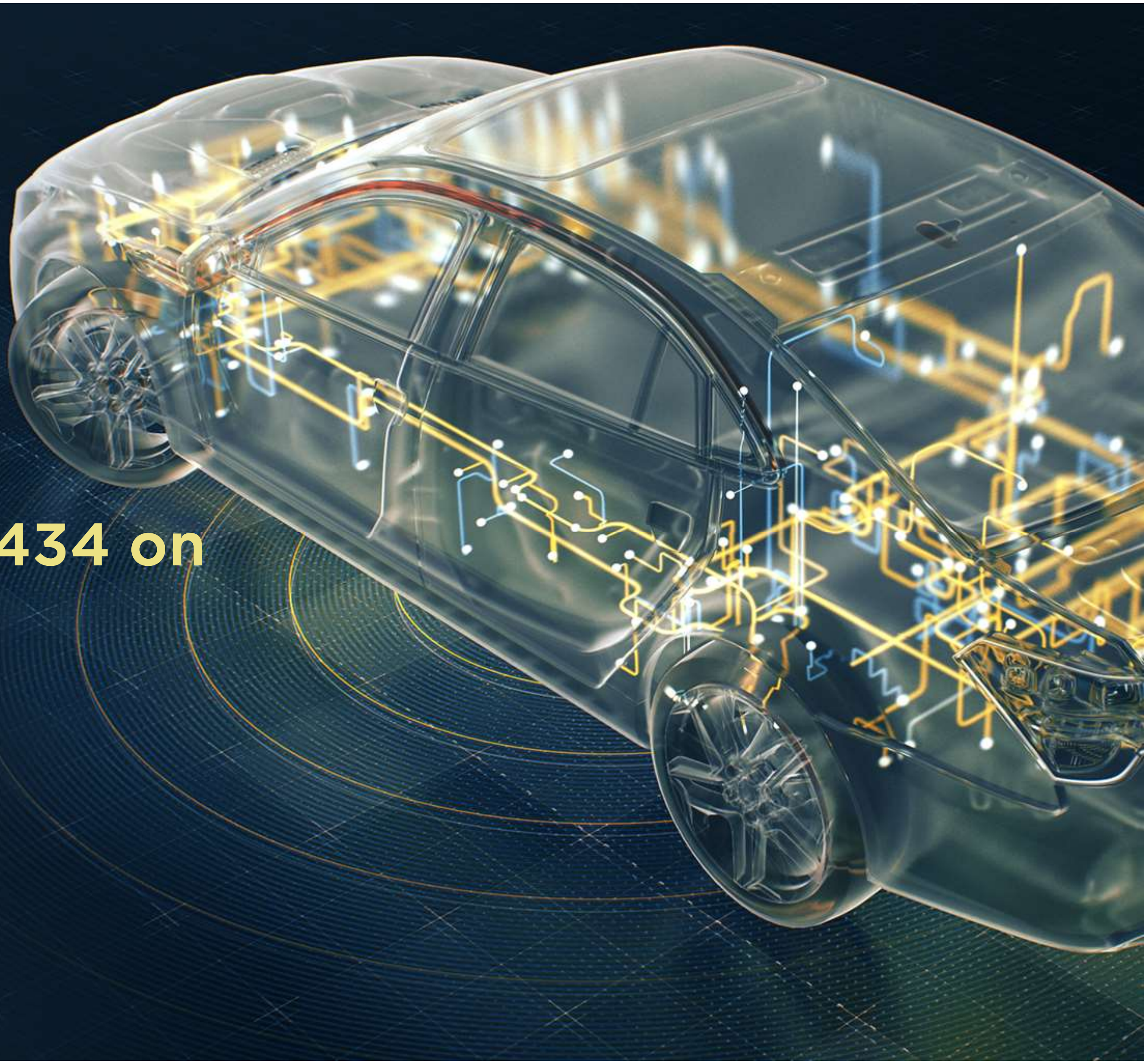
TASKING®



Impact of ISO26262/21434 on Tools and Software for Automotive

Joachim Hampp

Product Architect

September 2022

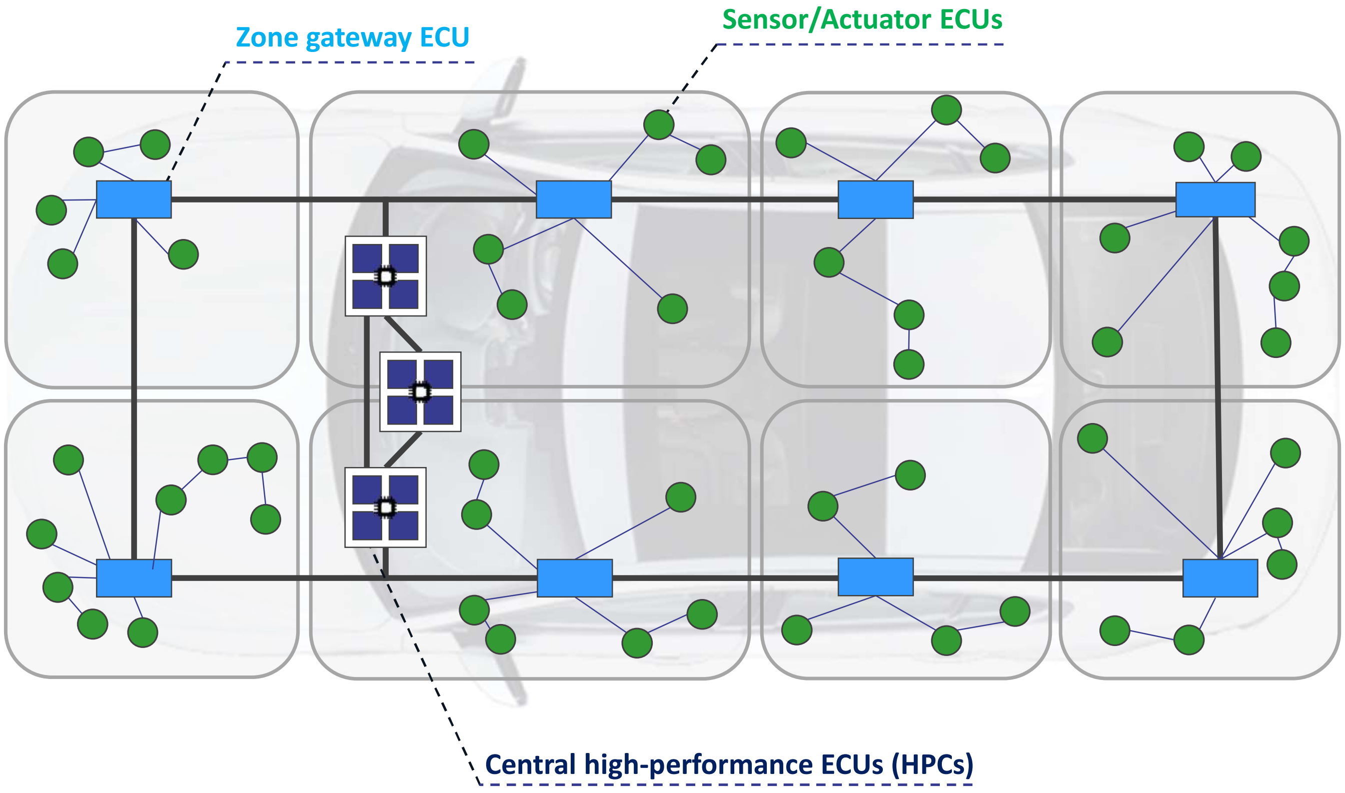
Agenda

1 **Parallel Processing Unit in Automotive Systems**

2 ISO 26262 Tool Qualification

3 ISO 21434 Tool Qualification

The new automotive architecture



HPC:

- Central Element of Server-based Vehicle Architecture
- Enable for connected and automated driving
- **ASIL-B** (separate ASIL-D controller)
- **Cyber Security**

Zone Controller:

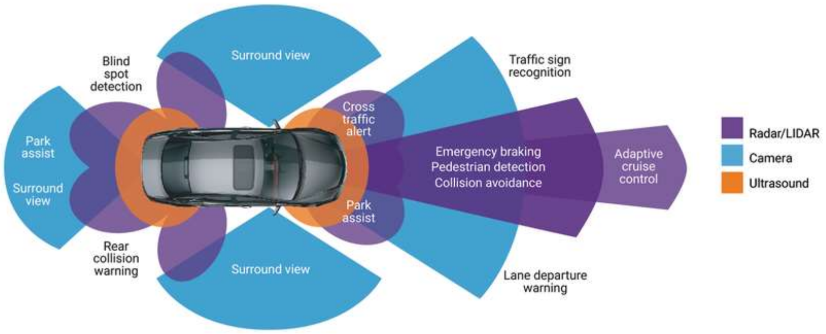
- Separate I/O from compute
- Gateway and com. backbone
- High Speed Ethernet for centralized data streams
- **ASIL-B to ASIL-D**
- **Cyber Security**

Sensor / Actuator (I/O):

- Radar, Lidar, Camera Devices
- Electric- steering, breaking
- Body and Chassis Controller, xEV Controller
- **ASIL-D and ASIL-B**
- **Cyber Security**

