



DPC: 25 / 30507626 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 21 February 2025

Origin: International

Latest date for receipt of comments: 16 April 2025

Project No. 2025/00135

Responsible committee: GEL/65/1 Systems aspects

Interested committees:

Title: BS EN IEC 61508-3 Functional safety of electrical/electronic/programmable electronic safety-related systems

Part 3: Software requirements

Please notify the secretary if you are aware of any keywords that might assist in classifying or identifying the standard or if the content of this standard

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

**WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD.
THIS DRAFT IS NOT CURRENT BEYOND 16 April 2025**

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <http://standardsdevelopment.bsigroup.com>

Responsible Committee Secretary: Ms Charmaine Mukherjee (BSI)

Direct tel:

E-mail: charmaine.mukherjee@bsigroup.com

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/Figure/ Table/Note	Type of comment	Commend (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	



65A/1169/CDV

COMMITTEE DRAFT FOR VOTE (CDV)

PROJECT NUMBER:

IEC 61508-3 ED3

DATE OF CIRCULATION:

2025-02-21

CLOSING DATE FOR VOTING:

2025-05-16

SUPERSEDES DOCUMENTS:

65A/1058A/CD, 65A/1077A/CC

IEC SC 65A : SYSTEM ASPECTS

SECRETARIAT:

United Kingdom

SECRETARY:

Ms Stephanie Lavy

OF INTEREST TO THE FOLLOWING COMMITTEES:

TC 8, TC 9, TC 22, TC 31, TC 44, TC 45, TC 56, TC 61, TC 62, TC 65, SC 65B, SC 65C, SC 65E, TC 66, TC 77, TC 80, TC 108, SyC AAL, SyC SM, SC 41

HORIZONTAL FUNCTION(S):

ASPECTS CONCERNED:

Safety

☒ SUBMITTED FOR CENELEC PARALLEL VOTING☐ NOT SUBMITTED FOR CENELEC PARALLEL VOTING

Attention IEC-CENELEC parallel voting

The attention of IEC National Committees, members of CENELEC, is drawn to the fact that this Committee Draft for Vote (CDV) is submitted for parallel voting.

The CENELEC members are invited to vote through the CENELEC online voting system.

This document is still under study and subject to change. It should not be used for reference purposes.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Recipients of this document are invited to submit, with their comments, notification of any relevant "In Some Countries" clauses to be included should this proposal proceed. Recipients are reminded that the CDV stage is the final stage for submitting ISC clauses. (SEE [AC/22/2007](#) OR [NEW GUIDANCE DOC](#)).

TITLE:

Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements

PROPOSED STABILITY DATE: 2028

NOTE FROM TC/SC OFFICERS:

Copyright © 2025 International Electrotechnical Commission, IEC. All rights reserved. It is permitted to copy and to print out the content for the sole purpose of preparing National Committee positions. You may not copy or "mirror" the file or printed version of the document, or any part of it, for any other purpose without permission in writing from IEC.

CONTENTS

1		
2	FOREWORD	7
3	INTRODUCTION	10
4	1 Scope	12
5	2 Normative references	16
6	3 Terms, definitions, symbols and abbreviations	16
7	4 Conformance to this document	16
8	5 Documentation	16
9	6 Requirements for management of safety-related software	16
10	6.1 Objectives	16
11	6.2 Requirements	16
12	7 Software safety lifecycle requirements	17
13	7.1 General	17
14	7.1.1 Objective	17
15	7.1.2 Requirements	17
16	7.2 Software safety requirements specification	24
17	7.2.1 Objectives	24
18	7.2.2 Requirements	24
19	7.3 Validation plan for software aspects of system safety	28
20	7.3.1 Objective	28
21	7.3.2 Requirements	28
22	7.4 Software design and development	29
23	7.4.1 Objectives	29
24	7.4.2 General requirements	29
25	7.4.3 Requirements for software architecture design	35
26	7.4.4 Requirements for support tools, including programming languages	37
27	7.4.5 Requirements for detailed design and development – software system	
28	design	38
29	7.4.6 Requirements for code implementation	39
30	7.4.7 Requirements for software module testing	39
31	7.4.8 Requirements for software integration testing	40
32	7.5 Programmable electronics integration (hardware and software)	41
33	7.5.1 Objectives	41
34	7.5.2 Requirements	41
35	7.6 Software operation and modification procedures	42
36	7.6.1 Objective	42
37	7.6.2 Requirements	42
38	7.7 Software aspects of system safety validation	43
39	7.7.1 Objective	43
40	7.7.2 Requirements	43
41	7.8 Software modification	44
42	7.8.1 Objective	44
43	7.8.2 Requirements	44
44	7.9 Software verification	46
45	7.9.1 Objective	46
46	7.9.2 Requirements	46

47	8	Functional safety assessment.....	50
48	9	Bibliography	51
49		Annex A (normative) Guide to the selection of techniques and measures.....	52
50		Annex B (informative) Detailed tables	61
51		Annex C (informative) Properties for systematic capability of software elements.....	65
52	C.1	Introduction.....	65
53	C.1.1	Structure of Annex C, relating to Annexes A and B	65
54	C.1.2	Method of use – 1	67
55	C.1.3	Method of use – 2	68
56	C.2	Properties for systematic capability.....	70
57	C.3	Properties for systematic capability – Detailed tables	96
58		Annex D (normative) Safety manual for compliant items – additional requirements for	
59		software elements	110
60	D.1	Purpose of the safety manual.....	110
61	D.2	Contents of the safety manual for a software element	110
62	D.3	Justification of claims in the safety manual for compliant items	112
63		Annex E (informative) Relationships between IEC 61508-2 and this document.....	113
64		Annex F (informative) Techniques for achieving non-interference between software	
65		elements on a single computer	116
66	F.1	Introduction.....	116
67	F.2	Domains of behaviour	116
68	F.3	Causal factor analysis.....	117
69	F.4	Achieving spatial independence	117
70	F.5	Achieving temporal independence.....	118
71	F.6	Requirements for supporting software	119
72	F.7	Independence of software modules – programming language aspects	120
73		Annex G (informative) Guidance for systems configured by application data	125
74	G.1	Introduction.....	125
75	G.2	Aspects influencing lifecycle requirements	125
76	G.2.1	Complexity of E/E/PE systems.....	125
77	G.2.2	Rationale for the lifecycle requirements	126
78	G.2.3	Complexity in software.....	127
79	G.2.4	Programming language characteristics and safety	128
80	G.2.5	Role of the tools	130
81	G.3	Guidance for tailoring the lifecycle	130
82	G.3.1	Recommendations for tailoring the lifecycle	130
83	G.3.2	Tailoring principles	130
84	G.3.3	Fixed functionalities and configuration	131
85	G.3.4	Fixed functionalities and programming.....	131
86	G.3.5	Limited functionalities and configuration	132
87	G.3.6	Limited functionalities and programming	133
88	G.3.7	Open functionalities and configuration	133
89	G.3.8	Open functionalities and programming.....	133
90		Annex H (normative) Confidence in the usage of Software Off-line Support Tools.....	135
91	H.1	Scope and conventions	135
92	H.2	Software off-line support tool usage confidence approach overview	135
93	H.2.1	Software off-line support tool usage confidence objectives	135
94	H.2.2	Software off-line support tool confidence approach.....	136

95	H.2.3	Software off-line support tool usage confidence measures overview	137
96	H.3	Measures concerning software off-line support tool usage	139
97	H.3.1	Software off-line support tool usage planning and qualification planning	139
98	H.3.2	Software off-line support tool classification in the usage context.....	140
99	H.3.3	Software off-line support tool user documentation.....	142
100	H.3.4	Software off-line support tool integration and assessment in the usage	
101		environment.....	143
102	H.3.5	Software off-line support tool configuration management in the tool user	
103		context	143
104	H.3.6	Software off-line support tool operation	144
105	H.4	Measures concerning software off-line support tool development including	
106		verification	144
107	H.4.1	Software off-line support tool development with avoidance of systematic	
108		faults	144
109	H.4.2	Confidence from software off-line support tool usage history	145
110	H.4.3	Software off-line support tool documentation and problem management	146
111	H.5	Required evidence for software off-line support tool usage confidence	147
112	Annex I (informative)	Model Based Software Development and Verification Guidelines	149
113	I.1	Objectives and scope of this annex.....	149
114	I.2	Rationale for this annex	149
115	I.3	Approach and relationship with the body of this document	150
116	I.4	Documentation.....	150
117	I.5	Safety lifecycle planning	150
118	I.6	Techniques and measures	152
119	I.7	Software safety requirements specification	153
120	I.8	Software architecture design.....	154
121	I.9	Requirements for software off-line support tools, including programming	
122		languages.....	155
123	I.10	Detailed design and development	155
124	I.11	Code Implementation	155
125	I.12	Software module testing.....	156
126	I.13	Software integration testing	157
127	I.14	Model Verification	157
128	I.15	Model-Based verification.....	158
129	Annex J (Informative)	Data Driven Systems	160
130	J.1	General.....	160
131	J.2	Systems with Minimal Configurable Parameters	161
132	J.3	Systems with Comprehensive Configurability	161
133	J.4	Role of the tools.....	162
134			
135	Figure 1 – Overall framework of the IEC 61508 series		14
136	Figure 2 – Overall E/E/PE system safety lifecycle		15
137	Figure 3 – E/E/PE system safety lifecycle (in realisation phase).....		19
138	Figure 4 – Software safety lifecycle (in realisation phase).....		19
139	Figure 5 – Relationship and scope for IEC 61508-2 and this document (see also		
140	Annex E).....		20
141	Figure 6 – Systematic capability for software and the development lifecycle (the V-		
142	model)		20
143	Figure H.1 – Tool Confidence Principle		137

144	Figure H.2 – Data flow overview of software off-line support tool usage confidence	
145	measures for an on-demand software off-line support tool	138
146	Figure H.3 – Data flow overview of software off-line support tool usage confidence	
147	measures for a COTS software off-line support tool	139
148		
149		
150	Table 1 – Software safety lifecycle – overview (1 of 4)	21
151	Table A.1 – Software safety requirements specification	53
152	Table A.2 – Software design and development – software architecture design	53
153	Table A.3 – Software design and development – programming language	55
154	Table A.4 – Software design and development – detailed design	55
155	Table A.5 – Software design and development – software module testing and	
156	integration	56
157	Table A.6 – Programmable electronics integration (hardware and software).....	57
158	Table A.7 – Software aspects of system safety validation	57
159	Table A.8 – Modification	57
160	Table A.9 – Software verification	59
161	Table A.10 – Not Used : Replaced by additional requirements in Table A.2	59
162	Table A.11 – Software life cycle through lifecycle activities	60
163	Table B.1 – Design and coding standards	61
164	Table B.2 – Dynamic analysis and testing	61
165	Table B.3 – Functional and black-box testing	62
166	Table B.4 – Not Used : Replaced by additional requirements in Table A.2	62
167	Table B.5 – Modelling	63
168	Table B.6 – Performance testing	63
169	Table B.7 – Not Used	63
170	Table B.8 – Static analysis	64
171	Table B.9 – Modular approach	64
172	Table C.1.1 – Extract of Table A.1 for illustration	65
173	Table C.1.2 – Extract of Table C.1 for illustration	66
174	Table C.1.3 – Extract of Table C.1 for illustration	66
175	Table C.1.4 – Rigour of techniques	67
176	Table C.1.5 – Use of rigour ranking	67
177	Table C.1.2.1 Method of Use 1 : link between rigour and CS.....	68
178	Table C.1 – Properties for systematic capability – Software safety requirements	
179	specification.....	70
180	Table C.2 – Properties for systematic capability – Software design and development –	
181	software Architecture Design	73
182	Table C.3 – Properties for systematic capability – Software design and development –	
183	programming language	84
184	Table C.4 – Properties for systematic capability – Software design and development –	
185	detailed design (includes software system design, software module design and	
186	coding)	85
187	Table C.5 – Properties for systematic capability – Software design and development –	
188	software module testing and integration	89

189	Table C.6 – Properties for systematic capability – Programmable electronics	
190	integration (hardware and software).....	91
191	Table C.7 – Properties for systematic capability – Software aspects of system safety	
192	validation	92
193	Table C.8 – Properties for systematic capability – Software modification	93
194	Table C.9 – Properties for systematic capability – Software verification	95
195	Table C.11 – Detailed properties – Design and coding standards.....	96
196	Table C.12 – Detailed properties – Dynamic analysis and testing.....	98
197	Table C.13 – Detailed properties – Functional and black-box testing.....	100
198	Table C.14 – Deleted – Replaced by additional information in Table C.2.....	101
199	Table C.15 – Detailed properties – Modelling.....	101
200	Table C.16 – Detailed properties – Performance testing.....	102
201	Table C.17 – Not Used.....	103
202	Table C.18 – Properties for systematic capability – Static analysis	103
203	Table C.19 – Detailed properties – Modular approach	104
204	Table C.20 – Properties for systematic capability - through lifecycle aspects	106
205	Technique/Measure	106
206	Table E.1 – Categories of IEC 61508-2 requirements.....	113
207	Table E.2 – Requirements of IEC 61508-2 for software and their typical relevance to	
208	certain types of software	113
209	Table F.1 – Types of module coupling	121
210	Table G.1 – Variability in complexity of E/E/PE systems	126
211	Table H.1 – TD Levels	141
212	Table H.2 –TIC/FSI/TD/TUS Balancing Rules	142
213	Table H.3 – Required Tool Usage Confidence Evidence per TUS.....	148
214	Table I.1 – Comparison of Separation vs Merging Development Layers	151
215		
216		

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**FUNCTIONAL SAFETY OF ELECTRICAL/ELECTRONIC/
PROGRAMMABLE ELECTRONIC SAFETY-RELATED SYSTEMS –****Part 3: Software requirements****FOREWORD**

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) IEC draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). IEC takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at <https://patents.iec.ch> [and/or] www.iso.org/patents. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 61508-3 has been prepared by subcommittee SC65A: SYSTEM ASPECTS, of IEC technical committee TC65: INDUSTRIAL PROCESS MEASUREMENT, CONTROL AND AUTOMATION. It is an International Standard.

