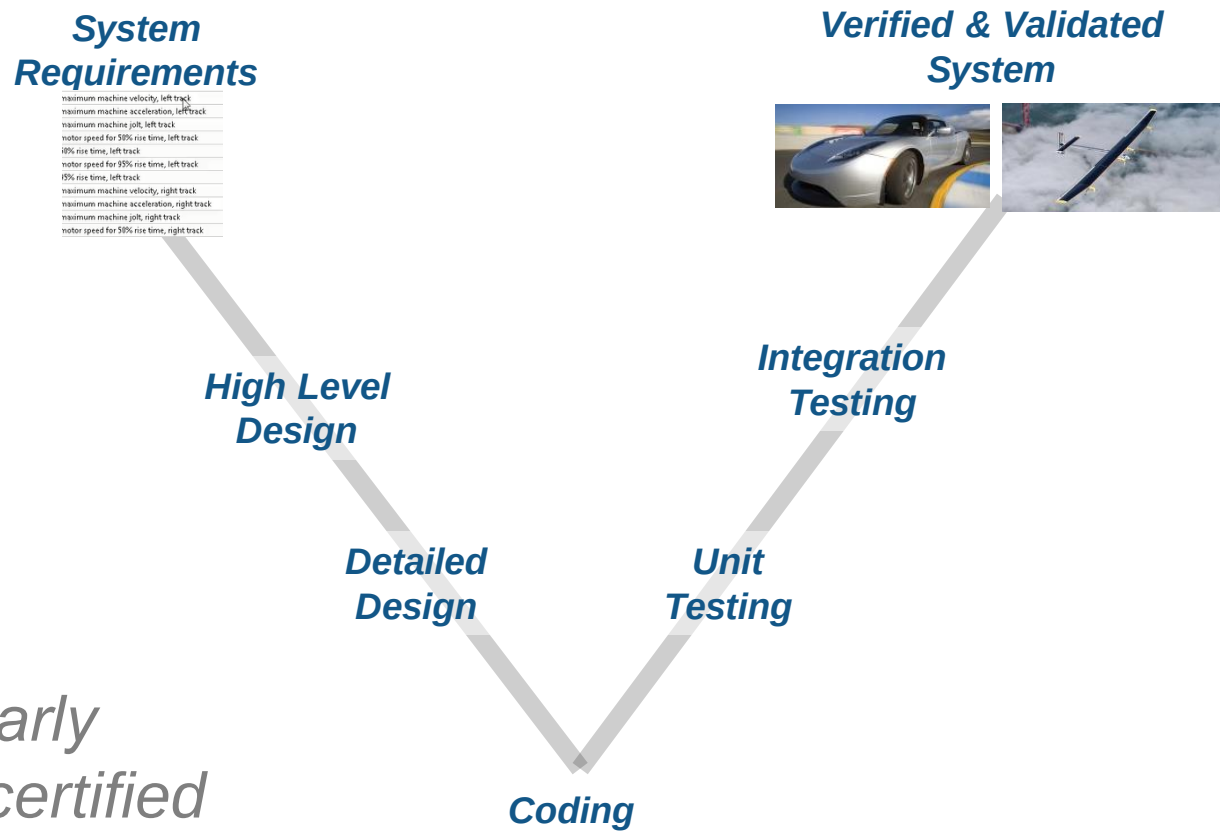
模型化基礎設計如何協助符合ISO 26262規範的開發需求

Jerry Tung
鈦思科技工程部經理

Key Takeaways

1. Find bugs early, develop high quality software
2. Replace manual verification tasks with workflow automation
3. Learn about reference workflow that conforms to safety standards



"Reduce costs and project risk through early verification, shorten time to market on a certified system, and deliver high-quality production code that was first-time right" Michael Schwarz, ITK Engineering

Safety of Electronic Systems

- Critical functionality in industries such as Automotive, Aerospace, Medical, Industrial Automation
- Real-time operation
  - Compute time lag cannot be tolerated
- Predictable behavior
  - No unintended functionality
- Must be robust
  - Program crash or reboot not allowed

Role of Certification Standards

- ISO 26262 (Automotive)
  - Defines functional safety for automotive electronic systems
  - Automotive Safety Integrity Level ASIL QM, A to D (least to most; derived from severity, controllability, probability)
  - ISO 26262-6 pertains to software development, verification, and validation
- DO-178 (Avionics)
  - Guidelines for the safety of software in certain airborne systems
  - Level A to E (most critical to least)
  - Verification activities include review of requirements and code, testing of software, code coverage
- IEC 62304 (Medical Device)
  - Describes software development and maintenance processes for medical device software
  - Safety levels Class A to C (least critical to most)
  - Identifies various verification and testing activities
- IEC 61508 (Industrial Automation & Machinery)
  - General functional safety standard, originally for process control industry
  - Safety Integrity Level SIL 1 to 4 (least to most; derived from exposure to demand needs and probability of failure)
  - Defines the software requirements and lifecycle for software, that includes validation and verification

