Planning model architecture and modeling patterns for ISO 26262 compliance

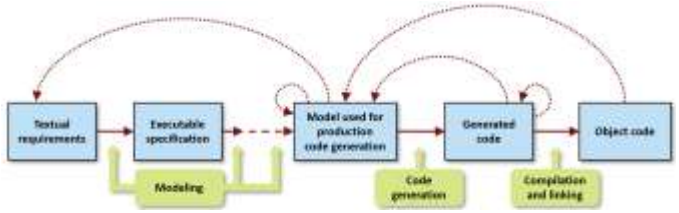
Dr. Tjorben Groß – MathWorks

ISO 26262 “Road Vehicles - Functional Safety”

- ISO 26262 is a functional safety standard for road vehicles
- Facilitates modern software engineering concepts such as
  - Model-Based Design
  - Early verification and validation
  - Code generation
- MathWorks IEC Certification Kit Compliant with ISO 26262 2nd Edition

ISO 26262:2018 Structure

ISO 26262-1	• Vocabulary
ISO 26262-2	• Management of functional safety
ISO 26262-3	• Concept phase
ISO 26262-4	• Product development: system level
ISO 26262-5	• Product development: hardware level
ISO 26262-6	• Product development: software level
ISO 26262-7	• Production, operation, service and decommissioning
ISO 26262-8	• Supporting processes
ISO 26262-9	• ASIL-oriented and safety-oriented analyses
ISO 26262-10	• Guidelines on ISO 26262
ISO 26262-11	• Guidelines on application of ISO 26262 to semiconductors
ISO 26262-12	• Adaptation of ISO 26262 for motorcycles



- Model-Based Design
- Early verification and validation
- Code generation

- Tool classification and qualification

Common questions regarding ISO 26262

- Is Simulink suitable for use for ISO 26262?
- Do I have an ISO 26262 compliant workflow?
- How to efficiently reach unit testing coverage criteria?
- How to achieve freedom from interference?
- Can we use AUTOSAR and meet ISO 26262 at the same time?

KOSTAL Asia R&D Center Receives ISO 26262 **ASIL D** Certification for Automotive Software Developed with Model-Based Design

Kostal's electronic steering column lock module.

Challenge

Develop automotive electronic steering column lock software and certify it to the highest-level functional safety standard

Solution

Use Model-Based Design to design, implement, and verify the application software via back-to-back PIL testing required for ISO 26262 ASIL D certification

Results

- **Development and certification time cut by 30%**
- 80% of errors identified in modeling phase
- PIL test framework for ISO 26262 established

"Using Model-Based Design to design, implement, and verify our software for the highest functional safety standard enabled our team to save costs, increase efficiency, and ensure software quality. Without Model-Based Design, more engineers would be needed to complete the project in the same time frame."
– Cheng Hui, KOSTAL

Link to user story

ISO 26262-6:2018 Update for MathWorks tools

Table 5 — Notations for software unit design

	Notations	ASIL A	ASIL B	ASIL C	ASIL D
1a	Natural language[a]	++	++	++	++
1b	Informal notations	++	++	+	+
1c	Semi-formal notations[b]	+	+	++	++
1d	Formal notations	+	+	+	+

[a] Natural language can complement the use of notations for example where some topics are more readily expressed in natural language or provide an explanation and rationale for decisions captured in the notations.

EXAMPLE To avoid possible ambiguity of natural language when designing complex elements, a combination of an activity diagram with natural language can be used.

Simulink® and Stateflow® are examples of suitable products

information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

NOTE In the case of model-based development with automatic code generation, the methods for representing the softw

model which serves as the basis for the code generation

Table 2 Software Architecture Design Notations has similar suitability wording for use of Simulink and Stateflow

ISO 26262-6:2018 Methods and Model-Based Design

Table 9 — Structural coverage metrics at the software unit level

	Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a	Statement coverage	++	++	+	+
1b	Branch coverage	+	++	++	++
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++

	T-1	T-2	T-3	T-4	T-5	T-6	T-7	T-8	T-9	T-10	T-11	T-12	T-13	T-14	T-15
a	++	++	++	o	++	++	o	++	+	++	++	++	++	++	++
b	++	+	++	++	+	++	+	++	++	++	++	++	++	++	++
c	++	++	++	++	++	++	++	++	++	++	++		++		++
d	++	+	++	++	+	++	++	+		++	+				++
e	++		++	+		++	+			++					++
f	++		++	++		++	++			++					++
g	++		++	++		++	++			++					
h	++		++	++		++	++			+					
i	+		++			++	+								
j						++	++								
k							++								
l							++								
m							++								
n							++								