This third edition cancels and replaces the second edition published in 2010. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

This document has been upgraded to the 2024 version of the ISO/IEC Directives; this does introduce a significant number of editorial changes and Clause renumbering throughout. In addition, the following technical changes have been made:

- a) All requirements on use of artificial Intelligence, security and human factors are limited to the references in Scope to external guidance
- b) The requirements of IEC TS 61508-3-1 have been incorporated into Clause 7 and extended to address requirements to achieve Systematic Capability SC3 and SC4
- c) The interfaces to other Parts of IEC 61508 have been clarified and duplication of requirements in other Parts has been removed
- d) The requirement to undertake software functional failure analysis has been transferred from the requirements for functional safety assessment to the requirements on architectural design and renamed software failure analysis; the differentiation between functional safety audit, functional safety assessment and software failure analysis has been clarified in IEC 61508-7
- e) The differentiation between the required Safety Integrity Level (SIL) and the achieved Systematic Capability (SC) has been clarified
- f) The requirements for formal methods has been clarified throughout all software lifecycle phases, the requirements of TS 61508-3-2 have been referenced in this document and IEC 61508-7 and the concept of semi-formal methods has been re-instated to the status in Edition 2 of this document; the meaning of 'unambiguously defined' methods has been clarified
- g) Rules for synthesis of software elements has been clarified; detailed requirements on software used in diagnostics is now addressed in IEC 61508-2
- h) The requirements on software off-line support tools have been refined
- i) The guidance on data driven systems has been provided in a separate Annex from the guidance on data configuration and limited variability programming; the guidance on limited variability programming has been updated to be consistent with IEC 61131-1.
- j) The requirements in Annex A on modification have been re-instated to the status in Edition 2 and clarified
- k) An additional table has been added to define requirements on management activities which occur throughout the entire software lifecycle; this table includes requirements on traceability, which have been removed from individual phase requirements
- l) Requirements on regression testing have been clarified
- m) The use of the terminology 'verification' and 'validation' has been brought in line with the definitions in IEC 61508-4; much supplementary information added at Ed 3 CD has been transferred to IEC 61508-7 as informative guidance, or else deleted
- n) The requirements on the use of object oriented design and development have been brought in line with practical applications; further detail is being developed in TR 61508-3-3
- o) Various minor editorial errors have been corrected.

NOTE In order to avoid the need for extensive editing to existing compliance tools, the methods and techniques specified in Annex A and Annex B of this document have retained their original reference ID. Where methods and techniques have been deleted the phrase 'Not Used' has been entered. Where methods and techniques have been added an additional row with additional reference ID have been added. As far as possible this also applies to Clauses in the body of the document, but some renumbering has been inevitable where information has been added.

This edition has been subject to a thorough review and incorporates many comments received at the various revision stages.

It has the status of a basic safety publication according to IEC Guide 104.

The text of this document is based on the following documents:

Draft	Report on voting
XX/XX/FDIS	XX/XX/RVD

318
319 Full information on the voting for its approval can be found in the report on voting indicated in
320 the above table.

321 The language used for the development of this document is English.

322 This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in
323 accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available
324 at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are
325 described in greater detail at www.iec.ch/publications.

326 A list of all parts of the IEC 61508 series, published under the general title *Functional safety of*
327 *electrical / electronic / programmable electronic safety-related systems*, can be found on the
328 IEC website.

329 The committee has decided that the contents of this document will remain unchanged until the
330 stability date indicated on the IEC website under webstore.iec.ch in the data related to the
331 specific document. At this date, the document will be

- 332 • replaced by a revised edition, or
333 • amended.

334

INTRODUCTION

Systems comprised of electrical and/or electronic elements have been used for many years to perform safety functions in most application sectors. Computer-based systems (generically referred to as programmable electronic systems) are being used in all application sectors to perform non-safety functions and, increasingly, to perform safety functions. If computer system technology is to be effectively and safely exploited, it is essential that those responsible for making decisions have sufficient guidance on the safety aspects on which to make these decisions.

This document sets out a generic approach for all safety lifecycle activities for systems comprised of electrical and/or electronic and/or programmable electronic (E/E/PE) elements that are used to perform safety functions. This unified approach has been adopted in order that a rational and consistent technical policy be developed for all electrically-based safety-related systems. A major objective is to facilitate the development of product and application sector international standards based on the IEC 61508 series.

NOTE 1 Examples of product and application sector international standards based on the IEC 61508 series are given in the bibliography.

In most situations, safety is achieved by a number of systems which rely on many technologies (for example mechanical, hydraulic, pneumatic, electrical, electronic, programmable electronic). Any safety strategy considers not only all the elements within an individual system (for example sensors, controlling devices and actuators) and humans as part of the system, as part of the realization process, operation, and management of verification/decision making), but also all the safety-related systems making up the total combination of safety-related systems. Therefore, while this document is concerned with E/E/PE safety-related systems, it can also provide a framework within which safety-related systems based on other technologies can be considered.

It is recognized that there is a great variety of applications using E/E/PE safety-related systems in a variety of application sectors and covering a wide range of complexity, hazard and risk potentials. In any particular application, the required safety measures will be dependent on many factors specific to the application. This document, by being generic, will enable such measures to be formulated in future product and application sector international standards and in revisions of those that already exist.

This document

- considers all relevant overall, E/E/PE system and software safety lifecycle phases (for example, from initial concept, through design, implementation, operation and maintenance to decommissioning) when E/E/PE systems are used to perform safety functions;

- has been conceived with a rapidly developing technology in mind; the framework is sufficiently robust and comprehensive to cater for future developments;

- enables product and application sector international standards, dealing with E/E/PE safety-related systems, to be developed; the development of product and application sector international standards, within the framework of this document, should lead to a high level of consistency (for example, of underlying principles, terminology etc.) both within application sectors and across application sectors; this will have both safety and economic benefits;

- provides a method for the development of the safety requirements specification necessary to achieve the required functional safety for E/E/PE safety-related systems;

- adopts a risk-based approach by which the safety integrity requirements can be determined;

- introduces safety integrity levels for specifying the target level of safety integrity for the safety functions to be implemented by the E/E/PE safety-related systems;

NOTE 1 The standard does not specify the safety integrity level requirements for any safety function, nor does it mandate how the safety integrity level is determined. Instead, it provides a risk-based conceptual framework and example techniques.

- sets target failure measures for safety functions carried out by E/E/PE safety-related systems, which are linked to the safety integrity levels;

- 386— sets a lower limit on the target failure measures for a safety function carried out by a single
387 E/E/PE safety-related system. For E/E/PE safety-related systems operating in
- 388 • a low demand mode of operation, the lower limit is set at an average probability of a
389 dangerous failure on demand of 10^{-5} ;
 - 390 • a high demand or a continuous mode of operation, the lower limit is set at an average
391 frequency of a dangerous failure of 10^{-9} [h⁻¹];
- 392 NOTE 2 A single E/E/PE safety-related system does not necessarily mean a single-channel architecture.
- 393 NOTE 3 It can be possible to achieve designs of safety-related systems with lower values for the target safety
394 integrity for non-complex systems, but these limits are considered to represent what can be achieved for relatively
395 complex systems (for example programmable electronic safety-related systems) at the present time.
- 396— sets requirements for the avoidance and control of systematic faults, which are based on
397 experience and judgement from practical experience gained in industry. Even though the
398 probability of occurrence of systematic failures cannot in general be quantified the standard
399 does, however, allow a claim to be made, for a specified safety function, that the target failure
400 measure associated with the safety function can be considered to be achieved if all the
401 requirements in the standard have been met;
- 402— introduces systematic capability as a measure of confidence that an E/E/PE system meets the
403 safety requirements with regards to avoidance and control of systematic faults;
- 404— adopts a broad range of principles, techniques and measures to achieve functional safety for
405 E/E/PE safety-related systems but does not explicitly use the concept of fail safe. However, the
406 concepts of “fail safe” can be applicable and adoption of such concepts is acceptable providing
407 the requirements of the relevant clauses in the standard are met.
- 408

FUNCTIONAL SAFETY OF ELECTRICAL/ELECTRONIC/ PROGRAMMABLE ELECTRONIC SAFETY-RELATED SYSTEMS –

Part 3: Software requirements

1 Scope

1.1 This part of the IEC 61508 series:

- a) is intended to be utilized only after a thorough understanding of, and in conjunction with, the requirements of IEC 61508-1 and IEC 61508-2;
- b) applies to any software forming part of a safety-related system or used to develop a safety-related system within the scope of IEC 61508-1 and IEC 61508-2. Such software is termed safety-related software (including operating systems, system software, software in communication networks, human-computer interface functions, and firmware as well as application software);
- c) provides specific requirements applicable to support tools used to develop and configure a safety-related system within the scope of IEC 61508-1 and IEC 61508-2;
- d) requires that the software safety functions and their systematic capability are specified;

NOTE 1 If this has already been done as part of the specification of the E/E/PE safety-related systems (see 7.2 of IEC 61508-2), then it does not have to be repeated in this part.

NOTE 2 Specifying the software safety functions and their systematic capability is an iterative procedure; see Figure 5 and Figure 6.

NOTE 3 See IEC 61508-1 Clause 5 and IEC 61508-1 Annex A for documentation structure. The documentation structure can be organised to take account of company procedures, and of the working practices of specific application sectors.

NOTE 4 See IEC 61508-4 3.5.8 for definition of the term "systematic capability".

- e) establishes requirements for safety lifecycle phases and activities which shall be applied during the design and development of the safety-related software (the software safety lifecycle model). These requirements include the application of measures and techniques, which are graded against the required systematic capability, for the avoidance of and control of faults and failures in the software;
- f) provides requirements for information relating to the software aspects of system safety validation to be passed to the organisation carrying out the E/E/PE system integration;
- g) provides requirements for the preparation of information and procedures concerning software needed by the user for the operation and maintenance of the E/E/PE safety-related system;
- h) provides requirements to be met by the organisation carrying out modifications to safety-related software;
- i) provides, in conjunction with IEC 61508-1 and IEC 61508-2, requirements for support tools such as development and design tools, language translators, testing and debugging tools, configuration management tools;

NOTE 5 Figure 5 shows the relationship between IEC 61508-2 and this document.

j) Not used;

k) Does apply to software algorithms

- i. software technology class I (see definition in IEC 61508-4 Clause 3.2.14);
- ii. software technology class II and III (see definitions in IEC 61508-4, Clause 3.2.15 and Clause 3.2.16)

457 NOTE 6 The software algorithms classes relate to the generically used term “artificial intelligence” as defined in
458 ISO/IEC 22989.

459 NOTE 7 Refer to ISO/IEC TR 5469:2024 and ISO/IEC TS 22440 series for further details.

460

461 1.2 This document is a basic safety publication to be used in conjunction with the other parts
462 of IEC 61508 for use by end users to evaluate functional safety applications, or by technical
463 committees in the preparation of standards in accordance with the principles contained in IEC
464 Guide 104 and ISO/IEC Guide 51. This document does not apply in the context of low
465 complexity E/E/PE safety-related systems (see IEC 61508-4 3.4.3).

466 1.3 One of the responsibilities of a technical committee is, wherever applicable, to make use
467 of basic safety publications in the preparation of its publications. In this context, the
468 requirements, test methods or test conditions of this basic safety publication will not apply
469 unless specifically referred to or included in the publications prepared by those technical
470 committees.

471 1.4 Figure 1 shows the overall framework of the IEC 61508 series and indicates the role that
472 IEC 61508-3 plays in the achievement of functional safety for E/E/PE safety-related systems.
473 Figure 2 shows the overall safety lifecycle specified in the IEC 61508 series, with this document
474 addressing part of box 10.

