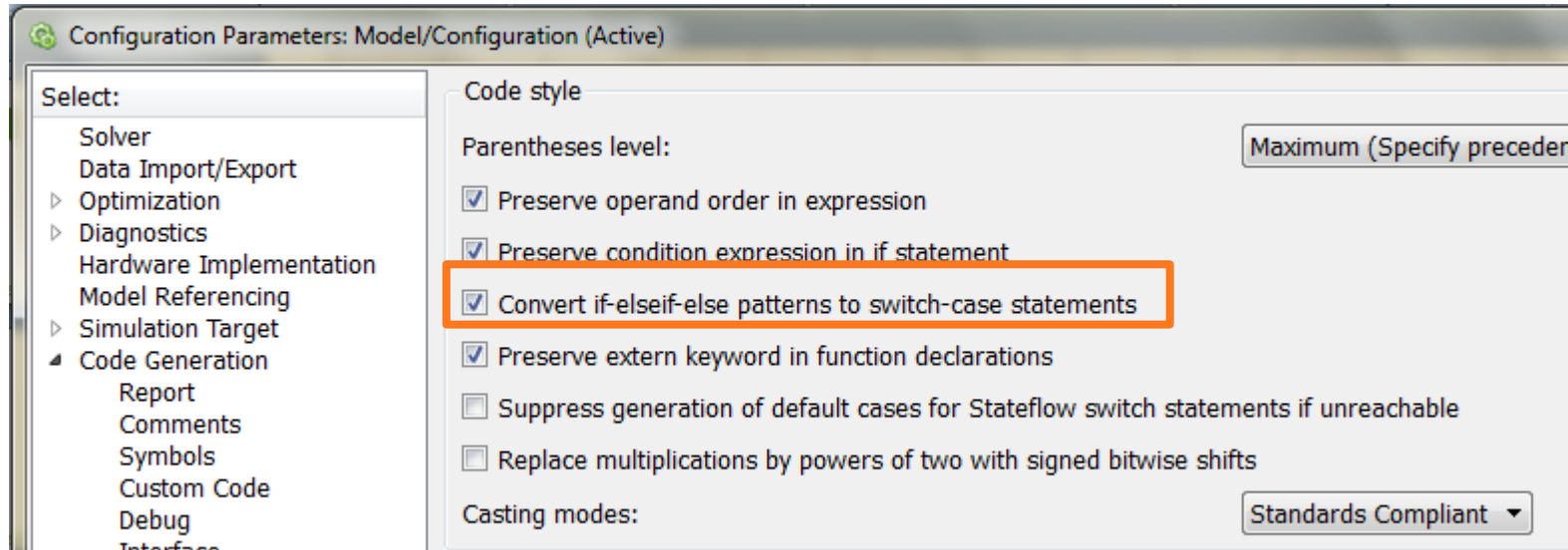
Non-Functional Requirements

- Execution time: Switch Case Vs If Else



Non-Functional Requirements

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



```
976 /* During 'NoPnPrNoCom': '<S304>:1' */
977 switch (localDW->u8_is_NoPnPrNoCom)
978 {
979     case CanSM_Model_IN_NPnCtrSlp:
980         NPnCtrSlp(u8NetId, localDW);
981         break;
982
983     case CanSM_Model_IN_NPnCtrStp:
984         NPnCtrStp(u8NetId, localDW);
985         break;
986
987     case CanSM_Model_IN_NPnTrcNor:
988         NPnTrcNor(u8NetId, localDW);
989         break;
990
991     case CanSM_Model_IN_NPnTrcStd:
992         NPnTrcStd(u8NetId, localDW);
993         break;
994
995     default:
996         /* During 'z': '<S304>:9' */
997         break;
998 }
```