	Supported by MBD
++	Highly Recommended
+	Recommended
o	No Recommendation

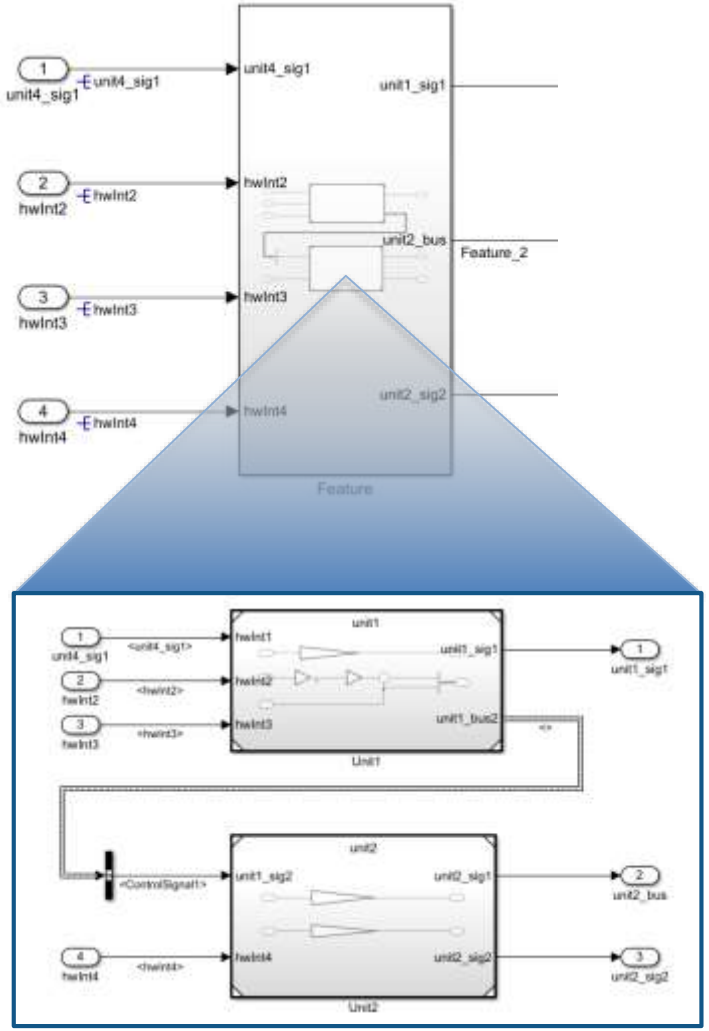
Common questions regarding ISO 26262

- Is Simulink suitable for use for ISO 26262?
- Do I have an ISO 26262 compliant workflow?
- How to efficiently reach unit testing coverage criteria?
- How to achieve freedom from interference?
- Can we use AUTOSAR and meet ISO 26262 at the same time?

Use Model Reference for Unit Level Model

- Challenges:
  - Modularize algorithms for reuse
  - Preform unit level testing
  - Configuration management
  - Achieve freedom from interference

- Best Practice
  - Use model reference for unit level model
  - Group units to form functional hierarchy with virtual subsystems