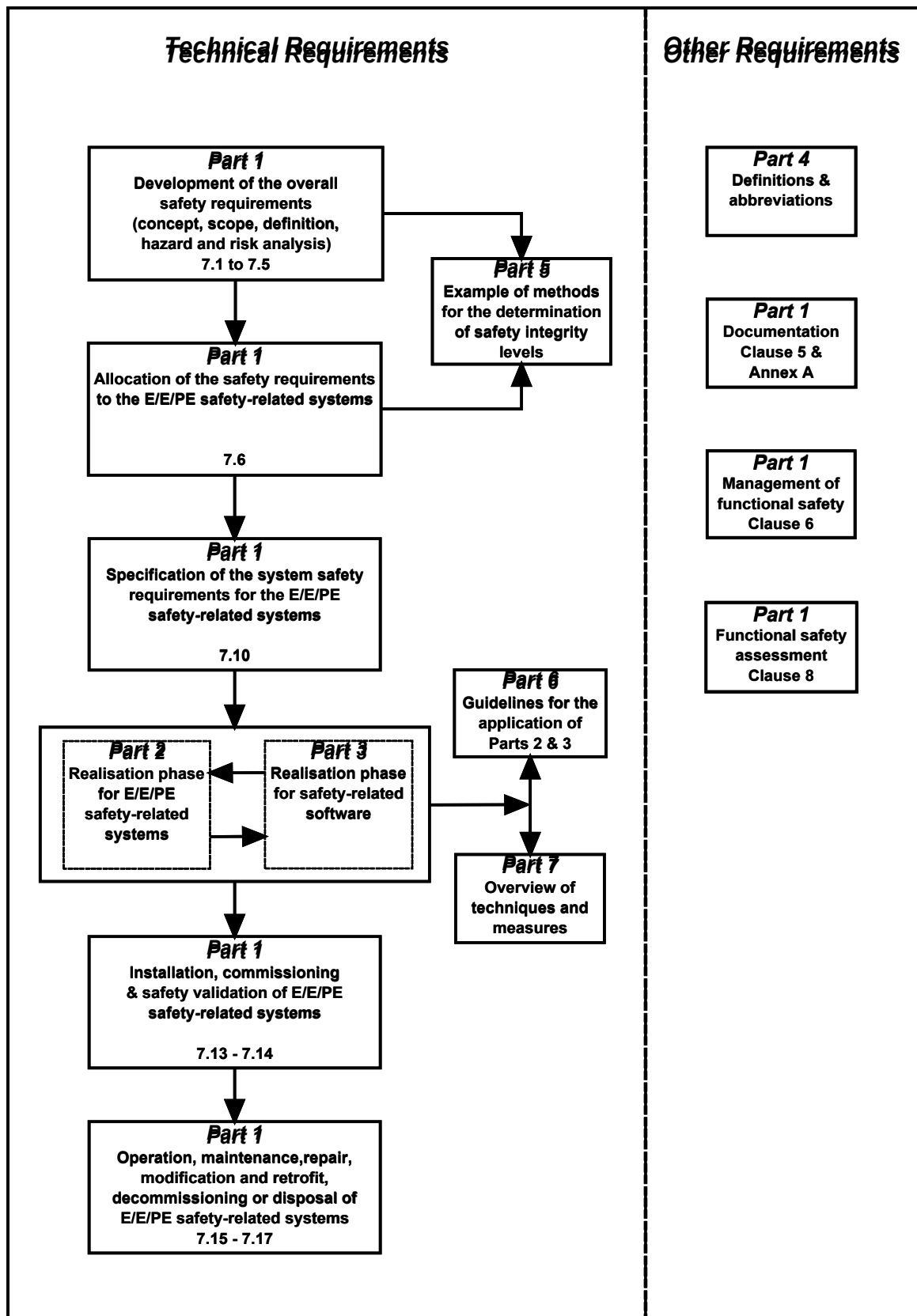


Figure 1 – Overall framework of the IEC 61508 series

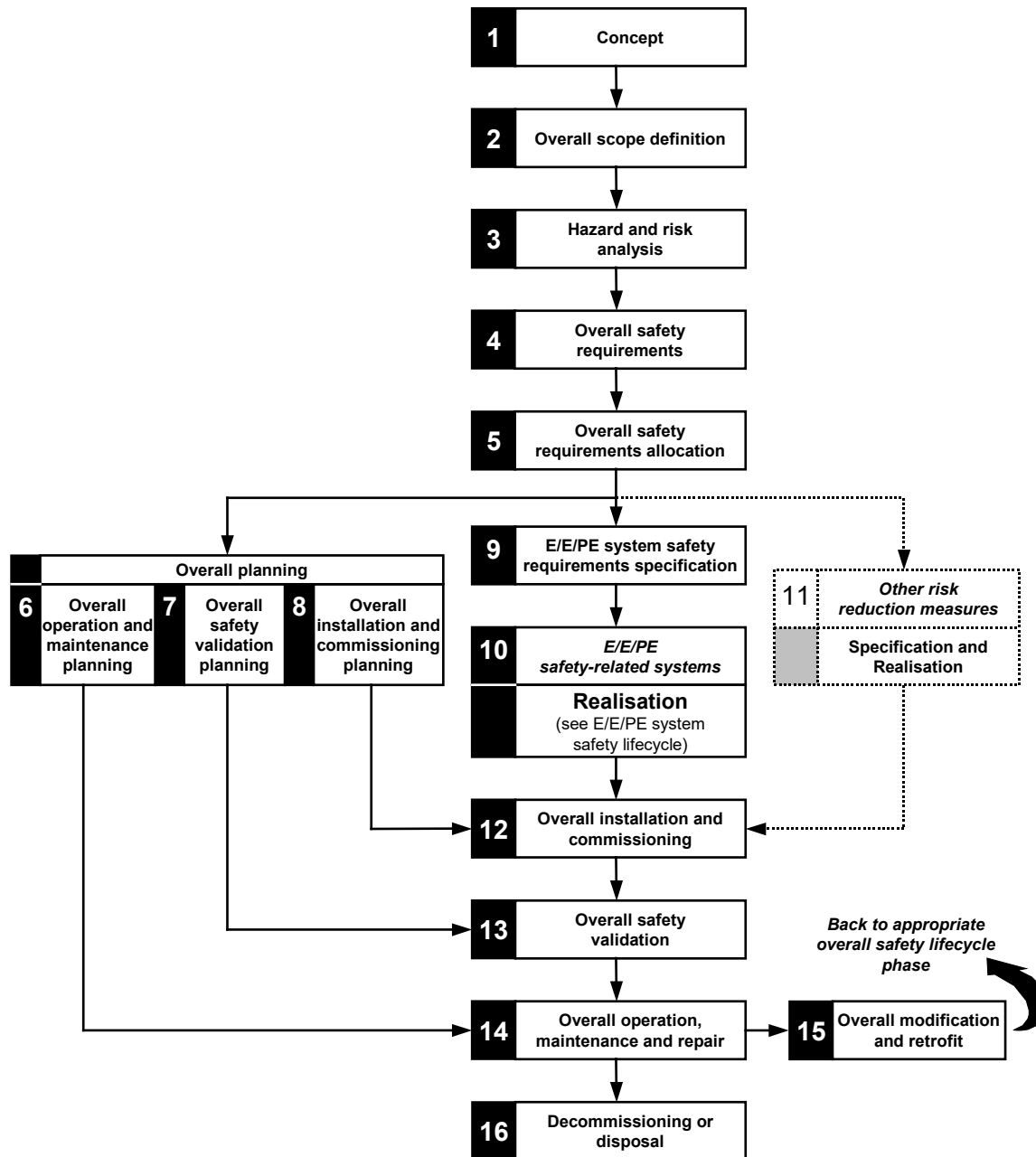


Figure 2 – Overall E/E/PE system safety lifecycle

481 **2 Normative references**

482 The following referenced documents are indispensable for the application of this document. For
483 dated references, only the edition cited applies. For undated references, the latest edition of
484 the referenced document (including any amendments) applies.

485 IEC 61508-1: 2025, *Functional safety of electrical/electronic/programmable electronic safety-*
486 *related systems – Part 1: General requirements*

487 IEC 61508-2: 2025, *Functional safety of electrical/electronic/programmable electronic safety-*
488 *related systems – Part 2: Requirements for electrical/electronic/programmable electronic*
489 *safety-related systems*

490 IEC 61508-4: 2025, *Functional safety of electrical/electronic/programmable electronic safety-*
491 *related systems – Part 4: Definitions and abbreviations*

492 IEC Guide 104:2019, *The preparation of safety publications and the use of basic safety*
493 *publications and group safety publications*

494 IEC/ISO Guide 51:2014, *Safety aspects – Guidelines for their inclusion in standards*

495 **3 Terms, definitions, symbols and abbreviations**

496 For the purposes of this document, the terms, definitions, symbols and abbreviations given in
497 IEC 61508-4 apply.

498 **4 Conformance to this document**

499 The requirements for conformance to this document are given in IEC 61508-1 Clause 4.

500 **5 Documentation**

501 The objectives and requirements for documentation are given in IEC 61508-1 Clause 5.

502 **6 Requirements for management of safety-related software**

503 **6.1 Objectives**

504 The objectives are as detailed in IEC 61508-1 6.1.

505 **6.2 Requirements**

506 6.2.1 The requirements are as detailed in IEC 61508-1 6.2 with the exception of IEC 61508-1
507 6.2.11, but with the following additional requirements.

508 6.2.2 The functional safety planning shall define the strategy for software procurement,
509 development, integration, verification, validation and modification to the extent required by the
510 safety integrity level of the safety functions implemented by the E/E/PE safety-related system.

511 NOTE The philosophy of this approach is to use the functional safety planning as an opportunity to customize this
512 document to take account of the required safety integrity level for each safety function implemented by the E/E/PE
513 safety-related system.

514 6.2.3 Software configuration management shall:

- 515a) apply administrative and technical controls throughout the software safety lifecycle to ensure
516 that the specified requirements for safety-related software are satisfied;
- 517b) guarantee that all necessary operations have been carried out to demonstrate that the required
518 systematic capability has been achieved;
- 519c) maintain accurately and with unique identification all configuration items which are necessary
520 to meet the safety integrity requirements of the E/E/PE safety-related system. Configuration
521 items include at least the following: safety analysis and requirements; software specification
522 and design documents; software source code modules; test plans and results; verification
523 documents; pre-existing software elements and packages which are to be incorporated into the
524 E/E/PE safety-related system; all tools and development environments which are used to create
525 or test, or carry out any action on, the software of the E/E/PE safety-related system;
- 526d) apply modification procedures to authorize, conduct and document modification requests and
527 prevent unauthorized modifications:

528 NOTE 1 When an explicit combination of versions is necessary, modification procedures need to ensure that
529 versions match for the intended purpose.

530 NOTE 2 For further information on configuration management, see IEC 61508-7 C.5.24

- 531 1) to prevent unauthorized modifications;
- 532 2) to document modification requests;
- 533 3) to analyse the impact of a proposed modification, and to approve or reject the request;
- 534 4) to document the details of, and the authorisation for, all approved modifications;
- 535 5) to establish configuration baseline at appropriate points in the software development, and
536 to document the (partial) integration testing of the baseline;
- 537 6) to guarantee the composition of, and the building of, all software baselines (including the
538 rebuilding of earlier baselines).
- 539 e) ensure that appropriate methods are implemented to load valid software elements and data
540 correctly into the run-time system;

541 NOTE 3 This can include consideration of specific target location systems as well as general systems. For
542 example, software including application software can be impacted by the systematic capability of the loading
543 method, e.g. firmware.

- 544 f) document the following information to permit a subsequent functional safety audit:
545 configuration status, release status, the justification (taking account of the impact analysis)
546 for and approval of all modifications, and the details of the modification;
- 547 g) formally document the release of safety-related software. Master copies of the software and
548 all associated documentation and version of data in service shall be kept to permit
549 maintenance and modification throughout the operational lifetime of the released software.

550 NOTE 4 This requirement is applicable regardless of the lifecycle phase in which the modification is decided.

551 7 Software safety lifecycle requirements

552 7.1 General

553 7.1.1 Objective

554 7.1.1.1 The objective of the requirements of this subclause is to structure the development of
555 the software into defined phases and activities (see Table 1 and Figure 3 to Figure 6).

556 7.1.2 Requirements

557 7.1.2.1 A safety lifecycle for the development of software shall be selected and specified
558 during safety planning in accordance with IEC 61508-1 Clause 7.

559 7.1.2.2 Any software lifecycle model can be used provided all the objectives and requirements
560 of this clause are met.

7.1.2.3 Each phase of the software safety lifecycle shall be divided into elementary activities with the scope, inputs and outputs specified for each phase.

NOTE See Figure 3, Figure 4 and Table 1.

7.1.2.4 Provided that the software safety lifecycle satisfies the requirements of Table 1, it is acceptable to tailor the V-model (see Figure 6) or incremental development models (e.g. agile software development methodologies) to take account of the E/E/PES system safety integrity requirements and the complexity of the project.

NOTE 1 A software safety lifecycle model which satisfies the requirements of this clause can be customized for the particular needs of the project or organisation. The full list of lifecycle phases in Table 1 is suitable for large newly developed systems. In small systems, it can be appropriate, for example, to merge the phases of software system design and architectural design.

NOTE 2 See Annex G for tailoring of the software safety life cycle against complexity and implementation characteristics of functionalities of the software.

NOTE 3 When performing iterative and incremental development, regression validation is important. See IEC 61508-7: C.5.25 Regression validation

NOTE 4 The incremental paradigm does not affect the need to meet the requirements of this document, such as verification, validation and independent assessment.

7.1.2.5 Any customisation of the software safety lifecycle shall be justified on the basis of functional safety.

7.1.2.6 Quality and safety assurance procedures shall be integrated into safety lifecycle activities.

7.1.2.7 For each lifecycle phase, appropriate techniques and measures shall be used. Annex A and Annex B provide a guide to the selection of techniques and measures, with references to IEC 61508-6 and IEC 61508-7, which parts give recommendations on specific techniques to achieve the properties required for systematic capability. Annex C provides information on the safety properties expected to arise from the use of the recommended techniques and measures.

NOTE 1 Selecting techniques from these recommendations does not guarantee by itself that the required systematic capability can be achieved.

NOTE 2 Success in achieving systematic capability depends on selecting techniques and measures with attention to the following factors:

- the consistency and the complementary nature of the chosen methods, languages and tools for the whole development cycle;
- whether the developers use methods, languages and tools they fully understand;
- whether the methods, languages and tools are well-adapted to the specific problems encountered during development.

7.1.2.8 The results of the activities in the software safety lifecycle shall be documented (see Clause 5).

NOTE IEC 61508-1 Clause 5 considers the documented outputs from the safety lifecycle phases. In the development of some E/E/PE safety-related systems, the output from some safety lifecycle phases can be in the form of a distinct document, while in others the documented outputs from several phases can be merged into a single document. The essential requirement is that the output of the safety lifecycle phase be fit for its intended purpose.

7.1.2.9 If at any phase of the software safety lifecycle, a modification is required pertaining to an earlier lifecycle phase, then an impact analysis shall determine:

- a) which software modules are impacted;
- b) which earlier safety lifecycle activities shall be repeated.

NOTE At one extreme, an impact analysis can be based on an informal assessment. At the other extreme, an impact analysis can be based on a rigorous analysis using formal methods of the potential adverse impact of all proposed

changes, including those changes which can be inadequately understood or implemented. See IEC 61508-7 C.5.23 for guidance on impact analysis.

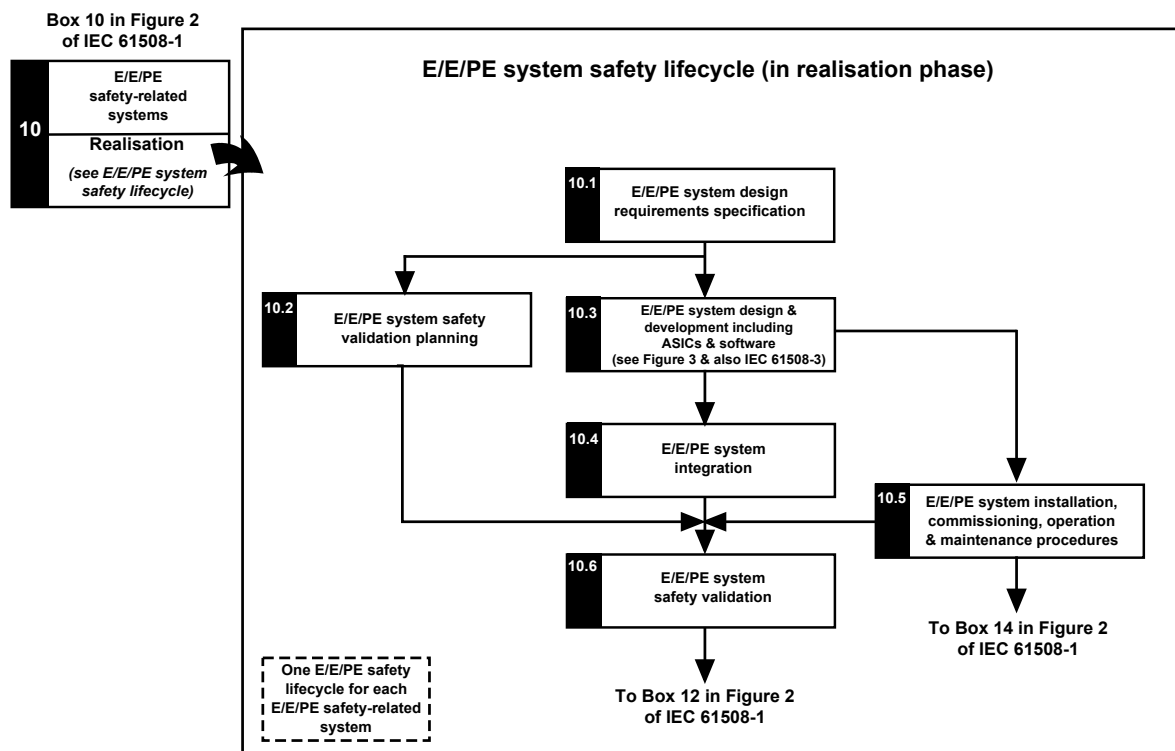


Figure 3 – E/E/PE system safety lifecycle (in realisation phase)