Addressing Design and Development Challenges

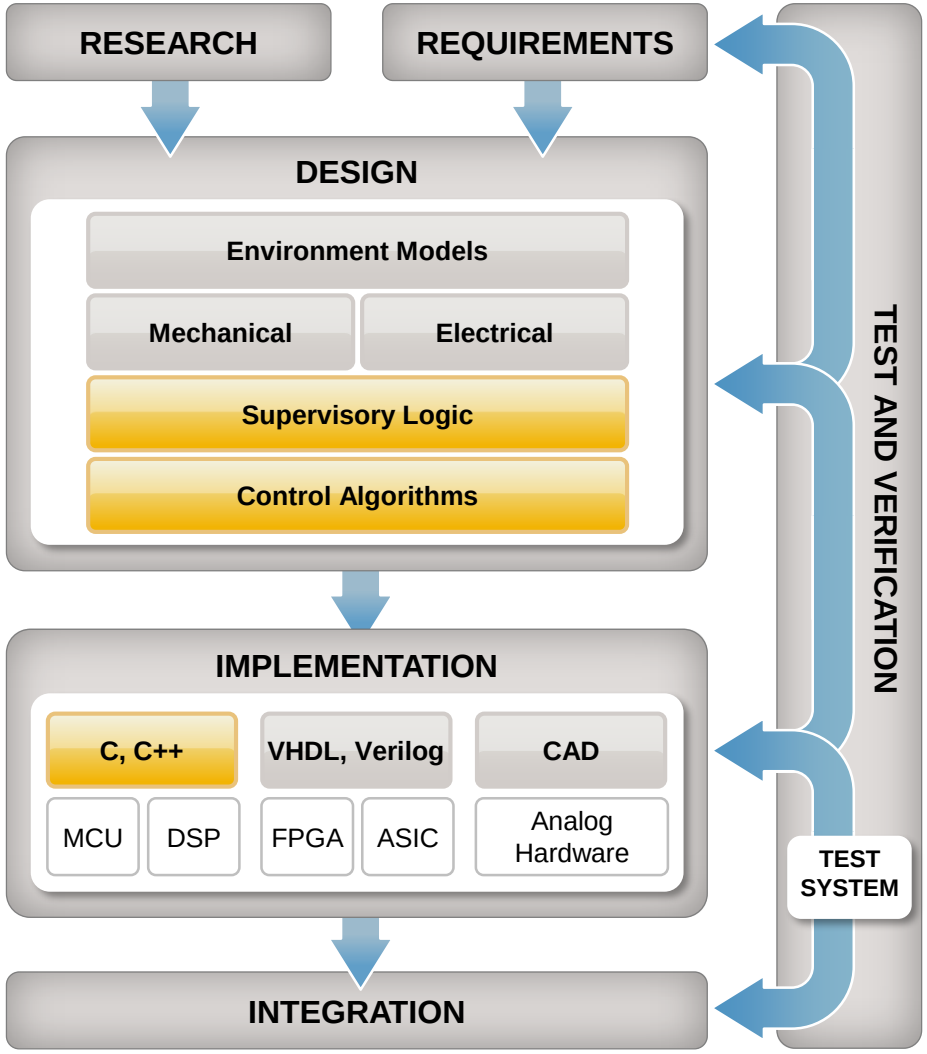
It is easier and less expensive to fix design errors early in the process when they happen.

Model-Based Design enables:

1. *Early testing to increase confidence in your design*
2. *Delivery of higher quality software for production use*
3. *Produce artifacts for certification to satisfy safety standards*

Model Based Design

- Modeling
  - Model algorithms and environment
  - Explore design alternatives and options
- Simulation
  - Design exploration with simulation
  - Find issues early, on your desktop PC
- Production code
  - Code generated automatically from model
  - Early verification for high quality code



ISO 26262 “Road Vehicles - Functional Safety”

- Functional safety standard for passenger cars
- **First edition published 11th November 2011**
- Facilitates modern software engineering concepts such as
  - **Model-Based Design**
  - **Early verification and validation**
  - **Code generation**
- Provides objectives / requirements for
  - Development process activities (software safety life cycle)
  - Development and verification tools (tool qualification)

> *• Safety deals with the protection against hazards and risks that originate from the operation of the system (product, device)*
>
> *• Functional safety is the absence of unreasonable risk due to hazards caused by malfunctioning behavior of E/E systems*

ISO 26262:
Applicable Model-Based Design Sections

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 and operation
ISO 26262-8 ISO 26262-8-11	Supporting Processes Qualification of software tools
ISO 26262-9	ASIL-oriented and safety oriented analysis
ISO 26262-10	Guideline

- ***Model-Based Design definitions***

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

- ***Tool classification and qualification***

ISO 26262-1 Vocabulary:
Definition of Model-Based Development

ISO/DIS 26262-1

1.74 model-based development

development that uses models to describe the functional behavior of the elements which are to be developed

NOTE Depending on the level of abstraction used for such a model it can be used for simulation or code generation or both.

ISO 26262-6 Product Development at the Software Level: Guidance for Model-Based Design Activities

Table 11 — Methods for the informal verification of software unit design and implementation

Methods		ASIL			
		A	B	C	D
1a	Inspection of the software unit design	+	++	++	++
1b	Walkthrough of the software unit design	++	+	O	o
1c	Model Inspection[a]	+	++	++	++
1d	Model Walkthrough[a]	++	+	O	o
1e	Inspection of the source code[b]	+	++	++	++
1f	Walkthrough of the source code[b]	++	+	O	o

[a] In the case of model-based software development the software unit specification can be verified at the model level by applying Method 1c and 1d.

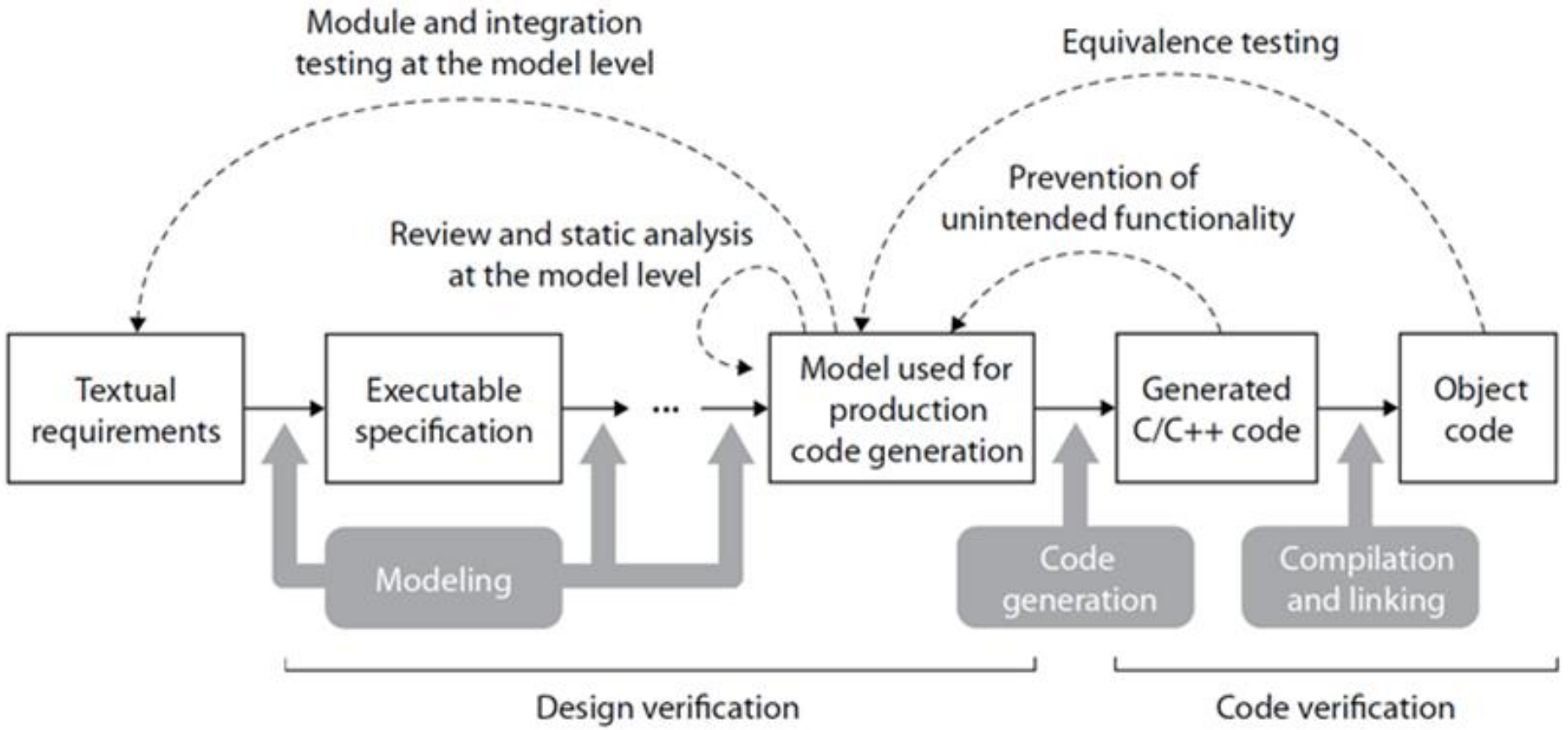
[b] In the case of model-based development with automatic code generation, methods for informal verification of the generated code can be replaced by automated methods and techniques if applicable.

