MATLAB EXPO 2019

规划符合 ISO 26262 的模型架构及建模模式

李智慧

ISO 26262 “Road Vehicles - Functional Safety”

- ISO 26262 是道路车辆功能安全标准
- 特别强调了现代软件工程概念，比如：
  - 基于模型的设计（MBD）
  - 早期验证与确认
  - 自动代码生成
- 迈斯沃克在符合 ISO 26262 流程方面一直在努力
  - 系统复杂度的逐渐增加
  - 来自于高级驾驶辅助系统（ADAS）和自动驾驶（AD）方面的需求

来自 ISO 26262 的挑战

- 有一个符合 ISO 26262 的开发**流程**吗？
- **Simulink** 能用于符合 ISO 26262 的产品开发吗？
- 如何**高效地**达到单元测试覆盖度要求？
- 怎样做到模块间的**互不干涉**？
- 能做到既符合 **AUTOSAR** 又满足 ISO 26262 吗？
- ……

迈斯沃克通过以下两种方式提供更加广泛的支持：
IEC 认证包和技术**咨询服务**

ISO 26262-6:2018 指出Simulink和Stateflow适合软件架构设计、产品开发以及代码生成

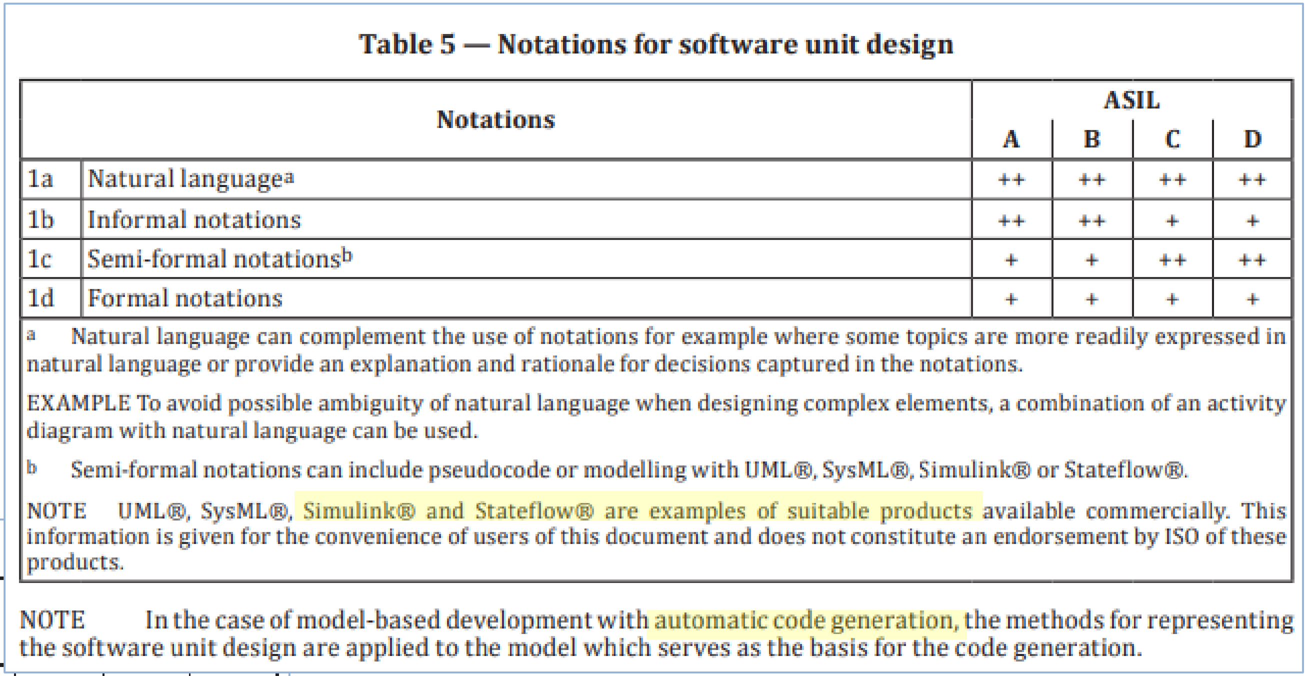


Table 5 — Notations for software unit design

	Notations	ASIL			
		A	B	C	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.

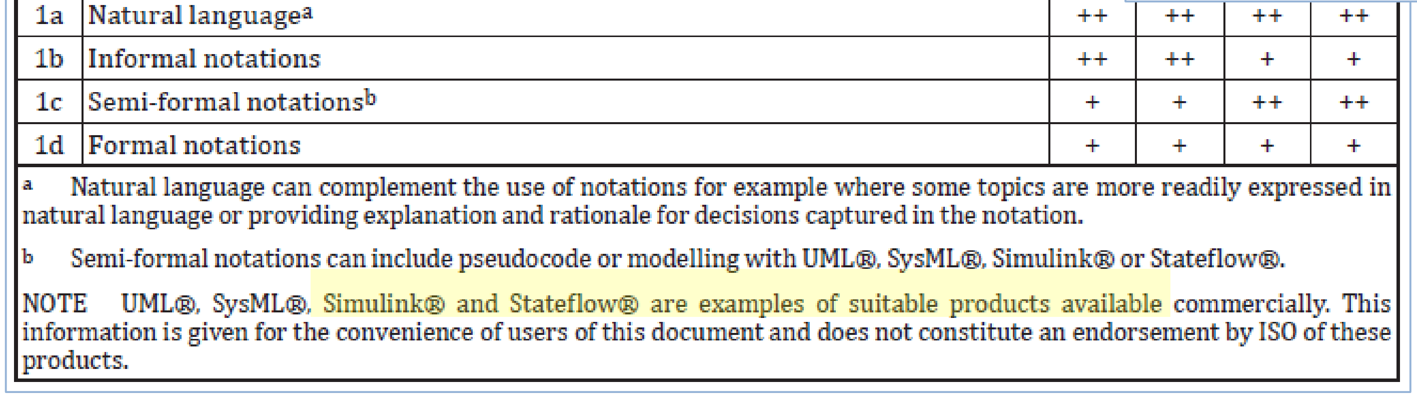
EXAMPLE To avoid possible ambiguity of natural language when designing complex elements, a combination of an activity diagram with natural language can be used.

[b] Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This 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 software unit design are applied to the model which serves as the basis for the code generation.

Table 2 — Notations for software architectural design



	Notations	A			
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 providing explanation and rationale for decisions captured in the notation.

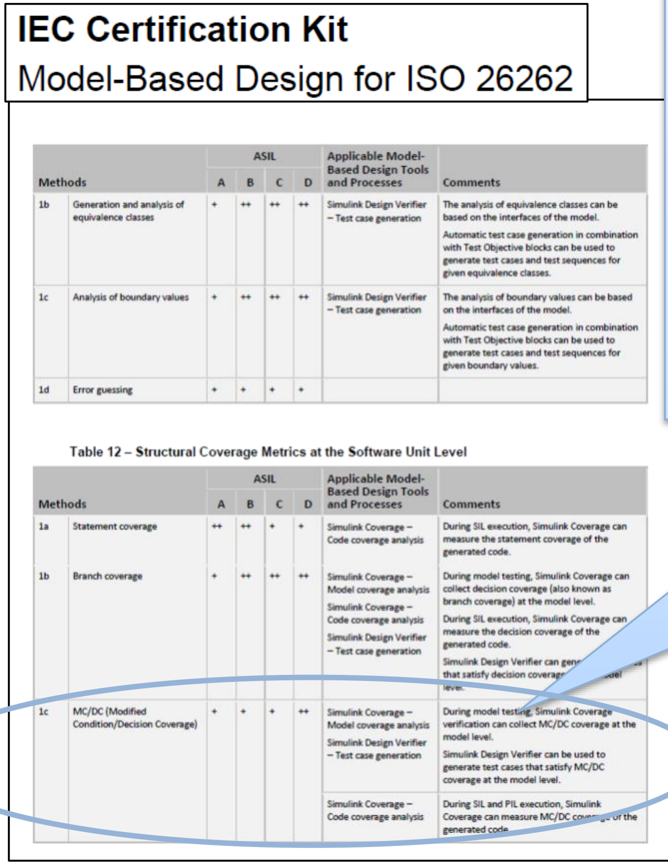
[b] Semi-formal notations can include pseudocode or modelling with UML®, SysML®, Simulink® or Stateflow®.

NOTE UML®, SysML®, Simulink® and Stateflow® are examples of suitable products available commercially. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO of these products.

迈斯沃克对 ISO 26262 的支持

IEC 认证包

- 迈斯沃克工具链与 ISO 26262 要求之间的对应



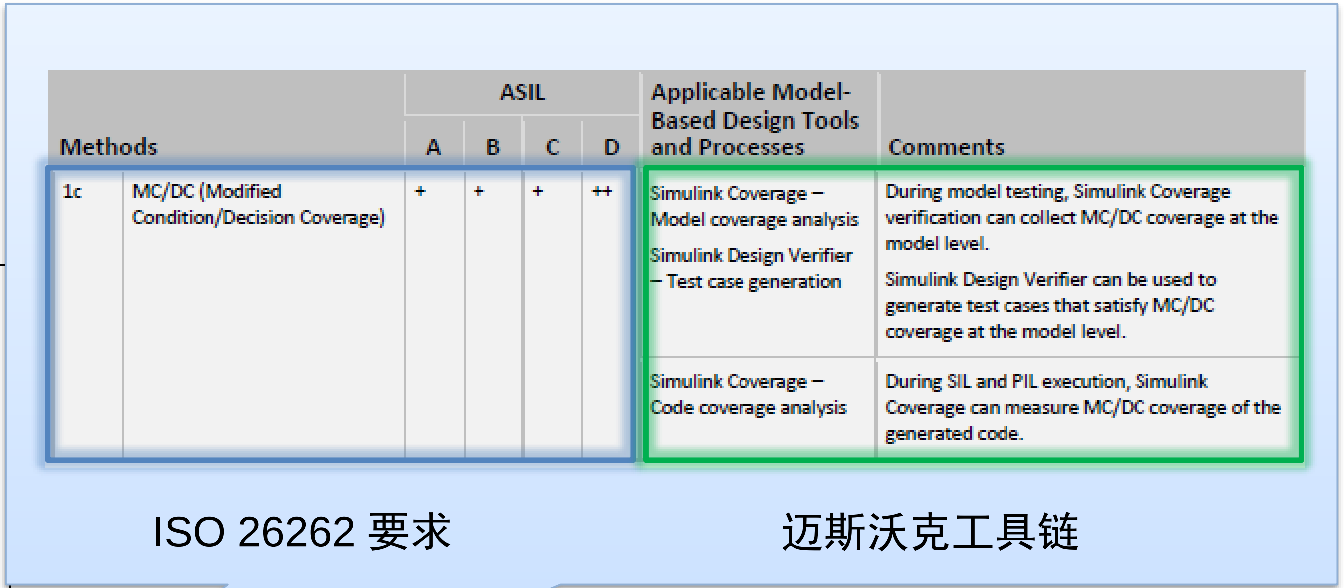
IEC Certification Kit

Model-Based Design for ISO 26262

Methods		ASIL				Applicable Model-Based Design Tools and Processes	Comments
		A	B	C	D		
1b	Generation and analysis of equivalence classes	+	++	++	++	Simulink Design Verifier – Test case generation	The analysis of equivalence classes can be based on the interfaces of the model. Automatic test case generation in combination with Test Objective blocks can be used to generate test cases and test sequences for given equivalence classes.
1c	Analysis of boundary values	+	++	++	++	Simulink Design Verifier – Test case generation	The analysis of boundary values can be based on the interfaces of the model. Automatic test case generation in combination with Test Objective blocks can be used to generate test cases and test sequences for given boundary values.
1d	Error guessing	+	+	+	+		

Table 12 – Structural Coverage Metrics at the Software Unit Level

Methods		ASIL				Applicable Model-Based Design Tools and Processes	Comments
		A	B	C	D		
1a	Statement coverage	++	++	+	+	Simulink Coverage – Code coverage analysis	During SIL execution, Simulink Coverage can measure the statement coverage of the generated code.
1b	Branch coverage	+	++	++	++	Simulink Coverage – Model coverage analysis Simulink Coverage – Code coverage analysis Simulink Design Verifier – Test case generation	During model testing, Simulink Coverage can collect decision coverage (also known as branch coverage) at the model level. During SIL execution, Simulink Coverage can measure the decision coverage of the generated code. Simulink Design Verifier can gen[illegible] that satisfy decision coverag[illegible]del level.
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++	Simulink Coverage – Model coverage analysis Simulink Design Verifier – Test case generation	During model testing, Simulink Coverage verification can collect MC/DC coverage at the model level. Simulink Design Verifier can be used to generate test cases that satisfy MC/DC coverage at the model level.
						Simulink Coverage – Code coverage analysis	During SIL and PIL execution, Simulink Coverage can measure MC/DC coverage of the generated code.