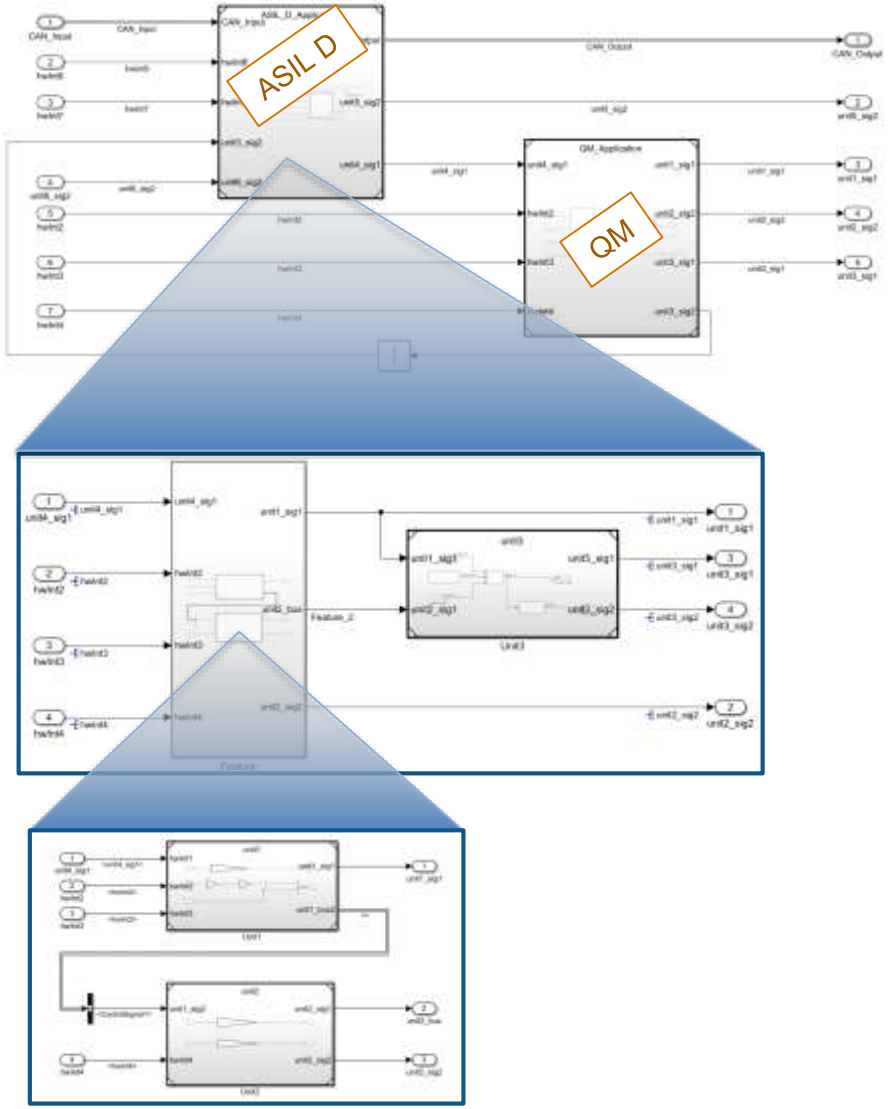


Use Model References when splitting ASIL and QM

- Challenges:
  - Achieving freedom from interference
  - Integration of generated code

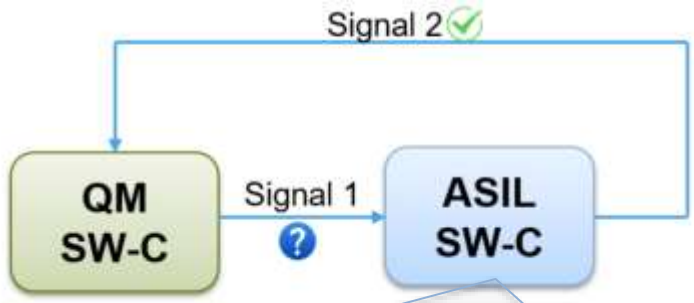
- Best Practice:
  - Generate code separately for each ASIL/QM component

Model Hierarchy	Modeling Pattern
Top level (ASIL / QM)	Model Reference
Features	Subsystem
Unit	Model Reference

