Tool Qualification in Multiple Domains: Status and Perspectives

Author(s):

Jean-Louis CAMUS (Esterel Technologies), Michael P DEWALT, (Federal Aviation Administration)
Frédéric POTHON (ACG Solutions), Gérard LADIER, (Aerospace Valley),
Jean-Louis BOULANGER (CERTIFER), Jean-Paul BLANQUART (Astrium),
Philippe QUERE (Renault), Bertrand RICQUE (Sagem Défense Sécurité)
Jean GASSINO (Institut de Radioprotection et de Sûreté Nucléaire)

Abstract

This paper provides a global perspective on qualification of tools used for development or verification of safety critical software. The increasing complexity and criticality of safety critical software requires a high degree of rigor in the development and verification processes. These processes are regulated by standards such as DO-178C/ED-12C for airborne software, EN 50128 for railway equipment, IEC 61508 / IEC 61511 / IEC 62061 for industry, ISO 26262 for automotive, ECSS (in particular Q80, E40) for European space and IEC 60880 for the nuclear industry. Development and verification of application software increasingly rely on the use of tools automating complex verification and/or development activities. This paper provides a comparative overview of the current major standards regarding tools, and proposes improvements in the approach for tool qualification.

Keywords

Tool, Qualification, Safety, Embedded, Software, DO-178C/ED-12C, DO-330/ED-215, EN 50128, IEC 61508, IEC 61511, IEC 62061, IEC 60880, ISO 26262, ECSS E40, ECSS Q80

1 Introduction

Safety critical software requires a high degree of rigor in its development and verification processes. These processes are regulated by standards such as DO-178C/ED-12C for airborne software, EN 50128 for railway equipment, IEC 61508 / IEC 61511 / IEC 62061 for industry, ISO 26262 for automotive, ECSS (in particular Q80, E40) for European space and IEC 60880 for the nuclear industry. As a consequence of the growing complexity of application software, its development and verification rely more and more on the use of tools automating complex verification and/or development activities. Failure of such tools may compromise the integrity of the application software itself. In order to mitigate this risk, the above mentioned standards put integrity requirements on tools in the form of qualification / certification requirements.

2 Comparison of Safety Related Standards Requirements Regarding Tool Qualification

2.1 Organization of this Section

This section provides a comparison of the approaches and requirements for tool qualification for some major standards in the different domains. There is one section per application domain, each of them summarizing the following characteristics of the tool qualification approach in that domain:

- Tool qualification history overview
- Tool categorization
- Requirements on tool development
- Tool Usage requirements

2.2 Civilian Airborne Software

2.2.1 <u>Tool Qualification History Overview</u>

Tool Qualification as a topic first appeared in DO-178B/ED-12B issued in December of 1992. In prior versions (DO-178, issued in November of 1981, DO-178A, issued in March of 1985) the need to consider the assurance of tools that can contribute to errors was largely ignored. DO-178A introduced the requirement for structural coverage of the source code. Time intensive manual means of achieving coverage was quickly supplanted by efficient structural coverage tools. Also other tools to reduce other areas of verification effort were introduced as well. Prior to DO-178B some programs were using development tools to convert graphical representations of control laws directly into assembly code. These programs were requesting relief from associated verification activities such as code reviews and traceability. This higher dependence on tools resulted in another uncontrolled source of potential errors. To address this emerging error source, DO-178B created a separate section requiring qualification of tools to explicitly address the need to address tools related to the verification process and tools that replaced any portion of the development process such as programming or requirements generation. With the increasing creation of and reliance upon tools for modern certification projects, DO-178C and associated DO-330 created a set of objectives and activities for tool qualification to replace the largely qualitative approach specified in DO-178B to address tools. DO-330 was created as a separate document for tool qualification that could be used to provide assurance for tools beyond the software development domain.

2.2.2 Tool Categorization

Section 12.2.2 in DO-178C/ED-12C defines three criteria that determine the applicable tool qualification level (TQL) with regard of the software level.

a. Criteria 1: A tool whose output is part of the airborne software and thus could insert an error.
b. Criteria 2: A tool that automates verification process(es) and thus could fail to detect an error, and whose output is used to justify the elimination or reduction of:
    1. Verification process(es) other than that automated by the tool, or
    2. Development process(es)that could have an impact on the airborne software.
c. Criteria 3: A tool that, within the scope of its intended use, could fail to detect an error.