Parallel Processing Unit (ARC EV 71) Use Case 1 of 2

- The Parallel Processing Unit (PPU / ARC EV71) is used as an **accelerator** for Host Cores to offload it for dedicated use cases.
  - **Linear Algebra** are a perfect use case to execute on the PPU (e.g. main use for Radar devices)
  - **Filter algorithms** to execute on PPU to execute in parallel to the Host Cores (e.g. Kalman Filter)
  - Any **Matrix operations** / **mathematics** are accelerable by the PPU

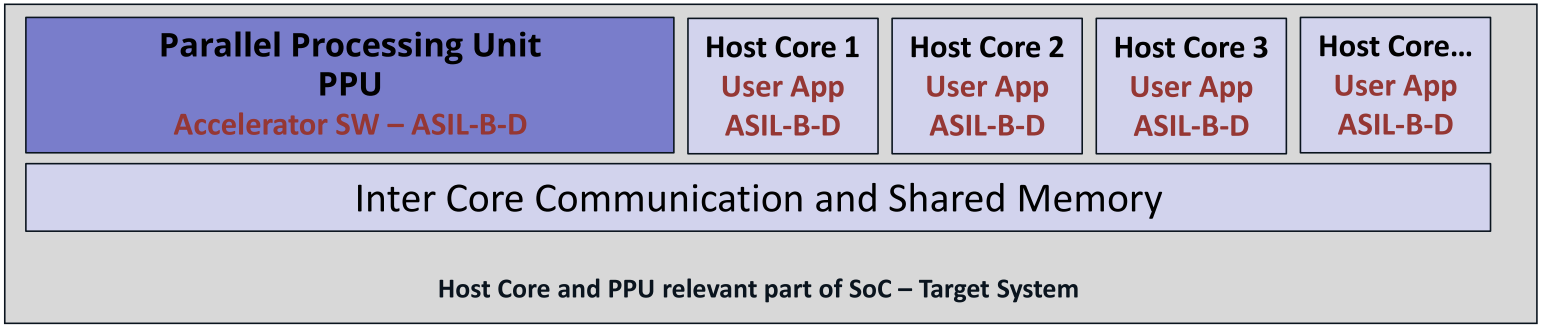


Parallel Processing Unit (ARC EV 71) Use Case 2 of 2

- The Host Core’s and the PPU must interact, communicate and exchange data
  - The input- and result- data requires a defined exchange mechanism
  - As there are more Host Cores than PPUs, a priority scheduling is required
  - Preempting a low priority job to enable a higher one

- The PPU executes code behalf of the Host Core. Safety/Security applicable for Host Core App. applies also to PPU