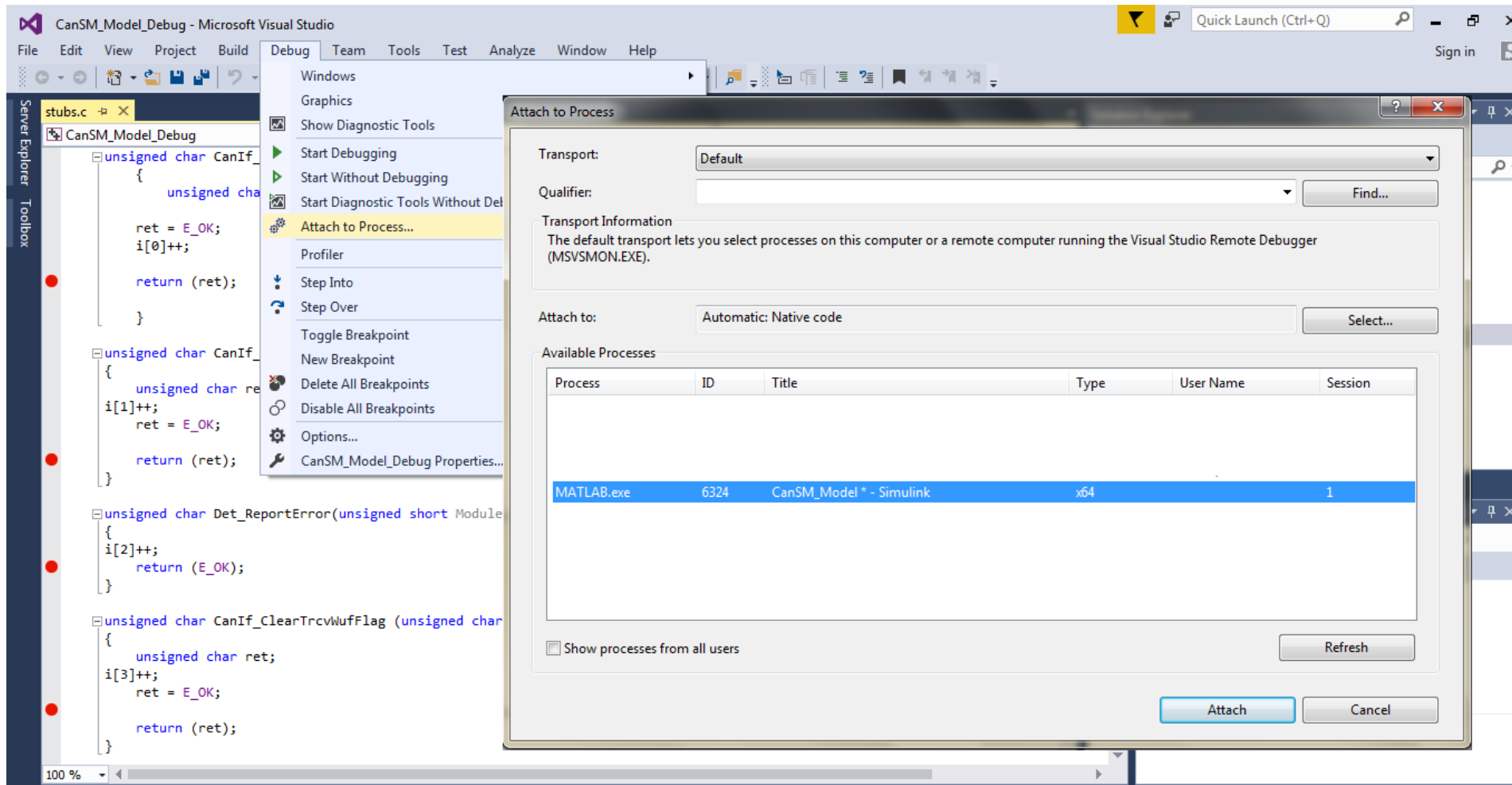
Smoke Testing

- 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.

The screenshot displays the MATLAB R2015a Stateflow editor interface. The main workspace shows a state transition diagram for the 'PncEnabled' state. A transition is triggered by a guard condition: `[/! Check if Trans ceiver is configured or partial networking is disabled */ [CANSM_BSM_G_PN_SUPPORTED == false]]`. This transition leads to two sub-states: `Link CANSM_BSM_DeinitPnNotSupported` and `CANSM_BSM_DeinitPnSupported`. The bottom console shows the current value of the variable `CANSM_BSM_G_PN_SUPPORTED = 0` and the command `debug>>`.

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.



Automotive technology, naturally

