BS EN 61508-3:2010

BSI Standards Publication

Functional safety of electrical/ electronic/programmable electronic safety-related systems

Part 3: Software requirements

bsi.

...making excellence a habit.™

Document Sharing Hub

Full document details: BS EN 61508-3:2010

National foreword

This British Standard is the UK implementation of EN 61508-3:2010. It is identical to IEC 61508-3:2010. It supersedes BS EN 61508-3:2002 which is withdrawn.

The UK participation in its preparation was entrusted by Technical Committee GEL/65, Measurement and control, to Subcommittee GEL/65/1, System considerations.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

© BSI 2010

ISBN 978 0 580 56235 8

ICS 13.260; 25.040.40; 29.020; 35.080

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 June 2010.

Amendments issued since publication

Amd. No.	Date	Text affected

Documents
Document Sharing Hub

Full document details: BS EN 61508-3:2010

EUROPEAN STANDARD

NORME EUROPÉENNE

EUROPÄISCHE NORM

EN 61508-3

May 2010

ICS 25.040.40

Supersedes EN 61508-3:2001

English version

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

(IEC 61508-3:2010)

Sécurité fonctionnelle des systèmes électriques/électroniques/électroniques programmables relatifs à la sécurité - Partie 3: Exigences concernant les logiciels
(CEI 61508-3:2010)

Funktionale Sicherheit sicherheitsbezogener elektrischer/elektronischer/programmierbarer elektronischer Systeme - Teil 3: Anforderungen an Software
(IEC 61508-3:2010)

This European Standard was approved by CENELEC on 2010-05-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Management Centre: Avenue Marnix 17, B - 1000 Brussels

© 2010 CENELEC - All rights of exploitation in any form and by any means reserved worldwide for CENELEC members.

Ref. No. EN 61508-3:2010 E

Documents
Document Sharing Hub

Full document details: BS EN 61508-3:2010

Foreword

The text of document 65A/550/FDIS, future edition 2 of IEC 61508-3, prepared by SC 65A, System aspects, of IEC TC 65, Industrial-process measurement, control and automation, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61508-3 on 2010-05-01.

This European Standard supersedes EN 61508-3:2001.

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

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN and CENELEC shall not be held responsible for identifying any or all such patent rights.

The following dates were fixed:

– latest date by which the EN has to be implemented at national level by publication of an identical national standard or by endorsement	(dop)	2011-02-01
– latest date by which the national standards conflicting with the EN have to be withdrawn	(dow)	2013-05-01

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61508-3:2010 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

[1] IEC 61511 series	NOTE	Harmonized in EN 61511 series (not modified).
[2] IEC 62061	NOTE	Harmonized as EN 62061.
[3] IEC 61800-5-2	NOTE	Harmonized as EN 61800-5-2.
[4] IEC 61508-5:2010	NOTE	Harmonized as EN 61508-5:2010 (not modified).
[5] IEC 61508-6:2010	NOTE	Harmonized as EN 61508-6:2010 (not modified).
[6] IEC 61508-7:2010	NOTE	Harmonized as EN 61508-7:2010 (not modified).
[7] IEC 60601 series	NOTE	Harmonized in 60601 series (partially modified).
[8] IEC 61131-3	NOTE	Harmonized as EN 61131-3.

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Annex ZA
(normative)

Normative references to international publications with their corresponding European publications

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

NOTE When an international publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

Publication	Year	Title	EN/HD	Year
IEC 61508-1	2010	Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements	EN 61508-1	2010
IEC 61508-2	2010	Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems	EN 61508-2	2010
IEC 61508-4	2010	Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 4: Definitions and abbreviations	EN 61508-4	2010
IEC Guide 104	1997	The preparation of safety publications and the use of basic safety publications and group safety publications	-	-
ISO/IEC Guide 51	1999	Safety aspects - Guidelines for their inclusion in standards	-	-

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

CONTENTS

INTRODUCTION 7
1 Scope 9
2 Normative references 12
3 Definitions and abbreviations 13
4 Conformance to this standard 13
5 Documentation 13
6 Additional requirements for management of safety-related software 13
6.1 Objectives 13
6.2 Requirements 13
7 Software safety lifecycle requirements 14
7.1 General 14
7.1.1 Objective 14
7.1.2 Requirements 14
7.2 Software safety requirements specification 21
7.2.1 Objectives 21
7.2.2 Requirements 21
7.3 Validation plan for software aspects of system safety 24
7.3.1 Objective 24
7.3.2 Requirements 24
7.4 Software design and development 25
7.4.1 Objectives 25
7.4.2 General requirements 26
7.4.3 Requirements for software architecture design 29
7.4.4 Requirements for support tools, including programming languages 30
7.4.5 Requirements for detailed design and development – software system design 33
7.4.6 Requirements for code implementation 34
7.4.7 Requirements for software module testing 35
7.4.8 Requirements for software integration testing 35
7.5 Programmable electronics integration (hardware and software) 36
7.5.1 Objectives 36
7.5.2 Requirements 36
7.6 Software operation and modification procedures 37
7.6.1 Objective 37
7.6.2 Requirements 37
7.7 Software aspects of system safety validation 37
7.7.1 Objective 37
7.7.2 Requirements 38
7.8 Software modification 39
7.8.1 Objective 39
7.8.2 Requirements 39
7.9 Software verification 41
7.9.1 Objective 41
7.9.2 Requirements 41
8 Functional safety assessment 41

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Annex A (normative) Guide to the selection of techniques and measures 46
Annex B (informative) Detailed tables 55
Annex C (informative) Properties for software systematic capability 60
Annex D (normative) Safety manual for compliant items – additional requirements for software elements 97
Annex E (informative) Relationships between IEC 61508-2 and IEC 61508-3 100
Annex F (informative) Techniques for achieving non-interference between software elements on a single computer 102
Annex G (informative) Guidance for tailoring lifecycles associated with data driven systems 107
Bibliography 111

Figure 1 – Overall framework of the IEC 61508 series 11
Figure 2 – Overall safety lifecycle 12
Figure 3 – E/E/PE system safety lifecycle (in realisation phase) 16
Figure 4 – Software safety lifecycle (in realisation phase) 16
Figure 5 – Relationship and scope for IEC 61508-2 and IEC 61508-3 17
Figure 6 – Software systematic capability and the development lifecycle (the V-model) 17
Figure G.1 – Variability in complexity of data driven systems 108

Table 1 – Software safety lifecycle – overview 18
Table A.1 – Software safety requirements specification 47
Table A.2 – Software design and development – software architecture design 48
Table A.3 – Software design and development – support tools and programming language 49
Table A.4 – Software design and development – detailed design 50
Table A.5 – Software design and development – software module testing and integration 51
Table A.6 – Programmable electronics integration (hardware and software) 51
Table A.7 – Software aspects of system safety validation 52
Table A.8 – Modification 52
Table A.9 – Software verification 53
Table A.10 – Functional safety assessment 54
Table B.1 – Design and coding standards 55
Table B.2 – Dynamic analysis and testing 56
Table B.3 – Functional and black-box testing 56
Table B.4 – Failure analysis 57
Table B.5 – Modelling 57
Table B.6 – Performance testing 58
Table B.7 – Semi-formal methods 58
Table B.8 – Static analysis 59
Table B.9 – Modular approach 59
Table C.1 – Properties for systematic safety integrity – Software safety requirements specification 64