- What needs to be qualified against ISO 26262 and ISO 21434



Parallel Processing Unit
PPU
Accelerator SW – ASIL-B-D

Host Core 1
User App
ASIL-B-D

Host Core 2
User App
ASIL-B-D

Host Core 3
User App
ASIL-B-D

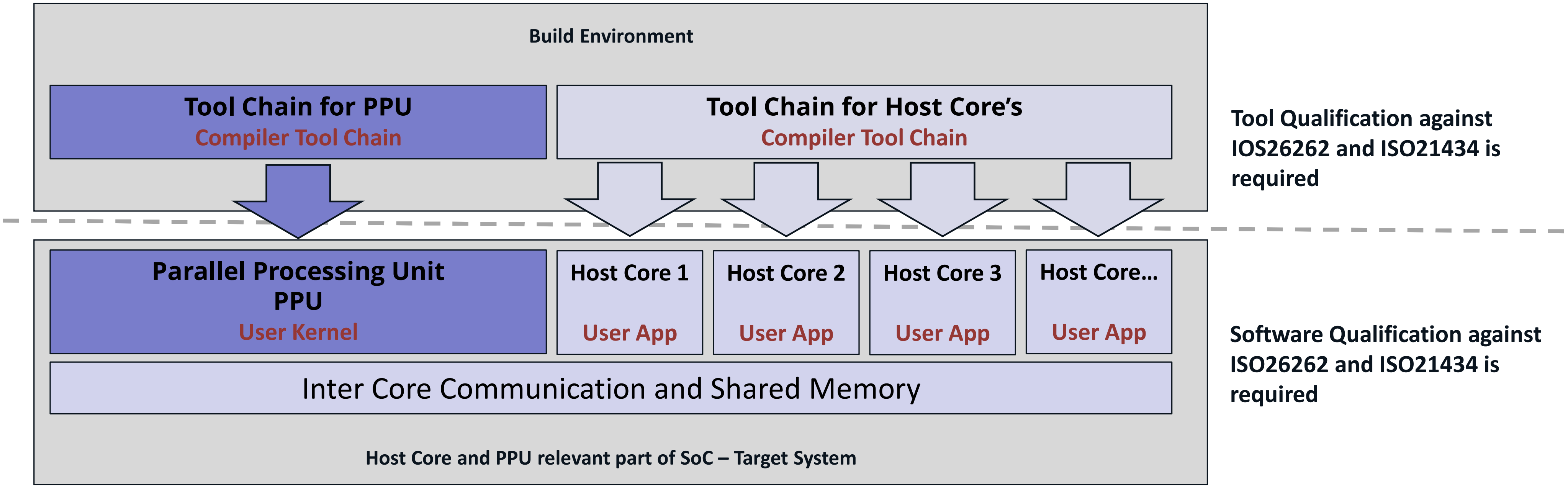
Host Core…
User App
ASIL-B-D

Inter Core Communication and Shared Memory

Host Core and PPU relevant part of SoC – Target System

Build environment overview

- A build environment consists of several tools to convert source code into an executable to operate on a SoC
  - A Compiler Tool Chain is the main part of the build environment
  - The source code is the main part that executes on the target system
  - Both parts require a qualification against ISO 26262 and ISO 21434.



Tool Qualification against IOS26262 and ISO21434 is required

Software Qualification against ISO26262 and ISO21434 is required

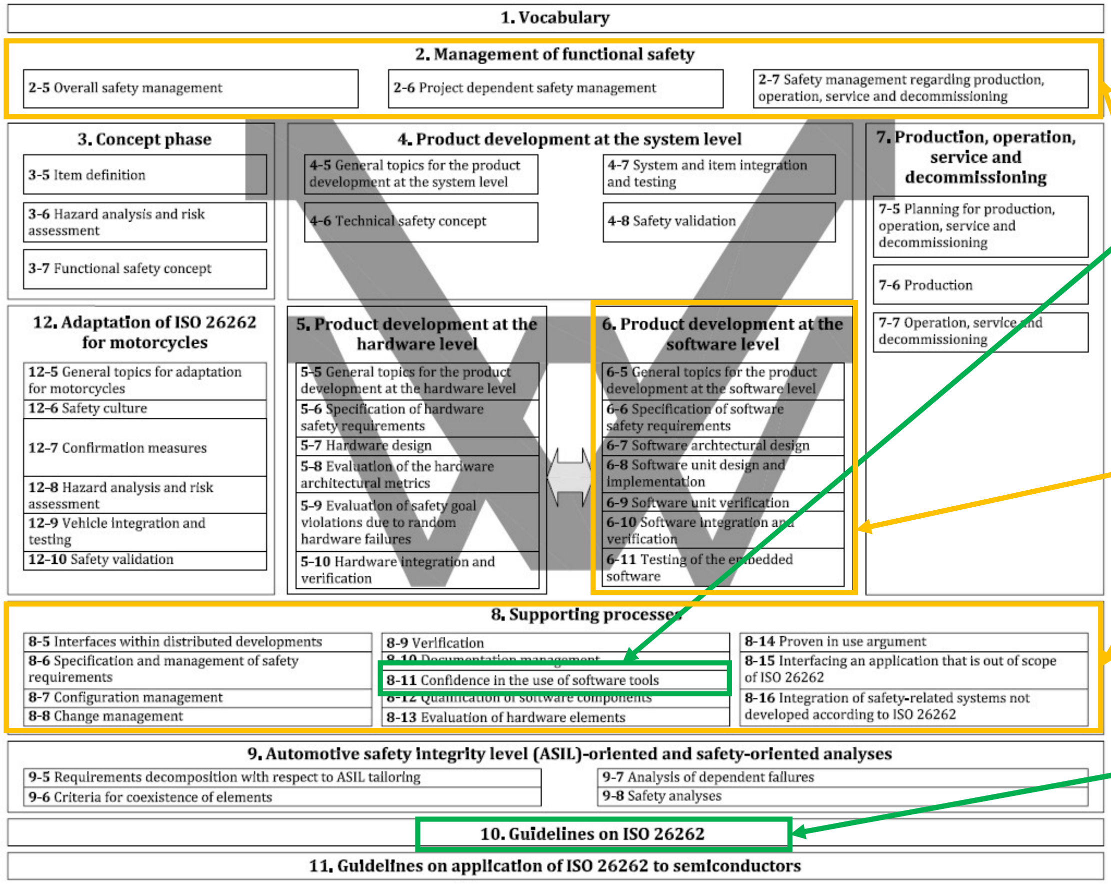
Agenda

1 Parallel Processing Unit ARC EV71 in Automotive

2 **ISO 26262 Tool Qualification**

3 ISO 21434 Tool Qualification

What Safety requirements apply for build tools



1. Vocabulary

2. Management of functional safety
- 2-5 Overall safety management
- 2-6 Project dependent safety management
- 2-7 Safety management regarding production, operation, service and decommissioning

3. Concept phase
- 3-5 Item definition
- 3-6 Hazard analysis and risk assessment
- 3-7 Functional safety concept

4. Product development at the system level
- 4-5 General topics for the product development at the system level
- 4-6 Technical safety concept
- 4-7 System and item integration and testing
- 4-8 Safety validation

7. Production, operation, service and decommissioning
- 7-5 Planning for production, operation, service and decommissioning
- 7-6 Production
- 7-7 Operation, service and decommissioning

12. Adaptation of ISO 26262 for motorcycles
- 12-5 General topics for adaptation for motorcycles
- 12-6 Safety culture
- 12-7 Confirmation measures
- 12-8 Hazard analysis and risk assessment
- 12-9 Vehicle integration and testing
- 12-10 Safety validation

5. Product development at the hardware level
- 5-5 General topics for the product development at the hardware level
- 5-6 Specification of hardware safety requirements
- 5-7 Hardware design
- 5-8 Evaluation of the hardware architectural metrics
- 5-9 Evaluation of safety goal violations due to random hardware failures
- 5-10 Hardware integration and verification