Methods		ASIL				Applicable Model-Based Design Tools and Processes	Comments
		A	B	C	D		
1c	MC/DC (Modified Condition/Decision Coverage)	+	+	+	++	Simulink Coverage – Model coverage analysis Simulink Design Verifier – Test case generation	During model testing, Simulink Coverage verification can collect MC/DC coverage at the model level. Simulink Design Verifier can be used to generate test cases that satisfy MC/DC coverage at the model level.
						Simulink Coverage – Code coverage analysis	During SIL and PIL execution, Simulink Coverage can measure MC/DC coverage of the generated code.

ISO 26262 要求

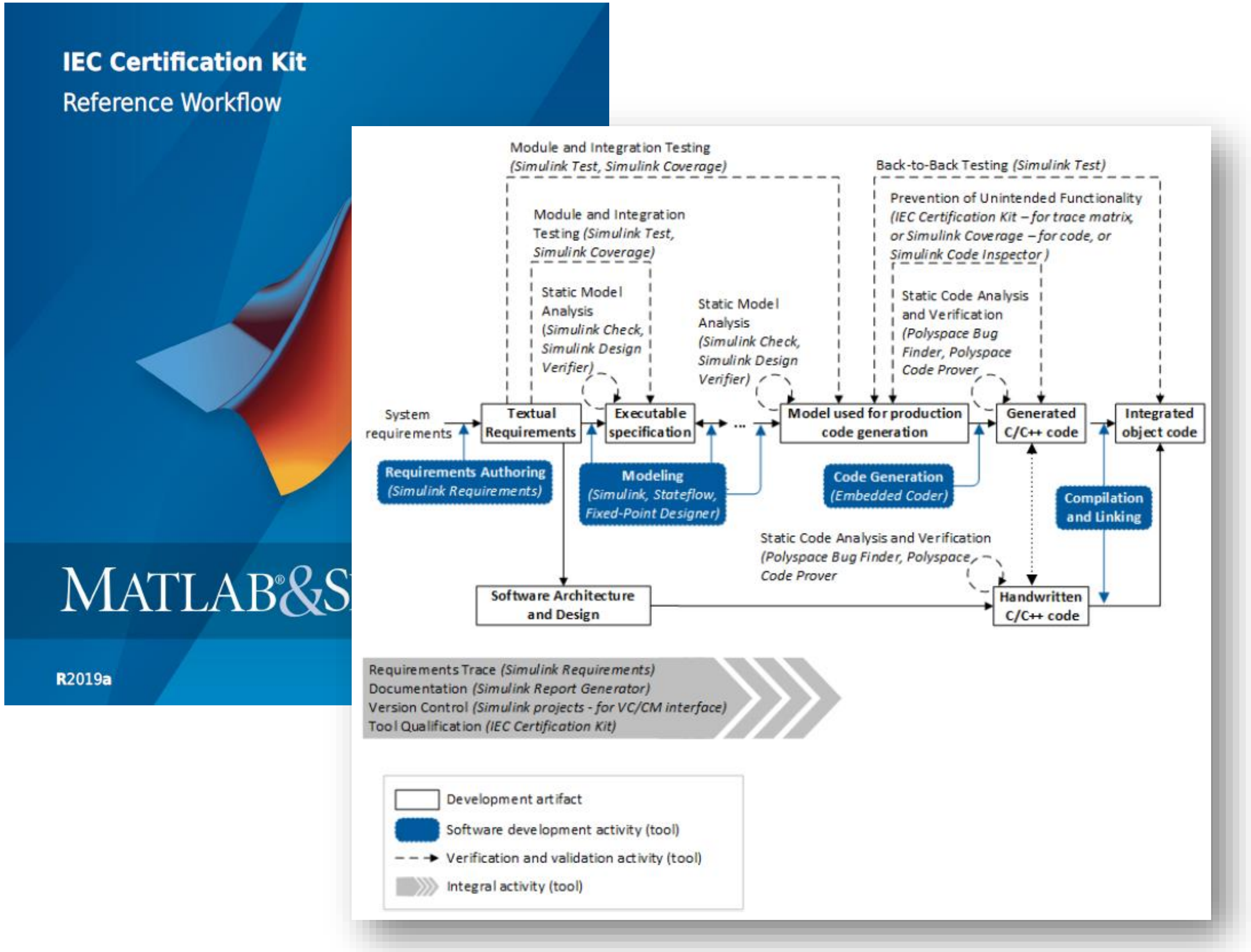
迈斯沃克工具链

迈斯沃克对 ISO 26262 的支持

IEC 认证包

- 迈斯沃克工具链与 ISO 26262 要求之间的对应

- 基于模型设计参考流程



IEC Certification Kit

Reference Workflow

MATLAB&S

R2019a

Module and Integration Testing (Simulink Test, Simulink Coverage)

Back-to-Back Testing (Simulink Test)

Module and Integration Testing (Simulink Test, Simulink Coverage)

Prevention of Unintended Functionality (IEC Certification Kit – for trace matrix, or Simulink Coverage – for code, or Simulink Code Inspector)

Static Model Analysis (Simulink Check, Simulink Design Verifier)

Static Model Analysis (Simulink Check, Simulink Design Verifier)

Static Code Analysis and Verification (Polyspace Bug Finder, Polyspace Code Prover

System requirements

Textual Requirements

Executable specification

...

Model used for production code generation

Generated C/C++ code

Integrated object code

Requirements Authoring (Simulink Requirements)

Modeling (Simulink, Stateflow, Fixed-Point Designer)

Code Generation (Embedded Coder)

Compilation and Linking

Static Code Analysis and Verification (Polyspace Bug Finder, Polyspace Code Prover

Software Architecture and Design

Handwritten C/C++ code

Requirements Trace (Simulink Requirements)

Documentation (Simulink Report Generator)

Version Control (Simulink projects - for VC/CM interface)

Tool Qualification (IEC Certification Kit)

□ Development artifact

■ Software development activity (tool)

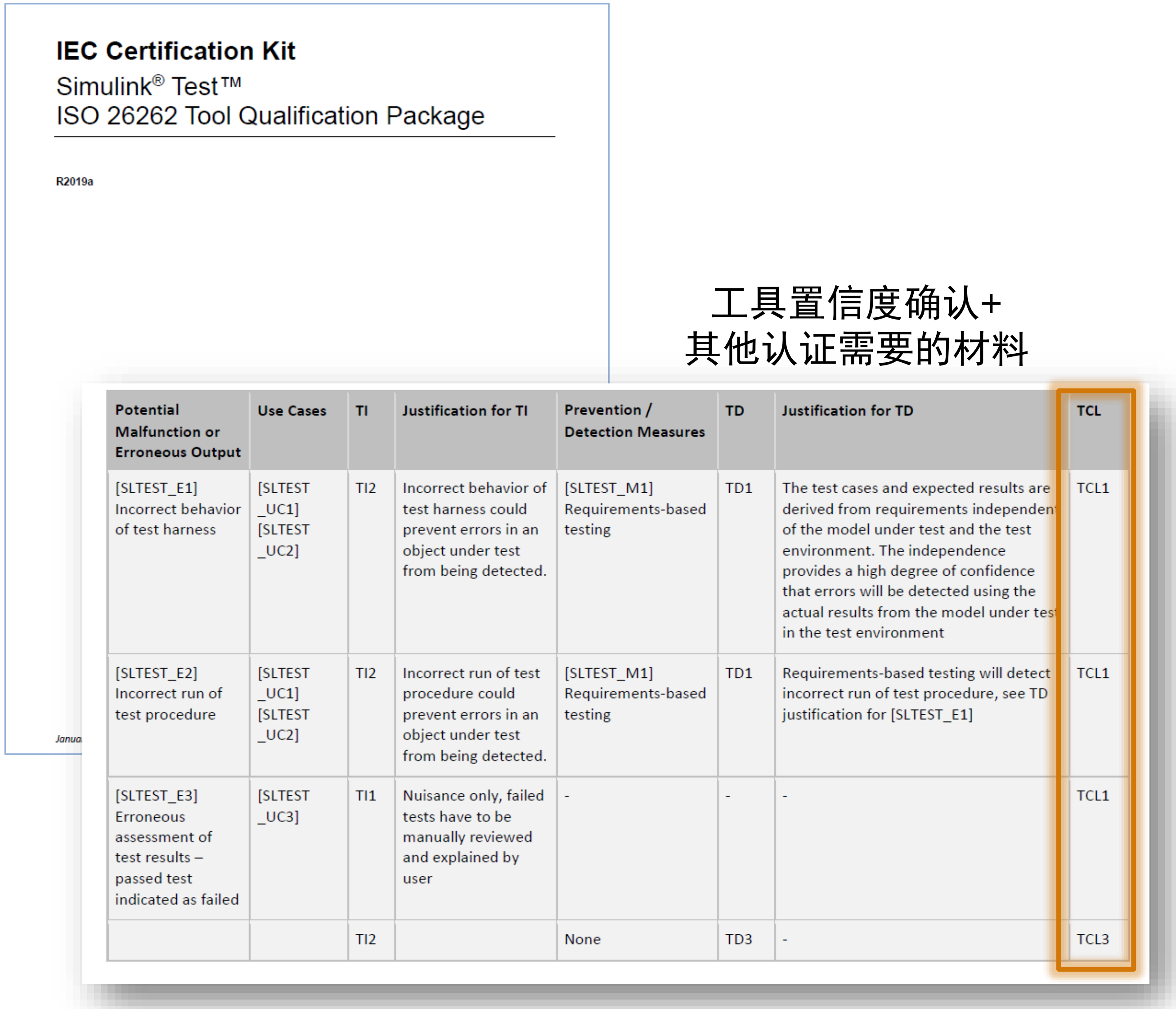
- - → Verification and validation activity (tool)

⋙ Integral activity (tool)

迈斯沃克对 ISO 26262 的支持

IEC 认证包

- 迈斯沃克工具链与 ISO 26262 要求之间的对应
- 基于模型设计参考流程
- 工具认证包
  - 软件工具评估准则报告
  - 软件工具鉴定报告（模板）
  - 参考认证报告



IEC Certification Kit
Simulink® Test™
ISO 26262 Tool Qualification Package

R2019a

Janua

Potential Malfunction or Erroneous Output	Use Cases	TI	Justification for TI	Prevention / Detection Measures	TD	Justification for TD	TCL
[SLTEST_E1] Incorrect behavior of test harness	[SLTEST_UC1] [SLTEST_UC2]	TI2	Incorrect behavior of test harness could prevent errors in an object under test from being detected.	[SLTEST_M1] Requirements-based testing	TD1	The test cases and expected results are derived from requirements independent of the model under test and the test environment. The independence provides a high degree of confidence that errors will be detected using the actual results from the model under test in the test environment	TCL1
[SLTEST_E2] Incorrect run of test procedure	[SLTEST_UC1] [SLTEST_UC2]	TI2	Incorrect run of test procedure could prevent errors in an object under test from being detected.	[SLTEST_M1] Requirements-based testing	TD1	Requirements-based testing will detect incorrect run of test procedure, see TD justification for [SLTEST_E1]	TCL1
[SLTEST_E3] Erroneous assessment of test results – passed test indicated as failed	[SLTEST_UC3]	TI1	Nuisance only, failed tests have to be manually reviewed and explained by user	-	-	-	TCL1
		TI2		None	TD3	-	TCL3

工具置信度确认+
其他认证需要的材料

迈斯沃克对 ISO 26262 的支持

IEC 认证包 – 用户案例

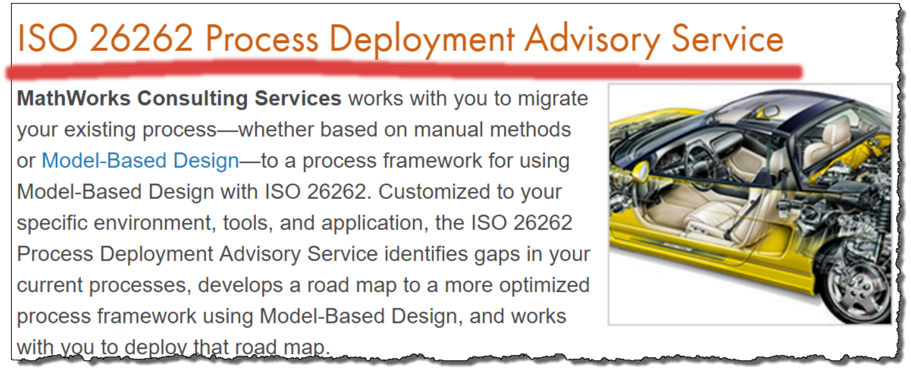
- 迈斯沃克工具链与 ISO 26262 要求之间的对应
- 基于模型设计参考流程
- 工具认证包
  - 软件工具评估准则报告
  - 软件工具鉴定报告（模板）
  - 参考认证报告

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

> “Without Model-Based Design, we would have needed at least 30% more time to develop and certify the ESCL application software. We saved time and effort by generating efficient code that satisfied all our speed and memory requirements.”
>
> — *Cheng Hui, platform and process manager, KOSTAL*

迈斯沃克对 ISO 26262 的支持

技术咨询服务



ISO 26262 Process Deployment Advisory Service

MathWorks Consulting Services works with you to migrate your existing process—whether based on manual methods or Model-Based Design—to a process framework for using Model-Based Design with ISO 26262. Customized to your specific environment, tools, and application, the ISO 26262 Process Deployment Advisory Service identifies gaps in your current processes, develops a road map to a more optimized process framework using Model-Based Design, and works with you to deploy that road map.

