Safety and Reliability

ISSN: (Print) (Online) Journal homepage: www.tandfonline.com/journals/tsar20

Mapping to IEC 61508 software developed to ISO 26262

Peter Okoh & Thor Myklebust

To cite this article: Peter Okoh & Thor Myklebust (2024) Mapping to IEC 61508 software developed to ISO 26262, Safety and Reliability, 43:3, 186-201, DOI: 10.1080/09617353.2024.2363073

To link to this article: https://doi.org/10.1080/09617353.2024.2363073

SAFETY AND RELIABILITY
2024, VOL. 43, NO. 3, 186–201
https://doi.org/10.1080/09617353.2024.2363073

RESEARCH ARTICLE
OPEN ACCESS

Mapping to IEC 61508 software developed to ISO 26262

Peter Okoh[a] and Thor Myklebust[b]

[a]Autronica Fire and Security, Trondheim, Norway; [b]SINTEF Digital, Trondheim, Norway

ABSTRACT
Several functional safety standards such as ISO 26262 (automotive), IEC 61511 (process), EN 5012X (railway), IEC 62061 (machinery), IEC 61513 (nuclear), etc. have evolved from IEC 61508 (generic) over the years. The evolution of the standards is accompanied with additional requirements and guidance that are industry-specific. However, in certain cases, technological advancements happen at a rate that is too rapid for a standard to regulate, thus creating room for unguided interpretation and confusion in addition to the potential to make existing designs obsolete. To address this problem, the reuse of resources (e.g., safety artefacts) across industries is being promoted, whereby an industry that is more aligned to the state-of-the-art will help the underprivileged one to fill gaps. However, it is important to clearly define the framework for industry-to-industry exchange in order to avoid confusion. The objective of the paper is to investigate whether and how safety levels for software developed to ISO 26262 (automotive) can be mapped to safety levels for software developed to IEC 61508. The paper builds on review of literature and standards and is focused on software elements.

ARTICLE HISTORY Received 23 August 2023; Revised 2 May 2024; Accepted 7 May 2024

KEYWORDS Automotive; industrial; software; functional safety; ASIL; SIL; ISO 26262; IEC 61508

1. Introduction

Before 2011, the automotive industry relied on IEC 61508 for functional safety developments. However, in 2011 the industry carved an identity for itself with the first version of ISO 26262 followed by the second edition at the end of 2018. Subsequently, rapid changes in the global technological space ensued and resonated with the automotive industry, leading it into a speedy evolution in the development of advanced hardware and software (based on ISO 26262). Meanwhile, developments based on IEC 61508 lagged. However, the general industrial sector (based on IEC 61508) identified an opportunity to reuse hardware and software from the automotive industry in order to keep abreast with the pace of technological change,

CONTACT Peter Okoh okohpee@yahoo.com Autronica Fire and Security, Trondheim, Norway.

reduce time to market and promote the financial bottom line with new products. Meanwhile, the automotive industry keeps its doors open to allow the flow of interesting safety elements from other domains into it, which is supported by the Safety Element out of Context (SEooC) provision in ISO 26262. The reuse of external safety manual in ISO 26262 is relevant to the SEooC concept. However, it is only defined and described in IEC 61508, where the content list is presented in the normative Annexe in parts 2 and 3. This implies that in certain cases, children standards are not a complete departure from the parent standard, and both can still strengthen each other in some ways. The situation is expected to be more challenging for software as will be seen in this paper compared to hardware which was handled in an earlier paper (Okoh & Myklebust, 2024).

Software, no matter their application, are not subject to random failure because they are not degradable (i.e., they have no failure rate), being intangible assets (Okoh, 2019). Therefore, there is no basis for a (quantitative) random safety integrity for software. However, software can experience systematic failure (e.g., deficiencies in architecture and/or coding). Hence, for Electrical, Electronic and Programmable Electronic (E/E/PE) safety-related systems, the safety integrity of the software is determined in relation to the development assurance of the software as defined by the functional safety standard being applied. Development assurance, as defined in Kritzinger (2017), is a process involving specific planned and systematic actions that together provide confidence that errors or omissions in requirements or design have been identified and corrected to the degree that the system, as implemented, satisfies applicable certification requirements. The level of software development assurance is basically defined by the level of software development rigour or stringency in relation to the implementation of methods, techniques or measures stipulated by standards. Methods is used in ISO 26262 and is similar to techniques and measures which is used in IEC 61508.

Currently, no standardised basis exists for requalifying software originally developed according to ISO 26262 but intended to be reused in relation to IEC 61508. The task of recertifying software from one domain to another can be challenging. Therefore, it is important for the certification body of knowledge to have a common conversion framework/template that would reduce the tedium and eliminate confusion. Besides, if a software being developed from scratch was not developed correctly according to the applicable domain standard (say, ISO 26262, 2018), it would be challenging to retrospectively get it certificated again according to the standard of another domain (say, IEC 61508, 2010). Hence, following rules diligently through the software development process in whatever domain will make the later process of closing gaps between standards cheaper and quicker in case a company decides to diversify its market or product design. A few peer-reviewed publications have been identified on cross-domain mapping of aspects of functional safety standards (Crots et al., 2014; Gerlach et al., 2011; Machrouh et al., 2012; Miller, 2020; Okoh et al., 2022; Ruiz et al., 2017), but only one (Miller, 2020) focused on the relationship between the software sections of IEC 61508:2010 (part 3) and ISO 26262 (2018) (part 6 and part 8) although it showed a coarse consistency of aspects of the former with those of the latter without drawing any final conclusion. Hence, the need for further work.