- ISO 26262-6 supports MBD activities like early verification at the model level and automated verification of the generated code

Model-Based Design and ISO 26262 Summary

- Model-Based Design is one of the software development paradigms explicitly addressed in ISO 26262
- Model-Based Design workflows are an integral part of *ISO 26262-6 Product Development: Software Level*
  - Modeling
  - Early verification and validation
  - Code generation

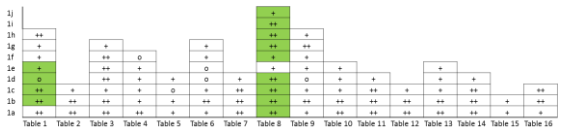
Reference Workflow - Model-Based Design for ISO 26262



→ Workflow with qualified tool use cases for developing safety-related software

→ An integrated verification and validation process with code generation

Polyspace MISRA Checking **Credits**



MISRA Only

Full Polyspace

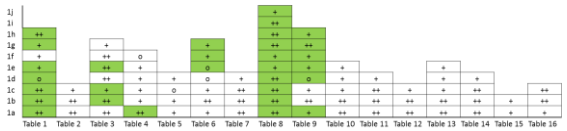
Full Model-Based Design

	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	Table 16
1j								+								
1i								++								
1h	++							++	+							
1g	+		+			+		++	++							
1f	+		++	o		+		+	+							
1e	+		++	+		o		+	+	+			+			
1d	o		++	+	+	o	+	++	o	+	+		+	+		
1c	++	+	+	+	o	+	++	++	+	+	++	+	+	++		++
1b	++	++	++	+	+	++	++	++	++	++	++	++	++	++	+	++
1a	++	++	++	++	+	+	++	++	+	++	++	++	++	++	+	+

	Applies to method
++	High Recommended
+	Recommended
o	No Recommendation

Full Polyspace **Credits**

MISRA Only



Full Polyspace

Full Model-Based Design

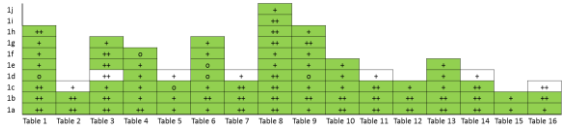
	Table 1	Table 2	Table 3	Table 4	Table 5	Table 6	Table 7	Table 8	Table 9	Table 10	Table 11	Table 12	Table 13	Table 14	Table 15	Table 16
1j								+								
1i								++								
1h	++							++	+							
1g	+		+			+		++	++							
1f	+		++	o		+		+	+							
1e	+		++	+		o		+	+	+			+			
1d	o		++	+	+	o	+	++	o	+	+		+	+		
1c	++	+	+	+	o	+	++	++	+	+	++	+	+	++		++
1b	++	++	++	+	+	++	++	++	++	++	++	++	++	++	+	++
1a	++	++	++	++	+	+	++	++	+	++	++	++	++	++	+	+

	Applies to method
++	High Recommended
+	Recommended
o	No Recommendation

Model-Based Design Workflow **Credits**

MISRA Only

Full Polyspace



Full Model-Based Design

	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	Table 16
1j								+								
1i								++								
1h	++							++	+							
1g	+		+			+		++	++							
1f	+		++	o		+		+	+							
1e	+		++	+		o		+	+	+			+			
1d	o		++	+	+	o	+	++	o	+	+		+	+		
1c	++	+	+	+	o	+	++	++	+	+	++	+	+	++		++
1b	++	++	++	+	+	++	++	++	++	++	++	++	++	++	+	++
1a	++	++	++	++	+	+	++	++	+	++	++	++	++	++	+	+

	Applies to method
++	High Recommended
+	Recommended
o	No Recommendation

Example – Not Covered by MW Tools

Table 5 – Mechanisms for Error Handling at the Software Architectural Level

Methods		ASIL				Applicable Model-Based Design Tools and Processes	Comments
		A	B	C	D		
1a	Static recovery mechanism	+	+	+	+	Simulink Stateflow	Simulink and Stateflow can be used to design fault detection, isolation, and recovery (FDIR) algorithms.
1b	Graceful degradation	+	+	++	++	Stateflow	Stateflow can be used to design graceful degradation behaviour.
1c	Independent parallel redundancy	o	o	+	++		
1d	Correcting codes for data	+	+	+	+		