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Table C.2 – Properties for systematic safety integrity – Software design and development – software Architecture Design 67
Table C.3 – Properties for systematic safety integrity – Software design and development – support tools and programming language 76
Table C.4 – Properties for systematic safety integrity – Software design and development – detailed design (includes software system design, software module design and coding) 77
Table C.5 – Properties for systematic safety integrity – Software design and development – software module testing and integration 79
Table C.6 – Properties for systematic safety integrity – Programmable electronics integration (hardware and software) 81
Table C.7 – Properties for systematic safety integrity – Software aspects of system safety validation 82
Table C.8 – Properties for systematic safety integrity – Software modification 83
Table C.9 – Properties for systematic safety integrity – Software verification 85
Table C.10 – Properties for systematic safety integrity – Functional safety assessment 86
Table C.11 – Detailed properties – Design and coding standards 87
Table C.12 – Detailed properties – Dynamic analysis and testing 89
Table C.13 – Detailed properties – Functional and black-box testing 90
Table C.14 – Detailed properties – Failure analysis 91
Table C.15 – Detailed properties – Modelling 92
Table C.16 – Detailed properties – Performance testing 93
Table C.17 – Detailed properties – Semi-formal methods 94
Table C.18 – Properties for systematic safety integrity – Static analysis 95
Table C.19 – Detailed properties – Modular approach 96
Table E.1 – Categories of IEC 61508-2 requirements 100
Table E.2 – Requirements of IEC 61508-2 for software and their typical relevance to certain types of software 100
Table F.1 – Module coupling – definition of terms 104
Table F.2 – Types of module coupling 105

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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 International Standard 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 (see references [1], [2] and [3]).

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 must therefore consider not only all the elements within an individual system (for example sensors, controlling devices and actuators) but also all the safety-related systems making up the total combination of safety-related systems. Therefore, while this International Standard is concerned with E/E/PE safety-related systems, it may also provide a framework within which safety-related systems based on other technologies may 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 International Standard, 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 International Standard

– 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 standard, 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 2 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.

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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

– sets a lower limit on the target failure measures for a safety function carried out by a single E/E/PE safety-related system. For E/E/PE safety-related systems operating in

  – a low demand mode of operation, the lower limit is set at an average probability of a dangerous failure on demand of 10^{-5};

  – a high demand or a continuous mode of operation, the lower limit is set at an average frequency of a dangerous failure of 10^{-9} [h^{-1}];

NOTE 3 A single E/E/PE safety-related system does not necessarily mean a single-channel architecture.

NOTE 4 It may be possible to achieve designs of safety-related systems with lower values for the target safety integrity for non-complex systems, but these limits are considered to represent what can be achieved for relatively complex systems (for example programmable electronic safety-related systems) at the present time.

– sets requirements for the avoidance and control of systematic faults, which are based on experience and judgement from practical experience gained in industry. Even though the probability of occurrence of systematic failures cannot in general be quantified the standard does, however, allow a claim to be made, for a specified safety function, that the target failure measure associated with the safety function can be considered to be achieved if all the requirements in the standard have been met;

– introduces systematic capability which applies to an element with respect to its confidence that the systematic safety integrity meets the requirements of the specified safety integrity level;

– adopts a broad range of principles, techniques and measures to achieve functional safety for E/E/PE safety-related systems, but does not explicitly use the concept of fail safe. However, the concepts of “fail safe” and “inherently safe” principles may be applicable and adoption of such concepts is acceptable providing the requirements of the relevant clauses in the standard are met.

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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 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 software 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 software systematic capability is an iterative procedure; see Figures 3 and 6.

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

NOTE 4 Note: See 3.5.9 of IEC 61508-4 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 4 Figure 5 shows the relationship between IEC 61508-2 and IEC 61508-3.

j) Does not apply for medical equipment in compliance with the IEC 60601 series.

1.2 IEC 61508-1, IEC 61598-2, IEC 61508-3 and IEC 61508-4 are basic safety publications, although this status does not apply in the context of low complexity E/E/PE safety-related systems (see 3.4.3 of IEC 61508-4). As basic safety publications, they are intended for use by technical committees in the preparation of standards in accordance with the principles contained in IEC Guide 104 and ISO/IEC Guide 51. IEC 61508-1, IEC 61508-2, IEC 61508-3 and IEC 61508-4 are also intended for use as stand-alone publications. The horizontal safety

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

function of this international standard does not apply to medical equipment in compliance with the IEC 60601 series.

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

1.4 Figure 1 shows the overall framework of the IEC 61508 series and indicates the role that IEC 61508-3 plays in the achievement of functional safety for E/E/PE safety-related systems.