The objective of this paper is to requalify for reuse with respect to the safety levels of IEC 61508, software which was originally developed according to ISO 26262. This paper is focused on software elements only, including its systematic safety integrity. The paper will compare software Techniques & Measures (T&M) in ISO 26262 with those in IEC 61508. The research outcome is expected to support the issuance of the Technical Report (TR), IEC TR 61508-6-1 "Treatment of hardware or software developed to ISO 26262" which is expected to be completed in 2025 by the JTG20 Work Group. The rest of the paper is structured as follows. Firstly, a comparison of software development assurance across four functional safety standards is presented. Secondly, a more detailed comparison of software development assurance of ISO 26262 and DO-178C is presented. Subsequently, the software developed according to ISO 26262 is mapped to IEC 61598, taking artefacts, supporting processes and software safety levels into consideration. Next, discussion and recommendations are presented, followed by a conclusion.

2. Review: comparing software development requirements of four safety standards

The work carried out by Machrouh et al. (2012) established three dimensions for comparing the development assurance of software developed according to different functional safety standards, namely: supporting processes, development processes and verification processes. But this should have included tool qualification, which is a key dimension, a gap that will be filled in a subsequent section of this paper. Similarities across domains in relation to whether means for software development and verification are specified in their functional safety standards are shown in Table 1. In addition, Machrouh et al. (2012) also established six dimensions for comparing the influence of software development assurance level (DAL) on software safety assurance level as shown in Table 2. The Three dimensions mentioned at the beginning

Table 1. Standards with software development and verification means (Machrouh et al., 2012).

	Automotive (ISO 26262)	Generic (IEC 61508)	Railway (EN 50128)	Nuclear (IEC 61513, 60880, 62138)
Design & programming rules	X	X	X	X
Requirement and architecture notations	X	X	X	X
Methods of verification by analysis	X	X	X	X
Types and methods of testing	X	X	X	X
Testing environments	X	X	X	X

Table 2. Development assurance level influence on safety assurance level (Machrouh et al., 2012).

	Automotive (ISO 26262)	Generic (IEC 61508)	Railway (EN 50128)	Nuclear (IEC 61513, 60880, 62138)	Aero (DO-178)
PRODUCT	**Influence on software safety assurance level**				
Software content (defensive programming, error detection mechanisms etc.)	Medium	Medium	Medium	Medium	None
PROCESS	**Influence on software safety assurance level**				
Quality assurance objectives	Medium	Medium	Medium	High	High
Processes' activities	None	None	None	Medium	High
Means (methods, tools, rules, standards)	High	High	High	Medium	None
Independence of verification activities (Software or process' conformity to standard)	Medium	Medium	High	Medium	Medium
Independence of validation (processes' work product conformity)	Medium	Medium	High	Medium	High

of Section 2 are a condensed form of the six dimensions mentioned in Table 2. Using these dimensions as basis for comparison, both tables indicate a coarse equivalence between the software of ISO 26262 (2018) and IEC 61508 (2010), thus providing a promising basis for more detailed gap analysis between both standards. According to Machrouh et al. (2012), establishing cross-domain equivalence between IEC 61508 and other standards is expected to be less tedious when limited to standards that are derived from or influenced by IEC 61508. This is a reason for cautious optimism ahead of further investigation where mapping in finer granularity is expected.

3. Review: mapping software development from ISO 26262 to D0-178C

The objective of this section is to identify a framework that can be used in the next section as a basis for mapping software development of ISO 26262 (2018) to IEC 61508:2010. To this end, a pairwise framework of Gerlach et al. (2011) and Crots et al. (2014) for mapping software between the automotive safety standard (ISO 26262) and its avionics counterpart (DO-178C) is being reviewed in this section.

Figures 1 and 2 show software-related mappings between ISO 26262 and D0-178C with respect to artefacts and supporting processes respectively. Artefacts (also called work product in ISO 26262) are deliverables that show the output of the design process, whereas supporting