- 可定制的服务
  - ISO 26262 软件开发流程差距分析

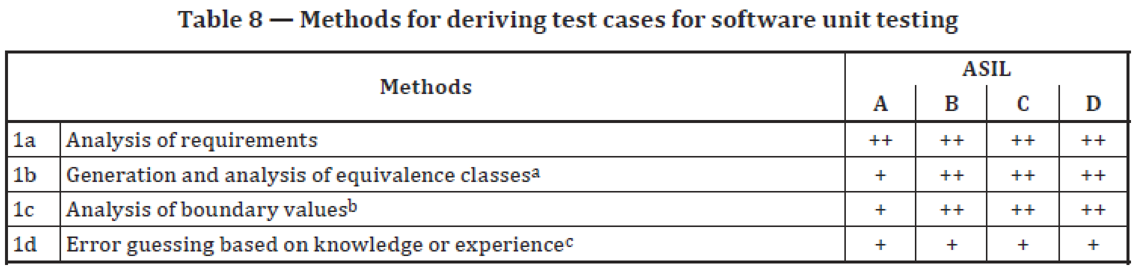


Table 8 — Methods for deriving test cases for software unit testing

	Methods	ASIL			
		A	B	C	D
1a	Analysis of requirements	++	++	++	++
1b	Generation and analysis of equivalence classes[a]	+	++	++	++
1c	Analysis of boundary values[b]	+	++	++	++
1d	Error guessing based on knowledge or experience[c]	+	+	+	+

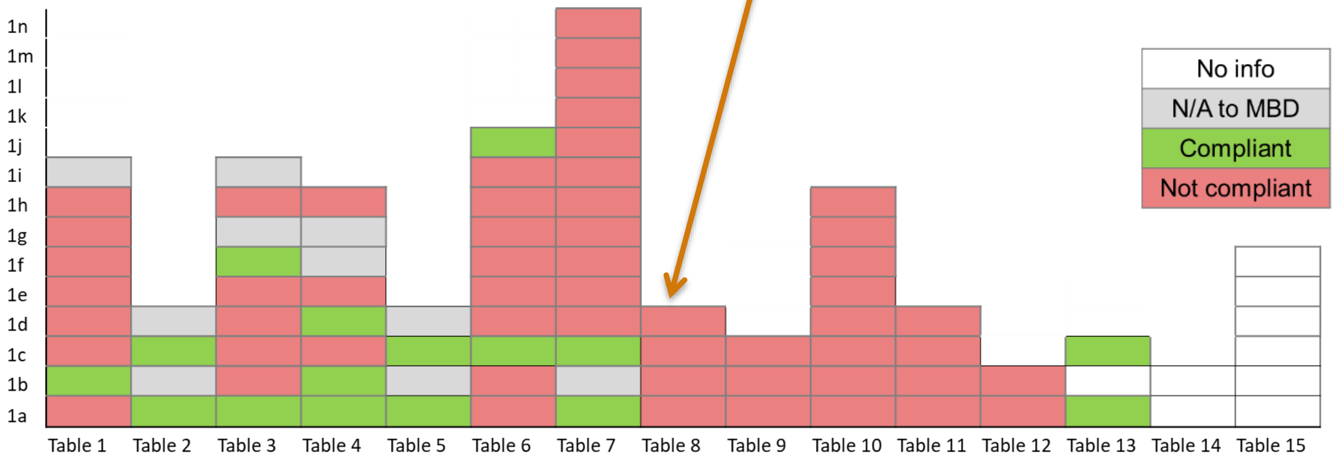


	Table 1	Table 2	Table 3	Table 4	Table 5	Table 6	Table 7	Table 8	Table 9	Table 10	Table 11	Table 12	Table 13	Table 14	Table 15
1n							🟥								
1m							🟥								
1l							🟥								
1k							🟥								
1j						🟩	🟥								
1i	▨		▨			🟥	🟥								
1h	🟥		🟥	🟥		🟥	🟥			🟥					
1g	🟥		▨	▨		🟥	🟥			🟥					
1f	🟥		🟩	▨		🟥	🟥			🟥					⬜
1e	🟥		🟥	🟥		🟥	🟥			🟥					⬜
1d	🟥	▨	🟥	🟩	▨	🟥	🟥	🟥		🟥	🟥				⬜
1c	🟥	🟩	🟥	🟥	🟩	🟩	🟩	🟥	🟥	🟥	🟥		🟩		⬜
1b	🟩	▨	🟥	🟩	▨	🟥	▨	🟥	🟥	🟥	🟥	🟥	⬜	⬜	⬜
1a	🟥	🟩	🟩	🟩	🟩	🟥	🟩	🟥	🟥	🟥	🟥	🟥	🟩	⬜	⬜

⬜ No info

▨ N/A to MBD

🟩 Compliant

🟥 Not compliant

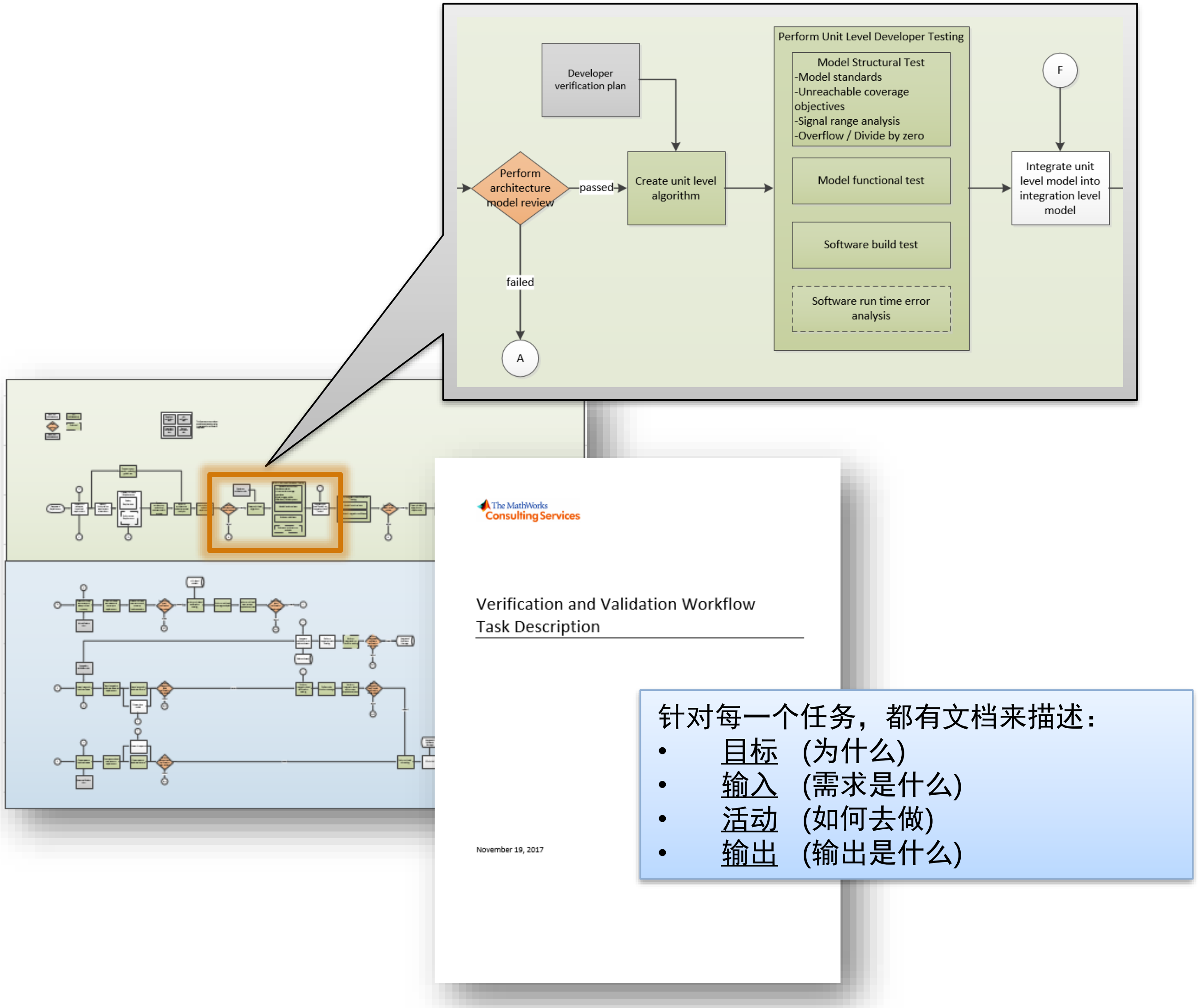
迈斯沃克对 ISO 26262 的支持

技术咨询服务

ISO 26262 Process Deployment Advisory Service

MathWorks Consulting Services works with you to migrate your existing process—whether based on manual methods or Model-Based Design—to a process framework for using Model-Based Design with ISO 26262. Customized to your specific environment, tools, and application, the ISO 26262 Process Deployment Advisory Service identifies gaps in your current processes, develops a road map to a more optimized process framework using Model-Based Design, and works with you to deploy that road map.

- 可定制的服务
  - ISO 26262 软件开发流程差距分析
  - 验证与确认（V&V）流程建立


Developer verification plan
Perform Unit Level Developer Testing
Model Structural Test
-Model standards
-Unreachable coverage objectives
-Signal range analysis
-Overflow / Divide by zero
F
Perform architecture model review
passed
Create unit level algorithm
Model functional test
Integrate unit level model into integration level model
Software build test
failed
Software run time error analysis
A
The MathWorks
Consulting Services

Verification and Validation Workflow
Task Description

November 19, 2017

针对每一个任务，都有文档来描述：
- 目标 (为什么)
- 输入 (需求是什么)
- 活动 (如何去做)
- 输出 (输出是什么)

MathWorks

迈斯沃克对 ISO 26262 的支持

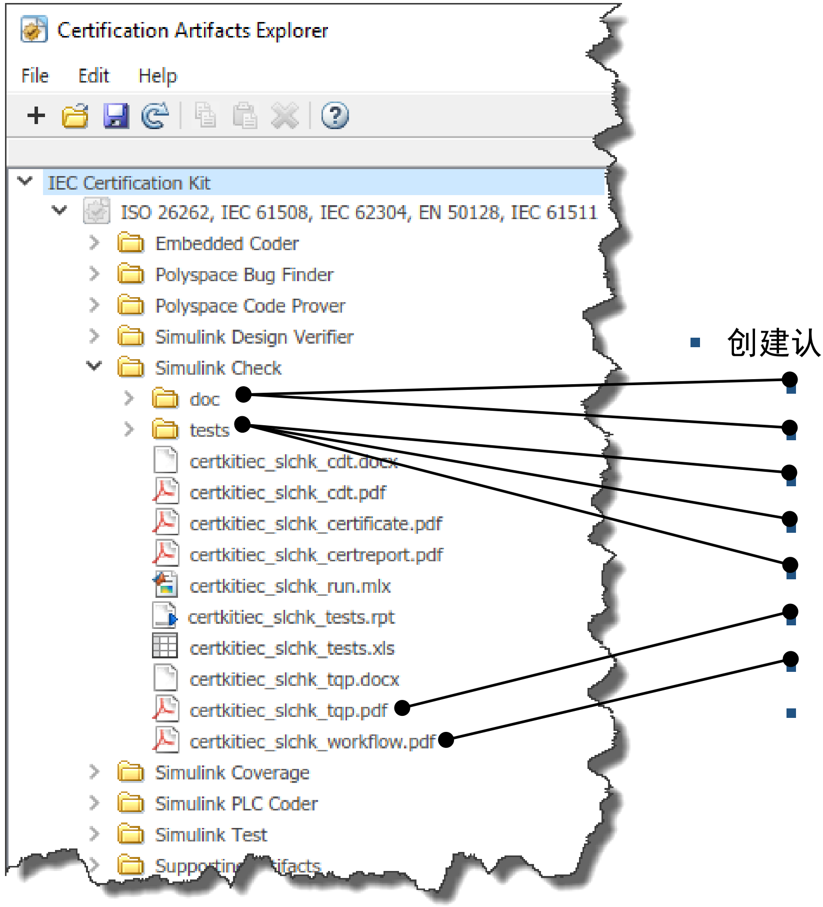
技术咨询服务

ISO 26262 Process Deployment Advisory Service

MathWorks Consulting Services works with you to migrate your existing process—whether based on manual methods or Model-Based Design—to a process framework for using Model-Based Design with ISO 26262. Customized to your specific environment, tools, and application, the ISO 26262 Process Deployment Advisory Service identifies gaps in your current processes, develops a road map to a more optimized process framework using Model-Based Design, and works with you to deploy that road map.