6. Product development at the software level
- 6-5 General topics for the product development at the software level
- 6-6 Specification of software safety requirements
- 6-7 Software architectural design
- 6-8 Software unit design and implementation
- 6-9 Software unit verification
- 6-10 Software integration and verification
- 6-11 Testing of the embedded software

8. Supporting processes
- 8-5 Interfaces within distributed developments
- 8-6 Specification and management of safety requirements
- 8-7 Configuration management
- 8-8 Change management
- 8-9 Verification
- 8-10 Documentation management
- 8-11 Confidence in the use of software tools
- 8-12 Qualification of software components
- 8-13 Evaluation of hardware elements
- 8-14 Proven in use argument
- 8-15 Interfacing an application that is out of scope of ISO 26262
- 8-16 Integration of safety-related systems not developed according to ISO 26262

9. Automotive safety integrity level (ASIL)-oriented and safety-oriented analyses
- 9-5 Requirements decomposition with respect to ASIL tailoring
- 9-6 Criteria for coexistence of elements
- 9-7 Analysis of dependent failures
- 9-8 Safety analyses

10. Guidelines on ISO 26262

11. Guidelines on application of ISO 26262 to semiconductors

Main chapter related to tools used in the build process

Relevant for safety critical target software development

Supporting guidlines for build tools and runtime software qualification

Confidence in the use of SW Tools

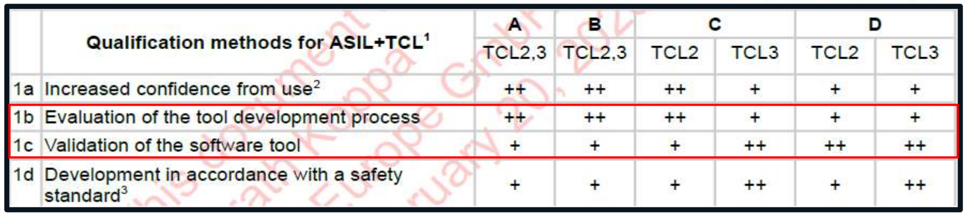
TASKING

Supplier delivered Information

- Toolset user manuals
- Pre-determined max ASIL level
- Toolset constraints for safety related development
- Known malfunctions and mitigations

Customer Supplied Information

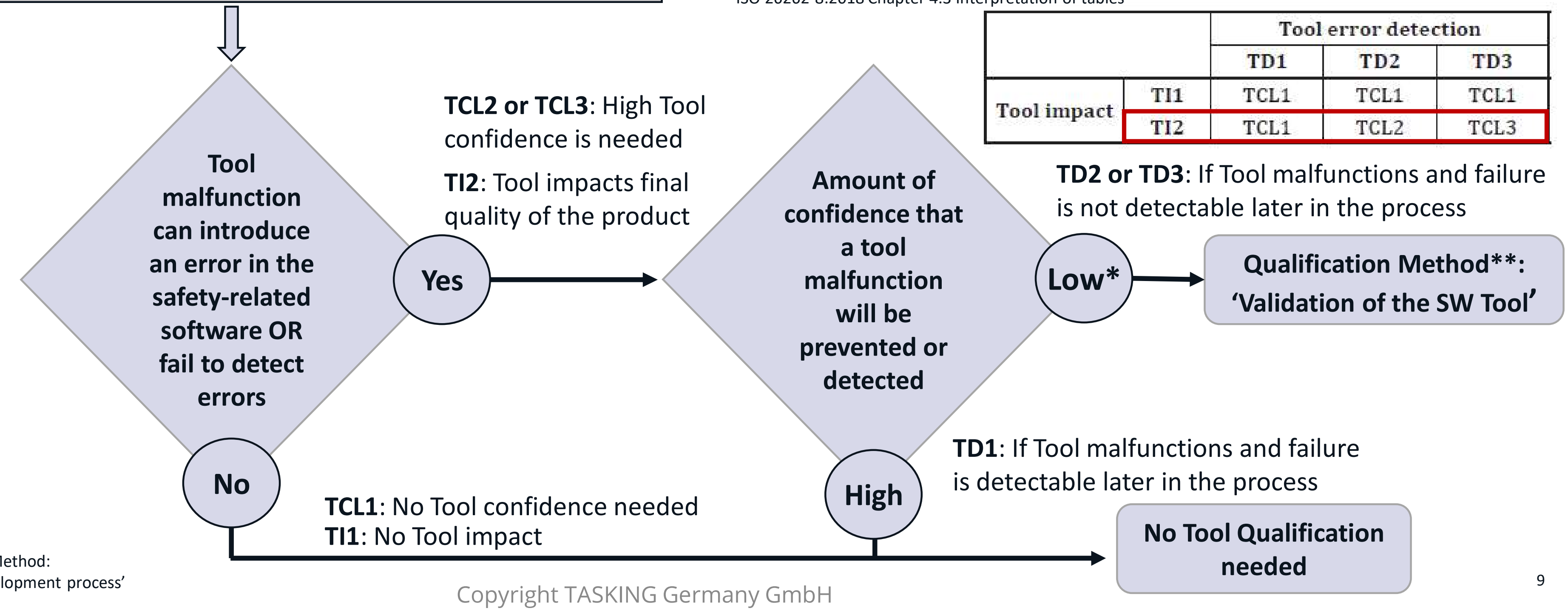
- Safety Plan
- Configuration of SW Tool
- Use Cases of SW Tool
- Tool environment
- Max ASIL of safety requirements



	Qualification methods for ASIL+TCL[1]	A	B	C		D	
		TCL2,3	TCL2,3	TCL2	TCL3	TCL2	TCL3
1a	Increased confidence from use[2]	++	++	++	+	+	+
1b	Evaluation of the tool development process	++	++	++	+	+	+
1c	Validation of the software tool	+	+	+	++	++	++
1d	Development in accordance with a safety standard[3]	+	+	+	++	+	++

"++" Indicates that the method is highly recommended for the identified ASIL
"+" Indicates that the method is recommended for the identified ASIL
ISO 26262-8:2018 Chapter 4.3 Interpretation of tables

		Tool error detection		
		TD1	TD2	TD3
Tool impact	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3



Tool malfunction can introduce an error in the safety-related software OR fail to detect errors

Yes

TCL2 or TCL3: High Tool confidence is needed

TI2: Tool impacts final quality of the product

Amount of confidence that a tool malfunction will be prevented or detected

Low*

TD2 or TD3: If Tool malfunctions and failure is not detectable later in the process

Qualification Method**: 'Validation of the SW Tool'