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

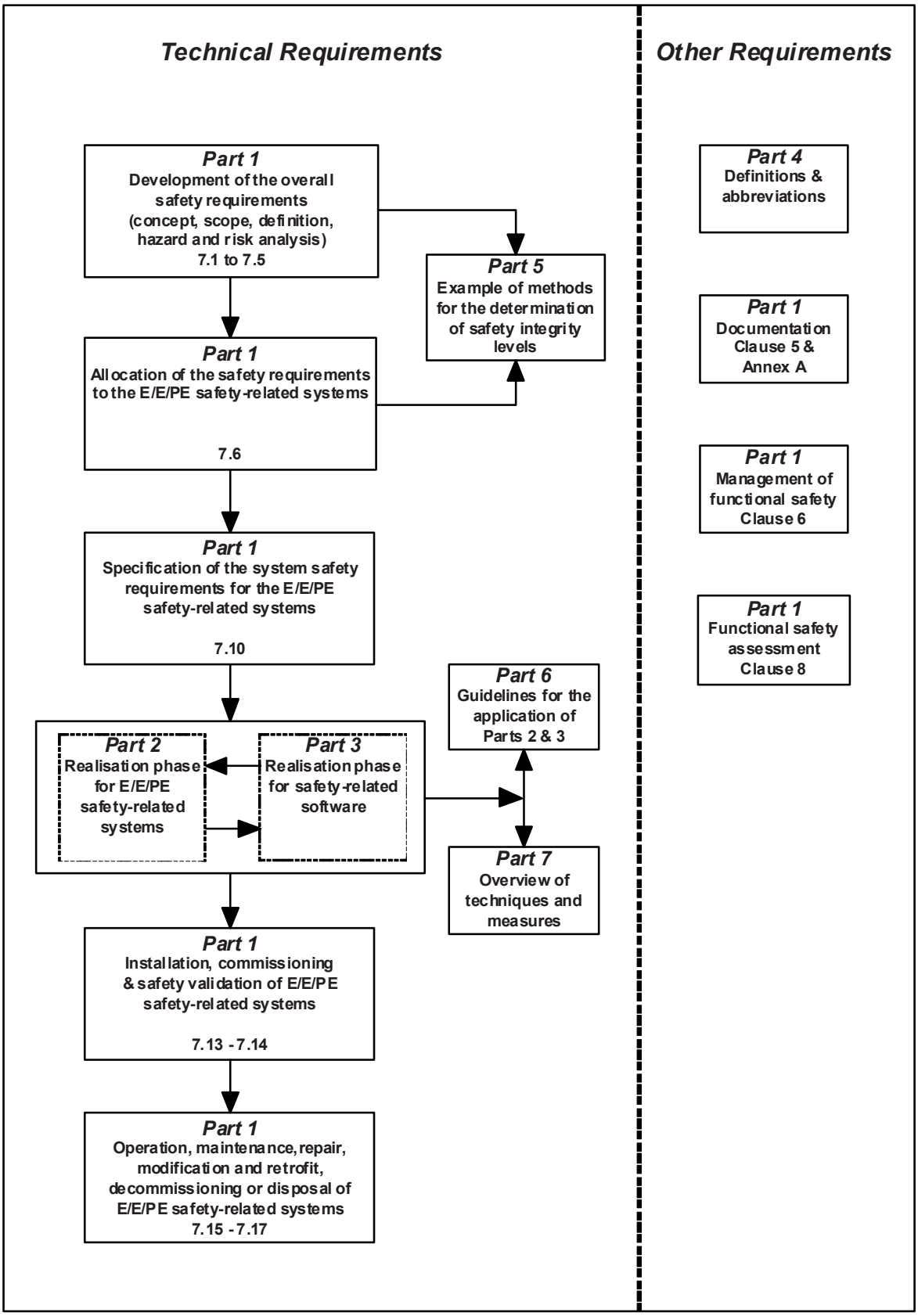

Figure 1 – Overall framework of the IEC 61508 series

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

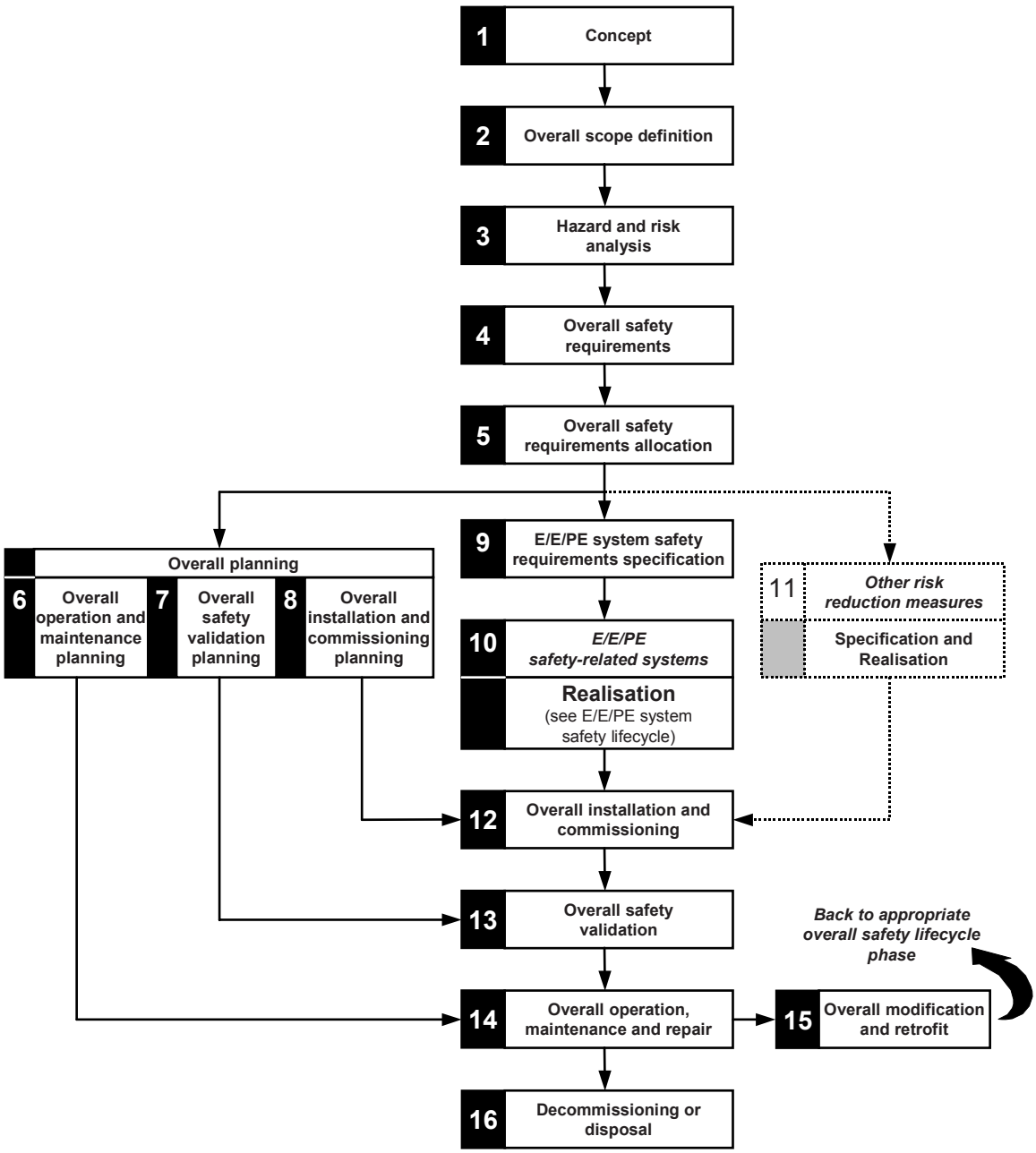

Figure 2 – Overall safety lifecycle

2 Normative references

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

IEC 61508-1: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 1: General requirements*

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

IEC 61508-2: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 2: Requirements for electrical/electronic/programmable electronic safety-related systems*

IEC 61508-4: 2010, *Functional safety of electrical/electronic/programmable electronic safety-related systems – Part 4: Definitions and abbreviations*

IEC Guide 104:1997, *The preparation of safety publications and the use of basic safety publications and group safety publications*

IEC/ISO Guide 51:1999, *Safety aspects – Guidelines for their inclusion in standards*

3 Definitions and abbreviations

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

4 Conformance to this standard

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

5 Documentation

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

6 Additional requirements for management of safety-related software

6.1 Objectives

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

6.2 Requirements

6.2.1 The requirements are as detailed in 6.2 of IEC 61508-1, with the following additional requirements.

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

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

6.2.3 Software configuration management shall:

a) apply administrative and technical controls throughout the software safety lifecycle, in order to manage software changes and thus ensure that the specified requirements for safety-related software continue to be satisfied;

b) guarantee that all necessary operations have been carried out to demonstrate that the required software systematic capability has been achieved;

c) maintain accurately and with unique identification all configuration items which are necessary to meet the safety integrity requirements of the E/E/PE safety-related system. Configuration items include at least the following: safety analysis and requirements; software specification and design documents; software source code modules; test plans

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

and results; verification documents; pre-existing software elements and packages which are to be incorporated into the E/E/PE safety-related system; all tools and development environments which are used to create or test, or carry out any action on, the software of the E/E/PE safety-related system;

d) apply change-control procedures:

- to prevent unauthorized modifications; to document modification requests;
- to analyse the impact of a proposed modification, and to approve or reject the request;
- to document the details of, and the authorisation for, all approved modifications;
- to establish configuration baseline at appropriate points in the software development, and to document the (partial) integration testing of the baseline;
- to guarantee the composition of, and the building of, all software baselines (including the rebuilding of earlier baselines).

NOTE 1 Management decision and authority is needed to guide and enforce the use of administrative and technical controls.