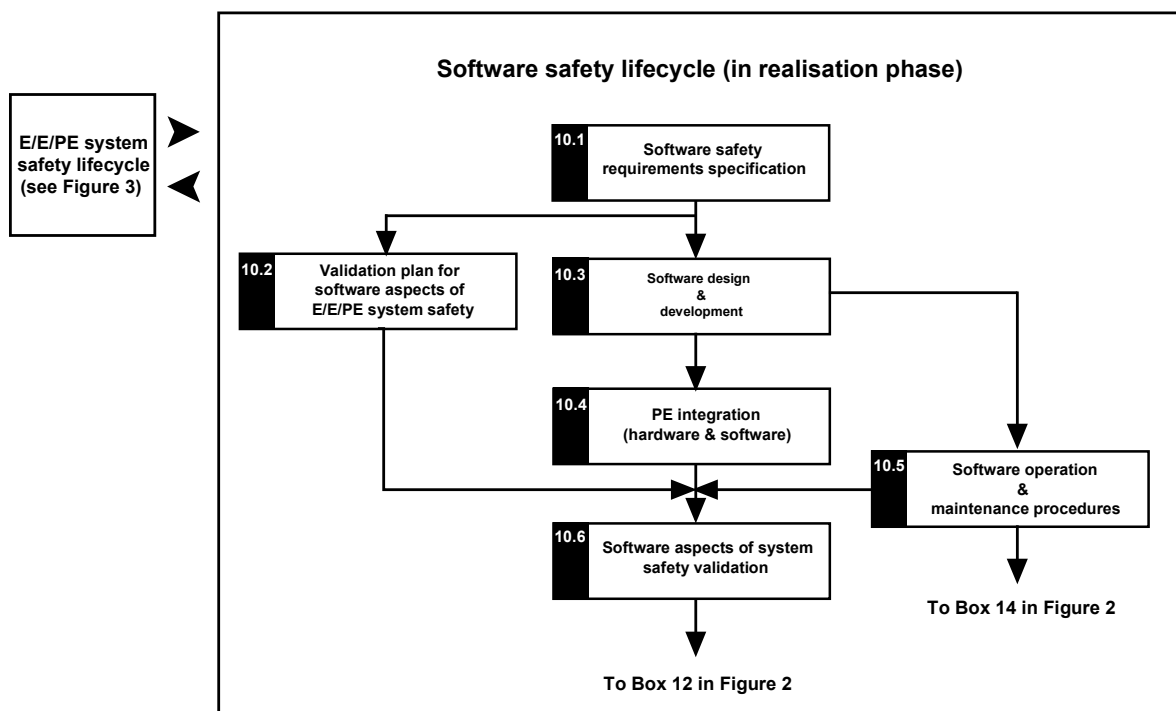
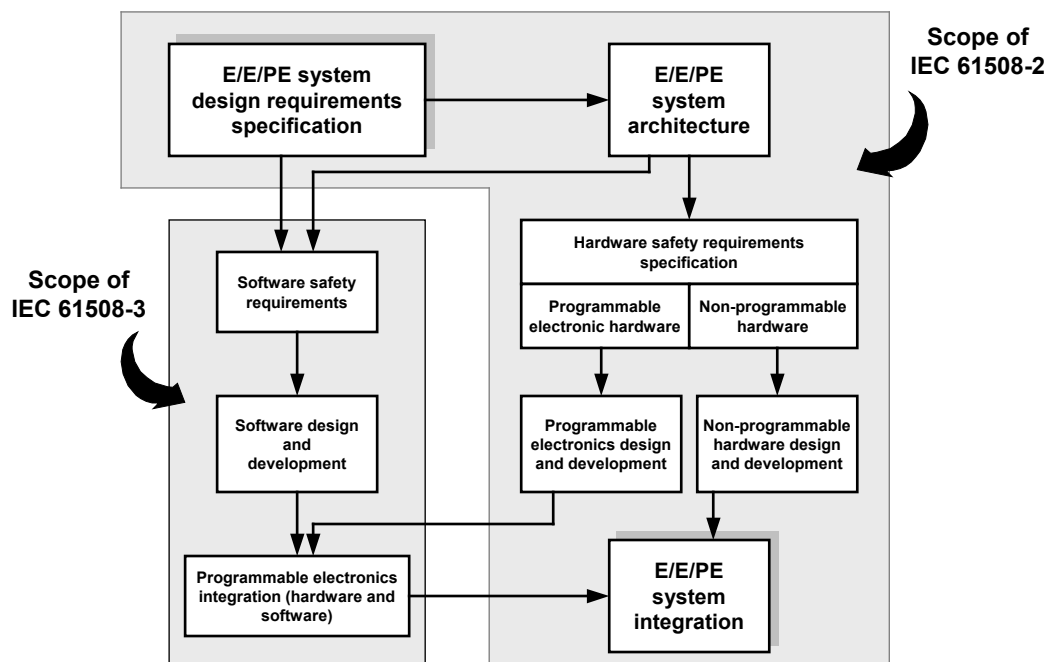


Figure 4 – Software safety lifecycle (in realisation phase)

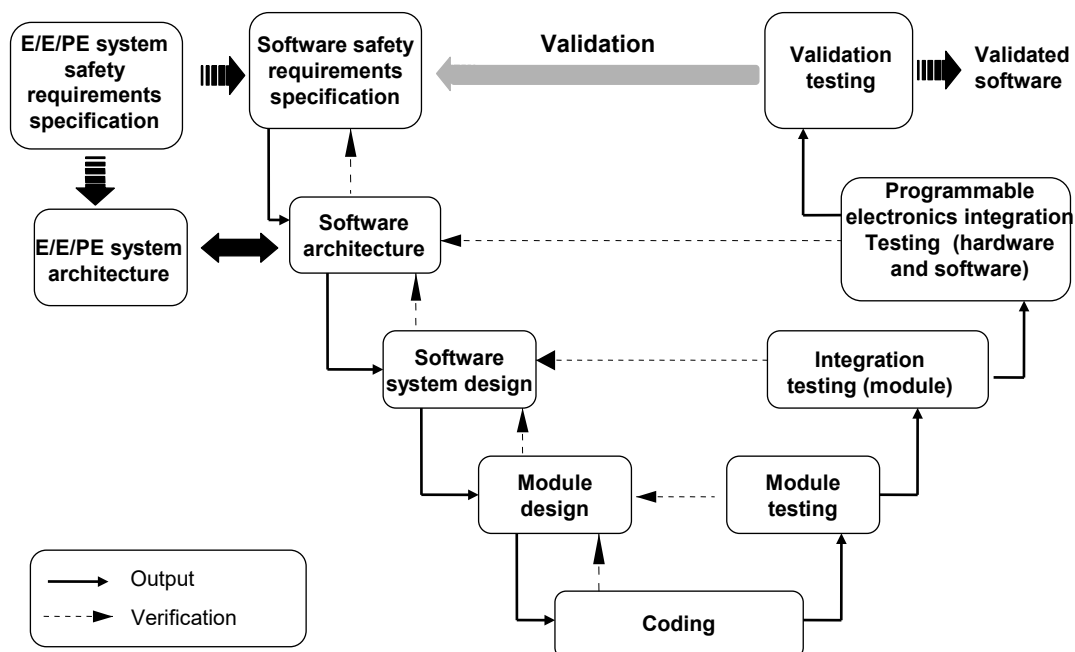
615



616

617

Figure 5 – Relationship and scope for IEC 61508-2 and this document (see also Annex E)



618

619

620

Figure 6 – Systematic capability for software and the development lifecycle (the V-model)

621

Table 1 – Software safety lifecycle – overview (1 of 4)

Safety lifecycle phase		Objectives	Scope	Requirements subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Safety Lifecycle Activity					
10.1	Software safety requirements specification	To specify the requirements for safety-related software in terms of the requirements for software safety functions and the requirements for systematic capability To specify the requirements for the software safety functions for each E/E/PE safety-related system necessary to implement the required safety functions To specify the requirements for systematic capability for each E/E/PE safety-related system necessary to achieve the safety integrity level specified for each safety function allocated to that E/E/PE safety-related system	PE system; software system	7.2.2	E/E/PE safety requirements specification as developed during allocation (see IEC 61508-1) E/E/PE system safety requirements specification (from IEC 61508-2) Hardware-software interface specification	hardware-software interface specification software safety requirements specification
10.2	Validation plan for software aspects of system safety	To develop a plan for validating the software aspects of system safety	PE system; software system	7.3.2	software safety requirements specification	validation plan for software aspects of system safety
10.3	Software design and development	Architecture: To create a software architecture that fulfils the specified requirements for safety-related software with respect to the required systematic capability To evaluate the requirements placed on the software by the hardware architecture of the E/E/PE safety-related system, including the significance of E/E/PE hardware/software interactions for safety of the equipment under control	PE system; software system	7.4.3	software safety requirements specification E/E/PE system hardware architecture design (from IEC 61508-2) Hardware-software interface specification	software architecture design software architecture integration test specification software/ PE integration test specification (NOTE also required by IEC 61508-2) Hardware-software interface specification
10.3	Software design and development	Support tools and programming languages: To select a suitable set of tools, including languages and compilers, run-time system interfaces, human-system interfaces, and data formats and representations for the required systematic capability, over the whole safety lifecycle of the software which assists verification, validation, assessment and modification	PE system; software system; support tools; programming language	7.4.4	software safety requirements specification software architecture design	support tools and coding standards selection of development tools

622

Table 1 (2 of 4)

Safety lifecycle phase		Objectives	Scope	Requirements subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Safety Lifecycle Activity					
10.3	Software design and development	Detailed design and development (software system design): To design and implement software that fulfils the specified requirements for safety-related software with respect to the required systematic capability, which is analysable and verifiable, and which is capable of being safely modified	major elements and subsystems of software architectural design.	7.4.5	software architecture design support tools and coding standards Hardware-software interface specification	software system design specification software system integration test specification
10.3	Software design and development	Detailed design and development (individual software module design): To design and implement software that fulfils the specified requirements for safety-related software with respect to the required systematic capability, which is analysable and verifiable, and which is capable of being safely modified	software system design	7.4.5	software system design specification support tools and coding standards	software module design specification software module test specification
10.3	Software design and development	Detailed code implementation: To design and implement software that fulfils the specified requirements for safety-related software with respect to the required systematic capability, which is analysable and verifiable, and which is capable of being safely modified	individual software modules	7.4.6	software module design specification support tools and coding standards	source code listing code review report
10.3	Software design and development	Software module testing: To verify that the requirements for safety-related software (in terms of the required software safety functions and the systematic capability) have been achieved To show that each software module performs its intended function and does not perform unintended functions To ensure, in so far as it is appropriate, that configuration of PE systems by data fulfils the specified requirements for the systematic capability	software modules	7.4.7	software module test specification source code listing code review report	software module test results verified and tested software modules

623

624

625

Table 1 (3 of 4)

Safety lifecycle phase		Objectives	Scope	Requirements subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Safety Lifecycle Activity					
10.3	Software design and development	Software integration testing: To verify that the requirements for safety-related software (in terms of the required software safety functions and the systematic capability) have been achieved To show that all software modules, elements and subsystems interact correctly to perform their intended function and do not perform unintended functions To ensure, in so far as it is appropriate, that configuration of PE systems by data fulfils the specified requirements for the systematic capability	software architecture software system	7.4.8	software system integration test specification	software system integration test results verified and tested software system
10.4	Programmable electronics integration (hardware and software)	To integrate the software onto the target programmable electronic hardware To combine the software and hardware in the safety-related programmable electronics to ensure their compatibility and to meet the requirements of the intended systematic capability	programmable electronics hardware integrated software	7.5.2	software architecture integration test specification software/PE integration test specification (also required by IEC 61508-2). Integrated programmable electronics Hardware-software interface specification	software architecture integration test results programmable electronics integration test results verified and tested integrated programmable electronics
10.5	Software operation and modification procedures	To provide information and procedures concerning software necessary to ensure that the functional safety of the E/E/PE safety-related system is maintained during operation and modification	as above	7.6.2	all above, as relevant	software operation and modification procedures

626

627

628

Table 1 (4 of 4)

Safety lifecycle phase		Objectives	Scope	Requirements subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Safety Lifecycle Activity					
10.6	Software aspects of system safety validation	To ensure that the integrated system complies with the specified requirements for safety-related software	as above	7.7.2	validation plan for software aspects of system safety	software safety validation results validated software
–	Software modification	To guide incremental lifecycle deployment as well as corrections, enhancements or adaptations to the validated software, ensuring that the required systematic capability is sustained	as above	7.8.2	software modification procedures software modification request software incremental development procedures	software modification impact analysis results software modification log software versioning documents (e.g. configuration documents, release notes)
–	Software verification	To test and evaluate the outputs from a given software safety lifecycle phase to ensure correctness and consistency with respect to the outputs and standards provided as input to that phase	depends on phase	7.9.2	appropriate verification plan (depends on phase)	appropriate verification report (depends on phase)
–	Software functional safety assessment	To investigate and arrive at a judgement on the software aspects of the functional safety achieved by the E/E/PE safety-related systems	all above phases	8	software functional safety assessment plan	software functional safety assessment report

629

630 7.2 Software safety requirements specification

631 7.2.1 Objectives

632 7.2.1.1 The first objective of the requirements of this subclause is to specify the requirements
 633 for safety-related software in terms of the requirements for software safety functions and the
 634 requirements for systematic capability.

635 7.2.1.2 The second objective of the requirements of this subclause is to specify the
 636 requirements for the software safety functions for each E/E/PE safety-related system necessary
 637 to implement the required safety functions.

638 7.2.1.3 The third objective of the requirements of this subclause is to specify the requirements
 639 for systematic capability for each E/E/PE safety-related system necessary to achieve the safety
 640 integrity level specified for each safety function allocated to that E/E/PE safety-related system.

641 NOTE This phase is Box 10.1 of Figure 4.

642 7.2.2 Requirements

643 7.2.2.1 The following properties of the software should be considered during development of
 644 the software safety requirements specification:

- a) completeness with respect to the safety needs to be addressed by software;
- b) correctness with respect to the safety needs to be addressed by software.
- c) freedom from intrinsic specification faults, including freedom from ambiguity.
- d) understandability of safety requirements.
- e) freedom from adverse interference of non-safety functions with the safety needs to be addressed by software;
- f) capability of providing a basis for verification and validation.

NOTE 1 These requirements can be achieved by a combination of system (i.e. generic embedded) software and application specific software. It is the combination of both that provides the features that can satisfy the following subclauses. The exact division between system and application specific software can depend on the chosen software architecture (see 7.4.3).

NOTE 2 Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and Annex B techniques and measures.

NOTE 3 The safety requirements to be addressed by software are the set of safety functions and corresponding safety integrity requirements assigned to software functions by the design of the E/E/PE system. The complete set of system safety requirements can include additional safety functions that do not depend on software. The completeness of the software safety requirements specification depends crucially on the effectiveness of earlier system lifecycle phases.

NOTE 4 The software safety requirements can be included in the E/E/PE system safety requirements specification (see IEC 61508-2 Clause 7), in which case they do not have to be repeated in a separate software requirements specification.

7.2.2.2 The software safety requirements specification shall be derived from the safety requirements specification of the E/E/PE safety-related system (see IEC 61508-1 7.10 and IEC 61508-2 7.2), and any requirements of safety planning (see Clause 6). This information shall be made available to the software developer.

NOTE 1 This requirement does not mean that there will be no iteration between the developer of the E/E/PE system (see IEC 61508-2) and the developer of the software (see this document). As the safety-related software requirements and the software architecture become more precise, there can be an impact on the E/E/PE system hardware architecture, and for this reason close co-operation between the hardware and software developer is essential. See Figure 5.

NOTE 2 Where a software design incorporates pre-existing reusable software, that software can be developed without taking account of the current system requirement specification. See 7.4.2.12 for the requirements on the pre-existing software to satisfy the software safety requirements specification.

7.2.2.3 The specification of the requirements for safety-related software shall be sufficiently detailed to allow the design and implementation to achieve the required systematic capability (including any requirement for independence, see 7.4.2.11), and to allow an assessment of functional safety to be carried out.

NOTE The level of detail of the specification varies with the complexity of the application. Depending on the application, an adequate specification of functional behaviour can include requirements for accuracy, timing and performance, capacity, robustness, overload tolerance, and other characterising properties of the specific application.