Expected Updates in the 2nd edition

ISO 26262 v2 Highlights
Testing -> Verification

- Renamed Sections 9 and 10
  - Testing -> Verification
- Section 9 (SW Unit Verification)
  - Static verification methods now explicitly shown in the Tables 7
    - Walk-Through
    - Inspection
    - Formal and Semi-formal verification
    - Static analysis

Table 7 — Methods for software unit verification

	Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a	Walk-through [a]	++	+	o	o
1b	Pair-programming [a]	+	+	+	+
1c	Inspection [a]	+	++	++	++
1d	Semi-formal verification	+	+	++	++
1e	Formal verification	o	o	+	+
1f	Control flow analysis [b, c]	+	+	++	++
1g	Data flow analysis [b, c]	+	+	++	++
1h	Static code analysis [d]	++	++	++	++
1i	Static analyses based on abstract interpretation [e]	+	+	+	+
1j	Requirements-based test [f]	++	++	++	++
1k	Interface test [g]	++	++	++	++
1l	Fault injection test [h]	+	+	+	++
1m	Resource usage evaluation [i]	+	+	+	++
1n	Back-to-back comparison test between model and code, if applicable [j]	+	+	++	++

3.182
walk-through
systematic examination of *work products* (3.185) in order to detect *safety anomalies* (3.134)

EXAMPLE During a walk-through, the developer explains the *work product* (3.185) step-by-step to one or more reviewers. The objective is to create a common understanding of the *work product* (3.185) and to identify any *safety anomalies* (3.134) within the *work product* (3.185). Both *inspections* (3.82) and walk-throughs are types of peer *review* (3.127), where a walk-through is a less stringent form of peer *review* (3.127) than an *inspection* (3.82).

3.82
inspection
examination of *work products* (3.185), following a formal procedure, in order to detect *safety anomalies* (3.134)

Note 3 to entry: A formal procedure normally includes a previously defined procedure, checklist, moderator and *review* (3.127) of the results.

Clarifications for Code Reviews for MBD

- Code reviews are not required "*if evidence that justifies confidence in the code generator used*"
- Our IEC certification kit provides the evidence needed to justify confidence
  - Reference workflow to detect and mitigate any translation errors
- Added "Static analyses based on abstract interpretation"

b) the compliance of the source code with its design specification;

NOTE 3 In the case of model-based development, requirement b) still applies.

	Methods	ASIL A	ASIL B	ASIL C	ASIL D
1a	Walk-through [a]	++	+	o	o
1b	Pair-programming [a]	+	+	+	+
1c	Inspection [a]	+	++	++	++
1d	Semi-formal verification	+	+	++	++
1e	Formal verification	o	o	+	+
1f	Control flow analysis [b, c]	+	+	++	++
1g	Data flow analysis [b, c]	+	+	++	++
1h	Static code analysis [d]	++	++	++	++
1i	Static analyses based on abstract interpretation [e]	+	+	+	+
1j	Requirements-based test [f]	++	++	++	++
1k	Interface test [g]	++	++	++	++
1l	Fault injection test [h]	+	+	+	++
1m	Resource usage evaluation [i]	+	+	+	++
1n	Back-to-back comparison test between model and code, if applicable [j]	+	+	++	++

[a] For model-based development these methods are applied at the model level, if evidence is available that justifies confidence in the code generator used.

Clarifications on Structural Coverage for MBD

- Structural coverage can be done at model level "*if it is shown to be equivalent with rationales based on evidence that the coverage is representative of the code level*"

- Our IEC certification kit provides the evidence needed to justify confidence
  - Reference workflow to detect and mitigate any translation errors
    - Limited Trace Review is called out as an alternative method to code coverage

Table 9 — Structural coverage metrics at the software unit level

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

NOTE 2 The structural coverage can be determined by the use of appropriate software tools.

NOTE 3 In the case of model-based development, the analysis of structural coverage can be performed at the model level using analogous structural coverage metrics for models.

EXAMPLE 4 The analysis of structural coverage performed at the model level may replace the source code coverage metrics if it is shown to be equivalent with rationales based on evidence that the coverage is representative of the code level.

Other MBD Aspects Clarification

- Reworked Annex B for MBD
  - More detailed and broken to the sections mapping to the ISO structure
  - Better examples provided for modelling limitations (accuracy, discretization)
- Clarification for implementation and design definition (8.2)
  - Model can be SW design but not implementation (code)
  - Was vague in the Rev 1
- Clarification for auto test generation

Annex B (informative) **Model-based development approaches**

B.1 Objectives

This annex explains possible usage benefits and potential issues of model-based development approaches (MBDV) during the development at the software level.

NOTE This annex does not imply that the model-based development approaches mentioned are restricted to software development only.