NOTE 2 At one extreme, an impact analysis may include an informal assessment. At the other extreme, an impact analysis may include a rigorous formal analysis of the potential adverse impact of all proposed changes which may be inadequately understood or implemented. See IEC 61508-7 for guidance on impact analysis.

e) ensure that appropriate methods are implemented to load valid software elements and data correctly into the run-time system;

NOTE 3 This may include consideration of specific target location systems as well as general systems. Software other than application might need a safe loading method, e.g. firmware.

f) document the following information to permit a subsequent functional safety audit: configuration status, release status, the justification (taking account of the impact analysis) for and approval of all modifications, and the details of the modification;

g) formally document the release of safety-related software. Master copies of the software and all associated documentation and version of data in service shall be kept to permit maintenance and modification throughout the operational lifetime of the released software.

NOTE 4 For further information on configuration management, see IEC 61508-7

7 Software safety lifecycle requirements

7.1 General

7.1.1 Objective

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

7.1.2 Requirements

7.1.2.1 A safety lifecycle for the development of software shall be selected and specified during safety planning in accordance with Clause 6 of IEC 61508-1.

7.1.2.2 Any software lifecycle model may be used provided all the objectives and requirements 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 Figures 3, 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) to take account of the safety integrity and the complexity of the project.

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

NOTE 1 A software safety lifecycle model which satisfies the requirements of this clause may be suitably 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 might be appropriate, for example, to merge the phases of software system design and architectural design.

NOTE 2 See Annex G for the characteristics of data-driven systems (e.g. full variability / limited variability programming languages, extent of data configuration) that may be relevant when customising the software safety lifecycle.

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. Annexes A and B provide a guide to the selection of techniques and measures, and references to IEC 61508-6 and IEC 61508-7. IEC 61508-6 and IEC 61508-7 give recommendations on specific techniques to achieve the properties required for systematic safety integrity. Selecting techniques from these recommendations does not guarantee by itself that the required safety integrity will be achieved.

NOTE Success in achieving systematic safety integrity depends on selecting techniques 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 Clause 5 of IEC 61508-1 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 may be a distinct document, while the documented outputs from several phases may be merged. 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 (1) which software modules are impacted, and (2) which earlier safety lifecycle activities shall be repeated.

NOTE At one extreme, an impact analysis may include an informal assessment. At the other extreme, an impact analysis may include a rigorous formal analysis of the potential adverse impact of all proposed changes which may be inadequately understood or implemented. See IEC 61508-7 for guidance on impact analysis.

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

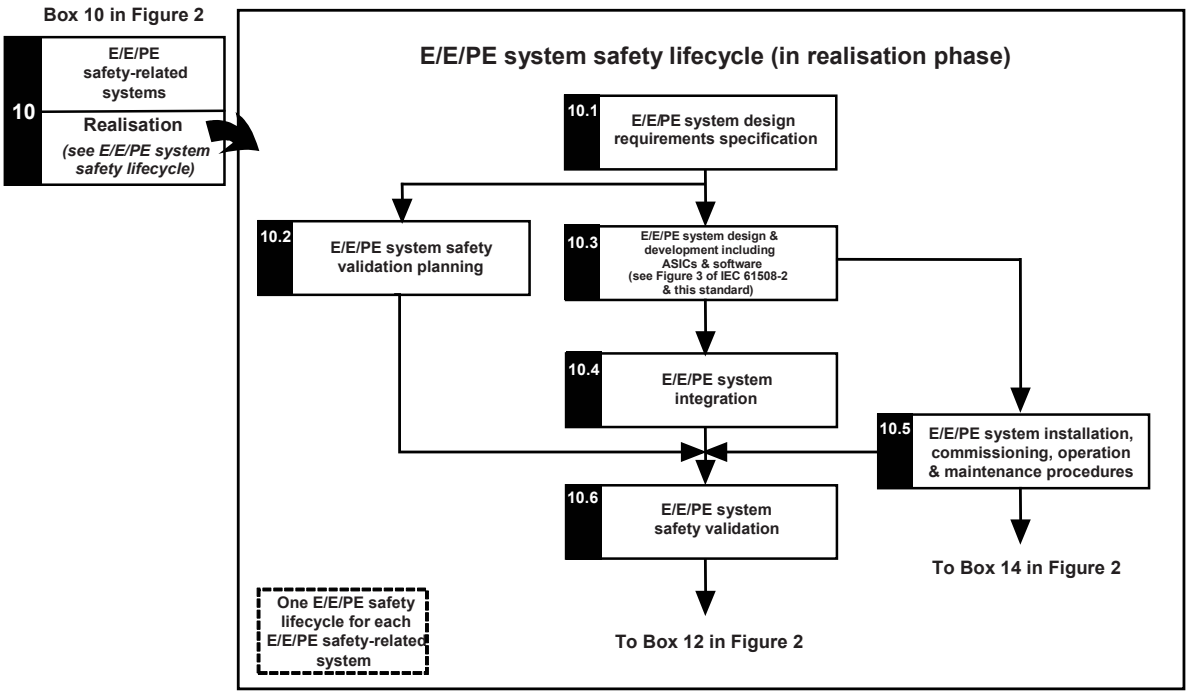

Figure 3 – E/E/PE system safety lifecycle (in realisation phase)

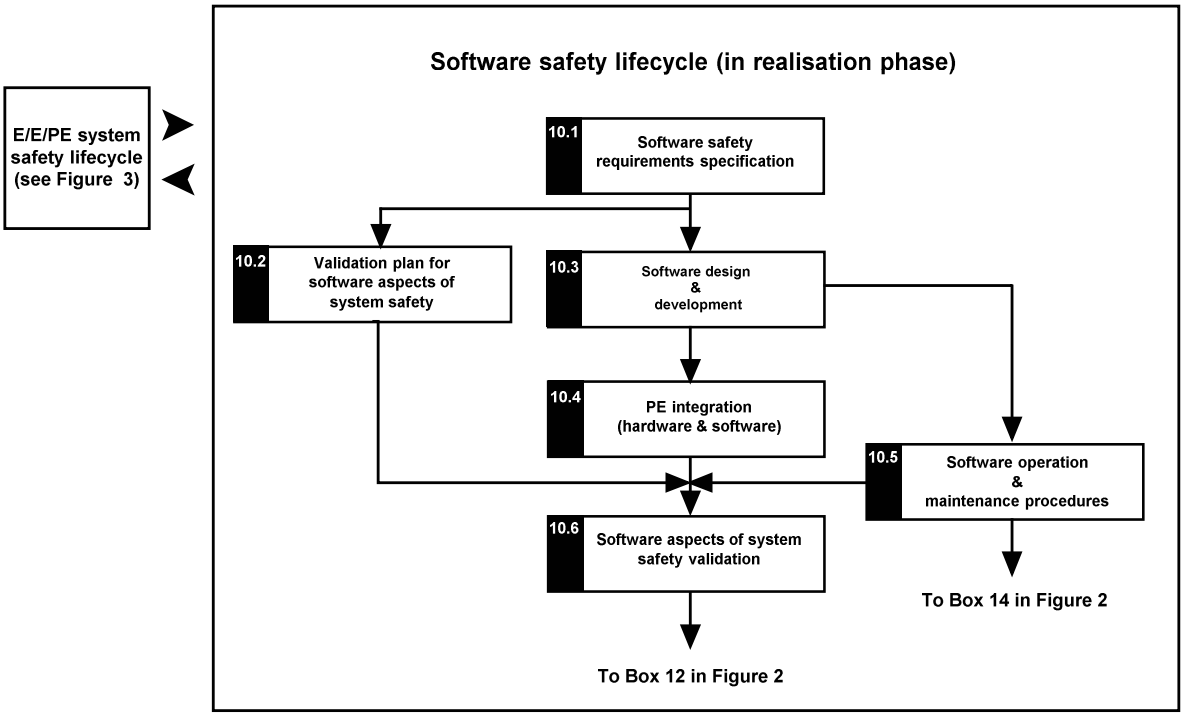

Figure 4 – Software safety lifecycle (in realisation phase)