- 可定制的服务
  - ISO 26262 软件开发流程差距分析
  - 验证与确认（V&V）流程建立
  - 工具认证支持

工具预认证材料

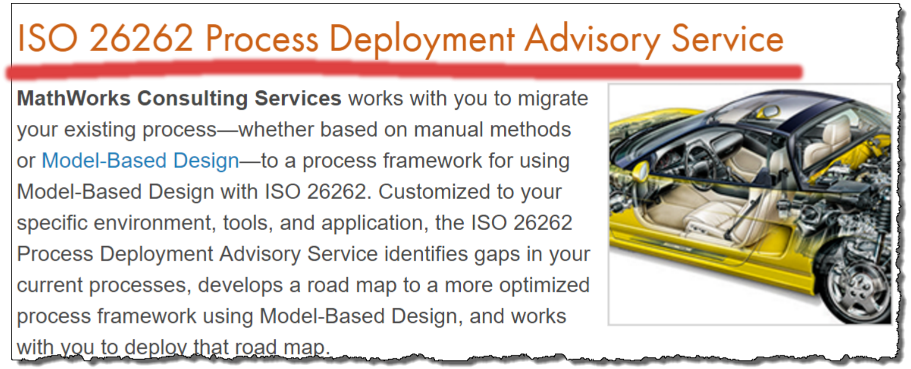


- IEC Certification Kit
  - ISO 26262, IEC 61508, IEC 62304, EN 50128, IEC 61511
    - Embedded Coder
    - Polyspace Bug Finder
    - Polyspace Code Prover
    - Simulink Design Verifier
    - Simulink Check
      - doc
      - tests
      - certkitiec_slchk_cdt.docx
      - certkitiec_slchk_cdt.pdf
      - certkitiec_slchk_certificate.pdf
      - certkitiec_slchk_certreport.pdf
      - certkitiec_slchk_run.mlx
      - certkitiec_slchk_tests.rpt
      - certkitiec_slchk_tests.xls
      - certkitiec_slchk_tqp.docx
      - certkitiec_slchk_tqp.pdf
      - certkitiec_slchk_workflow.pdf
    - Simulink Coverage
    - Simulink PLC Coder
    - Simulink Test
    - Supporting Artifacts

- 创建认证材料
  - 工具需求
  - 用户手册
  - 测试用例
  - 期望结果
  - 追踪矩阵
  - 置信度分类
  - 参考流程
  - ……

迈斯沃克对 ISO 26262 的支持

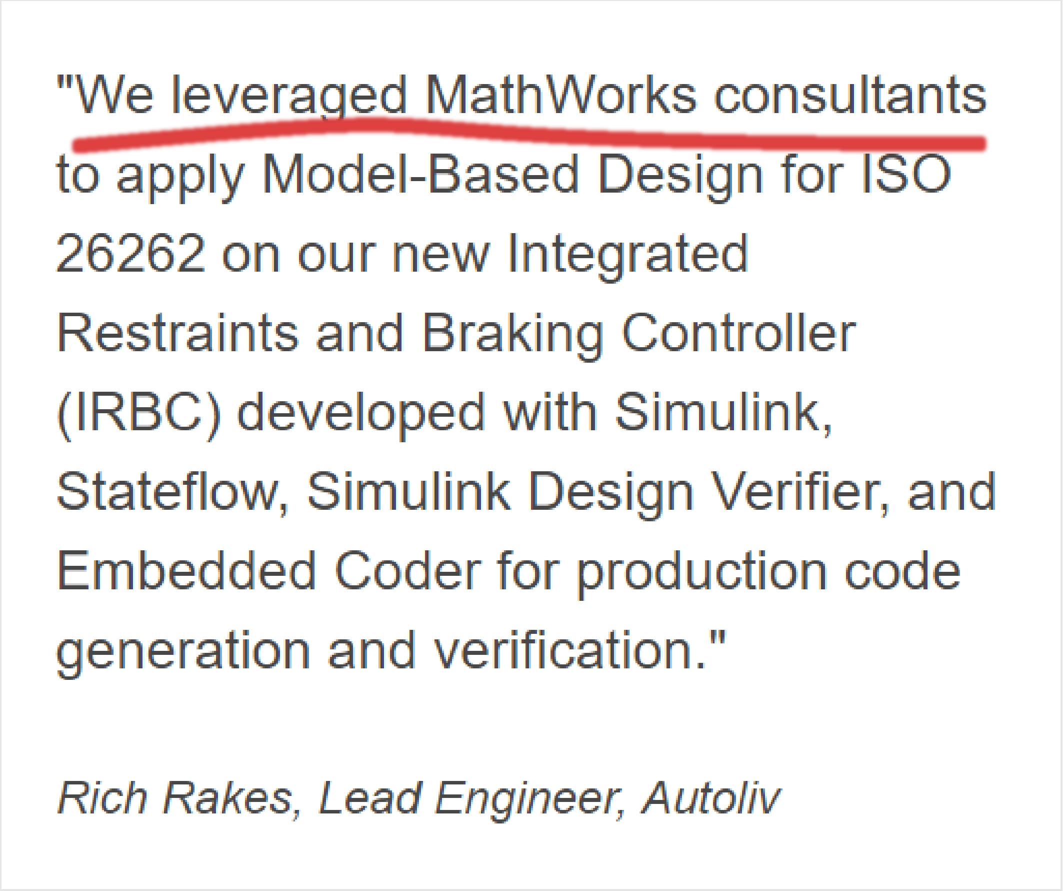
技术咨询服务



ISO 26262 Process Deployment Advisory Service

MathWorks Consulting Services works with you to migrate your existing process—whether based on manual methods or Model-Based Design—to a process framework for using Model-Based Design with ISO 26262. Customized to your specific environment, tools, and application, the ISO 26262 Process Deployment Advisory Service identifies gaps in your current processes, develops a road map to a more optimized process framework using Model-Based Design, and works with you to deploy that road map.

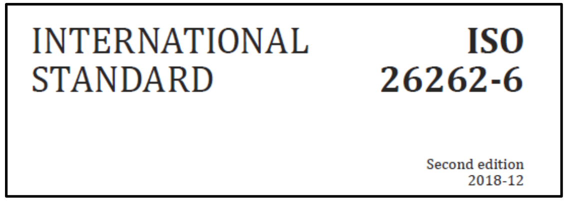
- 可定制的服务
  - ISO 26262 软件开发流程差距分析
  - 验证与确认（V&V）流程建立
  - 工具认证支持



"We leveraged MathWorks consultants to apply Model-Based Design for ISO 26262 on our new Integrated Restraints and Braking Controller (IRBC) developed with Simulink, Stateflow, Simulink Design Verifier, and Embedded Coder for production code generation and verification."

Rich Rakes, Lead Engineer, Autoliv

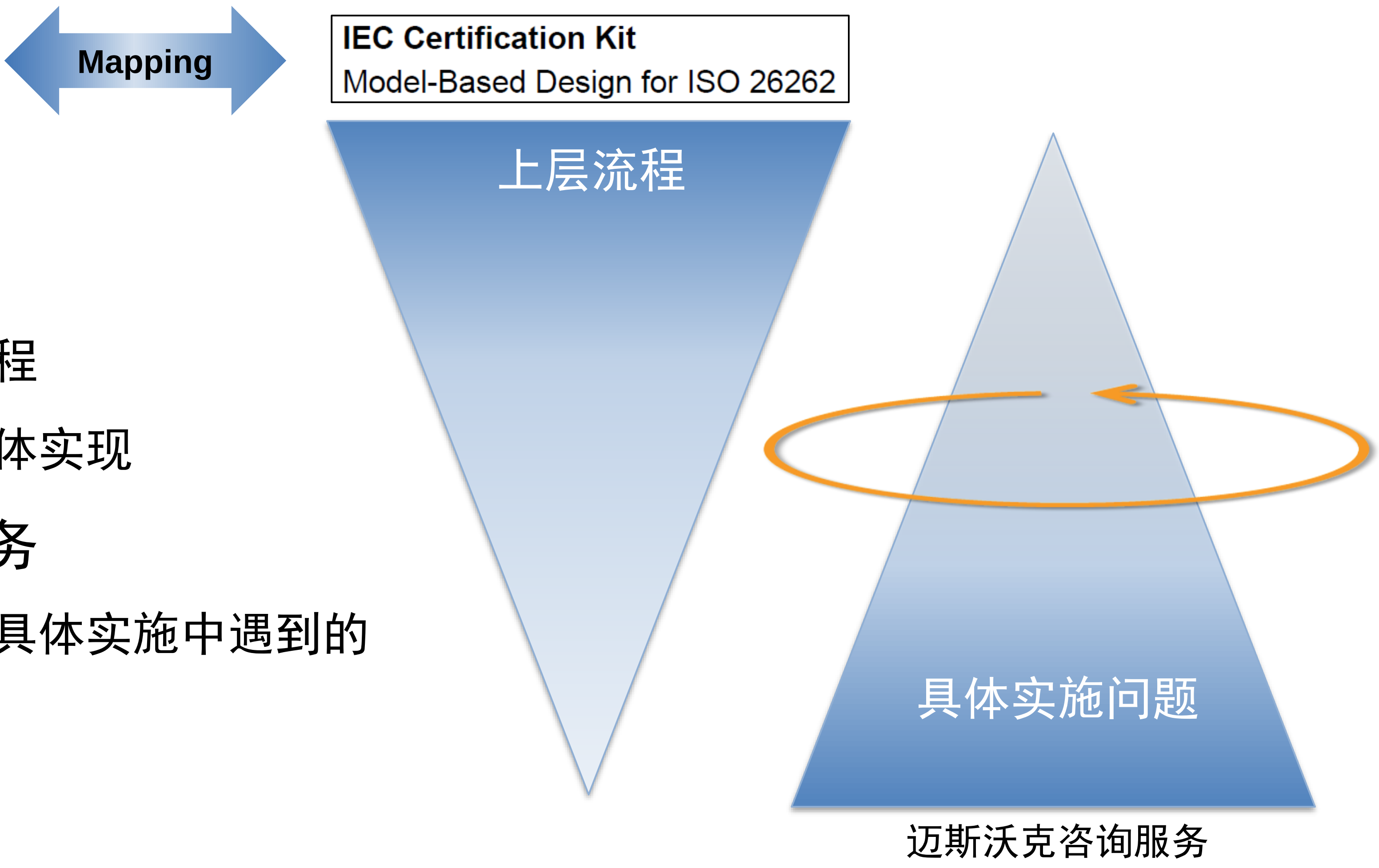
最佳实践



INTERNATIONAL STANDARD

ISO 26262-6

Second edition 2018-12

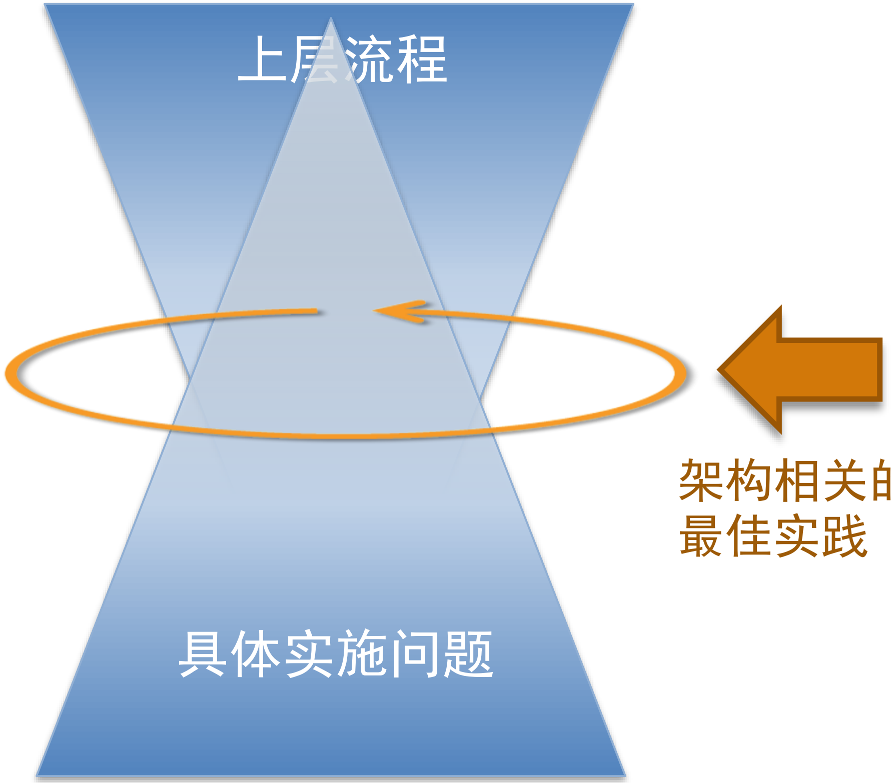


- IEC 认证包
  - 关注点在上层流程
  - 由用户来完成具体实现
- 迈斯沃克咨询服务
  - 关注点在于解决具体实施中遇到的问题
  - 提供实施建议