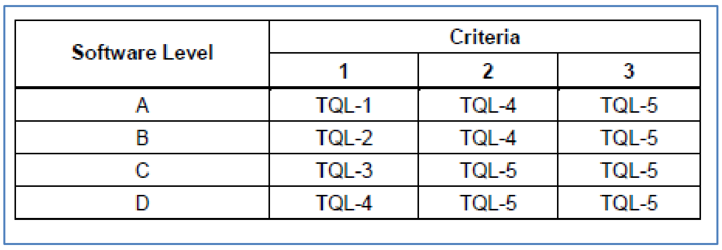
Criterion 1 subsumes what were called “development tools” in DO-178B/ED-12B, while the other two criteria split the former “verification tools” depending on the certification credit claimed by the qualification of the tool. Criterion 3 is the “classic” use of a verification tool: the purpose of the tool is to produce or verify an artifact, and the certification credit claim is only on objectives applicable to this artifact. For Criterion 2, the certification credit claimed is extended to objectives that are beyond the data directly verified by the tool.

An appendix of DO-330/ED-215 provides additional rationale for these 3 criteria and also some examples of the difference between Criteria 2 and 3. Typically a proof tool verifies the source code based on properties defined as part of the requirements. An applicant may claim alleviation of test effort based on these same requirements. In such case, the applicable criterion is 2. It should be noted that this example is at the origin of the definition of this additional tool qualification criteria.

One of the main principles of DO-178C/ED-12C is to require multiple verification filters to improve the error detection. The certification credit claimed with the application of criteria 2 is equivalent to the removal of one filter. This filter is considered as useless (in term of error detection) as the verification performed by the tool is claimed as being thorough enough to detect the possible errors. That is why for these tools the Tool Qualification Level (TQL) is higher than for a “classic” verification tool.

The applicable TQL is defined based on the qualification criterion and the software level:

Table 1: Tool Qualification Levels in DO-178C



Software Level	Criteria		
	1	2	3
A	TQL-1	TQL-4	TQL-5
B	TQL-2	TQL-4	TQL-5
C	TQL-3	TQL-5	TQL-5
D	TQL-4	TQL-5	TQL-5

The TQL applicable for Criterion 1 is the replacement of DO-178B/ED-12B development tool for each software level, while TQL-5 for Criterion 3 is the replacement of DO-178B/ED-12B verification tool. The TQL applicable for Criterion 2 basically requires a higher level of rigor for tools used on software at level A or B, in order to increase the confidence in the use of the tool (that is, TQL-4 instead of TQL-5).

2.2.3 Requirements on Tools Development

TQL-5 qualification is limited to an external assessment of the tool. User needs are defined in form of Tool Operational Requirements (TOR), and verification of compliance of the tool to this TOR in the operational environment is performed. However DO-330/ED-215 brought a novelty in form of "validation" objectives in two steps. First, the TOR is assessed for demonstration of the coverage of the certification credit claimed by the tool qualification. Second, through tool execution, it is demonstrated that the user needs are satisfied. This activity should identify gaps or misinterpretation of TOR.

TQL-4 requires that the Tool Requirements data describe all functionality implemented in the tool and provide additional detail about the tool architecture. TQL-4 also requires verification that the tool complies with Tool Requirements. The initial intent of the criteria 2 was to address the tools implementing formal methods. Therefore TQL-4 requires information about the maturity and technical background of any technology or theory implemented in the tool.

TQL-1, TQL-2 and TQL-3 are applicable to the criteria 1 tools, typically the autocode generators. So these TQL require a complete tool life cycle processes definition. These TQL are similar to the rigor applicable to the resulting software itself.

2.2.4 Tool Usage Requirements

One of the main improvements of DO-330/ED-215 is the identification of a separate set of objectives applicable to the tool user. Rationale for this addition is to recognize that some tool errors may be only detected in the tool user context. Independently of the organization, these objectives clearly identify some qualification activities that are necessary to perform from the tool user's perspective. They include:

- The definition of the certification credit, and of the tool qualification criteria and level
- The development of the TOR
- The installation of the tool in the operational environment
- The verification and validation of the TOR
- The verification (compliance to the TOR) and validation (compliance to user needs) of the tool in the operational environment.