No

TCL1: No Tool confidence needed
TI1: No Tool impact

High

TD1: If Tool malfunctions and failure is detectable later in the process

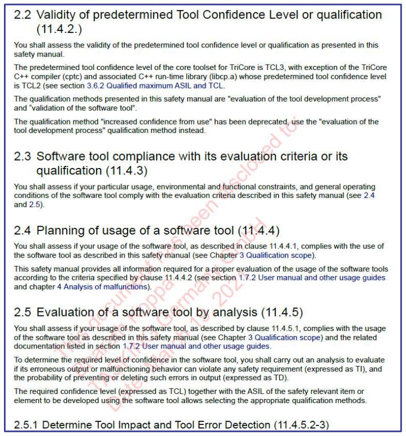
No Tool Qualification needed

*Low or Medium
**ASIL A, B, C Qualification Method: 'Evaluation of the tool development process'

Tool Qualification Safety Manual shall contain

- **Exact Product Version / Identification**
- Evidence of ISO compliance. **Validation test suite result‘s.** Test cases shall be provided upon customer request.
- Details of **tool options qualified** for ASIL development. Describing the safe use of the product.
- **Tool application guidelines for customer for ASIL qualification**
- **A document for safety manager to verify ISO 26262 requirements versus tool vendor qualification methods.**
- Description of the errata management
  - **Up to date and detailed issue listing**
  - Qualification of issues. E.g., Low, Mid or High impact
  - Description how to mitigate the issues
- Conformance to the defined/applied **software development process** (e.g., ASPICE)
- The Safety Manual can contain a security manual that addresses **Cybersecurity conformation.**

Highlighted in Green: Data to be provided by the Safety Manual vendor



2.2 Validity of predetermined Tool Confidence Level or qualification (11.4.2.)

You shall assess the validity of the predetermined tool confidence level or qualification as presented in this safety manual.

The predetermined tool confidence level of the core toolset for TriCore is TCL3, with exception of the TriCore C++ compiler (cptc) and associated C++ run-time library (libcp.a) whose predetermined tool confidence level is TCL2 (see section 3.6.2 Qualified maximum ASIL and TCL.

The qualification methods presented in this safety manual are "evaluation of the tool development process" and “validation of the software tool”.

The qualification method "increased confidence from use" has been deprecated, use the "evaluation of the tool development process" qualification method instead.

2.3 Software tool compliance with its evaluation criteria or its qualification (11.4.3)

You shall assess if your particular usage, environmental and functional constraints, and general operating conditions of the software tool comply with the evaluation criteria described in this safety manual (see 2.4 and 2.5).

2.4 Planning of usage of a software tool (11.4.4)

You shall assess if your usage of the software tool, as described in clause 11.4.4.1, complies with the use of the software tool as described in this safety manual (see Chapter 3 Qualification scope).

This safety manual provides all information required for a proper evaluation of the usage of the software tools according to the criteria specified by clause 11.4.4.2 (see section 1.7.2 User manual and other usage guides and chapter 4 Analysis of malfunctions).

2.5 Evaluation of a software tool by analysis (11.4.5)

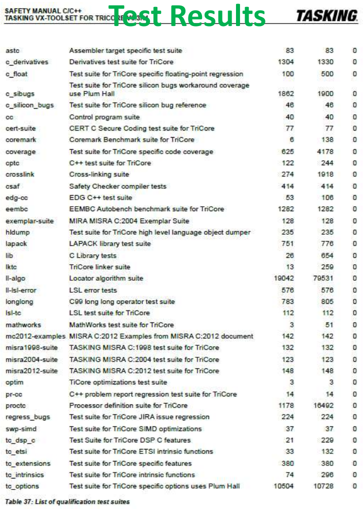
You shall assess if your usage of the software tool, as described by clause 11.4.5.1, complies with the usage of the software tool as described in this safety manual (see Chapter 3 Qualification scope) and the related documentation listed in section 1.7.2 User manual and other usage guides.

To determine the required level of confidence in the software tool, you shall carry out an analysis to evaluate if its erroneous output or malfunctioning behavior can violate any safety requirement (expressed as TI), and the probability of preventing or deleting such errors in output (expressed as TD).

The required confidence level (expressed as TCL) together with the ASIL of the safety relevant item or element to be developed using the software tool allows selecting the appropriate qualification methods.

2.5.1 Determine Tool Impact and Tool Error Detection (11.4.5.2-3)

Requirement mapping

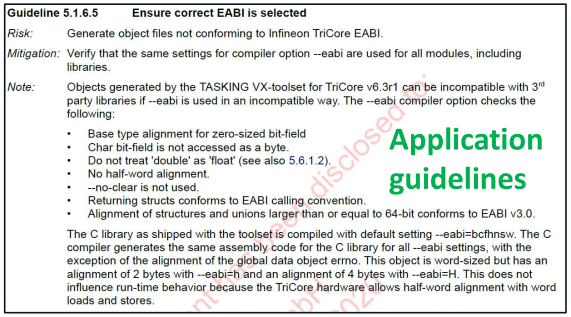


SAFETY MANUAL C/C++
TASKING VX-TOOLSET FOR TRICORE V6.3R1 — TASKING

Test Results

astc	Assembler target specific test suite	83	83	0
c_derivatives	Derivatives test suite for TriCore	1304	1330	0
c_float	Test suite for TriCore specific floating-point regression	100	500	0
c_sibugs	Test suite for TriCore silicon bugs workaround coverage use Plum Hall	1862	1900	0
c_silicon_bugs	Test suite for TriCore silicon bug reference	46	46	0
cc	Control program suite	40	40	0
cert-suite	CERT C Secure Coding test suite for TriCore	77	77	0
coremark	Coremark Benchmark suite for TriCore	6	138	0
coverage	Test suite for TriCore specific code coverage	625	4178	0
cptc	C++ test suite for TriCore	122	244	0
crosslink	Cross-linking suite	274	1918	0
csaf	Safety Checker compiler tests	414	414	0
edg-cc	EDG C++ test suite	53	106	0
eembc	EEMBC Autobench benchmark suite for TriCore	1282	1282	0
exemplar-suite	MIRA MISRA C:2004 Exemplar Suite	128	128	0
hldump	Test suite for TriCore high level language object dumper	235	235	0
lapack	LAPACK library test suite	751	776	0
lib	C Library tests	26	654	0
lktc	TriCore linker suite	13	259	0
ll-algo	Locator algorithm suite	19042	79531	0
ll-lsl-error	LSL error tests	576	576	0
longlong	C99 long long operator test suite	783	805	0
lsl-tc	LSL test suite for TriCore	112	112	0
mathworks	MathWorks test suite for TriCore	3	51	0
mc2012-examples	MISRA C:2012 Examples from MISRA C:2012 document	142	142	0
misra1998-suite	TASKING MISRA C:1998 test suite for TriCore	132	132	0
misra2004-suite	TASKING MISRA C:2004 test suite for TriCore	123	123	0
misra2012-suite	TASKING MISRA C:2012 test suite for TriCore	148	148	0
optim	TiCore optimizations test suite	3	3	0
pr-cc	C++ problem report regression test suite for TriCore	14	14	0
proctc	Processor definition suite for TriCore	1178	16492	0
regress_bugs	Test suite for TriCore JIRA issue regression	224	224	0
swp-simd	Test suite for TriCore SIMD optimizations	37	37	0
tc_dsp_c	Test Suite for TriCore DSP C features	21	229	0
tc_etsi	Test suite for TriCore ETSI intrinsic functions	33	132	0
tc_extensions	Test suite for TriCore specific features	380	380	0
tc_intrinsics	Test suite for TriCore intrinsic functions	74	296	0
tc_options	Test suite for TriCore specific options uses Plum Hall	10504	10728	0