最佳实践

- 通过大量具体项目的实施，我们归纳出一组适合于早期考虑的模型架构方面的最佳实践
  - 以下所列最佳实践基于自上而下的代码生成方法
  - 所列满足 ISO 26262 的建模模式以外部接口的方式实现
    - 具体的代码层面的实现由底层软件完成

IEC Certification Kit
Model-Based Design for ISO 26262



迈斯沃克咨询服务

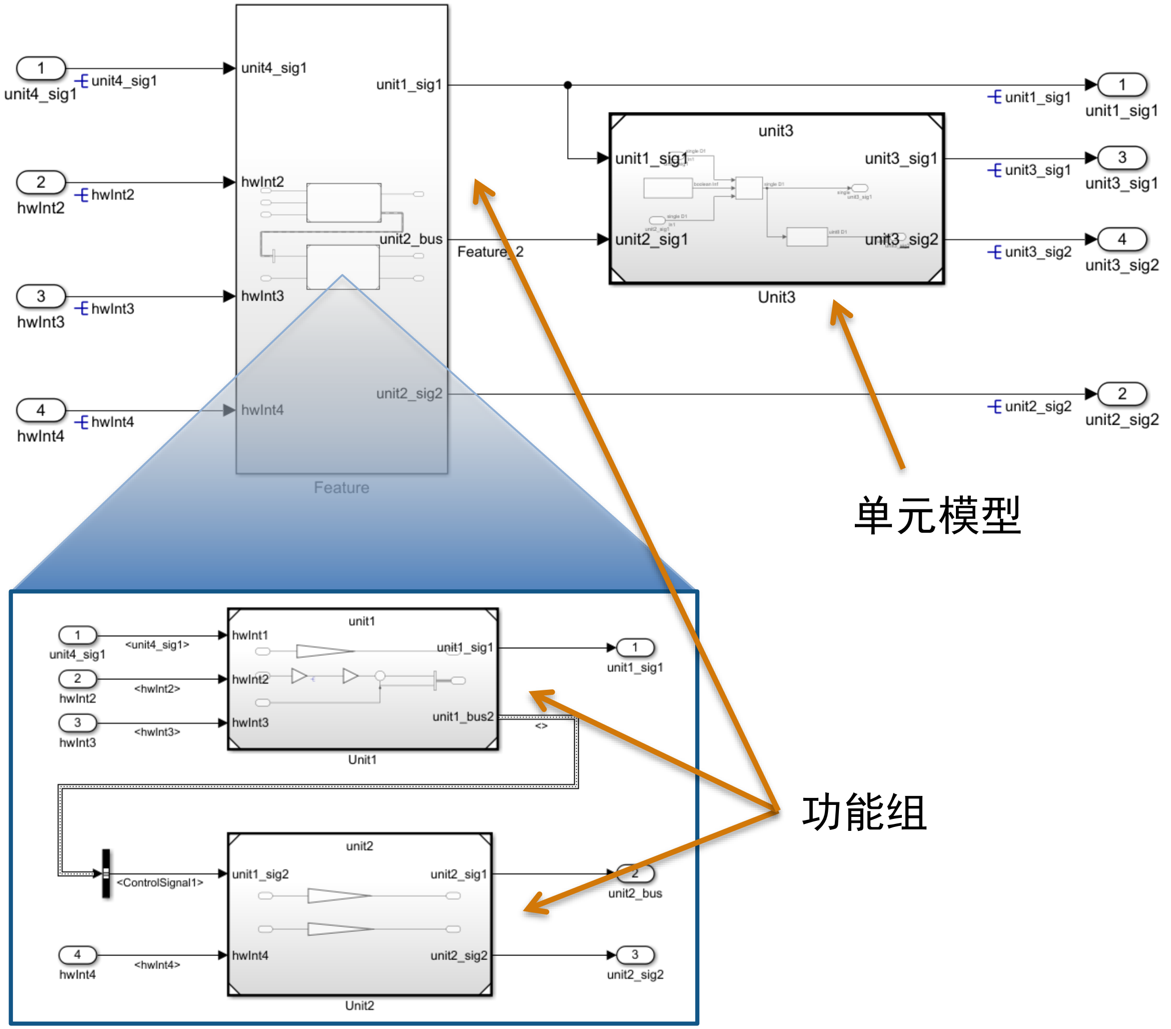
最佳实践

- 架构
- 信号传递
- 数据定义
- 代码生成

在单元级使用模型引用

架构

- 问题：
  - 算法模块化差，难以重用
  - 单元测试无法完成
  - 配置管理困难
  - 功能耦合严重
- 最佳实践：
  - 单元模型采用模型引用
  - 利用（虚拟）子系统进一步对功能进行分组，实现系统的层级化



单元模型

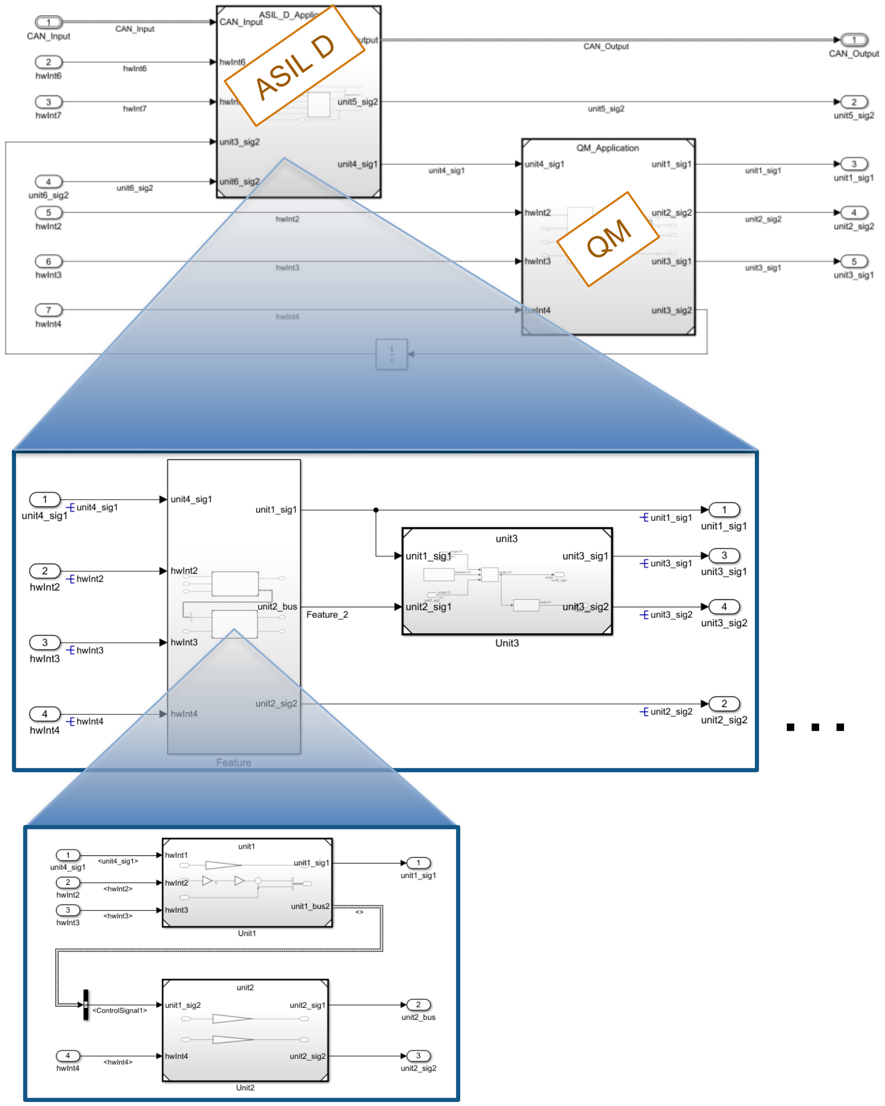
功能组

在顶层将有 ASIL 要求的和 QM 模块分开

架构

- 问题:
  - 无法实现功能模块间的互不干涉
- 最佳实践:
  - ASIL 功能和 QM 功能放在不同的模型
  - 尽量在上层生成代码从而降低代码集成的复杂性

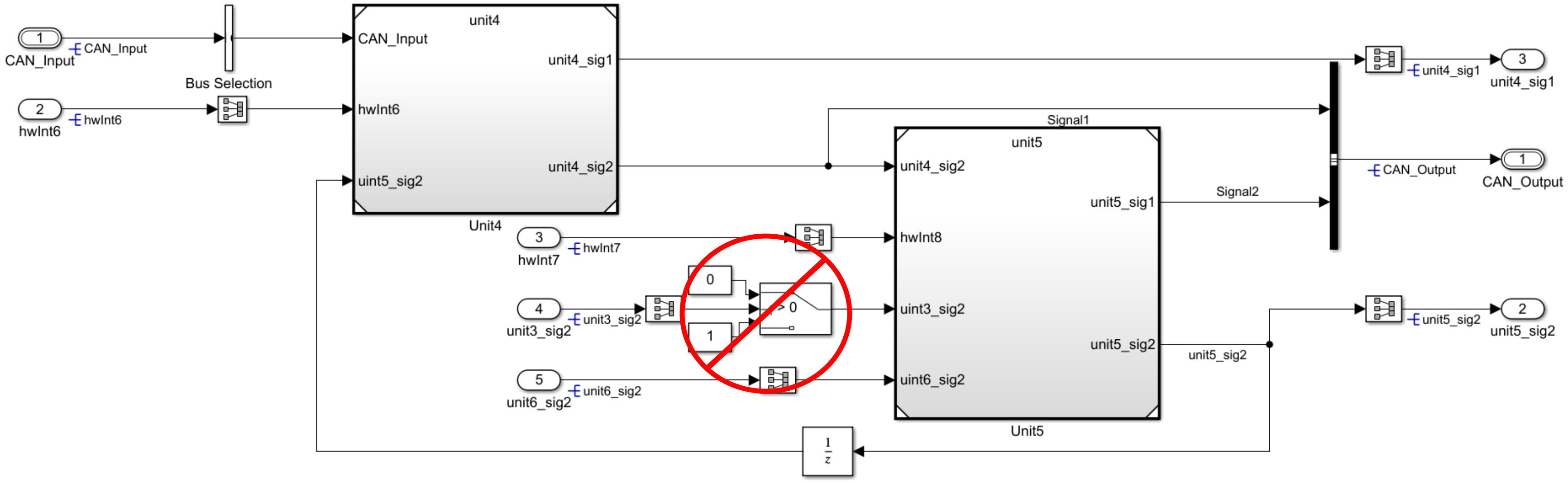
模型层级	建模模式
顶层(ASIL / QM)	顶层模型（生成代码）
集成	子系统（多层）
单元	模型应用