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

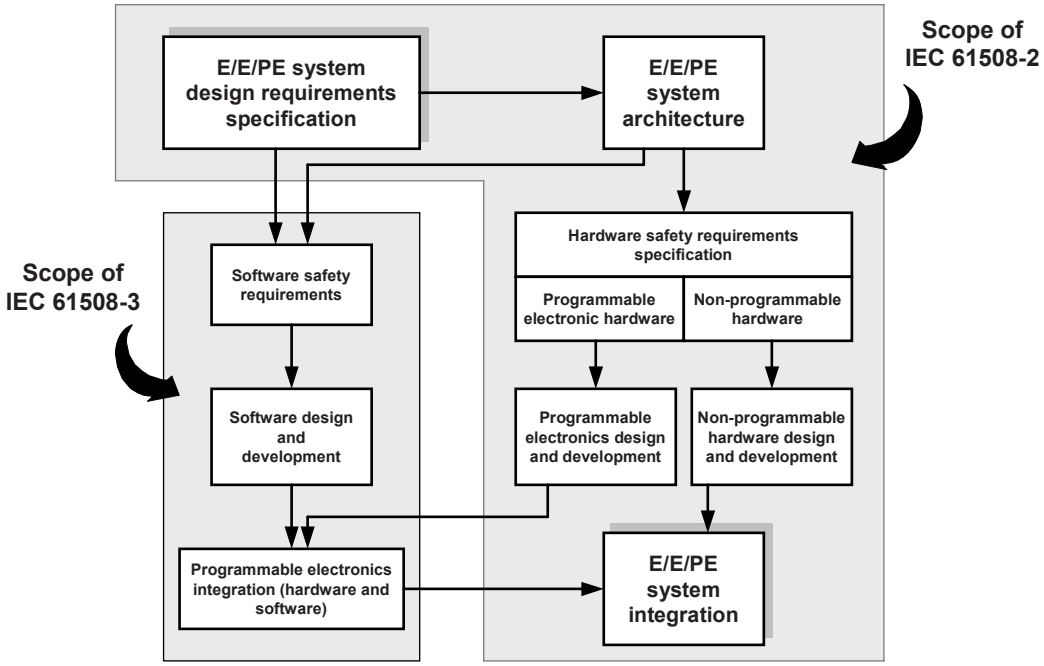

Figure 5 – Relationship and scope for IEC 61508-2 and IEC 61508-3

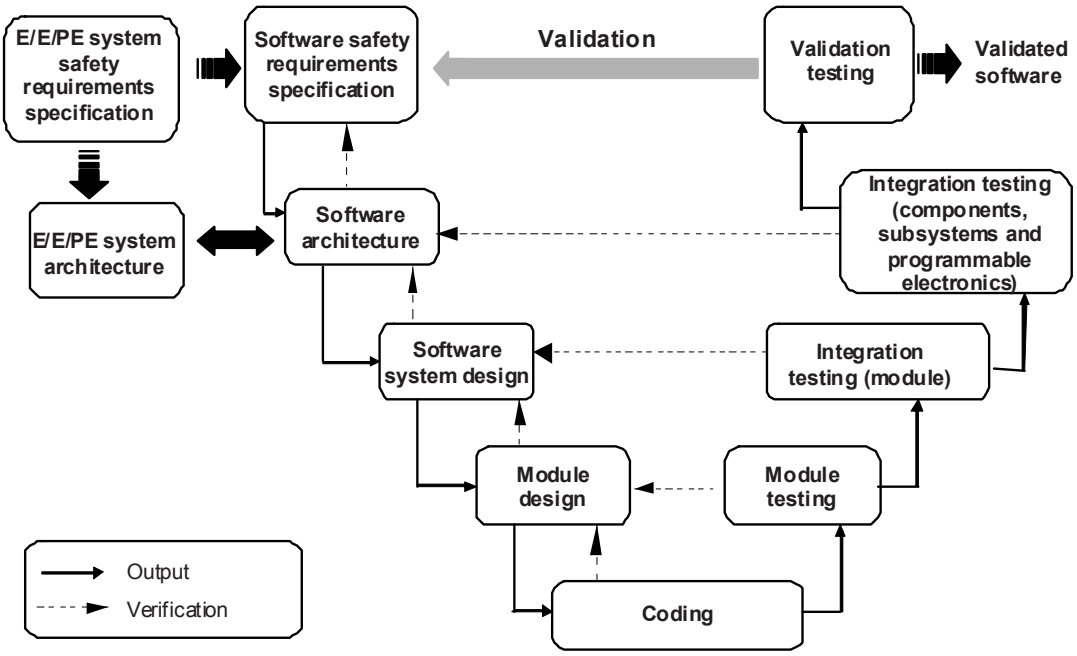

Figure 6 – Software systematic capability and the development lifecycle (the V-model)

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Table 1 – Software safety lifecycle – overview

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Title					
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 software 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 software 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)	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 safety integrity level; 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)	software architecture design; software architecture integration test specification; software/ PE integration test specification (also required by IEC 61508-2)
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, user interfaces, and data formats and representations for the required safety integrity level, 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

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Table 1 *(continued)*

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Title					
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 safety integrity level, 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.	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 safety integrity level, 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 safety integrity level, 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 software 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 software 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

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Table 1 *(continued)*

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Title					
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 software 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 software 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 safety integrity level	program-mable 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	software architecture integration test results; programmabl e electronics integration test results; verified and tested integrated programmabl e 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
10.6	Software aspects of system safety validation	To ensure that the integrated system complies with the specified requirements for safety-related software at the intended safety integrity level	as above	7.7.2	validation plan for software aspects of system safety	software safety validation results; validated software
–	Software modification	To guide corrections, enhancements or adaptations to the validated software, ensuring that the required software systematic capability is sustained	as above	7.8.2	software modification procedures; software modification request	software modification impact analysis results; software modification log

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

Table 1 *(continued)*

Safety lifecycle phase		Objectives	Scope	Require-ments subclause	Inputs (information required)	Outputs (information produced)
Figure 4 box number	Title					
–	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

7.2 Software safety requirements specification

NOTE This phase is Box 10.1 of Figure 4.

7.2.1 Objectives

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

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

7.2.1.3 The third objective of the requirements of this subclause is to specify the requirements for software 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.

7.2.2 Requirements

NOTE 1 These requirements will in most cases be achieved by a combination of generic embedded software and application specific software. It is the combination of both that provides the features that satisfy the following subclauses. The exact division between generic and application specific software depends on the chosen software architecture (see 7.4.3).