Table 37: List of qualification test suites



Guideline 5.1.6.5 Ensure correct EABI is selected

Risk: Generate object files not conforming to Infineon TriCore EABI.

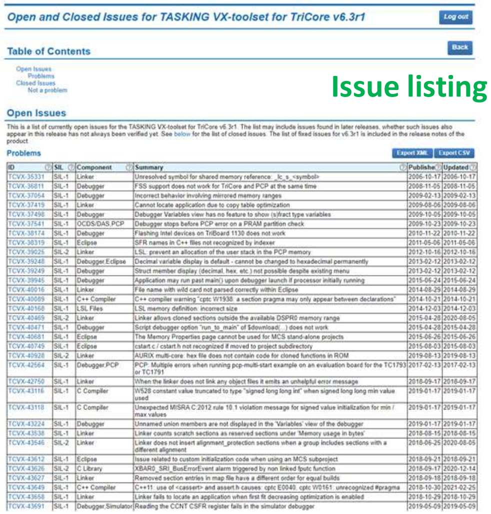
Mitigation: Verify that the same settings for compiler option --eabi are used for all modules, including libraries.

Note: Objects generated by the TASKING VX-toolset for TriCore v6.3r1 can be incompatible with 3rd party libraries if --eabi is used in an incompatible way. The --eabi compiler option checks the following:

- Base type alignment for zero-sized bit-field
- Char bit-field is not accessed as a byte.
- Do not treat 'double' as 'float' (see also 5.6.1.2).
- No half-word alignment.
- --no-clear is not used.
- Returning structs conforms to EABI calling convention.
- Alignment of structures and unions larger than or equal to 64-bit conforms to EABI v3.0.

The C library as shipped with the toolset is compiled with default setting --eabi=bcfhnsw. The C compiler generates the same assembly code for the C library for all --eabi settings, with the exception of the alignment of the global data object errno. This object is word-sized but has an alignment of 2 bytes with --eabi=h and an alignment of 4 bytes with --eabi=H. This does not influence run-time behavior because the TriCore hardware allows half-word alignment with word loads and stores.

Application guidelines



Open and Closed Issues for TASKING VX-toolset for TriCore v6.3r1

Table of Contents

Open Issues
Problems
Closed Issues
Not a problem

Issue listing

Open Issues

This is a list of currently open issues for the TASKING VX-toolset for TriCore v6.3r1. The list may include issues found in later releases, whether such issues also appear in this release has not always been verified yet. See below for the list of closed issues. The list of fixed issues for v6.3r1 is included in the release notes of the product.

Problems

ID	SIL	Component	Summary	Publishe	Updated
TCVX-35331	SIL-1	Linker	Unresolved symbol for shared memory reference: _lc_s_<symbol>	2006-10-17	2006-10-17
TCVX-36811	SIL-1	Debugger	FSS support does not work for TriCore and PCP at the same time	2008-11-05	2008-11-05
TCVX-37054	SIL-1	Debugger	Incorrect behavior involving mirrored memory ranges	2009-02-13	2009-02-13
TCVX-37419	SIL-1	Linker	Cannot locate application due to copy table optimization	2009-08-06	2009-08-06
TCVX-37498	SIL-1	Debugger	Debugger Variables view has no feature to show (s)fract type variables	2009-10-05	2009-10-05
TCVX-37541	SIL-1	OCDS/DAS,PCP	Debugger stops before PCP error on a PRAM partition check	2009-10-23	2009-10-23
TCVX-38174	SIL-1	Debugger	Flashing Intel devices on TriBoard 1130 does not work	2010-11-22	2010-11-22
TCVX-38319	SIL-1	Eclipse	SFR names in C++ files not recognized by indexer	2011-05-06	2011-05-06
TCVX-39025	SIL-2	Linker	LSL: prevent an allocation of the user stack in the PCP memory	2012-10-16	2012-10-16
TCVX-39248	SIL-1	Debugger,Eclipse	Decimal variable display is default - cannot be changed to hexadecimal permanently	2013-02-12	2013-02-12
TCVX-39249	SIL-1	Debugger	Struct member display (decimal, hex, etc.) not possible despite existing menu	2013-02-12	2013-02-12
TCVX-39945	SIL-1	Debugger	Application may run past main() upon debugger launch if processor initially running	2015-06-24	2015-06-24
TCVX-40016	SIL-1	Linker	File name with wild card not parsed correctly within Eclipse	2014-08-29	2014-08-29
TCVX-40089	SIL-1	C++ Compiler	C++ compiler warning "cptc W1938: a section pragma may only appear between declarations"	2014-10-21	2014-10-21
TCVX-40168	SIL-1	LSL Files	LSL memory definition: incorrect size	2014-12-03	2014-12-03
TCVX-40469	SIL-2	Linker	Linker allows cloned sections outside the available DSPR0 memory range	2015-04-28	2020-08-05
TCVX-40471	SIL-1	Debugger	Script debugger option "run_to_main" of $download(...) does not work	2015-04-28	2015-04-28
TCVX-40681	SIL-1	Eclipse	The Memory Properties page cannot be used for MCS stand-alone projects	2015-06-26	2015-06-26
TCVX-40749	SIL-1	Eclipse	cstart.c / cstart.h not recognized if moved to project subdirectory	2015-08-03	2015-08-03
TCVX-40928	SIL-2	Linker	AURIX multi-core: hex file does not contain code for cloned functions in ROM	2019-08-13	2019-08-13
TCVX-42564	SIL-1	Debugger,PCP	PCP: Multiple errors when running pcp-multi-start example on an evaluation board for the TC1793 or TC1791	2017-02-13	2017-02-13
TCVX-42750	SIL-1	Linker	When the linker does not link any object files it emits an unhelpful error message	2018-09-17	2018-09-17
TCVX-43116	SIL-1	C Compiler	W528 constant value truncated to type "signed long long int" when signed long long min value used	2019-01-17	2019-01-17
TCVX-43118	SIL-1	C Compiler	Unexpected MISRA C:2012 rule 10.1 violation message for signed value initialization for min / max values	2019-01-17	2019-01-17
TCVX-43224	SIL-1	Debugger	Unnamed union members are not displayed in the 'Variables' view of the debugger	2019-01-17	2019-01-17
TCVX-43538	SIL-1	Linker	Linker counts scratch sections as reserved sections under 'Memory usage in bytes'	2018-08-15	2018-08-15
TCVX-43546	SIL-2	Linker	Linker does not insert alignment_protection sections when a group includes sections with a different alignment	2018-06-25	2020-08-05
TCVX-43612	SIL-1	Eclipse	Issue related to custom initialization code when using an MCS subproject	2018-09-21	2018-09-21
TCVX-43626	SIL-2	C Library	XBAR0_SRI_BusErrorEvent alarm triggered by non linked fputc function	2018-09-17	2020-12-14
TCVX-43627	SIL-1	Linker	Removed section entries in map file have a different order for equal builds	2018-09-18	2018-09-18
TCVX-43649	SIL-1	C++ Compiler	C++11: use of <cassert> and assert.h causes: cptc E0040, cptc W0161: unrecognized #pragma	2018-10-30	2021-02-25
TCVX-43658	SIL-1	Linker	Linker fails to locate an application when first fit decreasing optimization is enabled	2018-10-29	2018-10-29
TCVX-43691	SIL-1	Debugger,Simulator	Reading the CCNT CSFR register fails in the simulator debugger	2019-05-09	2019-05-09