在模型集成层面不出现基本算法实现

架构

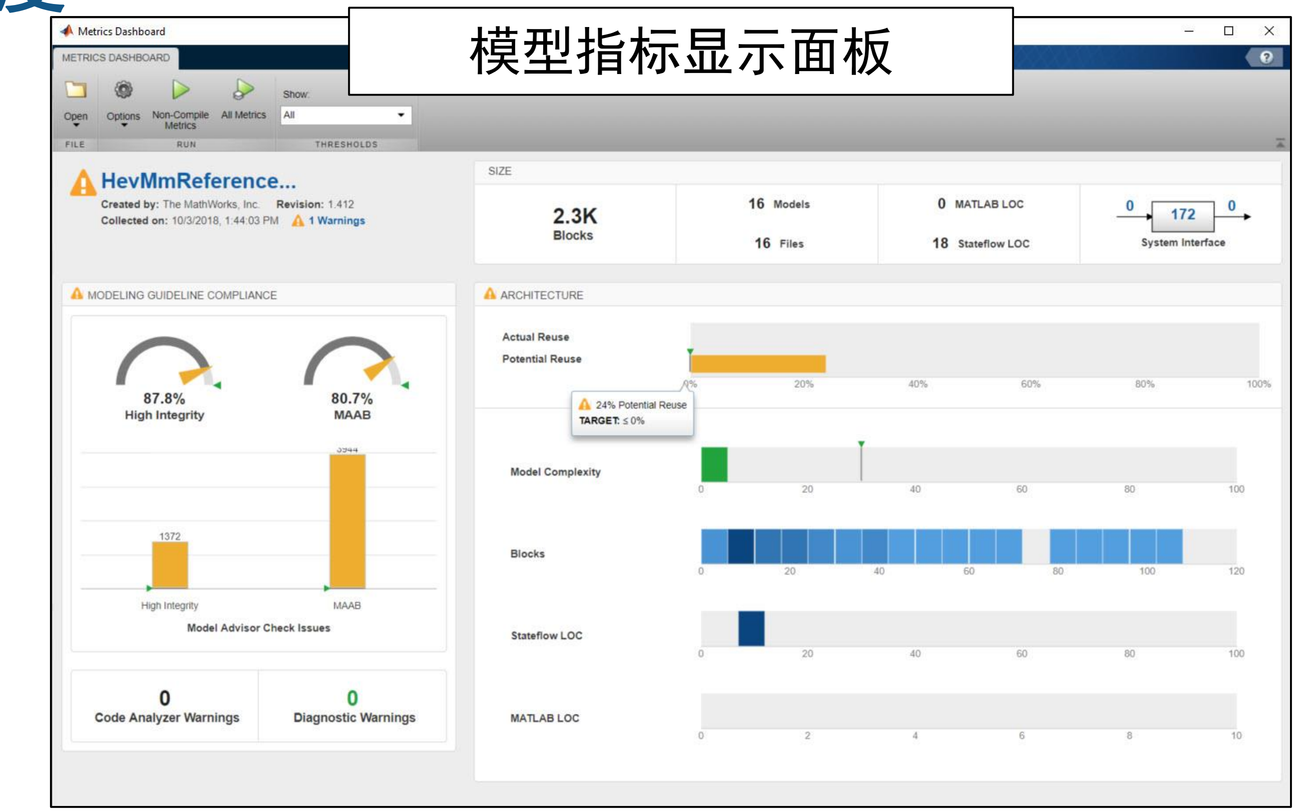
- 问题：
  - 不管是集成模型还是单元模型的测试都非常复杂
  - 需求、设计、测试之间的双向追踪困难
- 最佳实践：
  - 确保集成层面只有虚拟模块
  - 参考 (MAAB/JMAAB): db_0143: Similar block types on the model levels



利用模型指标监控单元模型复杂度

架构

- 问题：
  - 无法有效达到单元测试覆盖度要求
  - 模型验证困难
- 最佳实践：
  - 定义复杂度指标并进行必要的审查
    - I/O 数量
    - 库的重用度
    - 圈复杂度 (<=30)*
    - 基本元素个数 (<500)*
    - …
  - 使用持续集成的方法（比如Jenkins）监控模型指标



模型指标显示面板

HevMmReference...

Created by: The MathWorks, Inc. Revision: 1.412

Collected on: 10/3/2018, 1:44:03 PM 1 Warnings

24% Potential Reuse

TARGET: ≤ 0%

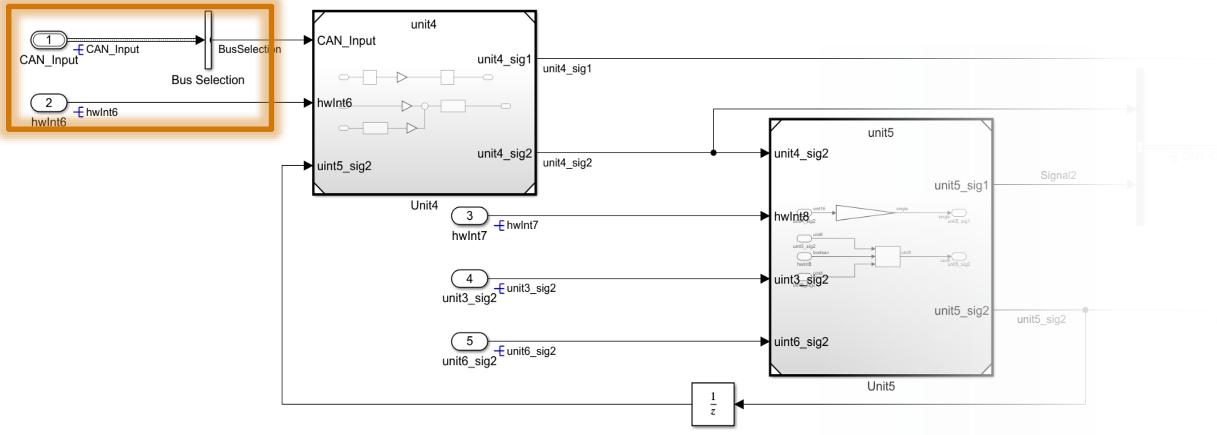
*参考文章: Model Quality Objectives

作者: Renault, Valeo, PSA, Bosch, Delphi, MathWorks

只把用到的信号传递给单元模型

信号传递

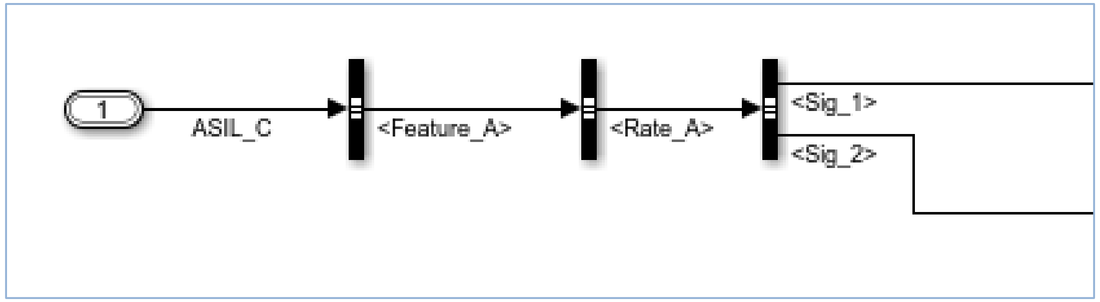
- 问题：
  - 成百上千个信号打包成总线导致测试验证难以进行
- 最佳实践：
  - 在进入单元之前使用 Bus Selector 将用到的信号抽取出来
  - 必要时可以使用虚拟子系统将总线处理模块封装起来，使得模型页面清晰



总线（Bus）设计层次化

信号传递

- 问题：
  - 多速率总线无法进入模型引用
  - 难以进行数据及信号流分析
  - 由于分类不清晰，用户可能误将 ASIL 要求低的或者 QM 的信号传递给 ASIL 要求高的模块
- 最佳实践：
  - 总线的层次要根据 ASIL 需求以及运行速率来设计
    - 类似于设计自上而下的模型层次结构

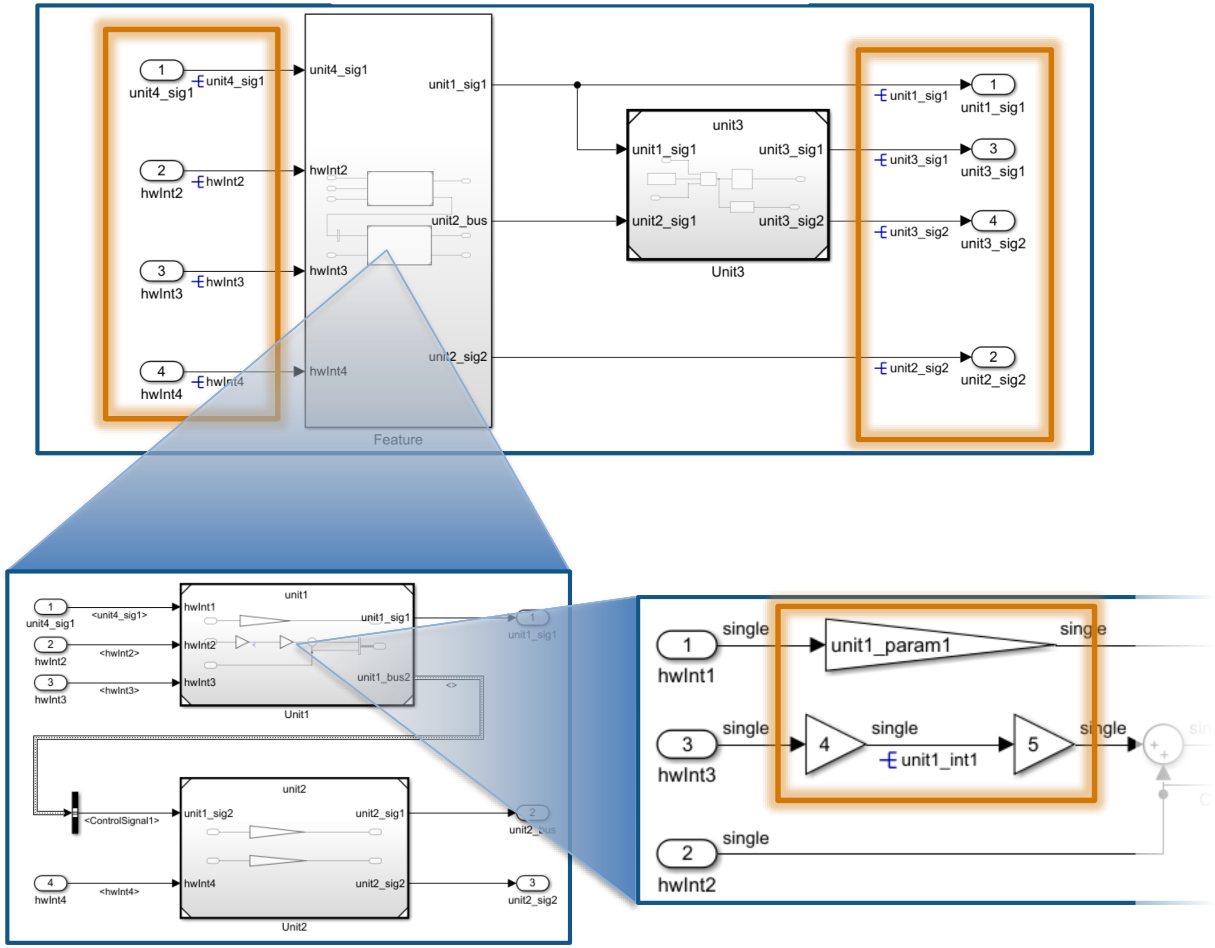


在模型上构建数据

数据定义

- 问题：
  - 由于同一信号既在输出端口上定义又在输入端口定义，结果导致系统仿真时出现信号冲突
- 最佳实践：
  - 让 Simulink 决定内部信号，只在代码生成的模型接口上定义信号对象（存储类）
    - 最大限度降低模型集成出现信号冲突的可能性。但是不一定完全适合于所有情况，比如为了打断代数环而加入的延迟模块。