NOTE 2 For the selection of appropriate techniques and measures (see Annexes A and B) to implement the requirements of this clause, the following properties (see Annex C for guidance on interpretation of properties, and Annex F of IEC 61508-7 for informal definitions) of the software safety requirements specification should be considered:

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

NOTE 3 The safety needs to be addressed by software is 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 needs is a larger set that includes also 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.

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

7.2.2.1 If the requirements for safety-related software have already been specified for the E/E/PE safety-related system (see Clause 7 of IEC 61508-2), then the specification of software safety requirements need not be repeated.

7.2.2.2 The specification of the requirements for safety-related software shall be derived from the specified safety requirements of the E/E/PE safety-related system (see IEC 61508-2, 7), 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 and the developer of the software (IEC 61508-2 and IEC 61508-3). As the safety-related software requirements and the software architecture become more precise, there may 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 may have been 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 safety integrity (including any requirement for independence, see 7.4.3 of IEC 61508-2), and to allow an assessment of functional safety to be carried out.

NOTE The level of detail of the specification may vary with the complexity of the application. An adequate specification of functional behaviour may 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 See Annex F for techniques for achieving one aspect of independence of software.

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 safety integrity requirements (programmable electronics, sensors, and actuators);

d) software systematic capability requirements;

e) capacity and response time;

f) equipment and operator interfaces, including reasonably foreseeable misuse.

NOTE Compatibility with any applications already in existence should be considered.

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.

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 considering the possible increase in complexity, the software safety requirements specification shall consider the following:

a) software self-monitoring (for examples see IEC 61508-7);

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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 Increased complexity resulting from the above considerations may 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 of 61508-1). 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:

   1) functions that enable the EUC to achieve or maintain a safe state;

   2) functions related to the detection, annunciation and management of faults in the programmable electronics hardware;

   3) functions related to the detection, annunciation and management of sensor and actuators faults;

   4) functions related to the detection, annunciation and management of faults in the software itself (software self-monitoring);

   5) functions related to the periodic testing of safety functions on-line (i.e. in the intended operational environment);

   6) 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);

   7) functions that allow the PE system to be safely modified;

   8) interfaces to non safety-related functions;

   9) capacity and response time performance;

   10) interfaces between the software and the PE system;

   NOTE 1 They include both off-line and on-line programming facilities.

   11) safety-related communications (see 7.4.11 of IEC 61508-2).

b) the requirements for the software systematic capability:

   1) the safety integrity level(s) for each of the functions in a) above;

   NOTE 2 See Annex A of IEC 61508-5 for information concerning the allocation of safety integrity to software elements.

   2) independence requirements between functions.

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.).

NOTE 1 This requirement on application data is particularly relevant to data-driven applications. These are characterized as follows: the source code is pre-existing and the primary objective of the development activity is to

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

provide assurance that the configuration data correctly states the behaviour required from the application. There may be complex dependencies between data items, and the validity of data may change over time.

NOTE 2 See Annex G 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 7.4.11 of IEC 61508-2:

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.

NOTE 1 Protection against unauthorized changes should be considered, taking account of both software-based and non-software mechanisms. Note that effective protection against unauthorized software changes can have adverse effects on safety e.g. when changes are needed rapidly and in stressful conditions.

NOTE 2 Although a person can form part of a safety-related system (see Clause 1 of IEC 61508-1), human factor requirements related to the design of E/E/PE safety-related systems are not considered in detail in this standard. However, the following human considerations should be addressed where appropriate:

- An operator information system should use the pictorial layout and the terminology the operators are familiar with. It should be clear, understandable and free from unnecessary details and/or aspects;
- Information about the EUC displayed to the operator should follow closely the physical arrangement of the EUC;
- If several display contents to the operator are feasible and/or if the possible operator actions allow interactions whose consequences cannot be seen at one glance, the information displayed should automatically contain at each state of a display or an action sequence, which state of the sequence is reached, which operations are feasible and which possible consequences can be chosen.

7.3 Validation plan for 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.

7.3.1 Objective

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

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;

AVA Documents
Document Sharing Hub

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 technical strategy for the validation (for example analytical methods, statistical tests etc.);

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 software systematic capability;

g) the required environment in which the validation activities are to take place (for example, for tests this could include calibrated tools and equipment);

h) the pass/fail criteria;

i) the policies and procedures for evaluating the results of the validation, particularly failures.

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

7.3.2.3 The validation shall give a rationale for the chosen strategy. The technical strategy for the validation of safety-related software shall include the following information:

a) choice of manual or automated techniques or both;

b) choice of static or dynamic techniques or both;

c) choice of analytical or statistical techniques or both.

d) 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 (see Clause 8 of IEC 61508-1). 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; and

c) other acceptance criteria, for example memory usage, timing and value tolerances.

7.4 Software design and development

NOTE This phase is box 10.3 of Figure 4.

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.

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, user interfaces, and data formats and representations for the required safety integrity level, over the whole safety lifecycle of the software which assists verification, validation, assessment and modification.

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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, which is analysable and verifiable, and which is capable of being safely modified.

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 software 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 software systematic capability.

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 the supplier of a safety related programming environment (e.g. PLC supplier) alone, or with the user of that environment (e.g. the application software developer) alone, or with both. The division of responsibility shall be determined during safety planning (see Clause 6).

NOTE 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 safety integrity level 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.

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, information hiding and encapsulation.

NOTE See F.7.

Documents
Document Sharing Hub

Full document details: BS EN 61508-3:2010

7.4.2.5 The design representations shall be based on a notation which is unambiguously defined or restricted to unambiguously defined features.

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 safety integrity level, self-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 safety integrity levels, then all of the software shall be treated as belonging to the highest safety integrity level, unless adequate independence between the safety functions of the different safety integrity levels can be shown in the design. It shall be demonstrated either (1) that independence is achieved by both in 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 safety integrity level of the 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 equals the safety integrity level of the safety function.

7.4.2.11 Where a safety function is implemented using a combination of software elements of known systematic capability, the systematic capability requirements of 7.4.3 of IEC 61508-2, shall apply to the combination of elements.

NOTE Distinguish consistently between (1) the *end-to-end safety function* that is supported by one or more elements and (2) the *element safety function* of each of the supporting elements. Where two elements combine to achieve a higher systematic capability in combination, each of the paired elements should be capable of preventing/mitigating the hazardous event, but the paired elements are not required to have identical element safety functions, and it is not required that each of the paired elements is independently capable of providing the whole safety functionality demanded from the combination.

EXAMPLE An electronic engine throttle control where the *end-to-end safety function* is “prevent undemanded acceleration”. The *end-to-end safety function* is implemented by two processors. The *element safety function* of the primary controller is the ideal demand/response behaviour of the throttle. The *element safety function* of the secondary processor is a diverse monitor (see IEC 61508-7 C.3.4) and applies an emergency stop if necessary. The combination of the two processors gives higher confidence that the end-to-end safety function “prevent undemanded acceleration” will be achieved.

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 systematic safety integrity:

a) meet the requirements of one of the following compliance routes:
   - Route 1_S: compliant development. Compliance with the requirements of this standard for the avoidance and control of systematic faults in software;
   - Route 2_S: proven in use. Provide evidence that the element is proven in use. See 7.4.10 of IEC 61508-2;
   - Route 3_S:assessment of non-compliant development. Compliance with 7.4.2.13.