8.2 General

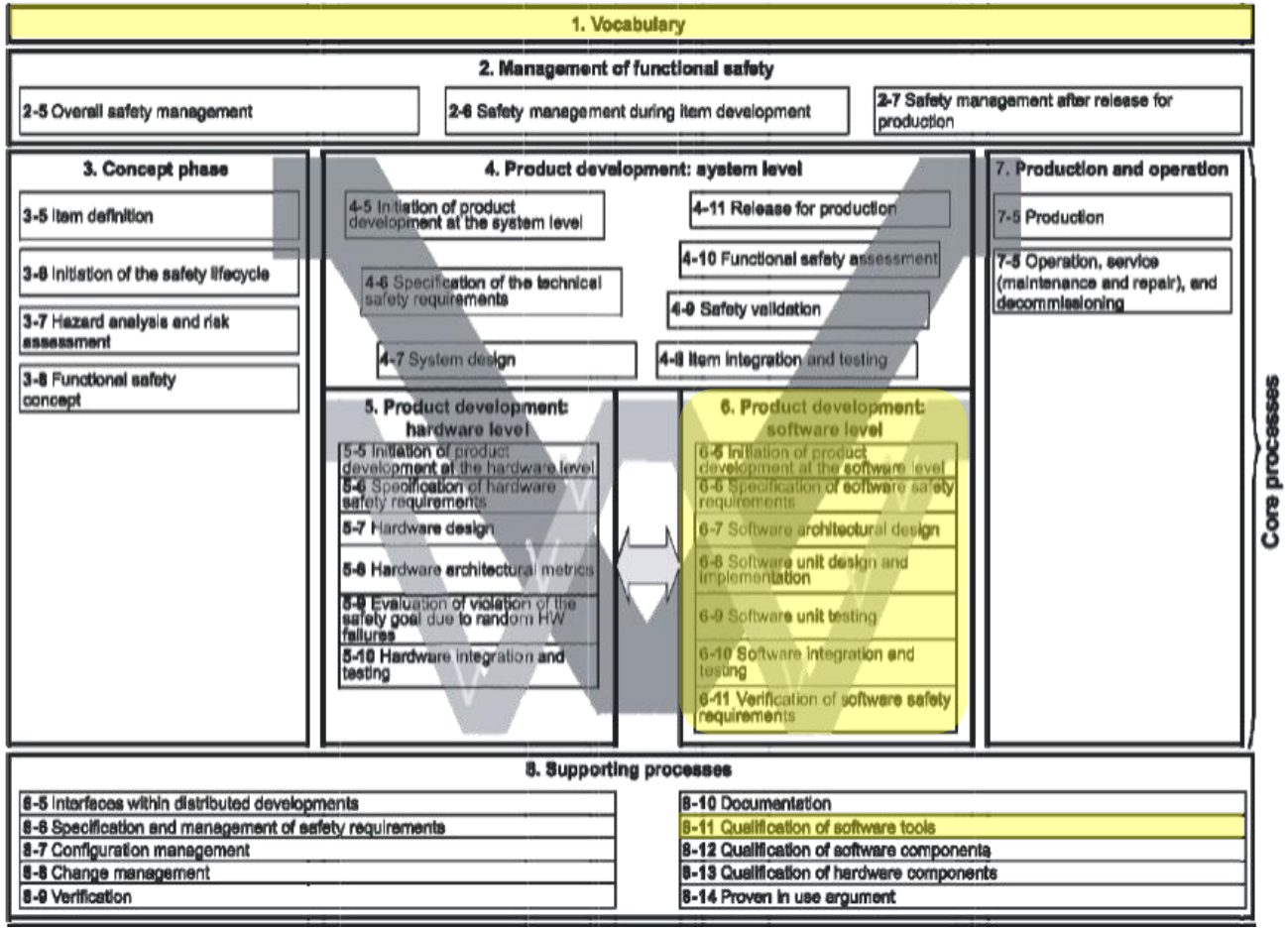
Based on the software architectural design, the detailed design of the software units is developed. The detailed design can be represented in the form of a model.

The implementation at the source code level can be manually or automatically generated from the design in accordance with the software development environment.

— Test cases generated from the same model which is also used for code generation cannot serve as the only source for verifying both the model itself and the code generated from it.

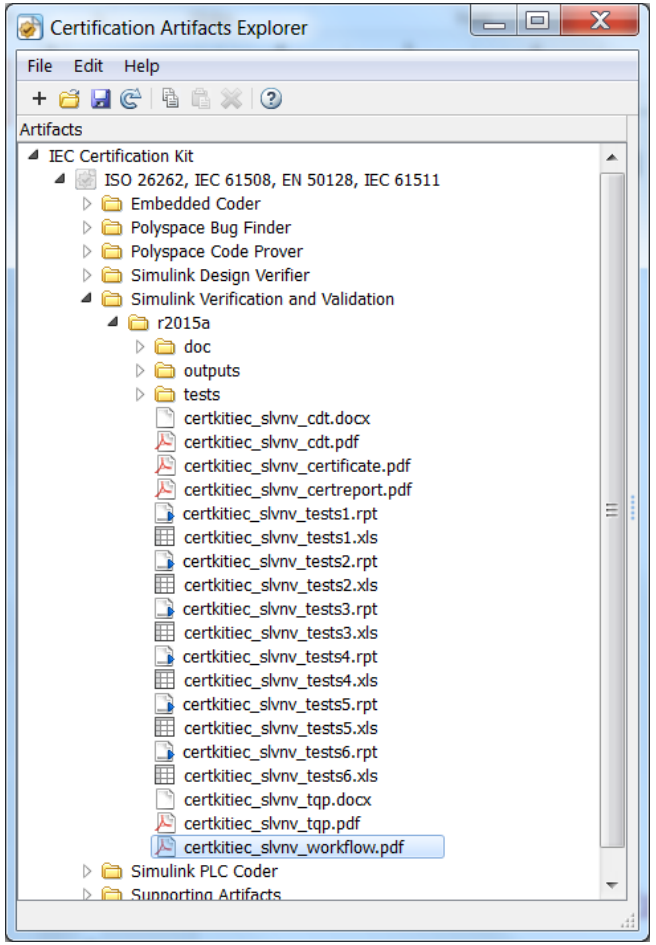
Tool Classification Workflow

ISO 26262 and Software Tool Qualification



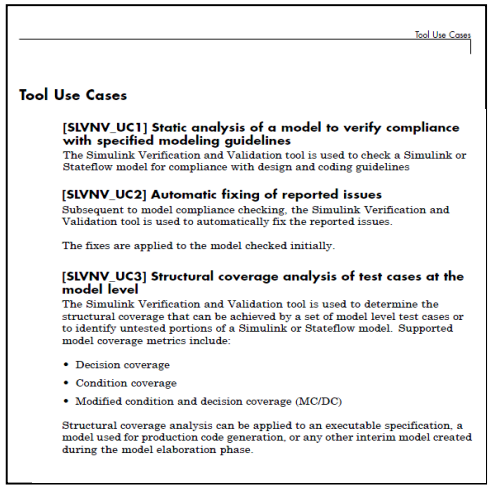
ISO 26262-1	Vocabulary
ISO 26262-2	Management of functional safety
ISO 26262-3	Concept phase
ISO 26262-4	Product development: system level
ISO 26262-5	Product development: hardware level
ISO 26262-6	Product development: software level
ISO 26262-7	Production and operation
ISO 26262-8 … ISO 26262-8-11 …	Supporting Processes … Qualification of software tools …
ISO 26262-9	ASIL-oriented and safety oriented analysis
ISO 26262-10	Guideline