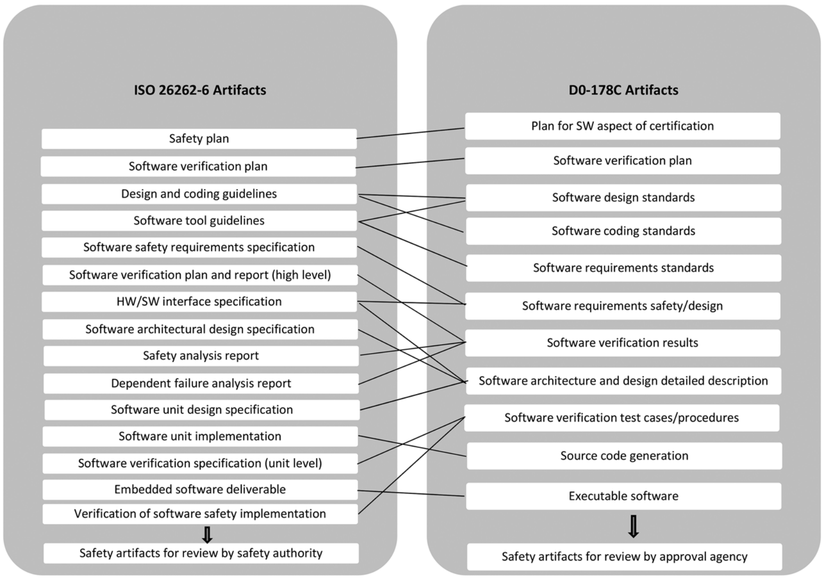
ISO 26262-6 Artifacts
Safety plan
Software verification plan
Design and coding guidelines
Software tool guidelines
Software safety requirements specification
Software verification plan and report (high level)
HW/SW interface specification
Software architectural design specification
Safety analysis report
Dependent failure analysis report
Software unit design specification
Software unit implementation
Software verification specification (unit level)
Embedded software deliverable
Verification of software safety implementation
Safety artifacts for review by safety authority
D0-178C Artifacts
Plan for SW aspect of certification
Software verification plan
Software design standards
Software coding standards
Software requirements standards
Software requirements safety/design
Software verification results
Software architecture and design detailed description
Software verification test cases/procedures
Source code generation
Executable software
Safety artifacts for review by approval agency

Figure 1. Comparing software artefacts of ISO 26262 and DO-178C - copied from Crots et al. (2014).

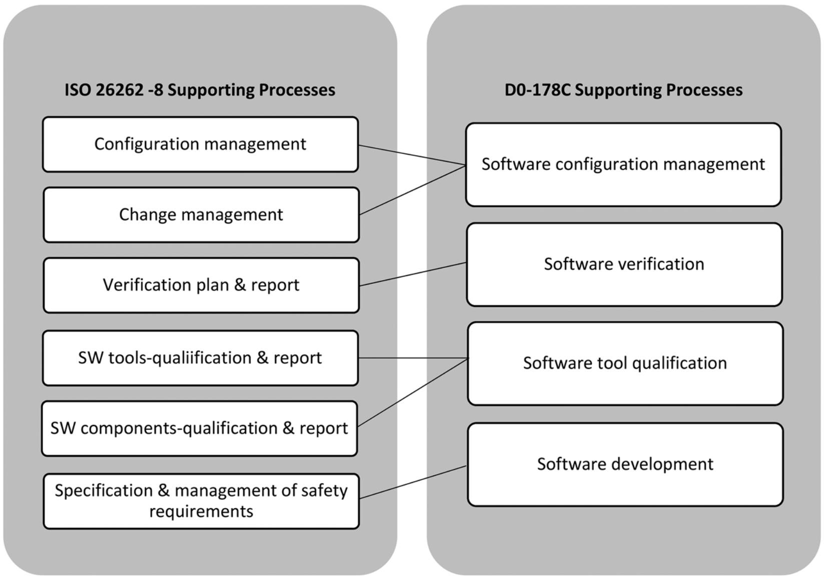
ISO 26262 -8 Supporting Processes
Configuration management
Change management
Verification plan & report
SW tools-qualiification & report
SW components-qualification & report
Specification & management of safety requirements
D0-178C Supporting Processes
Software configuration management
Software verification
Software tool qualification
Software development

Figure 2. Comparing supporting processes of ISO 26262 and DO-178C – copied from Crots et al. (2014).

processes are the horizontal processes that support the software development and establish the interface to review and certification activities (Gerlach et al., 2011). The selected artefacts as seen in Figure 1 are those that are relevant for the justification of safety (Gerlach et al., 2011). By comparing the names of the artefacts semantically and pragmatically, a promising correlation across both domains is seen, and this will improve by the time uncertainties in contents are eventually resolved. In the case of Figure 2, only mappable processes are considered, leaving out ISO 26262 supporting processes such as interfaces within distributed developments, documentation, and proven-in-use arguments (Gerlach et al., 2011). The implication is that there is an acceptable level of uncertainty when acknowledging evidence of supporting processes during mapping of software from ISO 26262 to D0-178C rather than the other way round. Based on this promising review, the approach of Gerlach et al. (2011) and Crots et al. (2014) is selected for further study in the next section and the unused supporting processes of ISO 26262 in this section will be reconsidered in relation to IEC 61508 to see whether they fit.

4. Mapping software safety level of ISO 26262 to IEC 61508

The aim in this section is to implement the lesson learned from the approach of Gerlach et al. (2011) and Crots et al. (2014) in Section 3, thus mapping software development assurance between ISO 26262 and IEC 61508. This will be combined with a more detailed investigation of the standards in relation to techniques/measures encompassing the three dimensions (i.e., development processes, verification processes and supporting processes) mentioned earlier by Machrouh et al. (2012).