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

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

NOTE 2 See 3.2.8 of IEC 61508-4. The pre-existing software could be a commercially available product, or it could have been developed by some organisation for a previous product or system. Pre-existing software may or may not have been developed in accordance with the requirements of this standard.

NOTE 3 Requirements on pre-existing elements apply to a run-time library or an interpreter.

b) provide a safety manual (see Annex D of IEC 61508-2 and Annex D of this standard) 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 may be derived from the element supplier's own documentation and records of the element supplier's development process, or may 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 may be required to create specification or 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 element may be developed during safety planning (see Clause 6).

7.4.2.13 To comply with Route 3_S a pre-existing software element shall meet all of the following requirements a) to i):

a) The software safety requirements specification for the element in its new application shall be documented to the same degree of precision as would be required by this standard for any safety related element of the same systematic capability. The software safety requirements specification shall cover the functional and safety behaviour as applicable to the element in its new application and as specified in 7.2. See Table A.1.

b) The justification for use of a software element shall provide evidence that the desirable safety properties specified in the referenced subclauses (i.e. 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, 7.9.2, and Clause 8) have been considered, taking account of the guidance in Annex C.

c) The element's design shall be documented to a degree of precision, sufficient to provide evidence of compliance with the requirement specification and the required systematic capability. See 7.4.3, 7.4.5 and 7.4.6, and Tables A.2 and A.4 of Annex A.

d) The evidence required in 7.4.2.13 a) and 7.4.2.13 b) shall cover the software's integration with the hardware. See 7.5 and Table A.6 of Annex A.

e) There shall be evidence that the element has been subject to verification and validation using a systematic approach with documented testing and 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 and Tables A.5 to A.7 and A.9 of Annex A as well as related tables in Annex B.

NOTE 1 Positive operational experience may be used to satisfy black-box and probabilistic testing requirements [see Tables A.7 and B.3].

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

NOTE 2 Ways to meet this requirement include:

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

g) There shall be evidence that all credible failure mechanisms of the software element have been identified and that appropriate mitigation measures have been implemented.

NOTE 3 Appropriate mitigation measures include:

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

h) The planning for use of the element shall identify the configuration of the software element, the software and hardware run-time environment and if necessary the configuration of the compilation / linking system.

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

i) The justification for use of the element shall be valid for only those applications which respect the assumptions made in the compliant item safety manual for the element (see Annex D of IEC 61508-2 and Annex D).

7.4.2.14 This Subclause 7.4.2 shall, in so far as it is appropriate, apply to data and data generation languages.

NOTE See Annex G for guidance on data-driven systems.

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

b) Where the safety-related functionality of a PE system is determined significantly or predominantly by configuration data, appropriate techniques and measures shall be used to prevent the introduction of faults during the design, production, loading and modification of the configuration data and to ensure that the configuration data correctly states the application logic.

c) The specification of data structures shall be:

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

   2) complete;

   3) self consistent;

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

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

7.4.3 Requirements for software architecture design

NOTE 1 The software architecture defines the major elements and subsystems of the software, how they are interconnected, and how the required attributes, particularly safety integrity, will be achieved. It also defines the overall behaviour of the software, and how software elements interface and interact. Examples of major software elements include operating systems, databases, EUC input/output subsystems, communication subsystems, application program(s), programming and diagnostic tools, etc.

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

NOTE 3 In some contexts of user application programming, particularly in PLCs (see Annex E of IEC 61508-6), the software architecture is provided by the supplier as a standard feature of the product. The supplier would, under this standard, be required to assure the user of the compliance of his products to the requirements of 7.4. The user tailors the PLC to the application by using the standard programming facilities, for example ladder logic. The requirements of 7.4.3 to 7.4.8 still apply. The requirement to define and document the software architecture can be seen as information that the user would use to select the PLC (or equivalent) for the application.

NOTE 4 From a safety viewpoint, the software architecture phase is where the basic safety strategy is developed for the software.

NOTE 5 Although the IEC 61508 series sets numerical target failure measures for safety functions carried out by E/E/PE safety-related systems, systematic safety integrity is usually unquantified (see 3.5.6 of IEC 61508-4), and software safety integrity (see 3.5.5 of IEC 61508-4) is defined as a systematic capability on a confidence scale of 1-4 (see 3.5.9 of IEC 61508-4). This standard 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 standard applies development and validation techniques with rigour that is qualitatively consistent with the required systematic capability.

NOTE 6 For the selection of appropriate techniques and measures (see Annexes A and B) to implement the requirements of this clause, the following properties (see Annex C for guidance on interpretation of properties, and Annex F of IEC 61508-7 for informal definitions) of the software architecture design should be considered:

– completeness with respect to software safety requirements specification;

– correctness with respect to software safety requirements specification;

– freedom from intrinsic design faults;

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

– simplicity and understandability;
– predictability of behaviour;
– verifiable and testable design;
– fault tolerance;
– defence against common cause failure from external events.

7.4.3.1 Depending on the nature of the software development, responsibility for conformance with 7.4.4 can rest with multiple parties. The division of responsibility shall be documented during safety planning (see Clause 6 of IEC 61508-1).

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

a) select and justify (see 7.1.2.7) an integrated set of techniques and measures necessary during the software safety lifecycle phases to satisfy the software safety requirements specification at the required safety integrity level. 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;
b) 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) software systematic capability of the subsystem/element.
c) determine all software/hardware interactions and evaluate and detail their significance;

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

d) use a notation to represent the architecture which is unambiguously defined or restricted to unambiguously defined features;
e) select the design features to be used for maintaining the safety integrity of all data. Such data may include plant input-output data, communications data, operator interface data, maintenance data and internal database data;
f) specify appropriate software architecture integration tests to ensure that the software architecture satisfies the software safety requirements specification at the required safety integrity level.

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

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

7.4.4 Requirements for support tools, including programming languages

NOTE For the selection of appropriate techniques and measures (see Annexes A and B) to implement the requirements of this clause, the following properties (see Annex C for guidance on interpretation of properties, and Annex F of IEC 61508-7 for informal definitions) of support tools should be considered:

– the degree to which the tool supports the production of software with the required software properties;
– the clarity of the operation and functionality of the tool;
– the correctness and repeatability of the output.

7.4.4.1 A software on-line support tool shall be considered to be a software element of the safety-related system

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

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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

NOTE 1 See 7.1.2 for software development lifecycle requirements.

NOTE 2 Appropriate off-line tools to support the development of software should be used in order to increase the integrity of the software by reducing the likelihood of introducing or not detecting faults during the development. Examples of tools relevant to the phases of the software development lifecycle include:

a) transformation or translation tools that convert a software or design representation (e.g. text or a diagram) from one abstraction level to another level: design refinement tools, compilers, assemblers, linkers, binders, loaders and code generation tools;

b) verification and validation tools such as static code analysers, test coverage monitors, theorem proving assistants, and simulators;

c) diagnostic tools used to maintain and monitor the software under operating conditions;

d) infrastructure tools such as development support systems;

e) configuration control tools such as version control tools;

f) application data tools that produce or maintain data which are required to define parameters and to instantiate system functions. Such data includes function parameters, instrument ranges, alarm and trip levels, output states to be adopted at failure, geographical layout.