2.3 Space

2.3.1 Tool Qualification History Overview

In European space, standards have been elaborated since the seventies under the form of the Procedures, Specifications and Standards (PSS) of the European Space Agency, now replaced by the series of standards established since the nineties' by the European Cooperation for Space Standards (ECSS) covering most, if not all parts of project management, product assurance and engineering. A major objective is to establish the means to appropriately control large complex projects. Issues related to the utilization of tools have been considered since the beginning of these standardization initiatives, with a strong focus on tools producing software code (first, compilers and now, automatic generators of code).

2.3.2 Tool Categorization

Even in their latest versions (issue C, March 2009) the software engineering standard and the software product assurance standard do not provide many details and in particular no categorization in the sense it is addressed in other domains e.g., aeronautics. The standards only mention in general "tools used to support software development, or validation" and contain a specific mention of tools supporting automatic generation of code. However even for the latter, the requirements (as shown below) are not very detailed; one reason being that the code produced by such tools is subject to the same verification requirements as manually produced code.

2.3.3 Requirements on Tools Development

There are no specific detailed requirements on tools development. Even if the notion of "tool qualification" is used, there is no precise indication of how a tool is to be qualified. The only requirement is that the selection of tools must be documented (and agreed with the customer), justifying that they are appropriate with respect to the intended usage and project and software characteristics (including software criticality), in addition to other characteristics on licensing scheme, availability along the project duration etc. It is worth noting that in practice, the software plans elaborated for critical space projects go far beyond the standard requirements for what concerns tool qualification, most often on the bases of practice and more detailed standards in particular from aeronautics.

2.3.4 Tool Usage Requirements

There are no specific detailed requirements on tool the user, except very general requirement on the consideration of installation, preparation, training and use in the selection of tools.

2.4 Rail

2.4.1 Tool Qualification History Overview

In the first version of CENELEC 50128:2001, the need for tools qualification is focalized on translators and compilers. In the body of the standard the clause 10.4.9 required that the programming language selected shall have a translator/compiler which has one of the following:

i) A "Certificate of Validation" to a recognized National/International standard;
ii) An assessment report which details its fitness for purpose;
iii) A redundant signature control based process that provides detection of the translation errors.

The last point is related to a specific technique used in the mono coded processor (used by many French companies), where safety is ensured (including w.r.t. software) on an architecture with only one processor.

In addition in the software design phase, table A.4 introduces specific requirements for tools.

Table 2: ISO 26262 Table A-4Tool Requirements

	SIL0	SIL1/SIL2	SIL3/SIL4
….			
11. Validated Translator	R	HR	HR
12. Translator Proven in Use	HR	HR	HR
13. Library of Trusted/Verified Modules and Components	R	R	R
….			

In the new 2011 version of CENELEC 50128, in section 6 called “software assurance”, section 6.7 is dedicated to tool qualification.

2.4.2 Tool Categorization

The objective of tool qualification is to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety related manner that is undetected by technical and/or organizational measures outside the tool. To this end, software tools are categorized into three classes namely,

- Class T1 includes tools generating no outputs, which can directly or indirectly contribute to the executable code (including data) of the safety related system.
- Class T2 includes tools supporting the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software.
- Class T3 includes tools generating outputs, which can directly or indirectly contribute to the executable code of the safety related system.

2.4.3 Requirements on Tools Development

In section 1.4, the standard indicates that it can be applied for the development of tools.

2.4.4 Tool Usage Requirements

The requirements established by CENELEC 50128 on tools are related to the selection of tools. In the body of the standard many requirements related to the use of the tools by the user are introduced.

One of the main ideas of CENELEC 50128 is that “when tools are being used as a replacement for manual operations, the evidence of the integrity of tools outputs can be adduced by the same process steps as if the output was done in manual operation”. These process steps might be replaced by alternative methods if an argumentation on the integrity of tools output is given and the integrity level of the software is not decreased by the replacement.

The new CENELEC 50128:2011 introduced eleven requirements for the tools qualification that cover: the identification, a safety analysis, the specification, the validation (based on return of experience, based on certificate, based on specific testing), the selection and management of code generator, the configuration management (tool name, reference, version, Tx, function, option, etc.) and the process for qualify a new version.

2.5 General Industry

The general industry sector situation, as far as the development of safety critical systems and software is concerned, is dominated by the emergence of the IEC 61508 umbrella standard in the late nineties. This standard has been quickly adopted by the manufacturers of safety critical equipment (sensors of any type, electrical and electro-mechanical actuators, programmable logic controllers, and communication networks) as well as end-users.