In Figure 3, mapping of software-related artefacts from ISO 26262 to IEC 61508 is achieved satisfactorily. This is probably due to the fact that when ISO 26262 was being derived from IEC 61508 to serve as an industry-specific safety standard for the automotive domain, the intention was not a clean break from IEC 61508, but an opportunity to tailor certain aspects more to the automotive industry and thus emphasise some of the domain's perspectives which differ with that of others in terms of e.g. risk management (Verhulst et al., 2013). However, there is still need for detailed content analysis of artefact when mapping from ISO 26262 to IEC 61508 to reduce uncertainty.

Regarding the comparison of supporting processes, Figure 4 shows a seemingly perfect mapping from ISO 26262 to IEC 61508. In this case, the ISO 26262 supporting processes such as interfaces within distributed developments, documentation, and proven-in-use arguments which were (as described in Section 3) omitted in the mapping between ISO 26262 and DO-178 are also included. However, the contents of the processes also need a detailed analysis to reduce uncertainty. This is treated for processes and artefacts alike in Tables 3–6.

In Tables 3–6, detailed mappings of the contents of artefacts and supporting processes are realised. These tables consist of normative and informative techniques/measures (that help to avoid or control systematic failures) and their recommendation (i.e., - = Not Applicable, O = optional, NR = Not Recommended, + or R = Recommended and ++ or HR = Highly

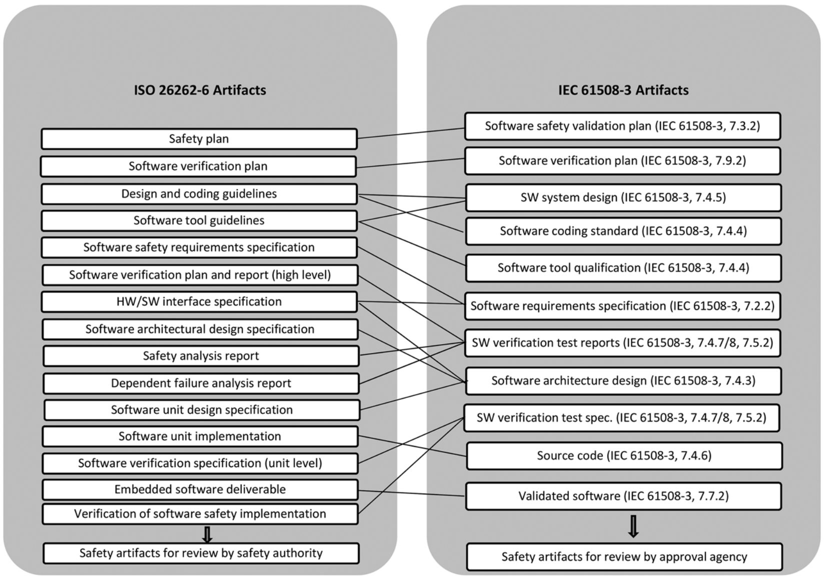
ISO 26262-6 Artifacts
Safety plan
Software verification plan
Design and coding guidelines
Software tool guidelines
Software safety requirements specification
Software verification plan and report (high level)
HW/SW interface specification
Software architectural design specification
Safety analysis report
Dependent failure analysis report
Software unit design specification
Software unit implementation
Software verification specification (unit level)
Embedded software deliverable
Verification of software safety implementation
Safety artifacts for review by safety authority
IEC 61508-3 Artifacts
Software safety validation plan (IEC 61508-3, 7.3.2)
Software verification plan (IEC 61508-3, 7.9.2)
SW system design (IEC 61508-3, 7.4.5)
Software coding standard (IEC 61508-3, 7.4.4)
Software tool qualification (IEC 61508-3, 7.4.4)
Software requirements specification (IEC 61508-3, 7.2.2)
SW verification test reports (IEC 61508-3, 7.4.7/8, 7.5.2)
Software architecture design (IEC 61508-3, 7.4.3)
SW verification test spec. (IEC 61508-3, 7.4.7/8, 7.5.2)
Source code (IEC 61508-3, 7.4.6)
Validated software (IEC 61508-3, 7.7.2)
Safety artifacts for review by approval agency

Figure 3. Comparing software artefacts of ISO 26262 and IEC 61508.