7.2.2.4 In order to address independence, a suitable common cause failure analysis shall be carried out. Where credible failure mechanisms are identified, effective defensive measures shall be taken.

NOTE 1 See Annex F for techniques for achieving one aspect of independence of software.

NOTE 2 A systematic approach and the methodology are described in 61508-2 Annex E.

7.2.2.5 The software developer shall evaluate the information in 7.2.2.2 to ensure that the requirements are adequately specified. In particular the software developer shall consider the following:

- a) safety functions;

- b) configuration or architecture of the system;
- c) hardware systematic capability requirements (programmable electronics, sensors, and actuators);
- d) software systematic capability requirements;
- e) capacity and response time;
- f) equipment and operator interfaces, including compatibility with existing operator interfaces and reasonably foreseeable misuse.

7.2.2.6 If not already adequately defined in specified safety requirements of the E/E/PE safety-related system, all relevant modes of operation of the EUC, of the E/E/PE system, and of any equipment or system connected to the E/E/PE system shall be detailed in the specified requirements for safety-related software. The safety related software requirements shall include provisions to guarantee that the global behaviour of the safety related software is not detrimental to each E/E/PE safety related function.

EXAMPLE Functions specified to share a variable representative of the EUC can be implemented with a shared software variable thus creating a data coupling with a potential undesired interaction between functions. Decoupling can be prevented by implementing this feature with two distinct software variables instead.

7.2.2.7 The software safety requirements specification shall specify and document any safety-related or relevant constraints between the hardware and the software.

7.2.2.8 To the extent required by the E/E/PE hardware architecture design, and in consideration of a possible increase in complexity, the software safety requirements specification shall consider the following:

- a) specification of the functions related to the detection, annunciation and management of faults in the software itself;
- b) monitoring of the programmable electronics hardware, sensors, and actuators;
- c) periodic testing of safety functions while the system is running;
- d) enabling safety functions to be testable when the EUC is operational;
- e) software functions to execute proof tests and all diagnostic tests in order to fulfil the safety integrity requirement of the E/E/PE safety-related system;

NOTE The increased complexity resulting from the above considerations a) to e) can require the architecture to be revisited.

7.2.2.9 When the E/E/PE safety-related system is required to perform non-safety functions, then the specified requirements for safety-related software shall clearly identify the non-safety functions.

NOTE See 7.4.2.8 and 7.4.2.9 for requirements on non-interference between safety functions and non-safety functions.

7.2.2.10 The software safety requirements specification shall express the required safety properties of the product, but not of the project as this is covered by safety planning (see Clause 6). With reference to 7.2.2.1 to 7.2.2.9, the following shall be specified as appropriate:

- a) the requirements for the following software safety functions:
- b) functions that enable the EUC to achieve or maintain a safe state;
- c) functions related to the detection, annunciation and management of faults in the programmable electronics hardware;
- d) functions related to the detection, annunciation and management of sensor and actuators faults;
- e) functions related to the detection, annunciation and management of faults in the software itself;

- f) functions related to the periodic testing of safety functions on-line (i.e. in the intended operational environment);
- g) functions related to the periodic testing of safety functions off-line (i.e. in an environment where the EUC is not being relied upon for its safety function);
- h) functions that allow the PE system to be safely modified;
- i) interfaces to non-safety-related functions;
- j) capacity and response time performance;
- k) interfaces between the software and the PE system;

NOTE 1 This is determined in interaction with the system development and documented in the hardware-software interface, see also IEC 61508-2 cl. 7.4.2.10.

NOTE 2 Interfaces include both off-line and on-line programming facilities.

- l) safety-related communications (see IEC 61508-2 7.4.11).
- m) the requirements for the systematic capability:
 - 1) for each of the software safety functions in a) above;

NOTE 3 See IEC 61508-5 Annex A for information concerning the allocation of safety integrity to the E/E/PE system functions.

- 2) for independence requirements between functions.

NOTE 4 Independence requirements are addressed in 7.2.2.4 and 7.4.2.9.

7.2.2.11 Where software safety requirements are expressed or implemented by configuration data, the data shall be:

- a) consistent with the system safety requirements;
- b) expressed in terms of the permitted range and authorized combinations of its operational parameters;
- c) defined in a manner which is compatible with the underlying software (for example sequence of execution, run time, data structures, etc.)
- d) protected against unintended change.

NOTE 1 This requirement on application data is particularly relevant to data driven systems.

NOTE 2 See Annex G for guidance on systems configured by application data and Annex J for guidance on data driven systems.

7.2.2.12 Where data defines the interface between software and external systems, the following performance characteristics shall be considered, in addition to IEC 61508-2 7.4.11:

- a) the need for consistency in terms of data definitions;
- b) invalid, out of range or untimely values;
- c) response time and throughput, including maximum loading conditions;
- d) best case and worst-case execution time, and deadlock;
- e) overflow and underflow of data storage capacity.

7.2.2.13 Operational parameters shall be protected against:

- a) invalid, out of range or untimely values;
- b) unauthorized changes;
- c) corruption.

7.2.2.14 Human-system interface design principles shall be implemented in accordance with IEC 61508-1 7.1.2.3.

7.3 Validation plan for software aspects of system safety

7.3.1 Objective

7.3.1.1 The objective of the requirements of this subclause is to develop a plan for validating the safety-related software aspects of system safety.

NOTE 1 This phase is Box 10.2 of Figure 4.

NOTE 2 Software usually cannot be validated separately from its underlying hardware and system environment.

7.3.2 Requirements

7.3.2.1 Planning shall be carried out to specify the steps, both procedural and technical, that will be used to demonstrate that the software satisfies its safety requirements.

7.3.2.2 The validation plan for software aspects of system safety shall consider the following:

- a) details of when the validation shall take place;
- b) details of those who shall carry out the validation;
- c) identification of the relevant modes of the EUC operation including:
 - 1) preparation for use including setting and adjustment;
 - 2) start up, teach, automatic, manual, semi-automatic, steady state operation;
 - 3) re-setting, shut down, maintenance;
 - 4) reasonably foreseeable abnormal conditions and reasonably foreseeable operator misuse.
- d) identification of the safety-related software which needs to be validated for each mode of EUC operation before commissioning commences;
- e) the test patterns combining the appropriate verification methods to address functional testing, structural testing and regression testing;
- f) in accordance with item e), the measures (techniques) and procedures that shall be used for confirming that each safety function conforms with the specified requirements for the safety functions, and the specified requirements for systematic capability;
- g) the required environment in which the validation activities are to take place (for example, for tests this can include calibrated tools and equipment);
- h) the pass/fail criteria;
- i) the policies and procedures for evaluating the results of the validation, particularly those activities to be taken in the case of a failure occurring during validation, including impact analysis.

NOTE These requirements are based on the general requirements given in IEC 61508-1 7.8.

7.3.2.3 The rationale for the chosen validation strategy shall be documented during the validation planning. The validation strategy for the safety-related software shall include the following information:

- j) choice of manual or automated validation techniques or both;
- k) choice of static or dynamic validation techniques or both;
- l) choice of analytical or statistical validation techniques or both;
- m) choice of acceptance criteria based on objective factors or expert judgment or both.

7.3.2.4 As part of the procedure for validating safety-related software aspects, the scope and contents of the validation plan for software aspects of system safety shall be agreed with the assessor or with a party representing the assessor, if required by the safety integrity level of the E/E/PE system (see IEC 61508-1 Clause 8). This procedure shall also make a statement concerning the presence of the assessor during testing.

7.3.2.5 The pass/fail criteria for accomplishing software validation shall include:

- a) the required input signals with their sequences and their values;
- b) the anticipated output signals with their sequences and their values;
- c) other acceptance criteria, for example memory usage, timing and value tolerances.

7.4 Software design and development

7.4.1 Objectives

7.4.1.1 The first objective of the requirements of this subclause is to create a software architecture that fulfils the specified requirements for safety-related software with respect to the required safety integrity level of the E/E/PE system.

7.4.1.2 The second objective of the requirements of this subclause is to evaluate the requirements placed on the software by the hardware architecture of the E/E/PE safety-related system, including the significance of E/E/PE hardware/software interactions for safety of the equipment under control.

7.4.1.3 The third objective of the requirements of this subclause is to select a suitable set of tools, including languages and compilers, run-time system interfaces, human-system interfaces, and data formats and representations for the required systematic capability, over the whole safety lifecycle of the software which assists verification, validation, assessment and modification.

7.4.1.4 The fourth objective of the requirements of this subclause is to design and implement software that fulfils the specified requirements for safety-related software with respect to the required safety integrity level of the E/E/PE system, which is analysable and verifiable, and which is capable of being safely modified.

NOTE At this level, architecture of the E/E/PE safety related systems and other risk reduction measures can be discussed. ISO/IEC/IEEE 15288:2023 provides guidance on the specification of architectures. The architecture specification can include different architecture views to depict the architecture from the point of view of the different disciplines involved in the system design (user, provider etc.). For larger systems, the architecture specification can consist of architecture frameworks that reduce failure possibilities by limiting complexity of single architecture parts. ISO/IEC/IEEE 42010:2022 provides guidance on architecture frameworks.

7.4.1.5 The fifth objective of the requirements of this subclause is to verify that the requirements for safety-related software (in terms of the required software safety functions and the systematic capability) have been achieved.

7.4.1.6 The sixth objective of the requirements of this subclause is to ensure, in so far as it is appropriate, that configuration of PE systems by data fulfils the specified requirements for the systematic capability.

NOTE This phase is box 10.3 of Figure 4.

7.4.2 General requirements

7.4.2.1 Depending on the nature of the software development, responsibility for conformance with 7.4 can rest with one or more suppliers and/or with the user. The division of responsibility shall be determined during safety planning (see IEC 61508-1 6.2.2).

NOTE 1 See 7.4.3 for aspects of system and software architecture that are relevant to deciding on a practical division of responsibility.

7.4.2.2. In accordance with the required systematic capability and the specific technical requirements of the safety function, the design method chosen shall possess features that facilitate:

- a) abstraction, modularity and other features which control complexity;
- b) the expression of:

- 1) functionality;
 - 2) information flow between elements;
 - 3) sequencing and time related information;
 - 4) timing constraints;
 - 5) concurrency and synchronized access to shared resources;
 - 6) data structures and their properties;
 - 7) design assumptions and their dependencies;
 - 8) exception handling;
 - 9) design assumptions (pre-conditions, post-conditions, invariants);
 - 10) comments.
- c) ability to represent several views of the design including structural and behavioural views;
 - d) comprehension by developers and others who need to understand the design;
 - e) verification and validation.

NOTE 1 See also the tables in Annex A and Annex B

NOTE 2 Concurrency is addressed in Annex F.

7.4.2.3 Testability and the capacity for safe modification shall be considered during the design activities in order to facilitate implementation of these properties in the final safety-related system.

NOTE Examples include maintenance modes in machinery and process plant.

7.4.2.4 The design method chosen shall possess features that facilitate software modification. Such features include modularity (see F.7), information hiding, encapsulation, testability and independence.

NOTE 1 Regression testing can be carried out using either automated or manual test.

NOTE 2 Automated test suites facilitate regression testing.

7.4.2.5 The design representations shall be based on a notation which is unambiguously defined or restricted to unambiguously defined features (see Table A.2 group 11 and Table A.4 group 1).

7.4.2.6 As far as practicable the design shall keep the safety-related part of the software simple.

7.4.2.7 The software design shall include, commensurate with the required systematic capability, monitoring of control flow and data flow. On failure detection, appropriate actions shall be taken.

7.4.2.8 Where the software is to implement both safety and non-safety functions, then all of the software shall be treated as safety-related, unless adequate design measures ensure that the failures of non-safety functions cannot adversely affect safety functions.

7.4.2.9 Where the software is to implement safety functions of different systematic capability, then all of the software shall be treated as belonging to the highest systematic capability, unless adequate independence between the safety functions of the different systematic capabilities can be shown in the design. It shall be demonstrated either (1) that independence is achieved in both the spatial and temporal domains, or (2) that any violation of independence is controlled. The justification for independence shall be documented.

NOTE See Annex F for techniques for achieving one aspect of independence of software.

7.4.2.10 Where the systematic capability of a software element is lower than the systematic capability of the E/E/PE safety function which the software element supports, the element shall be used in combination with other elements such that the systematic capability of the combination meets the systematic capability of the E/E/PE safety function.

7.4.2.11 Where a safety function is implemented using a combination of software elements of known systematic capability and with sufficient independence, the systematic capability requirements of IEC 61508-2 7.4.3, shall apply to the combination of elements with the following clarifications:

- a) The systematic capability of the synthesis of software elements may be applied to elements with identical element safety functions with diverse implementation or to elements with non-identical element safety functions;
- b) The applied techniques and measures during the safety lifecycle of the software elements shall be analysed and evaluated for the resulting systematic failures and common cause failures in the combined software element (see IEC 61508-2 7.4.3.2);
- c) Integration activities of the combined elements shall meet systematic capability of (N+1) (see IEC 61508-2 7.4.3.2).
- d) The selected measures and techniques for the systematic capability of a software element applied during the safety lifecycle shall be implemented and monitored as required by IEC 61508-1 6.2.17.

NOTE Argumentation can be supported using evidence from classification of the properties (see Annex C) of applied techniques and measures.

7.4.2.12 Where a pre-existing software element is reused to implement all or part of a safety function, the element shall meet both requirements a) and b) below for the required systematic capability:

- f) meet the requirements of one of the following compliance routes:

Route 1_s: compliant with the requirements of this document for the avoidance and control of systematic faults in software, excepting the clauses 7.4.2.13 and 7.4.2.14 as described below;

Route 2_s: proven in use, based on evidence that the element is proven in use (see 7.4.2.13);

Route 3_s: assessment of non-compliant development (see 7.4.2.14).

NOTE 1 Route 1_s, 2_s and 3_s are the element compliance routes of IEC 61508-2 7.4.2.2 b) with particular reference to software elements. They are reproduced here for convenience only, and to minimize references back to IEC 61508-2.

NOTE 2 For the definition of pre-existing software see IEC 61508-4 3.2.8.

NOTE 3 The requirements of this clause on pre-existing software elements also apply to a run-time library or an interpreter.

- g) has a safety manual (see IEC 61508-2 Annex D and Annex D of this document) that gives a sufficiently precise and complete description of the element to make possible an assessment of the integrity of a specific safety function that depends wholly or partly on the pre-existing software element.

NOTE 4 The safety manual can be derived from the pre-existing software element supplier's own documentation and records of the element supplier's development process or can be created or supplemented by additional qualification activities undertaken by the developer of the safety related system or by third parties. In some cases, reverse engineering can be necessary to create specification and design documentation adequate to meet the requirements of this clause, subject to the prevailing legal conditions (e.g. copyright or intellectual property rights).

NOTE 5 The justification of the software element can be developed during safety planning (see Clause 6).

7.4.2.13 To comply with Route 2_s, a pre-existing software element shall meet all of the following requirements;

NOTE 1 Any documentation required by a clause in this document can either be available with the pre-existing software or can be included as part of the documentation of the safety related function.