IEC Certification Kit – Tool Qualification Support



Artifacts Explorer

Assessment



Tool Use Cases

Potential malfunction or erroneous output	Use case(s)	TI	Justification for TI	Prevention and detection measures	TD	Justification for TD	TCL
[SLVNV_E2] Model Compliance Checking – False Positive	[SLVNV_UC1]	TI1	Nuisance only; model does not violate modeling guidelines.	-	-	-	TCL1
[SLVNV_E3] Model Compliance Checking – Non Interference	[SLVNV_UC1]	TI1	Error in the tool; does not affect analysis results.	-	-	-	TCL1
[SLVNV_E4] Model Compliance Checking – Incorrect hyperlinks	[SLVNV_UC1]	TI1	Nuisance only; model does not violate modeling guidelines.	-	-	-	TCL1
[SLVNV_E5] Model Compliance Checking – Incorrect fixing of reported issues	[SLVNV_UC1]	TI2	Incorrect fixing could introduce error in the model.	[M2a] Subsequent re-checking of the model for compliance with specified modeling guidelines	TD2	Re-checking of the model will detect modeling standard violations introduced by the automatic fixing but might miss other errors introduced.	TCL2

Tool Classification

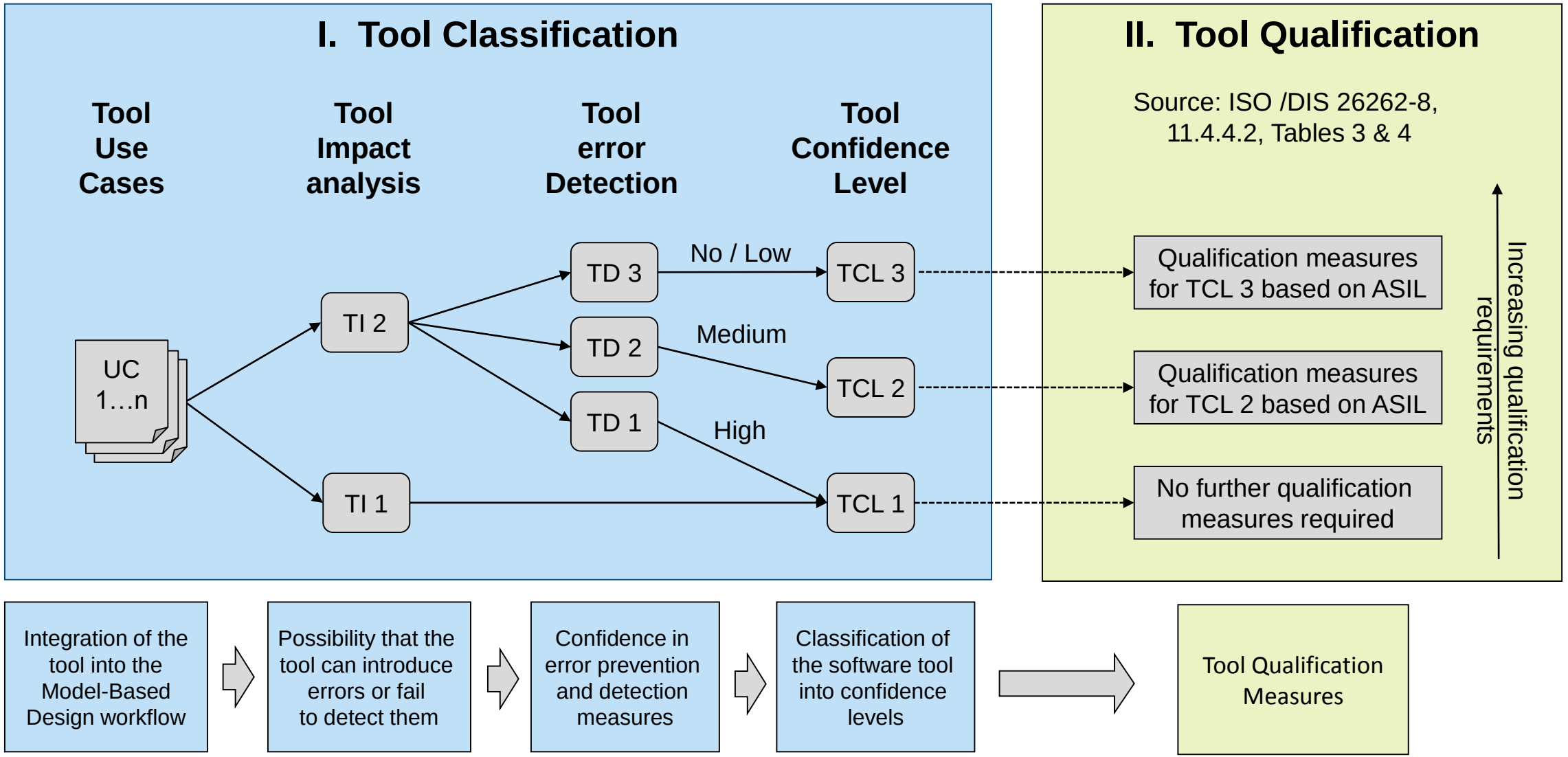
Checklist 1: Model Compliance Checking

	Technique / Measure	Associated Requirements	Used / Used to a limited degree / Not used	Interpretation in this application, Evidence
1	Adherence to modeling guidelines	• Designation of modeling guidelines • Review of modeling guidelines as suitable for use • Evidence for using the modeling guidelines		
2	Model compliance checking (Static analysis at the model level) (See "Tool Use Cases" in the *Simulink® Verification and Validation™ Reference Workflow*)	• Designation of model compliance checks in Model Advisor • Static analysis of model to verify compliance with specified modeling guidelines using Model Advisor • Generation of Model Advisor report to document results of model compliance checking • Review of Model Advisor report for detected guideline violations and errors • Corrective action on guideline violations and errors		
3	Preceding or subsequent dynamic verification (testing) of the model (See "Error Prevention and Detection Measures" in the *Simulink Verification and Validation Reference Workflow*)	• Execution of specified test cases against model • Documentation of the results of model tests • Corrective action on failure of model tests		