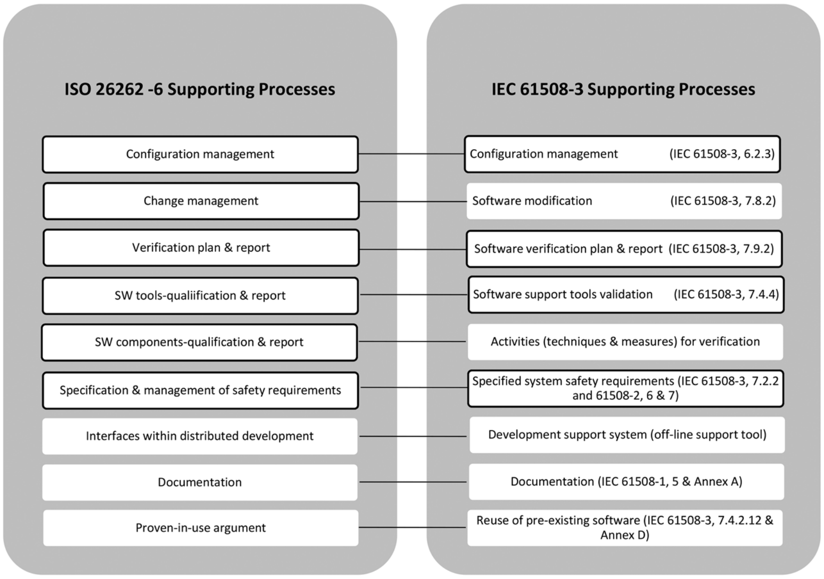
ISO 26262 -6 Supporting Processes
Configuration management
Change management
Verification plan & report
SW tools-qualiification & report
SW components-qualification & report
Specification & management of safety requirements
Interfaces within distributed development
Documentation
Proven-in-use argument
IEC 61508-3 Supporting Processes
Configuration management (IEC 61508-3, 6.2.3)
Software modification (IEC 61508-3, 7.8.2)
Software verification plan & report (IEC 61508-3, 7.9.2)
Software support tools validation (IEC 61508-3, 7.4.4)
Activities (techniques & measures) for verification
Specified system safety requirements (IEC 61508-3, 7.2.2 and 61508-2, 6 & 7)
Development support system (off-line support tool)
Documentation (IEC 61508-1, 5 & Annex A)
Reuse of pre-existing software (IEC 61508-3, 7.4.2.12 & Annex D)

Figure 4. Comparing supporting processes of ISO 26262-6 and IEC 61508-3.

Recommended). The rankings are associated with target software safety levels (ASIL as defined by ISO 26262 and SIL as defined IEC 61508) and are expected to refine the mapping process.

Table 3. Comparing techniques for software safety requirements management.

Techniques for software safety requirements management									
Table 2 and Table 5 of ISO 26262-6	ASIL				SIL				Table A.1 of IEC 61508-3
Notations for software design	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Software safety requirements specification**
Semi-formal notations	R	R	HR	HR	R	R	HR	HR	Semi-formal methods
Formal notations	R	R	R	R	–	R	R	HR	Formal methods
Table 10 of ISO 26262-6	**ASIL**				**SIL**				
Methods for verification of software integration	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	
Requirements-based test	HR	HR	HR	HR	R	R	HR	HR	Forward traceability between the system safety requirements and the software safety requirements
					R	R	HR	HR	Backward traceability between the safety requirements and the perceived safety needs
Back-to-back comparison test between model and code, if applicable	R	R	HR	HR	R	R	HR	HR	Computer-aided specification tools to support appropriate techniques/ measures above.

Let us consider the table of Verhulst et al. (2013) in Table 7 for the purpose of this discussion, since it is based on the most robust scientific argument among all the literature reviewed in a recent paper (Okoh & Myklebust, 2024) about the mapping of system safety levels between ISO 26262 and IEC 61508. Based on this, it is observed to a significant extent in Tables 3–6 that there exists a correlation between the recommendation of the techniques or measures mapped between ASIL D and SIL 3, between ASIL C and SIL 2, and between ASIL B and SIL 1. Thus, the mapping scheme of Verhulst et al. (2013) is validated for software.

Furthermore, after having established a robust mapping between ISO 16262 and IEC 61508 based on similarities in techniques/measures and their recommendation (i.e., a measure of importance described in Table 8) in relation to the desired safety levels, it is pertinent to consider the level of development effort expended (also known as rigour) on the techniques/measures by which the target safety levels (or systematic safety integrity) can be asserted. Systematic safety integrity is also known as systematic capability in IEC 61508.

Table 4. Comparing design and coding requirements.

Design and coding techniques and measures									
Table 1 of ISO 26262-6	**ASIL**				**SIL**				**Table A.4 of IEC 61508-3**
Modelling and coding guidelines topics	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Detailed Design**
Enforcement of low complexity	HR	HR	HR	HR	HR	HR	HR	HR	Structured programming
Use of defensive implementation technique	O	R	HR	HR	–	R	HR	HR	Defensive programming
Use of well-trusted design principles	R	R	R	HR	HR	HR	HR	HR	Modular approach
Use of unambiguous graphical representation	R	HR	HR	HR	R	R	HR	HR	Computer-aided design tools
Use of naming conventions	HR	HR	HR	HR	R	HR	HR	HR	Design and coding standards
Concurrency aspects	R	R	R	R	R	HR	HR	HR	Use of trusted/verified software elements (if available)
Table 3 of ISO 26262-6	**ASIL**				**SIL**				**Table B.1 of IEC 61508-3**
Principles for software architectural design	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Design and coding standards**
Restricted size and complexity of software components	HR	HR	HR	HR	HR	HR	HR	HR	Use of coding standard to reduce likelihood of errors
Restricted use of interrupts	R	R	R	HR	R	R	HR	HR	Limited use of interrupts
Table 6 of ISO 26262-6	**ASIL**				**SIL**				
Design principles for software unit design and implementation	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	
No dynamic objects or variables, or else online test during their creation	R	HR	HR	HR	R	HR	HR	HR	No dynamic objects
Restricted use of pointers	R	HR	HR	HR	–	R	HR	HR	Limited use of pointers
No implicit type conversions	R	HR	HR	HR	R	HR	HR	HR	No automatic type conversion
No hidden data flow or control flow	R	HR	HR	HR	R	HR	HR	HR	No unstructured control flow in programs in higher level languages
No recursions	R	R	HR	HR	–	R	HR	HR	Limited use of recursion

IEC 61508 sets out two criteria for establishing systematic safety integrity for software: 1) Selecting the techniques or measures that correspond to a given SIL, and 2) Demonstrating the rigour (described in Table 8) for assuring the fulfilment of the properties for software systematic safety integrity in the selected techniques or measures. The rigour, which is

Table 5. Comparing error detection and handling requirements.