Agenda

1 Parallel Processing Unit ARC EV71 in Automotive

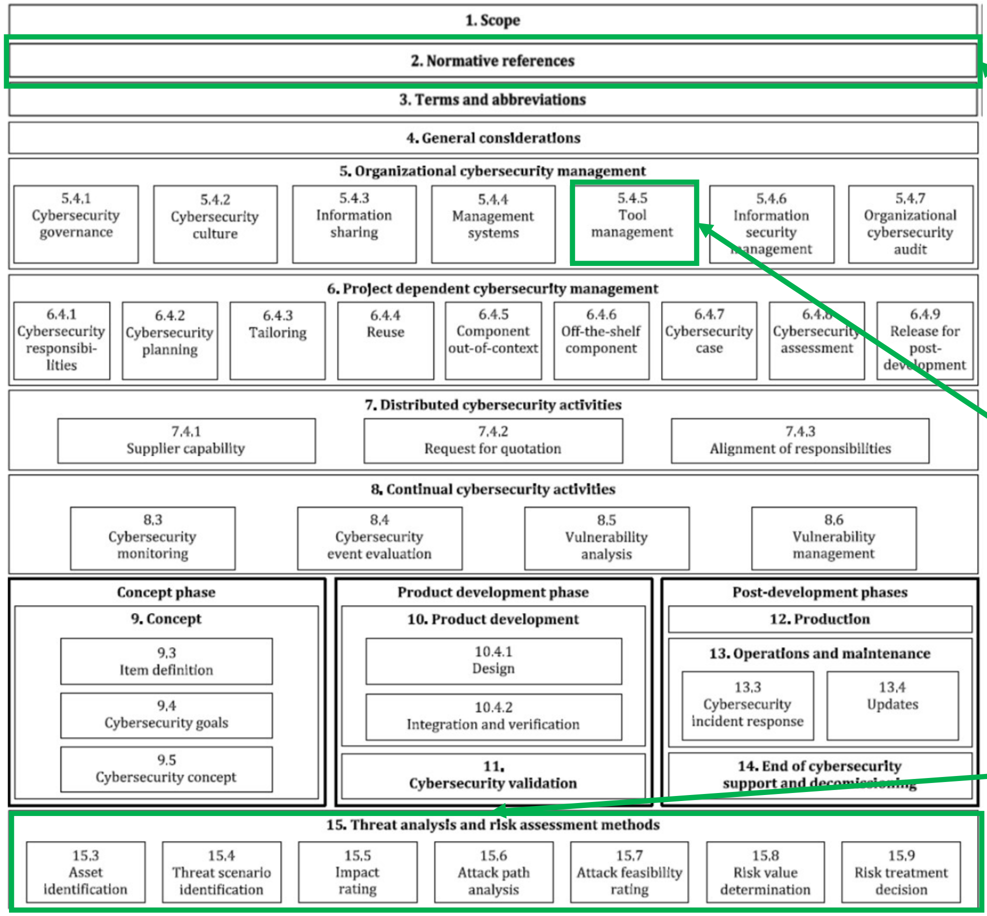
2 ISO 26262 Tool Qualification

3 **ISO 21434 Tool Qualification**

Importance & Urgence of Cybersecurity Compliance

- Determined by legislation regarding type approval
  - In 2020 the United Nations Economic Commission for Europe (UNECE) released a regulation(not legally binding) on uniform provisions concerning the **approval of vehicles** with regards **to cybersecurity and cybersecurity management system** (aka WP.29)
  - In the European Union, the new regulation on cybersecurity will become **mandatory** (turned into legislation) for all new vehicle types from **July 2022** and will become **mandatory** for all new vehicles produced from **July 2024**
  - Cybersecurity will be **nonnegotiable for securing market access** and type approval in the future (product liability is of secondary importance)
  - Regulation will affect the production of over 20 million cars yearly (excl. vans, trucks and busses)
- Japan & Korea are members of the UNECE and are implementing the “cybersecurity regulation” into legislation
- **The USA adopted the ISO/SAE 21434 Road vehicles** — Cybersecurity Engineering standard, which was developed in cooperation between the ISO and the US based Society of Automotive Engineers (SAE) it replaces the Cybersecurity Guidebook for Cyber-Physical Vehicle Systems that was released in 2016.