2.5.1 Tool Qualification History Overview

The edition 1 of IEC 61508 was approaching software development tools only through evaluation requirements. The tools had to be evaluated for their suitability and integrity taking into account the targeted safety integrity of the software to be developed. The use of “certified” tools was only quoted as a methodological possibility in an annex without describing in any way what this “certification” should be. The general industry sector specific standards (IEC 61511 and IEC 62061) follow as well this approach.

The edition 2 has recently introduced the concept of tool categorization in a way presenting strong analogies with DO178B/ED-12B. In addition the standard puts requirements on the relationship between the tools and the safety related systems.

2.5.2 Tool Categorization

IEC 61508 differentiates on-line and off-line tools. On-line support tools are part of the safety related system. Off-line tools are defined as supporting a phase of the software development lifecycle and not being capable of directly influencing the safety-related system during its run time. They are categorized into the following classes:

- Class T1 includes tools generating no outputs which can directly or indirectly contribute to the executable code (including data) of the safety related system.
- Class T2 includes tools supporting the test or verification of the design or executable code, where errors in the tool can fail to reveal defects but cannot directly create errors in the executable software.
- Class T3 includes tools generating outputs which can directly or indirectly contribute to the executable code of the safety related system.

The standard focuses on the use of classes T2 and T3 in safety related software development.

2.5.3 Requirements on Tools Development

IEC 61508 puts few direct requirements on the tool development. The standard expects some properties to be achieved by the tools: the main expected property being its suitability for the targeted application. Examples of requirements are:

- Have a translator which has been assessed for fitness for purpose;
- Use only defined language features;
- Match the characteristics of the application;
- Contain features that facilitate the detection of design or programming mistakes;
- Support features that match the design method.

2.5.4 Tool Usage Requirements

Most of the requirements put by IEC 61508 on tools are related to the selection and the use of the tools by the user. These requirements concern mandatory assessment leading to validation of the tools by a third party or by the user himself. This validation must provide the means to determine the level of reliance placed on the tools, and the potential failure mechanisms of the tools that may affect the executable software. Where such failure mechanisms are identified, appropriate mitigation measures shall be taken. The evidence may be based on a suitable combination of history of successful use in similar environments and for similar applications, and of tool validation.

2.6 Nuclear Industry

2.6.1 Tool Qualification History Overview

Tools for design and verification of safety critical software have been used in the nuclear industry since the 1980s. Standards such as IEC 60880 (released in 1986, revised in 2006) provide guidance for their qualification. This standard recognizes that using appropriate tools can reduce the risk of introducing faults in the development process but advises not to require tools to undertake tasks beyond their capability, for example, they cannot replace humans when judgment is involved.

As a major example, “application-oriented languages” and “code generators” (according to IEC 60880 wording) have been widely used since the end of the 1980s to develop the application specific part of safety critical software. In France, the first project based on this approach used SAGA (predecessor of SCADE), providing the user with domain-specific graphical constructs on top of the formal language LUSTRE. The excellent experience feedback has led to generalize this approach.

2.6.2 Tool Categorization

IEC 60880 identifies the following categories of tools: transformation tools such as code generators or compilers, verification tools such as static code analyzers or test coverage monitors, tools for on-line diagnosis, tools that support the development process and configuration control tools.

The limits of applicability of each tool must be documented. Tools must be verified and assessed consistently with their category, with the consequences of a fault in the tool and with their potential to introduce faults in the software or fail to detect such a fault. Principles of defense in depth and diversity may help reducing the requirements on individual tools, for example when the output of one tool can be fully verified or compared with the output of a functionally similar but diverse tool.

2.6.3 Requirements on Tools Development

Tools are qualified if they are developed according to the requirements applicable to the target software; otherwise the qualification strategy must consider the tool development process, the tool provider history, the adequacy of the documentation to allow verifying the outputs, the tool testing and validation, the evaluation of the tool over a period of use and its feedback of experience.

Specific requirements apply to translators and compilers, e.g. they must not remove defensive programming features without warning. Compiler libraries must fulfill the requirements of the standard about COTS.