Software error detection and handling techniques and measures									
ISO 26262	**ASIL**				**SIL**				**Table B.3 of IEC 61508-3**
Mechanisms for error detection at architectural level	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Functional and black-box testing**
Range check of input and output data	HR	HR	HR	HR	R	HR	HR	HR	Equivalence classes and input partition testing, including boundary value analysis.
Table 4 **of ISO 26262-6**	**ASIL**				**SIL**				
Methods for the verification of the software architectural design	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	
Simulation of dynamic behaviour of the design	R	R	R	HR	R	R	R	R	Process simulation
Prototype generation	0	0	R	HR	–	–	R	R	Prototyping/animation
Formal verification	O	O	R	R	R	R	HR	HR	Test case execution from model-based test case generation.
Data flow analysis	R	R	HR	HR	–	–	R	R	Test case execution from cause- consequence diagrams
ISO 26262	**ASIL**				**SIL**				**Table A.2 of IEC 61508-3**
Error handling mechanisms at architectural level	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Software architectural design**
Static recovery mechanisms	R	R	R	R	R	R	–	NR	Backward recovery
Graceful degradation	R	R	HR	HR	R	R	HR	HR	Graceful degradation
Independent parallel redundancy	O	O	R	HR	–	–	R	HR	Functionally diverse redundancy, implementing different SW safety requirements specification.
Correcting codes for data	R	R	R	R	R	R	R	HR	Error-detecting codes
Table 7 **of ISO 26262-6**	**ASIL**				**SIL**				**Table A.5 of IEC 61508-3**
Methods for software unit verification	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Software module testing & integration**
Requirements-based test	HR	HR	HR	HR	R	R	HR	HR	Forward traceability between the software design specification and the module and integration test specifications
Interface test	HR	HR	HR	HR	R	R	HR	HR	Interface testing
Resource usage evaluation	R	R	R	HR	R	R	HR	HR	Performance testing
Back-to-back comparison test between model and code, if applicable	R	R	HR	HR	R	HR	HR	HR	Dynamic analysis and testing
Semi-formal verification	R	R	HR	HR	R	R	HR	HR	Model-based testing
Formal verification	0	0	R	R	–	–	R	R	Formal verification
									Table B.8 of IEC 61508-3
									Static analysis
Walk-through	HR	R	O	O	R	R	R	R	Walk-through (software)
Inspection	R	HR	HR	HR	R	R	HR	HR	Formal inspections, including specific criteria
Control flow analysis	R	R	HR	HR	R	HR	HR	HR	Control flow analysis
Data flow analysis	R	R	HR	HR	R	HR	HR	HR	Data flow analysis
Static code analysis	HR	HR	HR	HR	HR	HR	HR	HR	Design review
Static analyses based on abstract interpretation	R	R	R	R	R	R	R	HR	Static analysis of run-time error behaviour
Table 8 **of ISO 26262-6**	**ASIL**				**SIL**				
Methods for deriving test cases for software unit testing	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	
Analysis of boundary values	R	HR	HR	HR	R	R	HR	HR	Boundary value analysis
Error guessing based on knowledge or experience	R	R	R	R	R	R	R	R	Error guessing

Table 6. Comparing requirements for software tools.

Software tools and programming language techniques and measures									
Table 4 of **ISO 26262-8**	**ASIL**				**SIL**				**Table A.3 of IEC 61508-3**
Qualification of software tools classified TCL3	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Support tools and programming language**
Increased confidence from use in accordance with 11.4.7	HR	HR	R	R	HR	HR	HR	HR	Tools and translators increased confidence from use.
Validation of the software tool in accordance with 11.4.9	R	R	HR	HR	R	HR	HR	HR	Certified tools and certified translators.
Table 5 of **ISO 26262-8**	**ASIL**				**SIL**				**Table A.3 of IEC 61508-3**
Qualification of software tools classified TCL2	**A**	**B**	**C**	**D**	**1**	**2**	**3**	**4**	**Support tools and programming language**
Increased confidence from use in accordance with 11.4.7	HR	HR	HR	R	HR	HR	HR	HR	Tools and translators increased confidence from use.
Validation of the software tool in accordance with 11.4.9	R	R	R	HR	R	HR	HR	HR	Certified tools and certified translators.

Table 7. Mapping of the safety levels of different domains (Verhulst et al., 2013).