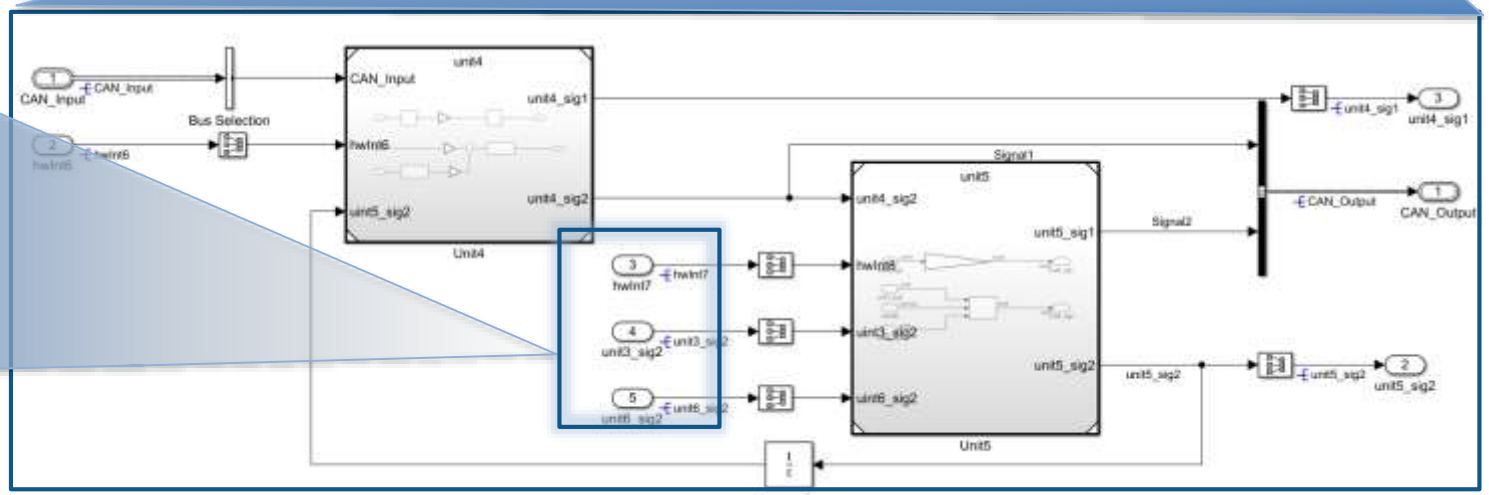


Establish Data Protection Between ASIL and QM Components

- Challenge:
  - How to provide signal protection between ASIL and QM components?
- Best Practice
  - Use Get/Set storage class for signals between ASIL and QM components

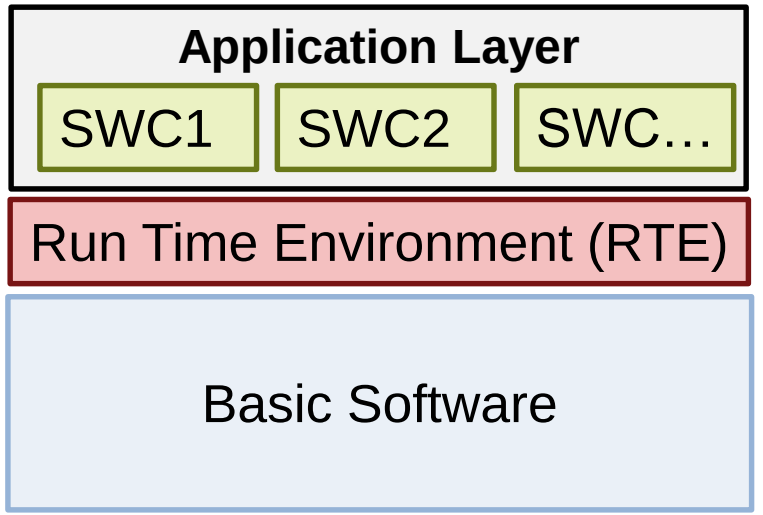



```
106  /* SignalConversion: '<Root>/Signal Conversion2' incorporates:
107   *  Inport: '<Root>/hwInt7'
108   */
109  rtb_SignalConversion2 = get_hwInt7();
110
111  /* SignalConversion: '<Root>/Signal Convers
112   *  Inport: '<Root>/unit3_sig2'
113   */
114  rtb_UnitDelay = get_unit3_sig2();
115
116  /* SignalConversion: '<Root>/Signal Convers
117   *  Inport: '<Root>/unit6_sig2'
118   */
119  rtb_SignalConversion4 = get_unit6_sig2();
120
121  /* ModelReference: '<Root>/Unit5' incorporates:
122   *  UnitDelay: '<Root>/Unit Delay'
123   */
```