NOTE 2 A reused software function in this document means a function specified on the level of the requirements specification. A reused software function does not refer to a programming language construct.

NOTE 3 Conditions are set for the data on the history of the pre-existing software in b). The fulfilment of these conditions does not entail that the software is deterministic: hidden internal states of the software can affect its execution even when the required combination as specified in b) is exactly the same.

NOTE 4 In some cases (e.g. input data are analogue data or a clock signal) the demonstration of the results of any existing proven in use for software can be difficult.

- a) For the pre-existing software element, proof of adequate previous use shall be provided;
- b) The functional profiles for a pre-existing software element shall be provided for previous and intended use when a pre-existing software element is reused for a safety function:
 - 1) For SC 1 and SC 2 pre-existing software elements these functional profiles shall describe the functionality of all combinations of the following items:
 - i. input data combinations;
 - ii. sequences of execution of the reused software function(s);
 - iii. timing relations within sequences of execution of the reused software function(s).
 - 2) For SC 3 and SC 4 pre-existing software elements these functional profiles shall include in addition to bullet 1) above:
 - i. the internal state of the pre-existing software element;
 - ii. the detection of combinations which are not part of the proven in use claim together with appropriate reactions to ensure that the safety function is able to respond to a demand or the equipment under control is commanded to a safe state.

NOTE 5 The functional profile takes into consideration the following:

- hardware (e.g. processor, memory, clock, bus behaviour) and demand profiles;
- configuration (e.g., compiler options used in compiling the source code for the proposed use, initialisation of program variables and constants, the hardware configuration on which the software will execute);
- software interfaces;
- libraries (including source code libraries as well as libraries of binary code);
- operating system, interpreters (for example, those used to emulate processor architectures on processors which do not share that architecture);
- translator (compiler), linker, code generators.

c) The functional profile required to be defined by b) shall include the following:

- 1) a complete description of the conditions of use of the pre-existing software;

NOTE 6 The conditions of use (functional profile) includes the factors that can trigger systematic faults in the software of the element, such as additional modes of use, functions performed, human factors.

NOTE 7 This information can be located in the safety manual.

- 2) Documented relation between the functions provided by the pre-existing software element and the safety requirements of the E/E/PE safety-related system;
- 3) Specification of the configuration;
- 4) Specification of the capacity and response time;
- 5) Specification of the used interfaces of the pre-existing software element to non safety-related functions, to other software and to the PE system;
- 6) application of the requirements of 7.2.2.11, 7.2.2.12 and 7.2.2.13 and 7.2.2.14 of this document.

- d) Evidence shall be given that functionality of the pre-existing software that is not included in the functional profile defined for the intended use in b), has no adverse effect on the safety integrity of the E/E/PE safety-related system;

NOTE 8 This requirement can be achieved by ensuring that the reused software functions are physically or electrically disabled or by demonstrating that the input data for the software are restricted in a way that these functions are excluded from the operational configuration, or by other forms of arguments and evidence.

- e) Differences between the previous and intended functional profiles defined in b) shall be documented. This includes the identification of the functionality from the pre-existing software which is not used as out of scope. An impact analysis on the differences shall be carried out and documented using a combination of appropriate analytical methods as well as testing and validation. The impact analysis shall demonstrate that the likelihood of systematic faults which could cause dangerous failures is low enough that the required systematic capability of the safety function(s) using the element is achieved. The validation shall include the validation of the intended functionality in the presence of the functionality which is not used from the pre-existing software to be sure that there is no unintended impact to the used functionality of the pre-existing software;

- f) Differences between the functional profile defined for the previous use and the intended use shall be justified and documented;

- g) The pre-existing software element together with the hardware on which it will run shall be verified subject to the following:

- 1) analysis of any operational experience of the integrated hardware and software element;
- 2) suitability analysis of the hardware and software element;
- 3) testing of the hardware and software element including:
 - i) specification of the test cases from the properties documented in b) and f);
 - ii) details of the test result for each test case;
 - iii) evidence that 100% structural test coverage of the functional profiles is reached for the parts of the functional profile where no evidence is available from the previous use.

NOTE 9 Suitability analysis and testing focuses on the demonstration of a hardware and software element's performance within the intended application. The results of any existing analysis and testing can be taken into account. This includes functional behaviour, accuracy, behaviour in the case of a fault, time response, response to overload, usability (e.g., avoidance of human error) and maintainability.

NOTE 10 A mathematical partitioning of the input data can be helpful to identify all proven in use combinations.

NOTE 11 By "input data" is meant all input data to the software element regardless of its source, for example input data from the hardware.

NOTE 12 The timing relations most often found and most in need of verification are linear timing relations of the form (fastest possible time is less than or equal to the execution time which is less than or equal to the longest possible time). There are practical methods for verifying and checking the mutual consistency of timing requirements of such a form.

NOTE 13 The collection of evidence for proven in use elements requires an effective system for reporting failures.

NOTE 14 See IEC 61508-7 Annex D for guidelines on statistical evaluation techniques for pre-existing software elements based on operational experience.

- h) After any modification at all, the formerly proven in use software shall no longer be taken to be proven in use. Any future modification to proven in use software shall comply with the requirements of 7.8;

- i) Use of the pre-existing software with input data beyond its pre-defined ranges shall be subject to a new proven in use claim;

- j) Provision of a safety manual (see IEC 61508-2 Annex D and Annex D of this document) that gives a sufficiently precise and complete description of the element to make an

1068 assessment possible of the integrity of a specific safety function that depends wholly or
1069 partly on the pre-existing software element.

1070 NOTE 15 See also 7.4.2.12 b) NOTE 4.

1071 NOTE 16 The justification of the pre-existing software element can be developed during safety planning (see
1072 Clause 6).

1073 7.4.2.14 To comply with Route 3_s a pre-existing software element shall meet all of the
1074 following requirements:

1075 a) The software safety requirements specification for the pre-existing software element in
1076 its new application shall be documented to the same degree of precision as would be
1077 required by this document for any safety related element of the same systematic
1078 capability. The software safety requirements specification shall cover the functional and
1079 safety behaviour as applicable to the element in its new application and as specified in
1080 7.2 and Table A.1;

1081 b) The justification for use of a software element shall provide evidence that the desirable
1082 safety properties identified in 7.2.2, 7.4.3, 7.4.4, 7.4.5, 7.4.6, 7.4.7, 7.5.2, 7.7.2, 7.8.2
1083 7.9.2 and 8 and have been considered, taking account of the guidance in Annex C;

1084 c) The pre-existing software element's design shall be documented to a degree of
1085 precision, sufficient to provide evidence of compliance with the requirement specification
1086 and the required systematic capability. See 7.4.3, 7.4.5 and 7.4.6 and Table A.2 and
1087 Table A.4.

1088 d) The evidence required in 7.4.2.14 a) and 7.4.2.14 b) shall meet the needs of the
1089 hardware-software interface specification. See IEC 61508-2 7.4.2.10, 7.4.2.11 and
1090 7.4.2.12;

1091 e) There shall be evidence that the pre-existing software element has been subject to
1092 verification and validation using a systematic approach with documented testing and
1093 review of all parts of the element's design and code. See 7.4.7, 7.4.8, 7.5, 7.7 and 7.9
1094 and Table A.5, Table A.6, Table A.7 and Table A.9 and related tables in Annex B;

1095 NOTE 1 Positive operational experience can be used to satisfy black-box and probabilistic testing requirements
1096 (see Table A.7 and the relevant Table in Annex B).

1097 f) Where the pre-existing software element provides functions which are not required in
1098 the safety related system, then evidence shall be provided that the unwanted functions
1099 will not prevent the E/E/PE system from meeting its safety requirements;

1100 NOTE 2 Ways to meet this requirement include:

- 1101 • removing the functions from the build;
- 1102 • disabling the functions;
- 1103 • appropriate system architecture (e.g. partitioning, wrappers, diversity, checking the credibility of outputs);
- 1104 • extensive testing.

1105 g) There shall be evidence that all credible failure mechanisms of the pre-existing software
1106 element have been identified and that appropriate mitigation measures have been
1107 implemented;

1108 NOTE 3 Appropriate mitigation measures include:

- 1109 • appropriate system architecture (e.g. partitioning, wrappers, diversity, credibility of checking of outputs);
- 1110 • exception handling.

1111 h) The planning for use of the pre-existing software element shall identify the configuration
1112 of the software element, the software and hardware run-time environment and if
1113 necessary the configuration of the compilation / linking system;

1114 i) The justification for use of the pre-existing software element shall be valid for only those
1115 applications which respect the assumptions made in the compliant item safety manual
1116 for the element (see IEC 61508-2 Annex D and Annex D of this document);

1117 7.4.2.15 Where a pre-existing software element includes the use of data and data generation
1118 languages the following shall apply as appropriate:

1119 NOTE See Annex G for guidance on systems configured by application data and Annex J for guidance on data
1120 driven systems.

1121 a) Where a PE system consists of pre-existing functionality that is configured by data to
1122 meet specific application requirements, the design of the application software shall be
1123 commensurate with the degree of application configurability, pre-delivered existing
1124 functionality and complexity of the PE safety-related system;

1125 b) Where the safety-related functionality of a PE system is determined significantly or
1126 predominantly by configuration data, appropriate techniques and measures shall be
1127 used according to the required systematic capability to prevent the introduction of faults
1128 during the design, production, loading and modification of the configuration data and to
1129 ensure that the configuration data correctly states the application logic. This shall include
1130 impact analysis and regression testing after changes of data;

1131 NOTE Software is both code and data. Regression testing is equally applicable to code and data changes. See
1132 IEC 61508-7 C.5.25 for more information.

1133 c) The specification of data structures shall be:

1134 1) consistent with the functional requirements of the system, including the application
1135 data;

1136 2) complete;

1137 3) self consistent;

1138 4) such that the data structures are protected against alteration or corruption.

1139 d) Where a PE System consists of pre-existing functionality that is configured by data to
1140 meet specific application requirements, the configuration process itself shall be
1141 documented appropriately.

1142 7.4.3 Requirements for software architecture design

1143 7.4.3.1 Depending on the nature of the software development, responsibility for conformance
1144 with this document can rest with multiple parties. The division of responsibility shall be
1145 documented during safety planning (see IEC 61508-1 6.2.19).

1146 NOTE 1 In some contexts of user application programming, for example in PLCs (see IEC 61508-6 Annex E.2), the
1147 software architecture is provided by the supplier as a standard feature of the product. In this case:

1148 a) the supplier would, under this document, be required to assure the user of the compliance of his products
1149 to the requirements of 7.4

1150 b) the user would develop the application software by using the standard programming facilities (for example
1151 in ladder logic, see Annex G) in accordance with the requirements of 7.4.3 to 7.4.8.

1152 c) the requirement to define and document the software architecture can be seen as information that the user
1153 would use to select the PLC (or equivalent) for the application.

1154 NOTE 2 In certain industrial sectors the software architecture would be called a function description or functional
1155 design specification (although these documents can also include the hardware).

1156 7.4.3.2 The software architecture design shall be established by the software supplier and/or
1157 developer and shall be detailed. The software architecture design shall:

1158 a) define the major elements and subsystems of the software, how they are interconnected,
1159 and how the required attributes, particularly systematic capability, will be achieved. It
1160 also defines the overall behaviour of the software, and how software elements interface
1161 and interact;

1162 NOTE 1 Examples of major software elements include operating systems, databases, EUC input/output
1163 subsystems, communication subsystems, application program(s), programming and diagnostic tools, etc.

1164 NOTE 2 From a safety viewpoint, the software architecture phase is where the basic safety strategy is
1165 developed for the software.

NOTE 3 Although the IEC 61508 series sets numerical target failure measures for safety functions carried out by E/E/PE safety-related systems, systematic capability is usually unquantified (see IEC 61508-4 3.5.8) while systematic capability is defined on a confidence scale of SC1 to SC4. This document recognizes that a software failure can be safe or unsafe depending on the specific use of the software. The system/software architecture needs to be such that unsafe failures of an element are limited by some architectural constraint, and that development methods should take account of these constraints. This document applies development and validation techniques with rigour that is qualitatively consistent with the required systematic capability.

- b) specify and justify an integrated set of techniques and measures necessary during the software safety lifecycle phases to satisfy the software safety requirements specification at the required systematic capability (see 7.1.2.7). These techniques and measures include software design strategies for both fault tolerance (consistent with the hardware) and fault avoidance, including (where appropriate) redundancy and diversity;

NOTE 4 Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and Annex B techniques and measures.

- c) be based on a partitioning into elements/subsystems, for each of which the following information shall be provided:

1) whether the elements/subsystems have been previously verified, and if yes, their verification conditions;

2) whether each subsystem/element is safety-related or not;

3) systematic capability of the subsystem/element.

- d) determine all software/hardware interactions and evaluate and detail their significance;

NOTE 5 Where the software/hardware interaction is already determined by the system architecture, it is sufficient to refer to the system architecture.

NOTE 6 This includes interactions determined within the hardware-software interface specification as specified in IEC 61508-2 7.4.2.11.

- e) use a notation to represent the architecture which:

i) is unambiguously defined;

ii) matches the notation used for architecture representation according to IEC 61508-2 7.4.4;

- f) select the design features to be used for maintaining the systematic capability of all data. Such data can include plant input-output data, communications data, operator interface data, maintenance data and internal database data;

- g) specify appropriate software architecture integration tests to ensure that the software architecture satisfies the software safety requirements specification at the required systematic capability.

7.4.3.3 The following properties of the software should be considered during development of the software architecture design:

a) completeness with respect to software safety requirements specification;

b) correctness with respect to software safety requirements specification;

c) freedom from intrinsic design faults;

d) simplicity and understandability;

e) predictability of behaviour;

f) verifiable and testable design;

g) fault tolerance;

h) defence against common cause failure from external events.

7.4.3.4 The software architecture design shall be analysed to identify potential failures of software functions or software elements which could lead to violations of software safety requirements, and to define measures against them and/or evaluate existing measures regarding their effectiveness.

NOTE 1 The approach to software failure analysis depends on:

- failure analysis already performed on system level;

- 1218 • the complexity of the safety functions;
- 1219 • the software design realisation;
- 1220 • the presence of software of different systematic capabilities;
- 1221 • the overall safety integrity requirements;
- 1222 • the planned verification strategies, activities and techniques.

1223 NOTE 2 For further information on foreseeable failures, methods and techniques for software failure analysis see
1224 IEC 61508-7 Annex C.6.2.

1225 NOTE 3 The software failure analysis can also provide input to test case definition – see 7.4.8, 7.5 and 7.7.

1226 7.4.3.5 Any changes required to the E/E/PE System Safety Requirements Specification (see
1227 7.2.2) after applying 7.4.3.2 shall be agreed with the E/E/PE developer and documented.

1228 NOTE There will inevitably be iteration between the hardware and software architecture (see Figure 5) and there is
1229 therefore a need to discuss with the hardware developer such issues as the test specification for the integration of
1230 the programmable electronics hardware and the software (see 7.5).