用来生成代码的模型

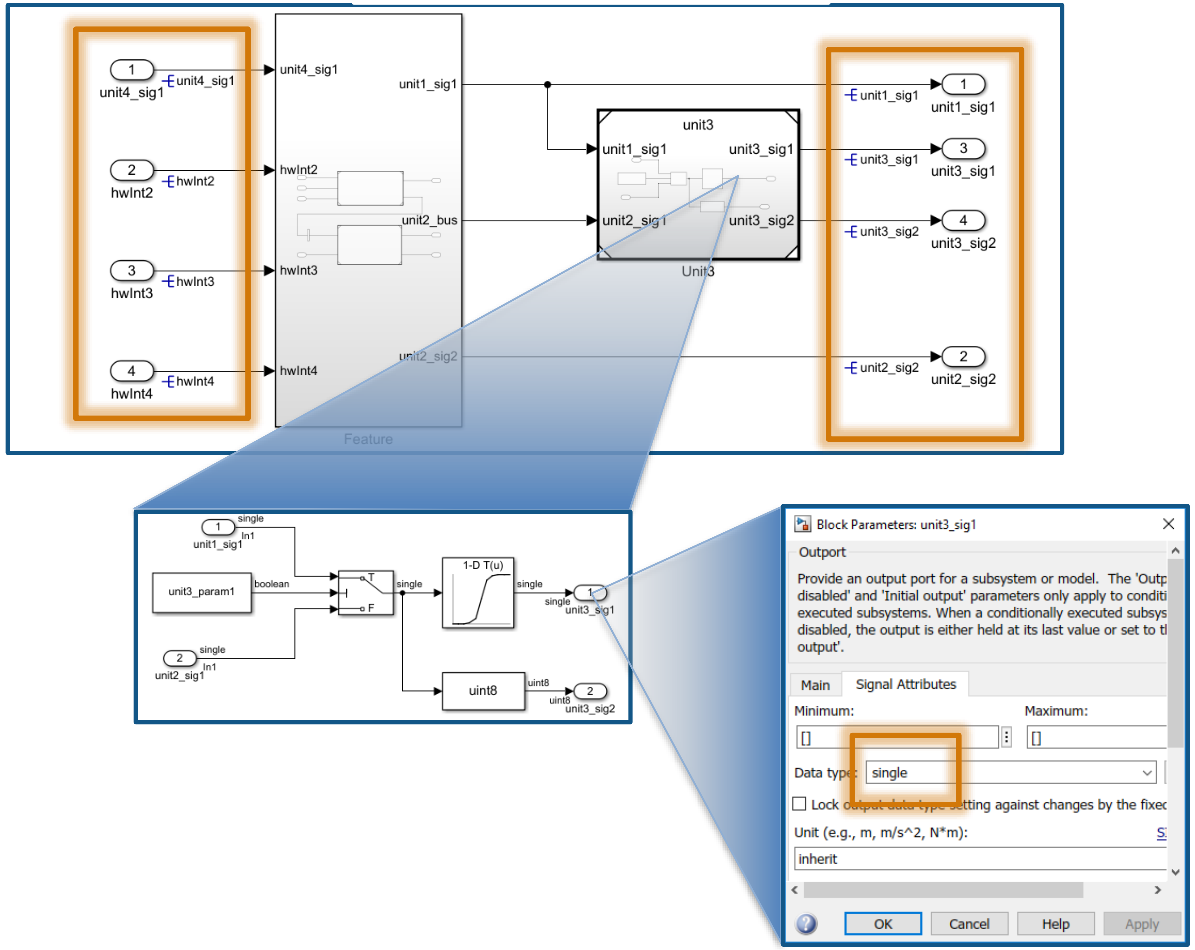


在模型上构建数据（续）

数据定义

- 除了标定量，其他的内部信号不定义信号对象
- 在单元模型最上层的输入输出端口模块上定义数据类型

用来生成代码的模型



Block Parameters: unit3_sig1

Outport

Provide an output port for a subsystem or model. The 'Outp disabled' and 'Initial output' parameters only apply to conditi executed subsystems. When a conditionally executed subsys disabled, the output is either held at its last value or set to tl output'.

Main | Signal Attributes

Minimum: [] Maximum: []

Data type: single

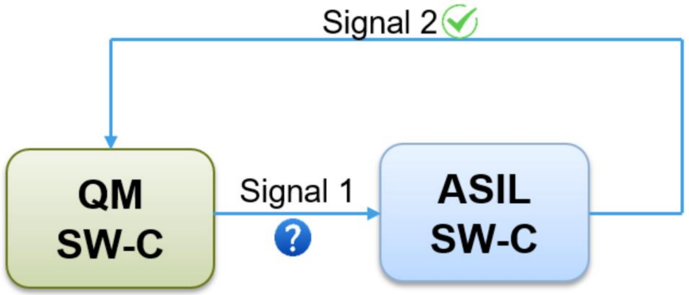
☐ Lock output data type setting against changes by the fixec

Unit (e.g., m, m/s^2, N*m): inherit

有ASIL要求的模块与QM模块间的数据保护

代码生成配置

- 问题：
  - 在 ASIL 功能和 QM 功能之间如何进行信号保护？

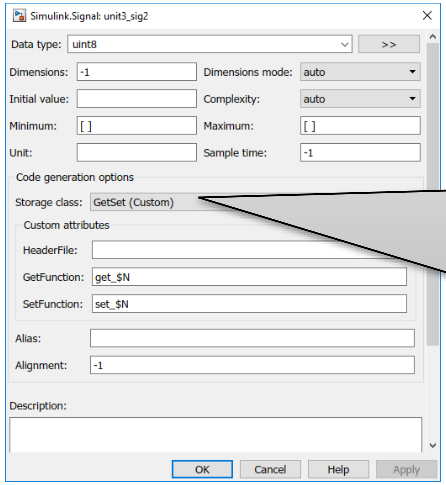


有 ASIL 要求的模块与 QM 模块间的数据保护（续）

代码生成配置

- 问题：
  - 在 ASIL 功能和 QM 功能之间如何进行信号保护？
- 最佳实践：
  - 使用 Get/Set 存储类

Get/Set 存储类



Simulink.Signal: unit3_sig2

Data type: uint8

Dimensions: -1 Dimensions mode: auto

Initial value: Complexity: auto

Minimum: [ ] Maximum: [ ]

Unit: Sample time: -1

Code generation options

Storage class: GetSet (Custom)

Custom attributes

HeaderFile:

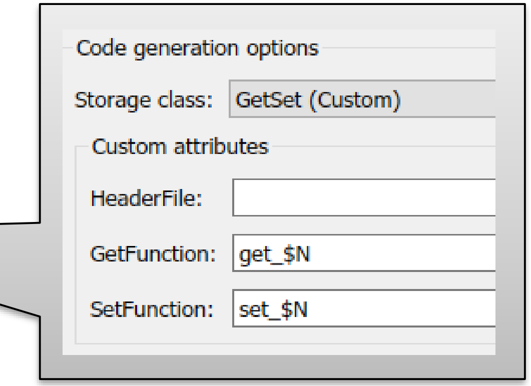
GetFunction: get_$N

SetFunction: set_$N

Alias:

Alignment: -1

Description:



Code generation options

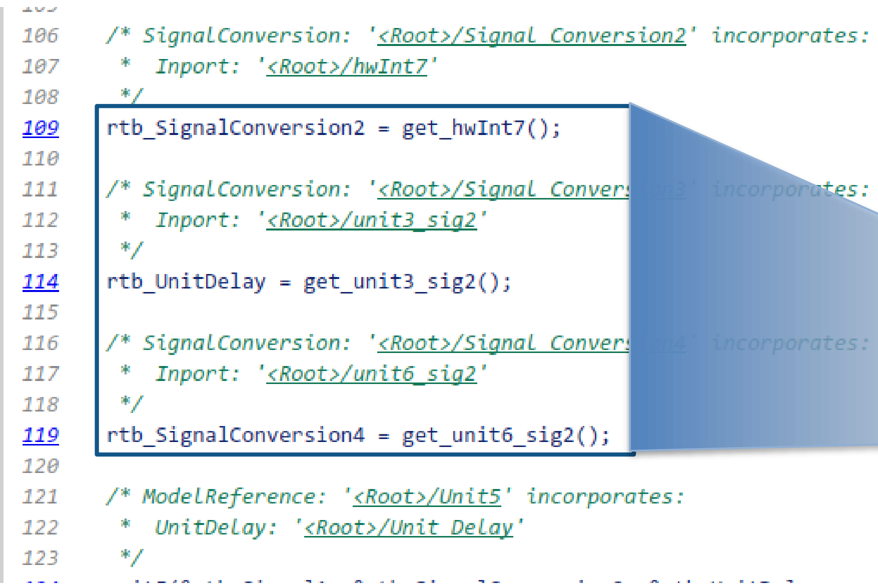
Storage class: GetSet (Custom)

Custom attributes

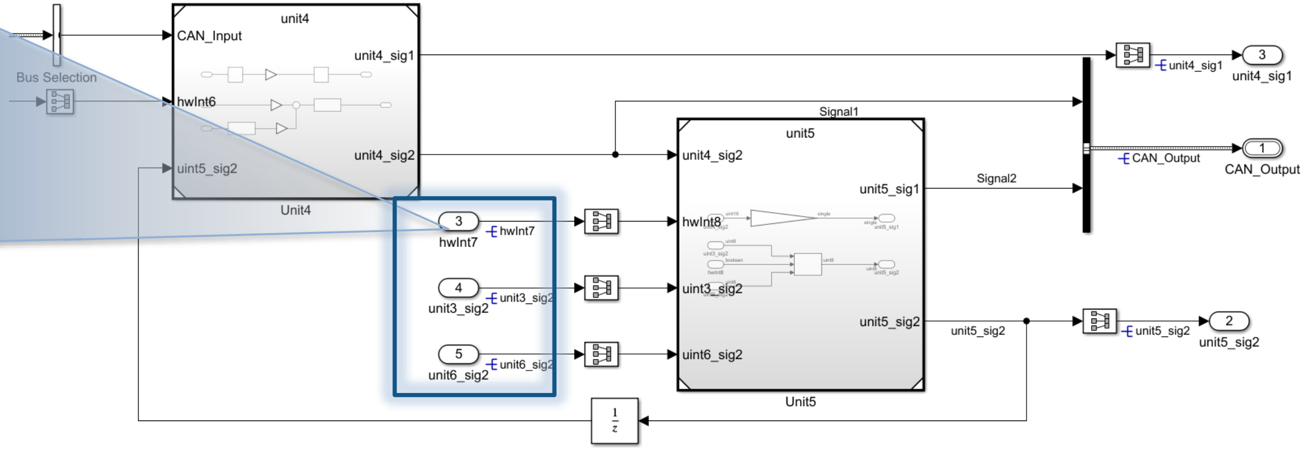
HeaderFile:

GetFunction: get_$N

SetFunction: set_$N



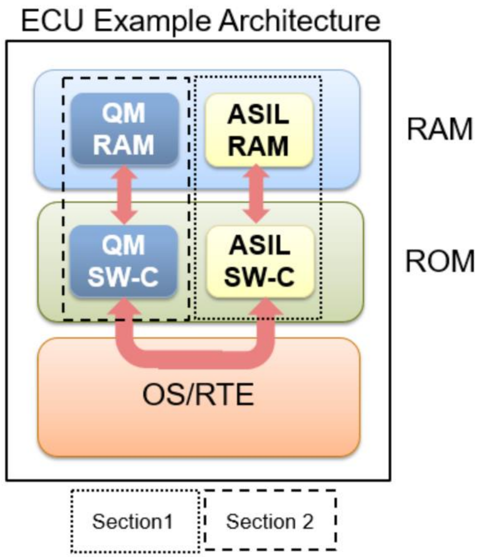
```
/* SignalConversion: '<Root>/Signal Conversion2' incorporates:
 *  Inport: '<Root>/hwInt7'
 */
rtb_SignalConversion2 = get_hwInt7();

/* SignalConversion: '<Root>/Signal Conversion3' incorporates:
 *  Inport: '<Root>/unit3_sig2'
 */
rtb_UnitDelay = get_unit3_sig2();

/* SignalConversion: '<Root>/Signal Conversion4' incorporates:
 *  Inport: '<Root>/unit6_sig2'
 */
rtb_SignalConversion4 = get_unit6_sig2();

/* ModelReference: '<Root>/Unit5' incorporates:
 *  UnitDelay: '<Root>/Unit Delay'
 */
```

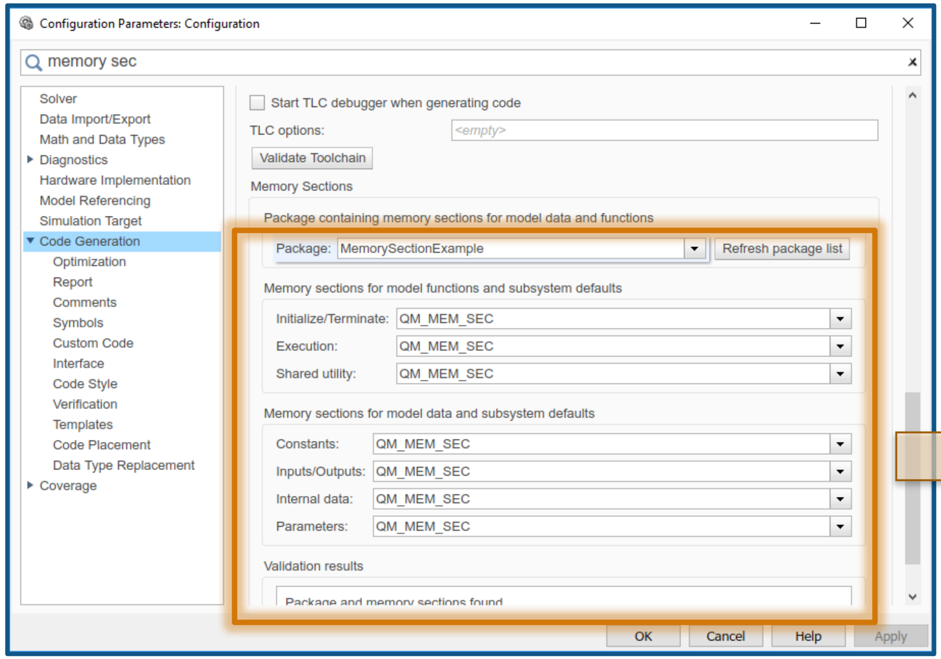


不同 ASIL 级别以及 QM 之间存储单元分区

代码生成配置

- 问题：
  - 生成的代码如何实现不同 ASIL 要求以及 QM 要求的代码放到不同的存储区域？
- 最佳实践：
  - 配置代码生成的 Memory Section 属性