NOTE 3 Off-line support tools should be selected to be integrated. In this context, tools are integrated if they work co-operatively such that the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimising the possibility of introducing human error in the reworking of intermediate results.

NOTE 4 Off-line support tools should be selected to be compatible with the needs of the application, of the safety related system, and of the integrated toolset.

NOTE 5 The availability of suitable tools to supply the services that are necessary over the whole lifetime of the E/E/PE safety-related system (e.g. tools to support specification, design, implementation, documentation, modification) should be considered.

NOTE 6 Consideration should be given to the competence of the users of the selected tools. See Clause 6 of IEC 61508-1 for competence requirements.

7.4.4.3 The selection of the off-line support tools shall be justified.

7.4.4.4 All off-line support tools in classes T2 and T3 shall have a specification or product documentation which clearly defines the behaviour of the tool and any instructions or constraints on its use. See 7.1.2 for software development lifecycle requirements, and 3.2.11 of IEC 61508-4 for categories of software off-line support tool.

NOTE This “specification or product documentation” is not a safety manual for compliant items (see Annex D of 61508-2 and also of this standard) for the tool itself. The safety manual for compliant item relates to a pre-existing element that is incorporated into the executable safety related system. Where a pre-existing element has been generated by a T3 tool and then incorporated into the executable safety related system, then any relevant information (e.g. the documentation for an optimising compiler may indicate that the evaluation order of function parameters is not guaranteed) from the tool’s “specification or product documentation” should be included in the compliant item safety manual that makes possible an assessment of the integrity of a specific safety function that depends wholly or partly on the incorporated element.”

7.4.4.5 An assessment shall be carried out for offline support tools in classes T2 and T3 to determine the level of reliance placed on the tools, and the potential failure mechanisms of the tools that may affect the executable software. Where such failure mechanisms are identified, appropriate mitigation measures shall be taken.

NOTE 1 Software HAZOP is one technique to analyse the consequences of potential software tool failures.

NOTE 2 Examples of mitigation measures include: avoiding known bugs, restricted use of the tool functionality, checking the tool output, use of diverse tools for the same purpose.

7.4.4.6 For each tool in class T3, evidence shall be available that the tool conforms to its specification or documentation. Evidence may be based on a suitable combination of history of successful use in similar environments and for similar applications (within the organisation or other organisations), and of tool validation as specified in 7.4.4.7.

NOTE 1 A version history may provide assurance of maturity of the tool, and a record of the errors / ambiguities that should be taken into account when the tool is used in the new development environment.

AVA Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

NOTE 2 The evidence listed for T3 may also be used for T2 tools in judging the correctness of their results.

7.4.4.7 The results of tool validation shall be documented covering the following results:

a) a chronological record of the validation activities;
b) the version of the tool product manual being used;
c) the tool functions being validated;
d) tools and equipment used;
e) the results of the validation activity; the documented results of validation shall state either that the software has passed the validation or the reasons for its failure;
f) test cases and their results for subsequent analysis;
g) discrepancies between expected and actual results.

7.4.4.8 Where the conformance evidence of 7.4.4.6 is unavailable, there shall be effective measures to control failures of the executable safety related system that result from faults that are attributable to the tool.

NOTE An example of a measure would be the generation of diverse redundant code which allows the detection and control of failures of the executable safety related system as a result of faults that have been introduced into the executable safety related system by a translator.

7.4.4.9 The compatibility of the tools of an integrated toolset shall be verified.

Note: tools are integrated if they work co-operatively such that the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results. See IEC 61508-7 B.3.5.

7.4.4.10 To the extent required by the safety integrity level, the software or design representation (including a programming language) selected shall:

a) have a translator which has been assessed for fitness for purpose including, where appropriate, assessment against the international or national standards;
b) use only defined language features;
c) match the characteristics of the application;
d) contain features that facilitate the detection of design or programming mistakes;
e) support features that match the design method.

NOTE 1 A programming language is a class of software or design representations. A translator converts a software or design representation (e.g. text or a diagram) from one abstraction level to another level. Examples of translators include: design refinement tools, compilers, assemblers, linkers, binders, loaders and code generation tools.

NOTE 2 The assessment of a translator may be performed for a specific application project, or for a class of applications. In the latter case all necessary information on the tool (the “specification or product manual”, see 7.4.4.4) regarding the intended and appropriate use of the tool should be available to the user of the tool. The assessment of the tool for a specific project may then be reduced to checking general suitability of the tool for the project and compliance with the “specification or product manual” (i.e. proper use of the tool). Proper use might include additional verification activities within the specific project.

NOTE 3 A validation suite (i.e. a set of test programs whose correct translation is known in advance) may be used to evaluate the fitness for purpose of a translator according to defined criteria, which should include functional and non-functional requirements. For the functional translator requirements, dynamic testing may be a main validation technique. If possible an automatic testing suite should be used.

7.4.4.11 Where 7.4.4.10 cannot be fully satisfied, the fitness for purpose of the language, and any additional measures which address any identified shortcomings of the language shall be justified.

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

Documents
Document Sharing Hub
Full document details: BS EN 61508-3:2010

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

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

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

b) description;

c) inputs and outputs;

d) configuration management history.

7.4.4.14 Where automatic code generation or similar automatic translation takes place, the suitability of the automatic translator for safety-related system development shall be assessed at the point in the development lifecycle where development support tools are selected.

7.4.4.15 Where off-line support tools of classes T2 and T3 generate items in the configuration baseline, configuration management shall ensure that information on the tools is recorded in the configuration baseline. This includes in particular:

a) the identification of the tool and its version;

b) the identification of the configuration baseline items for which the tool version has been used;

c) the way the tool was used (including the tool parameters, options and scripts selected) for each configuration baseline item.

NOTE The objective of this clause is to allow the baseline to be reconstructed.

7.4.4.16 Configuration management shall ensure that for tools in classes T2 and T3, only qualified versions are used.

7.4.4.17 Configuration management shall ensure that only tools compatible with each other and with the safety-related system are used.

NOTE The safety-related system hardware may also impose compatibility constraints on software tools e.g. a processor emulator needs to be an accurate model of the real processor electronics.

7.4.4.18 Each new version of off-line support tool shall be qualified. This qualification may rely on evidence provided for an earlier version if sufficient evidence is provided that:

a) the functional differences (if any) will not affect tool compatibility with the rest of the toolset; and

b) the new version is unlikely to contain significant new, unknown faults.

NOTE Evidence that the new version is unlikely to contain significant new, unknown faults may be based on (1) a clear identification of the changes made, (2) an analysis of the verification and validation actions performed on the new version, and (3) any existing operational experience from other users that is relevant to the new version.

7.4.4.19 Depending on the nature of the software development, responsibility for conformance with 7.4.4 can rest with multiple parties. The division of responsibility shall be documented during safety planning (see Clause 6 of IEC 61508-1).

7.4.5 Requirements for detailed design and development – software system design

NOTE 1 Detailed design is defined here to mean software system design: the partitioning of the major elements in the architecture into a system of software modules; individual software module design; and coding. In small applications, software system design and architectural design may be combined.

NOTE 2 The nature of detailed design and development will vary with the nature of the software development activities and the software architecture (see 7.4.3). In some contexts of application programming, for example ladder logic and function blocks, detailed design can be considered as configuring rather than programming.

AVA Documents
Document Sharing Hub