Domain	Domain-specific Safety Levels				
General (e.g., IEC 61508)	(SIL 0)	SIL 1	SIL 2	SIL 3	SIL 4
Automotive (e.g., ISO 26262)	ASIL A	ASIL B	ASIL C	ASIL D	
Aviation (e.g., DO178C)	DAL E	DAL D	DAL C	DAL B	DAL A
Railway (e.g., EN 50128)	SIL 0	SIL 1	SIL 2	SIL 3	SIL 4

described in Table 8, is ranked as follows: R1 (lowest - for SIL 1/SIL 2 applications), R2 (medium – for SIL 3 application) and R3 (highest – for SIL 4 application). The properties for software systematic safety integrity are: 1) Completeness with respect to the safety needs, 2) Correctness with respect to the safety needs, 3) Freedom from intrinsic specification faults, including freedom from ambiguity, 4) Understandability of safety requirements, 5) Freedom from adverse interference of non-safety functions with

Table 8. Recommendations of techniques/measures and rigour of properties for software systematic capability (IEC 61508, 2010).

Recommendation	Description	
HR	The technique or measure is highly recommended for this safety integrity level. If this technique or measure is not used then the rationale behind not using it should be detailed with reference to Annexe C during the safety planning and agreed with the assessor.	
R	The technique or measure is recommended for this safety integrity level as a lower recommendation to a HR recommendation.	
–	The technique or measure has no recommendation for or against being used.	
NR	The technique or measure is positively not recommended for this safety integrity level. If this technique or measure is used then the rationale behind using it should be detailed with reference to Annexe C during the safety planning and agreed with the assessor.	
Rigour	**Description**	**Target SIL**
R1	Without objective acceptance criteria, or with limited objective acceptance criteria. E.g., black-box testing based on judgement, field trials.	SIL 1 and SIL 2
R2 (If available)	With objective acceptance criteria that can give a high level of confidence that the required property is achieved (exceptions to be identified & justified); e.g., test or analysis techniques with coverage metrics, coverage of checklists.	SIL 3
R3 (If available)	With objective, systematic reasoning that the required property is achieved. E.g. formal proof, demonstrated adherence to architectural constraints that guarantee the property.	SIL 4
–	This technique is not relevant to this property.	

the safety needs, and 6) Capability of providing a basis for verification and validation (IEC 61508, 2010).

In Table 9, techniques stipulated by ISO 26262 which map to IEC 61508 (using Table 3 as an example) are treated with the rigour of the properties stipulated in Section C.2 of Annexe C of IEC 61508-3 for software systematic safety integrity.

5. Discussion and recommendations

The objective of this paper is to qualify for reuse with respect to the safety integrity levels of IEC 61508, software that was originally developed according to ISO 26262. This has been realised in Section 4 and this paper summarises and recommends the systematic approach as follows:

1. Map techniques/measures from ISO 26262-6 to IEC 61508-3 in terms of system safety levels (i.e. ASIL and SIL) based on the table of Verhulst et al. (2013).

Table 9. Properties for systematic safety integrity – Software safety requirements specification.

			Rigours of Properties for software systematic safety integrity							
Technique or measure in ISO 26262	Target ASIL	Recom-mendation for ASIL	Complete-ness wrt the safety needs	Correct-ness wrt the safety needs	Freedom from intrinsic specifica-tion faults, including freedom from ambiguity	Understand-ability of safety requirements	Freedom from adverse interfere-ence of non-safety function with the safety needs	Capability of providing a basis for V &V	Target SIL of similar technique or measure in IEC 61508	Recom-mendation for SIL
Semi-formal notations	ASIL B	R	R1:…	R1:…	R1:…	R1:…	–	R2:…	SIL 1	R
	ASIL C	HR	R1:…	R2:…	R2:…	R1:…	–	R2:…	SIL 2	R
	ASIL D	HR	R1:…	R2:…	R2:…	R2:…	–	R2:…	SIL 3	HR
Formal notations	ASIL B	R	R1:…	R1:…	R1:…	–	–	R3:…	SIL 1	–
	ASIL C	R	R1:…	R1:…	R1:…	–	–	R3:…	SIL 2	R
	ASIL D	R	R1:…	R2:…	R2:…	–	–	R3:…	SIL 3	R
Requirements-based test	ASIL B	HR	–	R1:…	–	R1:…	R1:…	R1:…	SIL 1	R
	ASIL C	HR	–	R1:…	–	R1:…	R1:…	R1:…	SIL 2	R
	ASIL D	HR	–	R1:…	–	R1:…	R1:…	R1:…	SIL 3	HR
Back-to-back comparison test between model and code, if applicable	ASIL B	R	R1:…	R1:…	R2:…	R1:…	R1:…	R1:…	SIL 1	R
	ASIL C	R	R1:…	R2:…	R2:…	R1:…	R1:…	R1:…	SIL 2	R
	ASIL D	HR	R2:…	R2:…	R2:…	R1:…	R1:…	R1:…	SIL 3	HR

2. Apply on the applicable techniques/measures of ISO 26262, the properties and rigour for assuring software systematic safety integrity provided in Section C.2 of Annexe C of IEC 61508-3.
3. Prioritise the applicable techniques/measures of ISO 26262 according to the recommendation of IEC 61508.