AUTOSAR Implications

- Best practices compatible with AUTOSAR

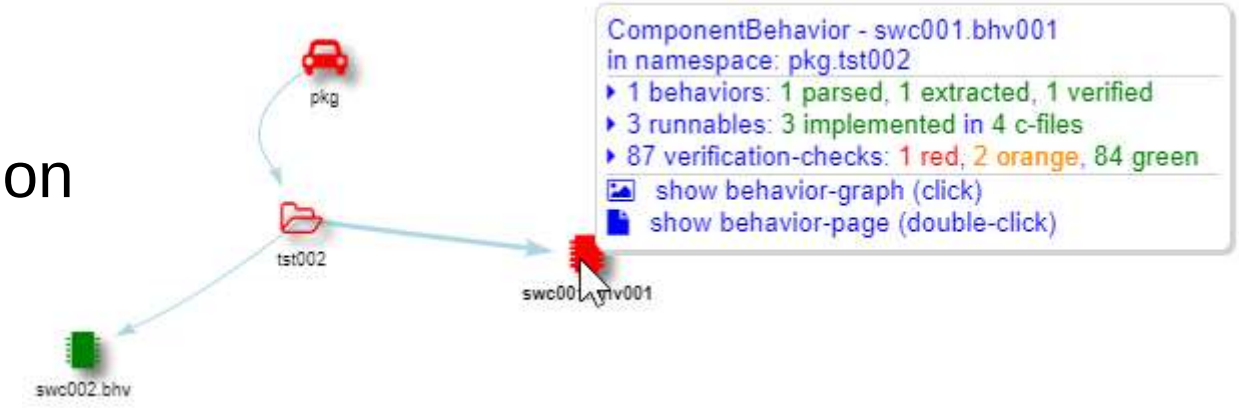


- Get/Set function can be implemented using Send/Receiver port

- Challenge:

  Ensure match between specification and implementation

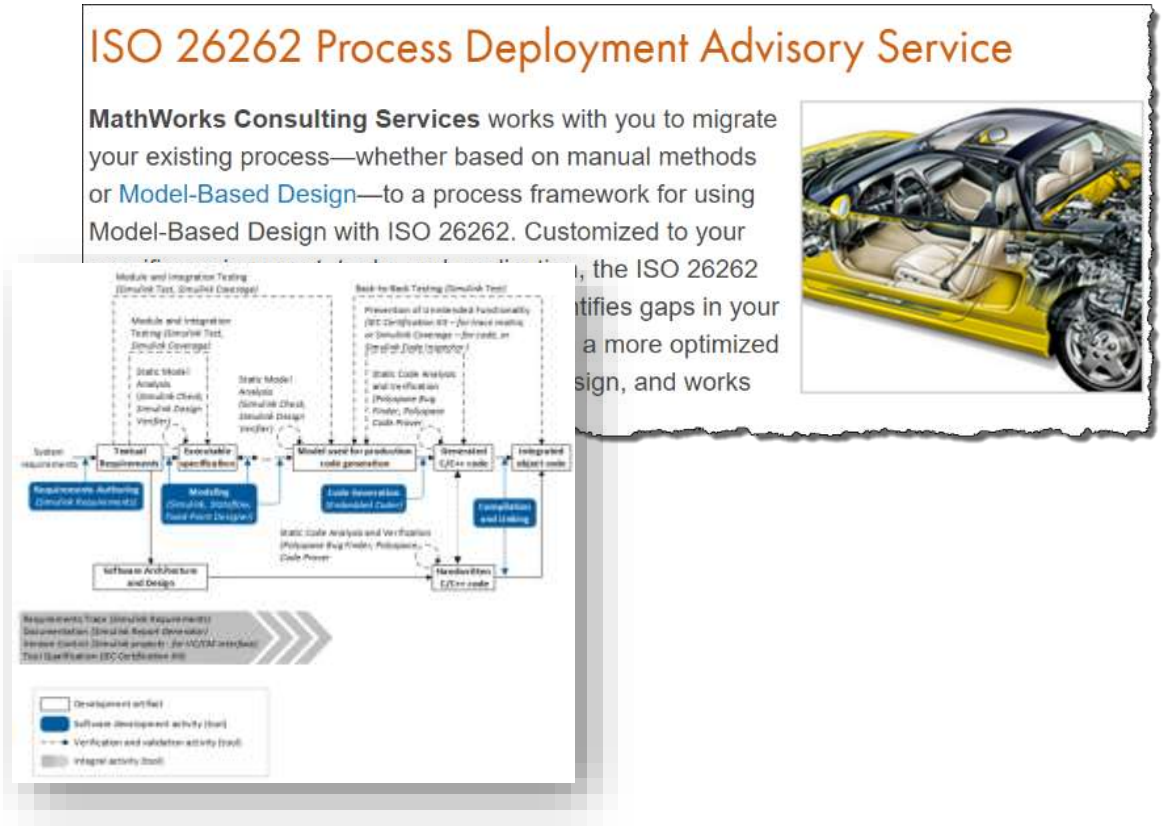
  → Polyspace AUTOSAR



Summary

- MathWorks support for ISO 26262
  - IEC Certification Kit
  - Consulting

- Best practice
  - Freedom from interference
  - Unit testing
  - AUTOSAR
  - More information:
    - tgross@mathworks.com
    - uwahner@mathworks.com
    - MathWorks ISO 26262 landing page



- See us at the demo booth

  Discuss how we can support your ISO 26262 project