2.6.4 Tool Usage Requirements

Other requirements concern the tool inputs and outputs and are provided only for “Application-oriented languages”: their formalism (usually graphical, e.g. logic diagrams) must be comprehensible to all participants such as process, system and software engineers; it must allow expressing the software requirements and taking into account constraints on the system architecture, e.g. mapping of functions to system components; it must support the standardization of layout, the modularity and the avoidance of unsafe features; it must preserve the required well-defined syntactic and semantic rules of the underlying language.

2.7 Automotive

2.7.1 Tool Qualification History Overview

ISO 26262 published as an international standard in November 2011 provides requirements and recommendations for the qualification of software tools used in the development of a system or its software or hardware elements. In the final release of the standard, the chapter on tool qualification is named “Confidence in the use of software tools” to highlight the purpose which is to increase the confidence in the tools.

2.7.1 Tool Categorization

ISO 26262 defines three “tool confidence levels”: TCL1 (no qualification), TCL2 (first set of qualification methods) and TCL3 (qualification with more constraining methods for higher ASIL).

Table 3: Determination of the tool confidence level (TCL) in ISO 26262

		Tool error Detection (TD)		
		TD1 (high confidence)	TD2 (medium confidence)	TD3 (other cases)
Tool Impact (TI)	TI1	TCL1	TCL1	TCL1
	TI2	TCL1	TCL2	TCL3

The tool impact indicates “the possibility that a malfunction of a particular software tool can introduce or fail to detect errors” [ISO 26262]. If there is an argument that there is no such possibility, TI1 shall be selected and no qualification needs to be performed. Otherwise, select TI2.

The tool error detection indicates “the confidence in measures that prevent the software tool from malfunctioning and producing corresponding erroneous output, or in measures that detect that the software tool has malfunctioned and has produced corresponding erroneous output” [ISO 26262].

In correspondence with tool classification level, the following methods can be applied to achieve the required level of confidence in a software tool:

Table 4: Highly recommended qualification methods per tool levels in ISO 26262

Methods	Highly recommended methods for TCL2	TCL3
1. Increased confidence from use	ASIL A, B and C	ASIL A and B
2. Evaluation of the tool development process	ASIL A, B and C	ASIL A and B
3. Validation of the software tool	ASIL D	ASIL C and D
4. Development in accordance with a safety standard	ASIL D	ASIL C and D

For each tool qualification method, requirements are specified in ISO 26262. The split between the tools development requirements and the tools usage requirements is not given in the standard. Nevertheless, we can try to present the possible impact on the tools development and on the tools usage.

2.7.2 Requirements on Tools Development

The two methods that can have an impact on the tools development are:

2. Evaluation of the tool development process

4. Development in accordance with a safety standard

For the *evaluation of the tool development process*, ISO 26262 requires the compliance of the process with an appropriate standard. The evaluation of this development process shall be provided based on an appropriate national or international assessment standard.

For the *development in accordance with a safety standard*, the ISO 26262 requires compliance with a relevant subset of the requirements of the selected safety standard (given examples are: ISO 26262, IEC 61508 or RTCA DO-178).

The current field experience and feedback is that the development of a software tool following a safety standard is difficult. Thus, the method 2 is seen as the most feasible.

2.7.3 Tool Usage Requirements

The two methods that have an impact on the tools usage are:

1. Increased confidence from use

3. Validation of the software tool

For the *increased confidence from use*, the idea is to confirm that the tool version and its usage are the same or close enough to the previous usages. Adequate and sufficient data on the tool functioning shall also be provided.

For the *validation of the software tool*, the requirements of ISO 26262 demands mainly the demonstration of the compliance of the tool with its specified requirements, the analyses of malfunctions during validation and the reaction of the software tool to anomalous operating conditions.

The current field experience and feedback is that the data collection, and frozen version necessary for method 1 makes it difficult for method 1 to argument in practice. The method 3, validation of the software tool, appears more feasible. In addition, on the usage side, "when using a qualified tool, it shall be ensured that its usage, its determined environmental and functional constraints and its general operating conditions comply with its evaluation criteria or its qualification" [ISO 26262].

2.8 Comparative Summary of Approaches in the Different Domains

2.8.1 Tool Qualification History

Avionics has started the earliest. All domains have introduced requirements for tool qualification and tend to refine these requirements at each new edition.

2.8.2 Tool Categorization

All standards categorize tools based on the potential impact of a tool failure on the application software. DO-178C/ED-12C andDO-330/ED-215 and ISO 26262 currently have the most elaborated categorization.