Workflow Conformance

www.mathworks.com/products/iec-61508/

ISO 26262 Tool Qualification Approach



Tool Qualification Methods/Measures based on ASIL

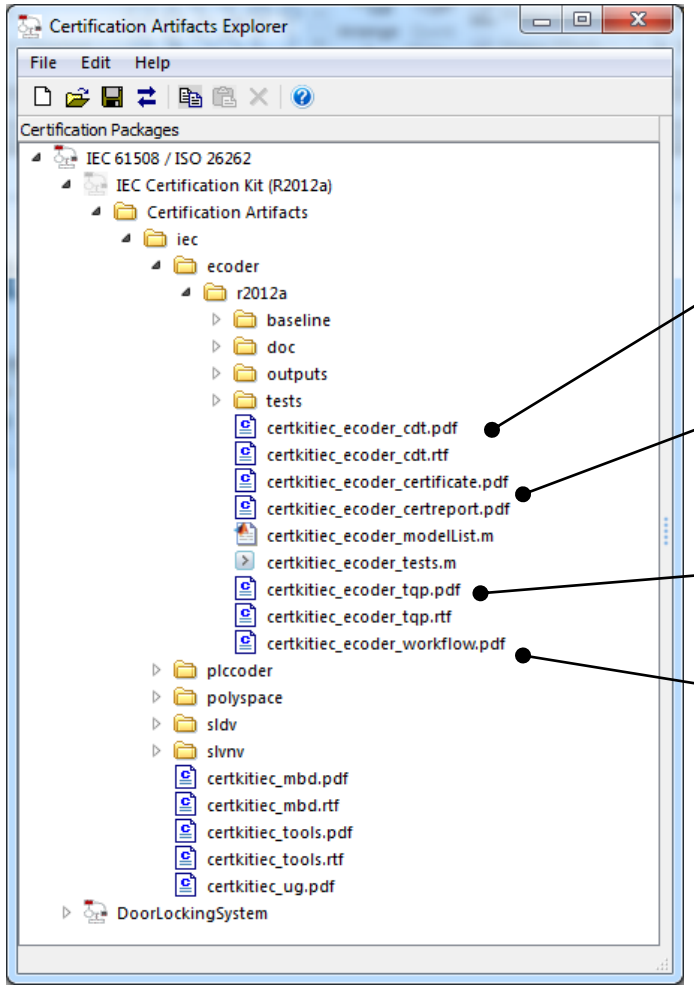
- For tools classified at **TCL 2** or **TCL 3**, at least one of the following **qualification methods** in the table shall be applied and documented
  - a) Increased confidence from use *(user responsible)*
  - **b) Evaluation of the tool development process** *(contained in Toolkit assessment report)*
  - **c) Validation of the software tool** *(contained in Toolkit validation test suites)*
  - d) Development in compliance with a safety standard *(tools not developed using a certified process)*
- **Tools classified at TCL 1 need no qualification measures**

Method	TCL 2				TCL 3			
	ASIL A	ASIL B	ASIL C	ASIL D	ASIL A	ASIL B	ASIL C	ASIL D
1a Increased confidence from use	++	++	++	+	++	++	+	+
1b Evaluation of the tool development process	++	++	++	+	++	++	+	+
1c Validation of the software tool	+	+	+	++	+	+	++	++
1d Development in compliance with a safety standard	+	+	+	++	+	+	++	++

Source: ISO /DIS 26262-8, 11.4.4.2, Tables 3&4 + … Recommended ++ … Highly recommended

Note: Embedded Coder, Simulink Verification and Validation, Simulink Design Verifier, and Polyspace products for C/C++ were not developed using certified processes

Certification Kit Artifacts for Tool Qualification



- Conformance demonstration template
- Evidence for independent assessment
  - Assessment report with TCL
  - Certificate
- Templates for tool qualification work products
- Reference workflow with respect to tool

IEC Certification Kit product provided pre-qualification results for users to re-use and adapt to their ISO 26262 projects

Qualification of MathWorks Tools: Workshare

I. Pre-qualification

A. Generic tool classification [MathWorks]

B. Generic pre-qualification [MathWorks]

C. Independent assessment [Cert. authority]

ISO 26262 tool qualification package (cert kit)

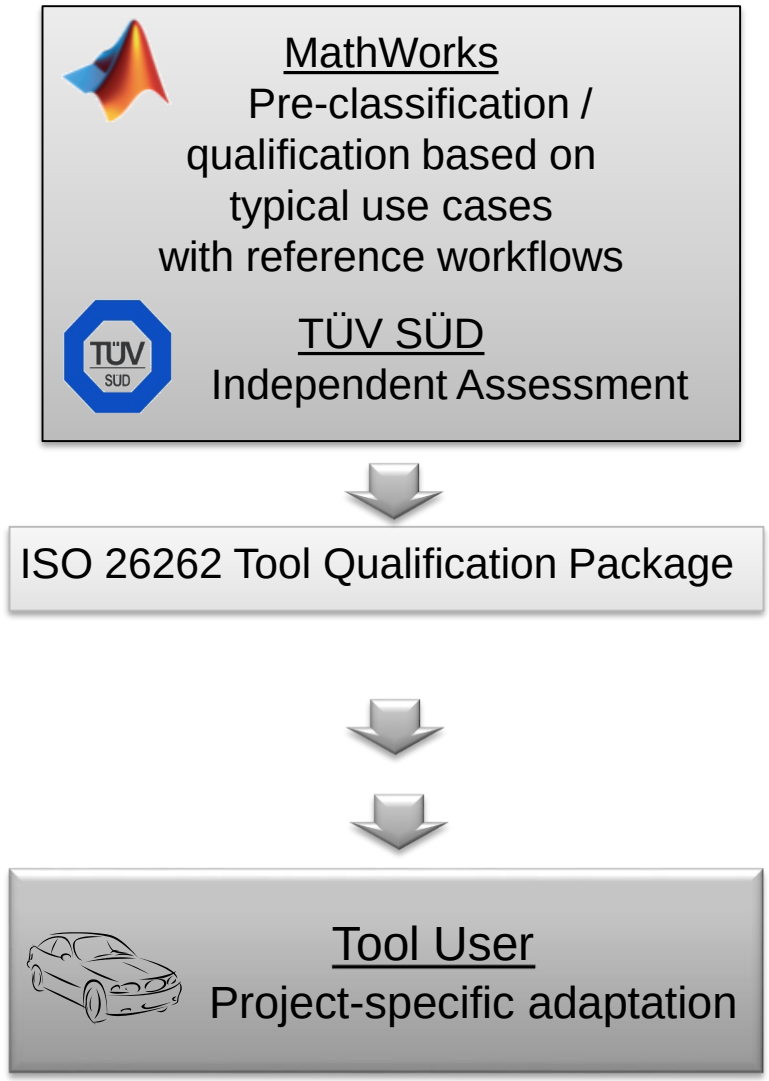
- Generic work products (pre-filled templates)
- Assessment results (assessment report, certificate)