1231 7.4.4 Requirements for support tools, including programming languages

1232 7.4.4.1 If the safety lifecycle incorporates the use of one or more software off-line support tool(s)
1233 for the development of software forming part of a safety function, such that activities or tasks
1234 required by this standard rely on the correct functioning of those tool(s), then these tools shall
1235 comply with the requirements of Annex H.

1236 7.4.4.2 A software on-line support tool shall be considered to be a software element of the
1237 safety-related system.

1238 NOTE See IEC 61508-4 3.2.10 and 3.2.11 for examples of software on-line support tools and off-line support tools.

1239 7.4.4.3 Software off-line support tools shall be selected as a coherent part of the software
1240 development activities.

1241 NOTE See 7.1.2 for software development lifecycle requirements.

1242 7.4.4.4 Software off-line support tools should be selected to be integrated.

1243 NOTE In this context, tools are integrated if they work co-operatively such that the outputs from one tool have
1244 suitable content and format for automatic input to a subsequent tool, thus minimising the possibility of introducing
1245 human error in the reworking of intermediate results.

1246 7.4.4.5 The availability of suitable tools to supply the services that are necessary over the whole
1247 lifetime of the E/E/PE safety-related system should be considered.

1248 EXAMPLE Tools to support specification, design, implementation, documentation, modification.

1249 7.4.4.6 Consideration should be given to the competence of the users of the selected tools.

1250 NOTE See IEC 61508-1 6.2 for competence requirements.

1251 7.4.4.7 The selection of the software off-line support tools shall be justified.

1252 7.4.4.8 To the extent required by the systematic capability, the software or design
1253 representation (including a programming language) selected shall:

- 1254 a) have a translator which has been assessed as a software off-line support tool;
- 1255 b) use only defined language features;
- 1256 c) match the characteristics of the application;
- 1257 d) contain features that facilitate the detection of design or programming mistakes;
- 1258 e) support features that match the design method.

1259 NOTE 1 A programming language is a class of software or design representations. See IEC 61508-4 3.2.20 for the
1260 definition of translator.

1261 NOTE 2 The assessment of a translator may be performed for a specific application project, or for a class of
1262 applications. For off-line translators, see Annex H.

1263 NOTE 3 A validation suite (i.e. a set of test programs whose correct translation is known in advance) can be used to
1264 evaluate the fitness for purpose of a translator according to defined criteria. Such criteria include functional and non-
1265 functional requirements. For the functional translator requirements, dynamic testing can be a main validation
1266 technique. Automatic testing suites can enhance the rigour of the validation.

1267 7.4.4.9 Where 7.4.4.8 cannot be fully satisfied, the fitness for purpose of the language, and
1268 any additional measures which address any identified shortcomings of the language shall be
1269 justified.

1270 7.4.4.10 Programming languages for the development of all safety-related software shall be
1271 used according to a suitable programming language coding standard.

1272 NOTE See IEC 61508-7 C.2.6.2 for guidance on coding standard aspects that relate to software safety.

1273 7.4.4.11 A programming language coding standard shall specify good programming practice,
1274 proscribe unsafe language features (for example, undefined language features, unstructured
1275 designs, etc.), promote code understandability, facilitate verification and testing, and specify
1276 procedures for source code documentation. Where practicable, the following information shall
1277 be contained in the source code:

1278 a) legal entity (for example company, author(s), etc.);

1279 b) description;

1280 c) inputs and outputs;

1281 d) configuration management history.

1282 7.4.4.12 Depending on the nature of the software development, responsibility for conformance
1283 with 7.4.4 can rest with multiple parties. The division of responsibility shall be documented
1284 during safety planning (see IEC 61508-1 6.2.19).

1285 **7.4.5 Requirements for detailed design and development – software system design**

1286 7.4.5.1 Depending on the nature of the software development, responsibility for conformance
1287 with 7.4.5 can rest with multiple parties. The division of responsibility shall be documented
1288 during safety planning (see IEC 61508-1 6.2.19).

1289 7.4.5.2 The following information shall be available prior to the start of detailed design: the
1290 specification of requirements for the E/E/PE safety related system; the software architecture
1291 design; the validation plan for software aspects of system safety.

1292 NOTE Detailed design is defined here to mean software system design: the partitioning of the major elements in
1293 the architecture into a system of software modules; individual software module design; and coding. In small
1294 applications, software system design and architectural design can be combined.

1295 7.4.5.3 The software should be designed in a structured way, including organising the software
1296 into a modular structure that separates out (as far as possible) safety-related parts; including
1297 range checking and other features that provide protection against data input mistakes; using
1298 previously verified software modules; and providing a design that facilitates future software
1299 modifications.

1300 7.4.5.4 The software shall be produced to achieve modularity, testability, and the capability
1301 for safe modification.

1302 NOTE The nature of detailed design and development will vary with the nature of the software development activities
1303 and the software architecture (see 7.4.3). In some contexts of application programming, for example ladder logic and
1304 function blocks, detailed design can be considered as configuring rather than programming.

1305 7.4.5.5 The following properties of the software should be considered during development of
1306 the design and development specifications:

- 1307 a) completeness with respect to software safety requirements specification;
- 1308 b) correctness with respect to software safety requirements specification;
- 1309 c) freedom from intrinsic design faults;
- 1310 d) simplicity and understandability
- 1311 e) predictability of behaviour;
- 1312 f) verifiable and testable design;
- 1313 g) fault tolerance / fault detection;
- 1314 h) freedom from common cause failure.

1315 NOTE 1 Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a
1316 description of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the
1317 Annex A and Annex B techniques and measures.

1318 NOTE 2 The hardware-software interface specification can contain information for establishing correct requirements.

1319 7.4.5.6 For each major element/subsystem in the software architecture design, further
1320 refinement of the design shall be based on a partitioning into software modules (i.e. the
1321 specification of the software system design). The design of each software module and the
1322 verification to be applied to each software module shall be specified.

1323 NOTE 1 For pre-existing software elements, see 7.4.2.12

1324 NOTE 2 Verification includes testing and analysis.

1325 7.4.5.7 Appropriate software system integration tests shall be specified to ensure that the
1326 software system satisfies the software safety requirements specification at the required
1327 systematic capability.

1328 7.4.6 Requirements for code implementation

1329 7.4.6.1 To the extent required by the systematic capability, the source code shall possess the
1330 following properties

- 1331 a) satisfy the specified requirements for software module design (see 7.4.5);
- 1332 b) satisfy the specified requirements of the coding standards (see 7.4.4);
- 1333 c) satisfy all relevant requirements specified during safety planning (see Clause 6).

1334 7.4.6.2 To the extent required by the systematic capability, the source code should be readable,
1335 understandable and testable.

1336 NOTE Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description
1337 of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and
1338 Annex B techniques and measures.

1339 7.4.6.3 Each module of software code shall be reviewed. Where the code is produced by an
1340 automatic tool, the requirements of 7.4.4 shall be met. Where the source code consists of
1341 reused pre-existing software, the requirements of 7.4.2.12 shall be met.

1342 NOTE Code review is a software verification activity (see 7.9). Code review can be carried out by means of an
1343 inspection of the code: (1) by an individual; (2) by a software walk-through (see IEC 61508-7 C.5.15); or (3) by a
1344 formal inspection (see IEC 61508-7 C.5.14), in increasing order of rigour.

1345 7.4.7 Requirements for software module testing

1346 7.4.7.1 Each software module shall be verified as required by the software module test
1347 specification that was developed during software system design (see 7.4.5).

1348 NOTE 1 Verification includes testing and analysis.

1349 NOTE 2 Testing that the software module correctly satisfies its test specification is a verification activity (see 7.9).
1350 It is the combination of code review and software module testing that provides assurance that a software module
1351 satisfies its associated specification, i.e. it is verified.

1352 7.4.7.2 This verification shall show whether or not each software module performs its intended
1353 function and does not perform unintended functions.

1354 NOTE 1 This does not imply testing of all input combinations, nor of all output combinations. Testing all equivalence
1355 classes or structure based testing can be sufficient. Boundary value analysis or control flow analysis can reduce the
1356 test cases to an acceptable number. Analysable programs make the requirements easier to fulfil. See IEC 61508-7
1357 Annex C for these techniques.

1358 NOTE 2 Where the development uses formal methods, such tests can be reduced in scope. See IEC 61508-7 Annex
1359 C for these techniques.

1360 NOTE 3 Although systematic capability is defined on a confidence scale of SC1 to SC4 (see IEC 61508-4 3.5.8),
1361 quantified statistical evidence is acceptable if all the relevant conditions in IEC 61508-7 Annex D are satisfied.

1362 NOTE 4 If the module is simple enough to make practicable an exhaustive test, then this can be the most efficient
1363 way to demonstrate conformance.

1364 7.4.7.3 The following properties of the software should be considered during software module
1365 testing:

- 1366 a) completeness of testing with respect to the software design specification;
- 1367 b) correctness of testing with respect to the software design specification (successful
1368 completion);
- 1369 c) repeatability;
- 1370 d) precisely defined testing configuration.

1371 NOTE Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description
1372 of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and
1373 Annex B techniques and measures.

1374 7.4.7.4 The results of the software module testing shall be documented.

1375 7.4.7.5 The procedures for corrective action on not passing the test shall be specified.

1376 7.4.8 Requirements for software integration testing

1377 7.4.8.1 Software integration tests shall be specified during the design and development phase
1378 (see 7.4.5).

1379 NOTE Testing that the software is correctly integrated is a verification activity (see 7.9).

1380 7.4.8.2 The software system integration test specification shall state the following:

- 1381 a) the division of the software into manageable integration sets;
- 1382 b) test cases and test data;
- 1383 c) types of tests to be performed;
- 1384 d) test environment, tools, configuration and programs;
- 1385 e) test criteria on which the completion of the test will be judged;
- 1386 f) procedures for corrective action on failure of test.

1387 7.4.8.3 The software shall be tested in accordance with the software integration tests specified
1388 in the software system integration test specification. These tests shall show that all software
1389 modules and software elements/subsystems interact correctly to perform their intended function
1390 and do not perform unintended functions.

1391 NOTE 1 This does not imply testing of all input combinations, nor of all output combinations. Testing all equivalence
1392 classes or structure based testing can be sufficient. Boundary value analysis or control flow analysis can reduce the

1393 test cases to an acceptable number. Analysable programs make the requirements easier to fulfil. See IEC 61508-7
1394 C.5 for these techniques.

1395 NOTE 2 Where regression tests are necessary (see IEC 61508-7, C.5.25) appropriate regression testing shall be
1396 conducted to demonstrate that defects have not been introduced into previously integrated software.

1397 NOTE 3 Where the development uses formal methods, such tests can be reduced in scope. See IEC 61508-7 B.2.2
1398 for these techniques.

1399 NOTE 4 Although systematic capability is defined on a confidence scale of SC1 to SC4 (see IEC 61508-4 3.5.8),
1400 quantified statistical evidence is acceptable if all the relevant conditions in IEC 61508-7 Annex D are satisfied.

1401 7.4.8.4 The results of software integration testing shall be documented, stating the test results,
1402 and whether the objectives and the test criteria have been met. If there is a failed integration
1403 test, the reasons for the failure shall be documented.

1404 7.4.8.5 During software integration, any modification to the software shall be subject to an
1405 impact analysis which shall determine all software modules impacted, and the necessary re-
1406 verification, regressions tests and re-design activities.

1407 **7.5 Programmable electronics integration (hardware and software)**

1408 **7.5.1 Objectives**

1409 7.5.1.1 The first objective of the requirements of this subclause is to integrate the software
1410 onto the target programmable electronic hardware.

1411 7.5.1.2 The second objective of the requirements of this subclause is to combine the software
1412 and hardware in the safety-related programmable electronics to ensure their compatibility and
1413 to meet the requirements of the intended systematic capability of the element.

1414 NOTE 1 Testing that the software is correctly integrated with the programmable electronic hardware is a verification
1415 activity (see 7.9).

1416 NOTE 2 Depending on the nature of the application, these activities can be combined with 7.4.8.

1417 NOTE 3 This phase is box 10.4 of Figure 4.

1418 **7.5.2 Requirements**

1419 7.5.2.1 Integration tests shall be specified during the design and development phase (see
1420 7.4.3) to ensure the compatibility of the hardware and software in the safety-related
1421 programmable electronics.

1422 NOTE Close co-operation with the developer of the E/E/PE system can be required in order to develop the
1423 integration tests.

1424 7.5.2.2 The software/PE integration test specification (hardware and software) shall state the
1425 following:

- 1426 a) the split of the system into integration levels;
- 1427 b) test cases and test data;
- 1428 c) types of tests to be performed;
- 1429 d) test environment including tools, support software and configuration description;
- 1430 e) test criteria on which the completion of the test will be judged.

1431 7.5.2.3 The software/PE integration test specification (hardware and software) shall
1432 distinguish between those activities which can be carried out by the developer on his premises
1433 and those that require access to the user's site.

1434 7.5.2.4 The software/PE integration test specification (hardware and software) shall
1435 distinguish between the following activities:

- 1436 a) merging of the software system on to the target programmable electronic hardware;
1437 b) E/E/PE integration, i.e. adding interfaces such as sensors and actuators;
1438 c) applying the E/E/PE safety-related system to the EUC.

1439 NOTE Items b) and c) are covered by IEC 61508-1 and IEC 61508-2 and are included here to put item a) in context
1440 and for completeness. They are not normally the responsibility of the software developers.

1441 7.5.2.5 The software shall be integrated with the safety-related programmable electronic
1442 hardware in accordance with the software/PE integration test specification (hardware and
1443 software).

1444 7.5.2.6 The following properties of the software should be considered during integration:

- 1445 a) completeness of integration with respect to the design specifications;
1446 b) correctness of integration with respect to the design specifications (successful
1447 completion);
1448 c) repeatability;
1449 d) precisely defined integration configuration.

1450 NOTE Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description
1451 of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and
1452 Annex B techniques and measures.

1453 7.5.2.7 During the integration testing of the safety-related programmable electronics
1454 (hardware and software), any change to the integrated system shall be subject to an impact
1455 analysis. The impact analysis shall determine all software modules impacted, and the necessary
1456 re-verification activities.

1457 7.5.2.8 Test cases and their expected results shall be documented for subsequent analysis.

1458 7.5.2.9 The integration testing of the safety-related programmable electronics (hardware and
1459 software) shall be documented, stating the test results, and whether the objectives and the test
1460 criteria have been met. If there is a failure, the reasons for the failure shall be documented. Any
1461 resulting modification or change to the software shall be subject to an impact analysis which
1462 shall determine all software elements/modules impacted, and the necessary re-verification and
1463 re-design activities.

1464 7.6 Software operation and modification procedures

1465 7.6.1 Objective

1466 The objective of the requirements of this subclause is to provide information and procedures
1467 concerning software necessary to ensure that the functional safety of the E/E/PE safety-related
1468 system is maintained during operation and modification.

1469 NOTE This phase is box 10.5 of Figure 4.

1470 7.6.2 Requirements

1471 The requirements are given in IEC 61508-2 7.6 and in 7.8 of this document.

1472 NOTE 1 In this document software (unlike hardware) is not capable of being maintained: it is always modified.
1473 However, software can play a role in system or hardware maintenance.