2.8.3 Requirements on Tools Development

All standards put requirements on the tools based on their category. DO-330/ED-215 currently has the most elaborated requirements.

2.8.4 Tool Usage Requirements

All standards require some user documentation. Only DO-330/ED-215 formally identifies a set of objectives with related activities and tool qualification data to be satisfied by each tool user.

Regarding the users skills, EN 50128, ISO 26262 and IEC 61508 put requirements on skills of software team members, including for tool users.

3 Perspectives

Several aspects of tool qualification deserve further investigation and refinement in particular:

- Tool Usage and Human Factors
- Error Impact Analysis
- Harmonizing Qualification Guidance amongst Standards

3.1 Tool Usage and Human Factor

Early tools used to be reasonably simple. Over the years, one has been confronted with a kind of quadratic growth in complexity: each individual tool may perform more complex functions, and the number of tools, running isolated or in combination with others, increase. It becomes more and more complex and tedious to consistently use these tools. In order to compensate for this, both the tool vendors and the tool integrators often encapsulate tools and tool-chains, with the intent to make the user's life simpler.

There is a human factor risk that tool users may lose understanding of what tools are supposed to do and what exactly occur when using their tool-chain. Awareness by the tool users of the state of a tool-chain under operation, and of the controls needed to ensure adequate tool execution should be ensured.

3.2 Error Impact Analysis

All current standards base their tool qualification objectives on the potential impact of tools errors. Typically, all standards consider tools that can directly inject an error into the final embedded software as a category that needs special care. This distinction is obviously the minimum that should be considered, but the impact analysis of errors should be refined in the specific context of an application project. This should take into account the way the impact of undetected errors may be leveraged or mitigated by other tools or processes using data containing the errors. This may require appropriate mitigation techniques and more or less demanding objectives for these tools.

3.3 Harmonizing Qualification Guidance amongst Standards

Current standards have different guidance for comparable tools functions. These requirements should be harmonized in order to ensure a degree of confidence commensurate with the potential risks of tool errors.

DO-330/ED-215 is the first document dedicated to tool qualification. One of the original goals of this document was to be usable across a variety of application domains. Yet, this document is to be used in conjunction with a domain-specific standard that governs the acceptability of the considered tool. To make DO-330/ED-215 applicable, the relevant domain-related document must:

- Identify that DO-330/ED-215 is applicable
- Define its own tool qualification criteria
- Define, based on these criteria and other considerations if necessary (e.g. the reliability of the product) the appropriate tool qualification level (TQL-1 to TQL-5)

If this separation allows each domain to be make independently its own decision in term of applicable criteria and level of tool qualification, the use of DO-330/ED-215 in a domain other than civilian airborne still requires adaptation to fulfill all the domain constraints.

4 Conclusion and Future Work

This paper has analyzed current standards and practice regarding the usage of tools and their qualification. Major advances have occurred towards a more systematic management of tools and standards tend to converge over the years.

It is suggested that the next steps should be the harmonization of standards, a systematic error impact analysis including the tools usage process and consideration of human factors.

5 References

[DO_178C/ED 12C] Software Considerations in Airborne Systems and Equipment Certification, DO-178C, RTCA Inc./EUROCAE, 2011

[DO_330/ED 215] Software Tool Qualification Considerations, DO-330, RTCA Inc. ./EUROCAE, 2011

[ECSS_E_ST40C] Space engineering Software, ECSS-E-ST-40C, 2009.

[ECSS_Q_ST_80C] Space product assurance Software product assurance, ECSS-Q-ST-80C, 2009

[EN_50128] Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems, EN 50128:2011.

[IEC 60880] Nuclear power plants Instrumentation and control systems important to safety Software aspects for computer-based systems performing category A functions, IEC 60880, 2006.

[IEC_61508_3_2010] Functional safety of E/E/PE safety-related systems / Part 3: Software Requirements, IEC 61508-3:2010, IEC.

[IEC 61511] Functional safety – Safety instrumented systems for the process industry sector. / IEC 61511 Parts 1-3, edition 1.0, 2003

[IEC 62061] Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems, IEC 62061, Edition 1.0, 2005

[ISO 26262] Road vehicles – Functional safety. Parts 1-9, first edition, 2011, Part 10, first edition, 2012.