II. Application-specific adaptation

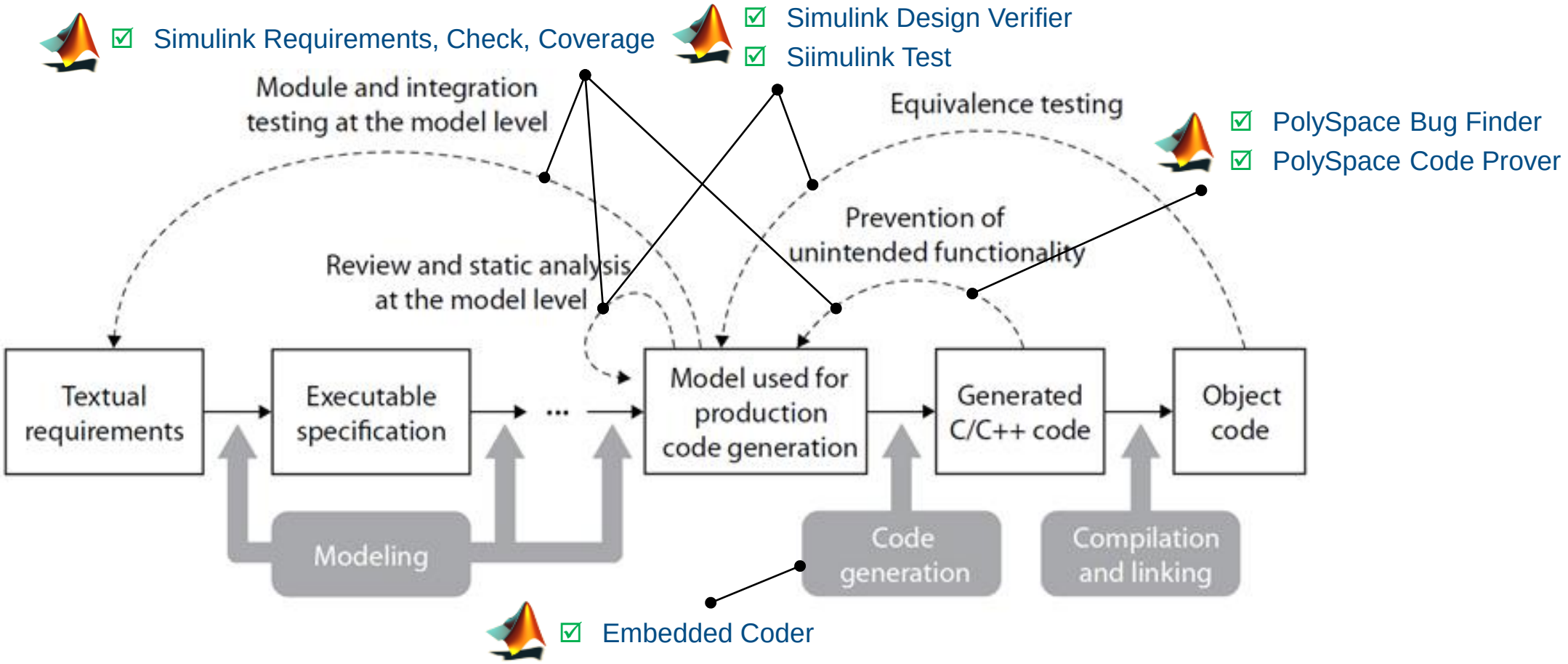
A. Review / adaptation of the tool qualification kit [Tool user]

ISO 26262 tool qualification work products

- Final work products (completed templates)
- Assessment results (assessment report, certificate)



Pre-Qualification of Model-Based Design Tools & Code Based Tools



☑ *pre-qualified for all ASILs according to ISO 26262*

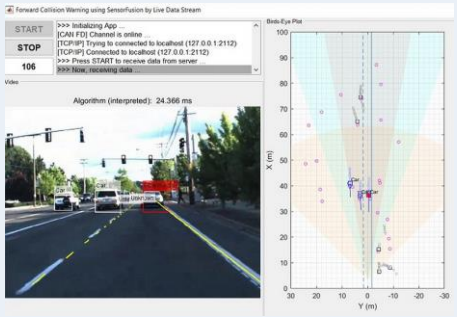
IEC Certification Kit Benefits

Without IEC Certification Kit (for all tools used)	With IEC Certification Kit (for all tools covered by the kit)
1. User need to provide documentation and installation instructions	1. MathWorks provides required documentation and installation instructions
2. User needs to document tools used in the development processes	2. MathWorks provides reference workflow describing typical development process and conformance demonstration template (CDT); user only needs to describe and justify deviations from reference workflow and document workflow compliance by completing CDT
3. User needs to carry out tool classification and determine TCL	3. MathWorks provides generic pre-classification and independent assessment; user only needs to verify assumptions in pre-classification
4. User needs to document tool classification	4. MathWorks provides pre-filled template for tool classification work product; user only needs to complete the template
5. (For tools with TCL 2 or higher) User needs to perform tool qualification measures (creation of a test suite)	5. MathWorks provides evidence for pre-qualification and independent assessment
6. (For tools with TCL 2 or higher) User needs to document tool qualification	6. MathWorks provides pre-filled template for tool qualification work product; user only needs to complete the template

Where you can use MATLAB and Simulink to develop Automated Driving / ADAS algorithms

Deep learning

Sensor fusion with live data

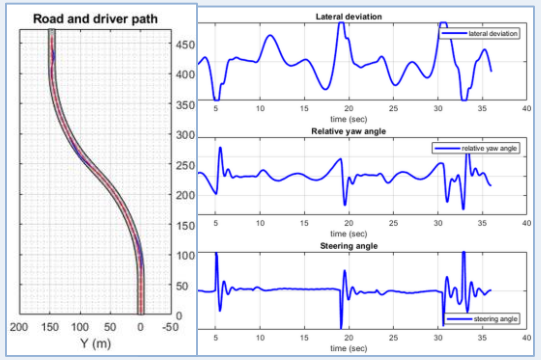


Perception

Control

Planning

Sensor models & model predictive control



Path planning