1474 NOTE 2 Similarly, all software failures are due to design faults, since software consists solely of design and does
1475 not wear out or suffer from physical failure.

7.7 Software aspects of system safety validation

7.7.1 Objective

7.7.1.1 The objective of the requirements of this subclause is to ensure that the integrated system complies with the software safety requirements specification at the required systematic capability.

NOTE 1 This phase is box 10.6 of Figure 4.

NOTE 2 Software usually cannot be validated separately from its underlying hardware and system environment.

7.7.2 Requirements

7.7.2.1 If the compliance with the requirements for safety-related software has already been established in the safety validation planning for the E/E/PE safety-related system (see IEC 61508-2 7.7), then the validation need not be repeated.

7.7.2.2 The validation activities shall be carried out as specified in the prepared validation plan (see 7.3) for software aspects of E/E/PE system safety.

7.7.2.3 The following properties of the software should be considered during safety validation:

- a) completeness of validation with respect to the software design specification;
- b) correctness of validation with respect to the software design specification (successful completion);
- c) repeatability;
- d) precisely defined validation configuration.

NOTE Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and Annex B techniques and measures.

7.7.2.4 Depending on the nature of the software development, responsibility for conformance with 7.7 can rest with multiple parties. The division of responsibility shall be documented during safety planning (see IEC 61508-1 6.2.19).

7.7.2.5 Documentation of the software aspects of system safety validation shall be produced which shall state for each safety function:

- a) a chronological record of the validation activities that will permit the sequence of activities to be retraced;

NOTE When recording test results, it is important to be able to retrace the sequence of activities. The emphasis of this requirement is on retracing a sequence of activities, and not on producing a timed/dated list of documents.

- b) the version of the validation plan for software aspects of system safety (see 7.3) being used;
- c) the safety function being validated (by test or analysis), together with reference to the validation plan for software aspects of system safety;
- d) tools and equipment used together with calibration data;
- e) the results of the validation activity;
- f) discrepancies between expected and actual results.

7.7.2.6 When discrepancies occur between expected and actual results, the analysis made and the decisions taken on whether to continue the validation, or to issue a change request and return to an earlier part of the development lifecycle, shall be documented as part of the results of validating the software aspects of system safety.

7.7.2.7 The validation of safety-related software aspects of system safety shall meet the following requirements:

1519 a) testing shall be the main validation method for software; analysis, animation and modelling
1520 can be used to supplement the validation activities;

1521 b) the software shall be exercised by simulation of:

1522 1) input signals present during normal operation;

1523 2) anticipated occurrences;

1524 3) undesired conditions requiring system action;

1525 the supplier and/or developer (or the multiple parties responsible for compliance) shall make
1526 available the documented results of the validation of software aspects of system safety and all
1527 pertinent documentation to the system developer to enable his product to meet the requirements
1528 of IEC 61508-1 and IEC 61508-2.

1529 NOTE The requirements of 7.7.2.2 and 7.7.2.5 to 7.7.2.7 are based on the general requirements given in
1530 IEC 61508-1 7.14.

1531 7.7.2.8 Software tools shall meet the requirements of 7.4.4.

1532 7.7.2.9 The results of the validation of safety-related software aspects of system safety shall
1533 meet the following requirements:

1534 a) the tests shall show that all of the specified requirements for safety-related software (see
1535 7.2) are correctly met and the software does not perform unintended functions;

1536 b) test cases and their results shall be documented for subsequent analysis and independent
1537 assessment (see Clause 8) as required by the systematic capability requirements;

1538 c) the documented results of validating the software aspects of system safety shall state either
1539 (1) that the software has passed the validation or (2) the reasons for not passing the
1540 validation.

1541 7.7.2.10 For the avoidance of faults during the E/E/PE system safety validation of software
1542 aspects an appropriate selection of techniques and measures according to Table A.7 shall be
1543 used.

1544 7.8 Software modification

1545 7.8.1 Objective

1546 The objective of the requirements of this subclause is to guide corrections, enhancements or
1547 adaptations to the software that forms a part of an installed and validated system, ensuring that
1548 the required systematic capability is sustained.

1549 NOTE This phase is Box 10.5 of Figure 4.

1550 7.8.2 Requirements

1551 7.8.2.1 Prior to carrying out any software modification, software modification procedures shall
1552 be made available (see IEC 61508-1 7.16).

1553 NOTE 1 Subclauses 7.8.2.1 to 7.8.2.11 apply primarily to changes occurring during the operational phase of the
1554 software. They can also be applied during the programmable electronics integration and overall installation and
1555 commissioning phases (see IEC 61508-1 7.13).

1556 NOTE 2 An example of a modification procedure model is shown in IEC 61508-1 Figure 9.

1557 7.8.2.2 In the case of incremental lifecycle the development procedures should include the
1558 provisions to control the impact of the modification between successive versions of the software.

1559 7.8.2.3 The following properties of the software should be considered during modification:

1560 a) completeness of modification with respect to its requirements;

1561 a) correctness of modification with respect to its requirements;

- 1562 b) freedom from introduction of intrinsic design faults;
1563 c) avoidance of unwanted behaviour;
1564 d) verifiable and testable design;
1565 e) regression testing and verification coverage.

1566 NOTE Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description
1567 of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and
1568 Annex B techniques and measures.

1569 7.8.2.4 A modification shall be initiated only on the issue of an authorized software modification
1570 request under the procedures specified during safety planning (see Clause 6) which details the
1571 following:

- 1572 a) the hazards which can be affected;
1573 b) the proposed modification;
1574 c) the reasons for modification.

1575 NOTE A request for modification can arise from, for example

- 1576 • functional safety is found to be less than required by the safety requirements specification;
- 1577 • systematic fault experience;
- 1578 • new or amended safety legislation;
- 1579 • modifications to the EUC or its use;
- 1580 • modification to the overall safety requirements;
- 1581 • analysis of operations and maintenance performance, indicating that the performance is below target;
- 1582 • routine functional safety audits.

1583 7.8.2.5 An analysis shall be carried out on the impact of the proposed software modification
1584 on the functional safety of the E/E/PE safety-related system:

- 1585 a) to determine whether or not a hazard and risk analysis is required;
1586 b) to determine which software safety lifecycle phases will need to be repeated.

1587 NOTE 1 Management decision and authority can guide and enforce the use of administrative and technical controls.

1588 NOTE 2 At one extreme, an impact analysis can be based on an informal assessment. At the other extreme, an
1589 impact analysis can be based on a rigorous analysis using formal methods of the potential adverse impact of all
1590 proposed changes, including those changes which can be inadequately understood or implemented. See IEC 61508-
1591 7 C.5.23 for guidance on impact analysis.

1592 7.8.2.6 The impact analysis results obtained in this clause shall be documented.

1593 NOTE The documentation can be in the form of dedicated documents or of updates of existing documents.

1594 7.8.2.7 All modifications which have an impact on the functional safety of the E/E/PE safety-
1595 related system shall initiate a return to an appropriate phase of the software safety lifecycle. All
1596 subsequent phases shall then be carried out in accordance with the procedures specified for
1597 the specific phases in accordance with the requirements in this document. Safety planning (see
1598 Clause 6) shall detail all subsequent activities.

1599 NOTE 1 It can be necessary to implement a full hazard and risk analysis, which can generate a need for different
1600 safety integrity levels than currently specified for the safety functions implemented by the E/E/PE safety-related
1601 systems.

1602 NOTE 2 Where the safety-related functionality of an E/E/PE safety related system is determined significantly or
1603 predominantly by configuration data, the requirements of this subclause apply to corrections, enhancements or
1604 adaptations to the configuration data as well as to the validated software.

1605 7.8.2.8 The safety planning for the modification of safety-related software shall meet the
1606 requirements given in IEC 61508-1 Clause 6.2.1 and 6.2.2. In particular:

- 1607 a) identification of staff and specification of their required competency

- 1608 b) detailed specification for the modification;
1609 c) verification planning;
1610 d) scope of revalidation and testing of the modification to the extent required by the safety
1611 integrity level.

1612 NOTE 1 Depending on the nature of the application, involvement of domain experts can be important.

1613 NOTE 2 Guidance on regression validation is located in IEC 61508-7 C.5.25.

1614 7.8.2.9 Modification shall be carried out as planned.

1615 7.8.2.10 Details of all modifications shall be documented, including references to:

- 1616 a) the modification/retrofit request;
1617 b) the results of the impact analysis which assesses the impact of the proposed software
1618 modification on the functional safety, and the decisions taken with associated
1619 justifications;
1620 c) software configuration management history;
1621 d) deviation from normal operations and conditions;
1622 e) all documented information affected by the modification activity.

1623 7.8.2.11 Information on the details of all modifications shall be documented. The
1624 documentation shall include the re-verification, regression tests and re-validation of data and
1625 results.

1626 7.8.2.12 The assessment of the required modification or retrofit activity shall be dependent on
1627 the results of the impact analysis and the systematic capability.

1628 7.9 Software verification

1629 7.9.1 Objective

1630 The objective of the requirements of this subclause is, to the extent required by the safety
1631 integrity level, to test and evaluate the outputs from a given software safety lifecycle phase to
1632 ensure correctness and consistency with respect to the inputs to that phase, including the
1633 requirements of this standard.

1634 NOTE 1 This subclause considers the generic aspects of verification which are common to several safety lifecycle
1635 phases. This subclause does not place additional requirements for the testing element of verification in 7.4.7
1636 (software module testing), 7.4.8 (software integration) and 7.5 (programmable electronics integration) because these
1637 are verification activities in themselves. Nor does this subclause require verification in addition to software validation
1638 (see 7.7), because in this document software validation is the demonstration of conformance to the safety
1639 requirements specification. Checking whether the safety requirements specification is itself correct is carried out by
1640 domain experts.

1641 NOTE 2 Depending on the software architecture, responsibility for the verification activity can be split between all
1642 organisations involved in the development and modification of the software.

1643 7.9.2 Requirements

1644 7.9.2.1 The requirements of IEC 61508-1 7.18 regarding verification apply along with the
1645 subsequent clauses of this clause.

1646 7.9.2.2 The software verification planning shall refer to the criteria, measures, techniques and
1647 tools to be used in the verification activities, and shall address:

- 1648 a) the evaluation of the systematic capability;
1649 b) the selection and documentation of verification strategies, regression test strategies,
1650 activities and techniques;

- 1651 c) the selection and utilisation of verification tools (test harness, special test software,
1652 input/output simulators etc.);
- 1653 d) the evaluation of verification results;
- 1654 e) the corrective actions to be taken.

1655 7.9.2.3 The software verification shall be performed as planned.

1656 7.9.2.4 The following properties of the software should be considered during verification:

- 1657 a) completeness of verification with respect to the previous phase;
- 1658 b) correctness of verification with respect to the previous phase (successful completion);
- 1659 c) repeatability;
- 1660 d) precisely defined verification configuration.

1661 NOTE 1 Selection of techniques, measures for verification and the degree of independence of the verification
1662 activities will depend upon a number of factors. In some sectors techniques and measures can be specified in
1663 application sector standards. The factors can include, for example:

- 1664 • size of project;
- 1665 • degree of complexity;
- 1666 • degree of novelty of design;
- 1667 • degree of novelty of technology.

1668

1669 NOTE 2 Selection of techniques and measures is described in Annex A and Annex B. Annex C provides a description
1670 of the properties (see IEC 61508-7 Annex F) that can be expected in the software from the use of the Annex A and
1671 Annex B techniques and measures.

1672

1673 7.9.2.5 Evidence shall be documented to show that the phase being verified has, in all
1674 respects, been satisfactorily completed.

1675 7.9.2.6 All essential information from phase N of the software safety lifecycle needed for the
1676 correct execution of the next phase N+1 shall be available and shall be verified. Outputs from
1677 phase N include:

- 1678 a) adequacy of the specification, design, or code in phase N for:
- 1679 1) functionality;
- 1680 2) systematic capability, performance and other requirements of safety planning (see
1681 Clause 6);
- 1682 3) readability by the development team;
- 1683 4) testability for further verification;
- 1684 5) safe modification to permit further evolution;
- 1685 adequacy of the validation planning and/or tests specified for phase N for specifying and
1686 describing the design of phase N;
- 1687 check for incompatibilities between:
- 1688 1) the tests specified in phase N, and the tests specified in the previous phase N-1;
- 1689 2) the outputs within phase N.

1690 7.9.2.7 Subject to the choice of software development lifecycle (see 7.1), the following
1691 verification activities shall be performed:

- 1692 a) verification of software safety requirements;
- 1693 b) verification of software architecture;
- 1694 c) verification of software system design;
- 1695 d) verification of software module design;
- 1696 e) verification of code;

- 1697 f) verification of data;
- 1698 g) verification of timing performance;
- 1699 h) software module testing (see 7.4.7);
- 1700 i) software integration testing (see 7.4.8);
- 1701 j) programmable electronics integration testing (see 7.5);
- 1702 k) software aspects of system safety validation (see 7.7).

1703 NOTE 1 For requirements a) to g) [see 7.9.2.8 to 7.9.2.14.](#)

1704 NOTE 2 Testing includes regression testing.

1705 7.9.2.8 Verification of software safety requirements: after the software safety requirements
1706 specification has been completed, and before the next phase of software design and
1707 development begins, verification shall:

- 1708 a) consider whether the software safety requirements specification adequately fulfils the
1709 E/E/PE system safety requirements specification (see IEC 61508-1 7.10 and IEC 61508-
1710 2 7.2) for functionality, safety integrity, performance, and any other requirements of
1711 safety planning;
- 1712 b) consider whether the validation plan for software aspects of system safety adequately
1713 fulfils the software safety requirements specification;
- 1714 c) check for incompatibilities between:
 - 1715 1) the software safety requirements specification, and the E/E/PE system safety
1716 requirements specification (see IEC 61508-1 7.10 and IEC 61508-2 7.2);
 - 1717 2) the software safety requirements specification, and the validation plan for software
1718 aspects of system safety.

1719 7.9.2.9 Verification of software architecture: after the software architecture design has been
1720 completed, verification shall:

- 1721 a) consider whether the software architecture design adequately fulfils the software safety
1722 requirements specification;
- 1723 b) consider whether the integration tests specified in the software architecture design are
1724 adequate;
- 1725 c) consider whether the attributes of each major element/subsystem are adequate with
1726 reference to:
 - 1727 1) feasibility of the safety performance required;
 - 1728 2) testability for further verification;
 - 1729 3) readability by the development and verification team;
 - 1730 4) safe modification to permit further evolution.
- 1731 d) check for incompatibilities between the following:
 - 1732 1) the software architecture design, and the software safety requirements specification;
 - 1733 2) the software architecture design and its integration tests;
 - 1734 3) the software architecture design integration tests and the validation plan for software
1735 aspects of system safety.

1736 7.9.2.10 Verification of software system design: after the software system design has been
1737 completed, verification shall:

- 1738 a) consider whether the software system design (see 7.4.5) adequately fulfils the software
1739 architecture design;
- 1740 b) consider whether the specified tests of the software system integration (see 7.4.5)
1741 adequately fulfil the software system design (see 7.4.5);