Overall, the software attributes of ISO 26262 and IEC 61508 can be conveniently mapped between each other such that safety levels are redefined in reuse as shown in this paper. This is consistent with Miller (2020) who identified a general consistency between them. It is pertinent to note that ISO 26262 adopts the basic principles of IEC 61508 and has some automotive-industry-specific requirements and guidance in addition, thus making a transition from ISO 26262 to IEC 61508 easier than vice versa. A joint use of ISO 26262 and IEC 61508 is also recommended in certain cases where the latter is silent on specific details for meeting certain requirements, e.g., in relation to tool qualification. Unlike ISO 26262, IEC 61508-3, 7.4.4.5 states that "Where such failure mechanisms are identified, appropriate mitigation measures shall be taken", without providing details on what these measures are, thus leaving more room for interpretation (Pitchford, 2022; Wiltgen, 2020).

6. Conclusion

This paper has realised a framework for mapping ISO-26262-based software to IEC 61508 in relation to safety levels. This is expected to guide the reuse of safety-related resources of the automotive industry in the generic industry without compromising safety. The paper built on a review of literature and standards. It is intended to give engineers, standard organisations and certification bodies more insight into inter-domain cooperation on software between the automotive (based on ISO 26262) and the generic industry (based on IEC 61508) in order to collectively catch up with the pace of technological development whereby one industry may already be ahead of the other in terms of alignment with the state-of-the-art.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Funding

This work has been supported by The Norwegian Research Council, Project name "Integrated Service & Safety Platform," project number: 309406.

Notes on contributors

Peter Okoh holds a PhD in Reliability, Availability, Maintainability and Safety (RAMS). He studied at the Department of Mechanical and Industrial Engineering, at The Norwegian University of Science and Technology, Trondheim, Norway.

Thor Myklebust is a senior researcher at SINTEF Digital and a member of the IEC 61508 maintenance committee.

ORCID

Peter Okoh http://orcid.org/0000-0001-5086-7989

References

Crots, K., Skentzos, P., & Bartz, D. (2014). A Comparative analysis of aviation and land vehicle software development standards. In *Proceedings of the 2014 Ground Vehicle Systems Engineering and Technology Symposium (GVSETS)*.

Gerlack, M., Hilbrich, R., & Weissleder, S. (2011). *Can cars fly? From avionics to automotive: Comparability of domain specific safety standards* [Paper presentation]. Embedded World Exhibition and Conference 2011, Nürnberg.

IEC 61508. (2010). *Functional safety of electrical/electronic/programmable electronic safety-related systems*. International Standard of The International Electrotechnical Commission (IEC).

ISO 26262. (2018). *Road vehicles – Functional safety*. International Standard of The International Organization for Standardization (ISO).

Kritzinger, D. (2017). 9 – Development assurance. In D. Kritzinger (Ed.), *Aircraft system safety* (pp. 193–324). Woodhead Publishing. ISBN 9780081008898. https://doi.org/10.1016/B978-0-08-100889-8.00009-X

Machrouh, J., Blanquart, J., Baufreton, P., Boulanger, J., Delseny, H., Gassino, J., Ladier, G., Ledinot, E., Leeman, M., Astruc, J., Quéré, P., Ricque, B., & Deleuze, G. (2012). A cross-domain comparison of software development assurance standards. https://api.semanticscholar.org/CorpusID:37197435

Miller, J. (2020). *Automotive system safety: Critical considerations for engineering and effective management*. John Wiley & Sons Ltd.

Okoh, P. (2019). Integrated logistics support and asset management (ILSAM). *Infrastructure Asset Management*, *6*(4), 245–257. https://doi.org/10.1680/jinam.17.00026

Okoh, P., Dong, H. S., & Liu, Y. (2022). Cross-acceptance of fire safety systems based on SIL equivalence in relation to IEC 61508 and EN 50129. *Safety and Reliability*, *41*(2), 103–120. https://doi.org/10.1080/09617353.2022.2107255

Okoh, P., & Myklebust, T. (2024). Mapping to IEC 61508 the hardware safety integrity of elements developed to ISO 26262. *Safety and Reliability*. Advance online publication. https://doi.org/10.1080/09617353.2024.2343959

Pitchford, M. (2022). Beyond TÜV: A path to high-criticality tool qualification. Retrieved from https://embeddedcomputing.com/technology/security/iec-iso-other-standards/beyond-tuv-a-path-to-high-criticality-tool-qualification

Ruiz, A., Juez, G., Espinoza, H., Luis De La Vara, J., & Larrucea, X. (2017). Reuse of safety certification artefacts across standards and domains: A systematic approach. *Reliability Engineering & System Safety*, *158*, 153–171. https://doi.org/10.1016/j.ress.2016.08.017

Verhulst, E., Vara, J., Sputh, B., & Florio, V. (2013). ARRL: A criterion for composable safety and systems engineering. https://api.semanticscholar.org/CorpusID:101124

Wiltgen, J. (2020). Leveraging ISO 26262 tool certification in IEC 61508. Retrieved from https://blogs.sw.siemens.com/verificationhorizons/2020/12/17/leveraging-iso-26262-tool-certification-in-iec-61508