ECU Example Architecture


Configuration Parameters: Configuration

memory sec

□ Start TLC debugger when generating code

TLC options: <empty>

Memory Sections

Package containing memory sections for model data and functions

Package: MemorySectionExample

Memory sections for model functions and subsystem defaults

Initialize/Terminate: QM_MEM_SEC

Execution: QM_MEM_SEC

Shared utility: QM_MEM_SEC

Memory sections for model data and subsystem defaults

Constants: QM_MEM_SEC

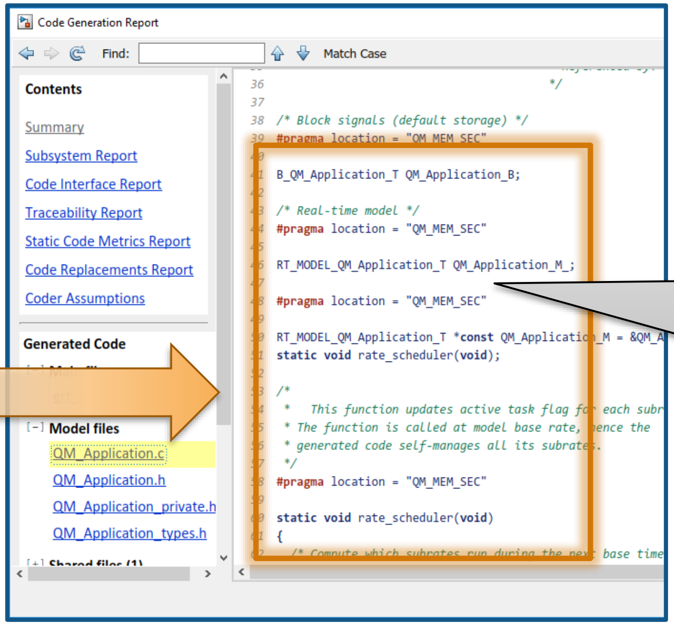
Inputs/Outputs: QM_MEM_SEC

Internal data: QM_MEM_SEC

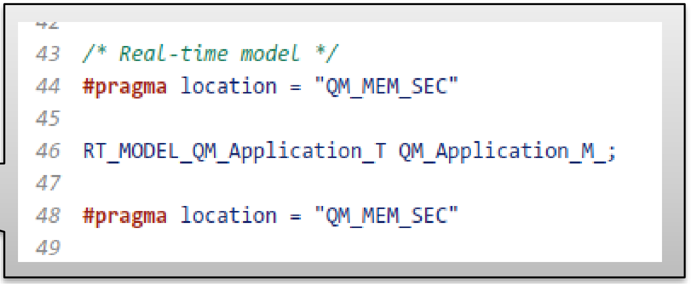
Parameters: QM_MEM_SEC

Validation results

Package and memory sections found


Code Generation Report

```
                                               */

/* Block signals (default storage) */
#pragma location = "QM_MEM_SEC"

B_QM_Application_T QM_Application_B;

/* Real-time model */
#pragma location = "QM_MEM_SEC"

RT_MODEL_QM_Application_T QM_Application_M_;

#pragma location = "QM_MEM_SEC"

RT_MODEL_QM_Application_T *const QM_Applicatio_M = &QM_A
static void rate_scheduler(void);

/*
 *   This function updates active task flag fo  each subr
 * The function is called at model base rate, hence the
 * generated code self-manages all its subrates.
 */
#pragma location = "QM_MEM_SEC"

static void rate_scheduler(void)
{
  /* Compute which subrates run during the next base time
```

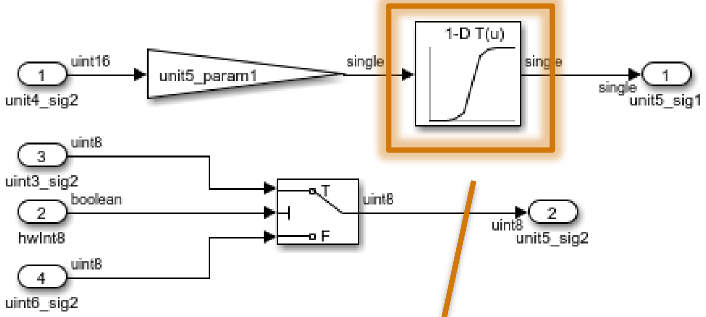


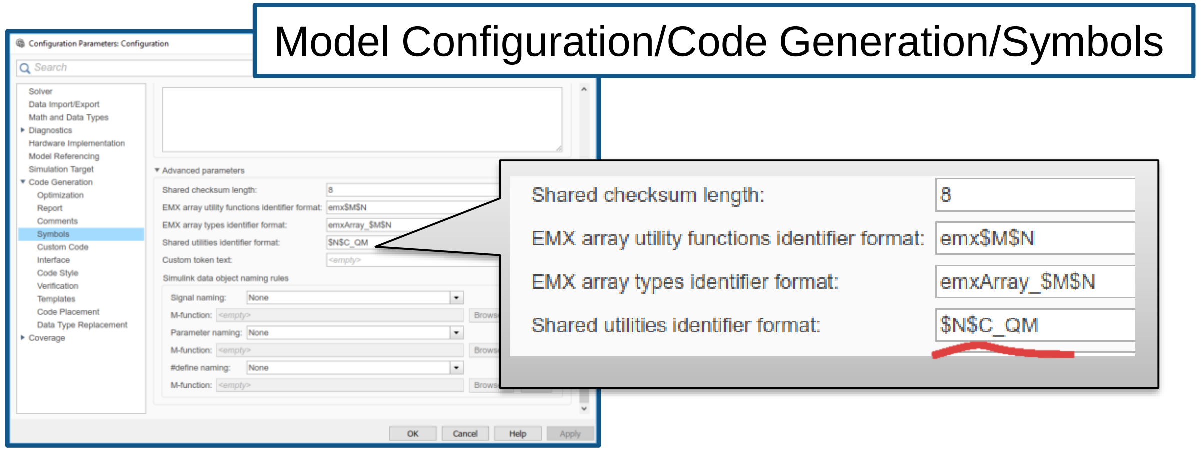
```
/* Real-time model */
#pragma location = "QM_MEM_SEC"

RT_MODEL_QM_Application_T QM_Application_M_;

#pragma location = "QM_MEM_SEC"

```

共享代码采用不同的命名标识

代码生成配置

- 问题：
  - ASIL 和 QM 要求的引用相同的函数（代码共享）时如何避免交叉干扰？
- 最佳实践：
  - 配置共享代码标识





Model Configuration/Code Generation/Symbols

Configuration Parameters: Configuration

Advanced parameters

Shared checksum length: 8

EMX array utility functions identifier format: emx$M$N

EMX array types identifier format: emxArray_MN

Shared utilities identifier format: NC_QM

Custom token text: <empty>

Simulink data object naming rules

Signal naming: None

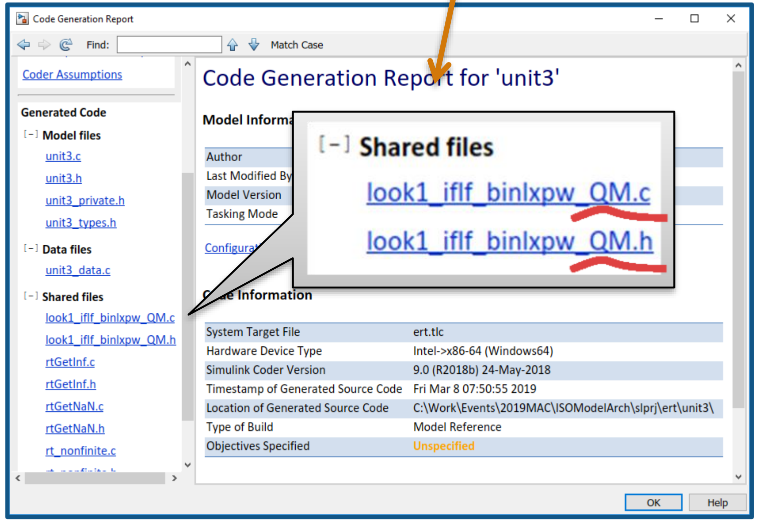
M-function: <empty>

Parameter naming: None

M-function: <empty>

#define naming: None

M-function: <empty>



Code Generation Report

Generated Code

[-] Model files

unit3.c
unit3.h
unit3_private.h
unit3_types.h

[-] Data files

unit3_data.c

[-] Shared files

look1_iflf_binlxpw_QM.c
look1_iflf_binlxpw_QM.h
rtGetInf.c
rtGetInf.h
rtGetNaN.c
rtGetNaN.h
rt_nonfinite.c

Code Generation Report for 'unit3'

Model Informa

Author
Last Modified By
Model Version
Tasking Mode

Configurat

[-] Shared files

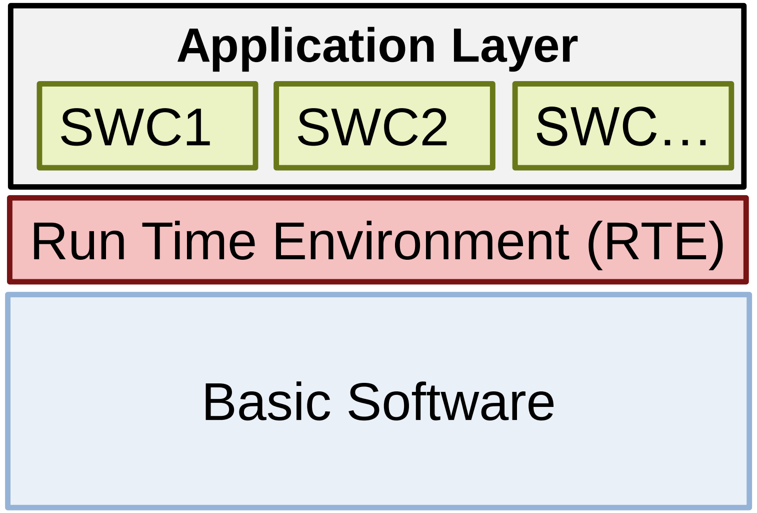
look1_iflf_binlxpw_QM.c
look1_iflf_binlxpw_QM.h

Code Information

System Target File	ert.tlc
Hardware Device Type	Intel->x86-64 (Windows64)
Simulink Coder Version	9.0 (R2018b) 24-May-2018
Timestamp of Generated Source Code	Fri Mar 8 07:50:55 2019
Location of Generated Source Code	C:\Work\Events\2019MAC\ISOModelArch\slprj\ert\unit3\
Type of Build	Model Reference
Objectives Specified	Unspecified

AUTOSAR 实施

- 以上最佳实践兼容于 AUTOSAR
- 许多配置和定制化都可以在 AUTOSAR 架构内实现
- AUTOSAR 具体实现需要做一些调整，比如:
  - 以上的 Get/Set 就要用 Sender/Receiver 端口代替，在 RTE 中实现数据保护
- IEC认证包中的参考流程支持 AUTOSAR 架构



AUTOSAR 层次化架构

注：一篇关于AUTOSAR 架构满足 ISO 26262 要求的论文正在准备中

总结

- 基于技术咨询的项目实践
- ISO 26262 建模最佳实践
  - 在单元级使用模型引用
  - 在顶层将有 ASIL 要求的和 QM 模块分开
  - 在模型集成层面不出现基本算法实现
  - 利用模型指标监控单元模型复杂度
  - 只把用到的信号传递给单元模型
  - 总线（Bus）设计层次化
  - 在模型上构建数据
  - 有 ASIL 要求的模块与 QM 模块间的数据保护
  - 不同 ASIL 级别以及 QM 之间存储单元分区
  - 共享代码采用不同的命名标识
  - AUTOSAR 实施

更多 ISO 26262 建模最佳实践，请联系我们的技术咨询部门 zli@mathworks.com