ISO 21434 Cybersecurity Engineering - Tool Management



1. Scope

2. Normative references

3. Terms and abbreviations

4. General considerations

5. Organizational cybersecurity management

- 5.4.1 Cybersecurity governance
- 5.4.2 Cybersecurity culture
- 5.4.3 Information sharing
- 5.4.4 Management systems
- 5.4.5 Tool management
- 5.4.6 Information security management
- 5.4.7 Organizational cybersecurity audit

6. Project dependent cybersecurity management

- 6.4.1 Cybersecurity responsibi-lities
- 6.4.2 Cybersecurity planning
- 6.4.3 Tailoring
- 6.4.4 Reuse
- 6.4.5 Component out-of-context
- 6.4.6 Off-the-shelf component
- 6.4.7 Cybersecurity case
- 6.4.8 Cybersecurity assessment
- 6.4.9 Release for post-development

7. Distributed cybersecurity activities

- 7.4.1 Supplier capability
- 7.4.2 Request for quotation
- 7.4.3 Alignment of responsibilities

8. Continual cybersecurity activities

- 8.3 Cybersecurity monitoring
- 8.4 Cybersecurity event evaluation
- 8.5 Vulnerability analysis
- 8.6 Vulnerability management

Concept phase

9. Concept

- 9.3 Item definition
- 9.4 Cybersecurity goals
- 9.5 Cybersecurity concept

Product development phase

10. Product development

- 10.4.1 Design
- 10.4.2 Integration and verification

11. Cybersecurity validation

Post-development phases

12. Production

13. Operations and maintenance

- 13.3 Cybersecurity incident response
- 13.4 Updates

14. End of cybersecurity support and decomissioning

15. Threat analysis and risk assessment methods

- 15.3 Asset identification
- 15.4 Threat scenario identification
- 15.5 Impact rating
- 15.6 Attack path analysis
- 15.7 Attack feasibility rating
- 15.8 Risk value determination
- 15.9 Risk treatment decision

The cybersecurity standard contains normative references to parts of the ISO 26262 FuSa standard thereby integrating the functional safety (ISO 26262) and the cybersecurity (ISO 21434) lifecycles

Tool qualification requirements can be deduced from this section. Details follow on next slide

Guidance from this section shall be used to analyze cybersecurity related threats, risks, and remediations related to your software and tools.

Tool Qualification Requirements – ISO 21434

- Text from ISO 21434:5.4.5 Tool Management
  - 5.4.5 Tool Management
    - [RQ-05-14] Tools that can influence the cybersecurity of an item or component shall be managed.
      NOTE: Such management can be established by:
      - Application of the user manual with errata;
      - **Protection against unintended usage or action;**
      - Access control for the tool users; and/or
      - Authentication of the tool
  - [RC-05-15] An appropriate environment to support remedial actions for cybersecurity incidents (see 13.3) should be reproducible until the end of cybersecurity support for the product.
- **The tool shall be managed!**
  - This is the only requirement that needs to be satisfied
  - How to interpret this requirement, what are we supposed to do?

Highlighted in Green: **Data to be provided by the Security Manual vendor**

Cyber Security Tool Management 5.4.5

- The ISO cybersecurity standard **does not describe** a tool qualification process

- To solve this issue, we look to ISO 26262 which is **frequently referenced** by ISO 21434.
  ISO 26262 specifies several tool qualification methods:
  1. Increased confidence from use
  2. Evaluation of the development process
  3. **Tool validation**
  4. **Development in accordance with a safety standard.**

- For **higher ASILs** either **method “Tool validation” or “Development in accordance with a safety standard”** shall be used

- So, to qualify a tool for cybersecurity related software development it shall be shown that either the **tool meets its cybersecurity related requirements** (tool validation) or that the tool is developed **in accordance with a cybersecurity standard**

Identification of Tool related Cybersecurity Requirements

- The process applied by TASKING is based on the **Systems Security Engineering (SSE) Techniques** published by the National Cybersecurity FFRDC (NCF) **part of MITRE**, a non-profit organization funded by the US government.
  - **FMEA**: Failure Mode and Effect Analysis
    To analyze the potential cybersecurity related risks that a Compiler Toolchain can introduce into the user's software
    - The behavior of the compiled software shall not violate the intentions of the user under both "normal" and under "cybersecurity attack" conditions.
    - Analysis is done by engineers with an in-depth knowledge about the Compiler Toolchain and its internals
  - **TARA**: Threat Analysis and Remediation Analysis
    - The Common Known Pattern Enumeration and Classification (**CAPEC**), Common Weakness Enumeration (**CWE**) and Common Vulnerability and Exposures (**CVE**) databases provide input for this analysis. These databases contain **known patterns of attack, known weakness types, and known cybersecurity vulnerabilities**, and provide ways for risk assessment and remediation.
- Tool supplier should execute this cybersecurity analysis and You should use the results

Identification of Tool related Cybersecurity Requirements

- The output of the **FMEA** and **TARA** process are:
  - **Security related requirements** to be implemented by the supplier of the toolset, and
  - **Security related guidelines** to be implemented by the user of the toolset

- The results of the compiler security qualification are described in the **Toolset’s Safety & Security Manual** and addresses:
  - The **correct usage of the tool**
    - By providing guidelines that explain how to prevent or mitigate security related risks associated with the compilation process and by explaining the facilities embedded in the compiler that protect against cybersecurity attacks
  - **Protection against unintended usage or action**
    - By implementing facilities in the compiler that protect against unintended actions and by explaining the residual risks related to the compilation and optimization process

Key Take Away

- Any Build Tool used within automotive software production must be described in the Safety Manual
  - Tools with a TCL 2,3 require a Tool qualification
  - The confidence level influences the tool qualification effort
- It is recommended to precisely define the tools to be used in the build process of productive software
  - The number of tools used influences the tool qualification effort
- ISO 21434 does not describe tool qualification in detail
  - It references ISO 26262 which is known already (lower risk)
  - As TCL1,2,3 does not exist in ISO 21434 a Cybersecurity Tool Management is required
- Safety Manual to be provided by the tool supplier
  - Upfront check if Safety Manual can be provided (e.g. Open Source Tools)
  - The Safety Manual describes the safe and secure use of the tool (e.g., recommended tool options)
- Third party software suppliers
  - Do they use exact the same build tools (e.g., exact same version / build number)
  - Different version of the same tool might not be covered by the Safety Manual
  - Different versions have different issues not covered by applied risk mitigation

Contact

STAY CONNECTED

Contact us directly
www.tasking.com/contact

Follow us on LinkedIn
www.linkedin.com/company/tasking-inc

Follow us on WeChat

Sign up to get the latest news from TASKING®
www.tasking.com/subscribe

Upcoming webinars, events and exhibitions
www.tasking.com/events

MEET US AT ARC SUMMIT

Demo at evening networking reception

TASKING presenting its SmartCode development environment for Infineon TC4x including an ARC EV71 (PPU)

Including Safety Manual

THANK YOU